INT 21H -> CALL

tablica wektorów przerwań -> nr funkcji jest traktowany jako indeks w tablicy adresowej, której elementami są adresy funkcji systemowych (tryb rzeczywisty) dla trybu chronionego tablica deskryptorów przerwań

Każdy adres (wektor) zajmuje 4
bajty i zawiera adres w postaci
segment:offset

Tablica wektorów przerwań zawiera 256 adresów 4-
bajtowych i umieszczona jest w pamięci operacyjnej
począwszy od adresu fizycznego 0.

Adresy podprogramów w tej tablicy zapisywane są w
układzie segment:offset; oba pola są 16-bitowe, a
pole offset zajmuje dwa bajty o niższych adresach.

 Adres fizyczny w tym przypadku określa formuła
segment*16 + offset

Rozkaz INT należy klasyfikować jako rozkaz skoku
do podprogramu typu pośredniego, podobny do
rozkazu CALL.

Przykładowo, rozkaz INT 21H powoduje skok do
podprogramu, którego adres zawarty jest w wektorze
o numerze 33 (=21H).

Rozkaz INT pozostawia ślad na stosie zawierający
rejestry (E)IP, CS i rejestr znaczników (E)FLAGS;
następnie wykonuje do podprogramu, którego adres
określony jest przez zawartość wektora przerwania o
numerze określonym przez argument rozkazu INT.

IRET -> używany gdy wywołano podprogram za pomocą INT albo wskutek wystąpienia przerwania
sprzętowego lub wyjątku procesora

przed wywołaniem funkcji INT 21H nalezy wpisać odpowiedni nr katalogowy do rejestru AH

funkcja katalogowa 1 odczytuje znak z klawiatury
funckja katalogowa 2 wypisuje na ekran

w linuxie funkcje systemowe wywołuje się za pomocą INT 80H, nr funkcji sys wpisuje sie do EAX

Przykładowo, wywołanie INT 80H przy EAX = 3

powoduje odczytanie zawartości pliku, przy czym wcześniej do rejestru EBX należy wpisać deskryptor (dojście) pliku, do ECX adres obszaru pamięci, do którego zostanie wpisana zawartość pliku, a do rejestru EDX liczbę bajtów do przeczytania.

BIOS zapisany jest w pamięci ROM i w przestrzeni adresowej 1 MB zajmuje obszar pamięci począwszy od adresu fizycznego F0000H (lub F000H:0000H w formacie segment:offset).

Operand rozkazu INT (np. INT 10H – operacje sterownika graficznego) stanowi indeks do tablicy wektorów przerwań, w której podane są adresy podprogramów realizujących funkcje systemowe.(BIOS)

 Następcą systemu BIOS w komputerach jest stopniowo rozwijany interfejs UEFI (ang. Unified Extensible Firmware Interface).
 UEFI rozszerza możliwości konfiguracji sprzętu, a w przyszłości ma oferować wiele nowych usług, np. możliwość komunikacji przez internet w przypadku uszkodzenia głównego systemu operacyjnego. Niektóre planowane jego właściwości wzbudzają pewne kontrowersje, np. ograniczenia dostępu do nośników multimedialnych.

Tak więc algorytmy dodawania liczb kodowanych w systemie U2 są takie same jak dla liczb bez znaku (tj. liczb kodowanych w naturalnym kodzie binarnym).
 Z tego powodu rozkaz dodawania ADD może służyć zarówno do dodawania liczb bez znaku, jak i do dodawania liczb ze znakiem w kodzie U2; również rozkaz odejmowania SUB może wykonywać działania na obu typach liczb.

CF (ang. carry flag) – dla przypadku, gdy dodawano liczby bez znaku;
•OF (ang. overflow flag) – dla przypadku, gdy dodawano liczby w kodzie U2.

Procesor ustawia znacznik CF na podstawie wartości przeniesienia, które powstaje przy dodawaniu najstarszych bitów obu liczb.
 Procesor ustawia znacznik OF na podstawie wartości wyrażenia pn pn–1 (xor)
, gdzie pn–1
i pn
 oznaczają
przeniesienia występujące podczas dodawania dwóch najbardziej znaczących bitów.

Zazwyczaj procesor wykonuje operację odejmowania poprzez zmianę znaku odjemnika, a następnie wykonuje dodawanie:
 Jeden ze sposobów zmiany znaku liczby w kodzie U2 polega na zanegowaniu wszystkich bitów, a następnie dodaniu 1 do otrzymanej wartości.

Operacja odejmowania liczb ze znakiem (w kodzie U2) i liczb bez znaku, podobnie jak w przypadku dodawania, wykonywana jest przez ten sam rozkaz SUB – taką samą rolę jak przy dodawaniu pełnią znaczniki CF i OF, przy czym w operacji odejmowania znacznik CF reprezentuje pożyczkę.

neg ecx -> liczba przeciwna (odejmuje od 0)

adc -> dodaje wartosc CF do wyniku
mul -> mnozenie liczb bez znaku
imul -> mnozenie liczb ze znakiem

div idiv analogicznie jw

BT bit nie ulega zmianie
BTS wpisanie 1 do bitu
BTR wpisanie 0 do bitu
BTC zanegowanie zawartości bitu

przy czym
przed wykonaniem operacji zawartość bitu jest kopiowana do znacznika CF

Każdy ww. rozkaz ma dwa operandy:
pierwszy operand określa rejestr lub komórkę

pamięci zawierającą modyfikowany bit;
drugi operand, w postaci liczby lub rejestru,
wskazuje numer bitu, na którym ma być
wykonana operacja.
 Przykład: rozkaz btc edi,29 powoduje
zanegowanie bitu nr 29 w rejestrze EDI.
 Uwaga: operandami omawianej grupy rozkazów nie
mogą być rejestry 8-bitowe.

Do testowania zawartości jednego lub kilku bitów
stosowany jest rozkaz TEST, który wyznacza iloczyn
logiczny, ale nigdzie nie wpisuje uzyskanego wyniku
– ustawia natomiast znaczniki (rozkaz TEST działa
analogicznie do rozkazu CMP).

SHL SHR przesuniecie logiczne, bity wychodzace z rejestru są tracone

ROL,ROR przesunięcie cykliczne, bity wychodzące są zawracane i wprowadzane od drugiej strony rejestru

Omawiany sposób kodowania wymaga więc
przechowywania liczby w postaci dwóch elementów:
 mantysy, stanowiącej wartość liczby
znormalizowaną do przedziału (–2,–1> lub <1,2)
 wykładnika, opisującego liczbę przesunięć kropki w
prawo lub w lewo

mantysa * 2^wykladnik

1<=|mantysa|<2

32 bity -> float 1 bit znaku 8 bitow wykladnika 23 bity mantysy wykładnik +127
64 bity -> double 1 bit znaku 11 bitow wykladnika 52 bity mantysy wykładnik +1023

nie kodujemy części całkowitej mantysy

binary16 mantysa na 10 (11) bitach, wykładnik na 5 bitach, stosowane w grafice komputerowej

Format 128-bitowy (ang. quadruple precision)
oznaczany jako binary128 – mantysa zapisywana
jest na 112 (113) bitach, a wykładnik na 15 bitach;
format ten używany jest w obliczeniach
wymagających bardzo dużej dokładności.

Liczby, na których wykonywane są obliczenia, składowane są w 8 rejestrach 80-bitowych tworzących stos rejestrów koprocesora arytmetycznego

Z każdym rejestrem związane jest 2-bitowe pole stanu rejestru (nazywane także polem znaczeń); wszystkie pola stanu tworzą 16-bitowy rejestr zwany rejestrem stanu stosu koprocesora; interpretacja pola stanu jest następująca:
0 – rejestr zawiera liczbę różną od zera,
1 – rejestr zawiera zero,
2 – rejestr zawiera błędny rezultat,
3 – rejestr jest pusty.

Zawartość rejestru stanu stosu koprocesora można skopiować do pamięci za pomocą rozkazu FSTENV lub FSAVE.

finit ->inicjalizacja koprocesora

fld -> zaladowanie wartosci na wierzcholek stsou

litera p (pop) w mnemoniku fsubp oznacza
; polecenie usunięcia liczby z wierzchołka stosu
; koprocesora

Zawartość rejestru stanu koprocesora może być przesłana do rejestru AX za pomocą rozkazu
fstsw ax
 W języku C rejestr ten można odczytać za pomocą funkcji _status87

Zawartość rejestru sterującego koprocesora można także odczytać i zapisać na poziomie języka C za pomocą funkcji _control87

Bity RC określają rodzaj stosowanego zaokrąglenia (do liczby reprezentowalnej w formacie zmiennoprzecinkowym, na rysunku w postaci kropki):
00 – zaokrąglenie do liczby najbliższej (zob. rys.)
11 – zaokrąglenie w kierunku zera (zob. rys.)
01 – zaokrąglenie w dół (w kierunku −)
10 – zaokrąglenie w górę (w kierunku +)

FLD – ładowanie na wierzchołek stosu koprocesora liczby zmiennoprzecinkowej pobranej z lokacji pamięci lub ze stosu koprocesora;

FST – przesłanie zawartości wierzchołka stosu do lokacji pamięci lub do innego rejestru stosu koprocesora.

Rozkaz FILD pobiera z pamięci liczbę całkowitą w kodzie U2, następnie zamienia ją na liczbę zmiennoprzecinkową w formacie 80-bitowym i zapisuje na stosie rejestrów koprocesora.

Rozkaz FIST przesyła liczbę z wierzchołka stosu do lokacji pamięci z jednoczesnym przekształceniem jej na format całkowity w kodzie U2.

Zawartości rejestrów procesora (np. ECX) nie mogą być argumentami rozkazów koprocesora – z tego względu poniższy rozkaz jest błędny:

FLDZ – ładowanie 0
FLD1 – ładowanie 1
FLDPI – ładowanie pi
FLDL2E – ładowanie log 2 e
FLDL2T – ładowanie log 2 10
FLDLG2 – ładowanie log 10 2
FLDLN2 – ładowanie log e
2

Jeśli operand w pamięci jest liczbą całkowitą w kodzie U2, to stosuje się mnemonik rozkazu z dodatkową literą I, np.
FIADD word PTR wspolczynnik

fstp usuwa liczbe z wierzcholka stosu

Rozkaz FCHS przeprowadza zmianę znaku liczby na wierzchołku stosu koprocesora (st(0)) i nie ma operandów

Rozkaz FXCH zamienia zawartość wierzchołka stosu koprocesora z zawartością podanego operandu. Rozkaz ma tylko jeden operand, np. fxch st(5). W postaci bez operandów rozkaz zamienia zawartości rejestrów koprocesora st(0) i st(1). W szczególności rozkazy fxch i fxch st(1) działają identycznie.

. Rozkaz FRNDINT zaokrągla liczbę znajdującą się w st(0) do najbliższej liczby całkowitej (typ zaokrąglenia zależy od zawartości bitów RC w rejestrze sterującym koprocesora)

Rozkaz MOVSB powoduje przesłanie bajtu

wskazanego przez adres zawarty w rejestrze ESI do lokacji pamięci wskazanej przez rejestr EDI, i następnie zawartości obu tych rejestrów są:
• zwiększane o 1, gdy bit DF w rejestrze stanu procesora zawiera 0;
• zmniejszane o 1, gdy bit DF w rejestrze stanu procesora zawiera 1.

Rozkaz MOVSW powoduje przesłanie słowa (16 bitów), a rozkaz MOVSD powoduje przesłanie podwójnego słowa (32 bity) – odpowiednio zmieniane są też (o 2 lub 4) zawartości rejestrów indeksowych ESI i EDI.

Rozkaz CLD wpisuje 0 do znacznika DF, a rozkaz STD wpisuje 1 do znacznika DF.

Przedrostek powtarzania REP (kod F3H) umieszczony przed rozkazem MOVSB, MOVSW lub MOVSD, powoduje powtarzanie wykonywania rozkazu – po każdym powtórzeniu zawartość rejestru ECX jest zmniejszana o 1, a gdy ECX = 0, to powtarzanie zostaje zakończone.

Rozkazy LODS… (np. LODSW) i STOS… można uważać za uproszczone wersje rozkazu MOVS…
 Rozkaz LODS… przesyła zawartość 1-, 2- lub 4 bajtowego obszaru wskazanego przez rejestr ESI do rejestru AL, AX lub EAX.
 Rozkaz STOS… przesyła zawartość rejestru AL, AX lub EAX do lokacji pamięci wskazanej przez rejestr EDI; rozkaz STOS… poprzedzony przedrostkiem REP jest przydatny do inicjalizacji dużych obszarów pamięci.
 Do porównywania zawartości obszarów pamięci używane są rozkazy CMPS i SCAS

Jeśli przed kodem rozkazu zostanie umieszczony dodatkowy bajt 66H (przedrostek rozmiaru operandu), to działanie zostanie wykonane na operandzie 16-bitowym (a nie na 32-bitowym)