Pomoc 2.0 stworzył Konrad „kodzak” Kochaniak - kodzak16@gmail.com
Podziękowania dla wszystkich którzy pomogli przy powstaniu tego pdf-a.

Kopiując tekst używamy instrukcji
stosb

Instrukcja zamieniająca liczbę bez znaku na podwójne słowo
movsx

Instrukcja zamieniająca liczbę ze znakiem bajt na podwójne słowo
movzx

Przesuniecie arytmetyczne w lewo realizuje instrukcja
sal

Przesuniecie arytmetyczne w prawo realizuje instrukcja:
sar

Przesuniecie logiczne w lewo realizuje instrukcja
shl

Przesuniecie logiczne w prawo realizuje instrukcja:
shr

Przeszukiwanie bitow wstecz realizuje instrukcja:
bsr

Przeszukiwanie bitow w przód realizuje instrukcja:
bsf

Ile operacji na słowach może wykonać jedna instrukcja MMX
4

Ile operacji na bajtach może wykonać jedna instrukcja MMX
8

Ile rejestrów ogólnego przeznaczenia dołożono w trybie EMT64T procesorów Intel
4

Do zmiany kolejnosci slow w rejestrze MMX sluzy instrukcja:
pshufw

Pakowanie z nasyceniem podwójnych słów ze znakiem do słów realizuje instrukcja
packssdw

Ktora z instrukcji nie jest poprawna:
fsubp st,st(1)

Ktora z instrukcji jest poprawna:
movsx ebx, al;

Która z instrukcji umożliwia wpisanie wartości do dwóch rejestrów
LDS

Ktora z instrukcji umozliwia dodanie trzech wart.
xadd

Ktora z instrukcji pozwala na poszukiwanie podanego znaku w tekscie
scasb

Prefix LOCK moze odnosic sie do instrukcji:
xchg

Do prostego szyfrowania danych moze sluzyc instrukcja:
xlatb

Do odwolania sie do zmiennych lokalnych stosuje sie rejestr:
EBP

Dla wartosci calkowitej wystepuje dla liczb zmiennoprzecinkowych:
rozszerzonej precyzji

Ktora z instrukcji tworzy rame stosu:
enter

Która instrukcja nie zmienia flagi CF:
inc (zmienia flagi Wpływa na flagi: OSZAP)

Ktora z instrukcji neguje flage CF:
cmc

Ktora z instrukcji zmienia flage C:
fcomi

Ktora komenda nie zmienia flagi P:
NOT

Która z instrukcji wpisuje 0 d flagi CF:
clc

Ktora z instrukcji wpisuje 1 do flagi CF:
stc

Ktora z instrukcji zmienia flage Z:
popf

Instrukcja dec zmienia flagi:
OSZAP

Instrukcja add al,80h w programie add al,bl...neguje flagi:
SF i CF

Do odwolania sie do parametrow aktualnych stosuje sie rejestr:
CS

Instrukcje łańcuchowe używają segmentów:
ES i CS - ES na pewno

Wspolczese procesory i7 zbudowane sa z okolo:
zadne z powyzszych

Procesory Core 2 posiadają współczynnik IPC równy:
3.5

Jednostka zarządzania pamięcią w procesorach Intel została wprowadzona w:
80386

Z ilu tranzystorow zbudowany jest 8086
29 tys.

Technologia EM64T po raz pierwszy pojawila sie w procesorze:
Pentium 4

Instrukcjie AVX Intel wprowadzil po raz pierwszy w proc:
Sandy Bridge

Instrukcje SSE Intel wprowadził po raz pierwszy w procesorze
Pentium III

Instrukcje SSE2 Intel wprowadził po raz pierwszy w procesorze
Pentium IV

Ktory z procesorow jako pierwszy mogl wspolpracowac z koprocesorem:
8088 lub 80486 dx

Dwa rdzenie po raz pierwszy pojawily sie w procesorze:
Pentium 4

Ile etapow przetwarzania rozkazu wystepuje w Intel Pentium III:
12

W ktorym procesorze Intel mozna obliczyc adres instrukcji w postaci CS*16+IP
8086

Glownym konstruktorem procesora 8086 byl:
Steven Morse

W ktorym procesorze Intel po raz pierwszy zastosoal tryb chroniony:
80286

Ile rejestrow XMM posiadaja w trybie EMT64T procesory Intel:
żadne z powyższych - EMT64T nie istnieje

Ile rejestrow XMM posiadaja w trybie EM64T procesory Intel:
16

FLAGI
CF (carry flag - flaga przeniesienia) - gdy przekroczysz FFFFh lub poniżej zera
OF (overflow flag - flaga przepełnienia) - gdy przekroczysz 8000h w górę lub w dół (przekroczona maksymalna dodania lub minimalna ujemna)
SF (sign flag - flaga znaku) - gdy najstarszy bit = 1 (liczba ujemna)
ZF (zero flag - flaga zera) - gdy wynik operacji = 0
PF (parity flag - flaga parzystości) - gdy liczba ma parzystą ilość jedynek
AF (auxiliary carry flag - flaga przeniesienia pomocniczego) - gdy nastąpiło przeniesienie lub pożyczka między 3 i 4 bitem liczby

instrukcje logiczne:
and modyfikuje flagi: OF i CF = 0; SF, ZF i PF nabywają wartość zależną od wyniku.
neg nie rusza żadnych flag
not nie rusza żadnych flag
or modyfikuje flagi: OF i CF = 0; SF, ZF i PF nabywają wartość zależną od wyniku.
shl modyfikuje flagi: CF (przyjmuje wartość ostatniego bitu "wyrzuconego" poza obręb); SF, ZF i PF (za dużo pierdolenia)
shr modyfikuje flagi: CF (przyjmuje wartość ostatniego bitu "wyrzuconego" poza obręb); SF, ZF i PF (za dużo pierdolenia)
test modyfikuje flagi: OF i CF = 0; SF, ZF i PF nabywają wartość zależną od wyniku; stan AF staje się niezdefiniowany.
xor modyfikuje flagi: OF i CF = 0; SF, ZF i PF nabywają wartość zależną od wyniku

instrukcje arytmetyczne:
adc modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
add modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
cmp modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
dec modyfikowane flagi: OF, SF, ZF, AF i PF.
div modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
idiv modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
inc modyfikowane flagi: OF, SF, ZF, AF i PF.
mul modyfikowane flagi: OF, SF, ZF, AF, PF i CF.
sbb modyfikowane flagi: OF, CF, SF, ZF, AF i PF.
sub modyfikowane flagi: OF, CF, SF, ZF, AF i PF.

instrukcje transferowe:
clc flaga CF = 0;
cld flaga DF = 0;
cli flaga IF = 0;
cmc negacja CF;
stc flaga CF = 1;
std flaga DF = 1;
sti flaga IF = 1;

NIE ZMIENIAJA ŻADNYCH FLAG:
MOV
XCHG
BSWAP
PUSH

POP
PUSHF/PUSHFD
PUSHA/PUSHAD
POPA/POPAD
CWD/CDQ
CBW/CWDE
MOVSX
MOVZX
CMOVcc
JZ
JMP
JCXZ/JECXZ
LOOP
LOOPZ/LOOPE
LOOPNZ/LOOPNE
CALL
RET
INT
INTO
IRET
BOUND
ENTER
LEAVE
LAHF
SETcc
RDTSC
MOVS/MOVSB
MOVS/MOVSW
MOVS/MOVSD
LODS/LODSB
LODS/LODSW
LODS/LODSD
STOS/STOSB
STOS/STOSW
LOCK
NOP
LEA
UD2
XLAT/XLATB
MOVBE
CPUID

ZMIENIAJĄ PRZYNAJMNIEJ JEDNA:
STC C
CLC C
CMC C
STD D
CLD D
STI I
CLI I
FCOMI/FCOMIP/FUCOMI/FUCOMIP ZF PF CF

ADRESOWANIE:

Tryby adresowania - rejestrowy
```
push ebx
mov edx,ebx
inc ecx
```

Tryby adresowania - prosty - natychmiastowy
```
mov al, 5
mov edi, offset tabela
jnz petla
```

Tryby adresowania - bezpośredni
```
mov al, [1234ec5fh]
mov edi, tabela ;pobiera pierwszy element
mov zmienna, edx
```

Tryby adresowania - pośredni - rejestrowy
```
mov al, [ecx]
mov edi, [ebx]
mov [edi], edx
```

Tryby adresowania - pośredni - bazowy
```
mov al, [ebx+5]
mov edi, [ebx+tablica]
mov [ebp+8], edx
```

Tryby adresowania - pośredni - indeksowy
```
mov al, [esi]
mov edi, [esi*4+tablica]
mov [edi*8+tablica], edx
```

Tryby adresowania - pośredni – bazowo-indeksowy
```
mov al, [ebx+esi+3]
mov edi, [ebx+eax*4]
mov [ebp+edi*4+tablica], edx
```