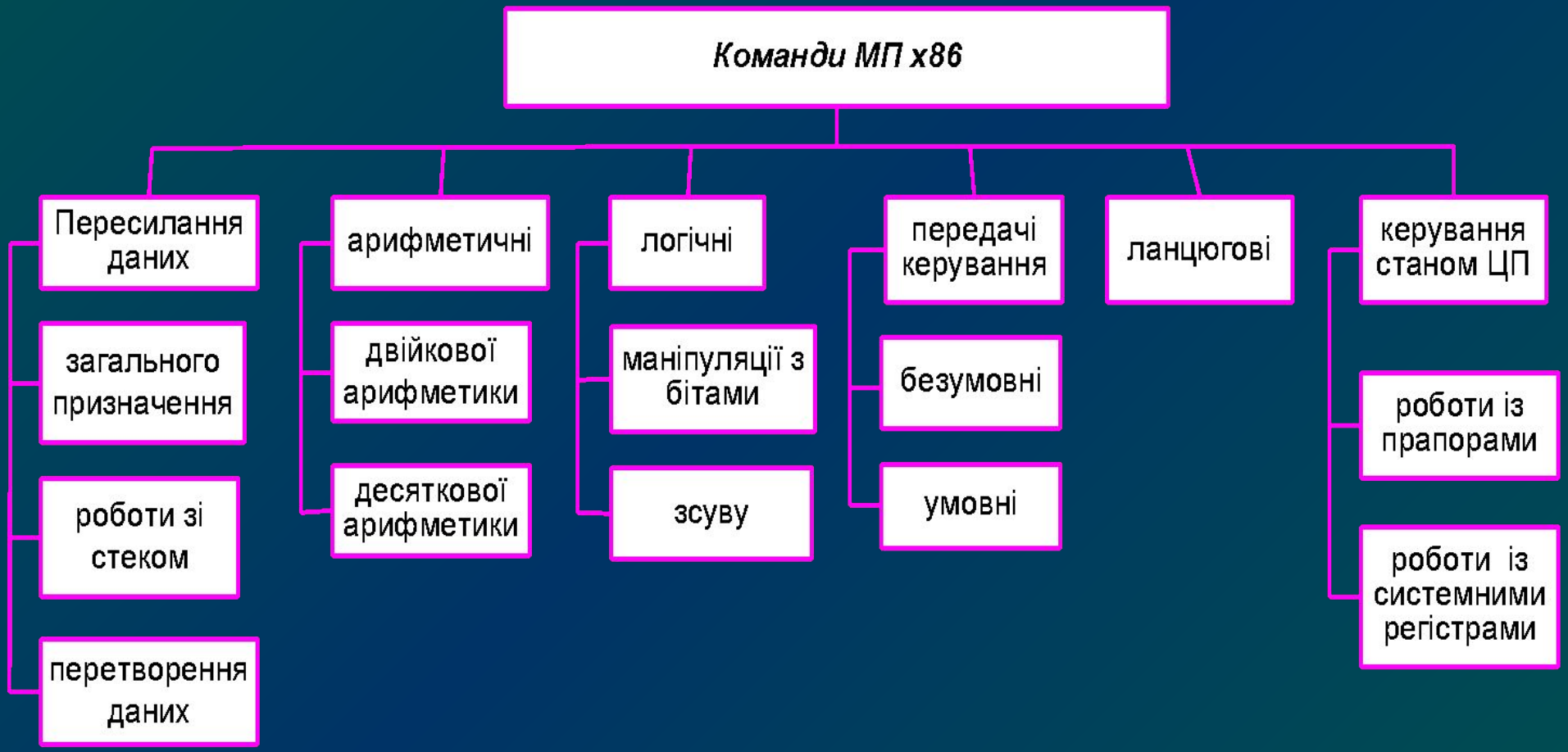
ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА МІКРОПРОЦЕСОРИ

(курс лекцій)

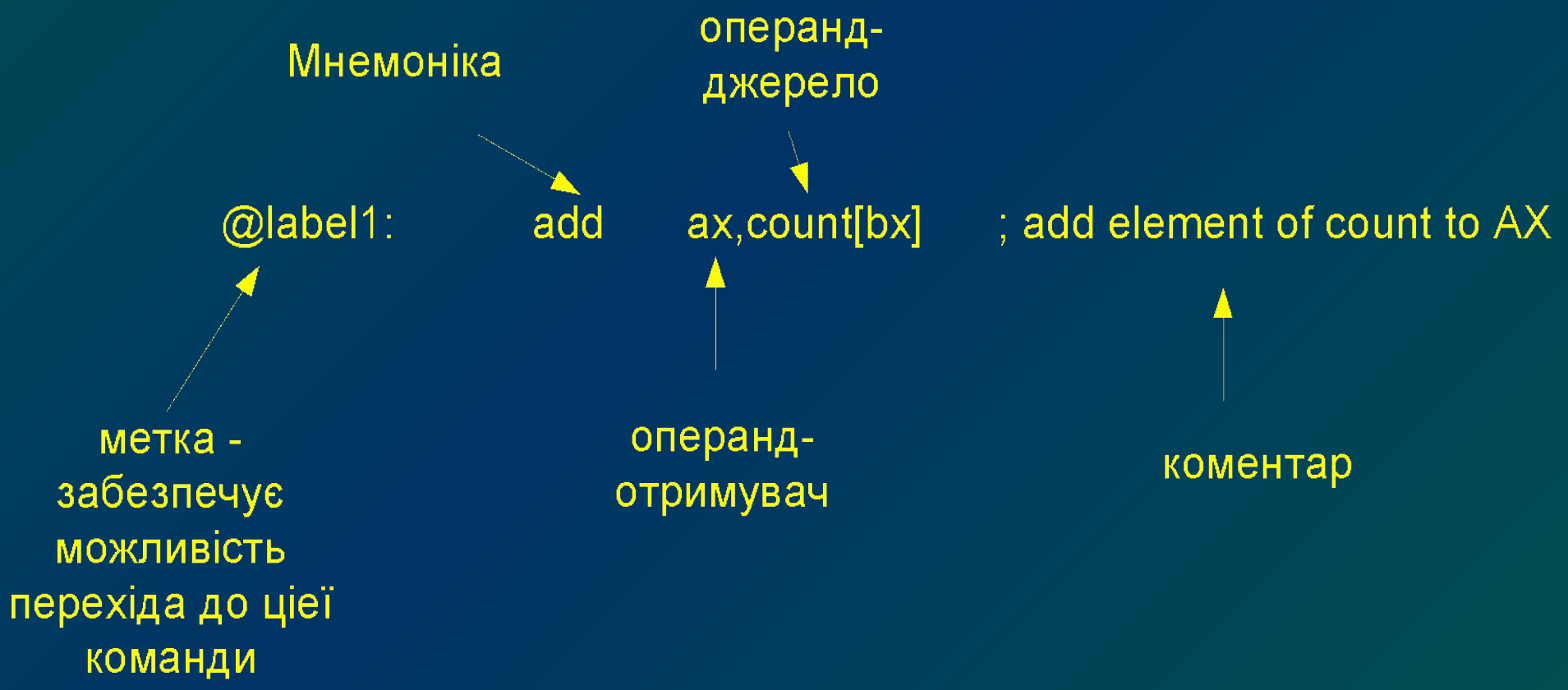
доцент ШВЕЦЬ Валеріян Анатолійович

Система команд МП x86


Команди МП x86
Пересилання даних
загального призначення
роботи зі стеком
перетворення даних
арифметичні
двійкової арифметики
десяткової арифметики
логічні
маніпуляції з бітами
зсуву
передачі керування
безумовні
умовні
ланцюгові
керування станом ЦП
роботи із прапорами
роботи із системними регістрами

Формат асемблерних команд

мітка: мнемоніка операнд, операнд; коментар



Команди пересилання даних

MOV - пересилка

(1-й Операнд) ← (2-й Операнд)

MOV reg8/mem8,reg8
MOV reg16/mem16,reg16
MOV reg8,reg8/mem8
MOV reg16,reg16/mem16
MOV Segreg,reg16/mem16
MOV reg16/mem16,Segreg
MOV reg8/mem8,data8
MOV reg16/mem16,data16

XLAT − передача з таблиці

(AL) ← ((BX) + (AL))

XCHG − заміна

(Тимчасове зберігання) ← **(Перший операнд)**
(Перший операнд) ← (Другий операнд)
(Другий операнд) ← (Тимчасове зберігання).
XCHG reg8,reg8/mem8
XCHG reg16,reg16/mem16

Команди пересилання даних

PUSH - запис у стек
(SP) ← (SP) - 2
(Вершина стека) ← (Операнд).
PUSH reg16/mem16
PUSH Segreg

POP – читання зі стека
(Операнд) ← (Вершина стека)
(SP) ← (SP) + 2
POP reg16/mem16
POP Segreg

PUSHF – запис у стек змісту регістра прапорів
(SP) ← **(SP) - 2**
(Вершина стека) ← (Регістр прапорів)

POPF – читання зі стека змісту регістра прапорів
(Регістр прапорів) ← **(Вершина стека)**
(SP) ← (SP) + 2

Команди пересилання даних

IN – ввід із порту

(Акумулятор) ← (Порт вводу-виводу).

IN AL,port8
IN AX,port8
IN AL,DX
IN AX,DX

OUT – вивід у порт

(Порт вводу-виводу) ← **(Акумулятор).**

OUT port8,AL
OUT port8,AX
OUT DX,AL
OUT DX,AX

Арифметичні команди *(додавання)*

ADD – додавання двох операндів

(1-й операнд) ← (1-й операнд)+ (2-й операнд)

ADD reg8/mem8,reg8
ADD reg16/mem16,reg16
ADD reg8,reg8/mem8
ADD reg16,reg16,mem16
ADD reg8/mem8,data8
ADD reg16/mem16,data16

ADC – додавання двох операндів із переносом

(1-й операнд) ← (1-й операнд)+ (2-й операнд)+ *CF*

ADC reg8/mem8,reg8
ADC reg16/mem16,reg16
ADC reg8,reg8/mem8
ADC reg16,reg16,mem16
ADC reg8/mem8,data8
ADC reg16/mem16,data16

***AAA* – корекція коду ASCII при додаванні**

Якщо *((AL) and 0FH) >9* або *(AF)=1*, то

(AL) ← (AL)+6
(AH) ← (AH)+1
(CF) ← 1
(AF) ← 1
(AL) ← (AL) and 0FH

Арифметичні команди *(додавання)*

INC – інкремент

(Операнд) ← (Операнд) + 1

INC reg8/mem8

INC reg16/mem16

INC reg16

DAA – десяткова корекція акумулятора при додаванні

Якщо *((AL) and 0FH) > 9* або *(AF) = 1*, то

(AL) ← (AL) + 6, (AF) ← 1.

***Якщо (AL) > 9FH* або *(CF) = 1*, то**

(AL) ← (AL) + 60, (CF) ← 1.

Арифметичні команди *(віднімання)*

SUB − віднімання
(1-й операнд) ← (1-й операнд) - (2-й операнд)
SUB reg8/mem8,reg8
SUB reg16/mem16,reg16
SUB reg8,reg8/mem8
SUB reg16,reg16,mem16
SUB reg8/mem8,data8
SUB reg16/mem16,data16

AAS − корекція коду ASCII при відніманні
Якщо *((AL) and 0FH) >9* або *(AF)=1*, то
(AL) ← (AL) - 6
(AH) ← (AH) - 1
(CF) ← 1
(AF) ← 1
(AL) ← (AL) and 0FH

SBB − віднімання з позикою
(1-й операнд) ← (1-й операнд) - (2-й операнд) − (прапор переносу *CF*)
SBB reg8/mem8,reg8
SBB reg16/mem16,reg16
SBB reg8,reg8/mem8
SBB reg16,reg16,mem16
SBB reg8/mem8,data8
SBB reg16/mem16,data16

Арифметичні команди *(віднімання)*

DEC – зменшення вмісту регістра або комірки пам'яті

(Операнд) ← (Операнд) -1

DEC reg8/mem8
DEC reg16/mem16
DEC reg16

NEG – заперечення

(Операнд) ← 0- (Операнд)

NEG reg8/mem8
NEG reg16/mem16

DAS – десяткова корекція при відніманні

Якщо *((AL) and 0FH) > 9* або *(AF) = 1*, то *(AL) ← (AL) - 6, (AF) ← 1.*

Якщо *(AL) > 9FH* або *(CF) = 1*, то *(AL) ← (AL) - 60, (CF) ← 1.*

CMP – порівняння двох операндів

(1-й операнд) – (2-й операнд)

CMP reg8/mem8,reg8
CMP reg16/mem16,reg16
CMP reg8,reg8/mem8
CMP reg16,reg16/mem16
CMP reg8/mem8,data8
CMP reg16/mem16,data16

Арифметичні команди *(множення)*

MUL – множення двох операндів

MUL reg8/mem8

MUL reg16/mem16

Однобайтова операція.

(*AX*) ← (*AL*)*(Операнд)

Якщо (*AH*) = 0, то (*CF*) ← 0, (*OF*) ← 0.

Якщо (*AH*) ≠ 0, то (*CF*) ← 1, (*OF*) ← 1.

(8)*(8)=(16)

AL*reg8/mem8=AX

Операція зі словами.

(DX,AX) ← (AX)*(Операнд)

Якщо (DX) = 0, то (CF) ← 0, (OF) ← 0.

Якщо (DX) ≠ 0, то (CF) ← 1, (OF) ← 1.

(16)*(16)=(32)

AX*reg16/mem16=DXAX

Арифметичні команди *(множення)*

IMUL – цілочислове множення зі знаком

IMUL reg8/mem8
IMUL reg16/mem16

Однобайтова операція.
(*AX*) ← (*AL*)*(Операнд)
Якщо (*AH*) = знакове розширення (*AL*), то (*CF*) ← 0, (*OF*) ← 0.
Якщо (*AH*) ≠ знакове розширення (*AL*), то (*CF*) ← 1, (*OF*) ← 1.

Операція зі словами.
(DX,AX) ← (AX)*(Операнд)
Якщо (DX) = знакове розширення (AX), то (CF) ← 0, (OF) ← 0.
Якщо (DX) ≠ знакове розширення (AX), то (CF) ← 1, (OF) ← 1.

AAM – корекція коду ASCII при множенні

(AL) ← (AH) × 10 + (AL)
(AH) ← 0

Арифметичні команди *(ділення)*

DIV – ділення двох операндів

DIV reg8/mem8
DIV reg16/mem16
((AX): (Операнд)
(AL) ← частка, (AH) ← залишок.
((AX): (Операнд) >FFH), генерує переривання типу 0.

(DX,AX) : (Операнд)
(AX) ← частка, (DX) ← залишок.
Якщо ((DX,AX) : (Операнд) > FFFFH), генерує переривання типу 0.

IDIV – цілочислове ділення зі знаком

DIV reg8/mem8
DIV reg16/mem16
((AX): (Операнд)
(AL) ← частка, (AH) ← залишок.
((AX): (Операнд) >FFH), генерує переривання типу 0.

(DX,AX) : (Операнд)
(AX) ← частка, (DX) ← залишок.
Якщо ((DX,AX) : (Операнд) > FFFFH), генерує переривання типу 0.

Арифметичні команди *(ділення)*

AAD – корекція коду ASCII при діленні

(AL) ← частка від ділення (AL) : 10

(AH) ← залишок від ділення (AL) : 10

Арифметичні команди *(поширення знаку)*

CBW – поширення знака регістра AL на регістр AH

Якщо *(AL) < 80H*, то *(AH) ← 0*

Якщо (AL) ≥ 80H*, то *(AH) ← FFH

CWD – поширення знака з регістра AX у регістр DX

Якщо (AX) < 8000H, то (DX) ← 0

Якщо (AX) ≥ 8000H, то (DX) ← FFFFH

Логічні команди *(маніпуляції з бітами)*

AND – логічне множення двох операндів

(1-й операнд) ← **(1-й операнд)** ***and*** **(2-й операнд)**

(CF) ← 0

(OF) ← 0

AND reg8/mem8,reg8

AND reg16/mem16,reg16

AND reg8,reg8/mem8

AND reg16,reg16,mem16

AND reg8/mem8,data8

AND reg16/mem16,data16

OR – логічне АБО

(1-й Операнд) ← (1-й Операнд) АБО (2-й Операнд).

(CF) ← 0

(OF) ← 0

OR reg8/mem8,reg8

OR reg16/mem16,reg16

OR reg8,reg8/mem8

OR reg16,reg16/mem16

OR reg8/mem8,data8

OR reg16/mem16,data16

Логічні команди *(маніпуляції з бітами)*

XOR – АБО, що виключає

(1-й операнд) ← (1-й операнд) *xor* (2-й операнд).

(*CF*) ← 0

(*OF*) ← 0

```
XOR     reg8/mem8,reg8
XOR     reg16/mem16,reg16
XOR     reg8,reg8/mem8
XOR     reg16,reg16,mem16
XOR     reg8/mem8,data8
XOR     reg16/mem16,data16
```

NOT – логічне заперечення

(Операнд) ← Інверсія всіх розрядів (Операнд)

TEST – тест

(Перший операнд) *and* (Другий операнд).

(*CF*) ← 0

(*OF*) ← 0.

```
TEST    reg8/mem8,reg8
TEST    reg16/mem16,reg16
TEST    reg8/mem8,data8
TEST    reg16/mem16,data16
```

Логічні команди *(команди зсуву)*

SAL/SHL – арифметичний зсув вліво/логічний зсув вліво

(CF) ← (Старший біт операнда),
(Операнд) ←(Операнд)*2.

SAL reg8/mem8,1
SAL reg16/mem16,1
SAL reg8/mem8,CL
SAL reg16/mem16,CL

SAR – арифметичний зсув вправо

(CF) ← (Молодший біт операнда),
(Операнд) ← (Операнд)/2.

SAR reg8/mem8,1
SAR reg16/mem16,1
SAR reg8/mem8,CL
SAR reg16/mem16,CL

Логічні команди *(команди зсуву)*

SHR – логічний зсув вправо
(CF) ← (Молодший біт операнда),
(Операнд) ← (Операнд)/2
SHR reg8/mem8,1
SHR reg16/mem16,1
SHR reg8/mem8,CL
SHR reg16/mem16,CL

Логічні команди *(команди циклічного зсуву)*

ROL – циклічний зсув вліво

(CF) ← (Старший біт операнда)

(Операнд) ← (Операнд)*2 +(*CF*)

ROL reg8/mem8,1

ROL reg16/mem16,1

ROL reg8/mem8,CL

ROL reg16/mem16,CL

ROR – циклічний зсув вправо

(CF) ← (Молодший біт операнда)

(Операнд) ← (Операнд)/2

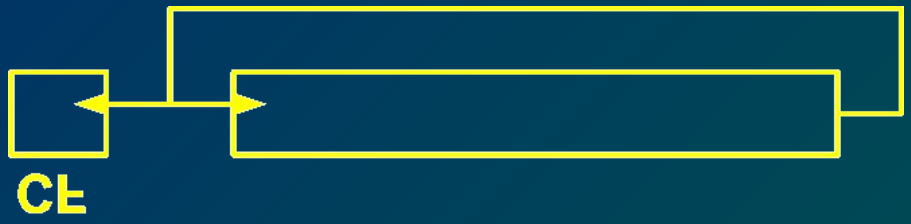
(Старший біт операнда) ← (CF)

ROR reg8/mem8,1

ROR reg16/mem16,1

ROR reg8/mem8,CL

ROR reg16/mem16,CL



Логічні команди *(команди циклічного зсуву)*

RCL – циклічний зсув уліво з переносом

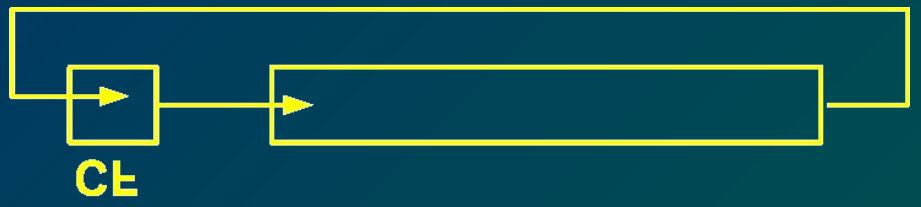
(Тимчасовий біт) ← (*CF*)
(*CF*) ← (Старший біт операнда)
(Операнд) ← (Операнд)*2 + (Тимчасовий біт).

RCL reg8/mem8,1
RCL reg16/mem16,1
RCL reg8/mem8,CL
RCL reg16/mem16,CL

RCR – циклічний зсув управо з переносом

(Тимчасовий біт) ← *CF*
(*CF*) ← (Молодший біт операнда)
(Операнд) ← (Операнд)/2
(Старший біт операнда) ← (Тимчасовий біт)

RCR reg8/mem8,1
RCR reg16/mem16,1
RCR reg8/mem8,CL
RCR reg16/mem16,CL



Команди передачі керування *(умовні)*

JA/JNBE − перехід, якщо більше/перехід, якщо не менше або дорівнює
Якщо ((*CF*) = 0 і (*ZF*) = 0), то (*IP*) ← (*IP*) + *disp* 8.
JA short-label
JNBE short-label

JAE/JNB/JNC − перехід, якщо більше або дорівнює/перехід,
якщо не менше/перехід, якщо немає переносу
Якщо (*CF*) = 0, то (*IP*) ← (*IP*) + *Disp8*

JB/JNAE/JC − перехід, якщо менше/перехід,
якщо не більше або дорівнює/перехід, якщо є перенос
Якщо (*CF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

JBE/JNA − перехід, якщо менше або дорівнює/перехід,
якщо не більше
Якщо ((*CF*) = 1 або (*ZF*) = 1), то (*IP*) ← (*IP*) + *Disp8*.

JCXZ − перехід, якщо вміст регістра CX дорівнює нулю
Якщо (*CX*) = 0, то (*IP*) ← (*IP*) +*Disp8*.

Команди передачі керування *(умовні)*

JE/JZ – перехід, якщо дорівнює/перехід по нулі
Якщо (*ZF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

JG/JNLE – перехід, якщо більше
ніж/перехід, якщо не менше ніж або дорівнює
Якщо ((*SF*) = (*OF*) *and* (*ZF*) = 0), то (*IP*) ← (*IP*) + *Disp8*.

JGE/JNL – перехід, якщо більше або дорівнює/перехід, якщо не менше ніж
Якщо (*SF*) = (*OF*), то (*IP*) ← (*IP*) + *Disp8*.

JL/JNGE – перехід, якщо менше/перехід, якщо не більше або дорівнює
Якщо (*SF*) ≠ (*OF*), то (*IP*) ← (*IP*) +*Disp8*.

JLE/JNG – перехід, якщо менше або дорівнює/перехід, якщо більше
Якщо (*SF*) ≠ (*OF*) або (*ZF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

Команди передачі керування *(умовні)*

JNE/JNZ – перехід по нерівності/перехід, якщо не нуль
Якщо (*ZF*) = 0, то (*IP*) ← (*IP*) + *Disp8*

JNO – перехід, якщо немає переповнювання
Якщо (*OF*) = 0, то (*IP*) ← (*IP*) + *Disp8*.

JNP/JPO – перехід при відсутності парності
Якщо (*PF*) = 0, то (*IP*) ← (*IP*) + *Disp8*.

JNS – перехід, якщо немає знака
Якщо (*SF*) = 0, то (*IP*) ← (*IP*) + *Disp8*.

JO – перехід по переповнюванню
Якщо (*OF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

JP/JPE – перехід по парності
Якщо (*PF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

JS – перехід за знаком
Якщо (*SF*) = 1, то (*IP*) ← (*IP*) + *Disp8*.

Команди передачі керування *(безумовні)*

JMP – безумовний перехід

Межсегментний перехід:

(*CS*) ← сегмент цільового операнда,

(*IP*) ← адреса цільового операнда, що переміщається

При внутрисегментном переході:

(*IP*) ← адреса цільового операнда, що переміщається.

CALL – виклик процедури (підпрограми)

Межсегментний виклик:

(SP) ← (SP) - 2

(Вершина стека) ← (CS)

(SP) ← (SP) - 2

(Вершина стека) ← (IP)

(CS) ← Сегмент процедури

(IP) ← Зміщення процедури

Внутрисегментний виклик:

(SP) ← (SP) - 2

(Вершина стека) ← (IP)

(IP) ← Зміщення процедури

```
CALL    FAR LABEL  ; межсегментний виклик
CALL    NEAR LABEL ;усередині сегментний виклик
CALL    reg16/mem16 ;усередині сегментний
CALL    mem16 ;межсегментний виклик
```

Команди передачі керування *(безумовні)*

RET − повернення з процедури

(IP) ← (Вершина стека),
(SP) ← (SP) + 2
Межсегментна процедура
(CP) ← (Вершина стека),
(SP) ← (SP) + 2
При наявності зміщення
(SP) ← (SP) + зміщення.

RET
RET disp16

IRET − повернення з переривання

(*IP*) ← (Вершина стека)
(*SP*) ← (*SP*) + 2
(*CS*) ← (Вершина стека)
(*SP*) ← (*SP*) + 2
(Регістр прапорів) ← (Вершина стека)
(*SP*) ← (*SP*) + 2

Команди передачі керування *(керування циклами)*

LOOP – цикл
(CX) ← (CX) - 1
Якщо (CX) ≠ 0, то (IP) ← (IP) + Disp8.
LOOP short label.

LOOPE/LOOPZ – цикл якщо дорівнює/цикл якщо нуль
(CX) ← (CX) - 1
Якщо ((CX) ≠ 0 and (ZF) = 1), то (IP) ← (IP) + Disp8.

LOOPNE/LOOPNZ – цикл якщо не дорівнює/цикл якщо не нуль
(CX) ← (CX) - 1
Якщо ((CX) ≠ AND (ZF) = 0), то (IP) ← (IP) + Disp8.

```
        mov cx,100
@lab1:  add  dx,bx
        loop @lab1
```

Команди обробки рядків

(префікси повторення)

REP/REPE/REPZ – повторення/повторення, якщо дорівнює/повторення, якщо нуль
REPNE/REPNZ – повторення, якщо дорівнює/повторення, якщо нуль
(вказівка умовного й безумовного повторення наступної за даною командою ланцюгової операції)

```
      mov     cx,100
rep movs      dest,source
```

(пересилання)

MOVS, MOVSB, MOVSW пересилка рядка
(Операнд за адресою в регістрі DI) ← (Операнд за адресою в регістрі SI).
Якщо (DF) = 0, то (SI) ← (SI)+ 1 (байт) (DI) ← (DI)+ 1 (байт),
або (SI) ← (SI)+ 2 (слово) (DI) ← (DI)+ 2 (слово)
Якщо (DF) = 1, то (SI) ← (SI) - 1 (байт) (DI) ← (DI) - 1 (байт),
або (SI) ← (SI) - 2 (слово) (DI) ← (DI)-2(слово)

Команди обробки рядків *(порівняння)*

CMPS, CMPSB, CMPSW – порівняння рядків
(операнд за адресою в регістрі SI) – (операнд за адресою в регістрі DI).
***Якщо (DF) = 0,* то *(SI) ← (SI)+ 1* (байт) *(DI) ← (DI)+ 1* (байт),**
***або (SI) ← (SI)+ 2* (слово) *(DI) ← (DI)+ 2* (слово)**
***Якщо (DF) = 1,* то *(SI) ← (SI) - 1* (байт) *(DI) ← (DI) - 1* (байт),**
***або (SI) ← (SI) - 2* (слово) *(DI) ← (DI) - 2* (слово)**

```
    lea si,sourse
    lea di,dest
    mov     cx,100
rep cmps    dest,source
rep cmpsb
rep cmpsw
```

Команди обробки рядків *(сканування)*

SCAS, SCASB, SCASW − сканування рядка

(Акумулятор) − (Операнд за адресою в DI)
Якщо (DF) = 0, то ← (DI)+ 1 (байт), або (DI) ← (DI)+ 2 (слово)
Якщо (DF) = 1, то (DI) ← (DI) - 1 (байт), або (DI)← (DI) - 2 (слово)

Команди обробки рядків *(завантаження й збереження)*

LODS, LODSB, LODSW - завантаження рядка

(Акумулятор) ← (Операнд за адресою в *SI*)
Якщо (*DF*) = 0, то (*SI*) ← (*SI*) + 1 (байт), або (*SI*) ← (*SI*) + 2 (слово).
Якщо (*DF*) = 1, то (*SI*) ← (*SI*) - 1 (байт), або (SI) ← (SI) - 2 (слово) .

STOS, STOSB, STOSW − запам'ятати рядок

(Операнд за адресою в регістрі DI) ← (Акумулятор).
Якщо (DF) = 0, то ← (DI)+ 1 (байт), або (DI) ← (DI)+ 2 (слово)
Якщо (DF) = 1, то (DI) ← (DI) - 1 (байт), або (DI)← (DI) - 2 (слово)

Команди керування станом процесора

(роботи з прапорами)

CLC – очищення прапора переносу

(CF) ← 0

CLD – очищення прапора напрямку

(DF) ← 0

CLI – очищення прапора переривання

(IF) ← 0

CMC – інвертування прапора переносу

Якщо (CF) = 0, то (CF) ← 1

Якщо (CF) = 1, то (CF) ← 0

STC – встановити прапор переносу

(CF) ← 1

STD – встановити прапор напрямку

(DF) ← 1

STI – встановити прапор переривання

(IF) ← 1

Команди керування станом процесора

(роботи зі системними регістрами)

LDS – завантаження покажчика в регістр DS

(1-й Операнд) ← (Виконавча адреса), (Регістр *DS*) ← (Виконавча адреса + 2).

LDS reg16,mem16.

LEA – завантаження виконавчої адреси

(1-й Операнд) ← Виконавча адреса 2-го операнда.

LEA reg16,mem16.

LES – завантаження покажчика з використанням регістра ES

(1-й Операнд) ← (Виконавча адреса 2-го операнда),

(Регістр ES) ← (Виконавча адреса 2-го операнда + 2).

LES reg16,mem16.

Команди керування станом процесора

(роботи зі системними регістрами)

INT – програмне переривання

(*SP*) ← (*SP*) - 2
(Вершина стека) ← (Регістр прапорів)
(*SP*) ← (*SP*) - 2
(Вершина стека) ← (*CS*)
(*SP*) ← (*SP*) - 2
(Вершина стека) ← (IP)
(*IF*) ← 0
(*TF*) ← 0
(*CS*) ← (Тип переривання * 4 + 2)
(*IP*) ← (Тип переривання * 4)
INT type8

IRET – повернення з переривання

(*IP*) ← (Вершина стека)
(*SP*) ← (*SP*) + 2
(*CS*) ← (Вершина стека)
(*SP*) ← (*SP*) + 2
(Регістр прапорів) ← (Вершина стека)
(*SP*) ← (*SP*) + 2

Команди керування станом процесора

(роботи зі системними регістрами)

INTO – переривання по переповнюванню

(*SP*) ← (*SP*) - 2
(Вершина стека) ← (Регістр прапорів)
(*SP*) ← (*SP*) - 2
(Вершина стека) ← (*CS*)
(*SP*) ← (*SP*) - 2
(Вершина стека) ← (IP)
(*IF*) ← 0
(*TF*) ← 0
(*CS*) ← (*12H*)
(*IP*) ← (*10H*)

HLT – зупин **NOP – відсутність операції**

ESC – видача

ESC opcode,reg8/mem8
ESC opcode,reg16/mem16
Якщо Mod ≠ 11B, то шина даних ← (EA)

LOCK - захоплення шини