Розділ 2. Команди МП 8088/86

2.1. Зміст команд

Мікропроцесор має ***92 команди***, які можна разділити на 7 груп:

1. Команди **пересилки** даних між регістрами, комірками і портами введення/виведення;
2. **Арифметичні команди**;
3. Команди над **бітами**, які здійснюють зсуви і логічні операції;
4. Команди **передачі управління**, виклику процедур і повернення з процедури;
5. Команди **обробки рядків**;
6. Команди **переривань** для обробки специфічних подій;
7. Команди **управління процесором** - встановлення і скидання прапорців стану, зміна режиму функціонування МП.

2.2 Арифметичні команди

Цілочисельний обчислювальний пристрій підтримує трохи більше десятка арифметичних команд.

В МП 8088/86 є група команд для реалізації 4 арифметичних дій (додавання, віднімання, множення, ділення) над цілими числами. Більшість двохоперандних команд діють так, що змінюється операнд-приймач, а операнд-джерело залишається незмінним.

Варіанти зміни прапорців:

1 - після виконання команди прапорець встановлюється (дорівнює 1);
0 - після виконання команди прапорець скидається (дорівнює 0);
r - значення прапорця залежить від результату роботи команди;
? - після виконання команди прапорець невизначений;
пробіл - після виконання команди прапорець не змінюється.

2.2.1. Команди додавання

Існують 3 команди:
ADD (add -- додати);
ADC (add with carry) – додати з переносом;
INC (increment) – додати одиницю.

ADD операнд_1, операнд_2 — команда додавання

Принцип дії:

операнд_1 = операнд_1 + операнд_2

ADD AX, CX ;	**AX + CX → AX**

Команда **ADD** працює в режимах:

```
ADD  AX, CX ;  регістр - регістр
ADD AX, ALPHA; регістр – пам’ять
ADD ALPHA, AX ; пам’ять - регістр
 ADD  AH, AL ; регістра байт - регістр байт
ADD  AX, 40 ; регістр - константа
ADD  BETA, 02Fh ; пам’ять - константа
```

Ця команда впливає на 6 прапорців

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	r

ADC операнд_1, операнд_2 — команда додавання з урахуванням прапору переносу **cf.**
Принцип дії команди:
операнд_1 = операнд_1 + операнд_2 + значення_cf

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	r

INC операнд - збільшення значення операнда в пам'яті або в регістрі на 1

11	07	06	04	02
OF	SF	ZF	AF	PF
r	r	r	r	r

2.2.2. Команди віднімання

Аналогічно командам додавання, існують три команди віднімання:

SUB (substract) - відняти;

SBB (substract with borrow) - відняти з займом;

DEC (DECrement) - відняти одиницю.

SUB операнд_1, операнд_2 - цілочисельне віднімання

операнд_1=операнд_1-операнд_2

Команда **SUB** працює в режимах:

SUB AX, CX ; ***регістр - регістр***
SUB AX, ALPHA; ***регістр – пам'ять***
SUB ALPHA, AX ; ***пам'ять - регістр***
SUB AH, AL ; ***регістра байт - регістр байт***
SUB AX, 40 ; ***регістр - константа***
SUB BETA, 02Fh ; ***пам'ять - константа***

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	r

SBB операнд_1, операнд_2 - цілочисельне віднімання з урахуванням результату попереднього віднімання командами **sbb** і **sub** (станом прапорця переносу **cf):**

1. виконати додавання операнд_2 = операнд_2 + **(cf);**

2. виконати віднімання операнд_1 = операнд_1- операнд_2**;**

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	r

DEC операнд - зменшення значення операнда в пам'яті або в регістрі на 1

11	07	06	04	02
OF	SF	ZF	AF	PF
r	r	r	r	r

DEC AX; AX -1 → AX

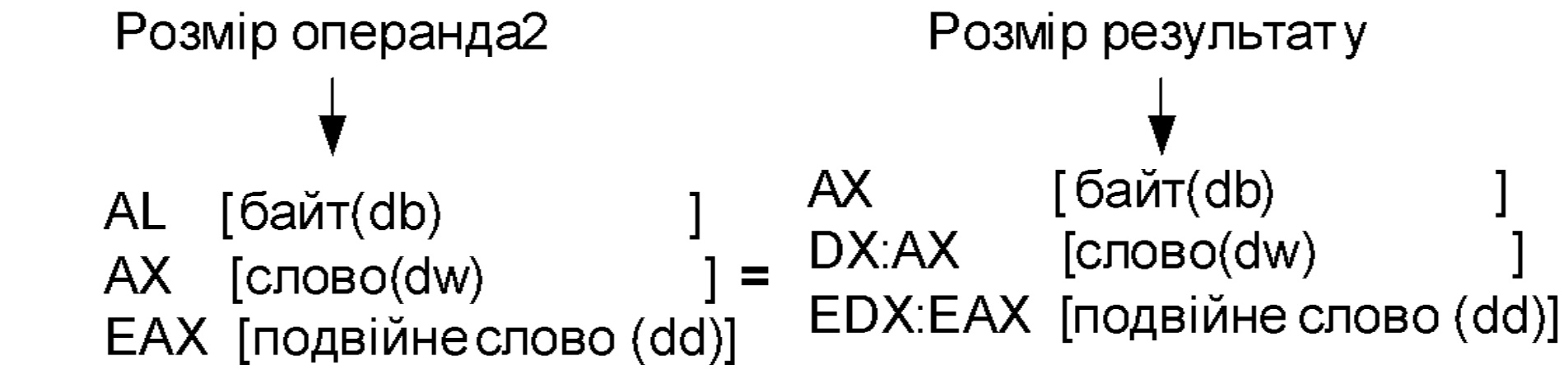
2.2.3. Команди множення

Існують дві команди:

MUL (multiply) - множення без знаку;

IMUL (integer multiply) – множення зі знаком.

MUL операнд2



Приклад:

```
MOV AL, ALPHA1 ; ALPHA1 - байт
MUL  ALPHA2 ; ALPHA1 + ALPHA2 → AX
```

```
MOV AX, ALPHAW1 ; ALPHAW1 - слово
MUL  ALPHAW2 ; ALPHAW1 + ALPHAW2 → DX:AX
```

```
MOV EAX, ALPHAD1 ; ALPHAD1 – 2 слова
MUL  ALPHAD2 ; ALPHAD1 + ALPHAD2 → EDX:EAX
```

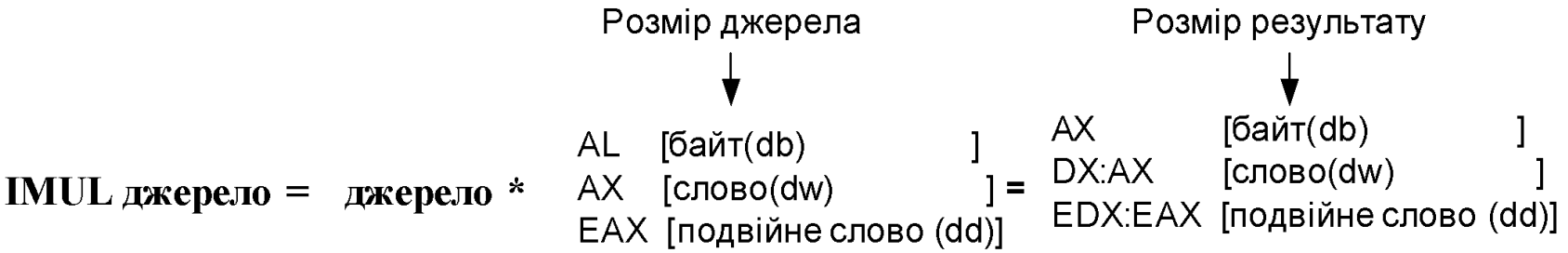
Стан прапорців після виконання команди (якщо ***старша половина результату нульова AH=0;DX=0;EDX=0***):

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
1	?	?	?	?	0

Стан прапорців після виконання команди (якщо ***старша половина результату ненульова AH≠0;DX≠0;EDX≠0***):

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
1	?	?	?	?	1

IMUL джерело



IMUL множ_1, множ_2 (IMUL AX, 5 AX = AX*5)

IMUL рез-т, множ_1, множ_2 (IMUL DI, AX, 5 DI = AX*5)

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	?	?	?	?	r

2.2.4. Команди ділення

Зазначимо, що ділення виконується як ділення цілих чисел, тобто, отримаємо частку (ціле) і залишок (ціле).

Отже, 19: 5 = 3 і 4 - залишок. Ніяких дійсних чисел не отримаємо.

DIV (divide) - ділення чисел без знаку;

IDIV (integer divide) - ділення цілих чисел (зі знаком).

DIV дільник

ділене / дільник	Розмір діленого: Байт (db)	Слово (dw)	Подвійне слово (dd)
ділене	AX	DX:AX	EDX:EAX
частка	AL	AX	EAX
залишок	AH	DX	EDX

Результати команд ділення на **прапорці не впливає**.
Коли ж частку повністю не можна розмістити в відведеному їй слові або в байті, то здійснюється переривання типу 0 - ділення на 0.

IDIV дільник

Залишок завжди має знак діленого. Знак частки залежить від стану знакових бітів (старших розрядів) діленого і дільника.

Приклад:

```
MOV AX, ALPHAW1 ; ALPHAW1 – слово
; ALPHAB1 - байт
DIV  ALPHAB1 ; ALPHAW1 / ALPHAB1 → AH:AL
```

```
MOV DX, ALPHAW1 ; ALPHAW1 – слово
MOV AX, ALPHAW2 ; ALPHAW2 – слово

; ALPHAW3 - слово
DIV  ALPHAW3
; (ALPHAW1: ALPHAW2) / ALPHAW3 → DX:AX
```

2.2.5. Команди зміни знаку

NEG (NEGate operand) - змінити знак операнда

NEG джерело

Стан прапорців після виконання команди (якщо результат нульовий):

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	0

Стан прапорців після виконання команди (якщо результат ненульовий):

11	07	06	04	02	00
OF	SF	ZF	AF	PF	CF
r	r	r	r	r	1

2.2.6. Команди розширення знаку

Щоб правильно подати код числа, яке записано в молодшому байті, при зчитуванні його з цілого слова і код числа, записаного в молодше слово, при зчитуванні його з молодшого слова, потрібно розширити знак числа відповідно на старший байт і старше слово.

Для цього існують дві команди:

CBW (convert byte to word) - перетворення байта в слово;

CWD (convert word to double word) - перетворення слова в подвійне слово;

CWDE (convert word to double word Extended) - перетворення слова в подвійне слово;

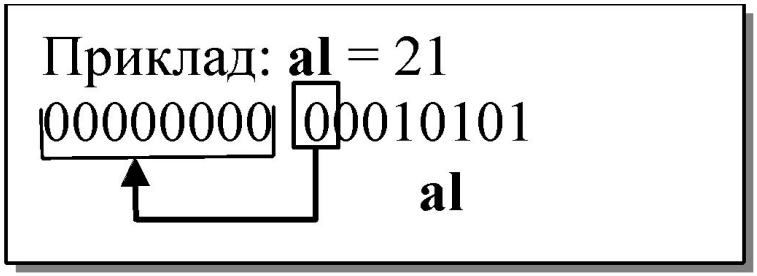
CDQ (Convert Double word to Quad word) - перетворення подвійного слова в два слова (чотири байти).

Алгоритм роботи:

CBW — при роботі команда використовує лише регістр **ax** и **al**:

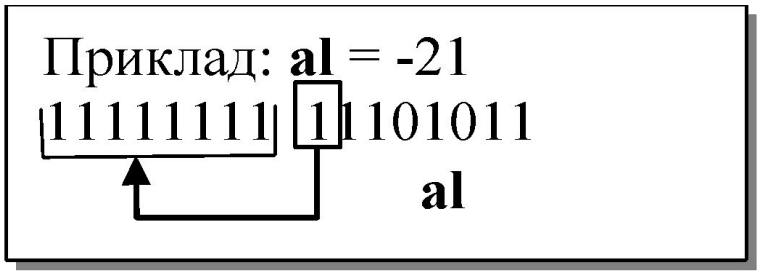
Додатнє число

аналіз знакового біту **al**:
якщо знаковий біт **al**=0, то **ah**=00h;



Від'ємне число

аналіз знакового біту **al**:
якщо знаковий біт **al**=1, то **ah**=0ffh



CWD - при роботі команда використовує лише регістри **ax** і **dx**:
аналіз знакового біта регістра al:
якщо знаковий біт **ax** = 0, то **dx** = 00h;
якщо знаковий біт **ax** = 1, то **dx** = 0ffh.

CWDE - при роботі команда використовує лише регістри **ax** і **eax**:
аналіз знакового біта регістра **ax**:
якщо знаковий біт **ax** = 0, то встановити старше слово **eax** = 0000h;
якщо знаковий біт **ax** = 1, то встановити старше слово **eax** = 0ffffh.
виконання команди ***не впливає на прапорці***

CDQ - при роботі команда використовує лише регістри **eax** і **edx**:
аналіз знакового біту регістра **eax**:
якщо знаковий біт **eax** = 0, то встановити старше слово **edx** = 0000h;
якщо знаковий біт **eax** = 1, то встановити старше слово **edx** = 0ffffh.
виконання команди ***НЕ впливає на прапорці***

Приклад:

Дані команди використовуються для приведення операндів до потрібної розмірності з урахуванням знаку. Така необхідність може, зокрема, виникнути при програмуванні арифметичних операцій.

```
.386 ;   лише для cwde

      MOV    EBX,10fecd23h
      MOV    AX,-3   ;AX=1111 1111 1111 1101
       CWDE    ;EAX=1111 1111 1111 1111 1111 1111 1111
1101
      ADD    EAX,EBX
```