Переваги мов програмування високого рівня – зручність програмування складних логічних задач.

Недоліки – порівняно невисока ефективність машинних програм.

Всі можливості ПЕОМ повністю не використовуються.

Що таке асемблер?

Приклад:

C++: **X=Y**

Асемблер: **MOV X,Y**

Після трансляції: **256 4725 0648**

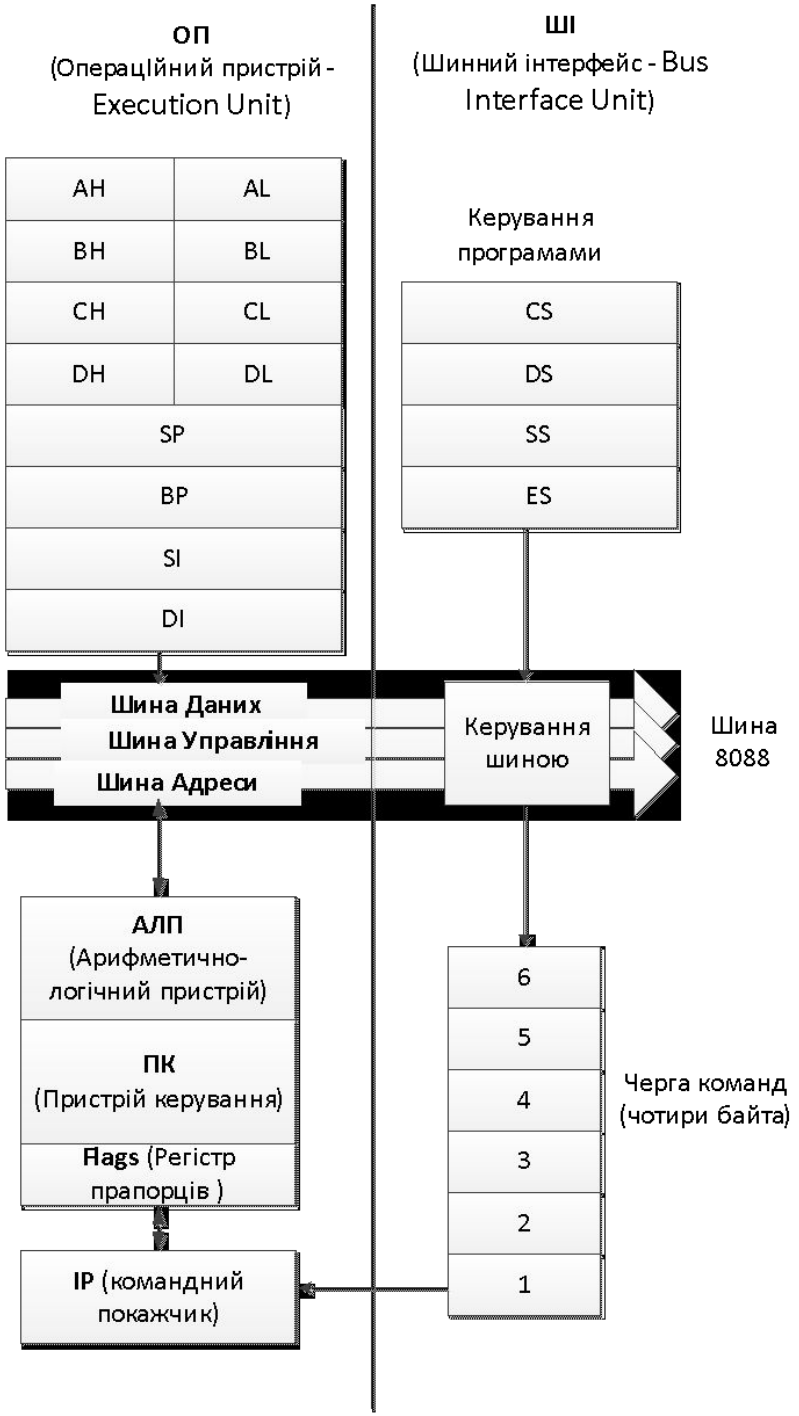
Де **256** - код команди;

4725 - адреса першого операнду;

0648 – адреса другого операнду.

ПЕОМ	Розрядність	Макс. Пам’ять, Мб
XT 8088/86	16	1
AT 80286	16	16
A 80386 DX	32	1024
80386 SX	16	16
80386 SL	16	16

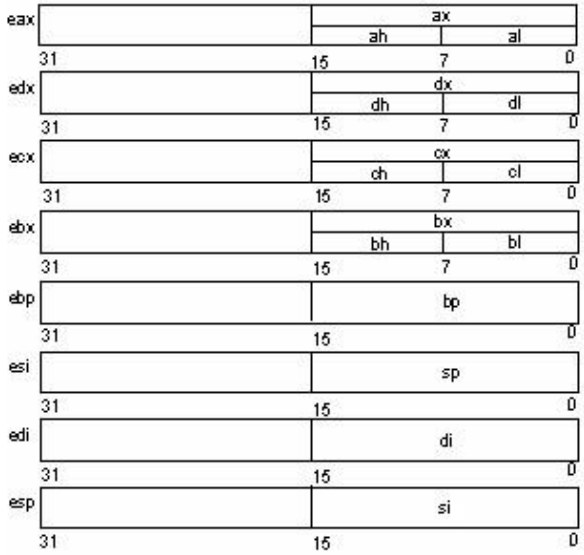
ПЕОМ складається з декількох функціональних пристроїв, які реалізують арифметичні і логічні операції, управління, запам’ятовування, занесення/одержання даних.



Розглянемо особливості використання окремих регістрів.
Регістр - це проміжна пам'ять, місткістю одне машинне слово (16 біт). В ЦП існують багато регістрів, більшість з яких недосяжні для програміста. Але є декілька регістрів, які використовуються в програмах мовою асемблер.

За регістрами закріплені назви і імена, через які до них можна звертатися:

1. 4 регістри загального призначення - **AX, BX, CX, DX**.
2. 4 регістри покажчики - **SP, BP, SI, DI**.
3. 4 сегментних регістри - **CS, DS, SS, ES**.
4. 2 керуючих регістри - покажчика команд **IP** і регістру прапорців **F**.



(Accumulator register) — **акумулятор**

(Data register) — **регістр даних**

(Count register) — **регістр-лічильник**

(Base register) — **базовий регістр**

(Base Pointer register) — **регістр покажчика бази кадру стеку**

(Stack Pointer register) — **регістр покажчика стеку**

(Destination Index register) — **індекс приймача(адресата)**

(Source Index register) — **індекс джерела**

(data segment register) — **сегментний регістр даних**

(code segment register) — **сегментний регістр коду**

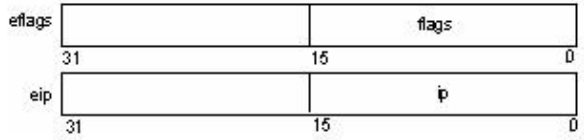
(stack segment register) — **сегментний регістр стеку**

Extension Data Segment registers
сегментні регістри додаткових сегментів даних

додаткового сегменту

універсальних сегментів

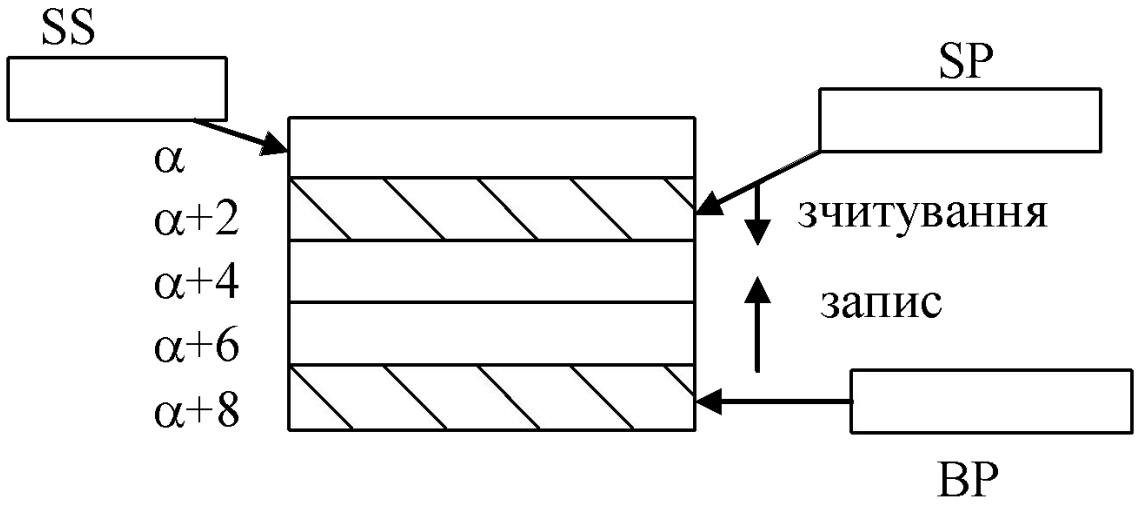
регістри прапорців і покажчиків команд



регістр прапорців

регістр покажчика команди (instruction pointer)

В сучасних ПЕОМ широко використовуються *стеки*, наприклад, при роботі з підпрограмами, при обробці переривань, а також для запам'ятовування проміжних результатів. *Стек* - це ділянка пам'яті, яка заповнюється в бік зменшених адрес, а звільняється в бік збільшення адрес. Тут реалізується метод "останній прийшов - першим обслуговується" (**LIFO**).
Стек характеризується своєю базовою адресою, яка записується через регістр **SS** адресою вершини, яка записується в регістр покажчика стеку **SP** (stack pointer), та бази стеку **BP**. В стек записуються слова (2 байти).



16 (hex)	10 (dec)	8 (oct)	2^4	2^3	2^2	2^1
0	0	0	0	0	0	0
1	1	1	0	0	0	1
2	2	2	0	0	1	0
3	3	3	0	0	1	1
4	4	4	0	1	0	0
5	5	5	0	1	0	1
6	6	6	0	1	1	0
7	7	7	0	1	1	1
8	8	10	1	0	0	0
9	9	11	1	0	0	1
A	10	12	1	0	1	0
B	11	13	1	0	1	1
C	12	14	1	1	0	0
D	13	15	1	1	0	1
E	14	16	1	1	1	0
F	15	17	1	1	1	1

Десяткове значення 42936

Перетворення десяткового формату в шістнадцятковий

Частка	Залишок	Шіст.		
42936 / 16	2683	8	8	(молодша цифра)
2683 / 16	167	11	B	
167 / 16	10	7	7	
10 / 16	0	10	A	(старша цифра)

Перетворення шістнадцяткового формату в десятковий

Перша цифра: A(10) 10

Помножити на 16 *16

160

Додати наступну цифру, 7 +7

167

Помножити на 16 *16

2672

Додати наступну цифру, B(11) +11

2683

Помножити на 16 *16

42928

Додати наступну цифру, 8 +8

Десяткове значення 42936

Для переведення необхідно поділити число, з залишком, на основу чисельника доти, доки частка більша за основу чисельника.

44_{10} Переведемо в двійкову систему

44 ділимо на 2. частка 22, залишок 0
22 ділимо на 2. частка 11, залишок 0
11 ділимо на 2. частка 5, залишок 1
5 ділимо на 2. частка 2, залишок 1
2 ділимо на 2. частка 1, залишок 0
1 ділимо на 2. частка 0, залишок 1
частка дорівнює нулю, ділення завешено. Тепер , записавши всі залишки справа наліво , отримаємо число:

101100_2

Перетворимо101100_2
Вісімкова— 101 100 $\rightarrow 54_8$
Шістнадцяткова— 0010 1100 $\rightarrow 2C_{16}$

```
; опис сегменту стеку
STSEG  SEGMENT  PARA  STACK  “STACK”
DB 64 DUP ( “STACK” )
STSEG  ENDS
; опис сегменту даних
DSEG  SEGMENT  PARA  PUBLIC  “DATA”
SOURCE  DB 10, 20, 30, 40
DEST  DB 4 DUP ( “?” )
DSEG  ENDS
; опис сегменту коду
CSEG  SEGMENT PARA PUBLIC  “CODE”
; код основной функции
MAIN  PROC  FAR
ASSUME  CS: CSEG, DS: DSEG, SS: STSEG
; адреса повернення
PUSH DS
MOV AX, 0    ; або XOR  AX, AX
PUSH AX
; ініціалізація DS
MOV AX, DSEG
MOV DS, AX
; обнулення масиву
```

```
MOV  DEST, 0
MOV  DEST+1, 0
MOV  DEST+2, 0
MOV  DEST+3, 0
; пересилання
MOV  AL, SOURCE
MOV  DEST+3, AL
MOV  AL, SOURCE+1
MOV  DEST+2, AL
MOV  AL, SOURCE+2
MOV  DEST+1, AL
MOV  AL, SOURCE+3
MOV  DEST, AL

RET

MAIN ENDP

CSEG ENDS

END MAIN
```