Регістрова архітектура системи команд

Формат команд

Структура і мікропрограми блоку виконання арифметичних операцій

Огляд IA-32

Формат команд на мові асемблера IA-32

Формат команд

Типова команда має операційну і адресну частини із наступною інформацією:

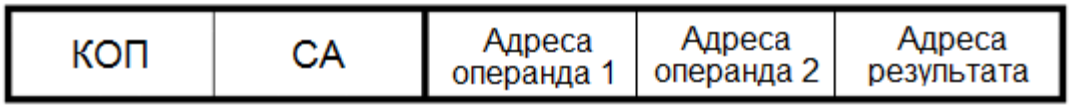
- Дія, яка виконується,
- Місцерозташування операндів,
- Місцерозташування результату.

Формат команд вказує на:

- Тип операцій в системі команд та їх кількість,
- Розрядність команд,
- Тип фрагментів команд («полів команди») та їх розрядність,
- Спосіб декодування команд,
- Кількість адрес в команді («адресність»),
- Способи доступу до даних («способи адресації»).

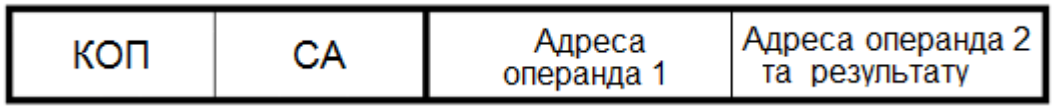
Адресність команд (3А, 2А)

Очевидний варіант адресності – наявність в команді трьох адрес:



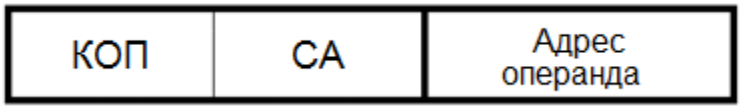
Раніше, наприклад в EDVAC (1952р), в команді була також четверта адреса для вказання номеру наступної команди. Необхідність в 4-й адресі зникла після введення в процесор «покажчика (лічильника) адреси команд» і використання впорядкованого розташування команд в пам’яті.

Використання адреси одного із операндів (наприклад, другого) в якості адреси результату дозволяє перейти до 2-адресних команд (після виконання команди операнд втрачений, тому що дані по відповідній адресі заміщуються на результат):



Адресність команд (1А, 0А)

Одноадресні команди приписують розташування одного із операндів та результату за фіксованою адресою (зазвичай, в одному із процесорних регістрів - «акумуляторі»):



Безадресні («нульадресні») команди приписують використання даних за фіксованими адресами розташування операндів та результату:

Критерії визначення адресності команд:
- Ємність пам’яті для розташування даних,
- Швидкодія виконання програми,
- Ефективність використання комірок пам’яті.

Приклад оцінки ефективності команд різної адресності

Необхідно обчислити: ***y*** = ***a*** x ***b*** + ***(c – d)*** x ***e*** / ***f***.

Для оцінки використовуються коефіцієнти:

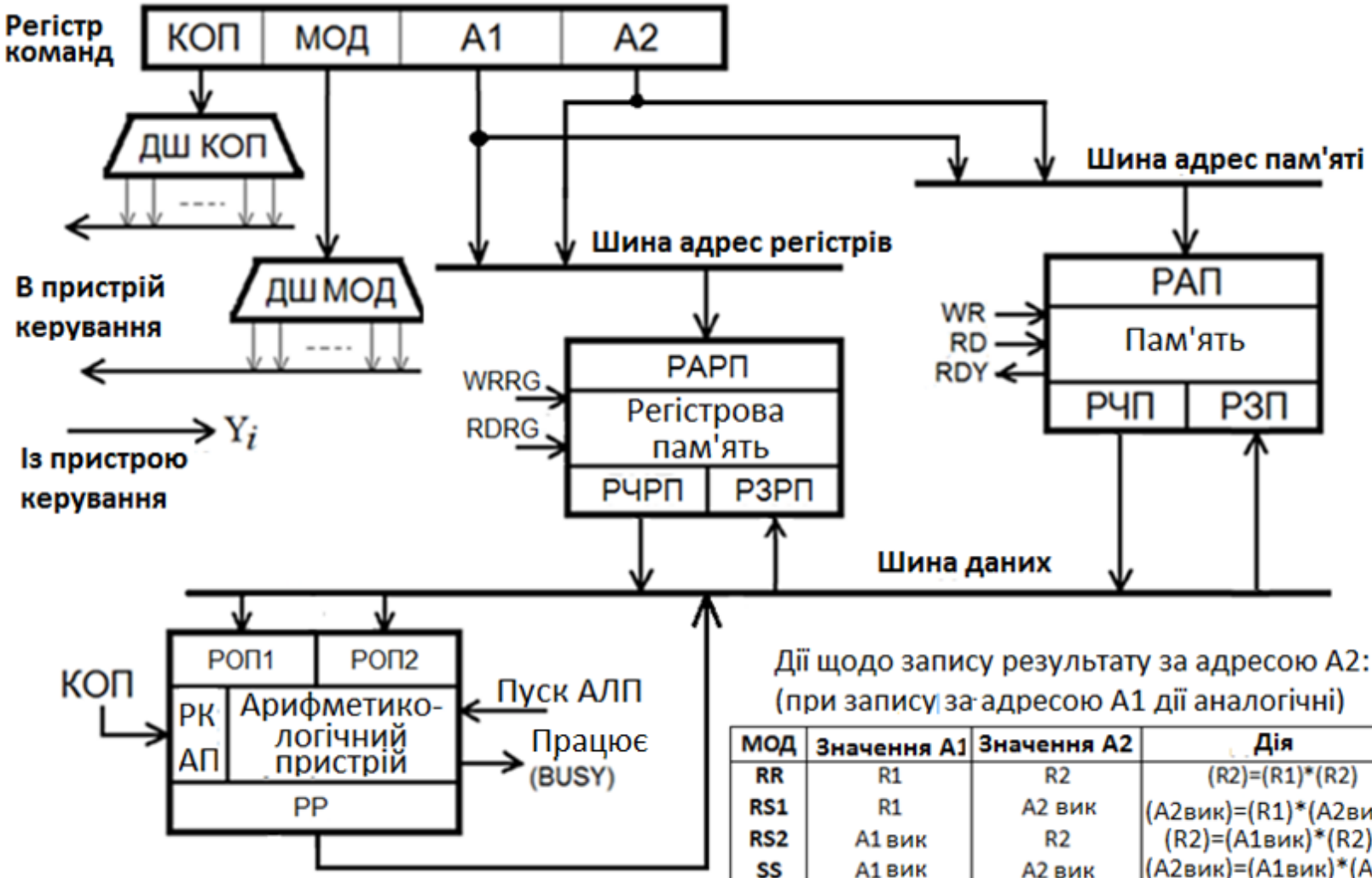
1. Використання адрес в команді: K = Ae/As (Ae – кількість адрес, що вказані в програмі, As – кількість адресних полів у всіх командах програми).
2. Звертань до пам'яті: T = Top + Tin (Top – кількість звертань до пам'яті для передавання даних, Tin - … команд).

Для прикладу: K3=9/15, K2=9/12, K1=9/9, T3=14(9+5), T2=15(9+6), T1=18(9+9).

В загальному випадку, ефективність залежить від прикладної області.

А) Команди 3А	Б) Команди 2А	В) Команди 1А
(***s*** - адреса комірки пам'яті, ***Acc*** - акумулятор):		
МН ***a, b, s ; a*** x ***b*** -> ***s***	МН ***a, b ; a*** x ***b*** -> ***Acc***	ЧТ ***a ; a*** -> ***Acc***
ВД ***c, d, - ; c – d*** -> ***Acc***	ЗП ***s, - ; Acc*** -> ***s***	МН ***b ; Acc*** x ***b*** -> ***Acc***
МН ***- , e, - ; Acc*** x ***e*** -> ***Acc***	ВД ***c, d ; c – d*** -> ***Acc***	ЗП ***s ; Acc*** -> ***s***
ДЛ ***- , f , - ; Acc / f*** -> ***Acc***	МН ***e, - ; Acc*** x ***e*** -> ***Acc***	ЧТ ***c ; c*** -> ***Acc***
ДД ***- , s, y ; Acc + s*** -> ***y***	ДЛ ***f , - ; Acc / f*** -> ***Acc***	ВД ***d ; Acc - d*** -> ***Acc***
	ДД ***s, y ; Acc + s*** -> ***y***	МН ***e ; Acc*** x ***e*** -> ***Acc***
		ДЛ ***f ; Acc / f*** -> ***Acc***
		ДД ***s ; Acc + s*** -> ***Acc***
		ЗП ***y ; Acc*** -> ***y***

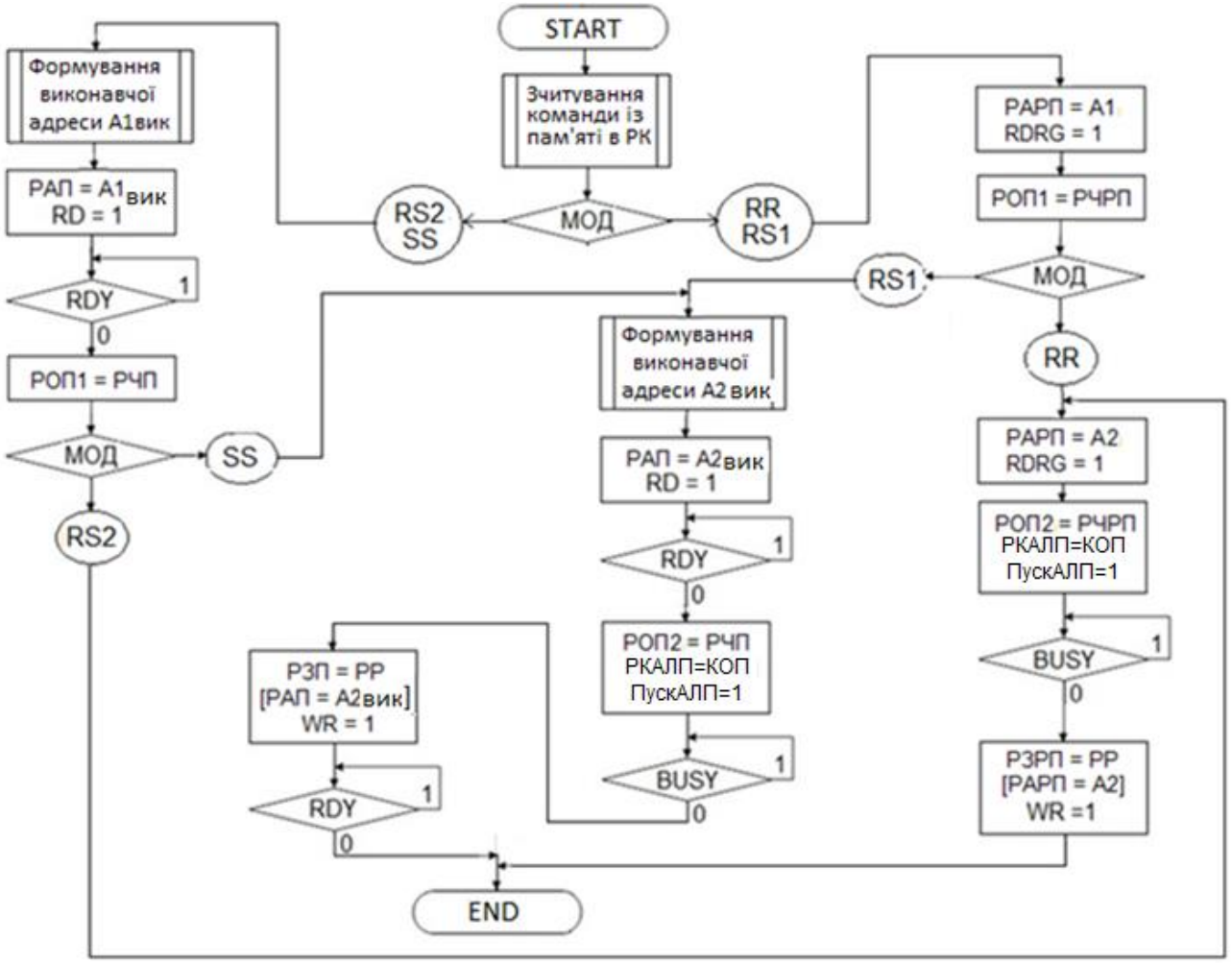
Структура блоку виконання арифметичних операцій адресності «2А» (із регістровою пам’яттю)



Дії щодо запису результату за адресою А2:
(при запису за адресою А1 дії аналогічні)

МОД	Значення А1	Значення А2	Дія
RR	R1	R2	(R2)=(R1)*(R2)
RS1	R1	А2 вик	(А2вик)=(R1)*(А2вик)
RS2	А1 вик	R2	(R2)=(А1вик)*(R2)
SS	А1 вик	А2 вик	(А2вик)=(А1вик)*(А2вик)

Мікропрограма виконання арифметичних операцій адресності «2А» (із регістровою пам’яттю)

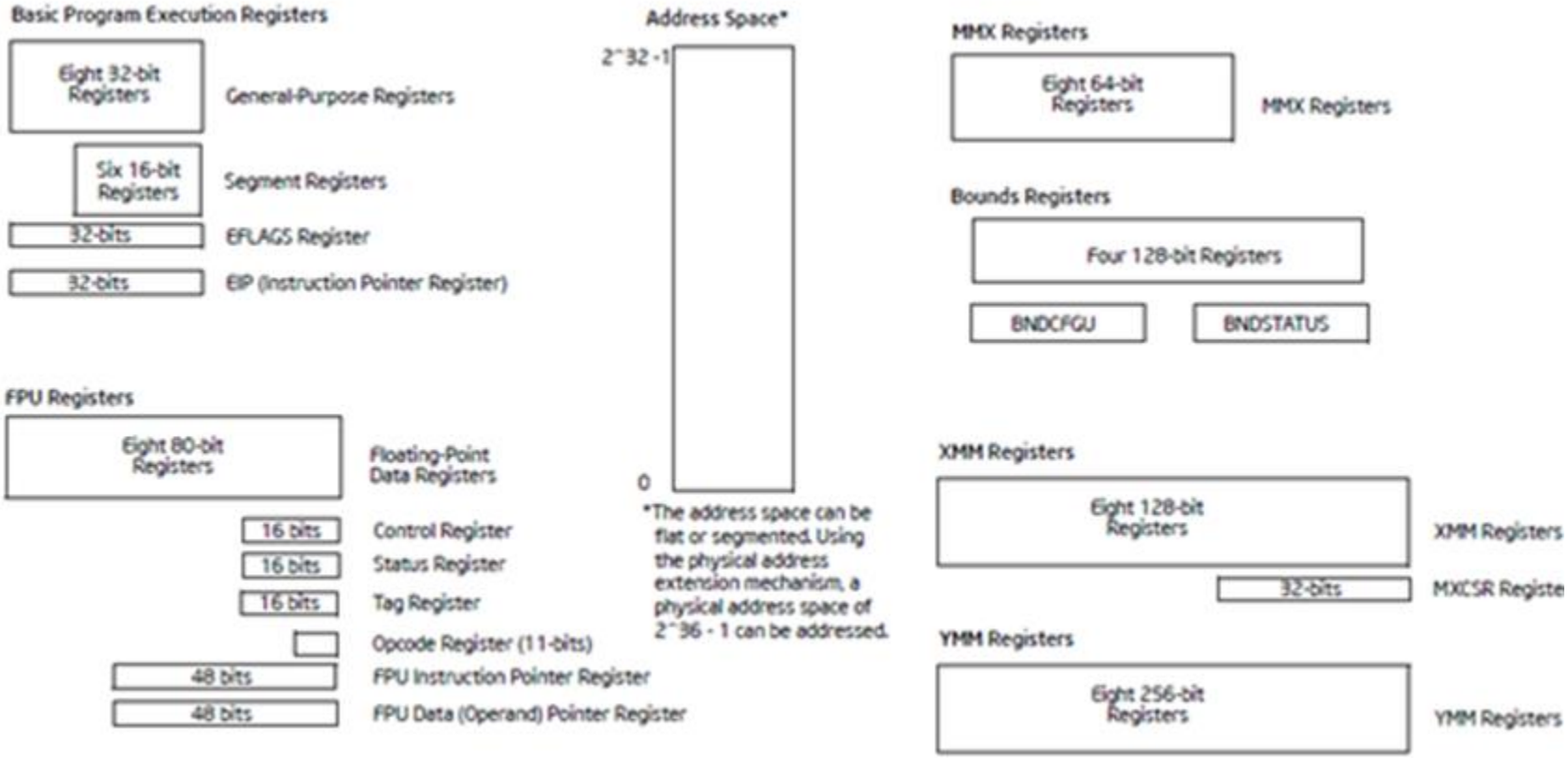


Дії блоку виконання арифметичних операцій адресності «2А»

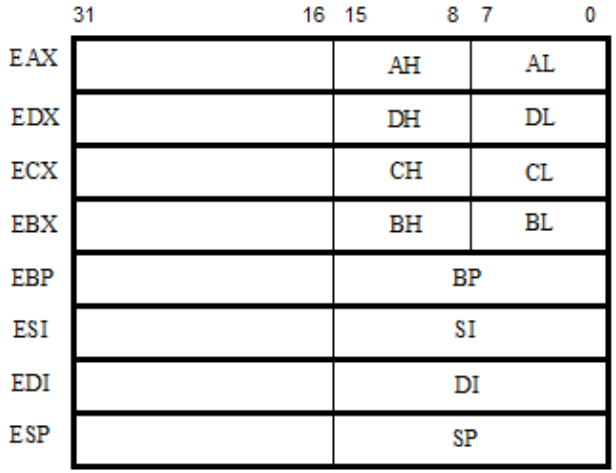
-Виявлення місцерозташування першого операнда,
-Формування виконавчої адреси (якщо перший операнд розташований в загальній пам’яті),
-Зчитування першого операнда (із регістра або пам’яті),
-Виявлення місцерозташування другого операнда,
-Формування виконавчої адреси (якщо другий операнд розташований в загальній пам’яті),
-Зчитування другого операнда (із регістра або пам’яті),
-Пересилання операндів в АЛП,
-Виконання операції в АЛП,
-Виявлення місцерозташування результату,
-Запис (занесення) результату (в регістр або пам’ять).

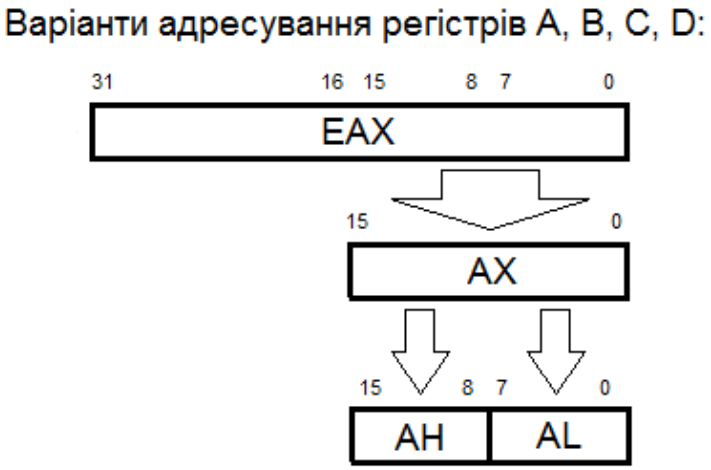
Виконавчий (ефективний) адрес – двійковий код номеру комірки пам’яті (що вказана в команді) для збереження операнда та/або результату.

Архітектура IA-32



Регістри загального призначення (General Purpose Registers)



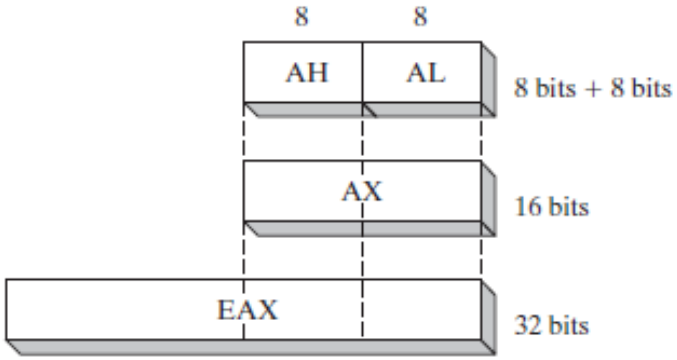


Використання регістрів в деяких командах (за умовчанням):

- EAX - акумулятор, джерело операнду або приймач результату;
- EBX - покажчик на дані в сегменті даних (DS);
- ECX - лічильник в рядкових (ланцюгових (MOVS)) і циклічних (із префіксом REP) командах;
- EDX - частина даних в арифметичних діях, адреса порту вводу-виводу в IN/INS, OUT/OUTS;
- ESI - покажчик на джерело операнду (індексний регістр джерела даних);
- EDI - покажчик на приймач операнду (індексний регістр приймача даних);
- EBP - покажчик на фрагмент даних в сегменті стеку (SS).

Регістр ESP завжди вказує на відносний адрес вершини стеку.

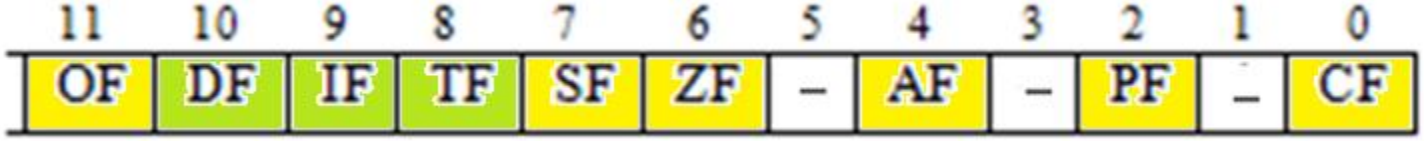
Розрядність і позначення регістрів



32-Bit	16-Bit	8-Bit (High)	8-Bit (Low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

32-Bit	16-Bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Регістр ознак/прапорів (EFLAGS Register) (12 молодших розрядів)

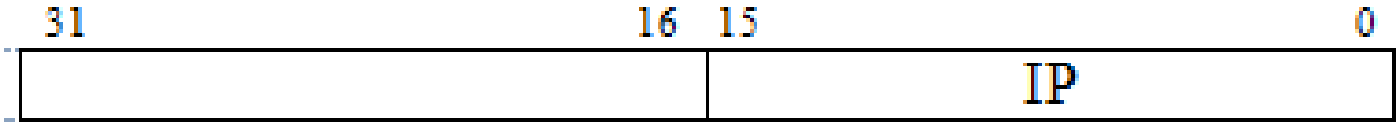


<u>Шість</u> ознак формуються відповідно до результату виконання операції:
- **CF** (carry) - перенесення із старшого розряду результату (із розряду (**n-1**) в неіснуючий розряд **n)**;
- **PF** (parity) – парний результат (кількість одиниць в молодшому байті результату є парним);
- **AF** (auxiliary carry) – перенесення між тетрадами молодшого байту (із 3 в 4 розряд результату);
- **ZF** (zero) - нульовий результат (всі розряди результату нульові);
- **SF** (sign) – знак результату (значення старшого розряду (**n-1**) результату);
- **OF** (overflow) - переповнення (перенесення із розряду (n-2) в розряд (n-1) не співпадає із перенесенням із розряду (n-1) в неіснуючий розряд n).

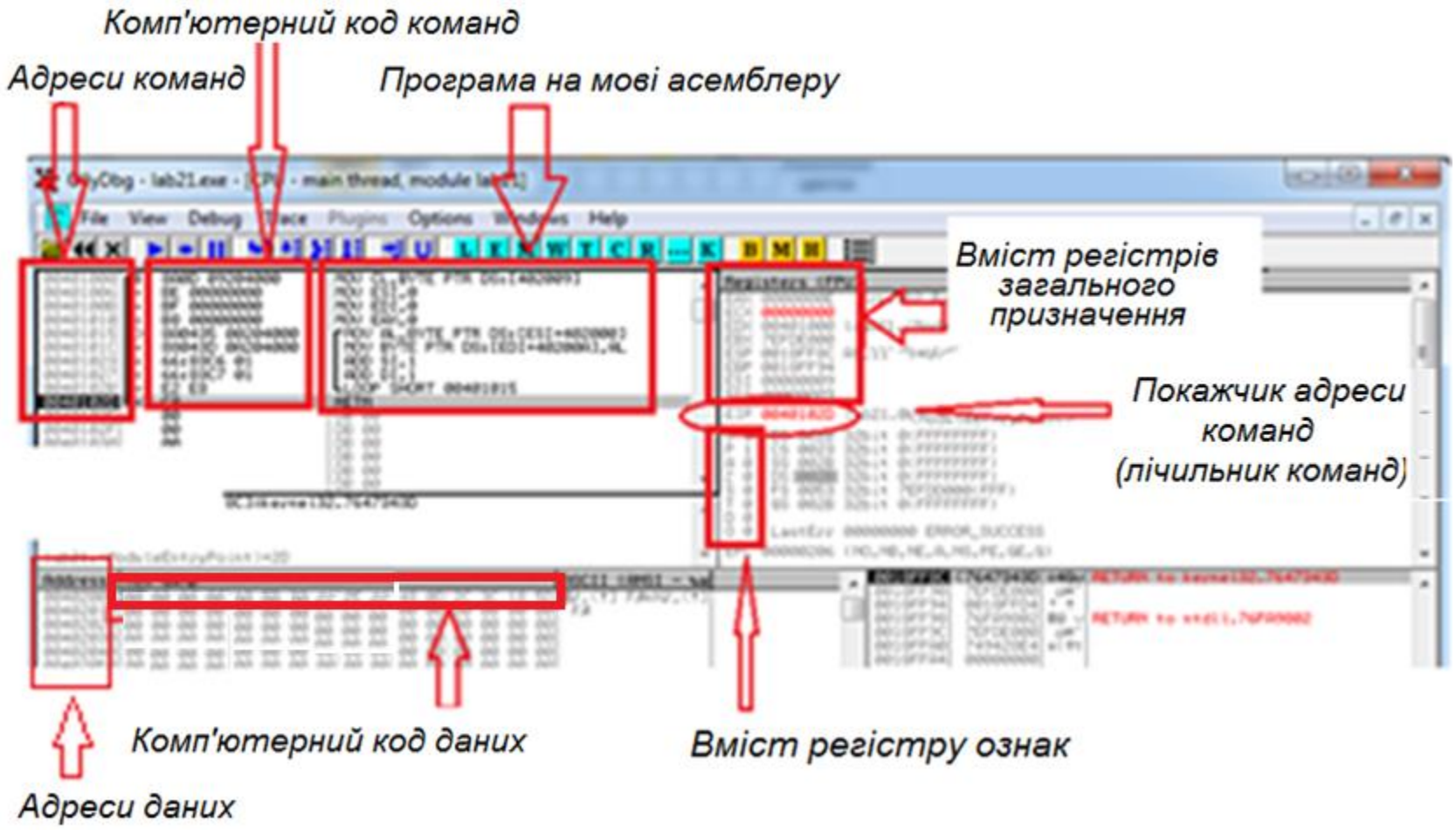
<u>Три</u> ознаки вказують на стан роботи процесору:
- **DF** (direction) - напрям автоадресації джерела/приймача в рядкових (ланцюгових) командах (DF = 0 - автоінкремент, DF = 1 - автодекремент);
- **IF** (interrupt enable) – наявність дозволу на переривання (IF =1 – дозволено);
- **TF** (trap) – наявність дозволу крокового (одна команда) режиму (TF=1 - дозволено).

Регістр адреси команд «покажчик команд» - (EIP)

Покажчик команд (EIP) - 32-розрядний регістр – адреса *наступної* команди на виконання.



Відображення архітектурних елементів в OllyDbg та x32Dbg



Формат команд на мові асемблеру

```
[label:] mnemonic [operands] [;comment]
```

- Мітка (Label) - починається з літери і завершується двокрапкою.
  Приклади - mit18:, ab55:, fah2:

- Мнемокод операції (Instruction mnemonic) – відділяється пробілом.
  Приклади - ADD, MOV, SUB, MUL, JMP

- Операнд/операнди (Operand(s)) – ім’я регістрів, мітки, безпосередні дані, вирази арифметичні, вирази для формування адрес операндів в пам’яті тощо.

  Приклади – eax, 52, 0A4h, [(18+20)/2], [ebx+esi]

- Коментар (Comment) - починається крапкою з комою.

Обов’язковим є вказання на мнемокод операції.

Правила вказання на операнди і результат

В командах на мові асемблеру вказання на послідовність операндів і результату обумовлено за умовчанням:

ADD dst, src

dst (destination) – приймач (приемник (рос))

src (source) – джерело (подавач?) (источник (рос))

Результат завантажується в ***dst***, наприклад, для операції:

- ***ADD***: ***dst = dst + src***
- ***SUB***: ***dst = dst - src***

Така погодженість присутня в більшості команд.

По іншому, наприклад, в рядкових (ланцюгових) командах.

Вимога – однакова розрядність операндів команди

Нотація в командах

Symbol	Description
reg	An 8-, 16-, or 32-bit general register from the following list: AH, AL, BH, BL, CH, CL, DH, DL, AX, BX, CX, DX, SI, DI, BP, SP, EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP.
reg8, reg16, reg32	A general register, identified by its number of bits.
segreg	A 16-bit segment register (CS, DS, ES, SS, FS, GS).
accum	AL, AX, or EAX.
mem	A memory operand, using any of the standard memory-addressing modes.
mem8, mem16, mem32	A memory operand, identified by its number of bits.
shortlabel	A location in the code segment within −128 to +127 bytes of the current location.
nearlabel	A location in the current code segment, identified by a label.
farlabel	A location in an external code segment, identified by a label.
imm	An immediate operand.
imm8, imm16, imm32	An immediate operand, identified by its number of bits.

Команди додавання та віднімання (основні)

ADD (Added) - Додавання

Формат:

```
ADD   reg,reg      ADD   reg,imm
ADD   mem,reg      ADD   mem,imm
ADD   reg,mem      ADD   accum,imm
```

Додає операнд із джерела до операнду із приймача і завантажує результат в приймач.

Розрядності операндів повинні бути однаковими.

A source operand is added to a destination operand, and the sum is stored in the destination.
Operands must be the same size.

Формування ознак:

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

SUB (Subtract) - Віднімання

Формат:

```
SUB   reg,reg      SUB   reg,imm
SUB   mem,reg      SUB   mem,imm
SUB   reg,mem      SUB   accum,imm
```

Віднімає значення операнда джерела із операнда приймача і завантажує результат в приймач.

Розрядності операндів повинні бути однаковими.

Subtracts the source operand from the destination operand.
Operands must be the same size.

Формування ознак:

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

Пересилання (копіювання) даних

MOV (Move) - Пересилання

(Копіювання)

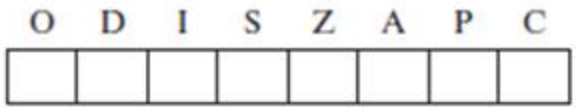
Формат:

```
MOV reg,reg          MOV reg,imm
MOV mem,reg          MOV mem,imm
MOV reg,mem          MOV mem16,segreg
MOV reg16,segreg     MOV segreg,mem16
MOV segreg,reg16
```

Копіювання байт/слово/подвійне слово із джерела в приймач

Copies a byte or word from a source operand to a destination operand.

Формування ознак немає:



Операція комп’ютерної логіки - XOR

XOR — **Exclusive OR**

O	D	I	S	Z	A	P	C
0			*	*	?	*	0

Виключне ЧИ (виключне АБО)

Each bit in the source operand is exclusive ORed with its corresponding bit in the destination. The destination bit is a 1 only when the original source and destination bits are different.

Instruction formats:

```
XOR   reg,reg      XOR   reg,imm
XOR   mem,reg      XOR   mem,imm
XOR   reg,mem      XOR   accum,imm
```

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Використання для обнулення регістрів: XOR *reg, reg*

Приклад (обнулення регістру EAX): `XOR EAX,EAX`

(комп код команди: `33C0` *)*

Альтернативні варіанти обнулення:

```
MOV EAX,0     |  B8 00000000
SUB EAX,EAX   |  2BC0
```

Приклади із 32-біт і 8-біт даними

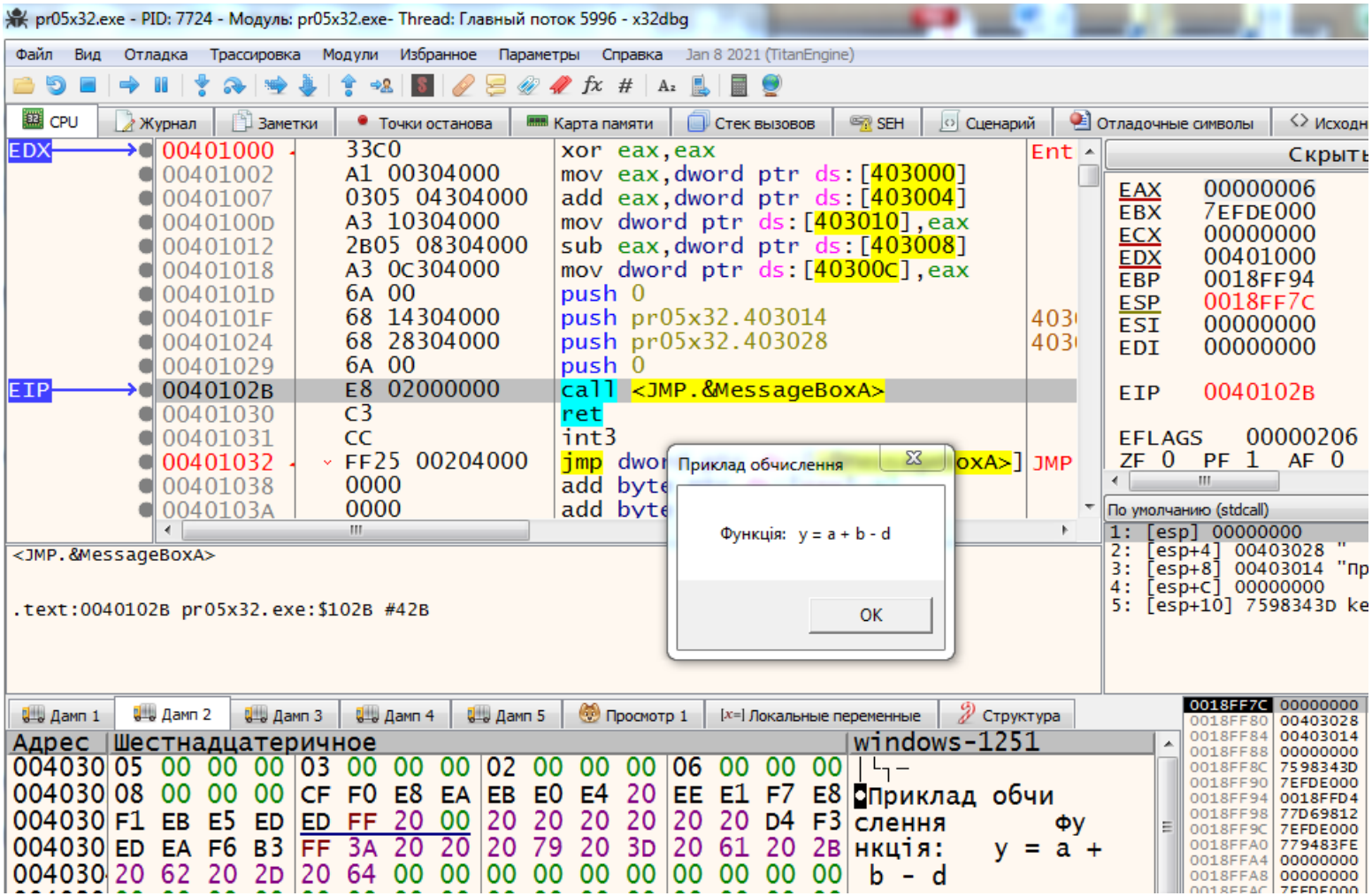
(pr05x32.asm)

```
TITLE < y=a+b-d >
.686
.model flat, stdcall
option casemap: none
.data
   a dd 5
   b dd 3
   d dd 2
   y dd 0Fh
   temp dd ?
.code
sty:
   xor    eax, eax
   mov eax, a
   add eax, b
   mov temp, eax  ; (a+b)
   sub eax, d
   mov y, eax  ; y=(a+b-d)
ret
end sty
```

(pr06x32.asm)

```
TITLE <y=a+b-d>
.686
.model flat, stdcall
option casemap: none
.data
   a db 5
   b db 3
   d db 2
   y db 0Fh
   temp db ?
.code
sty:
  xor    eax, eax
   mov al, a
   add al, b
   mov  temp, al  ; (a+b)
   sub al, d
   mov y, al  ; y=(a+b-d)
ret
end sty
```

Приклад в x32Dbg



Завдання до самостійної роботи (звіт не вимагається)

1. Особливості формату команд різної адресності.
2. Положення з регістрової архітектури команд.
3. Алгоритм виконання арифметичних операцій і операцій пересилання адресності «2А» в архітектурі із регістровою пам’яттю.
4. Архітектура IA-32.
5. Правила вказання на розрядність компонентів IA-32 в командах на мові асемблера.
6. Формат команд на мові асемблера.
7. Редагувати програму, що наведена на попередньому слайді, з іншими значеннями операндів та проаналізувати вміст регістрів і комірок пам’яті.

Література

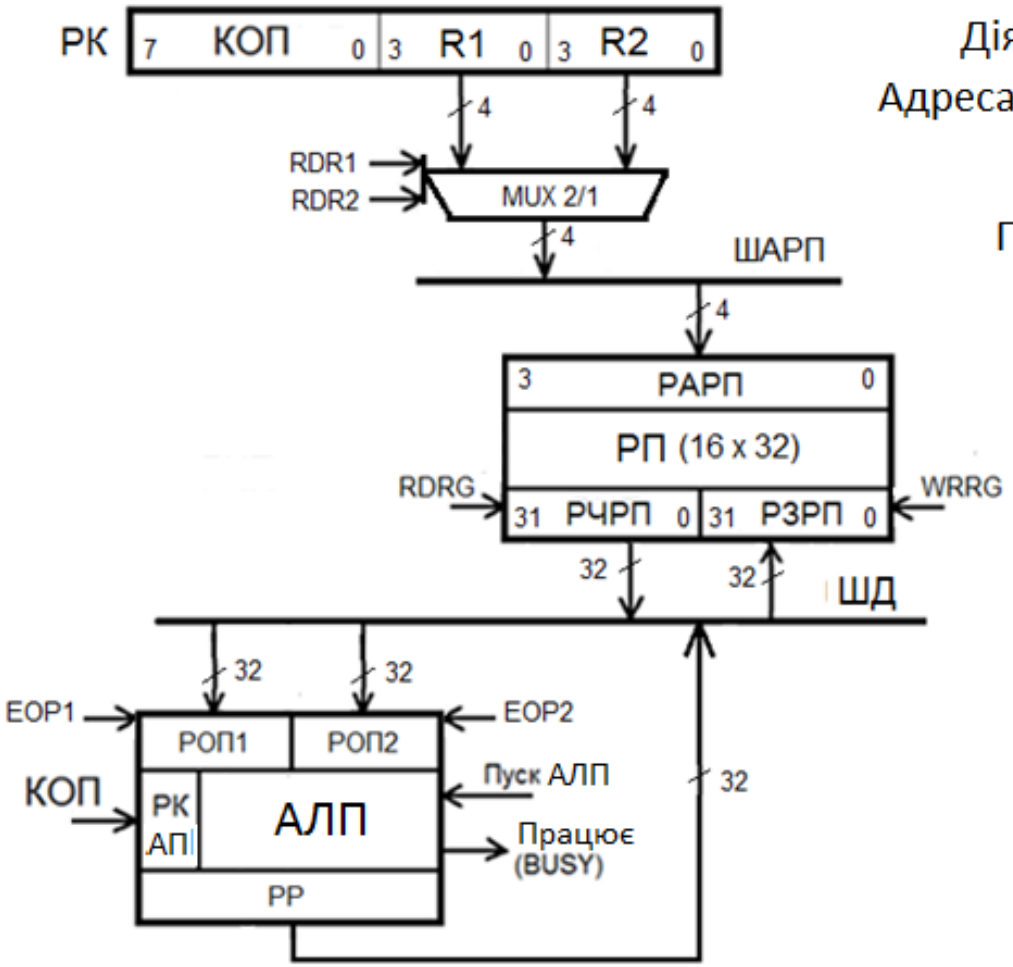
1. Intel® 64 and IA-32 Architectures Software Developer's Manual. - Order Number: 325462-067US, May 2018 (ch.4)
2. Kip R. Irvine. Assembly Language for x86 Processors. Florida International University School of Computing and Information Sciences. 7th Edition, 2014 (ch.2-3,app.A)
3. Andrew S. Tanenbaum, Todd Austin. Structured Computer Organization. – University of Michigan, Ann Arbor, Michigan, United States, 2013 (ch.5)
4. Бабич М.П., Жуков І.А. Комп'ютерна схемотехніка: Навчальний посібник. – К.: «МК-Прес», 2004 (гл.10, с.337-340)

Додаткове завдання до самостійної роботи (не є обов’язковим для виконання)

1. Засвоїти алгоритм виконання арифметичних операцій адресності «2А» із регістровою пам’яттю при форматі команд “RR”, “RS”, “RX” (наступні слайди).
2. Вказати на особливості мікропрограми виконання арифметичних операцій адресності «2А» із регістровою пам’яттю при форматі команд “RR”, “RS”, “RX”.

Barbara J. Burian. A simple approach to S/370 assembly language programming. — New Jersy: Prentice-Hall, Inc, 1977.

Приклад виконання арифметичних операцій адресності «2А» із регістровою пам’яттю
Формат команди “RR”



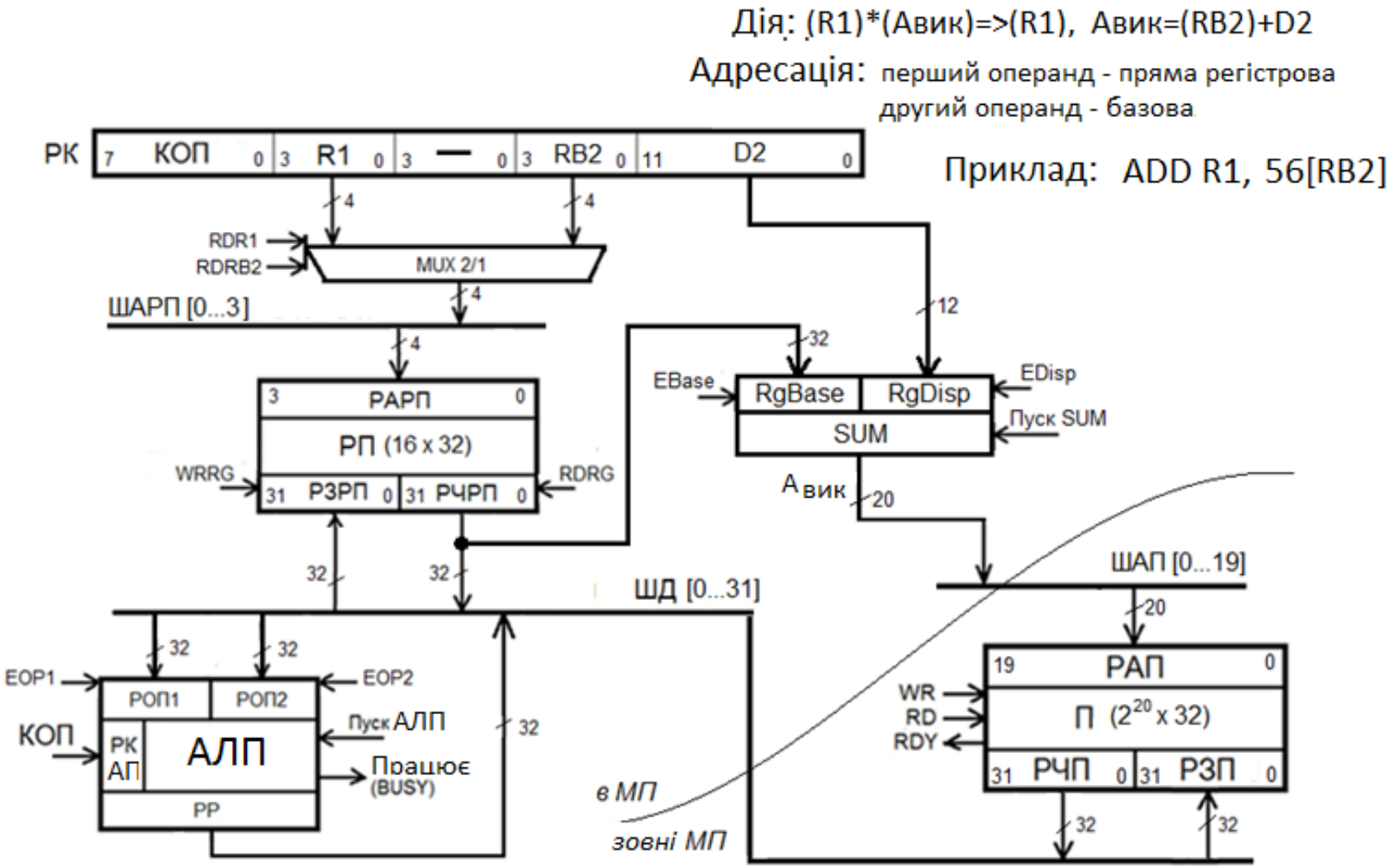
Дія: (R1) * (R2) => (R1)

Адресація: пряма регістрова

Приклад: ADD R1, R2

Приклад виконання арифметичних операцій адресності «2А» із регістровою пам’яттю

Формат команди “RS”



Приклад виконання арифметичних операцій адресності «2А» із регістровою пам’яттю
Формат команди “RX”