سازمان سما

وابسته به دانشگاه آزاد اسلامی

دانشگاه سما واحد حاجی آباد

حل مسئله
معماری کامپیوتر

منبع : معماری کامپیوتر-منوچهر بابایی

WWW.HREZAPOUR.IR

حمیدرضا رضاپور

معماری کامپیوتر

درس ششم:واحدپردازشگر مرکزی

(واحدپردازشگر مرکزی(CPU))

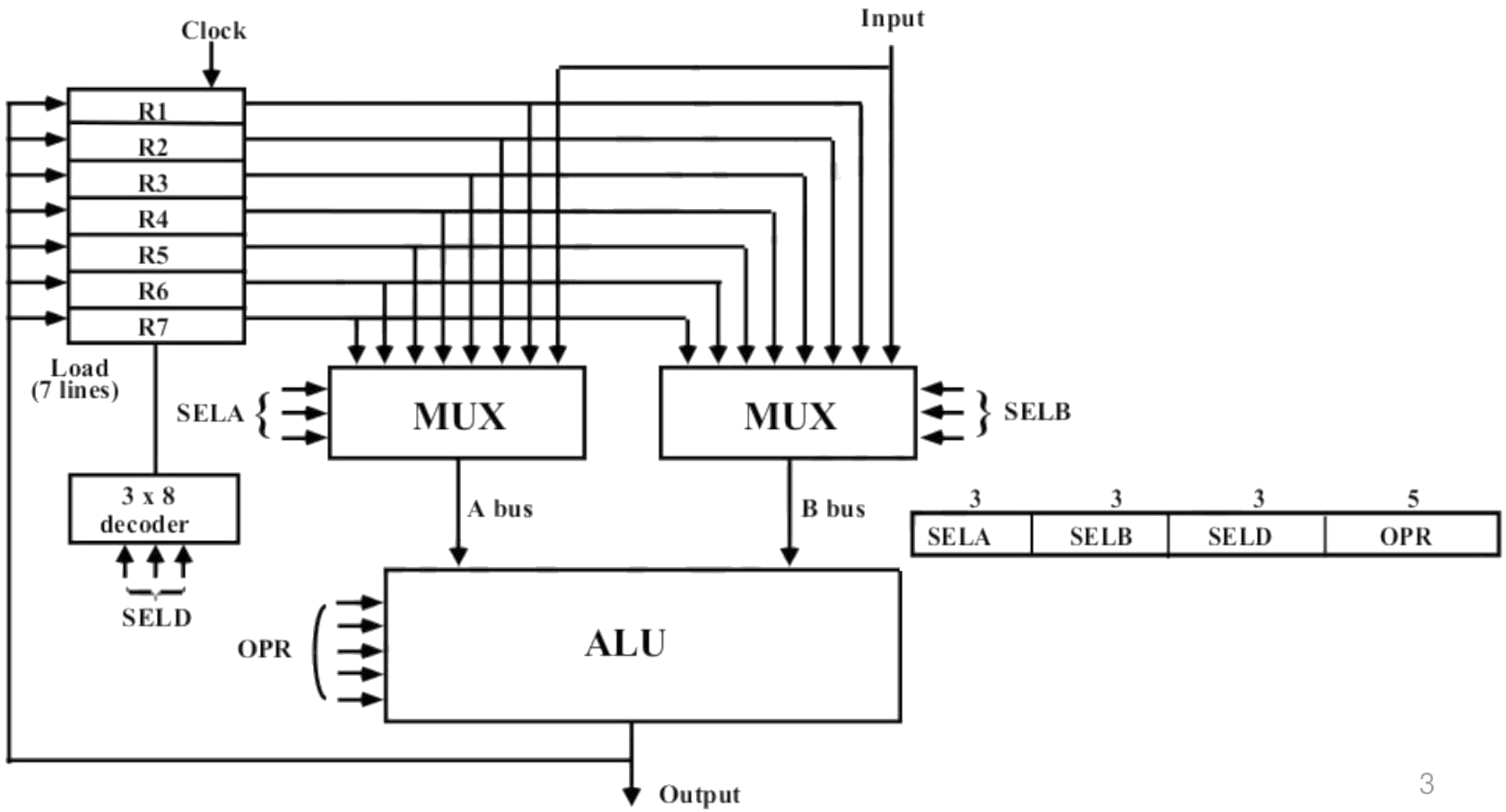
بخشی از کامپیوتر که عمده پردازش داده ها را انجام می دهد واحدپردازشگر مرکزی می باشد.که شامل سه بخش عمده می باشد:

۱-کنترل

۲-مجموعه ثبات ها

۳-واحدمحاسبه ومنطق(ALU)

(واحدپردازشگر مرکزی(CPU))



Microoperation	Symbolic Designation SELA	SELB	SELD	OPR	Control Word
R1 ← R2 - R3	R2	R3	R1	SUB	010 011 001 00101
R4 ← R4 or R5	R4	R5	R4	OR	100 101 100 01010
R6 ← R6 + 1	R6	—	R6	INCA	110 000 110 00001
R7 ← R1	R1	-	R7	TSFA	001 000 111 00000
Output ← R2	R2	-	None	TSFA	010 000 000 00000
Output ← Input	Input	-	None	TSFA	000 000 000 00000
R4 ← shl R4	R4	-	R4	SHLA	100 000 100 11000
R5 ← 0	R5	R5	R5	XOR	101 101 101 01100

مثال (۱):در یک سیستم گذرگاه با ۱۲ ثبات،و ۲۰ عمل برای واحد محاسبه و منطق،کلمه کنترل را مشخص کنید.

$\lceil \log 13 \rceil = 4$ خطوط انتخاب $\rightarrow sel_A = sel_B = 4$

MUX_A , MUX_B

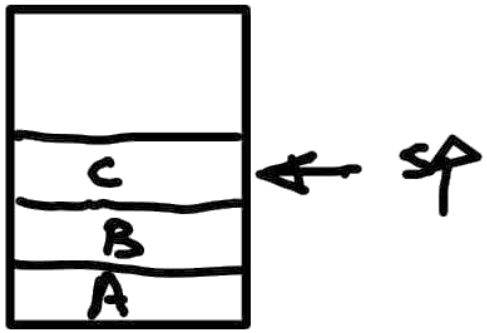
$\lceil \log 20 \rceil = 5$ کدعمل

4	4	4	5
A	sel_B	sel_A	op-code

sel_D

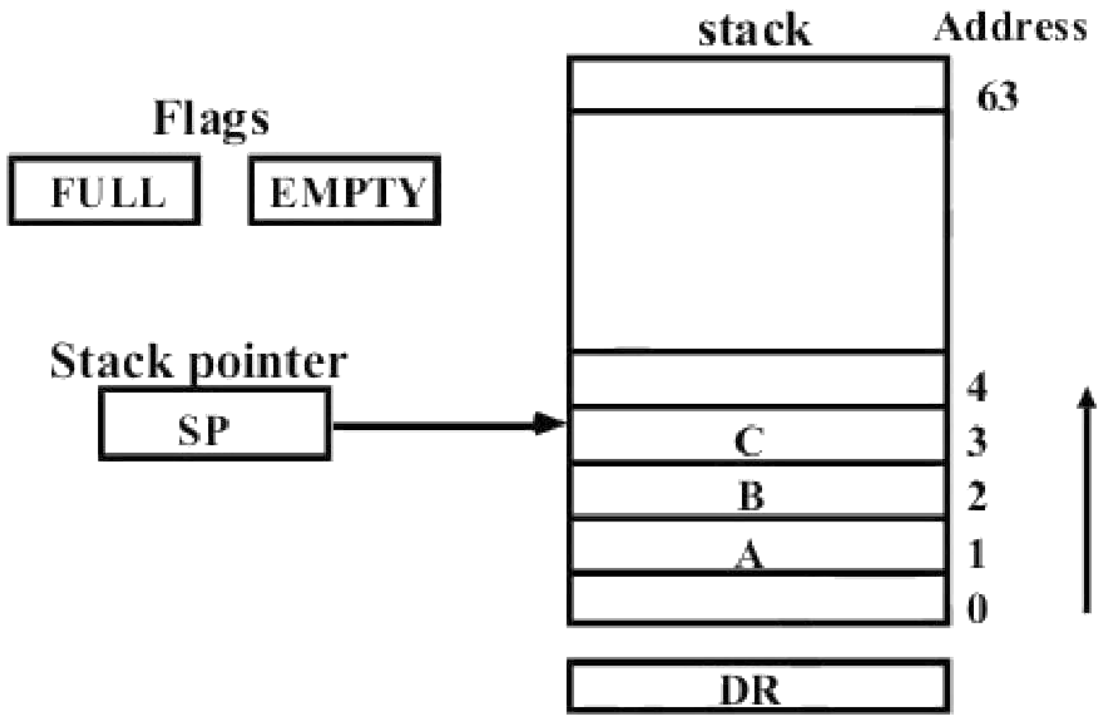
Decoder : $4 \rightarrow 16 \Longrightarrow sel_D = 4$

(سازمان پشته)



Last In first out

- یک ساختمان داده LIFO است.
- برای محاسبه عبارات محاسباتی کاربرد زیادی دارد
- آدرس پشته توسط ثبات SP نگه داشته می شود.
- فقط عملیات درج(PUSH) و حذف(POP) را دارد.


stack
Address
63
Flags
FULL
EMPTY
Stack pointer
SP
4
C
3
B
2
A
1
0
DR

/* Initially, SP = 0, EMPTY = 1, FULL = 0 */

PUSH

SP ← SP + 1

M[SP] ← DR

If (SP = 0) then (FULL ← 1)

EMPTY ← 0

POP

DR ← M[SP]

SP ← SP - 1

If (SP = 0) then (EMPTY ← 1)

FULL ← 0

M[sp] ← DR

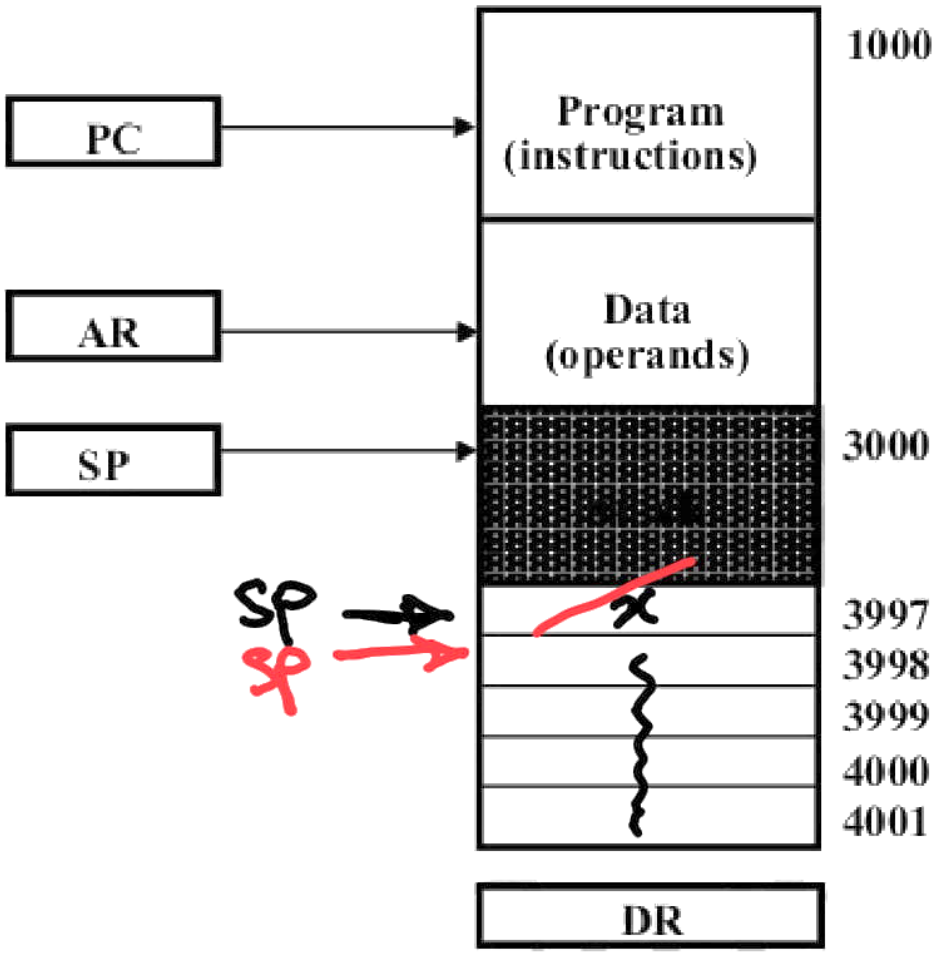
sp ← sp + 1

if (sp = 0) then Full ← 1

sp ← sp - 1

DR ← M[sp]

(پشته حافظه ای)

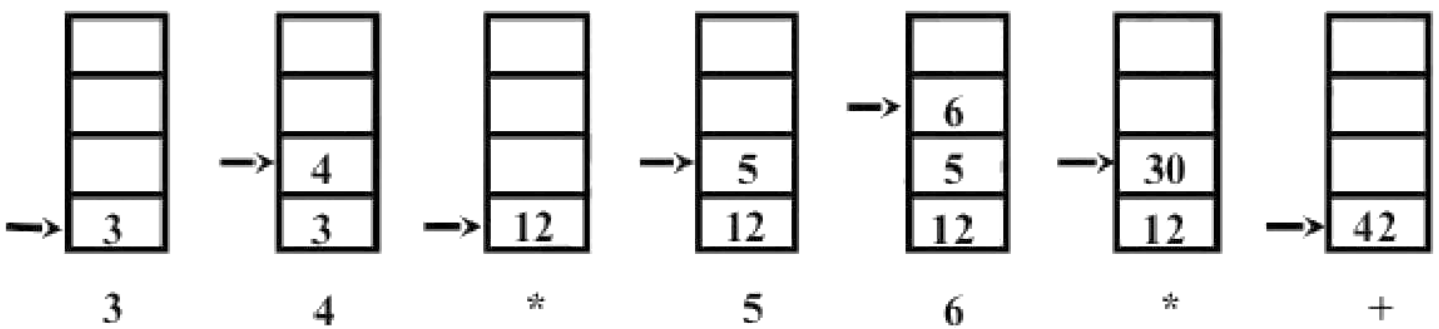


- PUSH: $SP \leftarrow SP - 1$
  $M[SP] \leftarrow DR$
- POP: $DR \leftarrow M[SP]$
  $SP \leftarrow SP + 1$

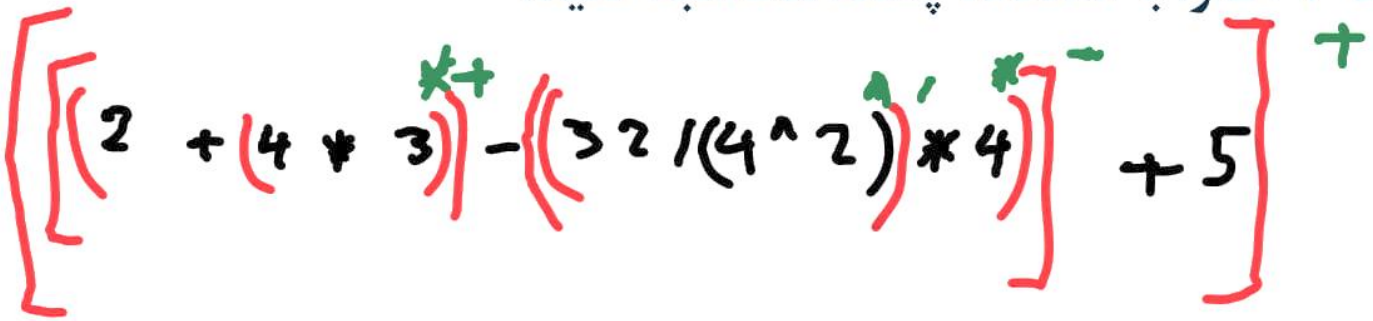
(نمایش لهستانی معکوس عبارات ریاضی)

A + B	Infix notation
+ A B	Prefix or Polish notation
A B +	Postfix or reverse Polish notation

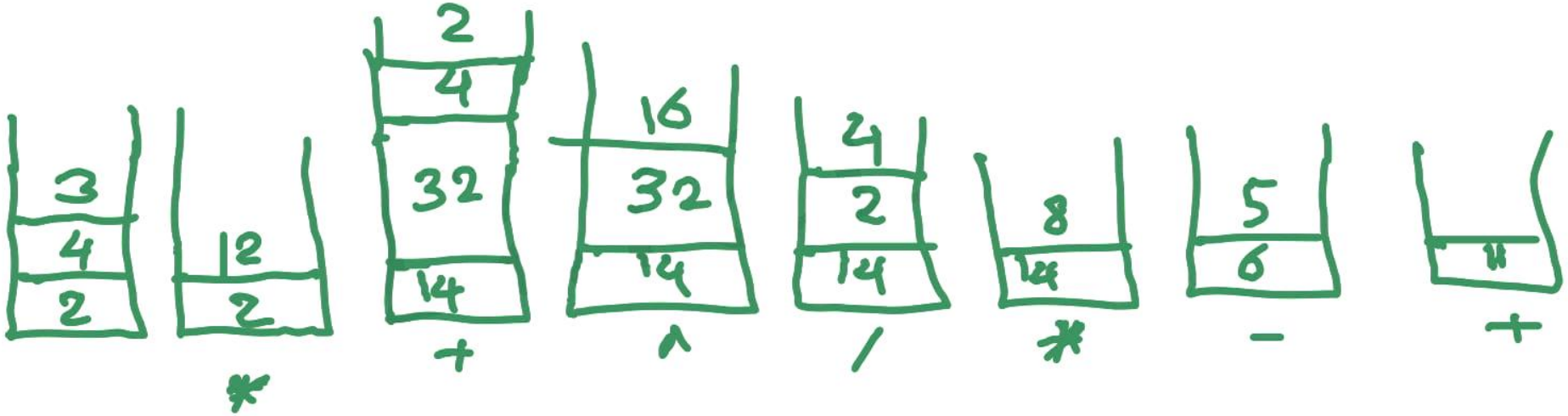
$$((3 * 4) + (5 * 6)) \Rightarrow \quad 3\ 4\ *\ 5\ 6\ *\ +$$



مثال (۲):عبارت 2+4*3-32/4^2*4+5 را با استفاده پشته محاسبه کنید.



2 4 3 * + 3 2 4 2 ^ / 4 * - 5 + : Rp لهستانی معکوس



(قالب دستورالعمل)

قالب دستورالعمل عموما شامل بخش های زیر است:

۱-میدان کد عمل

۲-میدان آدرس

۳-میدان روش آدرس دهی

(انواع پردازنده ها از نظر دستور)

Single accumulator organization: سازمان تک انباره ای

```
ADD     X                    /* AC ← AC + M[X]  */
```

سازمان چند منظوری

General register organization:

```
ADD     R1, R2, R3           /* R1 ← R2 + R3  */
ADD     R1, R2               /* R1 ← R1 + R2  */
MOV     R1, R2               /* R1 ← R2  */
ADD     R1, X                /* R1 ← R1 + M[X]  */
```

Stack organization: سازمان پشته ای

```
PUSH    X                    /* TOS ← M[X]  */
ADD
```

(دستورات از نظر تعداد آدرس)

Three-Address Instructions: دستورات سه آدرسه

Program to evaluate X = (A + B) * (C + D) :

```
ADD   R1, A, B      /* R1 ← M[A] + M[B]   */
ADD   R2, C, D      /* R2 ← M[C] + M[D]   */
MUL   X, R1, R2     /* M[X] ← R1 * R2     */
```

برنامه‌ها کوتاه‌تر شد
دستورالعمل‌ها طولانی

- Results in short programs
- Instruction becomes long (many bits)

Two-Address Instructions:

Program to evaluate X = (A + B) * (C + D) :

```
MOV   R1, A      /* R1 ← M[A]          */
ADD   R1, B      /* R1 ← R1 + M[B]     */
MOV   R2, C      /* R2 ← M[C]          */
ADD   R2, D      /* R2 ← R2 + M[D]     */
MUL   R1, R2     /* R1 ← R1 * R2       */
MOV   X, R1      /* M[X] ← R1          */
```

One-Address Instructions:

- Use an implied AC register for all data manipulation
- Program to evaluate X = (A + B) * (C + D) :

```
LOAD    A    /* AC ← M[A]          */
ADD     B    /* AC ← AC + M[B]     */
STORE   T    /* M[T] ← AC          */
LOAD    C    /* AC ← M[C]          */
ADD     D    /* AC ← AC + M[D]     */
MUL     T    /* AC ← AC * M[T]     */
STORE   X    /* M[X] ← AC          */
```

Zero-Address Instructions:

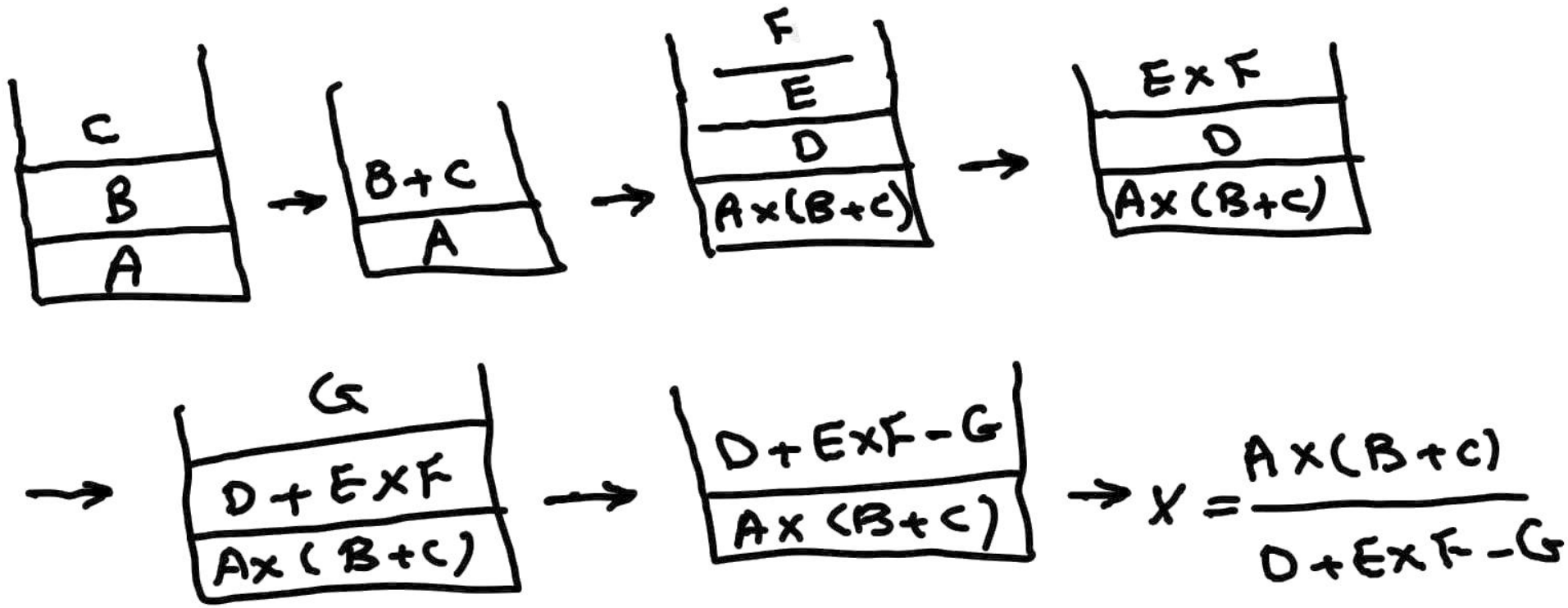
- Can be found in a stack-organized computer
- Program to evaluate X = (A + B) * (C + D) :

AB + CD + *

```
PUSH    A    /* TOS ← A                  */
PUSH    B    /* TOS ← B                  */
ADD          /* TOS ← (A + B)            */
PUSH    C    /* TOS ← C                  */
PUSH    D    /* TOS ← D                  */
ADD          /* TOS ← (C + D)            */
MUL          /* TOS ← (C + D) * (A + B)  */
POP     X    /* M[X] ← TOS               */
```

مثال (۳):دریک ماشین پشته ای،دستورات مقابل چه عبارتی را پیاده می کنند؟

Push A
Push B
Push C
Add
Mul
Push D
Push E
Push F
Mul
Add
Push G
Sub
Div
Pop X



مثال (۴):در یک ماشین سه آدرسه از شیوه های نشانی دهی مستقیم حافظه ای و ثباتی استفاده شده است.حجم حافظه اصلی 2^{20} واحد آدرس پذیر هشت بیتی و طول کلمه برابر چهار واحد آدرس پذیر است.اگر تعداد دستورات یک کلمه ای برابر تعداد دستورات نیم کلمه ای باشد،درآن صورت ماشین دارای چندثبات همه منظوره است؟

طول کلمه : $4 \times 8 = 32$ bit

دستور یک کلمه ای

۱	x	کد ثبات (۲)	کد ثبات (۲)	20

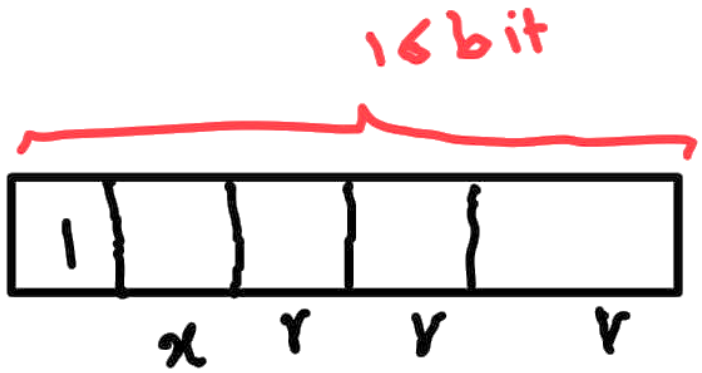
نحوه آدرس دهی — کد عمل — بخش های آدرس

آدرس حافظه : 20 bit

$2r + x + 21 = 32$

$\rightarrow 2r + x = 11$

دستور نیم کلمه ای:



$$3r + x + 1 = 16$$

$$\rightarrow 3r + x = 15$$

$$2r + x = 11 \rightarrow r = 4$$

در نتیجه $2^4 = 16$ بیت همه منظوره داریم.

مثال (۵):کامپیوتری دارای دستورالعمل های ۳۲ بیتی و آدرس های ۱۲ بیتی است.فرض کنید ۲۵۰ دستورالعمل دوآدرسه وجود دارد.حداکثر تعداد دستورالعمل های یک آدرسه چه تعداد می تواند باشد؟

توجه:شیوه نشانی دهی مستقیم است.

→ op-code = 8

→ 2^8 (تعداد دستورات قابل تعریف در فرمت دو آدرسه)

حداکثر تعداد دستورات تک آدرسه: 6×2^{12}

(روشهای آدرس دهی)

- روش انتخاب عملوندهای موردنیاز برای اجرای دستور ،وابسته به نحوه آدرس دهی دستورالعمل متفاوت است.
- مزایای استفاده از روش های آدرس دهی:

۱-انعطاف پذیری زیاد در استفاده از اشاره گرها،شمارنده حلقه های تکرار،اندیس دهی داده هاوتغییرمکان برنامه

۲-تعداد بیت های میدان آدرس دستورالعمل را کاهش می دهد.

روشهای آدرس دهی...

❖ روش ضمنی(Implied Mode)

-در این روش آدرس دهی عملوندها در خود دستور بصورت ضمنی آمده است.

-نیازی به مشخص نمودن آدرس در دستور نیست.

-تمامی دستورات ارجاع به ثبات که از انباره استفاده می کنند از این نوع هستند

-دستورات کامپیوترهای پشته ای(دستورات صفرآدرسه از این نوع هستند.

❖ روش بلافصل یا آنی(Immediate Mode)

-در این روش بجای آدرس،خود عملوند جزیی از دستورالعمل است.

-دستورالعمل های بلافصل برای مقداردهی اولیه ثبات ها با یک مقدار ثابت مناسبند.

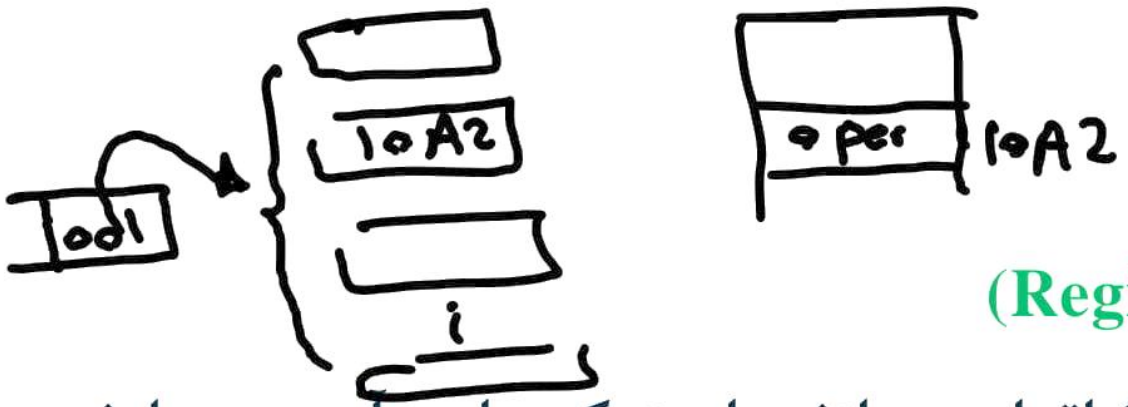
❖ روش ثباتی(Register Mode)

-در این روش،آدرس موجود در دستور آدرس یکی از R ثبات پردازنده است.

-عملوند مورد نیاز در یکی از ثبات های پردازنده است.

-میدان آدرس در این روش کوتاهتر است.

-دسترسی به عملوند در این روش نسبت به حافظه سریعتر است.



❖ روش غیرمستقیم ثباتی(Register Indirect Mode)

-در این روش،آدرس موجود در دستور آدرس یکی از ثباتهای پردازنده است که حاوی آدرس عملوند در حافظه است.

-نسبت به روش مستقیم بیت های کمتری به آدرس اختصاص داده می شود

-دسترسی به عملوند،کندتر است.

❖ روش خودافزایش یا خودکاهش(Auto Increment or Auto Decrement)

-مشابه روش غیرمستقیم ثباتی است با این تفاوت که پس از هربار استفاده از آدرس حافظه مقدار آن در ثبات بطور خودکار افزایش یا کاهش می یابد.

❖ روش مستقیم(Direct Address Mode)

- میدان آدرس دستورالعمل ،آدرس موثر عملوند در حافظه می باشد.
- درصورت بزرگ بودن حافظه اصلی ،بخش بزرگی از دستورالعمل را اشغال می کند.

❖ روش غیرمستقیم(Indirect Address Mode)

- میدان آدرس دستورالعمل ،مکان،آدرس موثر عملوند در حافظه را مشخص می کند.
- درصورت بزرگ بودن حافظه اصلی ،بخش بزرگی از دستورالعمل را اشغال می کند.

مثال (۵):در یک کامپیوتر چهار روش آدرس دهی direct==indirect==register mode==Immediate وجود دارد. اگر این کامپیوتر دارای یک کیلو کلمه ۱۶ بیتی باشد.و هر دستور یک کلمه حافظه باشد.و طول ثبات AC ، هشت بیت باشد،حداکثر تعداد دستورات در این کامپیوتر با ۱۶ ثبات چندتاست؟

	2	6 بیت	8 بیت
Imm:	11	op-code	operand

$\rightarrow 2^6 = 64$ بلافصل

reg	10	op-code	reg
	2	10 بیت	4 بیت

$\rightarrow 2^{10} = 1024$ روش نشانی

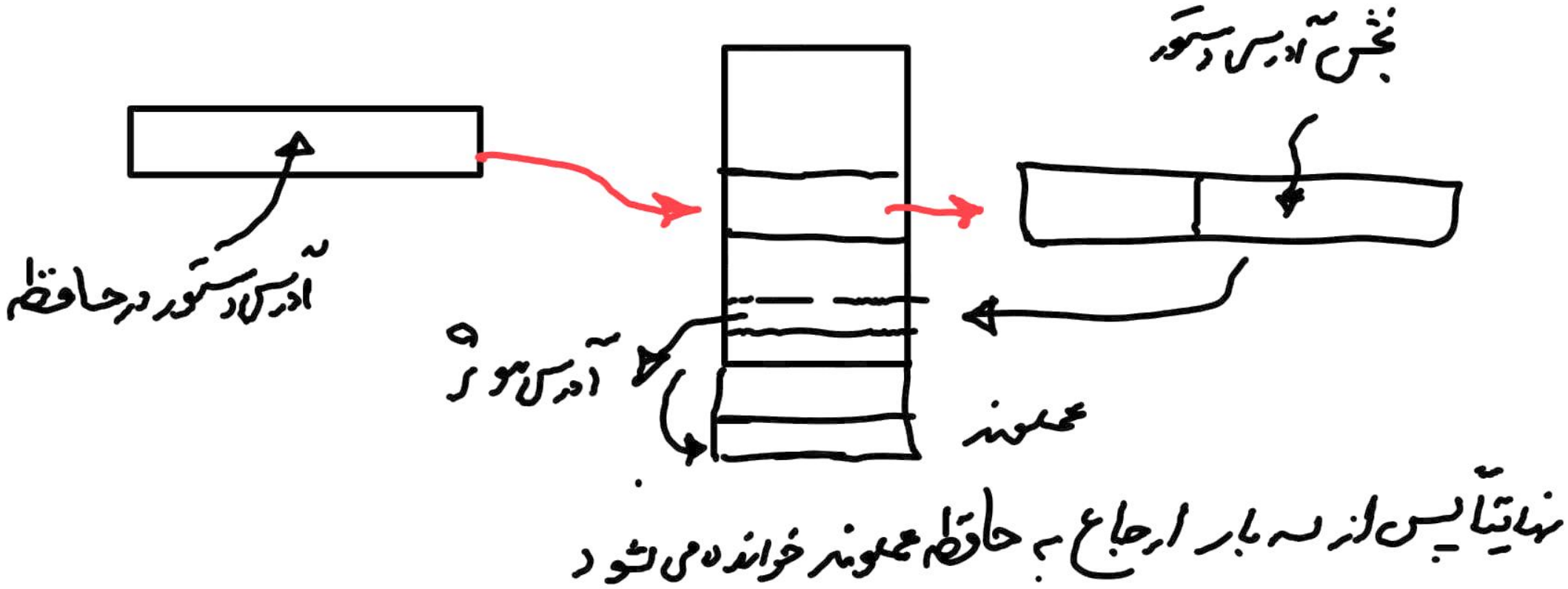
	2	4 بیت	10 بیت
	00	op-code	adr
	01		

$1K = 1024 = 2^{10} \rightarrow adr = 10$ bit

روش مستقیم و غیرمستقیم $= 2^4 + 2^4 = 32$

در مجموع $= 1024 + 64 + 32 = 1120$

مثال (۶):برای دسترسی به یک عملوند،از ترکیب روشهای آدرس دهی غیر مستقیم و غیرمستقیم ثباتی به قسمی که ابتدا با استفاده از روش غیرمستقیم ثباتی آدرس دستور مشخص شده و سپس از روش غیرمستقیم استفاده می شود. در این روش چند دسترسی به حافظه نیاز می باشد؟



مثال (۷): در یک کامپیوتر دستورات دارای ۲ فیلد آدرس شامل دستورات ۲۰بیتی ،۳۰بیتی و ۴۰بیتی می باشند.اگر در این کامپیوتر فقط یک روش آدرس دهی وجود داشته باشد.و تمامی دستورات مراجعه به حافظه باشند،آنگاه کدام گزینه در مورد تعداد دستورات از هر نوع صحیح نیست؟

۱)تعداد دستورات ۲۰بیتی ،۴ تا وتعداد دستورات ۳۰بیتی 4K و تعداد دستورات ۴۰بیتی ،4M

۲) تعداد دستورات ۲۰بیتی ،۲ تا وتعداد دستورات ۳۰بیتی 8K و تعداد دستورات ۴۰بیتی ،8M

۳) تعداد دستورات ۲۰بیتی ،2^x تا وتعداد دستورات ۳۰بیتی 2^{10+x} و تعداد دستورات ۴۰بیتی ،$2^{2(10+x)}$

۴) هیچکدام

نوع دستور

2	کدعمل	بخش آدرس

x

بخش آدرس → $20 - 2 - 2 = 16$

دستورات ۳۰بیتی → 2^{12-x}

کدعمل $= 30 - (20 - x - 2) = 12 - x$ →

مثال (۸): برای انجام عمل $M[X] \leftarrow M[X] + M[Y]$ با استفاده از دستورات تک آدرسه و صفر آدرسه بصورت LOADو STORE به ترتیب چند باراز ثبات انباره(AC) استفاده می شود؟(X و Y آدرسهای حافظه می باشند)

- در دستورات صفر آدرسه فقط از پشته استفاده می‌شود و نیازی به انباره نداریم

- دستورات تک آدرسه:

$$\begin{cases} AC \leftarrow M[X] \\ AC \leftarrow AC + M[Y] \\ M[X] \leftarrow AC \end{cases}$$

مثال (۹):یک ماشین دارای دستورات ۲۰ بیتی است و هر آدرس حافظه ۶ بیتی می‌باشد بعضی از دستورات تک آدرسی و بعضی ۲ آدرسی می‌باشند اگر n دستورالعمل دو آدرسی موجود باشد حداکثر تعداد دستورات ممکن تک آدرسی چندتا است؟(۵ روش آدرس دهی در این ماشین وجود دارد)

$\lceil \log 5 \rceil$

دو آدرسه :

3		6 بیت	6 بیت

$20 - 15 = 5$ کدعمل $\longrightarrow 2^5 = 32$ حداکثر تعداد دستورات دو آدرسه

تعداد دستورات استفاده نشده $= 32 - n$

3			6 بیت

حداکثر دستورات تک آدرسه $\rightarrow (32-n) \times 2^6$

مثال (۱۰):دریک ماشین سه آدرسه ،حجم حافظه اصلی 2^{18} کلمه ۱۸ بیتی(هرکلمه معادل سه واحد آدرس پذیر) است.شیوه آدرس دهی در این ماشین مستقیم(ثباتی و حافظه ای)و بلافصل است.ودستورات ماشین در قالب یک کلمه ای و دو کلمه ای کد می شوند.بافرض اینکه تعداد دستورات یک کلمه ای ۶۰ عدد باشد.با درنظر گرفتن تنوع روشهای آدرس دهی در دستور حداکثر تعداد دستورات دوکلمه ای را بیابید.

کد عمل روش آدرس دهی

2	6	r	r	6

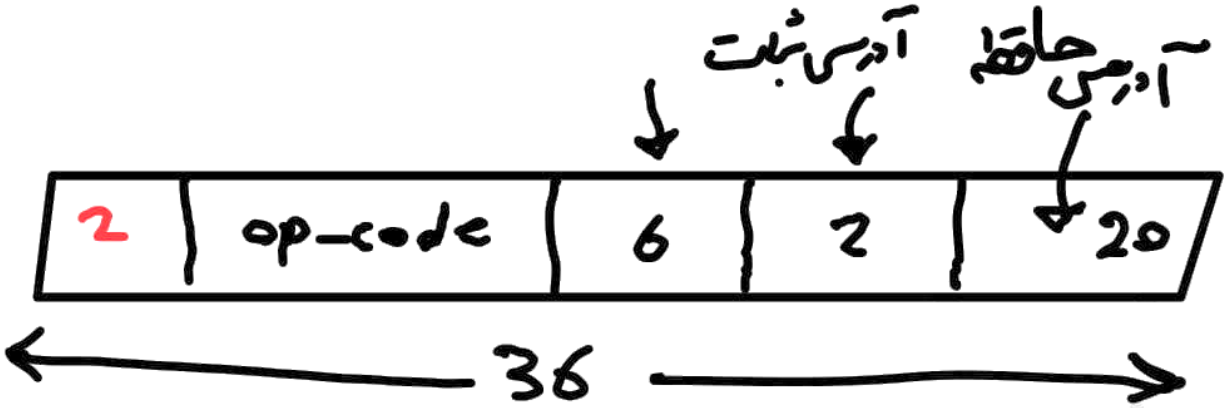
18

$\lceil \log 60 \rceil \longrightarrow$ کد عمل

$\longrightarrow 2r + 14 = 18 \longrightarrow r = 2$

آدرس حافظه : $2^{18} \times 3$ (واحد آدرس پذیر)

$\longrightarrow$ آدرس حافظه : 20 bit



$\rightarrow$ op-code $= 36 - 30 = 6 \rightarrow 2^6 = 64$ دستورالعمل‌های دوکلمه‌ای

❖ روش آدرس دهی نسبی(Relative addressing mode)

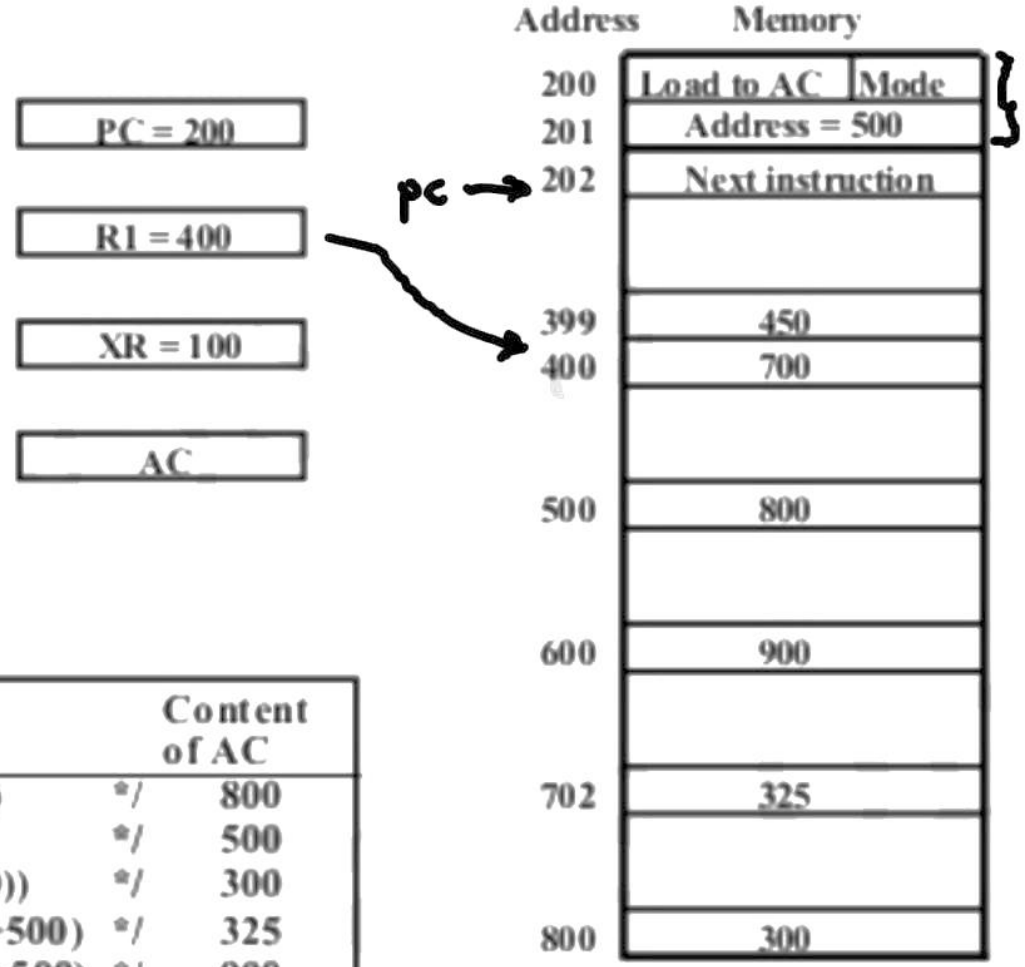
- در این روش شمارنده برنامه به بخش آدرس دستور اضافه می شود،تا آدرس موثر بدست آید.
- فیلد آدرس در این روش کوتاه است.
- با استفاده از این روش میتوان یک حافظه بزرگ را با تعداد بیت کم آدرس دهی کرد.

❖ روش آدرس دهی اندیسی یا شاخص دار (Indexed addressing mode)

- در این روش محتوای یک ثبات شاخص به بخش آدرس دستور اضافه می شود.تا آدرس موثر محاسبه شود.
- ثبات شاخص،یک ثبات خاص از پردازنده است.

❖ روش آدرس دهی باثبات پایه(Base Register addressing mode)

- در این روش محتوای یک ثبات پایه به بخش آدرس دستور اضافه می شود.تا آدرس موثر محاسبه شود.
- مشابه روش قبلی است با این تفاوت که در اینجا ثبات،یک ثبات پایه است.



Addressing Mode	Effective Address			Content of AC
Direct address	500	/* AC ← (500)	*/	800
Immediate operand	-	/* AC ← 500	*/	500
Indirect address	800	/* AC ← ((500))	*/	300
Relative address	702	/* AC ← (PC+500)	*/	325
Indexed address	600	/* AC ← (XR+500)	*/	900
Register	-	/* AC ← R1	*/	400
Register indirect	400	/* AC ← (R1)	*/	700
Autoincrement	400	/* AC ← (R1)+	*/	700
Autodecrement	399	/* AC ← -(R)	*/	450

(دستورالعمل های انتقال داده)

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

مستقیم
غیرمستقیم
نسبی
آنی
شاخص دار
مشارکتی
نمایه غیرمستقیم
خود افزایش
خود کاهش

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC + ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR + XR]
Register	LD R1	AC ← R1
Register indirect	LD (R1)	AC ← M[R1]
Autoincrement	LD (R1)+	AC ← M[R1], R1 ← R1 + 1
Autodecrement	LD -(R1)	R1 ← R1 - 1, AC ← M[R1]

(دستورالعمل های دستکاری داده ها)

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with Carry	ADDC
Subtract with Borrow	SUBB
Negate(2's Complement)	NEG

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI

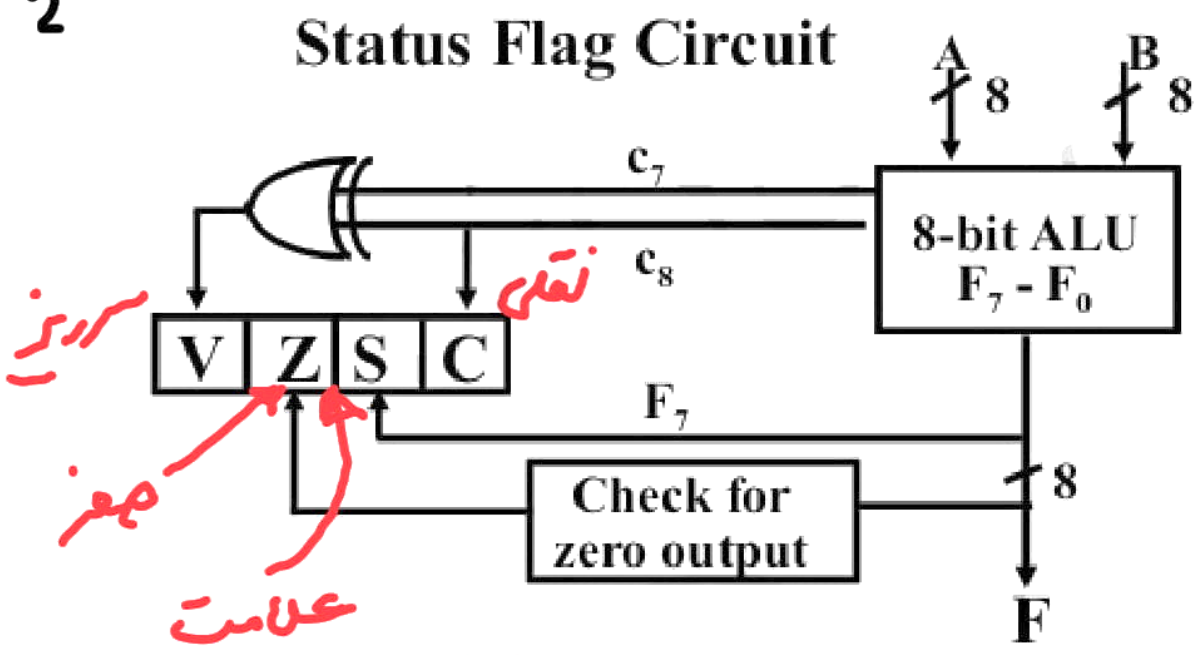
Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right thru carry	RORC
Rotate left thru carry	ROLC

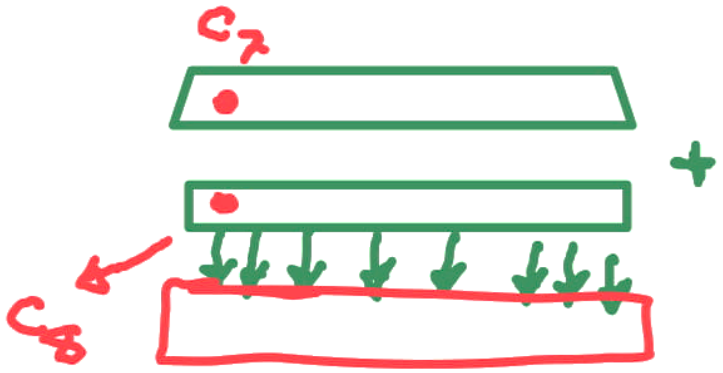
(دستورالعمل های کنترل برنامه)

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RTN
Compare(by -)	CMP
Test (by AND)	TST

* یک عدد هشت بیتی علامتدار در محدوده -128 تا +127 قرار دارد.

$-(2^{n-1}+1) \leq N \leq 2^{n-1}$





$\bar{A}+1 = 00010000 = +16$

$A = 11110000 = -16$

$B = 00101100 = +20$

$\bar{B} = 11010011$

$\bar{B}+1 = 11101100$

$$\begin{array}{r} 11 \\ 11110000 \\ +\ 11101100 \\ \hline 1 \leftarrow 11011100 \end{array}$$

$C=1 \quad V=0 \quad Z=0 \quad S=1$

Mnemonic	Branch condition	Tested condition
BZ	Branch if zero	Z = 1
BNZ	Branch if not zero	Z = 0
BC	Branch if carry	C = 1
BNC	Branch if no carry	C = 0
BP	Branch if plus	S = 0
BM	Branch if minus	S = 1
BV	Branch if overflow	V = 1
BNV	Branch if no overflow	V = 0
Unsigned compare conditions (A - B)		
BHI	Branch if higher	A > B
BHE	Branch if higher or equal	A ≥ B
BLO	Branch if lower	A < B
BLOE	Branch if lower or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B
Signed compare conditions (A - B)		
BGT	Branch if greater than	A > B
BGE	Branch if greater or equal	A ≥ B
BLT	Branch if less than	A < B
BLE	Branch if less or equal	A ≤ B
BE	Branch if equal	A = B
BNE	Branch if not equal	A ≠ B

(فراخوانی و بازگشت از زیرروال)

فراخوانی زیرروال:عبارت است،از یک کدعمل به همراه ادرسی که شروع زیرروال را مشخص می کند.

-آدرس دستورالعمل بعدی(موجود در شمارنده برنامه)در یک مکان موقت ذخیره می شود.

-کنترل به شروع زیرروال منتقل می شود.

آدرس برگشت میتواند به یکی از صورتهای زیر ذخیره شود:

-اولین مکان از حافظه مربوط به زیربرنامه

-یک مکان ثابت در حافظه

-در ثبات های پردازنده

-در حافظه پشته(بهترین راه)

```
CALL
  SP ← SP - 1
  M[SP] ← PC
  PC ← EA

RTN
  PC ← M[SP]
  SP ← SP + 1
```

آدرس شروع زیرروال

(وقفه)

❖ وقفه به منظور برخورد و حل مشکلات ناشی از مسایلی است که خارج از روال معمول برنامه ایجاد می شود. در هنگام وقفه کنترل اجرا مانند زیرروالها از برنامه جاری به برنامه سرویس دهی وقفه منتقل می شود.

تفاوت های رویه وقفه با زیر روالها در فراخوانی:

-وقفه معمولا توسط یک سیگنال داخلی یا خارجی رخ می دهدونه با اجرای یک دستورالعمل

-آدرس سرویس دهی وقفه با سخت افزار مشخص می شودنه بوسیله میدان آدرس دستور

-رویه وقفه معمولا تمام اطلاعات لازم را برای تعیین وضعیت پردازنده در آینده ذخیره می کند

(وقفه...)

وضعیت پردازنده پس ازاجرای برنامه سرویس وقفه با استفاده از موارد زیر مشخص می گردد:

۱-محتوای شمارنده برنامه

۲-محتوای تمامی ثبات های پردازنده

۳-محتوای بعضی از بیت های وضعیت یا PSW

(انواع وقفه)

۱-وقفه خارجی: خارج از پردازنده وحافظه-از وسایل ورودی،خروجی(I/O)-از وسایل زمانبندی-از مدارهای کنترل کننده منبع تغذیه.

عوامل تولید وقفه خارجی:

-وسایل I/O متقاضی انتقال داده

-وسایل I/O اعلام کننده پایان انتقال داده

-انقضای زمان یک رخداد

-از کار افتادن منبع تغذیه

-Time out

۲-وقفه داخلی: از اجرای برنامه جاری ناشی می شوند.-اجرای غیرمجاز یا اشتباه یک دستور-تله یا دام

عوامل تولید وقفه داخلی:

-سرریزثبات

-تقسیم بر صفر

-کدعمل غیرمعتبر

-سرریزپشته

-نادیده گرفتن مقررات محافظتی

❖ وقفه های داخلی با برنامه همگام هستند در حالیکه وقفه های خارجی با برنامه ناهمگام هستند،ومستقل از اجرای برنامه هستند.

۳-وقفه نرم افزاری: وقفه با اجرای یک یا چند دستور-برخلاف وقفه های دیگر با سیگنال های سخت افزاری پردازنده اتفاق نمی افتند.

انتقال کنترل اجرای پردازنده از حالت کاربر به حالت ناظر-اجرای دستورالعمل های ورودی-خروجی پیچیده که در حالت کاربر امکانپذیر نیست.

(کامپیوترهای CISC-RISC)

مشخصه های CISC:

۱-وجود دستورات زیاد –ساده کردن کامپایلر

۲-انواع متنوعی از روش های آدرس دهی

۳-قالب دستورات با طول متغیر

۴-دستوراتی که عملوندها را در حافظه دستکاری می کنند.

۵-دستورالعمل های که کارایی دارند اما بندرت استفاده می شوند.

۶-وجود مدارهای سخت افزاری زیاد بدلیل وجود روش های آدرس دهی و کاهش سرعت محاسبات

مشخصه های RISC:

۱-بیشتر دستورات در یک پالس ساعت اجرا می شوند.

۲-دستیابی به حافظه منحصر است به دستورات Load وStore

۳-بیشتر عملوندها در ثبات ها هستند تا در حافظه اصلی

۴-دستورات کوتاهتری دارند

۵-تعداد ثبات های بیشتری نیاز دارد

۶-دستورات و روش های ادرس دهی نسبتا کم

۷-دستورات با طول ثابت

۸-استفاده از خط لوله دستورالعمل

۹-عملیات ثبات به ثبات-ذخیره در حافظه

۱۰-کنترل سخت افزاری بجای ریزبرنامه

(پنجره های ثباتی)

هنگامی استفاده از یک رویه ،در زمان ترجمه به زبان ماشین:

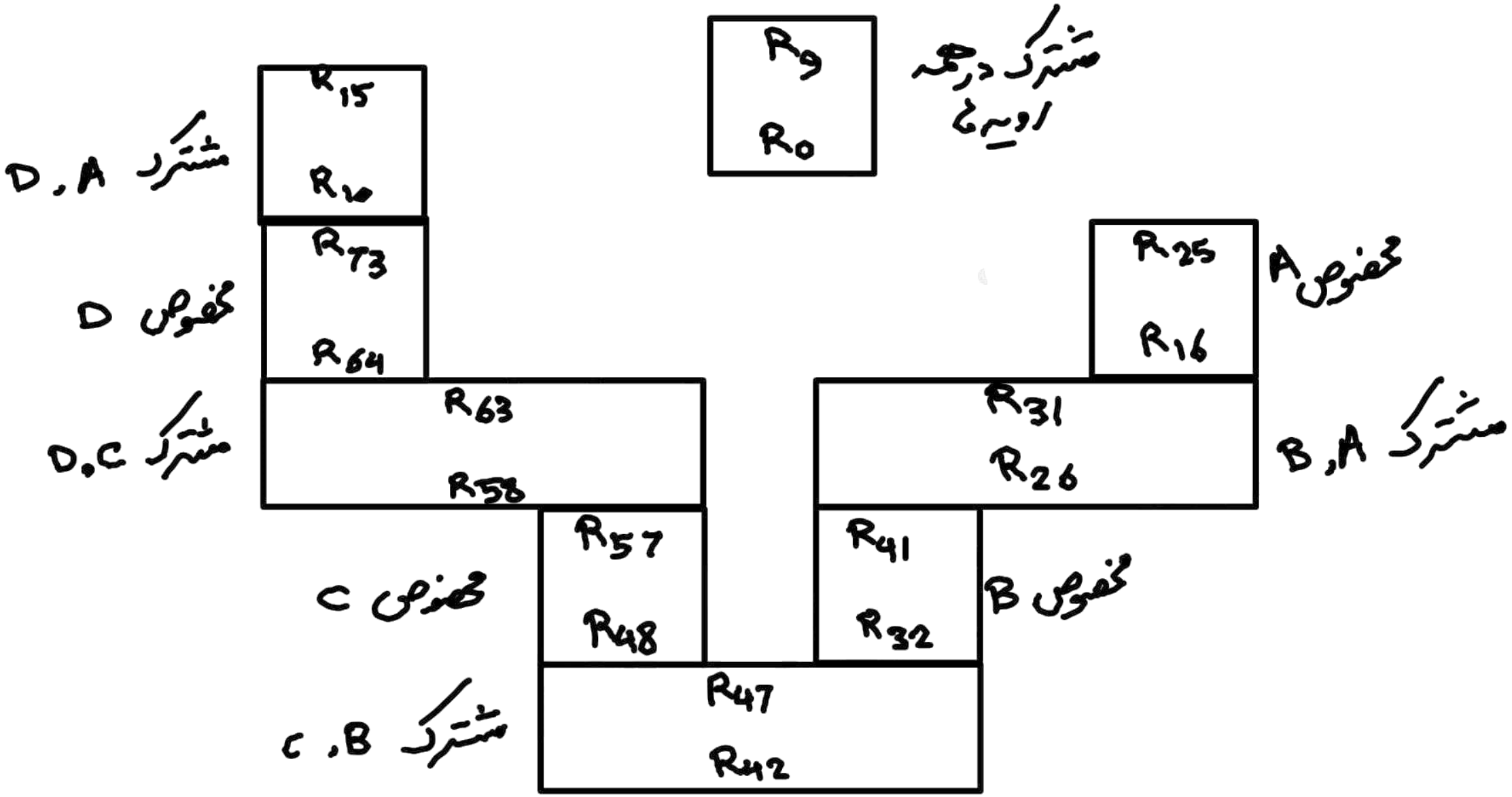
۱-مقادیر ثبات های مورد نیاز ذخیره می شوند

۲-پارامترهای لازم برای اجرای رویه تحویل داده می شوند.

۳-پس از بازگشت از رویه مقادیر قبلی ثبات ها بازیابی می شوندو نتایج به برنامه فراخواننده تحویل می شوند

۴-ذخیره و بازیابی ثباتها و تحویل نتایج یک کار زمانبر است

دربرخی از پردازنده های RISC برای اجتناب از ذخیره و بازیابی مقادیرثبات ها از پنجره ثباتی استفاده می شود.


R9
R0
مشترک در همه اویه ها
R15
R10
مشترک D , A
R73
R64
مخصوص D
R25
R16
مخصوص A
R63
R58
مشترک D , C
R31
R26
مشترک B , A
R57
R48
مخصوص C
R41
R32
مخصوص B
R47
R42
مشترک B , C

بطور کلی تعداد ثبات های قابل دسترس توسط هر پنجره بصورت زیر محاسبه می شود:

تعداد ثبات های مخصوص هر پنجره+تعداد ثبات های مشترک در تمامی پنجره ها +دوبرابر ثبات های مشترک بین دوپنجره مجاور هم

تعدادثبات های مورد نیاز در پردازنده :

(تعداد ثبات های مخصوص هر پنجره+تعداد ثبات های مشترک بین دو پنجره مجاور)*تعداد پنجره ها+تعداد ثبات های مشترک در تمامی پنجره ها

مثال (۱۱): در یک پردازنده RISC که از روش پنجره ثبات(Register Window) استفاده می کند،۸ ثبات سراسری و ۸ ثبات مشترک بین هردو پنجره مجاور وجود دارد.اگر مجموع ثبات های پردازنده ۱۲۰ عدد باشدوهر پنجره نیز ۸ ثبات محلی داشته باشد.تعداد پنجره های ثبات در این پردازنده چیست؟

$G = 8$

$C = 8$

$L = 8$

$$120 = (L + C)n + G$$

$$(8+8)n + 8 = 120 \longrightarrow n = \frac{112}{16} = 7$$