معماری کامپیوتر
سازمان و طراحی یک کامپیوتر پایه

فصل پنجم کتاب معماری کامپیوتر موریس مانو

محمّدعلی شفیعیان

http://shafieian-education.ir

سازمان و طراحی کامپیوتر پایه

- ✓ در این فصل:
  - ✓ کدهای دستورالعمل
  - ✓ ثبات های کامپیوتر
  - ✓ دستورات کامپیوتر
  - ✓ زمان بندی و کنترل
  - ✓ چرخه دستورالعمل
  - ✓ دستورات مراجعه به حافظه
  - ✓ وقفه و ورودی-خروجی

2

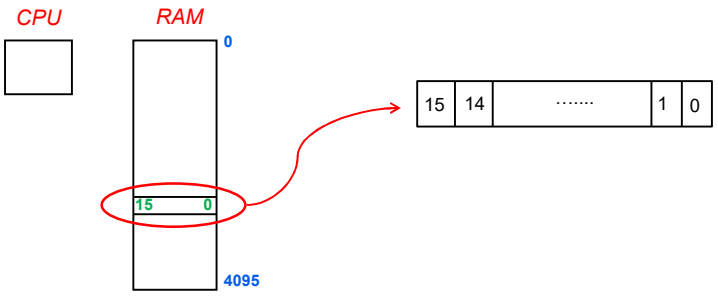
مقدمه

- ✓ هر پردازنده طراحی خاص خود(ثبات ها، گذرگاه ها، ریزعمل ها، دستورالعمل های ماشین و ...) را دارد.
- ✓ کامپیوترهای مدرن ساختار پیچیده ای دارند که شامل موارد زیر هستند:
  - ✓ ثبات های فراوان
  - ✓ چندین واحدهای محاسباتی هم برای اعداد صحیح هم برای اعداد ممیزدار
  - ✓ استفاده از چندین واحد خط لوله تا به این ترتیب سرعت اجرا افزایش یابد.
  - ✓ و موارد دیگر
- ✓ در ادامه برای فهم اینکه کامپیوتر چگونه کار می کند از یک مدل ساده شده استفاده شده است. این مدل را آقای مانو(*Mano*) معرفی کرده و نام آن را کامپیوتر پایه گزارده است. این مدل شبیه کامپیوترهایی است که ۳۰ سال پیش کار می کرده اند.
- ✓ ما از این مدل برای معرفی سازمان پردازنده و ارتباط *RTL* با سطح بالاتر پردازنده استفاده می کنیم.

3

کامپیوتر پایه

- ✓ کامپیوتر پایه دو جز (*component*) اصلی دارد: پردازنده و حافظه.
- ✓ حافظه ۴۰۹۶ کلمه دارد.
- ✓ $۴۰۹۶ = ۲^{۱۲}$، یعنی به ۱۲ خط آدرس نیاز داریم.
- ✓ هر کلمه ۱۶ بیت طول دارد.



4

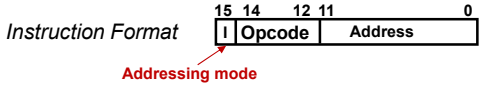
دستورالعمل ها

- ✓ برنامه
  - ✓ یک دنباله از دستورالعمل ها
- ✓ دستورالعمل
  - ✓ یک گروه از بیت ها که به کامپیوتر اعلام می کنند که یک عمل خاص را انجام دهند (یک دنباله از ریزعمل ها).
- ✓ دستورالعمل های یک کامپیوتر به همراه همه داده های لازم در حافظه ذخیره شده اند.
- ✓ CPU دستور بعدی را از حافظه می خواند.
- ✓ این دستور در یک ثبات به نام ثبات دستورالعمل (IR) ذخیره شده است.
- ✓ دستورالعمل به دنباله ای از ریزعمل ها تبدیل می شود تا با انجام ریزعمل ها دستورالعمل مورد نظر اجرا شود.

5

فرمت دستوالعمل ها

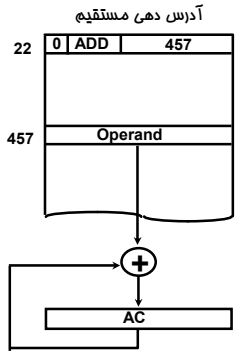
- ✓ یک دستورالعمل اغلب از دو بخش تشکیل شده است:
  - ✓ کد عملیات (*opcode*): عملی را که دستورالعمل باید انجام دهد، مشخص می کند.
  - ✓ آدرس (*address*): ثبات یا جایی از حافظه را که دستورالعمل باید عمل کند مشخص می کند.
- ✓ همانطور که گفتیم در کامپیوتر پایه ۱۲ بیت برای آدرس دهی حافظه داریم.
- ✓ در کامپیوتر پایه بیت ۱۵ دستورالعمل مد آدرس دهی (*Addressing Mode*) را مشخص می کند.
  - ✓ صفر: آدرس دهی مستقیم (*Direct Addressing*)
  - ✓ یک: آدرس دهی غیر مستقیم (*Indirect Addressing*)
- ✓ چون کلمه های حافظه و بنابراین دستورالعمل ها ۱۶ بیتی هستند، ۳ بیت باقی مانده برای کد دستورالعمل مورد استفاده قرار می گیرد.

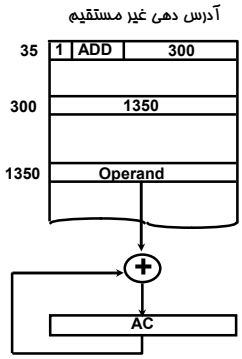


6

حالت (مد) آدرس دهی (*Addressing Mode*)

- ✓ نامیه آدرس یک دستورالعمل می تواند به یکی از دو شکل زیر تفسیر شود:
  - ✓ آدرس دهی مستقیم: آدرس داده مورد نظر در حافظه (عملوند)
  - ✓ آدرس دهی غیر مستقیم: آدرس آدرس داده مورد نظر در حافظه (عملوند)





- ✓ آدرس مؤثر (*Effective Address (EA)*)
  - ✓ آدرس نهایی برای یافتن عملوند، مثلا در شکل بالا سمت راست آدرس موثر ۱۳۵۰و در شکل بالا سمت چپ آدرس موثر ۴۵۷ است.

7

ثبات های پردازنده (۱)

- ✓ یک پردازنده تعداد زیادی ثبات برای نگهداری دستورالعمل ها، آدرس ها و داده ها و ... دارد.
- ✓ پردازنده یک ثبات به نام شمارنده برنامه (*Program Counter (PC)*) دارد که آدرس دستوری را که باید اجرا شود، نگه می دارد.
  - ✓ چون حافظه در کامپیوتر پایه ۴۰۹۶ کلمه دارد پس *PC* ۱۲ بیتی است.
- ✓ در آدرس دهی مستقیم یا غیر مستقیم پردازنده برای آنکه آدرس عملوند را نگه دارد از یک ثبات به نام ثبات آدرس (*Address register (AR)*) استفاده می کند.
  - ✓ چون حافظه در کامپیوتر پایه ۴۰۹۶ کلمه دارد پس *AR* ۱۲ بیتی است.
- ✓ پس از آنکه عملوند در حافظه پیدا شد، در آدرس دهی مستقیم یا غیر مستقیم، عملوند به یک ثبات به نام ثبات داده (*Data Register (DR)*) منتقل می شود.

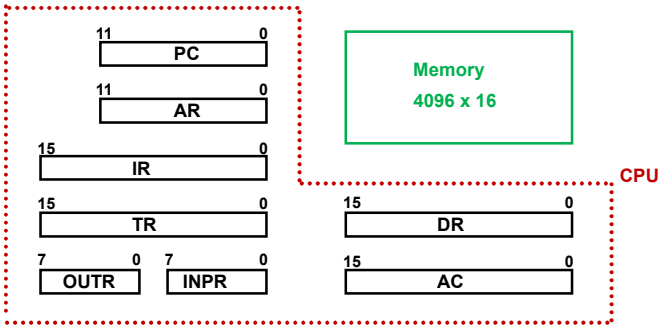
8

ثبات های پردازنده (۲)

- ✓ کامپیوتر پایه یک ثبات همه منظوره به نام انباره (*Accumulator (AC)*) نیز دارد.
- ✓ در کامپیوتر پایه از یک ثبات برای نگهداری داده های میانی یا موقتی استفاده شده است به این ثبات، ثبات موقت (*Temporary Register (TR)*) می‌گویند.
- ✓ کامپیوتر پایه یک مدل بسیار ساده ورودی/خروجی دارد.
  - ✓ دستگاه های ورودی کاراکترهای ۸ بیتی را به پردازنده می فرستند.
  - ✓ پردازنده کاراکترهای ۸ بیتی را به دستگاه های خروجی می فرستد.
- ✓ ثبات ورودی (*Input Register (INPR)*) داده ۸ بیتی را که از دستگاه ورودی رسیده است نگه می دارد.
- ✓ ثبات خروجی(*Output Register (OUTR)*) داده ۸ بیتی را که به دستگاه خروجی فرستاده می شود نگه می دارد.

9

ثبات های کامپیوتر پایه



List of BC Registers

DR	16	Data Register	مقدار عملوند را نگه می دارد
AR	12	Address Register	آدرس عملوند را نگه می دارد
AC	16	Accumulator	ثبات همه منظوره
IR	16	Instruction Register	کد عملیات را نگه می دارد
PC	12	Program Counter	آدرس دستورالعمل را نگه می دارد
TR	16	Temporary Register	داده های موقتی را نگه می دارد
INPR	8	Input Register	کاراکتر ورودی را نگه می دارد
OUTR	8	Output Register	کاراکتر خروجی را نگه می دارد

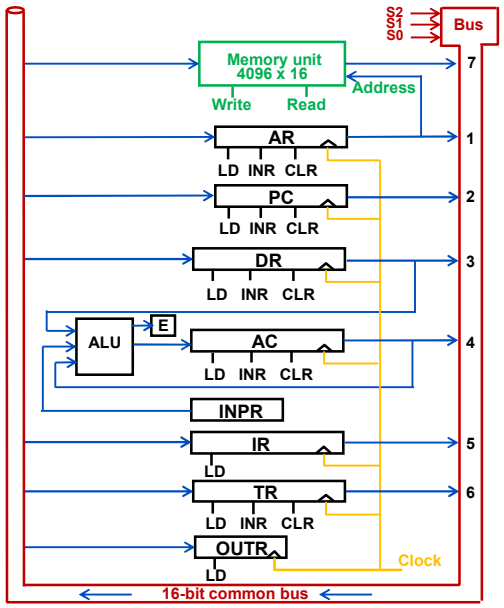
10

سیستم گذرگاه عمومی (۱)

- ✓ ثبات ها در کامپیوتر پایه با استفاده از گذرگاه به هم متصل شده اند
- ✓استفاده از گذرگاه نسبت به اتصال مستقیم ثبات ها به هم در سیم‌بندی صرفه جویی می کند.

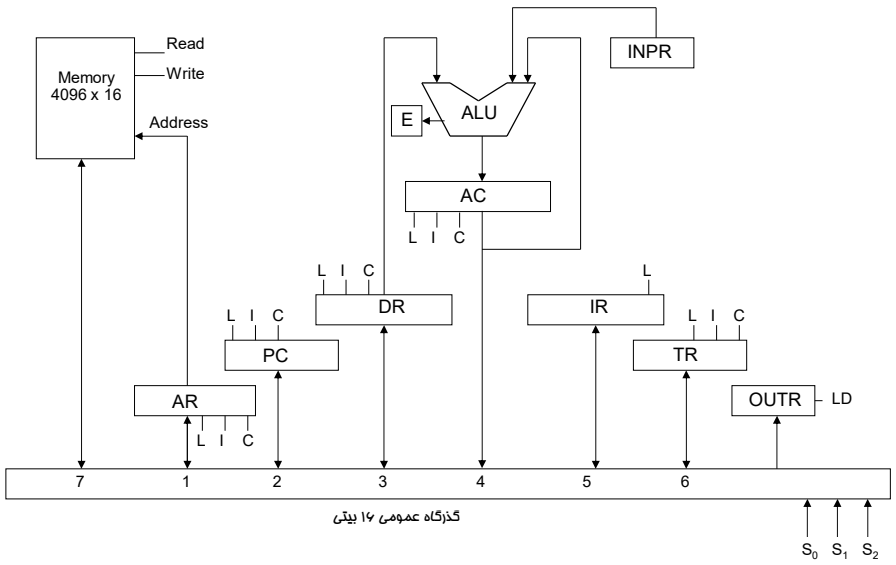
11

سیستم گذرگاه عمومی (۲)



12

سیستم گذرگاه عمومی (۳)



13

سیستم گذرگاه عمومی (۵)

- ✓ سه خط کنترل s_0، s_1 و s_2 کنترل می کنند که کدام ثبات به عنوان ثبات ورودی انتخاب شود.

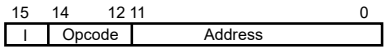
S_2 S_1 S_0	Register
0 0 0	x
0 0 1	AR
0 1 0	PC
0 1 1	DR
1 0 0	AC
1 0 1	IR
1 1 0	TR
1 1 1	Memory

- ✓ برای خروجی : پایه *Read* حافظه یا پایه *Load* ثبات ها فعال می شود.
- ✓ وقتی ثبات های ۱۲ بیتی (AR و PC) روی گذرگاه اطلاعات قرار می دهند، ۴ بیت با ارزش تر گذرگاه مقدار صفر می گیرد.

14

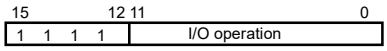
دستوالعمل های کامپیوتر پایه

فرمت دستوالعمل های کامپیوتر پایه



دستورالعمل های مراجعه به حافظه
(OP-code = 000 ~ 110)

دستورالعمل های مراجعه به ثبات
(OP-code = 111, I = 0)



دستورالعمل های ورودی خروجی
(OP-code =111, I = 1)

15

دستوالعمل های کامپیوتر پایه

سمبل	Hex Code I = 0	Hex Code I = 1	توضیح
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

16

کامل بودن مجموعهٔ دستورالعملها

- ✓ هر کامپیوتر باید مجموعه‌ای از دستورالعملها را داشته باشد که کاربر بتواند برای محاسبه هر تابعی که محاسبه پذیر بودن آن محرز است، برنامه‌ای به زبان ماشین بنویسد.

- ✓ نوع دستورالعملها
  - ✓ دستورالعمل های عملیاتی
    - ✓ حسابی، منطقی و جابجایی مانند: ADD, CMA, INC, CIR, CIL, AND, CLA
  - ✓ تبادل اطلاعات
    - ✓ تبادل اطلاعا ت با حافظه وثبات های کامپیوتر مانند: LDA, STA
  - ✓ دستورالعمل های کنترلی
    - ✓ دستورالعملهای کنترل برنامه و وارسی شرایط مانند: BUN, BSA, ISZ
  - ✓ دستورالعمل های ورودی خروجی
    - ✓ ورودی خروجی مانند: INP, OUT

17

واحد کنترل

- ✓ واحد کنترل همهٔ دستورالعمل های ماشین را به سیگنال های کنترلی تبدیل می‌کند. این سیگنالهای کنترلی برای کنترل ریزعمل ها بکار می‌روند.

- ✓ واحد کنترل به دو طریق قابل ساخت می‌باشد:
  - ✓ کنترل سخت‌افزاری (*Hardwired*)

واحد کنترل از مدارهای ترکیبی و ترتیبی ساخته شده است که کار آنها تولید سیگنالهای کنترلی است.

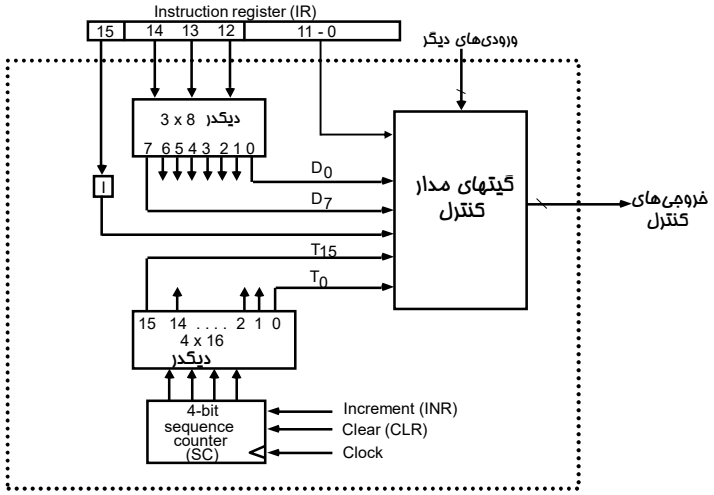
  - ✓ کنترل ریزبرنامه‌نویسی شده (*Microprogrammed*)

یک حافظهٔ کنترلی درپروسسور وجود دارد که شامل ریزعملهایی است که سیگنالهای کنترلی لازم را تولید می‌کند.

18

زمان‌بندی و کنترل

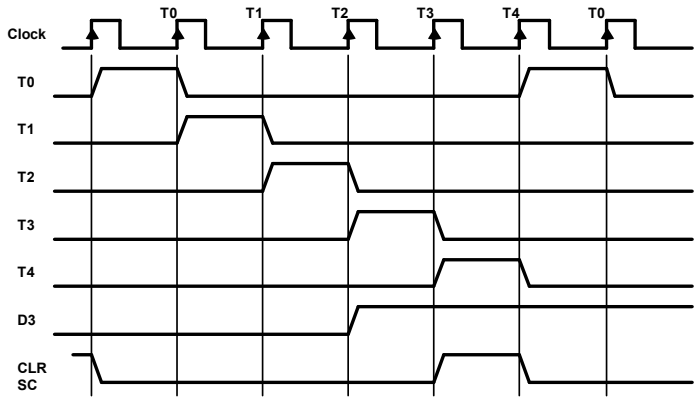
- ✓ واحد کنترل در یک کامپیوتر پایه



19

سیگنالهای زمان‌بندی

- ✓ شیگنال های زمان بندی، به وسیلهٔ دنباله شمار ۴ بیتی و دیکدر ۴*۱۶ تولید می‌شود.
- ✓ SC می‌تواند افزایش یافته یا پاک شود.
- ✓ به عنوان مثال: T0, T1, T2, T3, T4, T0, T1, . . .
- ✓ -فرض: در زمان T4، SCپاک می‌شود اگر خروجی دیکدر D3 فعال باشد.

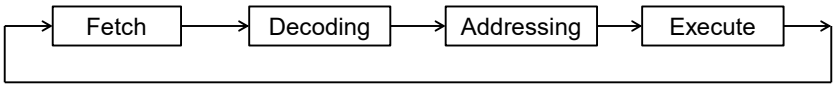


D_3T_4: SC ← 0

20

سیکل دستورالعمل

- ✓ یک برنامه مجموعه ای از دستورات است که به طور متوالی از حافظه خوانده و اجرا می شوند.
- ✓ اجرای هر دستور شامل مراحل زیر است:
  - ✓ الف ـ واکشی (Fetch)
  - ✓ ب ـ کدگشایی (Decoding)
  - ✓ پ ـ آدرس دهی (Addressing)
  - ✓ ت ـ اجرا (Execute)



21

واکشی

- ✓ در ابتدا PC با آدرس اولین دستور برنامه پر می شود و دنباله شمار (SC) را صفر می کنیم تا T_0 فعال شود. سپس با هر CP یک واحد به SC اضافه می شود تا سیگنال های T_0 تا T_{15} به ترتیب فعال شوند.

- ✓ ریزعمل هایی که در مرحله fetch انجام می شوند:

$$T_0 : AR \leftarrow PC$$

$$T_1 : IR \leftarrow M[AR] , PC \leftarrow PC +1$$

22

کدگشایی

- ✓ در مرحله کدگشایی (Decoding) در واقع نوع دستور و بیت های آدرس از یکدیگر متمایز می شوند.

- ✓ ریزعمل هایی که در مرحله کدگشایی انجام می شوند:

$T_2 : AR \leftarrow IR(0 - 11)\ ,\ D_7 - D_0 \leftarrow IR(12 - 14)\ ,\ I \leftarrow IR(15)$

23

آدرس دهی

- ✓ در این مرحله در صورتی نیاز به محاسبه آدرس مؤثر است که:
  - ✓ نخست آنکه دستور از نوع مراجعه به حافظه باشد (D_7=0).
  - ✓ دوم آنکه آدرس دهی از نوع غیر مستقیم باشد (I=1).

- ✓ بنابراین، ریزعمل هایی که در مرحله Addressing انجام می شوند:

$D_7' \ I \ T_3 : AR \leftarrow M[AR]$

$D_7' \ I' \ T_3 : \text{Nothing}$

$D_7 \ I' \ T_3 : \text{Nothing}$ ← دستور مراجعه به ثبات

$D_7 \ I \ T_3 : \text{Nothing}$ ← دستور مراجعه به I/O

} برای آدرس‌دهی کاری انجام نمی شود

24

اجرا

- دستورهای مراجعه به ثبات و مراجعه بـه I/O در همـان برهـه (اسـلات) زمانی T_3 به بعد قابل اجرا هستند.

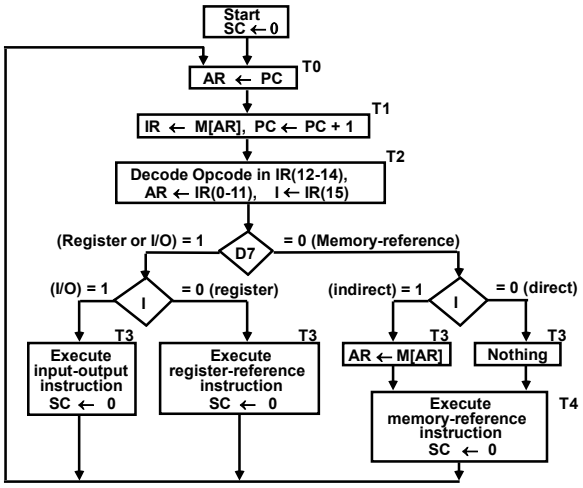
- مـثلاً، ریزعمـل هـایی کـه در مرحلـه اجـرای دسـتور AND بـا AC انجـام می‌شوند:

$$D_0\, T_4 : DR \leftarrow M[AR]$$
$$D_0\, T_5 : AC \leftarrow AC \wedge DR\ ,\ SC \leftarrow 0$$

- در پایان هر سیکل دستور، حتماً بایستی SC صفر شـود تـا بـرای سـیکل دستور بعدی آماده گردد.

25

تعیین نوع دستورالعمل



D'_7IT_3: AR ← M[AR]
$D'_7I'T_3$: Nothing
$D_7I'T_3$: Execute a register-reference instr.
D_7IT_3: Execute an input-output instr.

26

دستورات مراجعه به حافظه (۱)

نماد	عمل دیکدر	شرح نمادها
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], \quad E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, \quad PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$ $if \quad M[AR] + 1 = 0 \ Then \ PC \leftarrow PC + 1$

27

دستورات مراجعه به حافظه (۲)

AND:

D_0T_4: DR ← M[AR]
D_0T_5: AC ← AC ∧ DR, SC ← 0

ADD:

D_1T_4: DR ← M[AR]
D_1T_5: AC ← AC + DR, E ← C_{out}, SC ← 0

LDA:

D_2T_4: DR ← M[AR]
D_2T_5: AC ← DR, SC ← 0

STA:

D_3T_4: M[AR] ← AC, SC ← 0

28

دستورات مراجعه به حافظه (۳)

BUN:

D_4T_4: PC ← AR, SC ← 0

BSA:

D_5T_4: M[AR] ← PC, AR ← AR + 1
D_5T_5: PC ← AR, SC ← 0

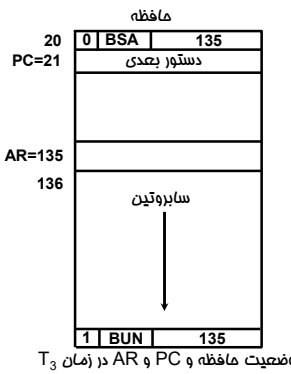
ISZ: Increment and Skip-if-Zero

D_6T_4: DR ← M[AR]
D_6T_5: DR ← DR + 1
D_6T_6: M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0

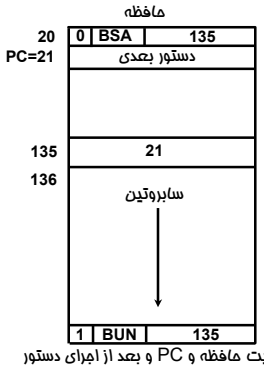
29

دستورات مراجعه به حافظه (۴)

✓ دستور انشعاب و ذخیره آدرس برگشت



وضعیت حافظه و PC و AR در زمان T_3



وضعیت حافظه و PC و بعد از اجرای دستور

30

لیست کلی ریزعملیات ها (۱)

Fetch	$R'T_0$:	AR ← PC
	$R'T_1$:	IR ← M[AR], PC ← PC + 1
Decode	$R'T_2$:	D0, ..., D7 ← Decode IR(12 ~ 14), AR ← IR(0 ~ 11), I ← IR(15)
Indirect	$D_7'IT_3$:	AR ← M[AR]
Interrupt		
	$T_0'T_1'T_2'$(IEN)(FGI + FGO):	R ← 1
	RT_0:	AR ← 0, TR ← PC
	RT_1:	M[AR] ← TR, PC ← 0
	RT_2:	PC ← PC + 1, IEN ← 0, R ← 0, SC ← 0
Memory-Reference		
AND	D_0T_4:	DR ← M[AR]
	D_0T_5:	AC ← AC ∧ DR, SC ← 0
ADD	D_1T_4:	DR ← M[AR]
	D_1T_5:	AC ← AC + DR, E ← C_{out}, SC ← 0
LDA	D_2T_4:	DR ← M[AR]
	D_2T_5:	AC ← DR, SC ← 0
STA	D_3T_4:	M[AR] ← AC, SC ← 0
BUN	D_4T_4:	PC ← AR, SC ← 0
BSA	D_5T_4:	M[AR] ← PC, AR ← AR + 1
	D_5T_5:	PC ← AR, SC ← 0
ISZ	D_6T_4:	DR ← M[AR]
	D_6T_5:	DR ← DR + 1
	D_6T_6:	M[AR] ← DR, if(DR=0) then (PC ← PC + 1), SC ← 0

31

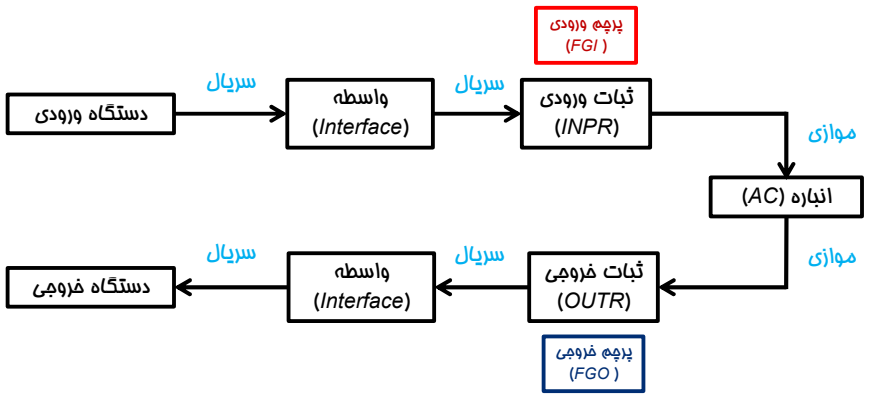
لیست کلی ریزعملیات ها (۲)

Register-Reference		
	$D_7I'T_3 = r$	(Common to all register-reference instr)
	$IR(i) = B_i$	(i = 0,1,2, ..., 11)
	r:	SC ← 0
CLA	rB_{11}:	AC ← 0
CLE	rB_{10}:	E ← 0
CMA	rB_9:	AC ← AC′
CME	rB_8:	E ← E′
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)
INC	rB_5:	AC ← AC + 1
SPA	rB_4:	If(AC(15) =0) then (PC ← PC + 1)
SNA	rB_3:	If(AC(15) =1) then (PC ← PC + 1)
SZA	rB_2:	If(AC = 0) then (PC ← PC + 1)
SZE	rB_1:	If(E=0) then (PC ← PC + 1)
HLT	rB_0:	S ← 0
Input-Output	$D_7IT_3 = p$	(Common to all input-output instructions)
	$IR(i) = B_i$	(i = 6,7,8,9,10,11)
	p:	SC ← 0
INP	pB_{11}:	AC(0-7) ← INPR, FGI ← 0
OUT	pB_{10}:	OUTR ← AC(0-7), FGO ← 0
SKI	pB_9:	If(FGI=1) then (PC ← PC + 1)
SKO	pB_8:	If(FGO=1) then (PC ← PC + 1)
ION	pB_7:	IEN ← 1
IOF	pB_6:	IEN ← 0

32

ورودی ـ خروجی (۱)

- ✓ دستورات و داده های ذخیره شده در حافظه بایستی توسط یک دستگاه وارد شوند و نتایج محاسبات نیز از طریق یک دستگاه خروجی نمایش داده شود.



33

ورودی ـ خروجی (۲)

- ✓ ثبات *INPR* و پرچم *FGI* به همراه هم اطلاعات را از ترمینال ورودی دریافت می کنند.
- ✓ ثبات *OUTR* و پرچم *FGO* به همراه هم اطلاعات را به ترمینال خروجی ارسال می کنند.
- ✓ در واقع سرعت دستگاه های جانبی با کامپیوتر تفاوت زیادی دارد که برای همگام سازی سرعت این دو از *FGI* و *FGO* استفاده می شود.
- ✓ وظیفه اصلی واسطه (*Interface*) تبدیل سیگنال ها به یکدیگر است.

34

ورودی ـ خروجی (۳)

- ✓ در هنگام ورود اطلاعات
  - ✓ ابتدا FGI=0 است.
  - ✓ فرضاً اگر کلیدی فشرده شود، کد ۸ بیتی آن به طور سریال وارد INPR شده و هنگامی که ۸ بیت INPR کامل شد، FGI=1 می شود تا کامپیوتر بفهمد که INPR پر شده است و دستگاه جانبی نیز فعلاً اطلاعاتی ارسال نمی کند.
  - ✓ وقتی که FGI=1 شد کامپیوتر محتوای INPR را به طور موازی به انباره (AC) منتقل می کند و FGI=0 می شود تا اطلاعات بعدی بتواند وارد شود.
- ✓ در هنگام خروج اطلاعات
  - ✓ ابتدا FGO=1 است و به معنای آن است که OUTR خالی است.
  - ✓ بنابراین اطلاعات AC به طور موازی به OUTR منتقل شده و FGO=0 می شود.

35

ورودی ـ خروجی (۴)

- ✓ اگر CPU بخواهد مرتباً آماده بودن INPR و OUTR را بررسی کند، وقت زیادی از CPU تلف می شود؛ به این روش، انتقال اطلاعات تحت کنترل برنامه گفته می شود.

- ✓ برای حل مشکل بالا، به جای آنکه رجیسترها مرتباً بررسی شوند، وقتی که رجیسترها آماده باشند با اعمال پالسی به CPU آماده بودن خود را اطلاع می دهند که به این روش، روش وقفه (Interrupt) گفته می شود.

36

ورودی ـ خروجی (۵)

✓ دستورات ورودی ـ خروجی

Input-Output	$D_7IT_3 = p$	(Common to all input-output instructions)
	$IR(i) = B_i$	(i = 6,7,8,9,10,11)
	p:	SC ← 0
INP	pB_{11}:	AC(0-7) ← INPR, FGI ← 0
OUT	pB_{10}:	OUTR ← AC(0-7), FGO ← 0
SKI	pB_9:	If(FGI=1) then (PC ← PC + 1)
SKO	pB_8:	If(FGO=1) then (PC ← PC + 1)
ION	pB_7:	IEN ← 1
IOF	pB_6:	IEN ← 0

37

وقفه (۱)

✓ در روش وقفه یک فلیپ فلاپ به نام R در نظر گرفته می شود.

✓ وقتی R=1 است که FGI=1 یا FGO=1 باشد.

✓ اگر فلیپ فلاپ R=1 شود به شرط آنکه پیش از آن توسط دستوری وقفه فعال شده باشد، سیکل وقفه قابل اجراست.

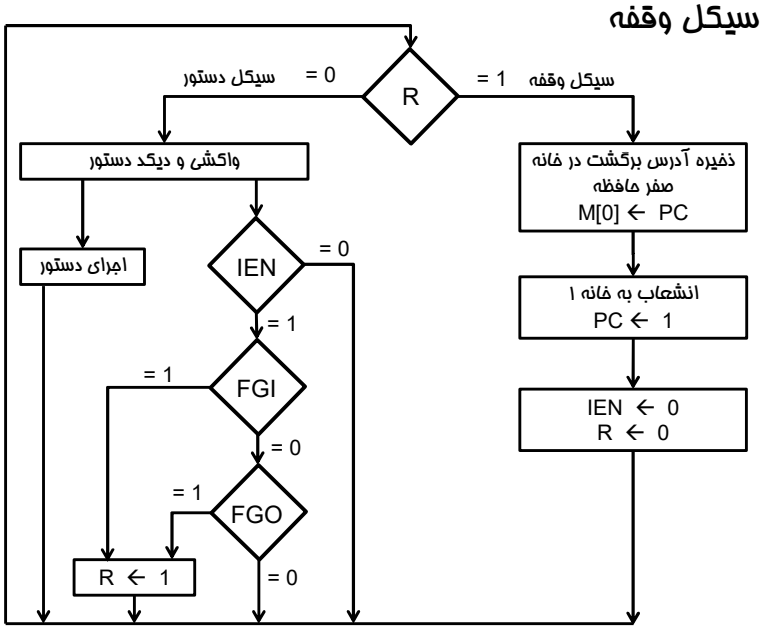
✓ عملیات سیکل وقفه کنترل برنامه را به قسمتی که سرویس مربوط به ورودی و خروجی وجود دارد منتقل می کند و پس از اتمام سرویس دهی وقفه به اجرای ادامه برنامه می پردازد.

✓ به هنگام اعمال وقفه و انجام سیکل وقفه، آدرس برگشت باید در محلی ذخیره شود که در کامپیوترهای اولیه از خانه صفر حافظه برای این منظور استفاده می شود.

38

وقفه (۲)

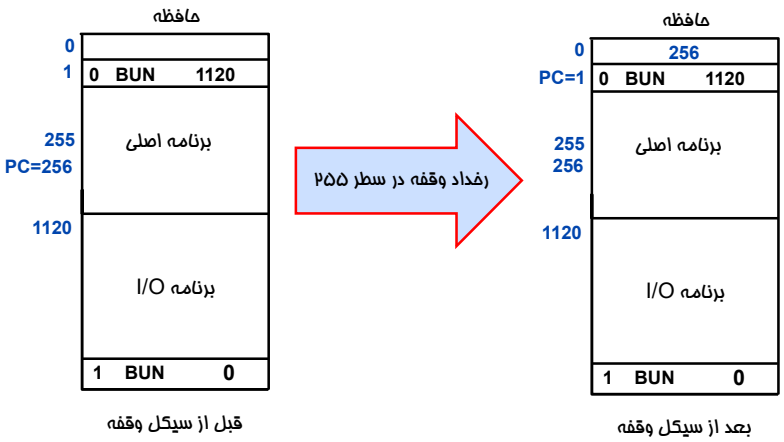
✓ فلوچارت سیکل وقفه



39

وقفه (۳)

✓مثالی از سیکل وقفه



قبل از سیکل وقفه

بعد از سیکل وقفه

40

وقفه (۴)

✓سیکل وقفه

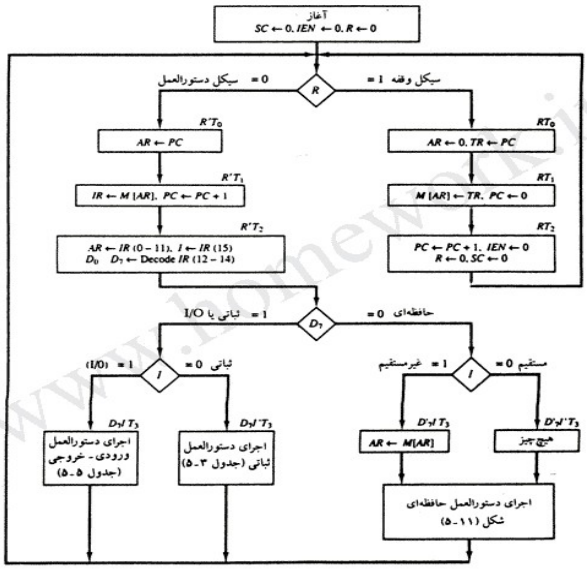
$T_0' T_1' T_2' (IEN) (FGI+FGO) : R \leftarrow 1$

$RT_0 : AR \leftarrow 0 , TR \leftarrow PC$
$RT_1 : M[AR] \leftarrow TR , PC \leftarrow 0$
$RT_2 : PC \leftarrow PC + 1 , IEN \leftarrow 0 , R \leftarrow 0 , SC \leftarrow 0$

41

وقفه (۵)

✓ فلوچارت برای اعمال کامپیوتر



42

پایان فصل
سازمان و طراحی یک کامپیوتر پایه

43

برای کسب اطلاعات بیشتر در مورد این درس می‌توانید به وب سایت آموزشی در لینک زیر مراجعه نمایید

http://shafieian-education.ir

44