معماری کامپیوتر

کنترل ریزبرنامه‌نویسی‌شده

فصل هفتم کتاب موریس مانو

محمدعلی شفیعیان

http://shafieian-education.ir

کنترل ریزبرنامه‌نویسی‌شده

- ✓ مقدمه
- ✓ حافظه واحد کنترل
- ✓ تولیدکننده آدرس بعدی
- ✓ زیرروال (سابروتین)
- ✓ نگاشت دستورالعمل
- ✓ مثال‌هایی از کنترل میکروپروگرام

2

مقدمه

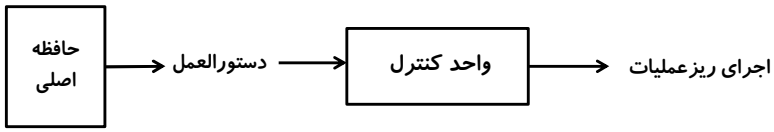
❖ **یادآوری**

✓ Micro-operation : ریزعمل

✓ Macro-operation یا Instruction : دستورالعمل : دنباله‌ای از ریزعملیات

❖ **وظیفه واحد کنترل**

✓ واحد کنترل در یک کامپیوتر، وظیفه تولید و اجرای ریزعملیات را به عهده دارد.



3

انواع واحد کنترل

❖ **واحد کنترل سخت افزاری یا سیم بندی شده (Hardwired)**

✓ سیگنال های کنترلی لازم برای پروسسور، به وسیله سخت افزار تولید می‌شود.

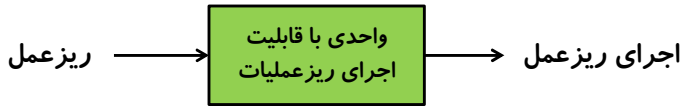
❖ **واحد کنترل ریزبرنامه ریزی شده (Micro-programmed)**

✓ تابع کنترلی که ریزعملیات را مشخص می کند یک تابع باینری است؛ زمانی که این تابع برابر ۱ باشد ریزعملیات نظیر اجرا می گردد.

4

واحد کنترل ریزبرنامه نویسی شده

❖ ابتدا از طریق تکنیک‌های سخت افزاری، واحدی طراحی می‌شود که قابلیت اجرای ریزعملیات مختلف را دارد.

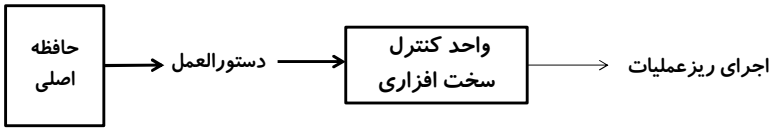


❖ برای اجرای هر دستورالعمل، کافی است ریزعملیات مربوطه را به واحد فوق بدهیم.

❖ ریزعملیات مربوط به هر دستورالعمل در حافظه‌ای خاص به نام حافظه کنترلی ذخیره می‌شود.

5

مقایسه‌ی واحدهای کنترل



6

مقایسه‌ی واحدهای کنترل

- کنترل سخت‌افزاری زمانی استفاده می‌شود که سیستم پیچیدگی کمی داشته باشدو تعداد ریزعمل‌ها محدود باشد.

- کنترل ریزبرنامه‌نویسی (میکروپروگرام) زمانی استفاده می‌شود که سیستم دارای پیچیدگی زیاد باشد.
  - در این حالت، سیگنال‌های کنترلی بیت‌های ذخیره شده در یک حافظه کنترلی هستند که به قسمت‌های مختلف ارسال می‌شوند.
  - مزیت عمده این روش، انعطاف‌پذیری آن است زیرا با تغییر در سخت‌افزار و سیستم کنترل تنها کافی است برنامه ذخیره شده در حافظه کنترلی را تغییر داد.

7

مقایسه‌ی واحدهای کنترل

❖ **واحد کنترل سخت افزاری**
- ✓ تغییر در آن مشکل است.
- ✓ می‌توان سخت افزار بهینه طراحی کرد: سرعت بالاتر.

❖ **واحد کنترل ریزبرنامه ریزی شده**
- ✓ تغییر در آن ساده است: تغییر در محتویات حافظه کنترل بدون نیاز به تغییر در سخت افزار.

8

تعریف‌ها

- کلمه کنترل (Control Word) : یک خانه از حافظه کنترلی است. در واقع، متغیرهای کنترلی که توسط واحد کنترل تولید می شوند، دنباله ای از صفرها و ۱ ها هستند که به آنها کلمه کنترلی گفته می شود.
- ریزدستور (Micro-instruction): هر کلمه کنترل دربرگیرنده یک ریزدستور است که اجرای یک یا چند ریزعمل را به عهده دارد.
- ریزبرنامه (Micro-program): مجموعه‌ای از ریزدستورات به‌طور متوالی را ریزبرنامه می‌گویند. در مجموع ریزبرنامه‌ها وظیفه کنترل سیستم دیجیتالی را بر عهده دارد. عموماً این برنامه نیازی به تغییر نداشته و بنابراین در حافظه‌های ROM ذخیره می‌شوند.

9

تعریف‌ها

❖ حافظه کنترلی جدا از حافظه اصلی و بخشی از سیستم کنترل می باشد.

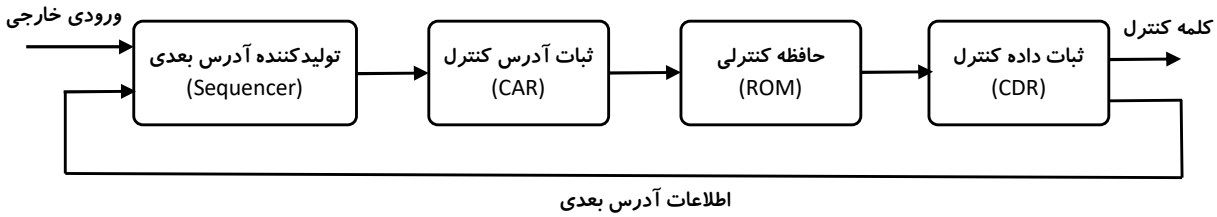
✓ هر دستور که در حافظه اصلی ذخیره شده است، موجب اجرای دنباله ای از ریزدستورات در حافظه کنترلی می شود، که این ریزدستورات، ریزعملیات مربوط به واکشی، آدرس دهی و اجرای دستور را تولید می کنند.

❖ بنابراین، در کنترل ریزبرنامه ریزی شده، دو حافظه در سیستم داریم :

- حافظه‌ی اصلی
  - ✓ در اختیار کاربر برای ذخیره‌ی برنامه‌ها و داده‌ها.
  - ✓ در حین اجرای برنامه ممکن است تغییر کند.
- حافظه‌ی کنترل
  - ✓ در اختیار طراح برای ذخیره‌ی ریزبرنامه‌ها.
  - ✓ محتوای آن در حین اجرای برنامه ثابت است.

10

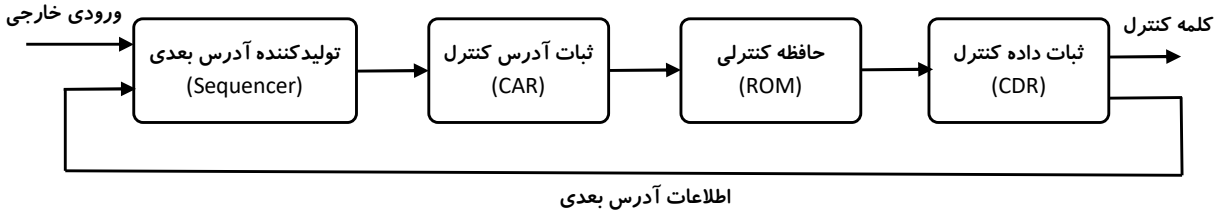
حافظه کنترل



- هر دستور ماشین که اجرا می‌شود باعث اجرای رشته‌ای از ریزعمل‌ها می‌شود تا مراحل مختلف اجرای یک دستور (Execute ،Addressing ،Fetch) اجرا شود.
- بعد از طراحی و تشخیص بیت‌های کنترلی که به‌صورت ریزدستور در حافظه کنترلی قرار می‌گیرند آنچه حائز اهمیت است، توالی اعمال ریزدستورات است.
  - این توالی باید به‌گونه‌ای باشد که فازهای مختلف اجرای دستور، متوالیاً اجرا شوند.

11

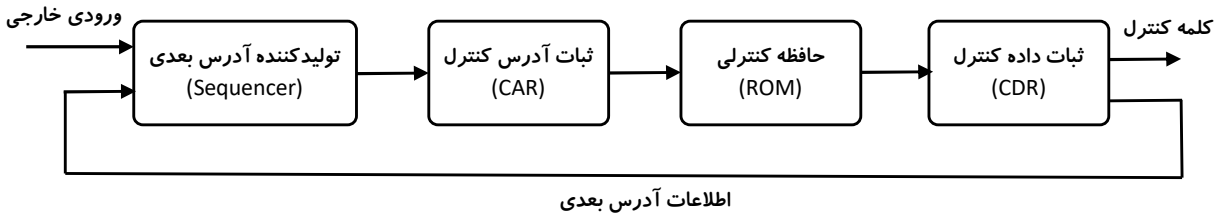
حافظه کنترل



- حافظه کنترل (ROM) حاوی ریزدستورات (اطلاعات) کنترلی است. حافظه کنترل جدا از حافظه اصلی و بخشی از سیستم کنترل می‌باشد.
- حافظه کنترل می‌تواند از نوع فقط خواندنی (ROM) باشد یا قابلیت نوشتن هم داشته باشد (RAM).
- اگر از RAM استفاده شود به آن ریزبرنامه‌نویسی پویا (dynamic microprogramming) گویند.
- معمولاً از ROM به‌عنوان حافظه کنترل استفاده می‌شود زیرا:
  - ✓ ارزان‌تر و سریع‌تر از RAM است
  - ✓ کاربر معمولی نمی‌تواند محتوای آن‌را تغییر دهد.

12

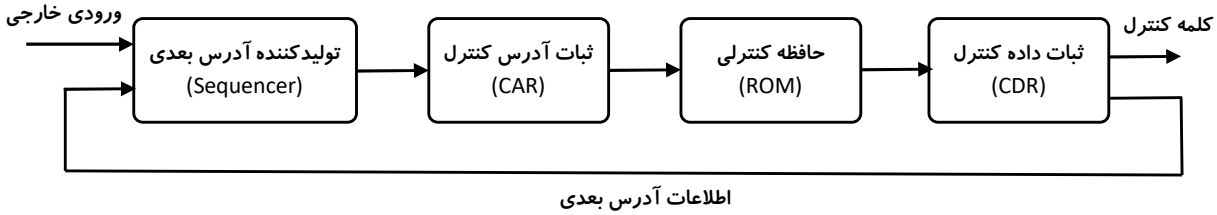
حافظه کنترل



- تولیدکننده آدرس بعدی (Sequencer) مداری است که آدرس ریزدستور بعدی حافظه کنترلی را تولید می‌کند که به آن تولیدکننده آدرس بعدی (Next Address Generator) یا توالی‌گر (Sequencer) گفته می‌شود.
- در ساده‌ترین شکل، این آدرس، آدرس ریزدستور فعلی بعلاوه یک است، اما حالاتی نیز وجود دارد که آدرس ریزدستور بعدی در مکانی دورتر از مکان فعلی است که در این حالت ریزدستور فعلی در تولید آدرس ریزدستور بعدی دخیل است. بنابراین، مداری برای تولید آدرس بعدی حافظه کنترلی مورد نیاز است.

13

حافظه کنترل



- ثبات آدرس کنترل (CAR) آدرس ریزدستورات مورد نظر برای اعمال به سیستم است.
- ثبات (رجستر) داده کنترل (CDR) حاوی ریزدستور می‌باشد و ریزدستوری که از حافظه کنترل خوانده می شود را در خود ذخیره می کند.

14

حافظه کنترل

رجیستر داده کنترل (CDR)

- ✓ این ثبات ریزدستور فعلی را در حالی در خود ذخیره می کند که همزمان ریزدستور بعدی محاسبه می شود.
- ✓ به CDR گاهی اوقات ثبات پایپلاین نیز می گویند.
- ✓ این ثبات اجازه می دهد، ریز عملیات مشخص شده توسط کلمه کنترلی، همزمان با تولید ریزدستور بعدی اجرا شوند.
- ✓ در صورت استفاده از CDR به یک کلاک با دو فاز احتیاج داریم، یک کلاک برای CDR و یکی برای CAR.
- ✓ در برخی از سیستم ها از CDR صرف نظر می شود. در این حالت سیستم فقط با کلاک تک فاز که به CAR اعمال می شود، کار می کند.

15

حافظه کنترل

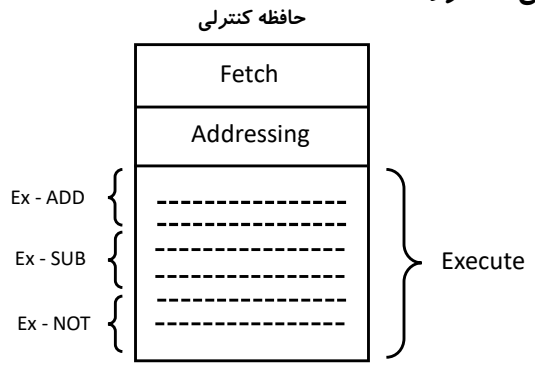
روال (Routine)

- به مجموعه‌ای از ریزدستورات که یک عمل خاص را انجام می‌دهند روال یا روتین گفته می‌شود.
- ریزدستورها در حافظه کنترلی به صورت گروهی ذخیره شده‌اند و هر گروه مربوط به یک دستور است یا مربوط به یک سیکل مانند واکش دستور یا سیکل پیدا کردن آدرس مؤثر دستور، به هر گروه، یک روتین یا روال گفته می شود.
- روتین‌های Fetch و Addressing در یک محل خاص از حافظه کنترلی هستند ولی برای اجرای هر دستور بایستی یک روتین خاص اجرا شود. بنابراین، هر دستور کامپیوتر، یک روتین مشخص در حافظه کنترل دارد که این روتین شامل ریزدستوراتی است که این ریزدستورات، ریزعملیات مربوط به آن دستور را تولید می کنند.

16

چگونگی عمل کنترل ریزبرنامه‌نویسی

- وقتی که کامپیوتر روشن می‌شود ابتدا یک آدرس اولیه در CAR بار می‌شودو معمولاً این آدرس، آدرس اولین خانه روال fetch است. بنابراین، اولین روتینی که از حافظه کنترل اجرا می شود، روتین واکشی دستور است.



17

چگونگی عمل کنترل ریزبرنامه‌نویسی

- در پایان روال fetch دستورالعمل در IR قرار گرفته است.

- پس از اتمام واکشی، حافظه کنترلی بایستی روتینی که آدرس مؤثر عملوند را می یابد(Addressing) را اجرا کند. این عمل از طریق یک ریزدستور پرش انجام می‌شود. پس از انجام روال Addressing آدرس نیز در AR قرار گرفته است.

- با توجه به بخش Opcode دستور در IR، یک زیروال خاص برای Execute اجرا می‌شود.

18

تولید آدرس ریزدستور

- لازم است برای ساده کردن ریزبرنامه نویسی، امکان فراخوانی و بازگشت از زیرروال و همچنین امکان پرش‌های شرطی و غیرشرطی فراهم شود.

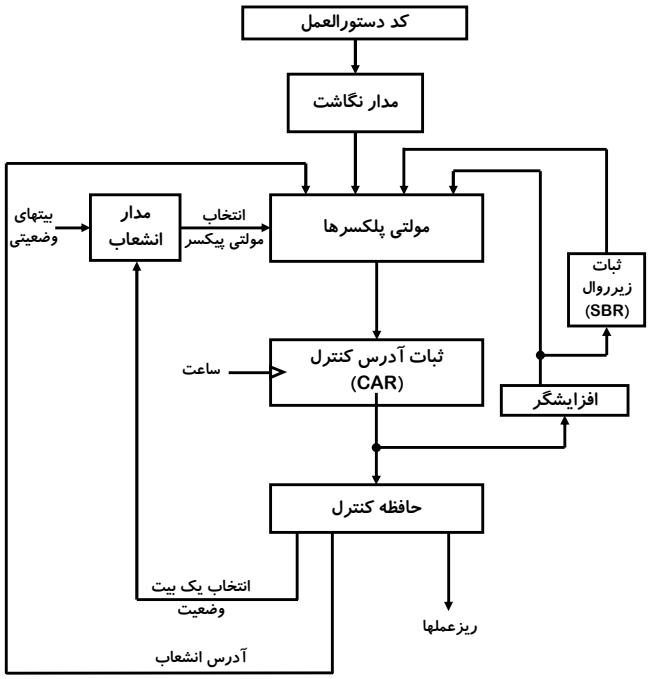
- بنابراین، آدرس ریزدستور بعدی در حافظه کنترل به یکی از روش های زیر به دست می‌آید:

قابلیت های مورد نیاز توالی گر

- ✓ نگاشت (Mapping)
- ✓ افزایش یک واحد CAR برای اجرای ریزدستورات متوالی
- ✓ پرش شرطی یا غیرشرطی (با توجه به بیتهای وضعیت)
- ✓ امکان فراخوانی و بازگشت از زیرروال

19

تولید آدرس ریزدستور



20

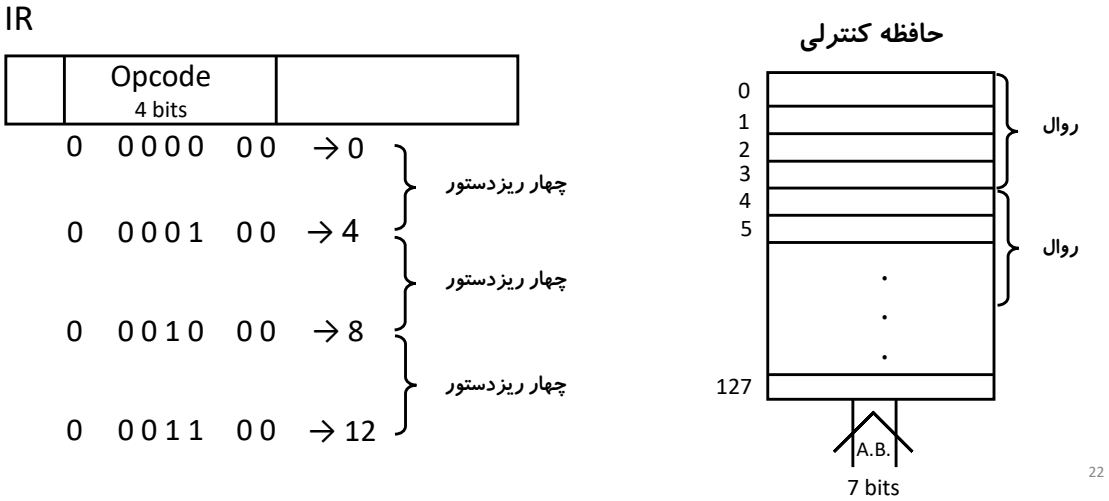
نگاشت (Mapping)

- تبدیل بیت‌های opcode به آدرس خاصی از حافظه کنترلی که در بردارنده روال مربوط به آن opcode است. یعنی هر دستور، ریزبرنامه مربوط به خود را در حافظه کنترل دارد و با توجه به کد عملیاتی دستور، آن ریز برنامه اجرا می شود.
- وقتی به روال مورد نظر دست یافتیم با افزایش CAR آدرس ریزدستورات متوالی از یک روال ایجاد می‌شود. بنابراین، توالی‌گر آدرس باید قابلیت افزایش مقدار CAR را داشته باشد تا کلیه ریزدستورات روال مربوط به یک دستورالعمل را آدرس دهی نماید.
- برای دستیابی به مکان ذخیره ریزدستور یک opcode خاص، بایستی opcode به یک آدرس در حافظه کنترلی تبدیل شود.

21

نگاشت (Mapping)

فرآیند تبدیل کد عملیاتی دستور به یک آدرس در حافظه کنترل (آدرس مربوط به روال آن دستورالعمل در حافظه کنترل) را عمل نگاشت (Mapping) می‌گویند.



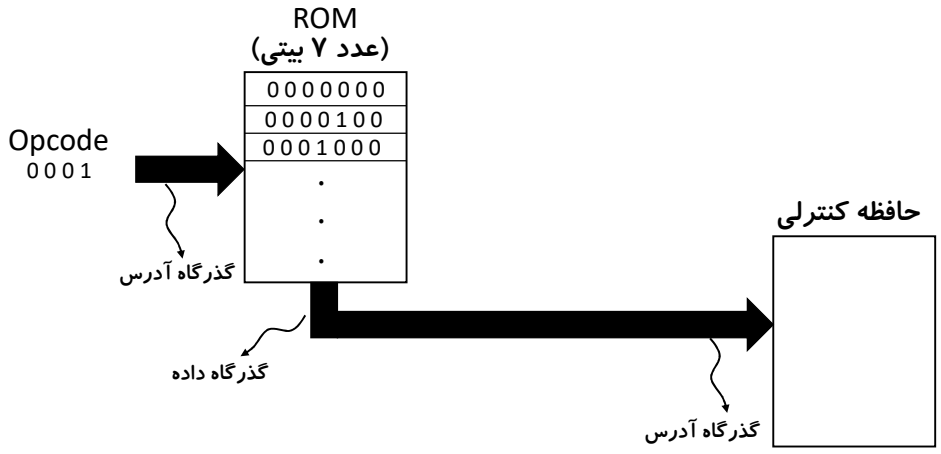
22

نگاشت (Mapping)

- روش دیگر برای نگاشت، استفاده از حافظه ROM است.
- در این حالت بیت های opcode به عنوان خطوط آدرس حافظه ROM استفاده می‌شوند.
- در این حالت طول هر کلمه حافظه ROM هفت بیت است، بنابراین با آمدن هر opcode معین، یک کلمه هفت بیتی که آدرس حافظه کنترلی است در خروجی ROM ظاهر می‌شود و به حافظه کنترلی ارسال می‌شود.

23

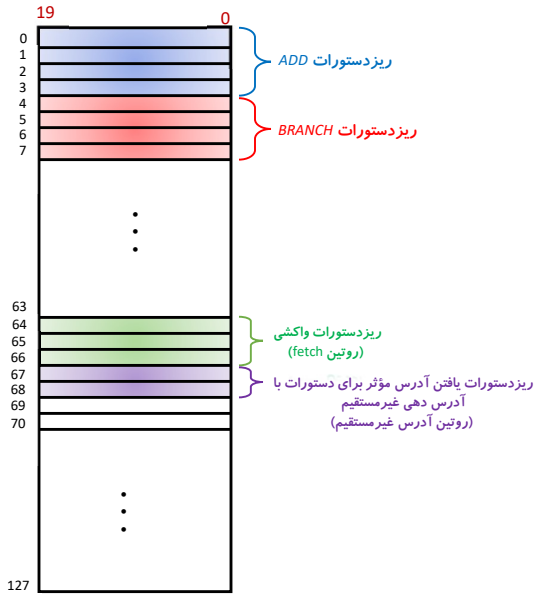
نگاشت (Mapping)



❖فرض: حافظه‌ی کنترل دارای ۱۲۸ کلمه است.

24

نگاشت (Mapping)



به عنوان مثال، دستور ADD دارای کد عملیاتی 0000 که با عمل نگاشت به 0000000 تبدیل می شود که آدرس اولین ریز دستور مربوط به ADD در حافظه کنترل است. یا دستور BRANCH که دارای کد عملیاتی 0001 می باشد پس از نگاشت به 0000100 تبدیل می شود که آدرس اولین ریزدستور مربوط به این روال است.

با توجه به این فرمول نگاشت هر دستور دارای ۴ ریزدستور می باشد. به عبارت دیگر، برای هر دستورالعمل می‌توان یک روال ریزبرنامه با ظرفیت چهار ریزدستورالعمل داشت.

- ✓ اگر روال به بیش از چهار ریزدستورالعمل نیاز داشت، می‌توان از آدرس‌های 1000000 تا 1111111 استفاده نمود.

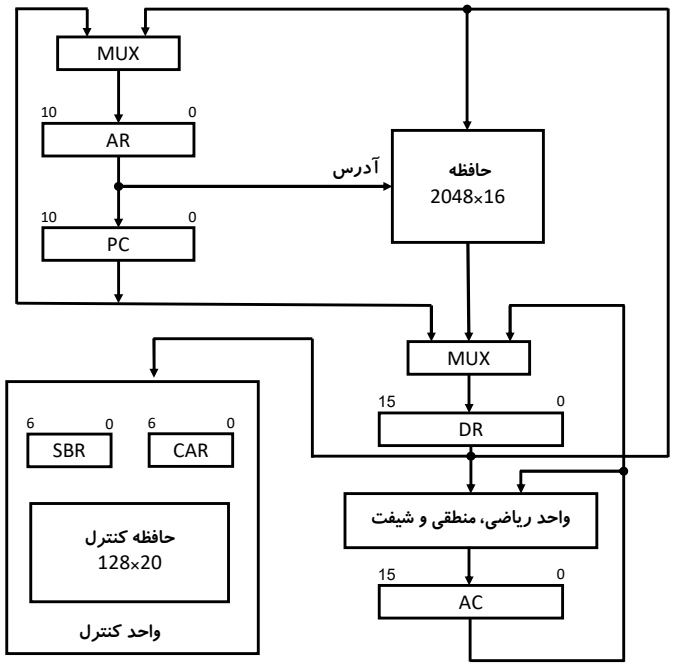
25

زیرروال

- زیرروال برنامه‌هایی هستند که روال‌های دیگر از آنها برای انجام یک کار خاص استفاده می‌کنند، لذا زیرروال ها باعث صرفه جویی در تعداد ریزدستورات می‌شود.
- زیرروال‌ها از هر کجای برنامه اصلی میکروپروگرام می‌تواند فراخوانی شود.
- ریزبرنامه هایی که زیرروال بکار می‌برند باید امکاناتی برای ذخیره آدرس برگشت به برنامه اصلی ریزبرنامه داشته باشند. برای این کار می توان آدرس برگشت را در ثبات زیرروال SBR قرار داد.
- بهترین وسیله برای ذخیره آدرس برگشت برای مراجعه به برنامه اصلی و ذخیره اطلاعات دیگر، استفاده از ثبات‌هایی است که به‌صورت حافظه پشته، LIFO، سازمان‌دهی شده باشند.

26

ساختار کامپیوتر

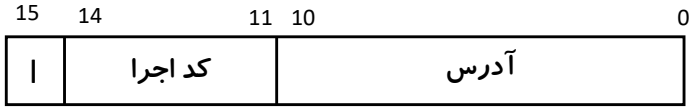


نقل و انتقال اطلاعات بین ثبات‌ها در پردازنده، به‌جای استفاده از گذرگاه، از طریق مالتی‌پلکسر انجام می‌شود.

27

دستورات کامپیوتر

✓ فرمت دستور

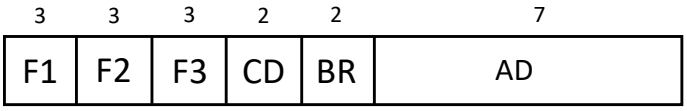


✓ چهار دستور کامپیوتر

نماد	کد اجرا	توضیح
ADD	0000	AC ← AC + M[EA]
BRANCH	0001	اگر (AC<0) آنگاه (PC ← EA)
STORE	0010	M[EA] ← AC
EXCHANGE	0011	AC ← M[EA] و M[EA] ← AC

28

فرمت ریزدستور



F1، F2 و F3: میدان‌های ریزعمل

CD: شرایط انشعاب

BR: میدان انشعاب

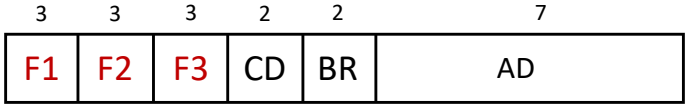
AD: میدان آدرس

29

فرمت ریزدستور

37 36 39 38 33

سمبل‌ها و کدهای باینری برای میدان‌های ریزدستورالعمل‌ها



F1	ریزعمل	سمبل
000	هیچ کار	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

F2	ریزعمل	سمبل
000	هیچ کار	NOP
001	AC ← AC - DR	SUB
010	AC ← AC v DR	OR
011	AC ← AC ^ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

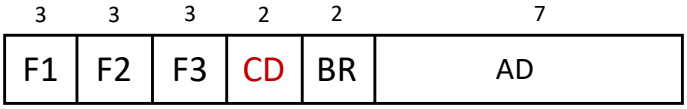
F3	ریزعمل	سمبل
000	هیچ کار	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← $\overline{AC}$	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

30

37 36 39 38 33

فرمت ریزدستور

سمبل‌ها و کدهای باینری برای میدان‌های ریزدستورالعمل‌ها



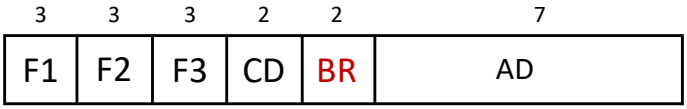
CD	شرط	سمبل	توضیح
00	1 = همیشه	U	انشعاب غیرشرطی
01	DR(15)	I	بیت آدرس غیرمستقیم
10	AC(15)	S	بیت علامت AC
11	AC=0	Z	بیت صفر در AC

31

37 36 39 38 33

فرمت ریزدستور

سمبل‌ها و کدهای باینری برای میدانهای ریزدستورالعمل‌ها



BR	سمبل	توضیح
00	JMP	اگر شرط برابر 1 باشد CAR ← AD اگر شرط برابر صفر باشد CAR ← CAR+1
01	CALL	اگر شرط برابر 1 باشد CAR ← AD, SBR ← CAR+1 اگر شرط برابر صفر باشد CAR ← CAR+1
10	RET	بازگشت از زیرروال CAR ← SBR
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

32

ریزدستورالعمل‌های سمبلیک

FETCH: PCTAR U JMP NEXT

- هر ریزدستورالعمل سمبلیک شامل ۵ فیلد می‌باشد:
  - فیلد سرفصل (عنوان) ممکن است خالی باشد یا اینکه یک آدرس سمبلیک را مشخص کند که در انتهای آن دونقطه، : ، قرار می‌گیرد.
  - فیلد ریزعمل‌ها که ممکن است شامل یک، دو یا سه سمبل باشد که با کاما از هم جدا شده‌اند. F
  - فیلد CD دارای یکی از حروف U، I، S و Z است. CD
  - فیلد BR دارای یکی از چهار سمبل JMP، CALL، RET و MAP است. BR
  - فیلد AD مشخص‌کننده مقدار آدرس ریزدستورات است.

33

ریزدستورالعمل‌های سمبلیک

تعیین مقدار آدرس ریزدستورات در فیلد AD به یکی از سه راه زیر انجام می‌شود:

- ✓ به صورت آدرس سمبلیک که باید به فرم سرفصل نیز وجود داشته باشد.
- ✓ به فرم سمبلیک NEXT که نشان‌دهنده مراجعه به آدرس بعدی است.
- ✓ زمانی که فیلد BR شامل سمبل RET یا MAP است فیلد AD خالی است و با برنامه اسمبلر به هفت عدد صفر تبدیل می‌شود.

37 36
39 38

34

ریزدستورالعمل‌های سمبلیک

- از شبه‌دستور ORG برای تعریف مبدأ یا اولین آدرس یک روال ریزبرنامه استفاده می‌شود.

- مثلاً سمبل ORG 64 به اسمبلر اطلاع می‌دهد که ریزدستورالعمل بعدی را در حافظه کنترل در آدرس‌دهی ۶۴ قرار دهد که معادل آدرس باینری 1000000 است.

35

روال واکشی دستورالعمل

AR ← PC

DR ← M[AR] , PC ← PC + 1

AR ← DR(0-10) , CAR(2-5) ← DR(11-14) , CAR(0,1,6) ← 0

F

CD

BR

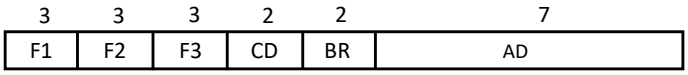
AD

```
          ORG  64
FETCH:    PCTAR          U       JMP     NEXT
          READ,INCPC     U       JMP     NEXT
          DRTAR          U       MAP
```

36

روال واکشی دستورالعمل

```
          ORG  64
FETCH:    PCTAR          U        JMP      NEXT
          READ,INCPC     U        JMP      NEXT
          DRTAR          U        MAP
```



F

CD

BR

AD

آدرس باینری F	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

37

Label	Microoperations	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
FETCH:	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	MAP	
INDRCT:	READ	U	JMP	NEXT
	DRTAR	U	RET	

F

CD

BR

AD

38

Micro Routine	Address		Binary Microinstruction					
	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

F

CD

BR

AD

39

If
Plan "A"
Doesn't work,
the alphabet has 25
more letters ...
204 if you're in Japan.

__ *Claire Cook*

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

http://shafieian-education.ir

41