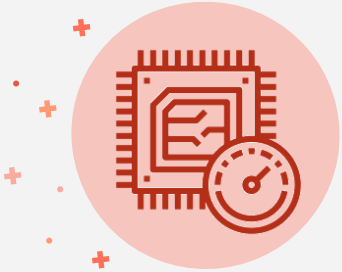


سیستم‌های دیجیتال ۲

اسلاید ۳

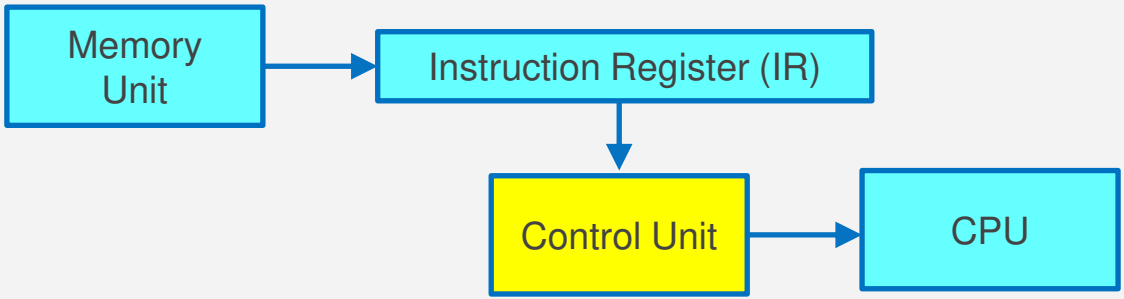
کنترل ریزبرنامه نویسی شده

محمدرضا رازقی‌زاده

رئوس مطالب

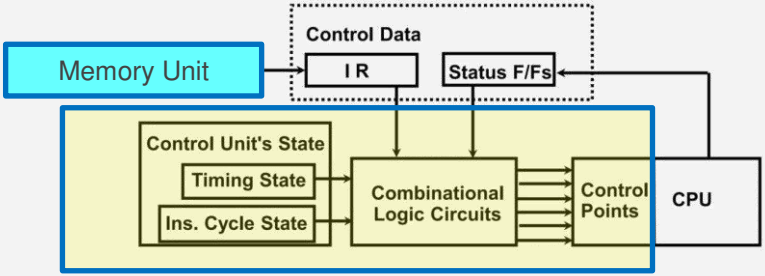
مقدمه

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

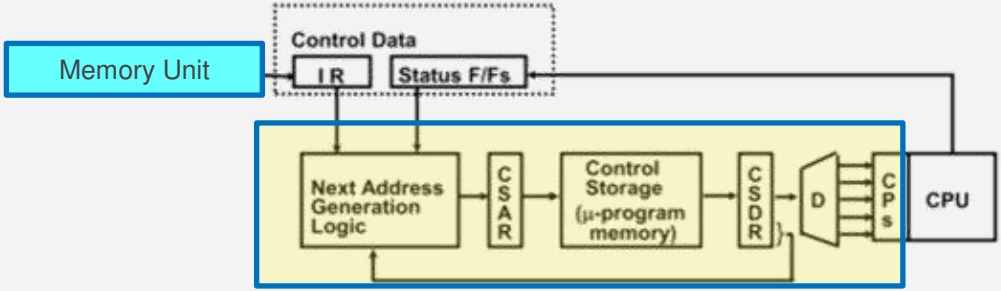


انواع واحد کنترل

- واحد کنترل سخت افزاری یا سیم بندی شده (Hardwired)
  - ✓ سیگنال های کنترلی لازم برای پروسسور، توسط سخت افزار تولید می شود.

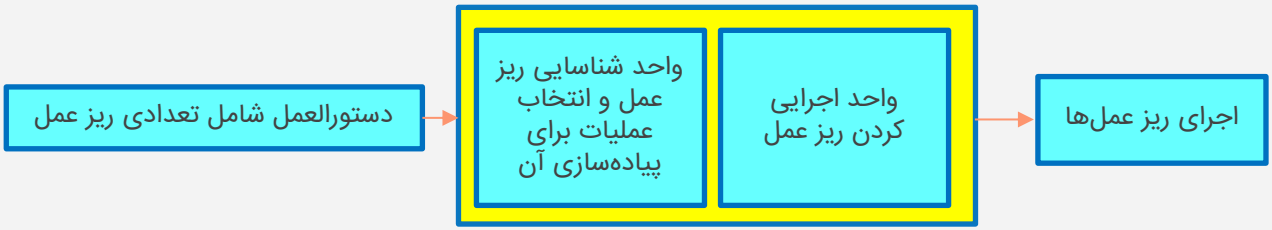


- واحد کنترل ریزبرنامه ریزی شده (Micro-programmed)
  - ✓ تابع کنترلی که ریزعملیات را مشخص می کند یك تابع باینری است.



واحد کنترل ریزبرنامه نویسی شده

- ابتدا از طریـق تکنیك هـای سـخت افـزاری، واحـدی طراحـی می شـود کـه قابلیـت اجـرای ریزعملیات مختلف را دارد.



- برای اجرای هر دستورالعمل، کافی است ریزعملیات مربوطه را به واحد فوق بدهیم.
- ریزعملیات مربوط به هر دستورالعمل در حافظه ای خـاص بـه نـام حافظـه کنترلـی ذخیـره می شود.
- در این حالت، سیگنال های کنترلی بیت های ذخیره شده در یك حافظه کنترلی هستند که به قسمت های مختلف ارسال می شوند.

مقایسه ی واحدهای کنترل

- ✓ کنترل سخت افزاری
- ✓ مناسب سیستم با پیچیدگی کم و تعداد ریزعمل ها محدود.
- ✓ انعطاف پذیری پایین: تغییر در آن مشکل است.
- ✓ سرعت بالاتر: می توان سخت افزار بهینه طراحی کرد.

- ✓ کنترل ریزبرنامه‌نویسی شده (میکروپروگرام)

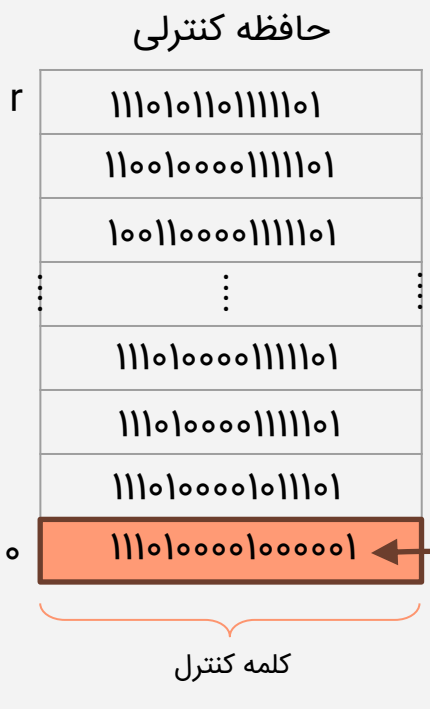
مناسب سیستم با پیچیدگی زیاد.

انعطاف پذیری بالا: برنامه ذخیره شده در حافظه کنترلی قابل تغییر است.

تغییر در آن ساده است: تغییر در محتویات حافظه کنترل بدون نیاز به تغییر در سخت افزار.

اصطلاحات مهم

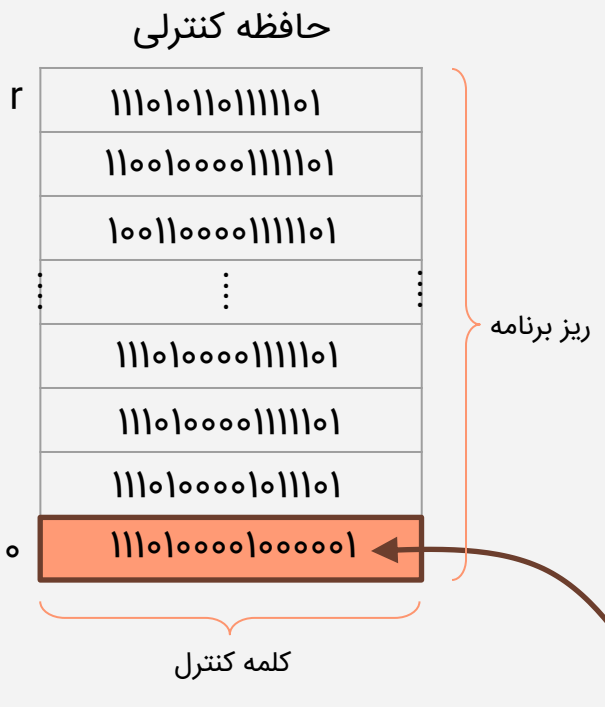
- ✓ کلمه کنترل : (Control Word) : یك خانه از حافظه کنترلی را گویند. در واقع، متغیرهای کنترلی که توسط واحد کنترل تولید می شوند، دنباله ای از ۰ و ۱ ها هستند که به آنها کلمه کنترلی گفته می شود.
- ✓ ریزدستور (Micro-instruction) : هر کلمه کنترل دربرگیرنده یك ریزدستور است که اجرای یك یا چند ریزعمل را به عهده دارد.



اصطلاحات مهم

✓ ریزبرنامه (Micro-program) :

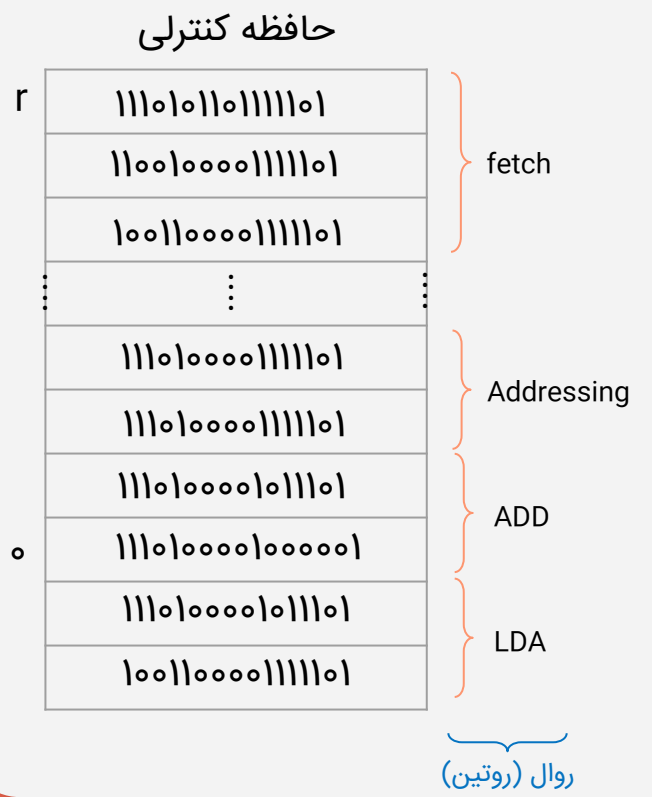
مجموعه ای از ریزدستورات به طور متوالی را ریزبرنامه می گویند. در مجموع ریزبرنامه‌ها وظیفه کنترل سیستم دیجیتالی را بر عهده دارد. عموماً این برنامه نیازی به تغییر نداشته و بنابراین در حافظه های ROM ذخیره می شوند.



اصطلاحات مهم

روال (Routine)

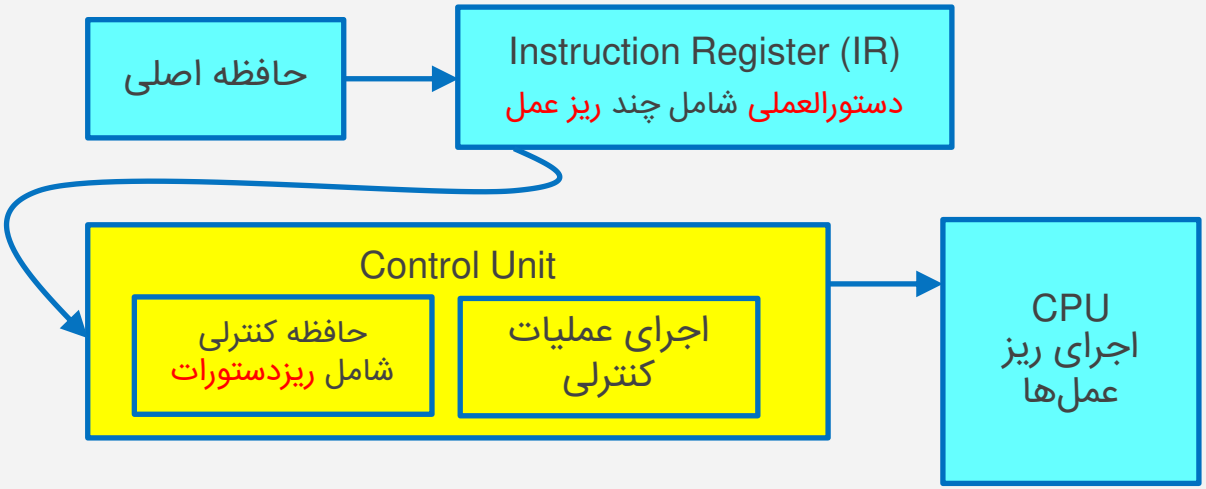
- ✓ به مجموعه ای از ریزدستورات که یك عمل خاص را انجام می دهند روال یا روتین گفته می شود.
- ✓ ریزدستورها در حافظه کنترلی به صورت گروهی ذخیره شده اند و هر گروه مربوط به یك سیکل مانند واکشی دستور یا سیکل پیدا کردن آدرس مؤثر دستور که به هر گروه، یك روتین یا روال گفته می شود.
- ✓ روتین های Fetch و Addressing در یك محل خاص از حافظه کنترلی هستند ولی برای اجرای هر دستور بایستی یك روتین خاص اجرا شود. بنابراین، هر دستور کامپیوتر، یك روتین مشخص در حافظه کنترل دارد که این روتین شامل ریزدستوراتی است که این ریزدستورات، ریزعملیات مربوط به آن دستور را تولید می کنند.



حافظه کنترل

- 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.



حافظه کنترل

- در کامپیوتر پایه با کنترل ریزبرنامه ریزی شده، دو حافظه در اختیار داریم :

انواع حافظه

حافظه اصلی

- ✓ در اختیار کاربر برای ذخیره برنامه‌ها و داده‌ها.
- ✓ در حین اجرای برنامه ممکن است تغییر کند.

حافظه کنترل

- ✓ در اختیار طراح برای ذخیره ریزبرنامه ها.
- ✓ محتوای آن در حین اجرای برنامه ثابت است.

حافظه کنترل

انواع حافظه کنترل

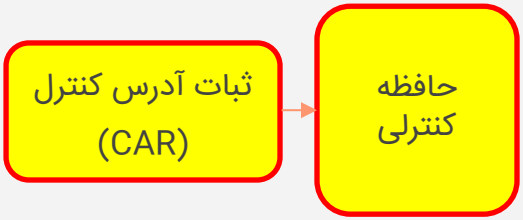
حافظه فقط خواندنی (ROM)

- ✓ ارزان تر و سریع تر از RAM است.
- ✓ کاربر معمولی نمی تواند محتوای آن را تغییر دهد.

حافظه با قابلیت نوشتن (RAM)

اگر از RAM به عنوان حافظه کنترل استفاده شود به آن ریزبرنامه نویسی پویا (microprogramming dynamic) گویند.

حافظه کنترل



- ✓ هر دستور ماشین که اجرا می شود باعث اجرای رشته ای از ریزعمل ها می شود تا مراحل مختلف اجرای یك دستور (Execute، Fetch،Addressing) اجرا شود. بنابراین واحد حافظه کنترل ریزبرنامه‌نویسی شده نیازمند اجزاء مختلفی است که باید طراحی شود.

- ✓ ثبات آدرس کنترل (CAR)

همچون ثبات آدرس (AR) در کامپیوتر پایه، ثبات آدرس کنترل نیز دربرگیرنده آدرس ریزدستورات مورد نظر برای اعلام به حافظه کنترلی است.

حافظه کنترل

- ✓ بعد از طراحی و تشخیص بیت‌های کنترلی که به صورت ریزدستور در حافظه کنترلی قرار می گیرند و آدرس آن‌ها اط طریق ثبات AR فراخوانی می شود، آنچه حائز اهمیت است، توالی اعمال ریزدستورات است.
  - این توالی باید به گونه ای باشد که فازهای مختلف اجرای دستور، متوالیاً اجرا شوند.
  - باید در نظر داشت که این توالی نیازمند مدیریت آدرس‌ها می باشد. زیرا روال‌های اجرا باید بسته به نوع دستور العمل دیکدشده، انتخاب و فراخوانی شوند. لذا فقط با شمارش خطوط آدرس از ۰ تا خانه آخر حافظه به صورت یکی یکی این هدف نتیجه نخواهد داد. بلکه برای پیاده سازی آن باید از انشعاب، پرش و ... استفاده نمود.

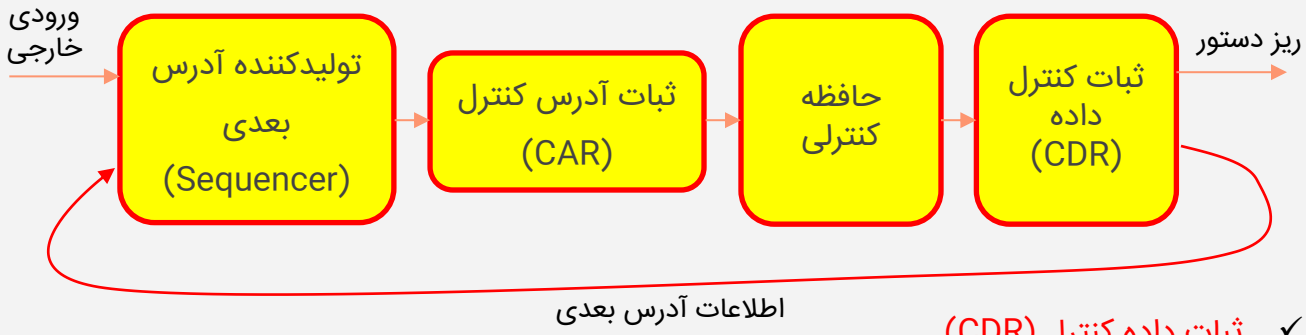
حافظه کنترل

- ✓ تولیدکننده آدرس بعدی (Sequencer)

مداری است که آدرس ریزدستور بعدی حافظه کنترلی را تولید می کند که به آن تولیدکننده آدرس بعدی (Next Address Generator) یا توالی گر (Sequencer) گفته می شود.

در ساده ترین شکل، این آدرس، آدرس ریزدستور فعلی بعلاوه یك است (مشابه ثبات PC در کامپیوتر پایه)، اما حالاتی نیز وجود دارد که آدرس ریزدستور بعدی در مکانی دورتر از مکان فعلی است که در این حالت ریزدستور فعلی باید در تولید آدرس ریزدستور بعدی دخیل باشد. بنابراین، مداری برای تولید آدرس بعدی حافظه کنترلی مورد نیاز است.

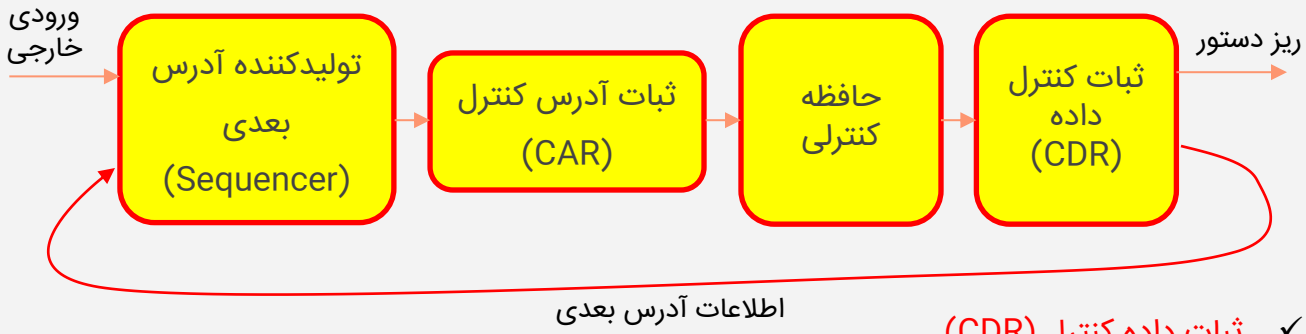
حافظه کنترل



✓ ثبات داده کنترل (CDR)

- حاوی ریزدستور می باشد و ریزدستوری که از حافظه کنترل خوانده می شود را در خود ذخیره می کند.
- 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.

حافظه کنترل

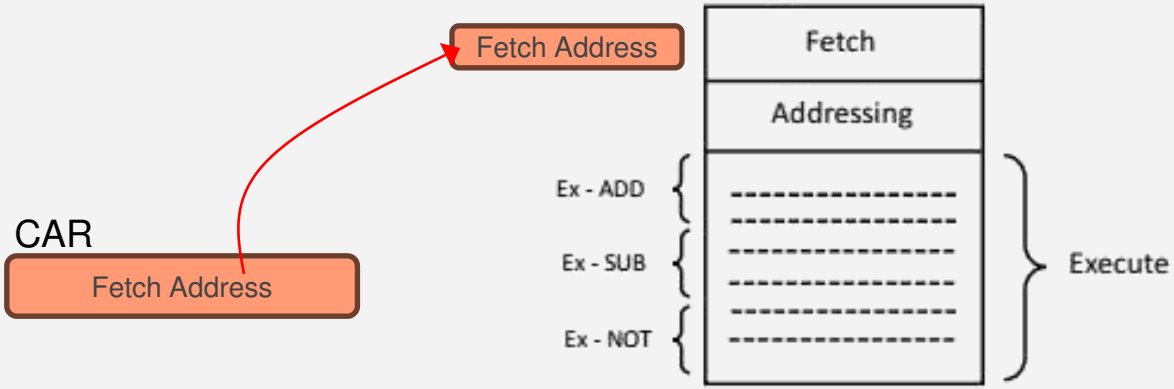


✓ ثبات داده کنترل (CDR)

- در صورت استفاده از CDR به یك کلاك با دو فاز احتیاج داریم، یك کلاك برای CDR و یکی برای CAR.
- در برخی از سیستم ها از CDR صرف نظر می شود. در این حالت سیستم فقط با کلاك تك فاز که به CAR اعمال می شود، کار می کند.

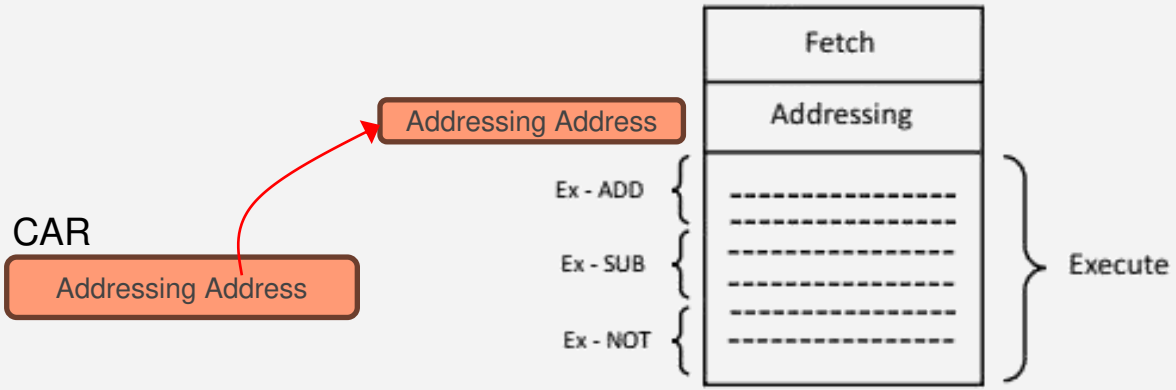
عملکرد کنترل ریزبرنامه نویسی

- ✓ وقتی کامپیوتر روشن می شود ابتدا یك آدرس اولیه در CAR بار می شود و معمولاً این آدرس، آدرس اولین خانه روال fetch است. بنابراین، اولین روتینی که از حافظه کنترل اجرا می شود، روتین واکشی دستور خواهد بود.



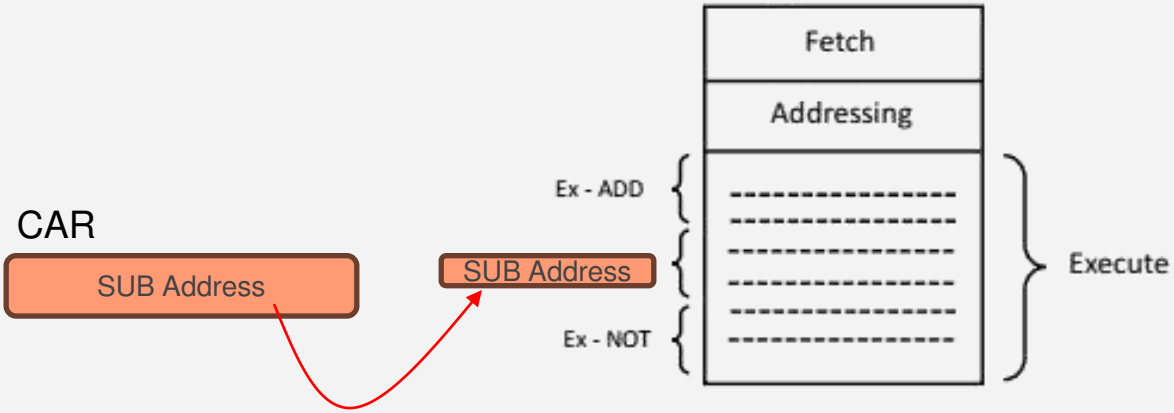
عملکرد کنترل ریزبرنامه نویسی

- ✓ 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 یك ریزدستور پرش انجام می شود. پس از انجام روال Addressing آدرس نیز در AR قرار گرفته است.



عملکرد کنترل ریزبرنامه نویسی

- ✓ با توجه به بخش Opcode دستور در IR، آدرس ریز دستورات مربوط به روال اجرا انتخاب شده و یك زیروال خاص برای Execute اجرا می شود.
- ✓ به طور مثال اگر opcode مربوط به دستور SUB باشد:



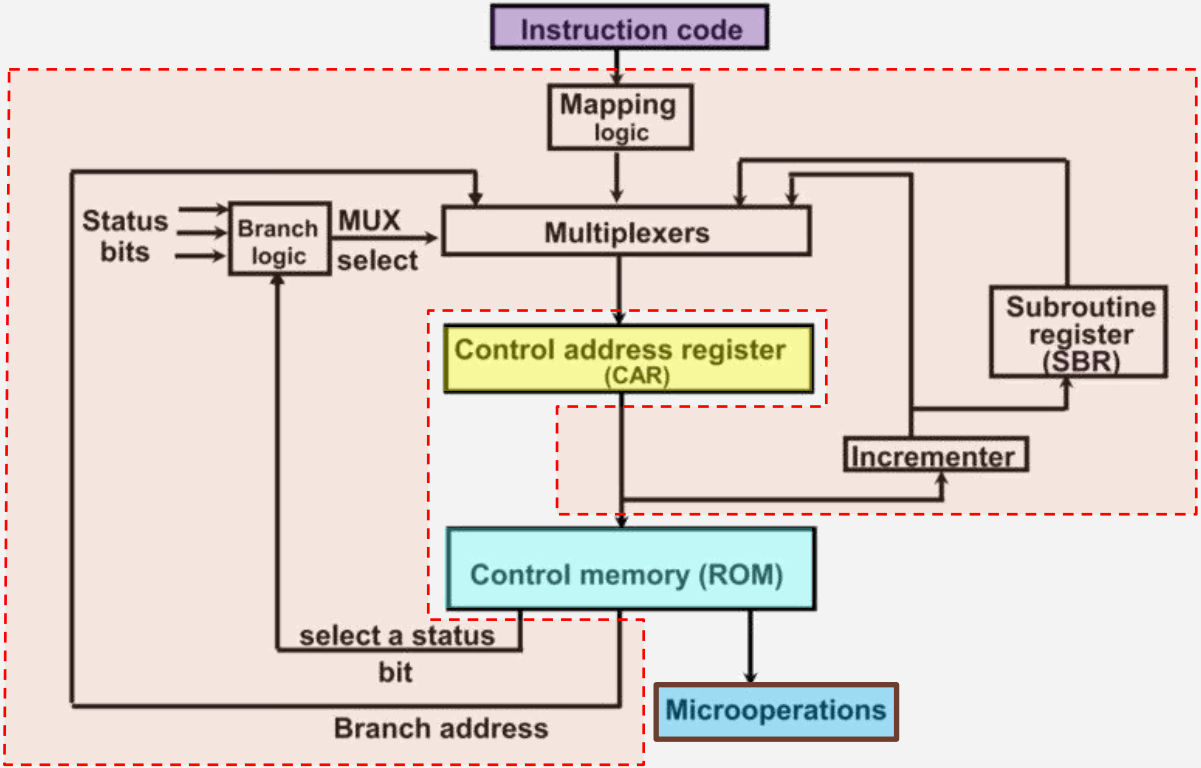
تولید آدرس ریزدستور

باتوجه به عملکرد واحد کنترل برای مشخص کردن آدرس ریز دستور بعدی در حافظه کنترلی و همچنین برای ساده کردن ریزبرنامه‌نویسی، لازم است تا امکان فراخوانی و بازگشت از زیرروال و همچنین امکان پرش های شرطی و غیرشرطی نیز فراهم شود. از این رو در کامپیوتر پایه انواع روش‌های محاسبه آدرس ریزدستور بعدی در حافظه کنترل به صورت سخت‌افزاری پیاده‌سازی شده است:

قابلیت های مورد نیاز توالی گر

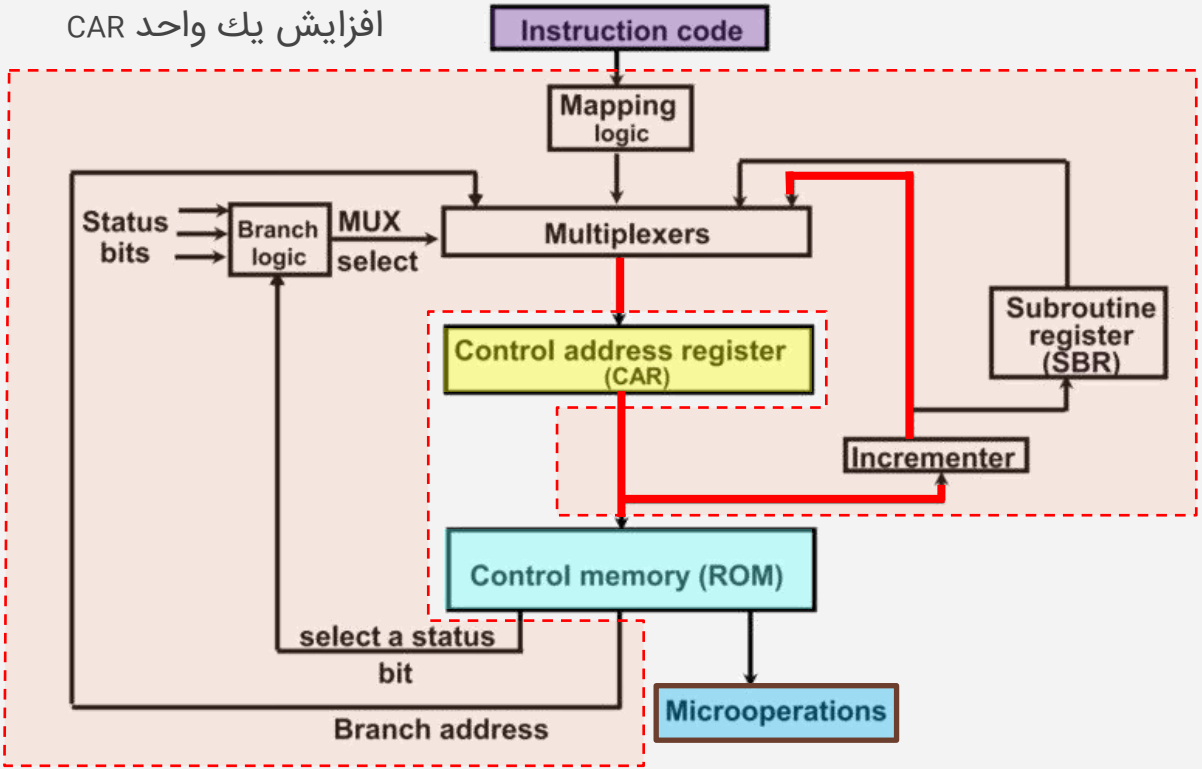
- ✓ نگاشت (Mapping)
- ✓ افزایش یك واحد CAR برای اجرای ریزدستورات متوالی
- ✓ پرش شرطی یا غیرشرطی (با توجه به بیتهای وضعیت)
- ✓ امکان فراخوانی و بازگشت از زیرروال

تولید آدرس ریزدستور



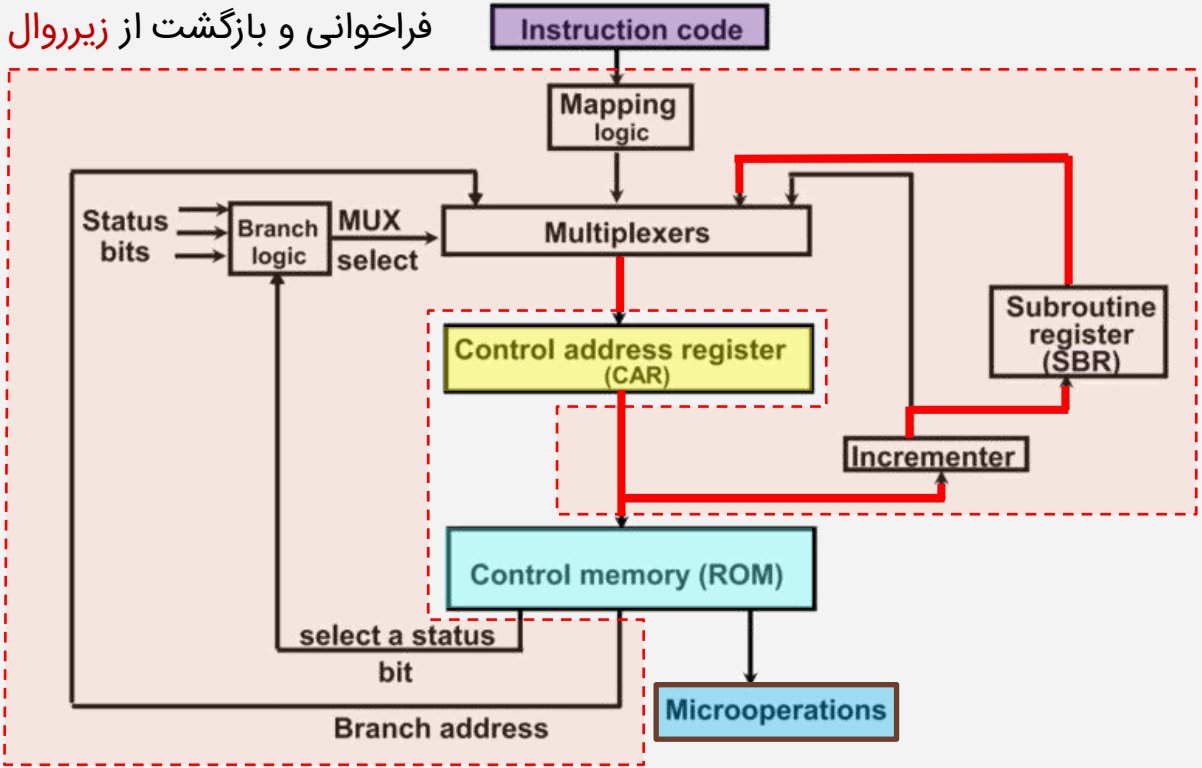
تولید آدرس ریزدستور

افزایش یك واحد CAR



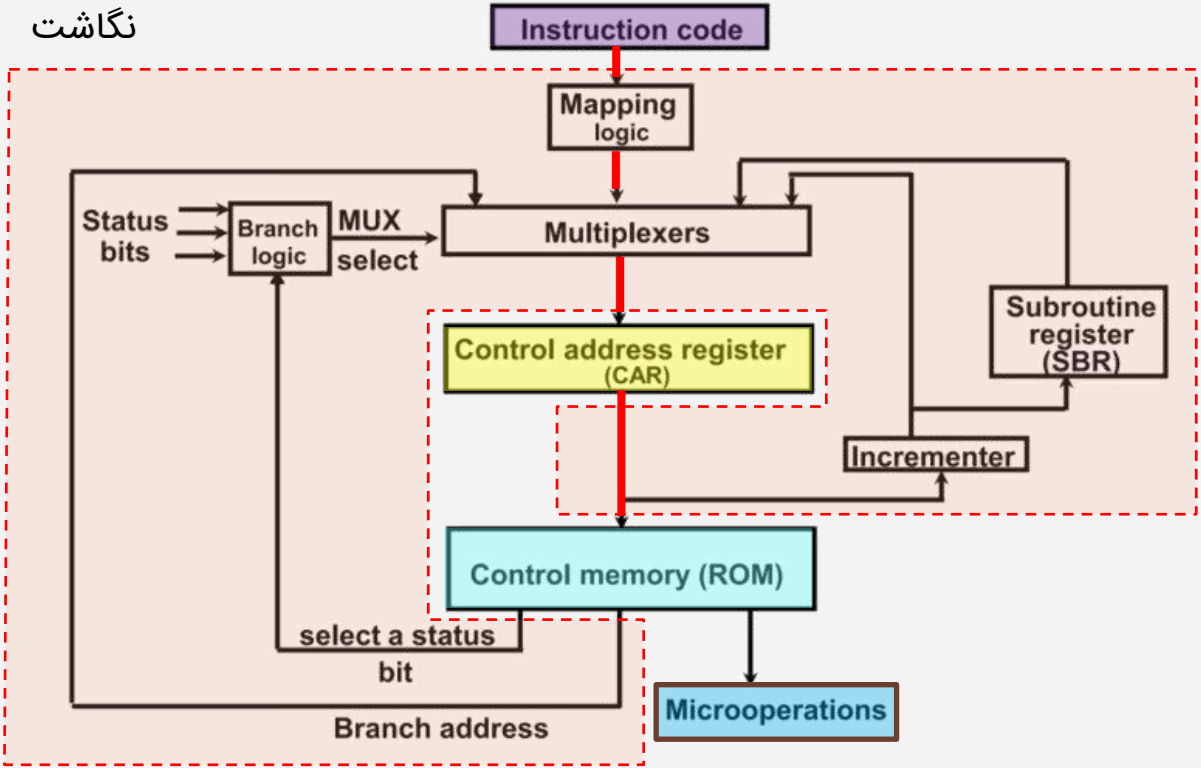
تولید آدرس ریزدستور

فراخوانی و بازگشت از زیرروال



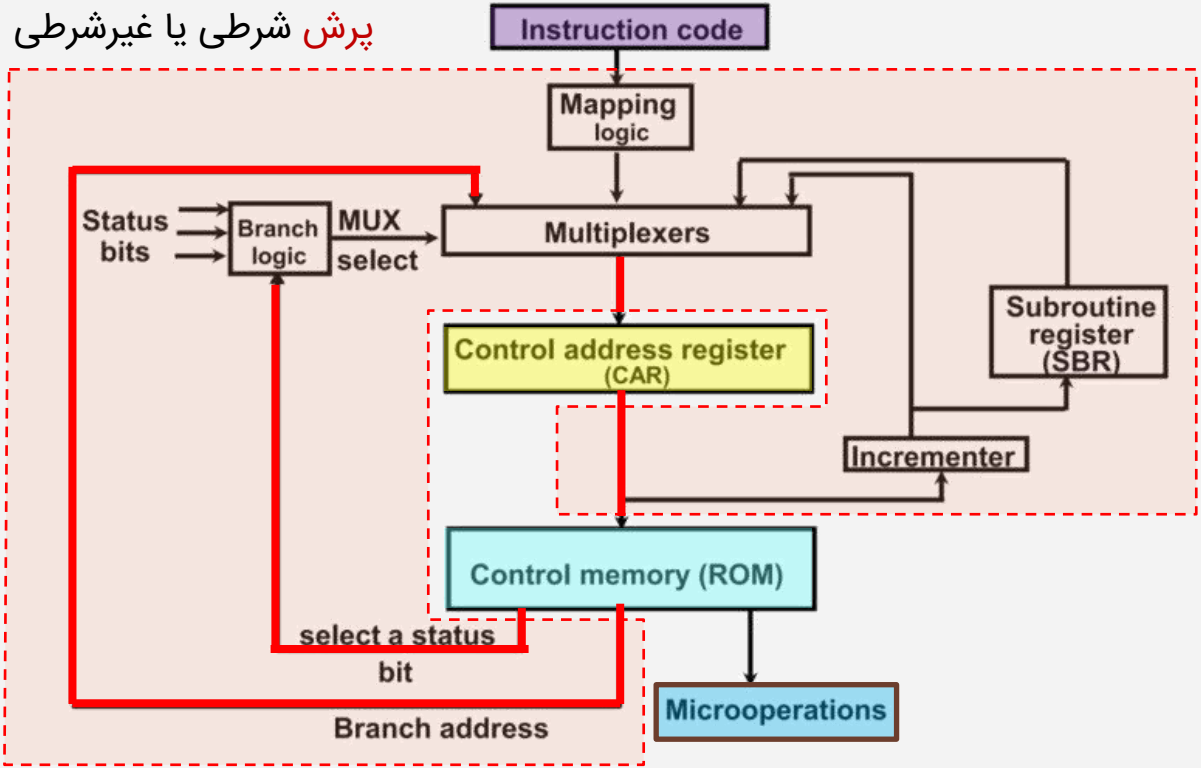
تولید آدرس ریزدستور

نگاشت



تولید آدرس ریزدستور

پرش شرطی یا غیرشرطی

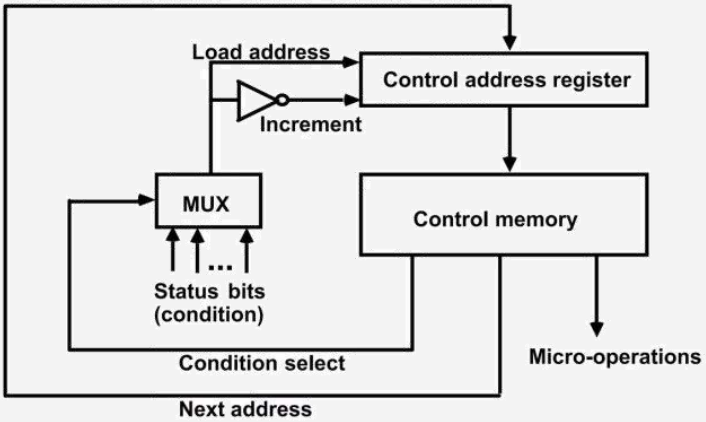


✓ پرش شرطی:

- اگر شرط برقرار باشد، سپس به آدرسی که از فیلد آدرس بعدی در ریزدستورالعمل فعلی بدست می‌آید، پرش می‌کند.
- در غیر این‌صورت به مسیر اصلی ادامه می‌دهد.
- شرایطی که باید بررسی شوند: سرریز O، منفی N، صفر Z، کری C و غیره.

✓ پرش بدون شرط:

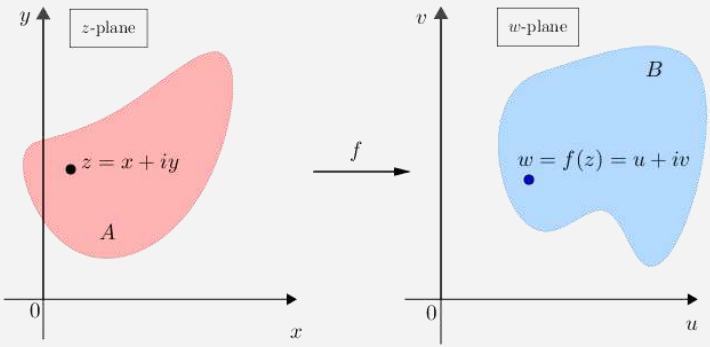
- با ثابت نگه داشتن مقدار یک بیت وضعیت در ورودی مالتی‌پلکسر روی ۱، پرش بدون قید و شرط صورت می‌گیرد.



نگاشت (Mapping)

- تبدیل بیت های opcode به آدرس خاصی از حافظه کنترلی که در بردارنده روال مربوط بـه آن opcode است. یعنی هر دستور، ریزبرنامه مربوط به خود را در حافظه کنترل دارد و بـا توجه به کد عملیاتی دستور، آن روال اجرا می شود. به عبارت دیگر برای دستیابی به مکان ذخیره ریزدستور یـك opcode خـاص، بایسـتی opcode بـه یـك آدرس در حافظه کنترلی تبدیل شود.

- وقتی به روال مورد نظر دست یافتیم با افـزایش CAR آدرس ریزدسـتورات متـوالی از یـك روال ایجاد می شود. بنابراین، توالی‌گر آدرس باید قابلیت افزایش مقدار CAR را داشته باشد تا کلیه ریزدستورات روال مربوط به یك دستورالعمل را آدرس دهی نماید.



نگاشت (Mapping)

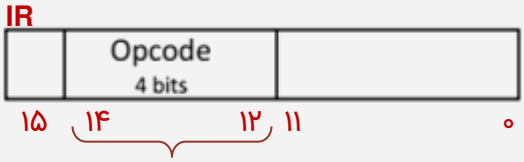
- فرآیند تبدیل کد عملیاتی دستور به یك آدرس در حافظه کنترل (آدرس مربوط به روال آن دستورالعمل در حافظه کنترل) را عمل نگاشت گویند.

- فرض: حافظه کنترلی در کامپیوتر پایه شامل ۱۲۷ (معادل 2^7) خانه می‌باشد که طول هر کلمه آن ۲۰ بیت است.

- از آنجا که تعداد خانه‌های حافظه کنترلی 2^7 است، لذا به ۷ بیت برای آدرس‌دهی نیاز داریم.

- در طراحی کامپیوتر پایه با کنترل ریزبرنامه‌نویسی شده تعداد بیت‌های opcode در IR برابر ۴ بیت در نظر گرفته شده است. لذا از آنجا که یک بیت هم مربوط به نوع آدرس دهی می‌باشد، ۱۱ بیت برای آدرس‌دهی در نظر گرفته می‌شود. از این رو تعداد خانه‌های حافظه اصلی از 2^{12} به 2^{11} تغییر می‌کند. به عبارت دیگر به جای ۴۰۹۶ کلمه شامل ۲۰۴۸ کلمه خواهد بود. همچنین ثبات‌های PC و AR نیز از ۱۲ بیت به ۱۱ بیت کاهش می‌یابند.

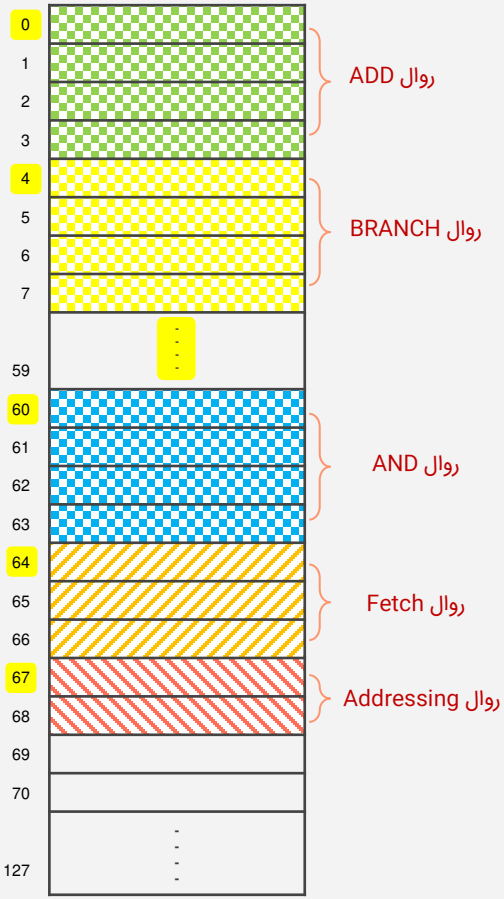
نگاشت (Mapping)



0	0	0	0	→	0	0	0	0	0	0	0	→	0
0	0	0	1	→	0	0	0	0	1	0	0	→	4
0	0	1	0	→	0	0	0	1	0	0	0	→	8
0	0	1	1	→	0	0	0	1	1	0	0	→	12
0	1	0	0	→	0	0	1	0	0	0	0	→	16
0	1	0	1	→	0	0	1	0	1	0	0	→	20
0	1	1	0	→	0	0	1	1	0	0	0	→	24
0	1	1	1	→	0	0	1	1	1	0	0	→	28
1	0	0	0	→	0	1	0	0	0	0	0	→	32
1	0	0	1	→	0	1	0	0	1	0	0	→	36
1	0	1	0	→	0	1	0	1	0	0	0	→	40
1	0	1	1	→	0	1	0	1	1	0	0	→	44
1	1	0	0	→	0	1	1	0	0	0	0	→	48
1	1	0	1	→	0	1	1	0	1	0	0	→	52
1	1	1	0	→	0	1	1	1	0	0	0	→	56
1	1	1	1	→	0	1	1	1	1	0	0	→	60

4

نگاشت (Mapping)

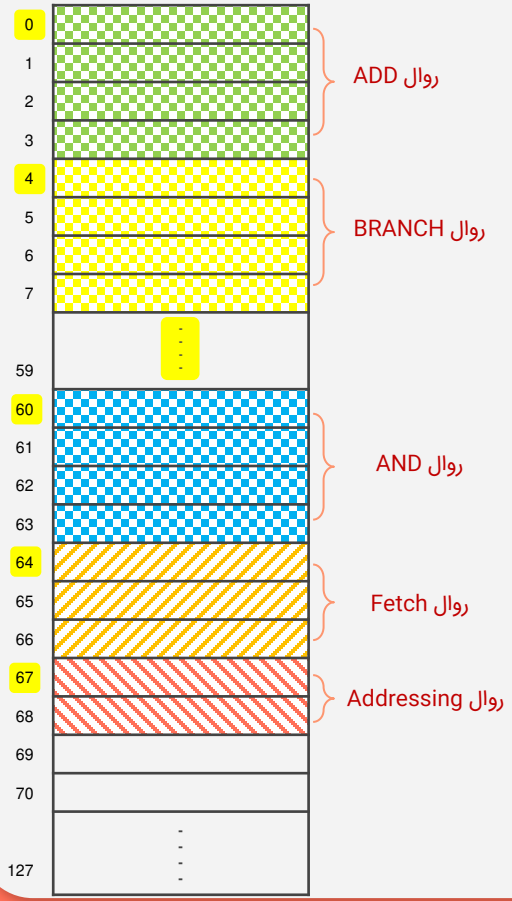


نگاشت (Mapping)

- به عنوان مثال، دستور ADD دارای کد عملیاتی ۰۰۰۰ که با عمل نگاشت به ۰۰۰۰۰۰۰ تبدیل می شود که آدرس اولین ریز دستور مربوط به ADD در حافظه کنترل است. یا دستور BRANCH که دارای کد عملیاتی ۰۰۰۱ می باشد پس از نگاشت به ۰۰۰۰۱۰۰ تبدیل می شود که آدرس اولین ریزدستور مربوط به این روال است.

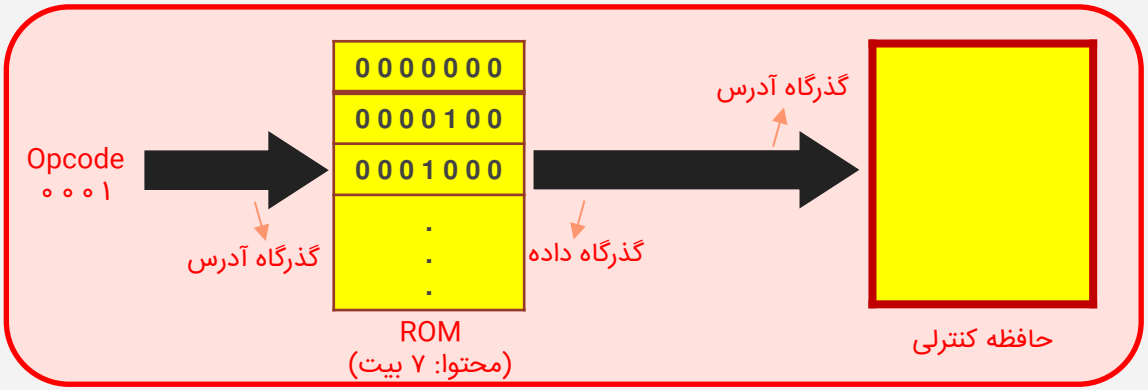
- با توجه به این فرمول نگاشت هر دستور دارای ۴ ریزدستور می باشد. به عبارت دیگر، برای هر دستورالعمل می توان یك روال ریزبرنامه با ظرفیت چهار ریزدستورالعمل داشت.

- اگر روال به بیش از چهار ریزدستورالعمل نیاز داشت، می توان از آدرس های ۱۰۰۰۰۰۰ تا ۱۱۱۱۱۱۱ استفاده نمود.

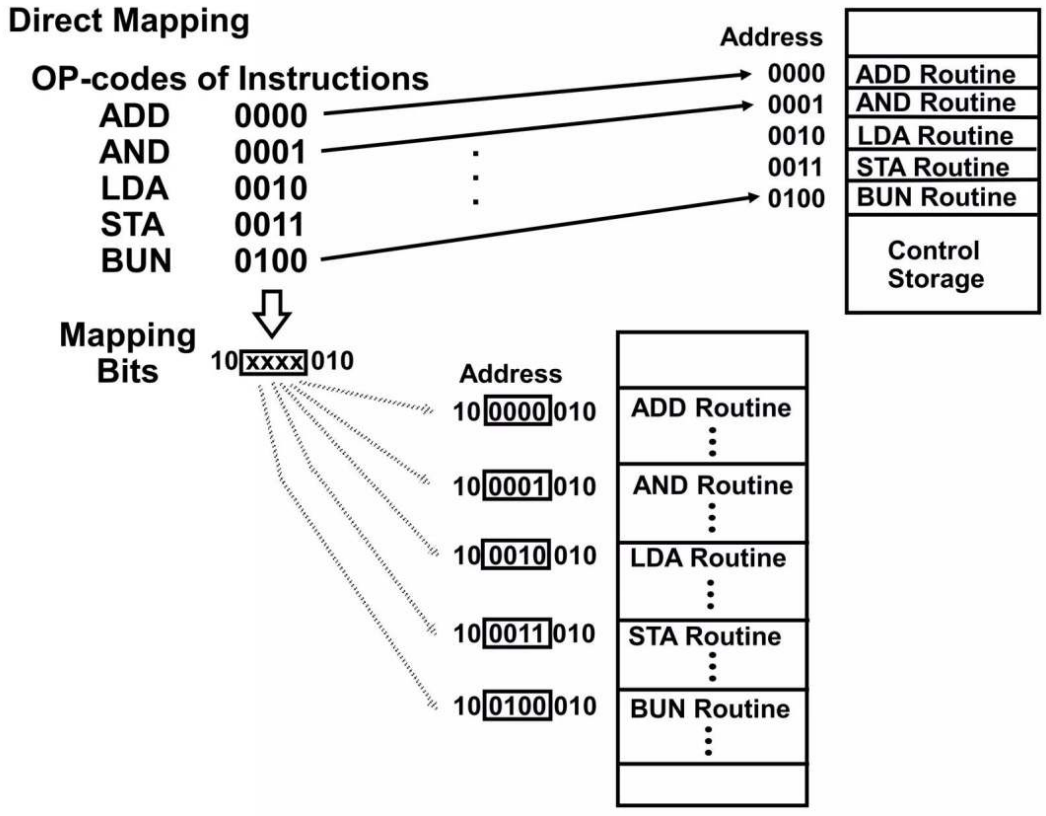


نگاشت (Mapping)

- ✓ یک روش پیاده‌سازی عملی برای نگاشت، استفاده از حافظه ROM برای ذخیره نگاشت یافته opcode 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، یك كلمه هفت بیتی كه آدرس حافظه كنترلی است و در خروجی ROM ظاهر می شود و به حافظه كنترلی ارسال می شود.



نگاشت (Mapping)



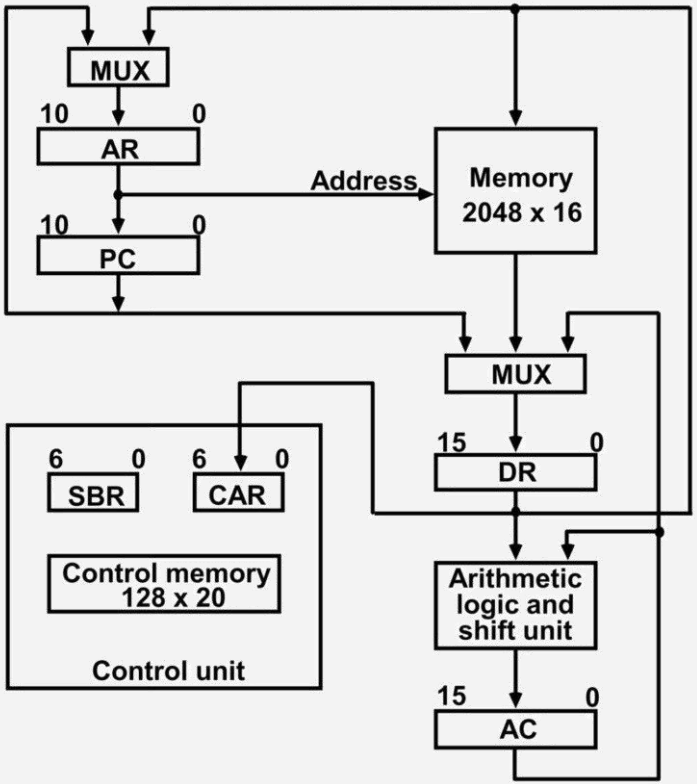
زیرروال

- ✓ زیرروال برنامه هایی هستند که روال های دیگر از آنها برای انجام یک کار خاص استفاده می کنند، لذا زیرروال ها باعث صرفه جویی در تعداد ریزدستورات می‌شوند.

- ✓ زیرروال ها از هر کجای برنامه اصلی میکروپروگرام می توانند فراخوانی شود.

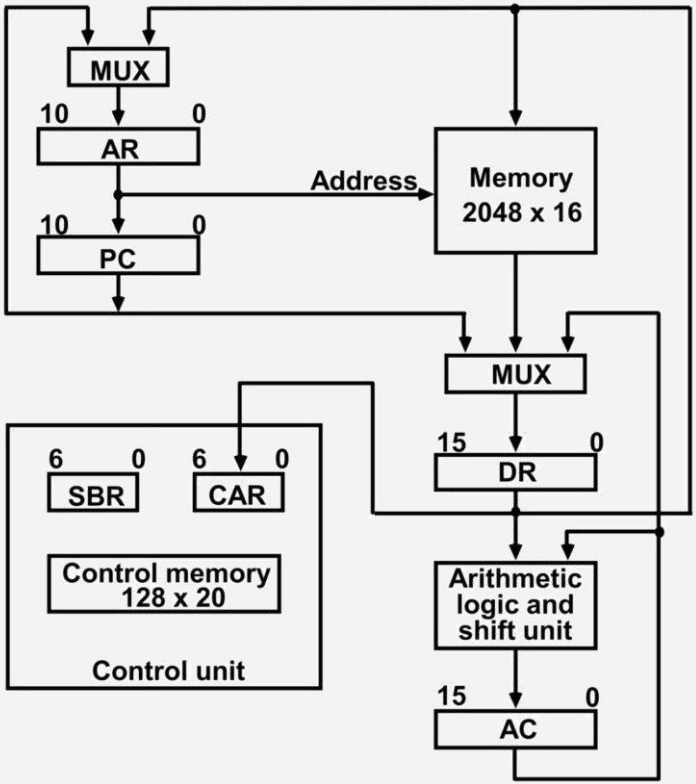
- ✓ ریزبرنامه‌هایی که زیرروال بکار می برند باید امکاناتی برای ذخیره آدرس برگشت به برنامه اصلی ریزبرنامه داشته باشند. برای این کار می توان آدرس برگشت را در ثبات زیرروال SBR قرار داد.

ساختار کامپیوتر



- ✓ پس از اتمام چیدمان یک کامپیوتر و طراحی واحد کنترل ریزبرنامه‌نویسی شده، مرحله تولید ریزکد برای حافظه کنترل توسط طراح فرا می‌رسد.
- ✓ به جهت درک بهتر چگونگی تولید ریزکد برای حافظه کنترل، ابتدا چیدمان فرضی یک کامپیوتر دیجیتال را مطابق شکل زیر در نظر می‌گیریم.
- ✓ در این چیدمان، انتقال اطلاعات بین ثبات‌های پردازنده، به جای استفاده از گذرگاه مشترک از طریق مالتی پلکسر صورت می‌گیرد.

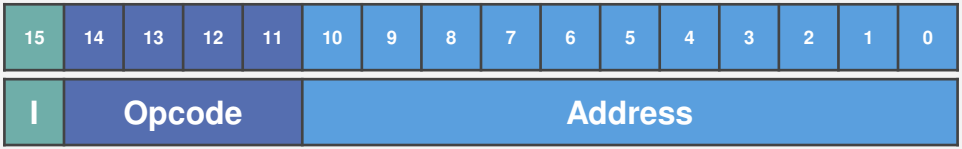
ساختار کامپیوتر



- ✓ در این چیدمان آدرس به صورت ۱۱ بیتی فرض شده است.
- ✓ داده‌ها همچون قبل ۱۶ بیتی فرض شده‌اند.
- ✓ حافظه اصلی ۲۰۴۸ کلمه ۱۶ بیتی فرض شده است.
- ✓ حافظه کنترل شامل ۱۲۸ کلمه ۲۰ بیتی می‌باشد.

قالب دستورالعمل‌های کامپیوتر

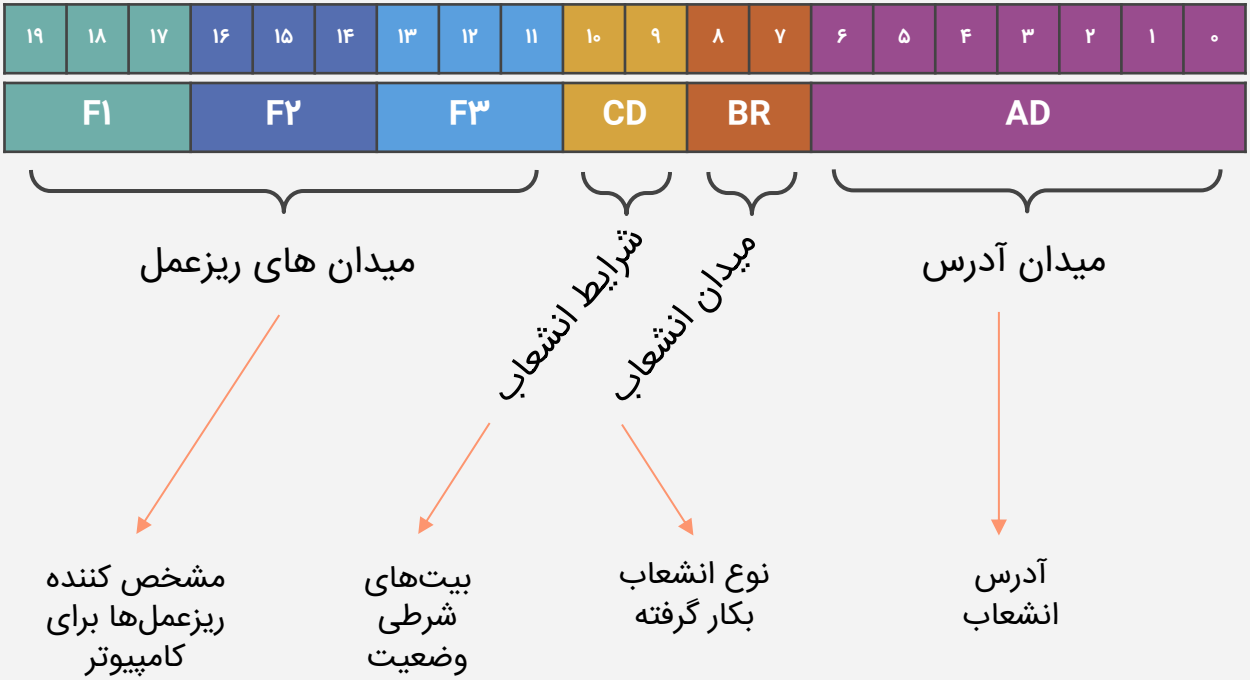
- ✓ فرمت دستور



- ✓ 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

field : میدان


۱۹ ۱۸ ۱۷ ۱۶ ۱۵ ۱۴ ۱۳ ۱۲ ۱۱ ۱۰ ۹ ۸ ۷ ۶ ۵ ۴ ۳ ۲ ۱ ۰
F1
F۲
F۳
CD
BR
AD
میدان های ریزعمل
شرایط انشعاب
میدان انشعاب
میدان آدرس
مشخص کننده ریزعمل‌ها برای کامپیوتر
بیت‌های شرطی وضعیت
نوع انشعاب بکار گرفته
آدرس انشعاب

قالب ریزدستورالعمل‌های واحد کنترل

سمبل	ریزعمل	F۳
NOP	هیچ کار	۰۰۰
XOR	AC ← AC ⊕ DR	۰۰۱
COM	AC ← $\overline{AC}$	۰۱۰
SHL	AC ← shl AC	۰۱۱
SHR	AC ← shr AC	۱۰۰
INCPC	PC ← PC + ۱	۱۰۱
ARTPC	PC ← AR	۱۱۰
	Reserved	۱۱۱

سمبل	ریزعمل	F۲
NOP	هیچ کار	۰۰۰
SUB	AC ← AC - DR	۰۰۱
OR	AC ← AC v DR	۰۱۰
AND	AC ← AC ^ DR	۰۱۱
READ	DR ← M[AR]	۱۰۰
ACTDR	DR ← AC	۱۰۱
INCDR	DR ← DR + ۱	۱۱۰
PCTDR	DR(۰-۱۰) ← PC	۱۱۱

سمبل	ریزعمل	F1
NOP	هیچ کار	۰۰۰
ADD	AC ← AC + DR	۰۰۱
CLRAC	AC ← ۰	۰۱۰
INCAC	AC ← AC + ۱	۰۱۱
DRTAC	AC ← DR	۱۰۰
DRTAR	AR ← DR(۰-۱۰)	۱۰۱
PCTAR	AR ← PC	۱۱۰
WRITE	M[AR] ← DR	۱۱۱

توضیح	سمبل	BR
اگر شرط برابر ۱ باشد CAR ← AD اگر شرط برابر ۰ باشد CAR ← CAR + ۱	JMP	۰۰
اگر شرط برابر ۱ باشد CAR ← AD , SBR← CAR + ۱ اگر شرط برابر ۰ باشد CAR ← CAR + ۱	CALL	۰۱
بازگشت از زیرروال CAR ← SBR	RET	۱۰
CAR(۰,۱,۶), CAR(۲-۵) ← DR(۱۱-۱۴) ← ۰	MAP	۱۱

توضیح	سمبل	شرط	CD
انشعاب غیرشرطی	U	همیشه = ۱	۰۰
بیت آدرس غیرمستقیم	I	DR(۱۵)	۰۱
بیت علامت AC	S	AC(۱۵)	۱۰
بیت صفر در AC	Z	AC = ۰	۱۱

ریزدستورالعمل های سمبلیک

- ✓ سمبل‌ها در ریزدستورالعمل‌ها همانند زبان اسمبلی مورد استفاده قرار می‌گیرند.
- ✓ یک ریزبرنامه‌ی سمبلیک می‌تواند با استفاده از یک اسمبلرِ ریزبرنامه به معادل باینری آن ترجمه شود.

فرمت ریزدستورالعمل‌های سمبلیک: (شامل ۵ میدان می‌باشد)

label; micro-ops; CD; BR; AD

میدان label می‌تواند خالی باشد یا یک آدرس نمادین که با علامت : خاتمه می‌یابد را مشخص کند.

میدان micro-ops شامل یک، دو یا سه نماد است که با کاما جدا می‌شوند.

میدان CD شامل یکی از {U, I, S, Z} می‌تواند باشد.

میدان BR شامل یکی از {JMP, CALL, RET, MAP} می‌تواند باشد.

میدان AD شامل آدرس ریزدستورات است و می‌تواند یکی از {آدرس سمبلیک، NEXT و خالی} باشد.

label; micro-ops; CD; BR; AD

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

- ✓ به صورت آدرس سمبلیك که باید به فرم سرفصل نیز وجود داشته باشد.
- ✓ به فرم سمبلیك NEXT که نشان دهنده مراجعه به آدرس بعدی است.
- ✓ زمانی که فیلد BR شامل سمبل RET یا MAP است فیلد AD خالی است و با برنامه اسمبلر بـه هفـت عدد صفر تبدیل می شود.

label; micro-ops; CD; BR; AD

تعیین مقدار میدان CD به یکی از چهار راه زیر انجام می شود:

- ✓ U: پرش بدون قید و شرط (Unconditional Branch)
- ✓ I: بیت آدرس غیرمستقیم (Indirect address bit)
- ✓ S: علامت رجیستر AC (Sign of AC)
- ✓ Z: مقدار صفر در رجیستر AC (Zero value in AC)

- ✓ از شبه دستور ORG برای تعریف مبدأ یا اولین آدرس یك روال ریزبرنامه استفاده می‌شود.
- ✓ مثلاً سمبل ۶۴ ORG به اسمبلر اطلاع می دهد که ریزدستورالعمل بعدی را در حافظه کنتـرل در آدرس دهی ۶۴ قرار دهد که معادل آدرس باینری ۱۰۰۰۰۰۰ است.

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

- ✓ هنگام اجرای روتین Fetch، یک دستورالعمل از حافظه خوانده شده و دیکد می‌شود. همچنین مقدار PC آپدیت می‌شود.

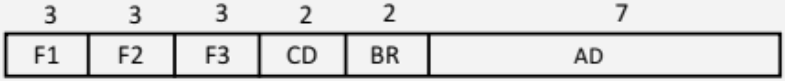
$AR \leftarrow PC$

$DR \leftarrow M[AR]$, $PC \leftarrow PC + 1$

$AR \leftarrow DR(0\text{-}10)$, $CAR(2\text{-}5) \leftarrow DR(11\text{-}14)$, $CAR(0,1,6) \leftarrow 0$

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

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



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

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	

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

razeghizade@gmail.com

Razeghizade.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution