معماری کامپیوتر

به نام خدا

تمرین سوم زبان انتقال ثبات، کدگذاری دستورات و مدهای آدرس دهی

1400/8/20

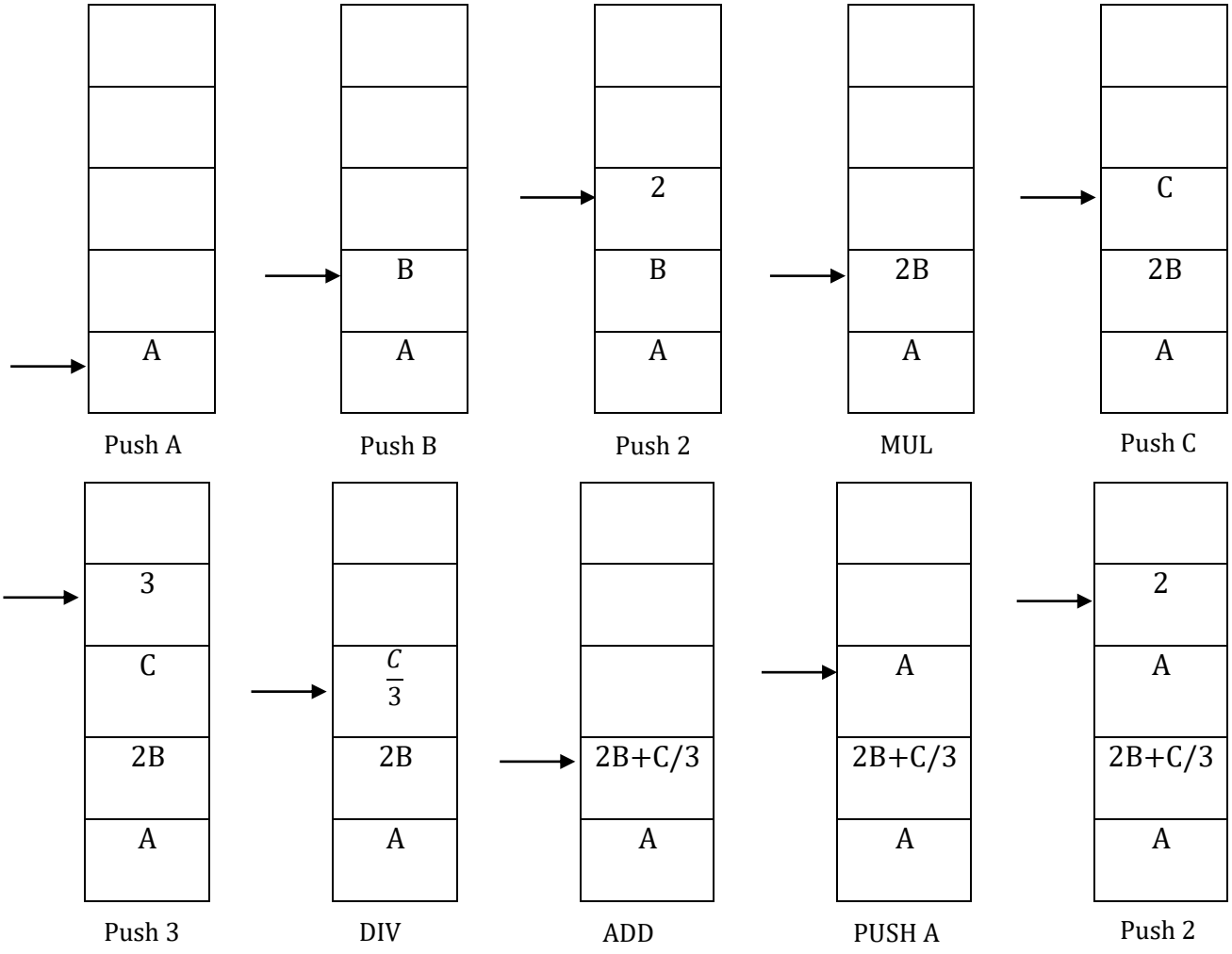
رضا صومی

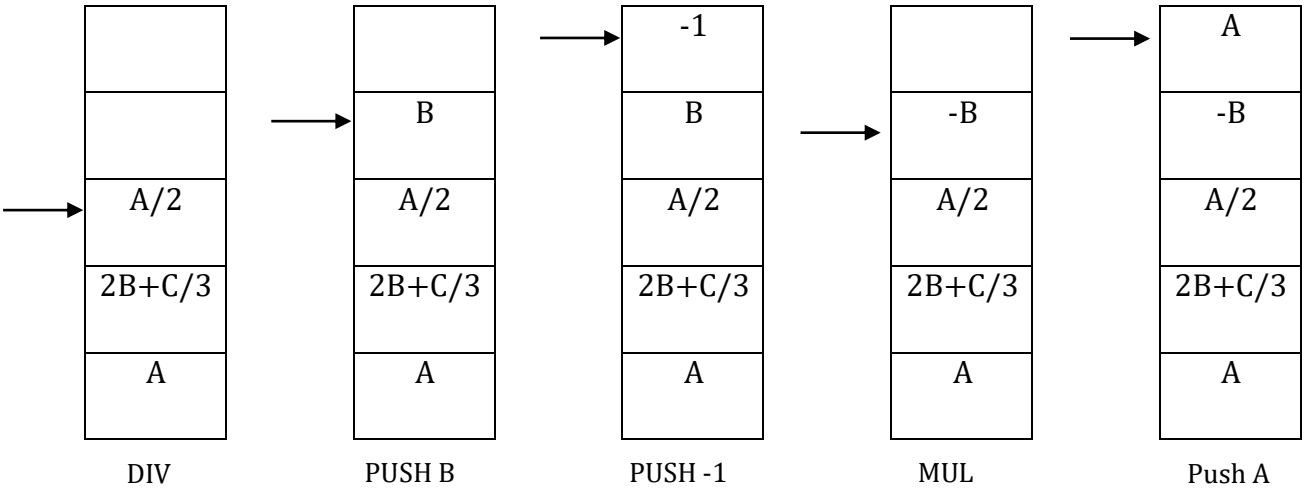
تمرین سوم زبان انتقال ثبات، کدگذاری دستورات و مدهای آدرس دهی

مفاهیم اولیه : 1.

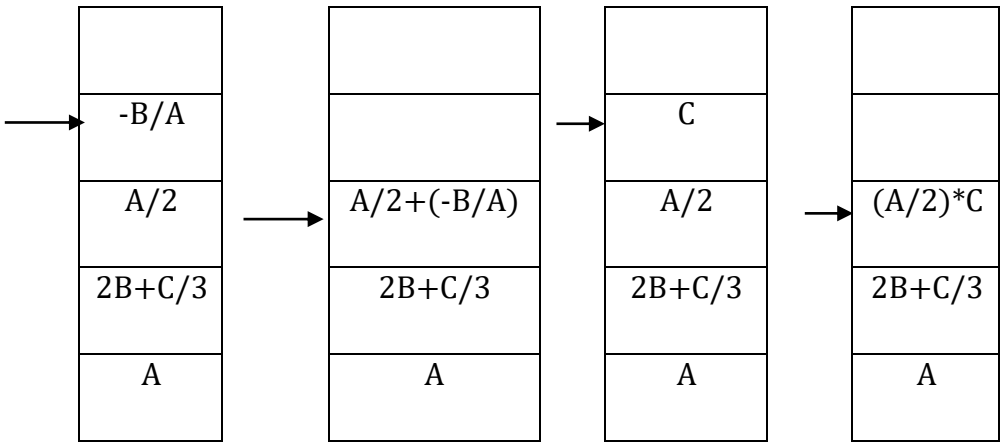
$$reverse\ polish\ notation\ of\ \ A-(2B+\frac{C}{3})\times(\frac{A}{2}+\frac{-B}{A})\times\rightarrow A\ B\ 2\ \times C\ 3\ \div\ +A\ 2\ \div B-1\ \times A\ \div\ +\ \times C\times\ -$$

2. مراحل به ترتیب از چپ به راست نمایش داده شده است.

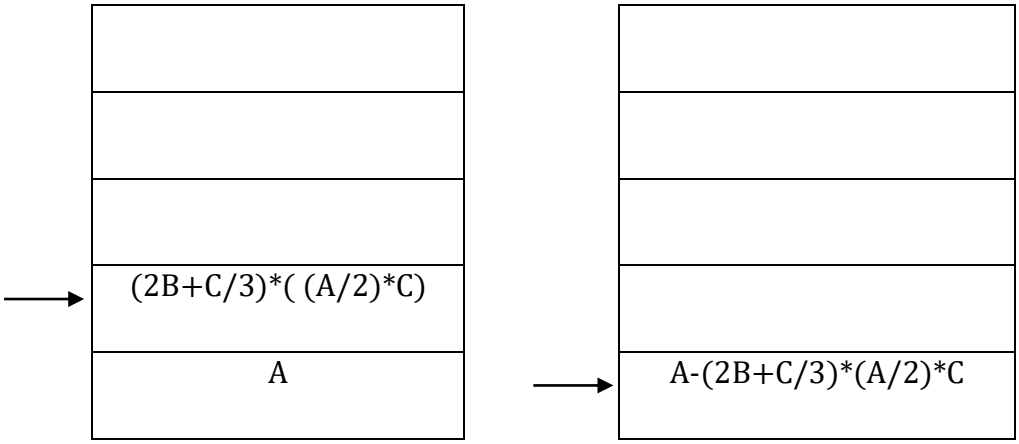




DIV PUSH B PUSH -1 MUL Push A



DIV ADD PUSH C MUL



MUL SUB

.3

اگر یک سیستم دارای طول دستورات ثابت باشد به این معناست که دستورهای موجود از قواعدی پیروی می کنند و تمامی اطلاعات را باید در آن طول ثابت قرار داد لذا با یک سیستم غیر انعطاف پذیر رو به رو هستیم و این به معنی این است که یک سری دستورات مشخص وجود دارد و برای اجرای دستورات پیچیده تر که در ISA این سیستم به طور مستقیم وجود ندارد نیاز به دو یا چند دستور در این سیستم است.(دستورات پیچیده تر به معنی داشتن اطلاعات بیشتر برای encode کردن آن است) این کار نه تنها باعث می شود کد برنامه طولانی تر شود بلکه باعث می شود ترافیک کاری بین memory و cpu نیز افزایش یابد و باعث کندی سیستم گردد. البته لازم به ذکر است که می توان طول دستورات موجود در سیستم را به اندازه کافی یا برای مثال به اندازه بزرگ ترین و پیچیده ترین دستور ممکن قرار داد که در این صورت برای یک دستوری که به تعداد بیت خیلی کم نیاز دارد نیز باید decode با بیت طولانی صورت بگیرد و این کار به شدت سرعت را کم می کند.

.4

این عدد به نوعی توان پردازشی cpu را نمایش می دهد. قلب کامپیوتر cpu است و همه اجزای کامپیوتر به نحوی به cpu متصل می شوند. هنگامی که می گوییم یک کامپیوتر 32 است به این معناست cpu توسط واحد های 32 بیتی با بقیه اجزا در ارتباط است. از نگاه بالاتر به این معناست که نرم افزار و سیستم عامل تنها با واحد های داده ای کار می کنند که طول آن 32 بیت است و نه بیشتر. حال وقتی cpu با 32 بیت کار می کند در نتیجه رنج اعدادی که شامل می شود برابر است با 2^{32} و در نتیجه می توان 2^{32} memory address در نظر گرفت که به عبارتی در کامپیوتر های 32 بیتی حداکثر رم ممکن برابر است با 4GB. همین تعاریف برای کامپیوتر های 64 بیتی نیز برقرار است و حداکثر مقدار رم در این کامپیوتر ها برابر می شود با 17,179,869,184GB !

RTL:

.1

هنگامی که F برابر صفر و S برابر یک باشد برنامه به نوعی استارت می خورد و در این مرحله عدد n در رجیستر R_1 قرار داده می شود و رجیستر R_2 مقدار صفر به خود می گیرد و F برابر یک می شود تا به نوعی دیگر این دستور اجرا نشود. n در ابتدا غیر صفر است در نتیجه دستور با شرط $F.OR(R_1)$ اجرا می شود چرا که $OR(R_1)$ به معنی or کردن تمامی بیت های R_1 است و وقتی n صفر نیست در نتیجه حداقل یک بیت R_1 مقدار 1 دارد. در این دستور $R_2 += 1$ می شود و R_1 یک بیت به راست شیفت داده می شود به عبارتی مقدار آن بر 2 تقسیم می شود و این دستور تا جایی انجام می پذیرد که تمامی بیت های R_1 صفر شده و $OR(R_1)$ برابر صفر شود. در این صورت دستور با شرط $F.\overline{OR(R_1)}$ اجرا می شود و R_3 مقدار رجیستر R_2 را می پذیرد و F مجددا صفر می شود تا دوباره این حلقه از

ابتدا تکرار شود. R_3 برابر می شود با کوچک ترین ضریب توان 2 که بزرگ تر از n است. به عبارتی کوچک ترین i برای عبارت $2^i > n$ است.

.2

در این سوال با یک کامپیوتر پشته ای یک آدرسه روبرو هستیم که از AC (accumulator register) برای ذخیره داده و عملیات ها روی آن استفاده می کند.

LD	A	$AC \leftarrow M[A]$
XOR	-1	XOR with 111...1
INC		$AC \leftarrow AC + 1$
ADD	B	$AC \leftarrow AC + M[B]$
STR	C	$M[C] \leftarrow AC$

در خط اول دستور LOAD اجرا شده و مقدار A در ثبات انباشتگر (AC) گذارده می شود. سپس این مقدار با 1- که در سیستم نمایش مکمل دو برابر با 111....1 است XOR می شود به عبارتی تمامی بیت های A نقیض می شوند و سپس در خط بعدی increment صورت می گیرد و مقدار موجود در AC با یک جمع می شود. در این سه خط به نوعی با نقیض کردن بیت های A و سپس جمع آن با 1 شاهد آن هستیم که A- در سیستم نمایش مکمل دو را ساخته ایم. حال این مقدار در خط بعد با مقدار B جمع زده می شود و نتیجه دوباره در ثبات قرار می گیرد. در انتها نیز دستور STORE را داریم که مقدار موجود در AC را در C که به یک آدرس در memory اشاره می کند قرار می دهیم. لذا این برنامه $C = B - A$ را انجام می پذیرد.

.3

$S: R_1 \leftarrow n, R_2 \leftarrow n, R_3 \leftarrow 2, F \leftarrow 0, S \leftarrow 0$

$(R_2 > 0).\bar{F}: R_2 \leftarrow R_2 - R_3$

پاسخ آقای احدی نیا فرمت بهتری دارد

$(R_2 < 0).\bar{F}: R_3 \leftarrow R_3 + 1$

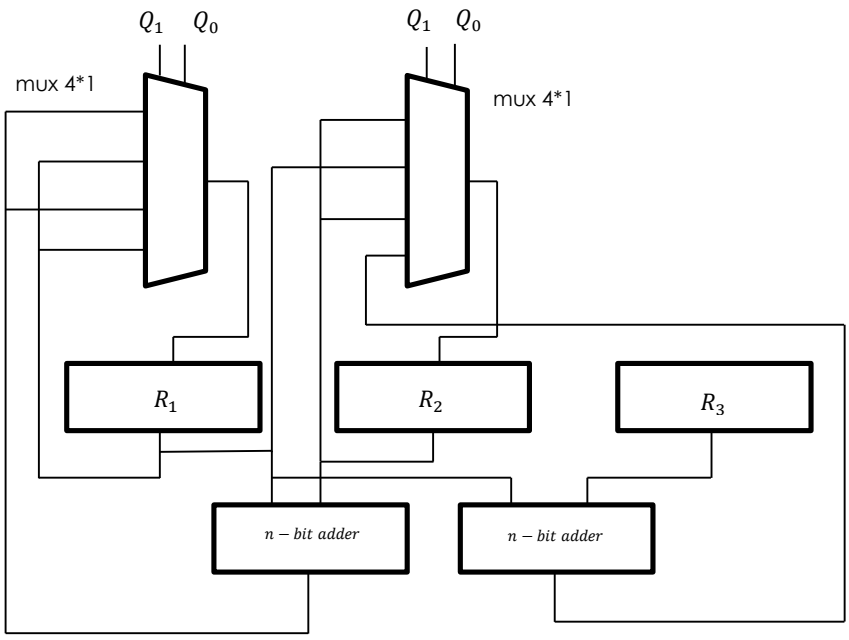
$R_2 == 0: R_4 \leftarrow 1, F \leftarrow 1$

$R_2 == R_1 - 1: R_4 \leftarrow 0, F \leftarrow 1$

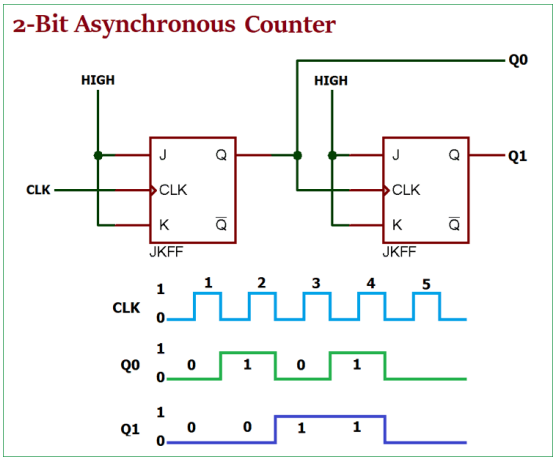
برای اینکه چک کنیم ورودی n اول است یا خیر چک می کنیم این عدد بر اعداد کمتر از خود بخش پذیر است یا خیر که در اینجا از عدد 2 چک شروع می شود و به طور پیوسته عدد n منهای عدد 2 می شود اگر به 0 رسیدیم در نتیجه این عدد بر دو بخش پذیر است اگر به 0 نرسیدیم و حاصل تفریق ها در نهایت منفی شد در نتیجه بر دو بخش پذیر نیست. این پیمایش را برای اعداد 2 تا $n - 2$ تکرار می کنیم

(البته به منظور کارایی می توان تا مقدار $\frac{n}{2}$ را چک کرد) اگر در این حین به صفر رسیدیم در نتیجه عدد اول نیست و خروجی را یک کرده و اگر هیچکدام بخش پذیر نبود آنگاه مقدار خروجی را 0 کرده و F را به منظور اتمام برنامه 1 می کنیم.

.4



Load تمامی رجیستر ها همیشه فعال است و ورودی را به خروجی منتقل می کند. خطوط select مالتی پلکسر یک شمارنده ای باید باشد که از 0 تا 3 را پیمایش کند و این فرآیند را تا بی نهایت ادامه دهد. برای این کار از 2 bit asynchronous counter استفاده می کنیم که پیاده سازی آن را در تصویر زیر مشاهده می کنید.



کدگذاری دستورات :

.1

حداکثر 32 بیت.

حجم حافظه 64 گیگابایت است در نتیجه 36 بیت برای آدرس دهی و دسترسی به memory نیاز داریم.

کلمات این ماشین 4 واحد آدرس پذیر هستند در نتیجه یک ردیف از حافظه که پردازنده می تواند آن را یکجا بخواند برابر 4 دستور است که هر کدام آدرس جدایی دارند. در نتیجه با توجه به اینکه هر بایت در حافظه آدرس مجزا دارد پس کلمات 4 بایتی هستند.

قالب دستورات یک کلمه ای (در اینجا 32 بیت) و دو کلمه ای (در اینجا 64 بیت) است.

مد های آدرس دهی آن مستقیم ثباتی و حافظه ای و این ماشین دو عملوندی است. در نتیجه دستورات دو کلمه ای باید یک عملوند را از حافظه خوانده و یک عملوند را از ثبات و همچنین دستورات یک کلمه ای باید دو عملوند را از رجیستر دریافت کند چرا که برای آدرس دهی حافظه 36 بیت نیاز داریم و دستورات یک کلمه ای تنها 32 بیت قابلیت انتقال دارند.

همچنین یک بیت برای تشخیص یک کلمه ای یا دو کلمه ای بودن دستورات در نظر می گیریم.

در نتیجه با توجه به اینکه تعداد دستورات یک کلمه ای نصف تعداد دستورات دو کلمه ای است لذا داریم :

$$\frac{2^{32-1-2n}}{2^{64-36-1-n}} = \frac{1}{2} \rightarrow n = 5$$

در نتیجه برای آدرس دهی رجیستر ها 5 بیت نیاز داریم و با توجه به این عدد حداکثر تعداد ثبات های همه منظوره برابر 32 خواهد بود.

.2

پاسخ صحیح توسط آقای علیپور : با توجه به اینکه طول دستورات 4n بیت و طول همه‌ی عملوند ها n بیت است، پس اندازه فیلد opcode در قالب دستورات قالب صفر عملوندی، تک عملوندی و سه عملوندی به ترتیب 4n، 3n و n بیت می باشد. بخشی از بیت های فیلد opcode بین قالب های مختلف مشترک است، برای این که حداکثر تعداد دستورات را تعیین کنیم باید از تعداد حالاتی بخش مشترک، بیش ترین را به قالبی اختصاص دهیم که بخش غیر مشترک بزرگ تری دارد و باعث می شود عدد بزرگ تری در تعداد حالات بخش مشترک ضرب شود. بین هر سه قالب، n بیت اول مشترک است، چون فیلد opcode قالب صفر عملوندی از بقیه بزرگ تر است از 2^n حالت کدی که بخش مشترک می تواند ایجاد کند، $2^n - 2$ را به قالب صفر عملوندی اختصاص می دهیم و به هر یک از قالب های یک عملوندی و دو عملوندی یک کد اختصاص می دهیم (حداقل باید یک کد اختصاص چون اگر صفر باشد آن نوع قالب حذف می شود). با این تخصیص هر نوع قالب بر

اساس n بیت اول برای پردازنده قابل تشخیص می باشد. بنابراین در کل یک کد برای قالب دستورات سه عملوندی، 1×2^{2n} کد برای قالب دستورات تک عملوندی و $(2^n - 2)2^{3n}$ کد برای قالب دستورات صفر عملوندی خواهیم داشت که در مجموع تعداد دستورات به صورت زیر خواهد بود:

$$(2^n - 2)2^{3n} + 2^{2n} + 1 = 2^{4n} - 2^{3n+1} + 2^{2n} + 1$$

.3

112 کد مختلف برای opcode در نظر گرفته شده است. برای 112 کد به 7 بیت برای opcode نیاز داریم. اندازه حافظه 4 مگابایت است در نتیجه برای آدرس دهی به یک بایت از حافظه با توجه به این حجم حافظه به 22 بیت نیاز داریم. با توجه به اینکه آدرس دهی به یک خانه حافظه از طریق ثبات صورت می گیرد لذا حداقل تعداد بیت های ثبات ها 22 است. قالب دستورات 3 آدرسه است لذا به سه آدرس نیاز داریم و سه رجیستر. از آنجایی که تعداد ثبات ها 32 است لذا با 5 بیت می توان رجیستر مورد نظر را انتخاب کرد و چون 3 رجیستر نیاز داریم در نتیجه به 15 بیت برای این قسمت نیاز داریم. در نتیجه 15 بیت برای ثبات ها و 7 بیت برای opcode روی هم برابر می شود با 22

.4

اگرچه این پاسخ هم پذیرفته است ولی پاسخ درست را اقای احدی نیا نوشتند

با توجه به اینکه دو نوع آدرس دهی مستقیم و آنی وجود دارد و تعداد بیت های لازم برای آدرس دهی مستقیم برابر MAR=13bit است و همچنین تعداد بیت در نظر گرفته شده برای مقدار immediate برابر AC=12bit است و این دو برابر نیستند لذا برای اینکه حداکثر دستورات ممکن را بدست آوریم یک بیت برای تشخیص نوع دستور (مستقیم یا آنی) قرار داده و برای هر کدام از نوع های دستورات تعداد آن را جداگانه حساب می کنیم: (IR: instruction register)

$$immediate: 2^{32-13-12-1} = 2^6$$

$$direct: 2^{32-13-13-1} = 2^5$$

$$total = 2^5 + 2^6$$

.5

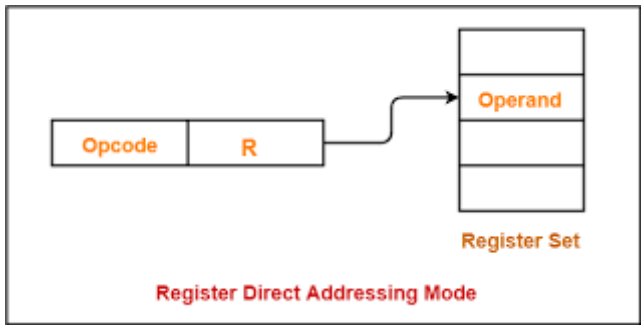
هر کلمه 8 بیتی است. تعداد دستورات یک کلمه ای این پردازنده 56 است لذا برای کدگذاری آن 6 بیت نیاز داریم. این پردازنده دو آدرسه است لذا در این حالت که مد آدرس دهی ثباتی و غیر مستقیم ثباتی است باید در این 2 بیت دو رجیستر را مشخص کنیم اما یک بیت نیز

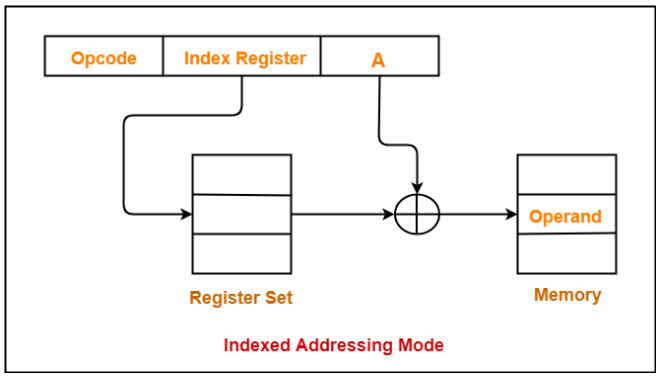
برای تشخیص یک کلمه ای یا سه کلمه ای بودن دستور نیاز داریم لذا تنها یک بیت باقی می ماند. با این یک بیت می توان یک رجیستر را انتخاب کرد و چون در این حالت تنها دو رجیستر می توان در اختیار داشت لذا ثبات دیگر نیز بدون آدرس دهی مشخص می شود. در نتیجه 2 ثبات موجود است. می دانیم حجم حافظه مورد استفاده 64 کیلو بایت است لذا برای آدرس دهی حافظه 16 بیت نیاز داریم در نتیجه چون دستورات سه کلمه ای 24 بیت گنجایش دارند لذا یکی از آدرس ها تنها می تواند از حافظه باشد و دیگری مشخص کننده ثبات است. حال 1 بیت برای آدرس دهی رجیستر، 1 بیت برای تشخیص یک کلمه ای یا سه کلمه ای بودن دستور و 16 بیت برای آدرس دهی حافظه نیاز است که روی هم برابر می شود با 18 بیت. در نتیجه 6 بیت باقی می ماند که با این 6 بیت می توان حداکثر 64 دستور سه کلمه ای اختیار کرد.

6. گزینه ج

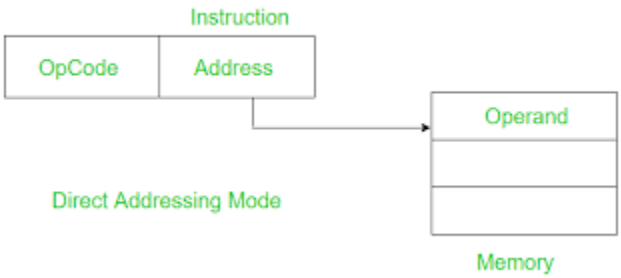
توضیحات این قسمت مناسب است ولی پاسخ درست را بنده در پاسخ منتخب دوم قرار دادم

Immediate Addressing Mode

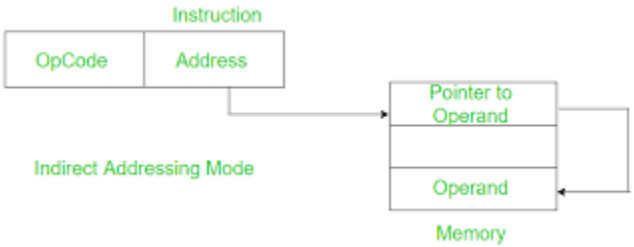




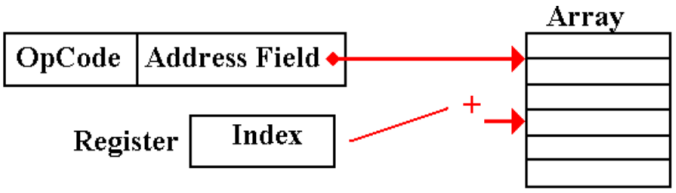
به سه مد آدرس دهی بالا دسترسی داریم.



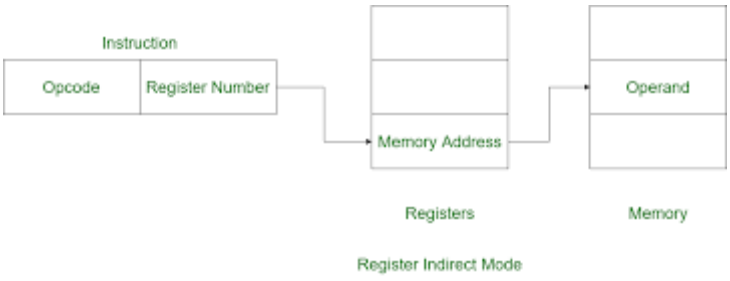
درست است که این مورد مشابه immediate addressing است اما ممکن است تعداد بیت آدرس دهی حافظه بیشتر باشد لذا به direct addressing را در اختیار نداریم.



به indirect addressing mode دسترسی داریم. به تصویر زیر که همان index addressing را به مدل بهتری نمایش می دهد توجه کنید. Address field به اندازه ای هست که بتوان memory را آدرس دهی کرد. لذا این مورد در دسترس است.



به *register indirect* نیز دسترسی نداریم چرا که نمی دانیم ثبات ها چند بیتی است و آیا با آن می توان خانه های حافظه را آدرس دهی کرد یا خیر.



با توجه به شباهت موجود بین تصویر زیر که آدرس دهی نسبی را نمایش می دهد و index addressing mode می توان با قرار دادن PC در index register و اضافه کردن rs و rt به کمک محدود کردن address field به مد آدرس دهی نسبی دست یافت.