تمرین سوم درس معماری کامپیوتر

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

آرین احدی نیا

شماره دانشجویی: [illegible]

استاد درس: جناب آقای دکتر جهانگیر

دستیار آموزشی: جناب آقای علی‌پور

پاییز ۱۴۰۰

فهرست عناوین

مفاهیم اولیه

سوال ۱

توجه بفرمایید که الویت محاسبه عملگرها به ترتیب زیر میباشد.

۱. پرانتز
۲. توابع
۳. نما و جذر
۴. ضرب و تقسیم
۵. جمع و تفریق

میان نما و جذر، ضرب و تقسیم و همچنین جمع و تفریق، الویت از سمت چپ به راست مشخص میشود.

به ترتیب همین الویت گذاری عبارات را انتخاب و آنها را به صورت پسوندی مینویسیم. در روابط زیر، رنگ قرمز نشان‌دهنده این است که آن عبارت هنوز به عبارت پسوندی تبدیل نشده است.

$$A - \left(2B + \frac{C}{3}\right) \times \left(\frac{A}{2} + \frac{-B}{A}\right) \times C = A - \left((2\ B\ \times) + (C\ 3\ \div)\right) \times \left((A\ 2\ \div) + \frac{(-1\ B\ \times)}{A}\right) \times C$$

$$= A - (2\ B\ \times C\ 3\ \div\ +) \times \left((A\ 2\ \div) + (-1\ B\ \times A\ \div)\right) \times C$$

$$= A - (2\ B\ \times C\ 3\ \div\ +) \times (A\ 2\ \div (-1)\ B\ \times A\ \div +) \times C$$

$$= A - (2\ B\ \times C\ 3\ \div\ + A\ 2\ \div (-1)\ B\ \times A\ \div + \times) \times C$$

$$= A - (2\ B\ \times C\ 3\ \div\ + A\ 2\ \div (-1)\ B\ \times A\ \div + \times C\ \times)$$

$$= A\ 2\ B\ \times C\ 3\ \div\ + A\ 2\ \div (-1)\ B\ \times A\ \div + \times C\ \times -$$

سوال ۲

توجه بفرمایید که ماشین پشته‌ای دقیقا دستورات را به صورت پسوندی اجرا میکند. به این صورت که حافظه اجرای دستورات یک پشته است. در صورت اعمال یک اپراتور، عملوندهای آن از پشته دریافت می‌شوند و نتیجه در پشته درج میگردد. تنها نکته قابل توجه در این نوع ماشین این است که اگر اپراتور مورد نظر یک اپراتور دو عملوندی باشد، مقدار اولی که پشته دریافت میشود برابر عملوند دوم و مقدار بعدی برابر عملوند اول خواهد بود. در جدول زیر عملیات‌های لازم برای محاسبه چنین عبارتی در ماشین پشته‌ای را به همراه وضعیت پشته پس از اجرا هر مرحله، نوشته‌ایم.

** Operation Start **	-- empty --
PUSH A	A
PUSH 2	$A, 2$
PUSH B	$A, 2, B$
MUL	$A, 2B$
PUSH C	$A, 2B, C$
PUSH 3	$A, 2B, C, 3$
DIV	$A, 2B, \frac{C}{3}$
ADD	$A, 2B + \frac{C}{3}$
PUSH A	$A, 2B + \frac{C}{3}, A$
PUSH 2	$A, 2B + \frac{C}{3}, A, 2$
DIV	$A, 2B + \frac{C}{3}, \frac{A}{2}$
PUSH 1	$A, 2B + \frac{C}{3}, \frac{A}{2}, 1$
COMPLEMENT	$A, 2B + \frac{C}{3}, \frac{A}{2}, -1$
PUSH B	$A, 2B + \frac{C}{3}, \frac{A}{2}, -1, B$
MUL	$A, 2B + \frac{C}{3}, \frac{A}{2}, -B$
PUSH A	$A, 2B + \frac{C}{3}, \frac{A}{2}, -B, A$
DIV	$A, 2B + \frac{C}{3}, \frac{A}{2}, \frac{-B}{A}$
ADD	$A, 2B + \frac{C}{3}, \frac{A}{2} + \frac{-B}{A}$
MUL	$A, \left(2B + \frac{C}{3}\right) \times \left(\frac{A}{2} + \frac{-B}{A}\right)$
PUSH C	$A, \left(2B + \frac{C}{3}\right) \times \left(\frac{A}{2} + \frac{-B}{A}\right), C$
MUL	$A, \left(2B + \frac{C}{3}\right) \times \left(\frac{A}{2} + \frac{-B}{A}\right) \times C$
SUB	$A - \left(2B + \frac{C}{3}\right) \times \left(\frac{A}{2} + \frac{-B}{A}\right) \times C$

در نهایت همانگونه که ملاحظه میفرمایید حاصل در پشته قرار گرفته است. اگر بخواهیم این حاصل را در خانه X حافظه قرار دهیم، میتوانیم با دستور X POP این کار را انجام دهیم. توجه بفرمایید که در نگارش فوق، دستور PUSH و POP به ترتیب معادل LOAD و STORE هستند.

سوال ۳

مشکل عمده که به ذهن بنده میرسد، عدم انعطاف‌پذیری است. این عدم انعطاف‌پذیری باعث می‌شود که برخی دستورات بیش از نیاز بیت در دسترس داشته باشند و برخی دیگر کمتر از نیاز بیت در دسترس داشته باشند. اجازه بفرمایید که موضوع را با یک مثال توضیح دهم.

پردازنده میپس را در نظر بگیرید. این پردازنده سه نوع دستور R-Type، J-Type و I-Type دارد. در دستورات R-Type، سه آدرس رجیستر و یک مقدار برای Shift amount در نظر گرفته شده است. در بسیاری از موارد حداقل یکی از این چهار مقدار خالی است.

اما از سوی دیگر برای آدرس‌دهی به حافظه تنها ۲۶ بیت در دسترس است که مجبور میشویم با روش‌هایی مانند alignment و PC-relative addressing، مقدار مورد نظر را در حافظه بگنجانیم.

بنابرین در مواردی بیش از نیاز و در مواردی کمتر از نیاز بیت برای کدگذاری دستور در اختیار داریم.

سوال ۴

در کلی ترین تعریف، کامپیوترهای ۶۴ بیتی، دارای آدرس حافظه، واحدهای محاسبه و Data Bus ۶۴ بیتی هستند در حالی که این موارد برای کامپیوتر ۳۲ بیتی، ۳۲ بیتی هستند. این افزایش عرض باعث مزایای بسیاری میشود از جمله اینکه حافظه در کامپیوتر ۳۲ بیتی به ۴ گیگابایت محدود میشود اما در کامپیوتر ۶۴ بیتی این حافظه حدود ۱۰ میلیون برابر این مقدار است. همچنین محاسبات با سرعت بسیار بیشتر انجام میشود و امکان موازی‌سازی نیز وجود خواهد داشت.

منابع:

Wikipedia

GeeksForGeeks

RTL

سوال ۱

توجه بفرمایید که شیفت به راست، معادل تقسیم صحیح است. تقسیم صحیح به این صورت است که خارج قسمت به عنوان حاصل برگردانده می‌شوند و باقی‌مانده نادیده گرفته می‌شود.

در شروع عملیات یک عدد در رجیستر R_1 قرار میگیرد و R_2 مقدار اولیه صفر را دریافت می‌کند همچنین سیگنال F برای کنترل $flow$ فعال میشود تا دو خط بعدی از این پس اجرا شوند.

مادامی که $R_1 \neq 0$ باشد، $OR(R_1)$ برابر ۱ خواهد بود. در این حین R_1 تقسیم صحیح بر دو می‌شود و یک واحد به مقدار R_2 اضافه میشود. این اتفاق تا زمان صفر شدن R_1 ادامه پیدا می‌کند. زمانی که R_1 صفر شود، مقدار R_2 در R_3 قرار میگیرید و flow کد پایان می‌یابد.

توجه کنید که از آنجایی که تقسیم به صورت متوالی انجام میشود، اگر m بار تقسیم انجام شود گویا که یکبار تقسیم بر 2^m انجام شده است. برای اینکه حاصل این تقسیم صفر شود، خواهیم داشت،

$$R_1(int)/2^m = 0 \Rightarrow 2^m \geq R_1 \Rightarrow m \geq \log R_1$$

بنابرین اولین جایی که تقسیم صفر می‌شود، پس از $\lceil \log R_1 \rceil$ مرحله است. که در حقیقت به این معنی است که حاصل نهایی R_3، برابر $\log n$ خواهد بود.

سوال ۲

ابتدا مقدار A از حافظه بارگیری و در پشته قرار میگیرد. سپس این مقدار از پشته خارج میشود و اپراتور XOR بر این مقدار و 1- به صورت bitwise اعمال می‌شود و حاصل در پشته قرار میگیرد. توجه کنید که در نمایش مکمل دوم مقدار 1- در حافظه برابر 111 ... 111 است. همچنین توجه بفرمایید که اگر یکی از ورودی‌های XOR برابر 1 باشد، این اپراتور نسبت به ورودی دیگر به مانند NOT عمل خواهد کرد. به عبارت دیگر XOR به مانند یک گیت NOT همراه با ورودی کنترلی است. بنابرین حاصل اعمال XOR برابر $\sim A$ خواهد شد. سپس این مقدار از پشته خارج شده، یک واحد به این مقدار اضافه میشود و حاصل در پشته قرار میگیرد. که این حاصل برابر $\sim A + 1$ می‌شود. توجه بفرمایید که این مقدار برابر مکمل دوم عدد A است. سپس در مرحله بعدی این مقدار با B جمع میشود. دقت بفرمایید که جمع شدن عدد B با مکمل دوم A معادل $B - A$ خواهد بود. در نهایت این مقدار در حافظه C ذخیره می‌شود. در نهایت حاصل این عملیات برابر

$$C = B - A$$

خواهد شد.

سوال ۳

$$\begin{aligned}
S.T &: R_1 \leftarrow n, R_3 \leftarrow 1, T \leftarrow 0, F \leftarrow 0 \\
F'.T' &: R_2 \leftarrow R_1, R_3 \leftarrow R_3 + 1, F \leftarrow 1 \\
G(R_1, R_3).G(R_2, R_3).F.T' &: R_2 \leftarrow R_2 - R_3 \\
G(R_1, R_3).E(R_2, R_3).F.T' &: P \leftarrow 0, T \leftarrow 1 \\
G(R_1, R_3).L(R_2, R_3).F.T' &: F \leftarrow 0 \\
G(R_1, R_3)'.F.T' &: P \leftarrow 1, T \leftarrow 1
\end{aligned}$$

سیگنال کنترلی S همان سیگنال Start است. T سیگنال Terminate است. زمانی که این سیگنال فعال شود، به این معنی است که پردازش به پایان رسیده. سیگنال P مشخص کننده این است که عدد مورد نظر اول است یا خیر و سیگنال F برای کنترل flow اجرای برنامه است. منطق اجرای این برنامه به این صورت است که از سه رجیستر استفاده می‌کنیم.

- R_1 که همواره محتوی عدد اصلی است.
- R_2 که در هر مرحله کپی R_1 را بر روی آن قرار می‌دهیم و با تفریق متوالی عملیات تقسیم را انجام می‌دهیم.
- R_3 که مقسوم‌علیه در یک مرحله را شامل میشود.

در هر مرحله ابتدا یک واحد به مقسوم‌علیه اضافه میکنیم و کپی R_1 را در R_2 میریزیم. سپس در مرحله بعد، به ۴ طریق عمل میکنیم.

- اگر مقسوم‌علیه بزرگتر یا مساوی عدد اصلی شود، به این معنی است که عدد به هیچ یک از اعداد کوچکتر بخش‌پذیر نبوده، بنابرین اعلام می‌کنیم که عدد اول است و پردازش را خاتمه می‌دهیم.
- در غیر این صورت:
  - اگر باقی‌مانده بزرگ‌تر از مقسوم‌علیه باشد، مقسوم‌علیه را برای پیش بردن تقسیم یک بار از باقی‌مانده کم می‌کنیم.
  - اگر باقی‌مانده برابر مقسوم‌علیه شود، به این معنی است که عدد مورد نظر بر مقسوم‌علیه بخش پذیر است. بنابرین پردازش را مختومه می‌کنیم و اعلام می‌کنیم عدد مورد نظر اول نیست.

- اگر باقی‌مانده کوچک‌تر از مقسوم‌علیه شود، به این معنی است که عدد مورد نظر بر این مقسوم‌علیه بخش‌پذیر نیست. بنابرین به ابتدای حلقه بر می‌گردیم.

سوال ۴

می‌توانیم دستور دوم را به شکل $R_2 = R_1 + 0$ در نظر بگیریم. در این صورت تمام دستورالعمل‌ها به فرمت جمع خواهند بود.

$R_1 \leftarrow R_1 + R_2$

$R_2 \leftarrow R_1 + 0$

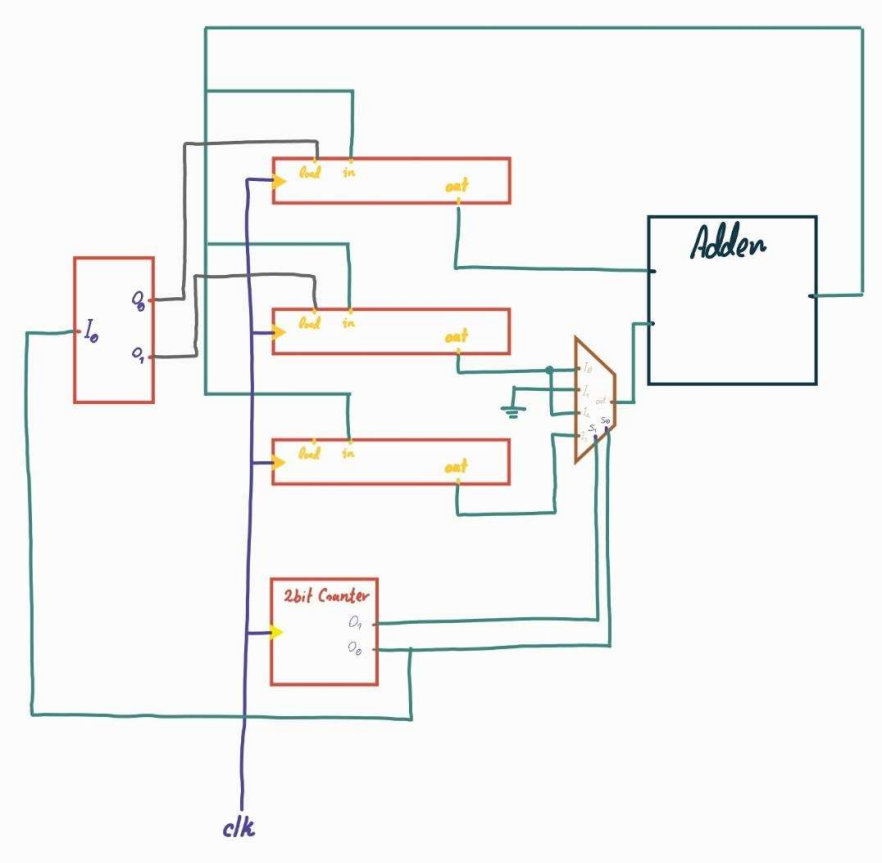
$R_1 \leftarrow R_1 + R_2$

$R_2 \leftarrow R_1 + R_3$

برای مشخص کردن اینکه اکنون در چه timestep هستیم، از یک counter دو بیتی استفاده میکنیم که بتواند از صفر تا سه بشمارد. برای هر یک از R_1، R_2 و R_3 نیز یک رجیستر n بیتی در نظر میگیریم. برای انجام عملیات یک Adder با اندازه n در نظر میگیریم.

همانگونه که ملاحظه می‌فرمایید همواره یکی از operandهای جمع برابر R_1 است بنابرین R_1 را مستقیما به Adder متصل میکنیم. برای انتخاب عملوند دیگر از یک $4:1$ MUX با عرض بیتی n استفاده میکنیم به این صورت که خروجی counter را که مشخص کننده زمان فعلی است، به ورودی انتخابی متصل می‌کنیم و به ورودی‌های MUX به ترتیب $I_0 = R_2$، $I_1 = 0$، $I_2 = R_2$ و $I_3 = R_3$ وصل میکنیم.

برای ذخیره خروجی ها، از یک رمزگشا استفاده می‌کنیم. به این صورت که بیت کم‌ارزش خروجی Counter را به ورودی آن وصل میکنیم. خط اول خروجی را به ورودی $Load$ ثبات R_1 و خط دوم را به ورودی $Load$ ثبات R_2 متصل میکنیم. و هم چنین خروجی جمع‌کننده را به ورودی موازی تمام ثبات‌ها متصل می‌کنیم. در نهایت مدار به شکل زیر خواهد شد.


load
in
out
Adder
2bit Counter
clk

کدگذاری دستورات

این پاسخ فقط با دیدگاه حداکثر کردن تعداد ثبات ها نوشته شده است. اگرچه پاسخ مورد انتظار، پاسخ اقای صومی است اما این تفکر هم می تواند جالب باشد

سوال ۱

توجه بفرمایید که ۶۴ گیگابایت معادل 64×2^{30} بایت است که معادل 2^{36} بایت می‌شود.

از آنجایی که هر بایت در حافظه آدرس‌پذیر است و کلمات چهار آدرس پذیر هستند، کلمات این پردازنده ۴ بایتی هستند.

از آنجایی که هر بایت در حافظه آدرس‌پذیر است و حافظه شامل 2^{36} بایت می‌شود، بنابرین برای آدرس‌دهی حافظه نیاز به ۳۶ بیت داریم.

چون دستورالعمل‌ها یک یا دوکلمه‌ای است و کلمه‌ها ۳۲ بیتی هستند، طول دستورالعمل‌های این ماشین برابر ۳۲ و ۶۴ بیت خواهد بود.

اگر 2^n رجیستر داشته باشیم، برای آدرس‌دهی رجیسترها نیاز به n بیت خواهیم داشت. توجه کنید برای اینکه تعداد رجیسترها بیشینه شود، باید تعداد آنها توانی از دو باشد در غیر این صورت برخی از ترکیب آدرس‌ها بی‌استفاده خواهند بود.

از آنجایی این پردازنده آدرس‌دهی مستقیم به حافظه دارد، بنابرین باید آدرس حافظه در دستورالعمل قرار بگیرد. از آنجایی که طول آدرس حافظه ۳۶ بیت است، بنابرین تنها در دستورالعمل‌های ۶۴ بیتی میتوانیم به حافظه آدرس دهی کنیم و تنها به یک خانه از حافظه می‌توانیم آدرس‌دهی انجام دهیم.

بنابرین در این پردازنده دستورات به دو شکل خواهند بود.

۱. ۶۴ بیتی: که شامل ۳۶ بیت آدرس حافظه و n بیت آدرس ثبات و OPCODE میشوند.
۲. ۳۲ بیتی: که شامل $2n$ بیت آدرس ثبات و OPCODE میشود.

بنابرین بیشترین مقدار قابل فرضی که میتوانیم برای n در نظر بگیریم، ۱۵ است. بنابرین OPCODE دستورات ۳۲ بیتی ۲ بیت خواهد بود و ۴ ترکیب مجاز خواهیم داشت. اگر از سه تا از این ترکیب‌های مجاز استفاده کنیم و یک ترکیب را برای ایجاد تمایز با دستورات ۶۴ بیتی حفظ کنیم، در دستورات ۶۴ بیتی ۱۳ بیت برای OPCODE خواهیم داشت که برای دو بیت ابتدایی آن تنهای یک ترکیب مجاز داریم. بنابرین 2^{11} ترکیب مجاز مختلف برای OPCODE میتوانیم داشته باشیم. اگر از ۶تای آنها استفاده کنیم، تعداد دستورات یک کلمه ای نصف دوکلمه‌ای ها خواهد بود.

سوال ۲

راه حل ساده تر را بنده در پاسخ منتخب اول نوشتم. اما این راه حل هم بسیار جالب است

طبق صورت سوال، دستورات این کامپیوتر به شکل زیر است.

4n-bit opcode			
3n-bit opcode			n-bit operand
n-bit opcode	n-bit operand	n-bit operand	n-bit operand

برای نوع سه‌عملوندی، حداکثر میتوانیم 2^n دستور مختلف از ترکیب مختلف opcodeها در نظر بگیریم. اما در این صورت راه تمایزی میان این نوع دستورالعمل و سایر دستورالعمل‌ها نخواهد بود. بنابرین نیاز است تا تعدادی از ترکیبات ممکن را برای سایر دستورالعمل‌ها کنار بگذاریم. اگر در نظر بگیریم که تعداد این ترکیب‌ها x باشد، $2^n - x$ دستور از نوع سه عملوندی خواهیم داشت.

برای دستورات تک عملوندی، $3n$ بیت برای opcode در اختیار داریم. توجه کنید که x ترکیب مجاز برای n بیت ابتدایی داریم و $2n$ بیت دیگر را می‌توانیم به دلخواه انتخاب کنیم. بنابرین $2^{2n}x$ ترکیب مختلف برای این حالت وجود دارد. همانگونه که گفتیم، برای اینکه تمایز بتوانیم میان دستورات از انواع مختلف قائل شویم، باید تعدادی از ترکیب‌ها را کنار بگذاریم. اگر در نظر بگیریم که تعداد این ترکیب‌ها y باشد، $2^{2n}x - y$ دستور از نوع تک عملوندی خواهیم داشت.

بنابر استدلال مشابه، تعداد دستورات صفر عملوندی برابر $2^n y$ خواهد بود. بنابرین مجموع تعداد دستورات برابر خواهد بود با

$$f(x,y) = (2^n - x) + (2^{2n}x - y) + (2^n y)$$

تعداد دستورات را تابعی از x و y در نظر میگیریم چراکه مقدار n ثابت است. اگر گرادیان این تابع را محاسبه کنیم، خواهیم داشت

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2^{2n} - 1 \\ 2^n - 1 \end{bmatrix}$$

بدیهی است که مقدار n باید بزرگتر از صفر باشد. در این صورت تمام مشتق‌های پاره‌ای تابع f در سراسر بازه اعداد حقیقی مثبت خواهند بود. در این صورت چون تابع f نسبت به این دو متغیر اکیدا صعودی است، هر چه مقدار x و y بزرگتر باشد، حاصل نهایی نیز بزرگتر خواهد بود. بنابرین مقدار x و y را بزرگترین مقدار ممکن انتخاب می‌کنیم. توجه کنید که حداقل یک دستور از هر نوع باید داشته باشیم.

$$\begin{cases} x = 2^n - 1 \\ y = 2^{2n}x - 1 \end{cases} \Rightarrow \begin{cases} x = 2^n - 1 \\ y = 2^{2n}(2^n - 1) - 1 \end{cases}$$

بنابرین در این حالت ۱ دستور از نوع سه عملوندی و یک دستور از نوع تک عملوندی و $2^n y$ دستور از نوع صفر عملوندی خواهیم داشت که برابر خواهد بود با

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

بنابرین در مجموع، $2^{4n} - 2^{3n} - 2^n + 2$ دستور مختلف خواهیم داشت.

سوال ۳

OPCODE	REG1	REG2	REG3

اندازه حافظه این کامپیوتر، 4×2^{20} بایت است که معادل 2^{22} بایت است. بنابرین برای آدرس‌دهی به این حافظه باید دست‌کم ۲۲ بایت داشته باشیم. از آنجایی که این آدرس‌دهی از طریق یک رجیستر انجام می‌شود و آدرس در رجیستر قرار می‌گیرد، بنابرین حداقل طول رجیستر باید ۲۲ باشد.

از آنجایی که آدرس‌دهی با حافظه از طریق یک ثبات‌ها انجام می‌شود، پس تمام عملوندهای دستورالعمل ثبات‌ها هستند. چون ۳۲ عدد ثبات داریم، برای مشخص کردن یک ثبات نیاز به ۵ بیت خواهیم داشت.

اگر طول دستورالعمل‌ها ۲۲ بیت باشد و شماره ۳ رجیستر در آن مشخص شده باشد، ۷ بیت برای opcode باقی خواهد ماند. یعنی می توانیم ۱۲۸ دستورالعمل مختلف داشته باشیم. از آنجایی که تعداد دستورالعمل‌های مورد نیاز برابر ۱۱۲ است، این مقدار کافی است. بنابرین حداقل طول ثبات و دستورالعمل‌ها در این پردازنده برابر ۲۲ بیت خواهد بود.

سوال ۴

پاسخ درست به سوال ۴ همین پاسخ است. اما پاسخ ۹۶ را هم از دانشجویان پذیرفتم

از آنجایی که این پردازنده دو آدرسه است و در حالت آنی، تنها یک آدرس میتواند به این صورت باشد، پس دستورالعمل‌های این پردازنده به شکل زیر است.

OPCODE	Memory Direct Address	Immediate Value
OPCODE	Memory Direct Address	Memory Direct Address

چون IR که همان ثبات دستورالعمل است ۳۲ بیتی است، طول کل دستورالعمل ۳۲ بیت است. از آنجایی که MAR یک رجیستر ۱۳ بیتی است، بنابرین آدرس‌های حافظه نیز ۱۳ بیتی هستند. و چون AC یک رجیستر به طول ۱۲ بیت است، قاعدتا مقدار آنی نیز باید ۱۲ بیتی باشد. بنابرین دستورالعمل‌ها به شکل زیر در خواهند آمد. بنابرین طول OPCODE در حالت آنی برابر ۷ و در حالت مستقیم برابر ۶ خواهد بود. همانگونه که در سوالات قبل گفتیم، لازم است که تعدادی از ترکیب‌های کدهای ۶ بیتی را برای تمایز قائل شدن با دستورات نوع دیگر کنار بگذاریم. اگر x ترکیب از حالت ۶ بیتی را کنار بگذاریم، تعداد دستورالعمل‌ها برابر خواهد بود با

$$f(x) = 2^6 - x + x2^1 = 2^6 + x$$

چون این تابع نسبت به x صعودی است، مقدار x را بزرگ‌ترین مقدار ممکنه در نظر میگیریم که برابر $2^6 - 1$ است چرا که حداقل یک دستور از هر نوع باید داشته باشیم. بنابرین تعداد کل حالت‌ها برابر میشود با

$$f(2^6 - 1) = 2^6 + 2^6 - 1 = 2^7 - 1$$

سوال ۵

این اشتباه را تعداد زیادی از دانشجویان مرتکب شدند. قرار نیست همیشه به یک شیوه فکر کنید. لطفا خلاقیت به خرج دهید
لطفا به پاسخ منتخب اول مراجعه بفرمایید

حجم حافظه این پردازنده برابر $64 \times 2^{10} = 2^{16}$ کلمه است. از آنجایی که کلمات یک بایتی هستند، حجم این حافظه برابر 2^{16} بایت است.

چون دستورات یک و سه کلمه‌ای هستند، بنابرین دستورات ۸ یا ۲۴ بیتی هستند.

توجه کنید که در حالت ثباتی مستقیم و غیرمستقیم، شماره ثبات مربوطه باید در دستورالعمل درج شود. بنابرین شماره دو ثبات باید در دستورات یک کلمه‌ای ذخیره شود. توجه کنید که ۵۶ دستور مختلف یک کلمه‌ای وجود دارد بنابرین برای مشخص کردن آن حداقل به $\lceil \log 56 \rceil = 6$ بیت OPCODE نیاز داریم. بنابرین در دستورات یک کلمه‌ای حداکثر ۲ بیت برای مشخص کردن شماره دو ثبات خواهیم داشت. از آنجایی که حداقل یک بیت برای مشخص کردن شماره ثبات ضروری است، در دستورات یک کلمه‌ای، ۶ بیت برای OPCODE، ۱ بیت برای مشخص کردن ثبات اول و یک بیت برای مشخص کردن ثبات دوم نیاز خواهیم داشت.

بدیهی است از آنجایی که در این نوع دستورات دو رجیستر مورد استفاده قرار میگیرد، این پردازنده حداقل باید دو رجیستر داشته باشد و از آنجایی که برای آدرس‌دهی رجیستر ۱ بیت در اختیار داریم، حداکثر ۲ رجیستر می‌توانیم داشته باشیم. بنابرین این پردازنده دقیقا ۲ رجیستر دارد.

در دستورات مستقیم و غیرمستقیم حافظه‌ای، باید آدرس یک حافظه در دستورالعمل معین شود. بنابرین از آنجایی که 2^{16} کلمه حافظه داریم، باید برای هر آدرس‌دهی حافظه‌ای ۱۶ بیت در اختیار داشته باشیم. از آنجایی که دستورات سه کلمه‌ای ۲۴ بیت دارند، تنها یک آدرس‌دهی به حافظه می‌توانیم داشته باشیم و آدرس دیگر را باید از نوع ثباتی در نظر بگیریم. بنابرین ۱۷ بیت در این نوع دستورات برای آدرس‌دهی استفاده میشوند و ۷ بیت برای OPCODE باقی می‌ماند.

در دستورات یک کلمه‌ای ۶ بیت برای OPCODE داریم. بنابرین ۶۴ ترکیب مختلف از OPCODEها میتوانیم داشته باشیم. ۵۶ تا از این ترکیب‌ها استفاده شده است بنابرین ۸ ترکیب برای ۶ بیت اول OPCODE دستورات ۳ کلمه‌ای باقی می‌ماند. و بیت هفتم به دو طریق می‌تواند انتخاب شود. بنابرین حداکثر ۱۶ ترکیب مختلف متمایز مجاز برای OPCODE کلمات سه کلمه‌ای وجود دارد.

سوال ۶

با وجود مد آدرس‌دهی شاخص، بخش عملوند (Operand) شامل دو قسمت است، یکی آدرس مبنا (Base Addr) و دیگری آدرس رجیستر اندیس (Reg Addr).

$$Operand = Base\ Addr \text{ و } Reg\ Addr \rightarrow Effective\ Addr$$
$$= Base\ Addr + Reg_{Reg\ Addr}$$

بنابراین اگر $Base\ Addr = 0$ شود، مد آدرس‌دهی اندیس به مد آدرس‌دهی رجیستری غیرمستقیم تبدیل می‌شود و اگر مقدار درونی رجیستر اندیس، صفر شود، مد آدرس‌دهی به مستقیم حافظه‌ای تبدیل خواهد شد. شکل زیر این مساله را نشان می‌دهد