تمرین دوم درس معماری کامپیوتر

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

آرین احدی نیا

استاد درس: جناب آقای دکتر جهانگیر

دستیار آموزشی: جناب آقای علی‌پور

پاییز ۱۴۰۰

فهرست عناوین

مفاهیم اولیه

سوال ۱

توجه فرمایید که در مبنای r در حالت روتین، مجموعه ارقام مورد استفاده برابر است با

$$\{0, 1, 2, \ldots, r-1\}$$

بنابرین بزرگ‌ترین اعشار n رقمی زمانی اتفاق می‌افتد که تمام n رقم اعشار برابر با $r-1$ باشد. در این حالت این مقدار برابر خواهد بود با:

$$\left(r^{-1} \times (r-1)\right) + \left(r^{-2} \times (r-1)\right) + \cdots + \left(r^{-n} \times (r-1)\right)$$

$$= (r^0 - r^{-1}) + (r^{-1} - r^{-2}) + \cdots + (r^{-n+1} - r^{-n})$$

$$= r^0 - r^{-n} = 1 - r^{-n}$$

سوال ۲

الف.

$$(0.\overline{23})_4 = 2 \times 4^{-1} + 3 \times 4^{-2} + 2 \times 4^{-3} + 3 \times 4^{-4} + 2 \times 4^{-5} + 3 \times 4^{-6} + \cdots$$

$$= (2 \times 4^{-1} + 2 \times 4^{-3} + 2 \times 4^{-5} + \cdots) + (3 \times 4^{-2} + 3 \times 4^{-4} + 3 \times 4^{-6} + \cdots)$$

$$= (2 \times 4^{-1})(4^0 + 4^{-2} + 4^{-4} + \cdots) + (3 \times 4^{-2})(4^0 + 4^{-2} + 4^{-4} + \cdots)$$

$$= \left(\frac{2}{4} + \frac{3}{16}\right)\left(\sum_{i=0}^{\infty} 16^{-i}\right) = \frac{11}{16}\left(\sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i\right)$$

توجه کنید که $\sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i$ یک سری هندسی با قدر نسبت بین صفر و یک است. بنابرین $\sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i$ به یک عدد حقیقی مانند S همگرا خواهد بود. مقدار S را می‌توانیم به طریق زیر محاسبه کنیم.

$$\begin{cases} S = \sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i \\ \frac{1}{16}S = \sum_{i=1}^{\infty}\left(\frac{1}{16}\right)^i \end{cases} \Rightarrow \left(1 - \frac{1}{16}\right)S = \left[\sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i\right] - \left[\sum_{i=1}^{\infty}\left(\frac{1}{16}\right)^i\right] = \left(\frac{1}{16}\right)^0 = 1 \Rightarrow S = \frac{16}{15}$$

بنابرین در نهایت خواهیم داشت

$$(0.\overline{23})_4 = \frac{11}{16}\left(\sum_{i=0}^{\infty}\left(\frac{1}{16}\right)^i\right) = \frac{11}{16}\times\frac{16}{15} = \frac{11}{15}$$

توجه کنید که چون عامل غیر از ۲ و ۵ در مخرج وجود دارد، این عدد در مبنای ۱۰ نیز دارای تناوب اعشاری خواهد بود. با انجام تقسیم مشاهده خواهیم کرد که در دور قرار میگیریم.

تبدیل‌شونده		مبنای تبدیل		حاصل یک مرحله تبدیل	رقم تولید شده در مبنای ۱۰	باقی‌مانده برای مرحله بعد
$\frac{11}{15}$	$\times$	10	$=$	$7+\frac{5}{15}$	7	$\mathbf{\frac{5}{15}}$
$\frac{5}{15}$	$\times$	10	$=$	$3+\frac{5}{15}$	3	$\mathbf{\frac{5}{15}}$
$\frac{5}{15}$	$\times$	10	$=$	$loop, back\ to\ step\ 2$		

با توجه به دوری که اتفاق می‌افتد، حاصل تقسیم برابر $0.7\overline{3}$ خواهد بود. توجه کنید که این موضوع به طور دقیق به وسیله استقرا ریاضی نیز قابل بیان است. بنابرین حاصل نهایی تبدیل مبنا برابر خواهد بود با

$$(0.\overline{23})_4 = 0.7\overline{3}$$

ب. برای سادگی بخش اعشاری و صحیح را جداگانه تبدیل میکنیم.

برای تبدیل بخش صحیح می‌توانیم مستقیم بر مبنای ۷ عدد را بسط دهیم و محاسبات را در مبنای ۷ انجام دهیم.

$$(231)_4 = (2)_7((4)_7)^2 + (3)_7((4)_7)^1 + (1)_7((4)_7)^0$$

$$= (2)_7(22)_7 + (3)_7(4)_7 + (1)_7 = (44)_7 + (15)_7 + (1)_7$$

$$= (44)_7 + (16)_7 = (63)_7$$

برای بخش اعشاری نیز محاسبات را مستقیما بین مبنای ۴ و ۷ انجام می‌دهیم.

تبدیل‌شونده		**مبنای تبدیل**		**حاصل یک مرحله تبدیل**	**رقم تولید شده در مبنای ۷**	**باقی‌مانده برای مرحله بعد**
$(0.3)_4$	$\times$	$(13)_4$	$=$	$(11.1)_4$	$(5)_7$	$\mathbf{(0.1)_4}$
$(0.1)_4$	$\times$	$(13)_4$	$=$	$(1.3)_4$	$(1)_7$	$\mathbf{(0.3)_4}$
$(0.3)_4$	$\times$	$(13)_4$	$=$	$loop, back\ to\ step\ 1$		

همانگونه که ملاحظه می‌کنید، بعد از دو مرحله در دور می‌افتیم و این روند تا بینهایت تکرار می‌شود. بنابرین حاصل اعشاری در مبنای ۷ متناوب و برابر $0.\overline{51}$ خواهد بود. بنابرین حاصل نهایی تبدیل مبنا عبارت است از

$$(231.3)_4 = \left(63.\overline{51}\right)_7$$

سوال ۳

۱. **از سیستم بدون علامت:** به سادگی با ضرب هر رقم در ارزش مرتبه‌اش و جمع این حاصل به ازای تمامی ارقام، به سادگی محاسبه مورد نظر را انجام می‌دهیم.

$$Unsigned(10011010) = 2^1 + 2^3 + 2^4 + 2^7 = 154$$

۲. **از سیستم مقدار_علامت:** برای این منظور رقم پرارزش را به عنوان علامت جدا می‌کنیم. ۱ بودن این رقم نشان از منفی بودن این عدد دارد. سپس سایر ارقام را مانند سیستم بدون علامت به مبنای ۱۰ تبدیل میکنیم و در نهایت علامت را بر حاصل اعمال می‌کنیم.

$$SignMagnitude(10011010) = -1 \times Unsinged(0011010)$$

$$= -(2^1 + 2^3 + 2^4) = -26$$

۳. **از سیستم مکمل اول:** از آنجایی که بیت پرارزش برابر یک است، یعنی عدد منفی است. بنابرین ابتدا مکمل هر یک از ارقام را محاسبه کرده و سپس مقدار بدون علامت حاصل بدست آمده را محاسبه می‌کنیم. توجه کنید که این روش مبتنی بر این است که جمع یک عدد n بیتی و مکمل اول آن برابر $2^n - 1$ خواهد بود.

$$Unsigend\big(Complement(10011010)\big) = -1 \times Unsigned(01100101)$$

$$-(2^0 + 2^3 + 2^5 + 2^6) = -101$$

۴. **از سیستم مکمل دوم:** از آنجایی که بیت پرارزش برابر یک است، یعنی عدد منفی است. بنابرین ابتدا مکمل هر یک از ارقام را محاسبه کرده و آن را با یک جمع می‌کنیم سپس مقدار بدون علامت حاصل بدست آمده را محاسبه می‌کنیم. توجه کنید که این روش مبتنی بر این است که جمع یک عدد n بیتی و مکمل دوم آن برابر 2^n خواهد بود.

$$Unsigend(Complement(10011010) + 1) = -1 \times Unsigned(01100110)$$

$$-(2^1 + 2^3 + 2^5 + 2^6) = -102$$

سوال ۴

حداکثر مقداری که در مبنای ۲ با n بیت قابل نمایش است، برابر خواهد بود با

$$2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2^0 = 2^n - 1$$

بنابرین اگر عدد x با n بیت قابل نمایش باشد، خواهیم داشت

$$x \le 2^n - 1 \Rightarrow x < 2^n \Rightarrow \log_2 x < n$$

توجه کنید که با توجه به تعریف جز صحیح در ریاضیات داریم

$$\lfloor \log_2 x \rfloor \le \log_2 x < \lfloor \log_2 x \rfloor + 1$$

بنابرین خواهیم داشت

$$\Rightarrow \lfloor \log_2 x \rfloor \le \log_2 x < n \Rightarrow \lfloor \log_2 x \rfloor < n$$

در نهایت با توجه به صحیح بودن مقادیر n و $\lfloor \log_2 x \rfloor$ خواهیم داشت.

$$\xRightarrow{n, \lfloor \log_2 x \rfloor \in \mathbb{Z}} \lfloor \log_2 x \rfloor + 1 \le n$$

بنابرین برای نمایش عدد x در مبنای ۲ حداقل $\lfloor \log_2 x \rfloor + 1$ بیت لازم خواهد بود. به عبارتی داشتن این تعداد بیت شرط لازم برای نمایش عدد x در مبنای ۲ است. توجه کنید که حداکثر مقدار قابل نمایش در مبنای ۲ با این تعداد رقم برابر است با $2^{\lfloor \log_2 x \rfloor + 1} - 1$. با توجه به تعریف اپراتور جز صحیح خواهیم داشت

$$\log_2 x < \lfloor \log_2 x \rfloor + 1 \Rightarrow 2^{\log_2 x} < 2^{\lfloor \log_2 x \rfloor + 1} \Rightarrow x < 2^{\lfloor \log_2 x \rfloor + 1}$$

بنابرین داشتن $\lfloor \log_2 x \rfloor + 1$ بیت شرط کافی برای نمایش عدد x خواهد بود.

بنابرین تعداد بیت‌های لازم و کافی برای نمایش عدد x در مبنای ۲ برابر $\lfloor \log_2 x \rfloor + 1$ می‌باشد (گزینه (د))

سوال ۵

فرض کنید که عدد a با استفاده از r رقم در مبنای x قابل نمایش باشد، حداقل و حداکثر مقدار این عدد برابر x^{r-1} و $x^r - 1$ خواهد بود. از آنجایی که هر چه عدد بزرگ‌تر باشد به ارقام بیشتری برای نمایش آن نیاز است. تعداد ارقام مورد نیاز برای نمایش این دو عدد را در مبنای t محاسبه خواهیم کرد.

تعداد ارقام مورد استفاده برای نمایش $x^r - 1$: اگر این عدد با استفاده از s رقم در مبنای t قابل نمایش باشد خواهیم داشت

$$x^r - 1 \le t^s - 1$$

$$\Rightarrow x^r \le t^s \Rightarrow \log_t x^r \le \log_t t^s \Rightarrow r.\log_t x \le s$$

$$\Rightarrow \lceil r.\log_t x \rceil \le \lceil s \rceil \xRightarrow{s\in\mathbb{Z}\Rightarrow\lceil s\rceil = s} \lceil r.\log_t x \rceil \le s$$

بنابرین برای نمایش این عدد در مبنای t حداقل به $\lceil r.\log_t x \rceil$ رقم نیاز است. حال می‌خواهیم نشان دهیم که این تعداد رقم برای نمایش چنین عددی کافی است. با توجه به تعریف اپراتور سقف داریم

$$r.\log_t x \le \lceil r.\log_t x \rceil$$

$$t^{\log_t x^r} \le t^{\lceil r.\log_t x \rceil} \Rightarrow x^r \le t^{\lceil r.\log_t x \rceil} \Rightarrow x^r - 1 \le t^{\lceil r.\log_t x \rceil} - 1$$

بنابرین داشتن $\lceil r.\log_t x \rceil$ رقم برای نمایش بزرگ ترین عدد r رقمی در مبنای x لازم و کافی است.

بنابرین داشتن $\lceil r.\log_t x \rceil$ رقم برای نمایش اعداد r رقمی در مبنای x کافی است.

بنابر استدلال مشابه برای نمایش بزرگ ترین عدد $r-1$ رقمی در مبنای x که برابر $x^{r-1} - 1$ است، داشتن $\lceil (r-1).\log_t x \rceil$ رقم لازم و کافی است. بنابرین برای نمایش x^{r-1} که کوچک‌ترین عدد r رقمی در مبنای x است، داشتن $\lceil (r-1).\log_t x \rceil$ لازم است.

بنابرین داشتن $\lceil (r-1).\log_t x \rceil$ رقم برای نمایش اعداد r رقمی در مبنای x لازم است.

به بیان ساده‌تر، اگر تعداد ارقام مورد نیاز برای نمایش یک عدد r رقمی در مبنای x برابر n باشد، خواهیم داشت:

$$\lceil (r-1).\log_t x \rceil \le n \le \lceil r.\log_t x \rceil$$

سوال ۶

سیستم BCD یک سیستم برای نمایش رقمی اعداد مبنای ۱۰ است به این صورت که هر رقم را به مبنای ۲ تبدیل میکنیم و به صورت ۴ بیتی پشت سر یکدیگر می‌نویسیم.

سیستم اعداد خود مکمل، سیستمی برای نمایش رقمی اعداد مبنای ۱۰ هستند که مکمل آن عدد، با مکمل گیری از تک‌تک بیت‌ها بدست میاید.

به سادگی یک سیستم خود مکمل قابل ساخت است. می‌توانیم به ازای هر یک از ارقام بین صفر تا چهار یک معادل باینری ۴ بیتی در نظر بگیریم و مکمل آن چهار بیت را به عنوان مکمل آن رقم در نظر بگیریم.

تعداد سیستم‌های خود مکمل با ۴ بیت برای نمایش اعداد را می‌توانیم به صورت زیر محاسبه کنیم.

$$16 \times 1 \times 14 \times 1 \times 12 \times 1 \times 10 \times 1 \times 8 \times 1 = 215040$$

ابتدا معادل باینری رقم صفر را انتخاب می‌کنیم، سپس معادل باینری رقم ۹ به صورت یکتا برابر مکمل رقم صفر مشخص میشود. سپس از بین ۱۴ ترکیب باقی‌مانده یک معادل باینری برای نمایش رقم ۱ انتخاب میکنیم و معادل باینری برای نمایش رقم ۸ به صورت یکتا انتخاب خواهد شد. با ادامه این روند به تمامی ارقام یک معادل باینری تخصیص داده میشود.

توجه فرمایید که در نمایش BCD، که یک سیستم خودمکمل نیست، ۱۰ ترکیب اول از ترکیب ۴ بیت را برای نمایش ارقام به ترتیب انتخاب می‌کنیم. اما توجه کنید که ترکیب بیتی i ام و $2^4 - i$ ام مکمل یکدیگر هستند، بنابرین اگر ترکیب‌ها را به صورت متقارن انتخاب کنیم، یک سیستم خودمکمل خواهیم داشت. در جدول زیر می‌توانید سه سیستم خود مکمل را در کنار BCD ملاحظه فرمایید. همانگونه که ملاحظه میفرمایید سه سیستم خودمکمل به صورت متقارن قرار گرفته‌اند. دو سیستم اول سیستم‌های شناخته شده‌ای هستند که به $excess - 3$ و $8,4,-2,-1$ معروف هستند. سیستم آخر را خودمان بر مبنای گفته‌های قبلی ساخته ایم. این سیستم را میتوانیم $excess - 12$ بنامیم چرا که اعداد ۱۲ واحد شیفت خورده‌اند.

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
BCD	0	1	2	3	4	5	6	7	8	9						
excess-3				0	1	2	3	4	5	6	7	8	9			
8, 4, -2, -1	0				4	3	2	1	8	7	6	5				9
	5	6	7	8	9							0	1	2	3	4

سوال ۷

بدون از دست دادن کلیت فرض کنید که $m = 2^k$ باشد.

از آنجایی که 2^k ثبات n بیتی داریم، برای انتخاب مقدار یک ثبات از بین سایرین برای وارد کردن دیتا به آدرس جدید، به یک تسهیم‌کننده $2^k:1$ با nبیت نیازمندیم. توجه بفرمایید که برای ساخت یک تسهیم‌کننده n بیتی، میتوان از n

تسهیم‌کننده یک بیتی با ورودی‌های $Select$ مشترک استفاده کرد. بنابرین به n تسهیم‌کننده $2^k:1$ نیازمندیم. چنین تسهیم‌کننده‌هایی دارای k ورودی $Select$ خواهند بود.

در حالت کلی که m توانی از ۲ نیست، میتوانیم در نظر بگیریم $k = \lceil \log_2 m \rceil$. در این حالت، برخی از ورودی‌های تسهیم کننده به جایی وصل نخواهند بود. توجه بفرمایید که بدیهی است که تسهیم‌کننده کوچک‌تر نمیتوانیم چنین مداری بسازیم چرا که تعداد ثبات‌ها بیشتر از تعداد ورودی‌های تسهیم‌کننده خواهند شد.

سوال ۸

در سوال قبل نشان دادیم که برای ساخت این مدار به n عدد تسهیم‌کننده $2^k \to 1$ برای ساخت این مدار نیازمندیم. ($k = \lceil \log_2 m \rceil$)

توجه کنید که یک تسهیم‌کننده $2^k \to 1$ را می‌توانیم با استفاده از یک رمزگشا $k \to 2^k$ و 2^k عدد بافر سه‌حالته بسازیم. برای این منظور کافی است که ورودی‌های $Select$ را به رمزگشا وصل کنیم و هر یک از خطوط خروجی رمزگشا را به ورودی $enable$ بافرهای سه‌حالته متصل کنیم و خروجی بافرهای سه‌حالته را به یکدیگر وصل کنیم. ورودی بافرهای سه‌حالته، ورودی‌های تسهیم‌کننده ساخته شده خواهند بود.

می‌توانیم با کنار هم قرار دادن n تا از تسهیم‌کننده‌هایی که به روش بالا ساختیم یک تسهیم‌کننده n بیتی بسازیم. اما نکته حائز اهمیت این است که نیازی نیست که از n عدد رمزگشا استفاده کنیم چرا که تمام این رمزگشاها به ورودی یکسانی که همان $Select$ تسهیم‌کننده است متصل می‌شوند و عملکرد مشابهی دارند. بنابرین تنها استفاده از یک رمزگشا کافی خواهد بود.

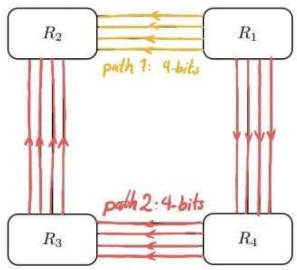
در حالت کلی که m توانی از ۲ نیست، نیازی نیست که از $2^k \times n$ بافر سه‌حالته استفاده کنیم. می‌توانیم از $m \times n$ بافر سه‌حالته استفاده کنیم. بنابرین در این حالت برخی خروجی‌های رمزگشا به جایی وصل نخواهند بود.

بنابرین در نهایت سخت افزارهای مورد نیاز عبارتند از ($k = \lceil \log_2 m \rceil$)

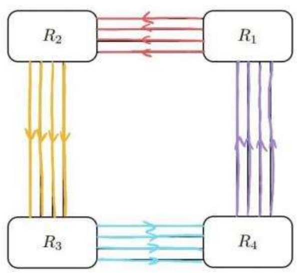
- یک عدد رمزگشا $k \to 2^k$
- mn عدد بافر سه‌حالته
- m عدد ثبات n بیتی

سوال ۹

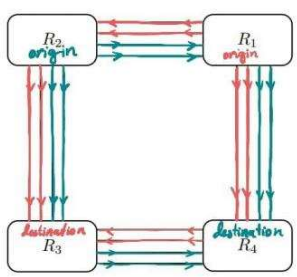
الف. از طریق دو مسیر زیر، میتوانیم یک گذرگاه حداکثر ۸ بیتی داشته بین این دو ثبات داشته باشیم. توجه فرمایید که تعداد خطوط انتقال داده متصل به هر یک از این ثبات‌ها ۸ عدد است. بنابرین گذرگاهی با عرض ۸ بیت نمی‌توان داشت.



ب. به صورت زیر، هر چهار ثبات میتوانند در هر کلاک ۴ بیت به ثبات کناری انتقال دهند. توجه بفرمایید که در این حالت تمامی خطوط مشغول به‌کار هستند و بیش از این انتقال داده نمی‌توان انجام داد.



ج. به صورت شکل زیر، میتوانیم ۴ بیت از R_1 به R_3 و ۴ بیت دیگر از R_2 به R_4 انتقال دهیم. توجه بفرمایید که در این حالت نیز تمامی خطوط مشغول به‌کار هستند و تمامی انتقال‌ها از مسیر بهینه انجام میشوند، بنابرین انتقال داده بیش از این حالت غیرممکن است.



ارزیابی کارایی

سوال ۱

توجه فرمایید که اگر فرکانس کلاک، زمان سپری شده و تعداد سیکل کلاک زده شده را به ترتیب f، Δt و C در نظر بگیریم، خواهیم داشت که

$$C = f.\Delta t$$

بنابرین تعداد کلاک‌های لازم برای اجرای این برنامه روی پردازنده A برابر است با

$$C_A = f_A.\Delta T_A = 400 \times 10^6\ Hz \times 10\ s = 4 \times 10^9$$

از آنجایی که تعداد کلاک مورد نیاز برای اجرای این برنامه در کامپیوتر B دو برابر کامپیوتر A است، خواهیم داشت.

$$C_B = 1.2C_A = 4.8 \times 10^9$$

از آنجایی که زمان اجرا در کامپیوتر B برابر ۶ ثانیه است، خواهیم داشت

$$f_B = \frac{C_B}{\Delta T_B} = \frac{4.8 \times 10^9}{6\ s} = 0.8 \times 10^9\ Hz = 800 \times 10^6\ Hz = 800\ MHz$$

بنابرین فرکانس این پردازنده برابر $800\ MHz$ است.

سوال ۲

فرض کنید که زمان اجرای برنامه برابر T باشد. بنابرین از آنجایی که ۸۰ درصد زمان اجرا مختص اجرای دستورات ضرب است، زمان اجرای دستورالعمل‌های ضرب $0.8T$ و زمان اجرای سایر دستورالعمل‌ها برابر $0.2T$ خواهد بود.

پس از بهبود می‌خواهیم که زمان اجرای برنامه $\frac{1}{5}$ شود. بنابرین زمان اجرای برنامه در حالت بهبود یافته برابر $T' = 0.2T$ خواهد بود. توجه فرمایید که این زمان T' برابر جمع زمان اجرای دستورالعمل‌های ضربی و غیر ضربی است.

$$T' = T'_{mul} + T'_{nonmul}$$

از آنجایی که بهبودی در زمان اجرای دستورالعمل های غیر ضربی نمی‌دهیم، زمان اجرای دستورالعمل‌های غیر ضربی برابر $0.2T$ خواهد بود.

$$0.2T = T'_{mul} + 0.2T \Rightarrow T'_{mul} = 0$$

بنابرین زمان اجرای دستورالعمل‌های ضربی باید در این پردازنده صفر باشد. این به معنی بهبود با ضریب بینهایت است. بدیهی است که این امر غیرممکن است و امکان این کار وجود ندارد.

سوال ۳

اگر P_i برابر احتمال رویت یک دستورالعمل در خلال اجرای یک برنامه باشد، متوسط تعداد کلاک برای هر دستورالعمل در اجرای برنامه با توجه به رابطه $Excpection$ از طریق رابطه زیر محاسبه می‌گردد.

$$CPI = \sum_{i \in Instructions} P_i \times ClockPerInstruction_i$$

از طرفی توجه بفرمایید که

$$\begin{cases} CPI = \dfrac{ClockCount}{InstructionsCount} \\ f = \dfrac{ClockCount}{\Delta T} \\ MIPS = \dfrac{InstructionCount}{\Delta T \times 10^6} \end{cases} \Rightarrow CPI = \frac{f}{MIPS \times 10^6} \Rightarrow MIPS = \frac{f}{CPI \times 10^6}$$

الف. طبق روابط فوق خواهیم داشت

$$CP_A = 0.45 \times 1 + 0.20 \times 3 + 0.20 \times 4 + 0.15 \times 6 = 2.75$$

$$MIPS_A = \frac{500 \times 10^6}{2.75 \times 10^6} = 181$$

ب. طبق روابط فوق خواهیم داشت

$$CPI_B = 0.45 \times 1 + 0.20 \times 2 + 0.20 \times 3 + 0.15 \times 5 = 2.20$$

$$MIPS_B = \frac{500 \times 10^6}{2.20 \times 10^6} = 227$$

ج. نسبت زمان اجرای یک برنامه خاص در پردازنده A نسبت به B برابر است با

$$\frac{T_A}{T_B} = \frac{\dfrac{InstructionsCount}{MIPS_A \times 10^6}}{\dfrac{InstructionsCount}{MIPS_B \times 10^6}} = \frac{MIPS_B}{MIPS_A} = \frac{\dfrac{f}{CPI_B \times 10^6}}{\dfrac{f}{CPI_A \times 10^6}} = \frac{CPI_A}{CPI_B} = \frac{2.75}{2.20} = 1.25$$

بنابرین زمان اجرای پردازنده B، ۱/۲۵ برابر سریع‌تر است.

سوال ۴

زمان اجرای یک محک برابر است با

$$T = InstructionsCount \times ClockPerInstruction \times ClockCycleTime$$

اگر تعداد دستورالعمل‌های لازم برای اجرای محک بر هر دو ماشین یکسان باشد، خواهیم داشت

$$\frac{T_A}{T_B} = \frac{ClockPerInst_A \times ClockCycleTime_A}{ClockPerInstruction_B \times ClockCycleTime_B} = \frac{4 \times 50\ ns}{2.5 \times 65\ ns} = 1.23$$

از آنجایی که زمان اجرا بر روی A به میزان ۱/۲۳ برابر B طول میکشد، پردازنده B، ۱/۲۳ برابر سریع‌تر است.

سوال ۵

اگر تعداد دستورالعمل‌های نوع A و B به ترتیب n_A و n_B باشد، تعداد کلاک‌های لازم برای اجرای این برنامه در دو پردازنده P_1 و P_2 برابر خواهد بود با

$$ClockCount_{P_1} = 3n_A + 4n_B$$

$$ClockCount_{P_1} = 5n_A + 3n_B$$

توجه بفرمایید که فرکانس از رابطه

$$f = \frac{ClockCount}{\Delta T}$$

محاسبه میگردد. از آنجایی که زمان اجرای دو برنامه یکسان است، خواهیم داشت که

$$\frac{ClockCount_{P_1}}{f_{P_1}} = \frac{ClockCount_{P_2}}{f_{P_2}} \Rightarrow \frac{f_{P_1}}{f_{P_2}} = \frac{ClockCount_{P_1}}{ClockCount_{P_2}}$$

بنابرین خواهیم داشت که

$$\frac{200\ MHz}{300\ MHz} = \frac{3n_A + 4n_B}{5n_A + 3n_B} \Rightarrow 10n_A + 6n_B = 9n_A + 12n_B \Rightarrow n_A = 6n_B \Rightarrow \frac{n_A}{n_B} = 6$$

بنابرین تعداد دستورالعمل‌های نوع A شش برابر تعداد دستورالعمل‌های نوع B خواهد بود. (گزینه (ج))

سوال ۶

توجه کنید که اگر یک عملیات را که به میزان t طول میکشد را با ضریب c بهبود بخشیم، زمان اجرای بهبود یافته برابر $\frac{t}{c}$ خواهد بود.

در نظر بگیرید که زمان اجرای کل برنامه برابر T باشد، پیشنهادات مذکور به صورت زیر بر روی زمان اثر میگذارند.

پیشنهاد ۱.

زمان بهبود یافته	ضریب بهبود	زمان اجرا	سهم اجرا	نوع عملیات
$0.02T$	10	$0.2T$	20%	تابع ریشه دوم
$0.8T$	1	$0.8T$	80%	غیر از تابع ریشه دوم
$\mathbf{0.82T}$		$\boldsymbol{T}$		**مجموع**

پیشنهاد ۲.

زمان بهبود یافته	ضریب بهبود	زمان اجرا	سهم اجرا	نوع عملیات
$0.25T$	2	$0.5T$	50%	ممیز شناور
$0.5T$	1	$0.5T$	50%	غیر از ممیز شناور
$\mathbf{0.75T}$		$\boldsymbol{T}$		**مجموع**

از آنجایی که زمان اجرا پس از اعمال راهکار اول $\frac{0.82}{0.75} = 1.09$ برابر زمان اجرا پس از ارائه راهکار دوم است، بنابرین راهکار دوم 1.09 برابر سریع‌تر است.

سوال ۷

توجه بفرمایید که تعداد دستورالعمل‌ها برای اجرای این برنامه ثابت است، چرا که همان پردازنده با همان ISA را بهبود بخشیدیم و تغییری در تعداد دستورالعمل‌ها اجرایی بوجود نمی‌آید.

از آنجایی که باقی‌مانده زمان مربوط به نوع سوم خواهد بود، ۴۰٪ زمان اجرا برای نوع سوم خواهد بود. فرض کنید که زمان اجرای برنامه T باشد، در نتیجه خواهیم داشت

نوع عملیات	سهم اجرا	زمان اجرا	ضریب بهبود	زمان بهبود یافته
اول	20%	$0.2T$	2	$0.1T$
دوم	40%	$0.4T$	4	$0.1T$
سوم	40%	$0.4T$	0.5	$0.8T$
مجموع		$\boldsymbol{T}$		$\boldsymbol{T}$

بنابرین همانگونه که ملاحظه میفرمایید زمان اجرای برنامه تغییری نکرده است. بنابرین با توجه به رابطه زیر

$$MIPS = \frac{InstructionsCount}{T}$$

از آنجایی که تعداد دستورالعمل‌ها و زمان اجرا ثابت است، $MIPS$ نیز ثابت خواهد بود. بنابرین سرعت اجرای برنامه روی پردازنده جدید برابر $200\ MIPS$ خواهد بود. (گزینه (الف))

سوال ۸

این پاسخ از اقای محمدعلی حسین نژاد عبدی است

خب ابتدا با توجه به روی سوال چند فرض را در نظر میگیریم:

- در شروع آزمون نقش مهمان، عادی هست.
- تا قبل شروع آزمون دانشجویان توانایی همفکری و پیاده سازی استراتژی دارند.
- قبل از شروع آزمون تعداد دانشجویان بر همه آشکار هست.

خب در ابتدا اشاره کنم که امکان برد وجود دارد. الگوریتم به این شکل هست که یکی از دانشجویان را انتخاب میکنیم مثلا آقا/ خانم X . مجموعه بقیه دانشجویان را Y مینامیم. حال نحوه کار به این صورت هست.

- الگوریتم برای X: اگر وقتی وارد جلسه میشود نقش کاربر عادی باشد آن را ارائه دهنده میکند و هیچ کار دیگری در هیچ شرایط دیگری با نقش مهمان نمیکند. اگر اولین باری باشد که مهمان را در نقش عادی میبیند شروع به شمردن میکند. در غیر اینصورت تعداد شمرده شده قبلی را یکی زیاد میکند. در صورتی که مقدار شمرده شده برابر با تعداد دانشجویان منهی یک باشد، جواب بله میدهد. در غیر اینصورت جواب خیر میدهد.
- الگوریتم برای مجموعه Y: اگر اولین بار هست که مهمان را در نقش ارائه دهنده میبیند، آن را به نقش عادی تغییر میدهد. در غیر اینصورت تغییری در نقش مهمان ایجاد نمیکند. و همیشه در جواب به سوال نهایی استاد جواب خیر میدهد.

خب در این صورت با شروع بازی که نقش مهمان عادی هست، تا زمان رسیدن به دانشجوی X اتفاقی در سیستم نمیافتد و نقش مهمان تغییر نمیکند. حال وقتی نوبت به دانشجوی X میرسد و دفعه اول هست که مهمان را در حالت عادی میبیند، نقش مهمان را به ارائه دهنده تغییر میدهد و در جواب به سوال آخر خیر میگوید. بعد از این هر بار که شخصی برای بار اول مهمان را ارائه دهنده میبیند، نقش آن را به عادی تغییر میدهد و تا زمانی که نوبت دوباره به X برسد نقش مهمان تغییری نمیکند و وقتی به X میرسد شمارنده ای که X در ذهن خود گرفته یکی بیشتر میشود. خب با این الگوریتم تعداد کسانی که حداقل یکبار در آزمون شرکت کرده اند مشخص میشود و یکجا جمع میشود. و زمانی که X به تعداد کل دانشجویان منهی یک، شمرده باشد به سوال مورد نظر جواب بله میدهد و همه نمره کامل را میگیرند.(بنده کد این الگوریتم را پیاده سازی کردم. برای 19 دانشجو حدود 400 آزمون مورد نیاز هست تا به نتیجه برسد.)

```python
import random
experiments_exam_counter = 0
for experiment in range(1000):
    guest = False
    students = [True for i in range(19)]
    student_x = random.randint(0, 18)
    x_counter = 0
    total_exams_counter = 0
    student_x_counter_started = False
    while x_counter != 18:
        student = random.randint(0, 18)
        if student == student_x:
            if not guest:
                if student_x_counter_started:
                    x_counter += 1
                    guest = True
                else:
                    student_x_counter_started = True
                    guest = True
        else:
            if students[student] and guest:
                students[student] = False
                guest = False
        total_exams_counter += 1
    experiments_exam_counter += total_exams_counter


print(experiments_exam_counter / 1000)
```