پاسخ سوال ۱)

$$\frac{۲۵۶ \times ۲^{۱۰} \times ۵۰ns}{۴۰ms} \times ۱۰۰ = ۳۲/۷۶۸ \text{ درصد}$$

پاسخ سوال ۲)

نرخ فقدان زمانی کمینه است که دستورهایی که وارد حافظه نهان شده‌اند در طول اجرای برنامه از حافظه نهان خارج نشوند. برای این امر باید هیچیک از دو بلوک دستور، به یک بلوک از حافظه نهان نگاشت نشوند، یعنی باید تعداد کل دستورهای برنامه از اندازه حافظه نهان کوچکتر باشد.

اندازه حافظه نهان یک کیلوبایت یعنی ۲۵۶ کلمه می‌باشد. هر دستور نیز یک کلمه است پس ظرفیت حافظه نهان ۲۵۶ دستور است. پس:

$$I_{max} = ۲۵۶ - ۳ = \mathbf{۲۵۳}$$

توجه: اگر اولین دستور از خانه صفر حافظه شروع نشود باید فاصله اولین دستور برنامه تا ابتدای بلاکی که در آن قرار می‌گرفت از عدد بالا کم شود. مثلاً اگر آدرس اولین دستور در خانه ۲۰ حافظه بود، از ابتدای بلوک متناظرش که بلوک صفر می‌باشد به اندازه ۱۲ بایت یعنی $\frac{۲۰}{۴} = ۵$ کلمه فاصله دارد لذا $I_{max} = ۲۵۶ - ۳ - ۵ = ۲۴۸$. و اگر اولین دستور در خانه ۱۵۲ از حافظه اصلی قرار داشت، از ابتدای بلوک متناظرش که ۱ می‌باشد به اندازه ۶ کلمه فاصله داشت، در نتیجه $I_{max} = ۲۵۶ - ۳ - ۶ = ۲۴۷$

پاسخ سوال ۳)

برای محاسبه زمان اجرا کافی است تعداد سیکلهای لازم برای اجرای برنامه را بدون تاثیر سلسله مراتب حافظه با تعداد سیکلهای تعلیق به خاطر فقدان جمع کرده و در پریود کلاک ضرب کنیم.

برای فهم بهتر سوال، در سه حالت زمان اجرا را حساب می‌کنیم:

حالت اول $I = I_{max}$

تمامی دستورها در حافظه نهان جای می‌گیرند. ابتدا از تاثیر سلسله مراتب حافظه صرفنظر کنیم. دستور اول که مقدار ۱۰۰ را در $t1 قرار می‌دهد، یک بار اجرا می‌شود اما بقیه دستورها به تعداد دفعات اجرای حلقه یعنی ۱۰۰ بار اجرا می‌شوند. لذا تعداد سیکلها برابر است با $۲۵۵ \times ۱۰۰ + ۱$

برای محاسبه تعداد سیکلهایی که صرف انتقال به حافظه نهان شده است کافی است تعداد فقدان‌ها را حساب کرده و در جریمه فقدان ضرب کنیم. جریمه فقدان طبق سوال ۱۰۰۰ سیکل است. چون اندازه برنامه با اندازه حافظه نهان برابر است برای هر بلوک از دستورها یک بار فقدان داریم. کلاً ۲۵۶ دستور داریم و بلوک‌ها ۳۲ دستوری هستند پس ۸ بلوک داریم. پس تعداد فقدان‌ها ۸ است. لذا تعداد سیکل‌های جریمه برابر با ۸ × ۱۰۰۰ است. پس زمان اجرا برابر است با:

$$CPU\ Time = \left(۱ + ۲۵۵ \times ۱۰۰ + ۱۰۰۰ \times ۸\right) \times ۱ns = ۳۳/۵۰۱\mu s$$

آ) حالت دوم، $I = I_{max} + ۱$ بدون استفاده از *Loop Fusion*

تعداد سیکل اجرا بدون در نظر گرفتن تاثیر سلسله مراتب حافظه همان تعدادی است که در قسمت قبل محاسبه شد. اما برای محاسبه تعداد فقدان‌ها. دستورها ۲۵۷ دستور هستند. در نتیجه ۹ بلاک را اشغال خواهند کرد. اما حافظه نهان دارای ۸ بلاک است. چون دستورها پشت سر هم هستند و نگاشت مستقیم است، پس بلاک صفر و ۸ دستور به یک نقطه از حافظه نهان نگاشت می‌شوند. پس بلاک‌های ۱ تا ۷ از دستور تنها یکبار فقدان خواهند داشت، اما بلاک‌های صفر و ۸ به تعداد دفعات اجرای حلقه فقدان خواهند داشت. پس تعداد فقدان‌ها ۲ × ۱۰۰ + ۷ و جریمه آن ۱۰۰۰ × $\left(۲ \times ۱۰۰ + ۷\right)$ می‌باشد. پس زمان اجرا برای این حالت برابر است با:

$$CPU\ Time = \left(۱ + ۲۵۵ \times ۱۰۰ + ۱۰۰۰ \times \left(۷ + ۱۰۰ \times ۲\right)\right) \times ۱ns = ۲۳۲/۵۰۱\mu s$$

ب) حالت سوم، $I = I_{max} + ۱$ با استفاده از *Loop Fusion* و شکستن به دو حلقه با $I - ۱$ دستور و یک دستور

اگر از تکنیک Loop Fusion استفاده کنیم باید یک حلقه را به دو حلقه با همان محدوده اندیسی بشکنیم. لذا ساختار دو حلقه مشابه و به صورت زیر است:

```
addi $t1,$t1,100
L: addi $t1,$t1,-1
253 Instructions here
bne $t1,$0,L
```

```
addi $t1,$t1,100
L: addi $t1,$t1,-1
1 Instruction here
bne $t1,$0,L
```

در واقع کد بالا همانند دو برنامه مجزا عمل می‌کند. قسمت اول همان حالت اول است که در بالا بررسی شد! قسمت دوم نیز مشابه است و به صورت زیر حساب می‌شود:

تعداد سیکل‌های ساعت بدون در نظر گرفتن سلسله مراتب حافظه برابر است با ۱۰۰ × ۳ + ۱. تنها یک بار فقدان داریم. پس سیکل جریمه ۱ × ۱۰۰۰ است. پس تعداد سیکل لازم برای قسمت دوم ۱۳۰۱ = ۱ × ۱۰۰۰ + ۱۰۰ × ۳ + ۱می‌باشد. زمان اجرای این حالت برابر است با:

$$CPU\ Time = ۳۳/۵۰۱\mu s + ۱۳۰۱ \times ۱ns = ۳۴/۸۰۲\mu s$$

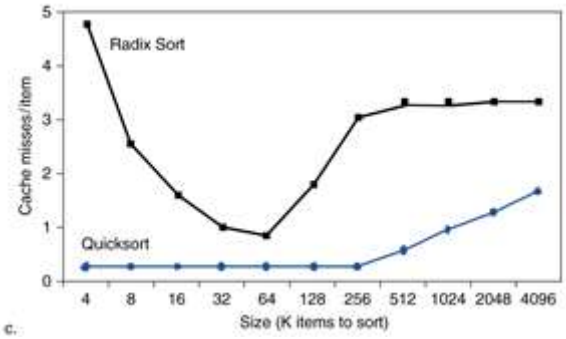
اگر مقدار دو حالت دوم و سوم را مقایسه کنید، مشاهده می‌کنید که در حالت دوم زمان اجرا تقریباً ۲۳۳ میکروثانیه است در حالیکه در حالت سوم حدود ۳۵ میکروثانیه است یعنی با وجود اینکه تعداد دستورهای حالت سوم بیشتر است و برخی دستورها به جای ۱۰۰ بار، ۲۰۰ بار اجرا می‌شوند، اما بازهم زمان اجرای حالت سوم کمتر است. نسبت زمان اجرای حالت دوم و سوم حدود ۷ برابر است.

این مثال تاثیر آشنایی با سخت‌افزار در کارایی نرم‌افزارها را نشان می‌دهد. (و این یکی از دلایل نیاز به یادگیری معماری کامپیوتر برای دانشجویان گرایش نرم‌افزار است)

پاسخ سوال ۴)

زمان اجرا تحت تاثیر سه عامل فرکانس، CPI و تعداد دستورها می‌باشد. در یک پردازنده ثابت، فرکانس برای هر دو الگوریتم یکسان است. اگر چه مرتب‌سازی مبنایی دارای تعداد دستورهای کمتری است، اما دارای CPI میانگین بیشتری است و بیشتر بودن CPI به حدی است که زمان اجرای آنرا بیشتر می‌کند.

دلیل بیشتر بودن CPI، وجود فقدان‌های بیشتر در اجرای مرتب‌سازی مبنایی است. قسمت آخر از همان شکل ۵-۱۸ این مطلب را نشان می‌دهد:



اگر فقدان اتفاق افتد، باید تعدادی سیکل تلف شود تا داده مورد نظر از حافظه اصلی به حافظه نهان انتقال یابد و این به معنی افزایش CPI برای اجرای یک دستور است. در واقع عاملی که روشهای کلاسیک و استاندارد تحلیل الگوریتم‌ها از آن چشم‌پوشی می‌کنند، **تاثیر سلسله مراتب حافظه در زمان اجرای یک الگوریتم** است! (این نیز دلیل دیگری برای نیاز به یادگیری معماری کامپیوتر برای دانشجویان گرایش نرم‌افزار است)

پاسخ سوال ۵)

حافظه مجازی هر نگاشتی که داشته باشد، هر سطر آن دارای ساختار زیر است (طبق صورت سوال از بیت‌های LRU چشم‌پوشی شد) :

Valid Bit	Tag	Block Data

در این سوال، حافظه نهان یک حافظه ۱۶ سطری ۱۰۲۴ کلمه‌ای است. یعنی حافظه نهان این سوال ۱۶ سطر همانند شکل بالا دارد که بخش Block Data آن ۱۰۲۴ کلمه‌ای است. پس اگر تعداد بیت‌های لازم برای Tag را T و اندازه هر کلمه را w فرض کنیم، اندازه حافظه نهان برابر است با:

$$\text{اندازه حافظه نهان} = ۱۶ \times \left(۱ + T + ۱۰۲۴w\right)$$

در حالت کلی آدرس یک خانه از حافظه را می‌توان به فیلدهای زیر تقسیم کرد:

Tag	Set	Block Offset	Byte Offset

در این سوال حافظه اصلی ما دارای واحد آدرس‌پذیر کلمه است. پس Byte Offset دارای صفر بایت خواهد بود. تعداد کلمات داخل یک بلاک ۱۰۲۴ کلمه است پس تعداد بیت‌های Block Offset برابر با $\log_۲ ۱۰۲۴ = ۱۰$ است. تعداد بیت‌های Set به درجه آزادی بستگی دارد و اگر درجه آزادی a باشد تعداد بیت‌های لازم برای فیلد Set برابر با $\log_۲ \frac{۱۶}{a}$ خواهد بود. همچنین حافظه اصلی دارای $۲^{۲۰}$ کلمه است و آدرس‌دهی بر مبنای کلمه است (واحد آدرس‌پذیر کلمه است نه بایت) پس اندازه آدرس برابر با ۲۰ بیت است. پس تعداد بیت‌های لازم برای Tag برابر است با:

$$T = ۲۰ - ۰ - ۱۰ - \log_۲ \frac{۱۶}{a}$$

آ) نگاشت مستقیم:

نگاشت مستقیم همان نگاشت شبه‌انجمنی با درجه آزادی (درجه انجمنی) یک است. لذا $a = ۱$

با جایگذاری در فرمول $T = ۶$ پس

$$\text{اندازه حافظه نهان} = ۱۶ \times \left(۱ + ۶ + ۱۰۲۴w\right)$$

آ) نگاشت انجمنی کامل:

نگاشت انجمنی کامل همان نگاشت شبه‌انجمنی است با درجه آزادی ۱۶ (برای این سوال). لذا $a = ۱۶$. پس $T = ۱۰$ در نتیجه:

$$\text{اندازه حافظه نهان} = ۱۶ \times \left(۱ + ۱۰ + ۱۰۲۴w\right)$$

آ) نگاشت شبه‌انجمنی با درجه آزادی ۲:

$a = ۲$. پس $T = ۷$. لذا

$$\text{اندازه حافظه نهان} = ۱۶ \times \left(۱ + ۷ + ۱۰۲۴w\right)$$

پاسخ ۶)

نگاشت مستقیم:

در این نگاشت شماره بلوک در حافظه نهان برابر است با باقیمانده آدرس بلوک در حافظه اصلی بر تعداد بلاک‌ها در حافظه نهان. تعداد بلاک‌ها در حافظه نهان طبق صورت سوال ۴ است. لذا داریم:

آدرس بلوک	شماره بلوک
۰	$۰ \bmod ۴ = ۰$
۴	$۴ \bmod ۴ = ۰$
۸	$۸ \bmod ۴ = ۰$
۰	$۰ \bmod ۴ = ۰$
۴	$۴ \bmod ۴ = ۰$
۸	$۸ \bmod ۴ = ۰$

چون حافظه نهان در ابتدا خالی است، پس درخواست اول همواره با فقدان روبه‌رو می‌شود. درخواست دوم بلوک ۴ از حافظه اصلی را درخواست می‌کند که باید در بلوک ۰ از حافظه نهان قرار گرفته باشد که وجود ندارد. لذا با فقدان رو به رو می‌شود. درخواست سوم نیز همانند درخواست دوم با فقدان رو به رو می‌شود. درخواست سوم بلوک صفر از حافظه اصلی را می‌خواهد که باید در خانه ۰ از حافظه نهان قرار داشته باشد، اما در این خانه، بلوک ۸ از حافظه اصلی قرار دارد، لذا بازهم فقدان داریم. دو درخواست دیگر نیز با فقدان رو به رو می‌شوند.

نگاشت انجمنی:

در این نگاشت هر بلوک از حافظه اصلی می‌تواند در هر یک از بلوک‌های حافظه نهان باشد یا قرار گیرد. طبق صورت سوال، سیاست جانشینی، LRU است یعنی بلاکی از حافظه نهان جانشین می‌شود اخیراً کمتر استفاده شده باشد. در این سوال برای درخواست‌ها داریم:

۱. بلاک صفر از حافظه اصلی درخواست می‌شود حافظه نهان خالی است پس فقدان اتفاق می‌افتد. اکنون باید بلاک درخواستی از حافظه اصلی به حافظه نهان منتقل شود. چون حافظه نهان خالی است، هر جایی از آن می توان این بلاک را قرار داد. ما خانه صفر را انتخاب می‌کنیم (معمولاً در کامپیوتر شروع از صفر است)

۲. بلاک ۴ از حافظه اصلی درخواست می‌شود اما در حافظه نهان وجود ندارد پس فقدان داریم. پس از انتقال بلاک درخواستی از حافظه اصلی باید آنرا در بلاک مناسب از حافظه نهان قرار دهیم. بلاک صفر از حافظه نهان پر است اما بلاک ۱ و ۲ خالی است. طبق سیاست LRU بلاک ۴ حافظه اصلی در بلاک ۱ حافظه نهان قرار می‌گیرد.

۳. بلاک ۸ از حافظه نهان درخواست می‌شود. بازهم فقدان . بلاک‌های ۱ و ۲ از حافظه نهان پر است. طبق سیاست LRU این بلاک حتماً باید در بلاک ۲ حافظه نهان قرار گیرد.

۴. اکنون بلاک صفر حافظه اصلی درخواست می‌شود که در بلاک صفر از حافظه نهان قرار دارد. لذا برخورد اتفاق می‌افتد.

۵. بلاک ۴ از حافظه اصلی درخواست می‌شود که در بلاک ۱ از حافظه نهان است. پس برخورد داریم.

۶. بلاک ۸ از حافظه اصلی درخواست می‌شود که در بلاک ۲ از حافظه نهان جای گرفته است. بنابراین بازهم برخورد داریم.

آدرس بلوک	شماره بلوک	Hit/Miss
۰	۰	Miss
۴	۱	Miss
۸	۲	Miss
۰	۰	Hit
۴	۱	Hit
۸	۲	Hit

نگاشت شبه‌انجمنی با درجه آزادی ۲:

در این نگاشت ۴ بلاک از حافظه نهان به دو مجموعه دو بلاکی تقسیم می‌شود. باقیمانده تقسیم آدرس بلاک بر تعداد مجموعه‌ها شماره مجموعه را مشخص می‌کند. پس از تعیین شماره مجموعه، هر یک از بلاک‌های آن مجموعه می‌تواند حاوی بلاک مورد نظر ما باشد یا بلاک مورد نظر ما در صورت فقدان می‌تواند با صلاحدید LRU در یکی از آنها جانشین شود. تعداد مجموعه‌ها برابر است با تعداد بلاک‌های حافظه نهان تقسیم بر درجه آزادی که برابر با $\frac{۴}{۲} = ۲$ است. تمامی آدرسهای درخواستی دارای باقیمانده ۰ در تقسیم به ۲ هستند. لذا همه درخواست‌ها به مجموعه صفر از حافظه نهان نگاشت می‌شوند. پس در یکی از بلاک‌های صفر یا یک حافظه نهان قرار دارند یا درصورت فقدان در آنها جانشین می‌شوند. در این سوال برای درخواست‌ها داریم:

۱. بلاک صفر از حافظه اصلی درخواست می‌شود حافظه نهان خالی است پس فقدان اتفاق می‌افتد. اکنون باید بلاک درخواستی از حافظه اصلی به حافظه نهان منتقل شود. چون مجموعه صفر خالی است، هر جایی از آن می توان این بلاک را قرار داد. ما خانه صفر را انتخاب می‌کنیم (معمولاً در کامپیوتر شروع از صفر است)

۲. بلاک ۴ از حافظه اصلی درخواست می‌شود اما در حافظه نهان وجود ندارد پس فقدان داریم. پس از انتقال بلاک درخواستی از حافظه اصلی باید آنرا در بلاک مناسب از مجموعه صفر از حافظه نهان قرار دهیم. بلاک صفر از حافظه نهان پر است اما بلاک ۱ خالی است. پس حتماً در بلاک ۱ جانشین می‌شود.

۳. بلاک ۸ از حافظه نهان درخواست می‌شود. باید در مجموعه صفر این بلاک را بیابیم. اما بلاک ۸ در این مجموعه قرار ندارد. پس فقدان. هر دو بلاک‌های ۰ و یک مربوط به مجموعه صفر بوده و هر دو پر هستند. طبق LRU باید بلاک صفر جانشین شود چون دیرتر مورد استفاده قرار گرفته است.

۴. اکنون بلاک صفر حافظه اصلی درخواست می‌شود بازهم در مجموعه صفر وجود ندارد. پس فقدان داریم. اما این دفعه بلاک صفر حافظه اصلی در بلاک ۱ حافظه نهان جانشین می‌شود.

۵. بلاک ۴ از حافظه اصلی درخواست می‌شود که در مجموعه صفر وجود ندارد. لذا فقدان اتفاق می‌افتد و طبق LRU باید در بلاک صفر جانشین شود.

۶. بلاک ۸ از حافظه اصلی درخواست می‌شود که در مجموعه صفر وجود ندارد. پس فقدان داریم و طبق LRU بلاک مقصد ۱ است.

آدرس بلوک	شماره مجموعه	شماره بلوک	Hit/Miss
۰	۰	۰	Miss
۴	۰	۱	Miss
۸	۰	۰	Miss
۰	۰	۱	Miss
۴	۰	۰	Miss
۸	۰	۱	Miss

بنابر آنچه گفته شد پاسخ نهایی سوال به صورت زیر است:

آدرس بلوک مورد نظر	Direct Map		Fully associative		2-way set associative	
	Hit or Miss	شماره بلوک	Hit or Miss	شماره بلوک	Hit or Miss	شماره بلوک
۰	Miss	۰	Miss	۰	Miss	۰
۴	Miss	۰	Miss	۱	Miss	۱
۸	Miss	۰	Miss	۲	Miss	۰
۰	Miss	۰	Hit	۰	Miss	۱
۴	Miss	۰	Hit	۱	Miss	۰
۸	Miss	۰	Hit	۲	Miss	۱

پاسخ سوال ۷)

مقصود از CPI موثر همان CPI کل است. کل CPI لازم برابر است با CPI پایه بعلاوه کلاک‌هایی که به خاطر فقدان در خط لوله تعلیق اتفاق می‌افتد. چون دو سطح حافظه نهان داریم پس باید برای هر یک جداگانه حساب کنیم. پس:

$$\begin{aligned} Effective\ CPI &= Total\ CPI \\ &= Base\ CPI + Primary\ Stalls\ per\ instruction \\ &\quad + Secondary\ Stalls\ per\ instruction \end{aligned}$$

$$Effective\ CPI = ۱ + ۰/۰۵ \times ۱۰ + ۰/۰۲ \times ۴۰۰ = ۹/۵$$

موفق باشید

گروه حل تمرین