Homework 2

Lec 6,7,8

۱- صحیح یا غلط بودن عبارات زیر را با ذکر دلیل مشخص کنید.

الف) با استفاده از write buffer می توانیم مطمئن باشیم CPU معطل نوشتن داده در حافظۀ اصلی نمی ماند.

ب) با افزایش سایز بلوکهای حافظۀ نهان پدیدۀ Cache Pollution کاهش مییابد.

ج) اگر فرکانس پردازندهای بالاتر باشد، در شرایطی که نیاز به کار با حافظه داریم، CPU مجبور است تعداد کلاکهای بیشتری منتظر حافظه بماند.

د) اگر بتوانیم با روشی عملکرد CPU را بهبود دهیم، میتوانیم تاثیری که Miss Penalty بر روی عملکرد سیستم میگذارد را کاهش دهیم.

الف) غلط - با این که استفاده از write buffer میتواند معطلی CPU را کاهش دهد، اما همچنان ممکن است در مواردی این بافر پر شود. در این شرایط CPU باید منتظر بماند تا دادهای از بافر به حافظۀ اصلی منتقل شود تا بتواند دادۀ جدید را بر روی بافر بنویسد، در غیر این صورت دادهای گم میشود.

ب) غلط – با افزایش سایز بلوکها شانس این که از دادهای که به حافظۀ نهان آورده ایم استفاده کنیم کاهش می یابد. بنابراین ممکن است با افزایش بیش از حد سایز بلوکها حجم زیادی داده را به حافظۀ نهان منتقل کنیم که استفادهای ندارند و بیهوده فضای آن را اشغال کنیم.

ج) صحیح – سرعت خواندن از حافظه به سرعت حافظه مربوط است و با افزایش فرکانس پردازنده تغییر چندانی نمی کند. بنابراین به دلیل این که تعداد کلاکهای پردازنده در واحد زمان افزایش یافته، مجبور است کلاکهای بیشتری منتظر حافظۀ اصلی بماند.

د) غلط – بالعکس، هرچه عملکرد پردازنده بهبود یابد تاثیر Miss penalty بر عملکرد سیستم مشهودتر است.

۲- فرض کنید پردازندهای دارای حافظۀ نهان میخواهد دادهای را روی حافظۀ اصلی بنویسد.

الف) چه زمانی می گوییم write-miss رخ داده؟

ب) حال فرض کنید دادۀ مورد نظر در حافظۀ نهان وجود دارد و می خواهد توسط CPU به روز رسانی شود. دو سیاست موجود برای این کار را مختصر توضیح دهید.

ج) مشخص کنید write buffer در هر یک از این دو سیاست در چه زمانی و چطور باعث کاهش معطلی CPU در روند نوشتن داده بر روی حافظۀ اصلی میشود.

**پاسخ**

الف) هنگامی که بخواهیم داده ای را در حافظه write کنیم ابتدا باید بلوک حاوی آن word را داشته باشیم، word مورد نظر را آپدیت کنیم و سپس آن را به حافظه اصلی منتقل کنیم زیرا واحد ارتباط حافظه اصلی و حافظه نهان بلوک ها هستند و نمی توان یک word را مستقیما از حافظه نهان به حافظه اصلی نوشت. در صورتی که این بلوک داخل حافظه نهان وجود نداشته باشد write-miss رخ می دهد

ب) فرض کنید پردازنده دادهای را از حافظۀ نهان را خوانده، عملیاتی را بر آن انجام داده و مقدار جدید داده را بر روی حافظۀ نهان مینویسد. میدانیم در حافظۀ نهان هر لحظه با آمدن دادۀ جدید احتمال اخراج داده از حافظه وجود دارد، بنابراین باید توجه داشته باشیم که پیش از اخراج دادۀ آپدیت شده توسط CPU مقدار آن را در حافظۀ اصلی نیز تغییر دهیم. دو سیاست برای این کار وجود دارد:

1. Write-Through**:** در این سیاست همان لحظه ای که داده را درحافظۀ نهان آپدیت کردیم آن را در حافظۀ اصلی نیز آپدیت میکنیم. با این روش ساده می توانیم مطمئن باشیم دادۀ جدید از دست نمی رود و به حافظۀ اصلی منتقل میشود. اما این روش روشی بهینه نیست زیرا به ازای هر دستور نوشتن بر حافظۀ اصلی باید به آن مراجعه کنیم. به علاوه تعداد این دستورات قابل چشم پوشی نیستند، لذا این روش بسیار کند عمل خواهد کرد. هدف استفادۀ ما از حافظۀ نهان سرعت بخشیدن به تعامل میان پردازنده و حافظۀ اصلی بود اما با این روش دستورات نوشتن حتی کندتر از پیش انجام می شوند، زیرا باید یک بار داده را بر روی حافظۀ نهان و یک بار بر روی حافظۀ اصلی بنویسیم.
2. Write-Back: در این روش آپدیت کردن دادۀ حافظۀ اصلی را تا جای ممکن به تعویق می اندازیم. دادهای که در حافظۀ نهان قرار دارد طبق اصل همجواری زمانی احتمالاً بیش از یک بار استفاده خواهد شد، بنابراین آپدیت کردن حافظۀ اصلی به ازای هر تغییر داده کار بیهوده ایست و به همین دلیل این روش از روش قبلی بهتر عمل میکند (برای مثال فرض کنید به ازای هر تغییر متغیر i در یک حلقۀ for مجبور به رجوع به حافظۀ اصلی باشیم!). اما دربارۀ این روش باید به دو پرسش پاسخ دهیم.

یک این که چه زمانی داده را آپدیت کنیم تا هم تا جای ممکن مجبور به تکرار آن نباشیم و هم مطمئن باشیم داده گم نمیشود. گفتیم باید این کار را تا جای ممکن به تعویق بیاندازیم، لذا بهترین زمان برای این کار هنگام اخراج داده از حافظۀ نهان میباشد.

دو این که زمان اخراج داده از حافظۀ نهان از کجا متوجه شویم داده تغییر کرده یا خیر و نیاز به آپدیت در حافظۀ اصلی دارد یا خیر. برای این کار از یک تک بیت به نام بیت dirty استفاده میکنیم. زمانی که دادۀ جدید به حافظۀ نهان منتقل میشود مقدار این بیت به طور پیشفرض برابر با صفر در نظر گرفته میشود، یعنی داده با دادۀ موجود در حافظۀ اصلی منطبق است. زمانی که پردازنده برای اولین بار مقدار این داده را تغییر دهد این بیت به یک تغییر پیدا میکند، زیرا مقدار داده دیگر با مقدار درون حافظۀ اصلی منطبق نیست. بنابراین زمانی که میخواهیم داده را از حافظۀ نهان اخراج کنیم کافیست مقدار موجود در این بیت را بررسی کرده و تصمیم بگیریم که نیاز به نوشتن آن در حافظه داریم یا خیر.

ج) گفتیم دلیل ضعف سیاست Write-Through کند بودن آن و معطل نگه داشتن پردازنده به ازای هر دستور نوشتن بر حافظه است. برای بهبود عملکرد این روش می توانیم از یک بافر سریع میان پردازنده و حافظۀ اصلی به نام write buffer استفاده کنیم تا CPU مجبور نباشد تا نوشتن داده بر حافظۀ اصلی منتظر بماند. هر زمان که قرار بود دادهای بر حافظۀ اصلی نوشته شود ابتدا بر روی write buffer نوشته میشود و پردازنده میتواند به کارهای خود ادامه دهد تا write buffer در زمان مناسب داده را به حافظۀ اصلی منتقل کند. البته اگر تعداد داده هایی که میخواهند بر حافظه نوشته شوند زیاد باشد ممکن است این بافر پر شود و پردازنده مجبور شود منتظر بماند تا بافر خالی شود، پس همچنان احتمال معطلی پردازنده وجود دارد.

استفاده از این بافر با همین ایده میتواند در سیاست Write-Back نیز مفید باشد اما زمان استفاده از آن کمی متفاوت است. طبق قسمت (الف) در این سیاست زمانی نیاز به نوشتن بر حافظۀ اصلی داریم که دادهای dirty قرار است از حافظۀ نهان اخراج شود. این زمان زمانیست که پردازنده دادهای جدید درخواست کرده و miss رخ داده، حال حافظۀ نهان باید به حافظۀ اصلی رجوع کرده و دادۀ جدید را جایگزین یکی از دادههای فعلی بکند. درصورتی که از write-buffer استفاده نکنیم پردازنده مجبور است در این زمان به مدت دو miss penalty صبر کند، یک بار برای این که حافظۀ نهان دادۀ قبلی را بر روی حافظۀ اصلی آپدیت کند و یک بار برای این که دادۀ جدید را از حافظۀ اصلی بخواند. برای حل این مشکل این بار یک بافر میان حافظۀ نهان و حافظۀ اصلی قرار میدهیم تا حافظۀ نهان دادۀ قدیمی را همان لحظه بر روی بافر نوشته و بدون معطلی برای خواندن دادۀ جدید به حافظۀ اصلی رجوع کند. پس از آن دادۀ روی بافر هم میتواند به حافظۀ اصلی منتقل شود.

بنابراین استفاده از write-buffer میتواند در هر دو سیاست مفید باشد.

۳- فرض کنید هنگام خواندن دادهای از حافظۀ نهان miss رخ داده و می خواهیم یک بلوک 4 کلمه ای از داده را از حافظۀ اصلی بخوانیم. پهنای گذرگاه حافظه به اندازۀ یک کلمه است و همچنین هر کلاک پردازنده معادل 3 کلاک گذرگاه حافظه میباشد. در این روند 1 کلاک گذرگاه[[1]](#footnote-0) برای انتقال آدرس داده از حافظۀ نهان به حافظۀ اصلی، 9 کلاک برای دسترسی به دادۀ درون حافظه و 2 کلاک برای انتقال داده از حافظۀ اصلی به حافظۀ نهان زمان صرف می شود.

الف) Miss Penalty را در این شرایط حساب کنید.

ب) فرض کنید در کنار این پردازنده از یک Instruction Cache با نرخ برخورد 97% و یک Data Cache با نرخ برخورد 95% استفاده شده است. همچنین دستورات نوشتن و خواندن از حافظه در مجموع 40% از دستورات پردازنده را تشکیل میدهند. در صورتی که Base CPI برای این پردازنده 3 باشد، میزان CPI واقعی را برای مجموعۀ پردازنده و حافظه بدست آورید.

ج) اگر در قسمت (ب) با همان پردازنده از حافظه های نهان بهتری استفاده کنیم، به طوری که نرخ برخورد برای  
I-Cache به 98% و برای D-Cache به 96% افزایش یابد، بار دیگر محاسبات قسمت (ب) را انجام داده و در مورد میزان اهمیت Miss Rate در حافظه های نهان نتیجه گیری کنید.

**پاسخ**

الف) داده های موجود در مساله به شرح زیر هستند:

* 1 block = 4 words
* 1 bus cycle needed for address transfer
* 9 bus cycles needed per DRAM access
* 2 bus cycle needed per data transfer

در نتیجه می توان نوشت:

Miss penalty = 1 + 9 × 4 + 2 × 4 = 45 bus cycles = 135 CPU cycles

ب) با توجه به داده های موجود در مساله و قسمت الف داریم:

* I-Cache miss rate = 3%
* D-Cache miss rate = 5%
* Miss penalty = 135 cycles
* Base CPI = 3
* Load & stores are 40% of instructions

بنابراین داریم:

Miss cycles per instruction:

* I-cache: 0.03 × 135 = 4.05
* D-cache: 0.4 × 0.05 × 135 = 2.7

Actual CPI = 3 + 4.05 + 2.7 = 9.75

ج) با توجه به داده های موجود در مساله و قسمت الف داریم:

* I-Cache miss rate = 2%
* D-Cache miss rate = 4%
* Miss penalty = 135 cycles
* Base CPI = 3
* Load & stores are 40% of instructions

بنابراین داریم:

Miss cycles per instruction:

* I-cache: 0.02 × 135 = 2.7
* D-cache: 0.4 × 0.04 × 135 = 2.16

Actual CPI = 3 + 2.7 + 2.16 = 7.86

با این که تنها یک درصد عملکرد حافظه های نهان را بهبود دادیم مشاهده میشود مقدار سرعت سیستم تقریباً 24% بهبود یافت که برای یک پردازنده مقدار قابل توجهیست. بنابراین اهمیت زیادی دارد که روشهایی برای بهبود هرچند کم عملکرد حافظه های نهان پیدا کنیم.

۴- حافظه نهانی به اندازه 16KByte داریم که اندازه هر بلوک آن 256Byte می باشد. سیاست جایدهی این حافظه نهان 2way Set Associative می باشد و اندازه حافظه اصلی نیز 128KByte است. (اندازه هر word یک byte است)

الف) قالب آدرس دهی را در این حافظه نشان دهید.

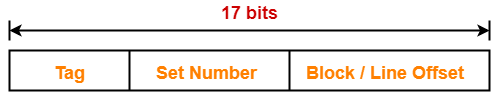
ب) برای ذخیره tag در این حافظه نهان مجموعا از چند byte استفاده می شود.

**پاسخ**

الف)

در مرحله اول، تعداد بیت مورد نیاز برای هر word را پیدا می کنیم:

اندازه حافظه اصلی 128Kbyte است ( بایت)، در نتیجه برای آدرس دهی به 17 بیت نیاز داریم:

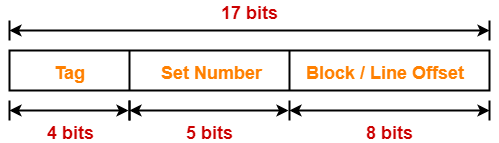


در مرحله بعد، تعداد بیت مورد نیاز برای بخش Block (آدرس دهی word داخل بلوک) و همین طور تعداد بیت مورد نیاز برای بخش Set Number (آدرس دهی set داخل حافظه نهان) را به دست می آوریم. تعداد بیت مورد نیاز برای Tag نیز از کم کردن جمع این دو عدد از عدد 17 (تعداد بیت کل آدرس دهی word) به دست می آید.

اندازه هر بلوک 256Byte، و اندازه هر word یک بایت است. در نتیجه داخل هر بلوک 256 یا word داریم، پس برای آدرس دهی هر word داخل بلوک آن، به 8 بیت نیازمندیم.  
حال تعداد set های داخل حافظه نهان را حساب می کنیم. می دانیم داخل هر set دو بلوک داریم، همین طور با تقسیم اندازه کل حافظه نهان به اندازه هر بلوک، میتوانیم تعداد بلوک های داخل حافظه نهان را محاسبه کنیم :

این عدد را تقسیم بر تعداد بلوک های هر set میکنیم () و تعداد set داخل حافظه نهان را بدست می آوریم. اکنون برای آدرس دهی این تعداد set به 5 بیت نیاز داریم.

همان طور که گفته شد تعداد بیت لازم برای بخش tag هم از جمع و تفریق همین اعداد بدست می آید : .



ب)

در این قسمت باید حجم کل مصرف شده برای ذخیره tag ها در این حافظه نهان را حساب کنیم. می دانیم به ازای هر بلوک ذخیره شده در حافظه نهان، به یک tag نیاز داریم، در نتیجه برای محاسبه عدد خواسته شده در این قسمت، کافیست تعداد بلوک های داخل حافظه نهان را در تعداد بیت های لازم برای tag ضرب کنیم، عدد بدست آمده تعداد بیت لازم برای ذخیره کل tag ها در حافظه نهان است:

256bits -> 32 bytes

۵- یک حافظه نهان 4 بلوکه را درنظر بگیرید که در ابتدا خالی است و در آن از سیاست جایگزینی FIFO استفاده می‌شود بررسی کنید برای شماره بلوک‌های مورد نظر cpu که در زیر از چپ به راست مشخص شده است، عملکرد cache را در  دو حالت  4way Set Associative و 2way Set Associative بررسی کنید. آیا با افزایش K میزان  hit rate ما افزایش پیدا میکند؟

2, 4,7,6,5,2,8,12,7,16,15,4,13

2 way Set Associative

| 13 | 4 | 15 | 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | #set | |
| 13 | 4 | 15 | 16 | 5 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | bloks in each set | |
| 15 | 16 | 5 | 12 | 7 | 8 | 2 | 6 | 7 | 4 | - | 2 | - |
| m | m | m | m | h | m | m | m | m | m | m | m | m | h/m | |

hit rate = 1/13

| 13 | 4 | 15 | 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 13 | 4 | 15 | 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | bloks in each set | |
| 4 | 15 | 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | - |
| 15 | 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | - | - |
| 16 | 7 | 12 | 8 | 2 | 5 | 6 | 7 | 4 | 2 | - | - | - |
| m | m | m | m | m | m | m | m | m | m | m | m | m | h/m | |

hit rate = 0/13

۶- به موارد زیر پاسخ دهید.

الف) شماره بلوک‌های مورد نظر cpu از چپ به راست مشخص شده است میزان hit rate را برای یک cache  حاوی چهار بلوک در دو حالت  2way Set Associative و 4way Set Associative را  با سیاست LRU مشخص کنید.

4,2,5,7,2,6,4,8,12,7,5,6,13,4

الف)

| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | #set | |
| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | bloks in each set | |
| 6 | 5 | 12 | 7 | 5 | 8 | 4 | 6 | 2 | 4 | 5 | - | 4 | - |
| m | m | m | h | h | m | m | m | m | h | m | m | m | m | h/m | |

hit rate = 3/14

| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | bloks in each set | |
| 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 |  |
| 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | - |  |
| 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 4 | 4 | - | - |  |
| m | m | m | m | m | m | m | m | m | h | m | m | m |  | h/m | |

hit rate = 1/14

ب)

| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 1 | 3 | 0 | 0 | 0 | 2 | 2 | 3 | 1 | 2 | 0 | #set | |
| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | bloks in each set | |
| 12 | 5 | 2 | - | - | 8 | 4 | - | 2 | - | - | - | - | - |
| m | m | h | h | h | m | m | h | m | h | m | m | m | m | h/m | |

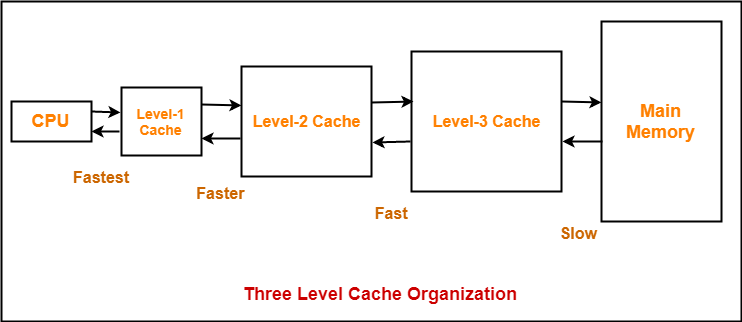
hit rate = 5/14

| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | #set | |
| 4 | 13 | 6 | 5 | 7 | 12 | 8 | 4 | 6 | 2 | 7 | 5 | 2 | 4 | bloks in each set | |
| 6 | 5 | 12 | 7 | 5 | 8 | 4 | 6 | 2 | 4 | 5 | - | 4 | - |
| 12 | 7 | 8 | - | - | 4 | 6 | 2 | 4 | - | - | - | - | - |
| 8 | - | 4 | - | - | 6 | 2 | - | - | - | - | - | - | - |
| h | m | h | h | h | m | m | h | m | h | m | m | m | m | h/m | |

hit rate = 6/14

اختلال بلیدی پدیده‌ای است که در آن افزایش تعداد بلوک‌ها موجود برای یک cache منجر به افزایش تعداد missها برای الگوهای دسترسی خاص به حافظه می شود. در این مثال مشاهده میکنیم که با افزایش ظرفیت cache از 4 بلوک به 8 بلوک hit rate ما بالا رفته و این به معنی کاهش miss rate است و در این مثال اختلال بلیدی رخ نداده است

۷- در درس با Average Memory Access Time (AMAT) آشنا شدید. حال فرض کنید یک حافظۀ نهان 3 مرحله ای دارید. نمونه ای از این نوع حافظه نهان را میتوانید در شکل زیر مشاهده کنید.



عملکرد این حافظۀ نهان به صورتی است که پردازنده ابتدا داده را از حافظۀ نهان Level-1 درخواست میکند. در صورتی که داده در این حافظۀ نهان وجود داشت به پردازنده تحویل داده میشود، در غیر این صورت حافظه نهان Level-1 به حافظه نهان Level-2 رجوع میکند و به همین ترتیب. در انتها اگر داده در حافظه نهان مرحلۀ آخر نیز وجود نداشت به حافظۀ اصلی رجوع میشود.

فرض کنید Miss Penalty عددی مانند P بوده و برای هر لایه Miss Rate برابر با Ri و Hit Time برابر با Ti باشد.  سعی کنید رابطه ای برای AMAT با فرض استفاده از چنین حافظه نهانی پیدا کنید.

**پاسخ**

طبق مطالب موجود در اسلایدهای درس میدانیم:

AMAT = Hit time + Miss rate × Miss penalty

به علاوه با توجه به طرز کار این حافظۀ نهان می توان دریافت که AMAT در هر لایه به عملکرد لایۀ بعدی خود وابسته است، زیرا در صورت miss باید به لایۀ بعد رجوع کنیم. بنابراین میتوان رابطۀ زیر را برای AMAT هر لایه نوشت:

AMATi = Ti + Ri × AMATi+1

و برای لایۀ آخر که در ارتباط با حافظۀ اصلی میباشد میتوان از رابطۀ اصلی استفاده کرد:

AMATn = Tn + Rn × Miss penalty

بنابراین برای محاسبۀ AMAT کل این حافظه نهان 3 لایه میتوان رابطۀ زیر را نوشت:

AMAT = T1 + R1 × (T2 + R2 × (T3 + R3 × Miss Penalty))

1. Bus cycle [↑](#footnote-ref-0)