**Negar - Lecture 6**

# ~~(تمرین، امتحان)سوال x از مجموعه سوالات تمرین اول (اسلاید 6)~~

~~الف) سیاست write-back و بلوک dirty را توضیح دهید.~~

~~ب) write-miss چیست و سیاست write-back با چه روشهایی به آن پاسخ میدهد؟~~

**~~پاسخ~~**

~~الف - در این سیاست به جای اینکه داده هایی که قرار است در حافظه اصلی نوشته شوند را CPU مستقیما یا با استفاده از buffer در آن بنویسد، ابتدا فقط در حافظه نهان نوشته می شوند. به بلوکی در حافظه نهان که یکی از word های آن تغییر کرده (در واقع توسط CPU بازنویسی شده) بلوک dirty گفته می شود. در این سیاست، وقتی که درخواست یک block داده از cache می شود، اگر miss رخ دهد و اگر بلوکی که در خانه ای از حافظه نهان قرار دارد که بلوک جدید می خواهد جایگزین آن شود (بلوکی که قرار است از حافظه نهان پاک شود) بلوک dirty باشد، قبل از آنکه بلوک جدید جایگزین بلوک قدیمی شود آن بلوک قدیمی باید در حافظه اصلی write شود. برای اینکه در این فرایند حافظه نهان معطلی اضافه نداشته باشد این فرایند توسط یک buffer انجام می شود، یعنی ابتدا بلوک در یک buffer که نزدیک حافظه نهان است و نوشتن در آن سریع انجام می شود، نوشته می شود و پس از آن که حافظه نهان به کار های بعدی خود رسیدگی می کند buffer آن داده را داخل حافظه اصلی ذخیره می کند.~~

~~ب - 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، از آنجایی که CPU نیز دسترسیی به آن ندارد، ابتدا باید این بلوک از حافظه اصلی خوانده شود و روی حافظه نهان نوشته شود، word مورد نظر در آن بازنویسی شود و پس از آن، آن بلوک دوباره به حافظه اصلی منتقل شود. البته روش دیگری هم وجود دارد که اصلا بلوک داخل حافظه نهان آپدیت نشود و word مستقیما در حافظه اصلی نوشته شود اما معمولا در سیاست write-back از روش اول استفاده می کنند.~~

## (کوییز)سوال اول (اسلاید 6)

~~در مورد یک حافظۀ نهان نگاشت مستقیم به سوالات زیر پاسخ دهید و دلیل آن را مختصر ذکر کنید.~~

~~الف) اصل همجواری مکانی چطور در حافظۀ نهان نمود پیدا میکند؟~~

در مورد یک حافظۀ نهان نگاشت مستقیم با توجه به اصول همجواری زمانی و مکانی به سوالات زیر پاسخ داده و دلیل پاسخ خود را مختصر ذکر کنید.

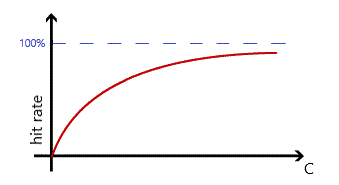
الف) فرض کنید در این حافظۀ نهان اندازۀ بلوکهای حافظه ثابت است. با این فرض نمودار hit rate بر حسب اندازۀ حافظۀ نهان را رسم کنید.

ب) حال فرض کنید اندازۀ حافظۀ نهان ثابت است. با این فرض نمودار hit rate بر حسب اندازۀ بلوکهای حافظۀ نهان را رسم کنید.

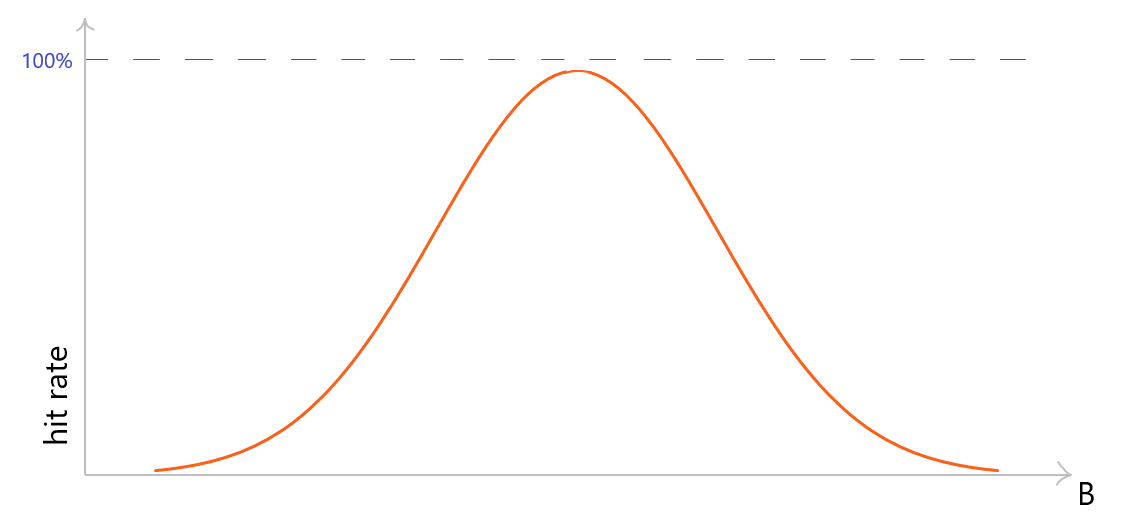
**پاسخ**

اصل همجواری مکانی بیان می دارد اگر پردازنده دادهای را درخواست کرده به احتمال بالا دادهای نزدیک (از نظر محل قرارگیری در حافظۀ اصلی) به آن را نیز در آینده خواهد خواست. اساس بلوک بندی حافظۀ نهان نیز بر همین اصل است. به جای این که با هر رجوع به حافظه تنها دادهای که میخواهیم را در حافظۀ نهان ذخیره کنیم، داده های مجاور آن (که در بلوک یکسانی قرار دارند) را نیز به حافظۀ نهان می آوریم. با این کار تعداد مراجعه ها به حافظۀ اصلی کاهش یافته و عملکرد سیستم بهبود مییابد.

الف) با افزایش سایز حافظۀ نهان میزان hit rate نیز افزایش می یابد، زیرا حافظۀ نهان می تواند دادۀ بیشتری را در خود ذخیره کند تا در صورت نیاز به CPU بدهد. مقدار hit rate زمانی که اندازۀ حافظۀ نهان با حافظۀ اصلی برابر باشد میتواند به 100% برسد، هر چند که در عمل انجام چنین کاری توجیه ندارد و سعی میشود با کوچک نگهداشتن سایز حافظۀ نهان و استفاده از تکنیک هایی برای پیشبینی درخواستهای CPU این مقدار را به 100% نزدیک کنیم.

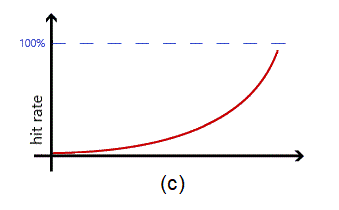
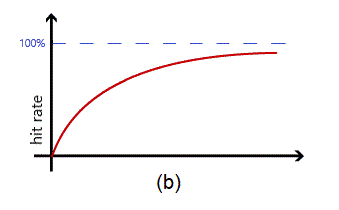
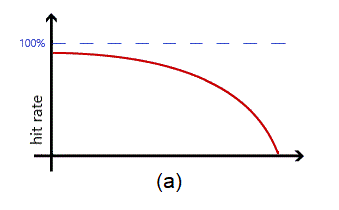


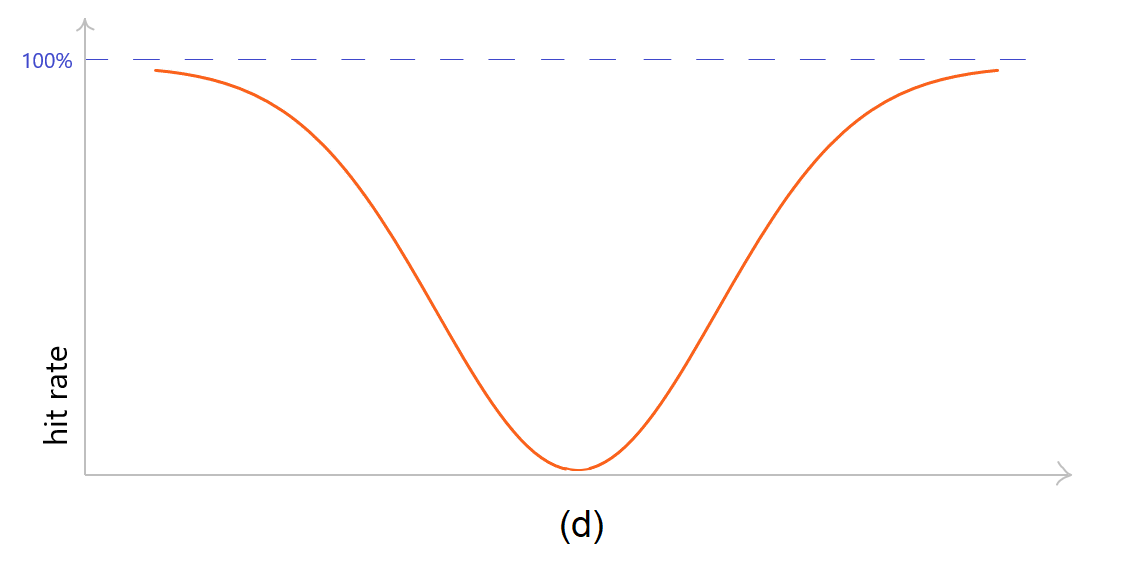
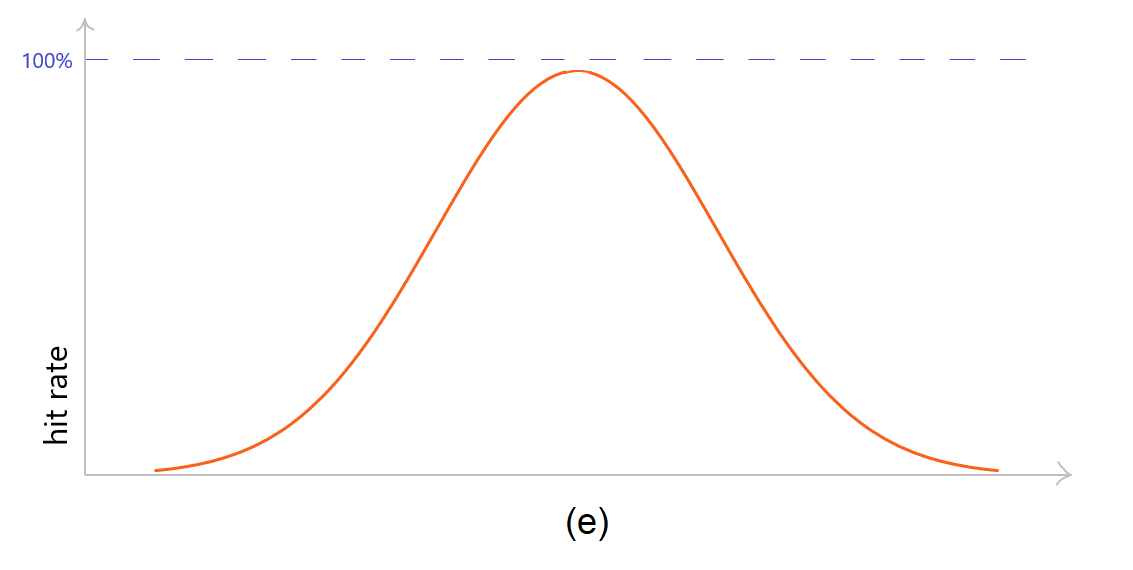
ب) در نیمۀ اول نمودار میزان hit rate افزایش مییابد زیرا به اصل همجواری مکانی توجه بیشتری میشود (طبق توضیحات داده شده در ابتدای پاسخ). اما در نیمۀ دوم نمودار به همجواری زمانی توجه کمتری شده زیرا میزان overwrite زیاد شده و حافظۀ نهان نمیتواند دادههای مفید را برای مدت زیادی در خود نگه دارد و لذا میزان hit rate کاهش مییابد (میزان cache pollution زیادتر می شود و درون حافظۀ نهان دادۀ مفید نداریم). مقدار 100% برای این نمودار نیز یک مقدار مجانبی میباشد.



**تستی:**

در مورد یک حافظۀ نهان نگاشت مستقیم با توجه به اصول همجواری زمانی و مکانی به سوالات زیر پاسخ دهید (از راست به چپ).





الف) فرض کنید در این حافظۀ نهان اندازۀ بلوکهای حافظه ثابت است. با این فرض نمودار hit rate بر حسب اندازۀ حافظۀ نهان کدام یک از شکل های (a) تا (e) می باشد؟

ب) حال فرض کنید اندازۀ حافظۀ نهان ثابت است. با این فرض نمودار hit rate بر حسب اندازۀ بلوکهای حافظۀ نهان کدام یک از شکل های (a) تا (e) می باشد؟

الف) e - b

ب) a - e

ج) d - c

د) b - b

## (امتحان) سوال دوم (اسلاید 6)

برای کاهش Miss Penalty بهتر است اندازۀ بلوکهای حافظۀ نهان کوچکتر باشند یا بزرگتر؟ پاسخ خود را توضیح داده و دو روش موجود برای کاهش اثر این پدیده را مختصر بیان کنید.

**پاسخ**

ابتدا بررسی میکنیم که miss penalty چیست تا به این سوال پاسخ دهیم. زمانی که میخواهیم دادهای را از حافظۀ اصلی بخوانیم و به حافظۀ نهان درخواست میدهیم، اگر داده در حافظۀ نهان یافت نشد علاوه بر هزینۀ جستجو در آن باید هزینۀ خواندن داده از حافظۀ اصلی را نیز بدهیم. داده ها word به word از حافظۀ اصلی به حافظۀ نهان منتقل می شوند، بنابراین زمانی که این انتقال داده طول میکشد رابطۀ مستقیم با تعداد word هایی که میخواهیم منتقل کنیم دارد. همانطور که میدانیم در هر انتقال از حافظۀ اصلی به حافظۀ نهان یک بلوک از کلمات را جابجا میکنیم، بنابراین می توان گفت میزان miss penalty رابطۀ مستقیم با اندازۀ بلوکهای حافظۀ نهان دارد. لذا برای کاهش آن بهتر است اندازۀ بلوکها "کوچکتر" باشد.

از طرفی بنابر اصل همجواری مکانی می دانیم اندازۀ بلوکهای بزرگتر میتواند منجر به نرخ برخورد بالاتری شود، بنابراین به دنبال راهی هستیم که افزایش miss penalty را برای بلوکهای بزرگتر را کاهش دهیم تا برآیند این دو در نهایت بهبود عملکرد سیستم را نتیجه دهد.

دو روش متداول برای این کار وجود دارد که به ازای miss penalty ثابت معطلی پردازنده را به ازای هر miss کاهش میدهد (طبیعتاً این دو روش مقدار miss penalty را تغییر نمیدهند بلکه اثر آن بر روی پردازنده را کاهش میدهند):

1. Early restart: به طور عادی زمانی که حافظۀ نهان یک بلوک از حافظۀ اصلی را درخواست می کند، حافظۀ اصلی کلمات آن بلوک را یکی یکی و به ترتیب به حافظۀ نهان تحویل میدهد. در این تکنیک به جای این که منتظر بمانیم تا تمام بلوک به حافظۀ نهان منتقل شود، هر زمان که در این جابجایی کلمه ای را که پردازنده درخواست داده بود دریافت کردیم آن را مستقیم به پردازنده تحویل میدهیم. بنابراین پیش از رسیدن تمام بلوک پردازنده دادهای که منتظر آن بود را دریافت کرده و میتواند به کار خود ادامه دهد. پس از آن می توانیم به جابجایی باقی بلوکها بپردازیم.
2. Critical-Word-First: جابجایی یک بلوک از حافظه به جای یک کلمه از آن به دلیل این بود که طبق اصل همجواری مکانی بتوانیم درخواستهای احتمالی CPU در آینده را پاسخ دهیم، اما واقعیت این است که در این لحظه تنها به یک کلمۀ مشخص از بلوکی که جابجا می شود نیاز داریم. بنابراین میتوانیم این کلمه را زودتر به پردازنده تحویل دهیم تا مجبور نباشد معطل کلمه هایی که اکنون به آنها نیاز ندارد بشود. از این طریق می توان اثر miss penalty را برای بلوک بندی های بزرگتر کاهش داد.

# (تمرین، ترکیب با سوال x) سوال سوم (اسلاید 6)

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.

## ‌(کوییز و میان ترم تغییر عدد) سوال چهارم (اسلاید 6)

فرض کنید در یک پردازنده از سیاست write-back استفاده میکنیم و مقدار base CPI برابر با 3 میباشد. به علاوه در حدود 15% از دستورات پردازنده مربوط به نوشتن در حافظه بوده و هر بار نوشتن در حافظه به طور متوسط 120 کلاک طول میکشد. وجود دستورات نوشتن بر حافظۀ اصلی پردازنده را چند برابر کندتر کرده است؟ برای کاهش این مشکل چه ترفندی را میتوان به کار برد؟

**پاسخ**

برای مقایسه عملکرد سیستم در دو حالت، CPI واقعی را با base CPI مقایسه میکنیم.

Effective CPI = 3 + 0.15 × 120 = 3 + 18 = 21

بنابراین:

Effective CPI / Base CPI = 21 / 3 = 7

در نتیجه میتوان گفت پردازنده 7 برابر کندتر شده است.

برای کاهش این مشکل باید تا جای ممکن ارجاعات مستقیم پردازنده به حافظۀ اصلی را کاهش دهیم. برای این کار از یک write buffer که یک بافر سریع میان پردازنده و حافظۀ اصلی است استفاده میکنیم تا CPU مجبور نباشد تا نوشتن داده بر حافظه 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.

**تستی:**

فرض کنید در یک پردازنده از سیاست write-back استفاده میکنیم و مقدار base CPI برابر با 3 میباشد. به علاوه در حدود 15% از دستورات پردازنده مربوط به نوشتن در حافظه بوده و هر بار نوشتن در حافظه به طور متوسط 120 کلاک طول میکشد. وجود دستورات نوشتن بر حافظۀ اصلی پردازنده را چند برابر کندتر کرده است؟

الف) 18

ب) 6

ج) 54

د) 7

## (کوییز) سوال پنجم (اسلاید 6)

تحت چه شرایطی بلوکی از حافظۀ نهان dirty تلقی میشود؟ فرض کنید در یک حافظۀ نهان مجموعه انجمنی 4 راهه[[1]](#footnote-0) 512 set داریم. هر بلوک از این حافظه نیز حاوی هشت word چهار بایتی میباشد. در نهایت نیاز به چند بیت dirty در این حافظۀ نهان داریم؟

**پاسخ**

برای اینکه یک بلوک از حافظۀ نهان dirty تلقی شود، تنها کافیست یکی از دادههای آن تغییر کرده باشد. زمانی که پردازنده دادهای را از حافظۀ اصلی درخواست میکند این داده داخل حافظۀ نهان ذخیره میشود. حال اگر پردازنده پس از خواندن داده آن را تغییر دهد و دادۀ تغییر یافته را بر روی حافظۀ نهان بنویسد کل بلوک مربوط به آن داده dirty محسوب می شود زیرا دادهای دارد که مقدار آن با وضعیت فعلی حافظۀ اصلی مطابقت ندارد.

از آنجایی که هر بیت dirty به یک بلوک از حافظه متناظر میشود کافیست تعداد بلوکهای حافظۀ نهان را بدست آوریم. تعداد بلوکها در حافظۀ نهان توصیف شده برابر است با:

4 way × 512 blocks/way = 2048 blocks

بنابراین این حافظۀ نهان در مجموع نیاز به 2048 بیت dirty دارد.

**تستی:**

چه شرطی کافیست تا بلوکی از حافظۀ نهان dirty تلقی شود؟ فرض کنید در یک حافظۀ نهان مجموعه انجمنی 4 راهه[[2]](#footnote-1) 512 set داریم. هر بلوک از این حافظه نیز حاوی هشت word چهار بایتی میباشد. در نهایت نیاز به چند بیت dirty در این حافظۀ نهان داریم؟

الف) تمام داده های آن تغییر کرده باشد - 211 بیت

ب) تنها یکی از داده های آن تغییر کرده باشد - 214 بیت

ج) تمام داده های آن تغییر کرده باشد - 214 بیت

د) تنها یکی از داده های آن تغییر کرده باشد - 211 بیت

## (امتحان) سوال ششم (اسلاید 6)

پردازندهای با فرکانس 3GHz کار میکند. همچنین هر کلاک گذرگاه حافظه معادل 3 کلاک پردازنده بوده و پهنای این گذرگاه به اندازۀ یک کلمۀ 2 بایتی است. فرض کنید هنگام خواندن دادهای از حافظۀ نهان miss رخ داده و می خواهیم یک بلوک 8 کلمه ای از داده را از حافظۀ اصلی بخوانیم. در این روند 8 کلاک گذرگاه[[3]](#footnote-2) برای دسترسی به دادۀ درون حافظه، 1 کلاک برای انتقال آدرس داده از حافظۀ نهان به حافظۀ اصلی و 2 کلاک برای انتقال داده از حافظۀ اصلی به حافظۀ نهان زمان صرف می شود. در این شرایط نرخ ارسال داده میان حافظۀ نهان و حافظۀ اصلی چند بایت بر ثانیه است؟

**پاسخ**

داده های موجود در مساله به شرح زیر هستند:

* CPU frequency: 3GHz 🡪 3 × 109 clocks/second
* 1 CPU clock = 3 bus cycle
* 1 word = 2 bytes
* 1 block = 8 words
* 1 bus cycle needed for address transfer
* 8 bus cycles needed per DRAM access
* 2 bus cycle needed per data transfer

در نتیجه می توان نوشت:

Miss penalty = 1 + 8 × 4 + 2 × 4 = 41 bus cycles

بنابراین

Band width = (4 × 2 bytes) / 41 cycles ≈ 0.2 bytes/cycle

هر کلاک پردازنده معادل 3 کلاک گذرگاه میباشد، بنابراین گذرگاه در یک ثانیه 109 سیکل را طی خواهد کرد. پس در نهایت می توان نوشت:

Band width = 0.2 bytes/cycle × 109 cycles/second = 2 × 108 bytes/second

# (تمرین)سوال هفتم (اسلاید 6)

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.

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

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

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

## کوییز امتحان

ح) کلاکی که CPU از آن استفاده میکند معمولا با کلاک گذرگاه حافظۀ اصلی یکسان نیست.

ب) حافظۀ اصلی خود مدیریت می کند که داده درون Instruction Cache ذخیره شود یا Data Cache.

د) به فرایندی که در آن CPU درون Instruction Cache دستوری را مینویسد store گفته میشود.

ه) اگر زمان اجرای دستوری نیاز به دادهای داشته باشیم که در حافظۀ نهان وجود نداشته باشد دستور در CPU معطل میماند.

**پاسخ**

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 – حافظه از این که چیزی که تحویل میدهد داده است یا دستور خبر ندارد، بلکه CPU است که میداند نیاز به داده داشته یا دستور و به D-Cache رجوع کند یا I-Cache. پس از آن هر یک از این حافظه های نهان بسته به نیاز به حافظۀ اصلی رجوع میکنند، اما همچنان حافظه خبر ندارد که کدام یک به او درخواست دادهاند.

ج) 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.

د) غلط – پردازنده هیچ گاه دستوری را روی حافظه نمی نویسد و دستورات برای CPU حالت Read-Only دارند. به فرآیند نوشتن داده بر روی Data Cache اما store گفته میشود.

ه) صحیح – در این صورت پردازنده ناچار است پردازش دستور را متوقف کند تا داده از حافظۀ اصلی خوانده شود.

و) 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.

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

ح) صحیح – کلاکی که پردازنده از آن استفاده میکند نسبت به کلاک گذرگاه حافظه سریعتر است.

# (تمرین) سوال هشتم (اسلاید 6)

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 گذرگاه داده معادل 3 کلاک پردازنده میباشد. در این روند 1 کلاک گذرگاه[[4]](#footnote-3) 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 نتیجهگیری 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:

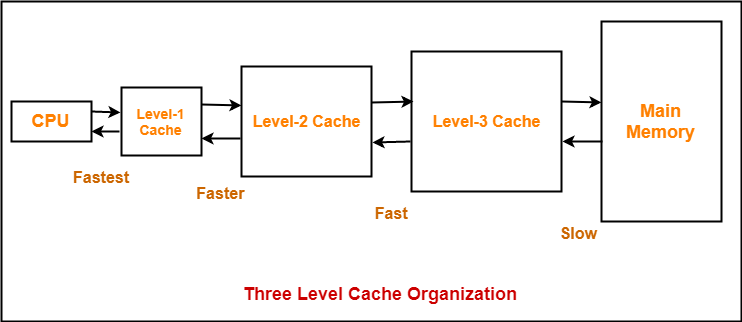
* I-cache: 0.02 × 135 = 2.7
* D-cache: 0.4 × 0.04 × 135 = 2.16

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.

# (تمرین) سوال نهم (امتیازی) (اسلاید 6)

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.

\* راهنمایی: سعی کنید AMAT را ابتدای برای آخرین لایه پیدا کرده و از آن برای پیدا کردن AMAT در لایۀ قبلی استفاده 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))

# Arta lecture 7

**سوال اول (اسلاید 7)**فرض کنید حافظه نهانی داریم که به اندازه 4 بلوک است. محتوای آدرس های زیر به ترتیب از چپ به راست درخواست شده است، وضعیت موفقیت آنها و نرخ موفقیت را برای هر یک از سیاست های جایدهی زیر بدست آورید. ([برای تبدیل اعداد هگز به دسیمال](https://www.rapidtables.com/convert/number/how-hex-to-decimal.html))

3A, 4C, 13, 3B, 11, 3A, 4C, 12, 45, A3, D9, D8, 45, 12, B2, 45, 17, D8, D9, A3

* Direct Mapping
* 2 - way set associative (with FIFO)
* Fully associative (with LRU)

**پاسخ**

Direct mapping :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 58 | 76 | 19 | 59 | 17 | 58 | 76 | 18 | 69 | 58 |
| Block | 2 | 0 | 3 | 3 | 1 | 2 | 0 | 2 | 1 | 2 |
| h/m | m | m | m | m | m | m | h | m | m | m |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 217 | 216 | 69 | 18 | 178 | 69 | 23 | 216 | 217 | 58 |
| Block | 1 | 0 | 1 | 2 | 2 | 1 | 3 | 0 | 1 | 2 |
| h/m | m | m | m | m | m | h | m | h | m | m |

Hits : 3 , Total : 20

Hit rate : 0.15

2 - way set associative (with FIFO) :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 58 | 76 | 19 | 59 | 17 | 58 | 76 | 18 | 69 | 58 |
| Set | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
|  | 3A | 4C | 13 | 3B | 11 | 4C | 4C | 12 | 45 | 3A |
|  | - | 3A | - | 13 | 3B | 3A | 3A | 4C | 11 | 12 |
| h/m | m | m | m | m | m | h | h | m | m | m |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 217 | 216 | 69 | 18 | 178 | 69 | 23 | 216 | 217 | 58 |
| Set | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
|  | D9 | D8 | D9 | 12 | B2 | D9 | 17 | D8 | 17 | A3 |
|  | 45 | 3A | 45 | D8 | 12 | 45 | D9 | B2 | D9 | D8 |
| h/m | m | m | h | m | m | h | m | m | h | m |

Hits : 5 , Total : 20

Hit rate : 0.25

Fully associative (with LRU) :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | 3A |
|  | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 |
|  | - | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 |
|  | - | - | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C |
| h/m | m | m | m | m | m | m | m | m | m | h |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
|  | 3A | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 |
|  | 45 | 3A | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 |
|  | 12 | 45 | 3A | D9 | D8 | D8 | 12 | B2 | 45 | 17 |
| h/m | m | m | h | m | m | h | m | m | m | m |

Hits : 3 , Total : 20

Hit rate : 0.15

# (تمرین) سوال دوم (اسلاید 7) [link](https://www.gatevidyalay.com/set-associative-mapping-practice-problems/)

حافظه نهانی به اندازه 16KByte داریم که اندازه هر بلوک آن 256Byte می باشد. سیاست جایدهی این حافظه نهان 2way set associatvie 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.

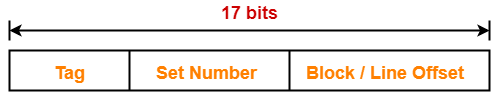
**ب-** برای ذخیره tag در این حافظه نهان مجموعا از چند byte استفاده می شود.

**پاسخ**

**الف**

اول ببینیم که برای آدرس دهی هر word در حافظه اصلی به چند بیت نیاز داریم:

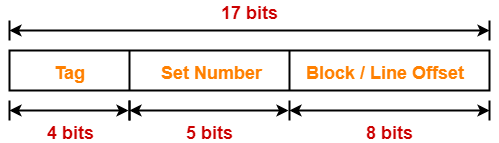
اندازه حافظه اصلی 128Kbyte هست یعنی پس یعنی برای آدرس دهی کلش به 17 بیت نیاز داریم.



حالا به ترتیب میبینیم که بخش block (آدرس دهی word داخل بلوک) و بخش set number (آدرس دهی یا همون مشخص کردن set داخل حافظه نهان) به چند بیت نیاز دارند و در نهایت تعداد بیت های بخش tag هم خود به خود به دست میاد.

از اونجایی که اندازه هر بلوک 256Byte یا می باشد پس برای آدرس دهی آن به 8 بیت نیاز داریم.  
حالا برای بخش set number باید ببینیم چند تا set داخل حافظه نهان وجود داره. خوب اول تعداد بلوک های توی حافظه نهان رو حساب می کنیم که میشه 16Kbyte/256Byte که میشه که میشه 64. داخل هر set هم 2 بلوک داریم. پس نهایتا ما 64/2=32 set داخل حافظه نهان داریم پس به 5 بیت برای آدرس دهیشون نیاز داریم.

خوب حالا در نهایت تعداد بیت آدرس دهی tag هم میشه .



**ب**

خوب تو این قسمت می خواهیم ببینیم برای ذخیره tag ها مجموعا چند بایت نیاز داریم. خیلی ساده تعداد بیت هر tag رو در تعداد بلوک های داخل حافظه نهان (64) ضرب می کنیم (چون به ازای هر بلوک یک tag داریم) که میشه :

256bits -> 32 bytes

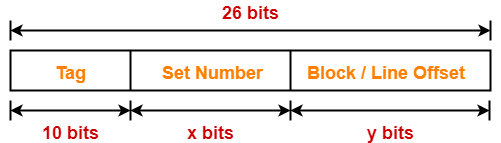
## (میانترم)سوال سوم (اسلاید 7) [link](https://www.gatevidyalay.com/set-associative-mapping-practice-problems/)

فرض کنید در یک سیستم حجم حافظه اصلی 64MByte می باشد و در آدرس دهی 10 بیت برای بخش tag درنظر گرفته شده. اگر سیاست جای دهی حافظه نهان آن 4way set associative باشد، اندازه این حافظه نهان چند بایت است ؟

**پاسخ**

برای محاسبه حجم حافظه نهان ابتدا باید ببینیم تعداد کل بیت های آدرس دهی چند تاست:

بعد از تعداد بیت های tag کم می کنیم. که میشه جمع بیت های block و set :



با توجه به اینکه نمیدونیم داخل هر بلوک چند word وجود داره نمیتونیم جدا جدا x و y رو حساب کنیم اما دانستن جمعشون برای ما کفایت می کنه. حجم حافظه نهان برابر با (تعداد set ها داخل حافظه نهان \* تعداد بلوک های هر set \* تعداد word هر بلوک). تعداد بلوک های هر set که با توجه به 4way set associative بودن حافظه میدونیم 4 هست، جمع x و y هم میدونیم برابر با 16. در نتیجه :

cache memory size = =

**سوال چهارم (اسلاید 7) (تمرین سری 5، ca\_fall2021)**در برنامه های کاربردی عام منظوره، درخواست های دسترسی به حافظه معمولا پراکندگی نامتوازن دارند، این پراکندگی سبب می شود درخواست بر روی بخشی از انجمنها (set) زیاد باشد در حالی که باقی حافظه ی نهان به طور کارآمد مورد استفاده قرار نمیگیرد. برای حل این مشکل حافظه های نهان V-Way ارائه شده اند. با مطالعه ی لینک به سوالات پاسخ دهید .

الف) تغییرات به وجود آمده در بخش tag را بیان کنید.

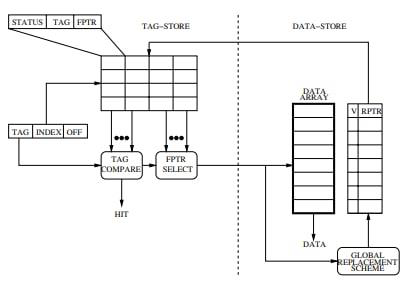
ب) ایده ی کلی و نحوه ی عملکرد این روش را بیان کنید.

<https://sci-hub.mksa.top/10.1109/ISCA.2005.52>

**پاسخ**

حافظه ی نهان V-Way از دو ساختار جدا شده تشکیل شده است store-tag و store-data ویژگی حافظه ی نهان V-Way وجود تعداد tag-store های بیشتر از خطوط داده ی ما است. در این حافظه های نهان، نسبت tag به data یا به اختصار TDR اهمیت زیادی دارد. در صورتی که 1=TDR باشد، حافظه ی نهان ما معادل همان حافظه هایی ست که قبلا می شناختیم. اگر

TDR > 1 باشد، ساختار حافظه ی نهان تغییراتی خواهد داشت. برای مثال اگر 2 = TDR باشد یعنی باید تعداد تگها دو برابر خطوط داده باشد و ساختار کلی حافظه ی نهان مشابه تصویر زیر خواهد شد .



هر ورودی tag-store حاوی یک سری اطلاعات وضعیتی شامل بیت معتبر یا valid ، بیت کثیف یا dirty و اطلاعات جایگزینیست. علاوه بر این اطلاعات، خود tag و یک اشاره گر رو به جلو (FPTR) نیز در ورودی قرار دارد که آن ورودی tag-store را به ورودی منحصر به فردی در data-store نگاشت میدهد. اگر بیت معتبر در ورودی tag-store پاک شود، تمام اطلاعات دیگر در ورودی، از جمله FPTR ، نامعتبر در نظر گرفته می شود. هر ورودی data-store حاوی یک خط داده، یک بیت معتبر و یک اشاره گر معکوس (RPTR) است که برعکس FPTR عمل می کند و یک ورودی منحصر به فرد را در tag-store شناسایی می کند. برای هر ورودی معتبر tag-store ،یک جفت (FPTR ، RPTR) وجود دارد که به یکدیگر اشاره می کنند .

**سوال پنجم (اسلاید 7)**فرض کنید حافظه نهانی داریم که به اندازه 8 بلوک جا دارد. محتوای آدرس های زیر به ترتیب از چپ به راست درخواست شده است، وضعیت موفقیت آنها و نرخ موفقیت را برای هر یک از سیاست های جایدهی زیر بدست آورید. ([برای تبدیل اعداد هگز به دسیمال](https://www.rapidtables.com/convert/number/how-hex-to-decimal.html))

5, C, 11, 12, 13, 4, 5, 20, 21, 11, 12, 13, 1, 2A, 3B, C, 11, 12, 13, 2A

* 2 - way set associative (with FIFO)
* fully associative (with LRU)

**پاسخ**

2 - way set associative (with FIFO)

| Hex | 5 | C | 11 | 12 | 13 | 4 | 5 | 20 | 21 | 11 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 5 | 12 | 17 | 18 | 19 | 4 | 5 | 32 | 33 | 17 |
| Set | 1 | 0 | 1 | 2 | 3 | 0 | 1 | 0 | 1 | 1 |
|  | 5 | C | 11 | 12 | 13 | 4 | 11 | 20 | 21 | 21 |
|  | - | - | 5 | - | - | C | 5 | 4 | 11 | 11 |
| h/m | m | m | m | m | m | m | h | m | m | h |

| Hex | 12 | 13 | 1 | 2A | 3B | C | 11 | 12 | 13 | 2A |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 18 | 19 | 1 | 42 | 59 | 12 | 17 | 18 | 19 | 42 |
| Set | 2 | 3 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 2 |
|  | 12 | 13 | 1 | 2A | 3B | C | 11 | 2A | 3B | 2A |
|  | - | - | 21 | 12 | 13 | 20 | 1 | 12 | 13 | 12 |
| h/m | h | h | m | m | m | m | m | h | h | h |

Hits : 7 , Total : 20

Hit rate : 0.35

# Javad lecture 8

1- حافظه نهانی با ساختار 2way Set Associative در نظر بگیرید که 4 بلاک دارد و هر بلاک ظرفیت معادل با 3 word دارد. شماره بلاک‌های زیر را از چپ به راست توسط CPU صدا زده می‌شوند. حافظه نهان در ابتدا خالیست.

الف) اگر سیاست جایگزنی ما LRU باشد ساختار کلی cache و set ها به همراه hit rate مشخص کنید

ب) اگر ساختار cache به صورت Fully associative باشد و سیاست ما Most-recently used باشد آیا عملکرد بهتری از منظر hit-rate خواهیم داشت؟

4,7,8,12,15,6,14,13,6,14,15,16

الف)

| 16 | 15 | 14 | 6 | 13 | 14 | 6 | 15 | 12 | 8 | 7 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | #set | |
| 16 | 15 | 14 | 6 | 13 | 14 | 6 | 15 | 12 | 8 | 7 | 4 | bloks in each set | |
| 14 | 13 | 6 | 14 | 15 | 6 | 12 | 7 | 8 | 4 | - | - |
| m | h | h | h | m | m | m | m | m | m | m | m | h/m | |

hit rate = 3/12

| 16 | 15 | 14 | 6 | 13 | 14 | 6 | 15 | 12 | 8 | 7 | 4 | #block  in MM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | bloks  in  set | |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | - |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | - | - |
| 16 | 15 | 14 | 6 | 13 | 14 | 6 | 15 | 12 | - | - | - |
| m | m | m | m | m | m | m | m | m | m | m | m | h/m | |

hit rate = 0/12

# (تمرین)سوال دوم

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

# (تمرین) موراد زیر را 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.

ب) عملیات بالا را با یک cache حاوی هشت بلوک تکرار کنید و بعد از تعریف اختلال بلیدی (Bélády's anomaly) بررسی کنید که آیا این اختلال در این مثال رخ می‌دهد یا خیر.

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

hit rate = 1/14

ج)

| 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

1. 4-Way Set associative [↑](#footnote-ref-0)
2. 4-Way Set associative [↑](#footnote-ref-1)
3. Bus cycle [↑](#footnote-ref-2)
4. Bus cycle [↑](#footnote-ref-3)