به نام خداوند بخشنده مهربان

معماری کامپیوتر

دانشکده مهندسی کامپیوتر

دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین نهم

سوال ۱

۱. آ) فرض کنید یک واحد حافظه نهان[1] به اندازه ۳ و کاملا خالی در اختیار دارید، دسترسی‌های حافظه زیر به ترتیب از راست به چپ برای شما ارسال می‌شود با الگوریتم FIFO آن‌ها را درون حافظه نهان قرار دهید و تعداد missهای حافظه را محاسبه کنید.

$$4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5$$

ب) حال فرض کنید یک حافظه به اندازه ۴ در اختیار دارید، مجددا دسترسی‌های بخش الف به حافظه رخ می‌دهند و باید آن‌ها را با همان الگوریتم FIFO در حافظه خود قرار دهید. تعداد missهای حافظه را به دست آورید.

ج) به پدیده‌ای که مشاهده کردید Bélády's anomaly می‌گویند، در مورد آن تحقیق کنید.

(آ) .

5	1	2	3	4	5	3	4	1	2	3	4	
miss	miss	miss	miss	miss	miss	hit	hit	miss	miss	miss	miss	10 miss

(ب) .

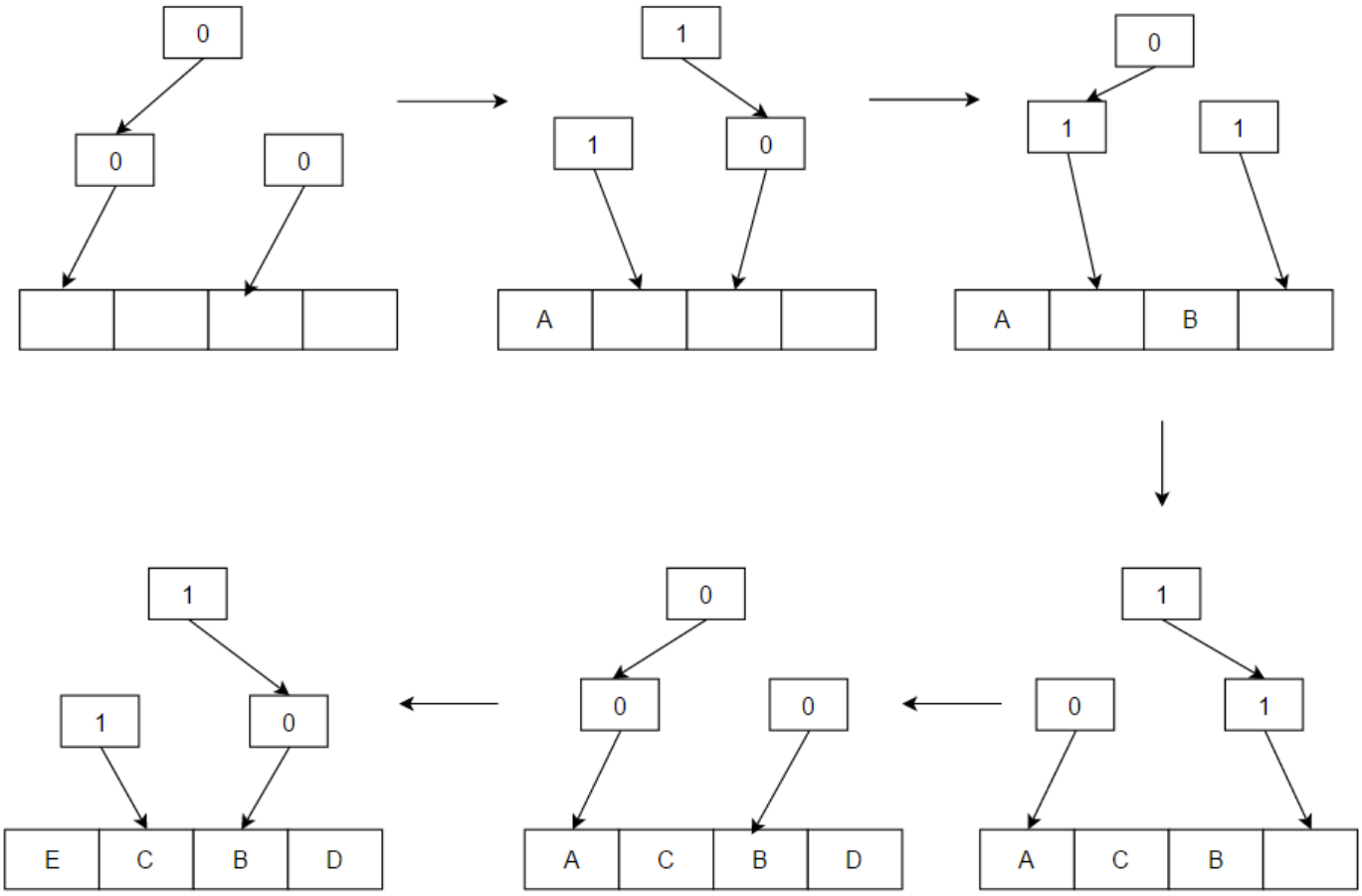
5	1	2	3	4	5	3	4	1	2	3	4	
miss	miss	miss	miss	miss	miss	hit	hit	miss	miss	miss	miss	10 miss

(ج) این پدیده اکثرا زمانی رخ می دهد که از سیاست هایی مثل FIFO استفاده کنیم و در این شرایط، با افزایش فضای حافظه ی نهان ممکن است شاهد کاهش یا ثابت ماندن hit-rate در wrokload های خاص باشیم اما در الگوریتم های پشته ای مانند LRU شاهد همچین پدیده ای نیستیم و با افزایش میزان cache قطعا شاهد کاهش تعداد miss ها خواهیم بود.

سوال ۲

۲. در مورد الگوریتم‌های Pseudo LRU و Clock تحقیق کنید و نحوه کارکرد آن‌ها را توضیح دهید. سپس به صورت خلاصه توضیح دهید که چرا به نظر شما استفاده از خود LRU در سخت افزار واقعی کار منطقی نیست.

سیاست Pseudo LRU یکی از مشتقات LRU است که به جای نگه داشتن سن دقیق یک بلاک کش، تقریبی از آن را نگه می دارد. به این صورت است که نیاز است یک درخت نگه داریم که تعداد برگ های آن با تعداد بلوک های کش برابری می کند. در هر راس نیز تک بیتی نگه میداریم که در صورت صفر بودن به سمت چپ و در صورت یک بودن از آن نود به سمت راست می رویم و این روند را به قدری ادامه می دهیم تا به یک بلاک حافظه برسیم و پس از عبور از هر نود، بیت داخل آن را toggle می کنیم.



الگوریتم Clock نیز به این صورت عمل می کند که بعد هر لود در بلاک حافظه، پوینتر به خانه ی بعدی می رود و در صورت رسیدن به انتها به اول بر می گردد، همچنین برای هر خانه، یک بیت شاتس دوباره نگه می داریم، هنگامی که کش پر شود، پونتر در هر خانه ای که بود به بیت شانس دوباره ی آن نگاه می کند، اگر یک بود، آن را صفر می کند و اگر صفر بود آن بلاک را ریپلیس می کند، و با هر دسترسی به بلاک، بیت شانس دوباره ی آن یک می شود. این پوینتر هنگام ریپلیس کردن انقدر به حرکت ادامه می دهد تا بالاخره موفق به خالی کردن یک خانه شود.
الگوریتم LRU در سطح سخت افزار الگوریتم خوبی نیست، زیرا برای استفاده از آن نیاز به tracking های زیادی داریم، برای هر بلاک کش نیاز است یک شمارنده یا یک نگه دارنده ی زمان آخرین دسترسی باشد که سخت افزارهای اختصاصی خود را دارند و الگوریتم های کم هزینه تر و بهتری موجود است.

سوال ۳

۳. یک پردازنده، حافظه‌ای byte-addressable دارد که آدرس‌های آن ۳۲ بیتی هستند. یک حافظه نهان به‌اندازه‌ی ۵۱۲ کیلوبایت داریم که اندازه‌ی هر بلوک آن ۳۲ بایت است. فرض کنید این حافظه نهان از نوع two-way set associative است. به سوالات زیر پاسخ دهید:

آ) این حافظه نهان درمجموع چند بلوک دارد؟

ب) چند set دارید؟

ج) فیلدهای tag و index و offset چندبیتی هستند؟

(آ) برای تعداد بلاک ها داریم:

$$\text{num}_{\text{blocks}} = \frac{\text{cache size}}{\text{block size}} = \frac{512\text{KB}}{32\text{B}} = \frac{2^{19}}{2^5} = 2^{14} \quad \text{blocks}$$

(ب) برای تعداد مجموعه ها نیز داریم:

$$\text{num}_{\text{set}} = \frac{\text{num}_{\text{blocks}}}{\text{num}_{\text{lines}}} = \frac{2^{14}}{2} = 2^{13} \quad \text{sets}$$

(ج) داریم:

$$\begin{aligned}
\text{offset} &= \log(\text{block size}) = 5 \\
\text{index} &= \log(\text{num}_{\text{sets}}) = 13 \\
\text{tag} &= 32 - (\text{offset} + \text{index}) = 14
\end{aligned}$$

سوال ۴

۴. یک سیستم از حافظه ۸ گیگابایتی با کلمات ۶۴ بیتی استفاده می‌کند. هر بلوک حافظه ۱۶ کلمه را در خود قرار می‌دهد. اگر قرار باشد از یک حافظه نهان نگاشت مستقیم [۲] متشکل از ۱۲۸ بلوک استفاده کنیم، نحوه آدرس‌دهی را مشخص کنید. اگر به جای این حافظه نهان از یک حافظه نهان 4-way set associative استفاده کنیم، آدرس‌دهی به چه صورت خواهد بود؟

فرض می کنیم سیستم word addressable است. در حالت ۴-way set associative داریم.

$$\text{word size} = 2^6 \quad \text{bit}$$
$$\text{memory size} = 2^{36} \quad \text{bit} = 2^{30} \quad \text{words}$$
$$\text{block size} = 2^4 \quad \text{words}$$
$$\text{number of blocks} = 2^7 \quad \text{blocks}$$
$$\text{number of sets} = \frac{2^7}{2^2} = 2^5 \quad \text{sets}$$
$$\text{address size} = \log(\text{memory size}) = 30 \quad \text{bits}$$
$$\text{offset size} = \log(\text{block size}) = 4 \quad \text{bits}$$
$$\text{set index size} = \log(\text{number of sets}) = 5 \quad \text{bits}$$
$$\text{tag size} = \text{address size} - (\text{offset size} + \text{set index size}) = 30 - (4 + 5) = 21 \quad \text{bits}$$

برای حالت نگاشت مستقیم نیز کافیست تعداد مجموعه ها را برابر با تعداد بلاک ها بگیریم که در این صورت ست ایندکس برابر با ۷ بیت و تگ بیت برابر با ۱۹ بیت می شود.

سوال ۵

۵. کد زیر را در نظر بگیرید:

```
for (i = 0; i < 64; i++) {
    for (j = 0; j < 64; j++) {
        sum1 += a[i][j];
    }
    for (j = 0; j < 32; j++){
        sum2 += b[i][2*j];
    }
}
```

در صورتی که خط‌های ۳ و ۵ در حافظه نهان باشند ۱۰ چرخه و سایر خطوط به ۴ چرخه زمانی نیاز دارند. اگر در حافظه نهان MISS اتفاق بیفتد ۴۰ چرخه برای انتقال داده به حافظه نهان نیاز است. فرض کنید حافظه نهان در ابتدا خالی باشد.

آ) در صورت استفاده از حافظه نهان 2-way associative با سیاست جایگذاری LRU شامل ۱۶ جایگاه و با بلوک‌های ۱۶ بایتی چند چرخه برای اجرا لازم است؟

ب) چه نوع محلیتی در کد بالا مشهودتر است؟ توضیح دهید.

ج) تغییر کدام ویژگی‌ها حافظه نهان می‌تواند باعث بهبود سرعت اجرا شود؟

(آ) با توجه به ۲-way associative بودن، حافظه نهان دارای ۸ set خواهد بود. حال در ابتدا با ورود به خط ۳، ابتدا درخواست به کش ، miss می شود (+۱۰)، سپس داده از حافظه اصلی انتقال داده می شود (+۴۰)، و دوباره از حافظه ی نهان خوانده می شود (+۱۰) پس به ازای هر miss، ۶۰ کلاک نیاز است. پس از این miss ، ۳ اجرای بعدی آن hit می شود و دوباره اجرای ۴ام miss می شود، همچنین از آنجایی که کلا ۶۴ عدد خوانده می شود و کش نیز ۱۶ جایگاه ۴ عددی دارد، و حافظه ها متوالی است، نیازی به جایگذاری در یک حلقه نداریم. برای خط ۶ نیز همینگونه است با این تفاوت که در این خط به ازای هر hit یک miss داریم ولی همچنان چون آخرین آدرس فاصله اش با اولی آدرس بیشتر از ۶۴ نیست باز هم نیازی به جایگذاری نیست.
پس برای خط اول یکبار ۴ حلقه برای $i = ۰$ سپس $۴(n+۱) = ۴ \times ۶۵$ حلقه برای شرط، $۴n = ۴ \times ۶۴$ حلقه هم برای $i++$. سپس برای خط ۲ نیز به طریق مشابه داریم $۴n = ۴ \times ۶۴$ حلقه برای $j = ۰$، $۴n(k+۱) = ۴ \times ۶۴ \times ۶۵$ حلقه برای شرط و $۴nk = ۴ \times ۶۴ \times ۶۴$ حلقه برای $j++$، حال برای خط ۳، به ازای هر miss سه hit خواهیم داشت پس در کل $nk * (\frac{۶۰+۳*۱۰}{۴}) = ۶۴ * ۶۴ * ۲۲/۵$ حلقه نیاز است، برای خط ۵ نیز دقیقا مشابه خط ۲ محاسبه می کنیم و خط ۶ نیز در کل $nk * (\frac{۶۰+۱۰}{۲}) = ۶۴ * ۳۲ * ۳۵$ حلقه نیاز است. در مجموع داریم:

$$4 + 4 * 65 + 4 * 64 + 64 * (4 + 4 * 65 + 4 * 64 + 64 * (22.5) + 4 + 4 * 33 + 4 * 32 + 32 * (35)) = 214536 \quad \text{clock}$$

(ب) چون داریم دسترسی هایی به آدرس های متوالی می کنیم بیشتر شاهد محلیت مکانی هستیم.

(ج) در این شرایط افزایش سایز هر بلاک می تواند به افزایش سرعت اجرا کمک کند.

سوال ۶

۶. یک حافظه اصلی به بزرگی ۲۵۶K کلمه و یک حافظه نهان به بزرگی ۴ بلوک ۴ کلمه‌ای موجود است. فرض کنید از روش نگاشت مستقیم استفاده می‌کنیم در این صورت با فرض خالی بودن حافظه نهان، نرخ برخورد[۳] در انتهای صدور آدرس‌های ذیل از چپ به راست کدام است؟

170, 257, 168, 246, 176, 175, 176, 177, 175, 176, 177, 175, 176, 177, 176, 175, 174, 173, 172, 171, 170, 169, 168, 167, 168, 165, 164

170	miss
257	miss
168	hit
246	miss
176	miss
175	miss
176	hit
177	hit
175	hit
176	hit
177	hit
175	hit
176	hit
177	hit
176	hit
175	hit
174	hit
173	hit
172	hit
171	hit
170	hit
169	hit
168	hit
167	miss
168	hit
165	hit
164	hit

$$\text{hit rate} = \frac{21}{27}$$