دانشکده مهندسی کامپیوتر

معماری کامپیوتر

نیم‌سال دوم ۱۴۰۳-۱۴۰۲

تمرین امتیازی | **مهلت تحویل ساعت ۲۴ روز جمعه ۱۸ خرداد**

به موارد زیر توجه کنید:

۱- حتما نام و شماره دانشجویی خود را روی پاسخ‌نامه بنویسید.

۲- کل پاسخ تمرینات را در قالب یک فایل pdf با شماره دانشجویی خود نام‌گذاری کرده در سامانه CW بارگذاری کنید.

۳- این تمرین ۱۰۰ نمره دارد که معادل یک نمره از نمره کلی درس است.

۴- در صورت مشاهده هر گونه مشابهت نامتعارف هر دو (یا چند) نفر کل نمره این تمرین را از دست خواهند داد.

۱- (۲۰ نمره) به سوالات زیر پاسخ دهید. در هر بند فرض کنید اگر برای اجرای بخشی از برنامه از i پردازنده استفاده شود، سرعتِ اجرای آن بخش i برابر می‌شود.

الف- فرض کنید x کسری از یک برنامه است که فقط به صورت سریال قابل اجرا است. بقیهٔ برنامه را می‌توان روی هر تعداد پردازنده، به صورت موازی اجرا کرد. اگر بخواهید با داشتن p پردازنده، به تسریع S برسید، بیشینهٔ x چقدر است؟ بیشینهٔ x را به ازای $p = 10$ و $s = 8$ به‌دست آورید.

$$s = \frac{T}{xT + \frac{(1-x)T}{p}} = \frac{p}{xp + 1 - x} \quad \Longrightarrow \quad x = \frac{p-s}{s(p-1)}$$

$$8 = \frac{10}{10x + 1 - x} \quad \Longrightarrow \quad 72x = 2 \quad \Longrightarrow \quad x = 0.028$$

یعنی اگر بخواهیم با ۱۰ پردازنده به تسریع ۸ برسیم، فقط ۲٫۸٪ از برنامه می‌تواند به صورت سریال اجرا شود.

ب- حال فرض کنید $F(i,p)$ کسری از یک برنامه است که با داشتن p پردازنده می‌تواند روی i پردازنده اجرا شود. بنابراین:

$$\sum_{i=1}^{p} F(i,p) = 1$$

با استفاده از قانون آمدال، تسریع حاصل از به‌کارگیری p پردازنده برای اجرای برنامه را به‌دست آورید.

$$speedup = \frac{1}{\sum_{i=1}^{p} \frac{F(i,p)}{i}}$$

ج- اجرای یک برنامه روی یک پردازنده T ثانیه زمان می‌برد. اگر بدانیم بخش‌های مختلف برنامه طبق جدول زیر قابلیت اجرا روی چند پردازنده را دارند، تسریعِ حاصل از اجرای این برنامه را زمانی که ۸ پردازنده داریم حساب کنید.

Fraction of T	20%	20%	10%	5%	15%	20%	10%
processors (p)	1	2	4	6	8	16	128

$$ExecTime(8) = \left(0.2 + \frac{0.2}{2} + \frac{0.1}{4} + \frac{0.05}{6} + \frac{0.45}{8}\right)T = 0.39T$$

$$speedup = \frac{T}{0.39T} = 2.57$$

۲- (۲۰ نمره) فرض کنید برنامه‌ای قرار است روی یک کامپیوتر چندپردازنده با ساختار NUMA (Nonunifor Memory Access) اجرا شود. در این ساختار، هر پردازنده یک حافظهٔ محلی دارد ولی می‌تواند در صورت لزوم به حافظهٔ سایر پردازنده‌ها هم دسترسی داشته باشد. طبعا دسترسی به حافظهٔ محلی بسیار سریع‌تر از دسترسی به حافظه‌های دور (حافظه‌های سایر پردازنده‌ها) خواهد بود.
فرض کنید در یک کامپیوتر ۳۲ پردازنده‌ای، ارجاع به حافظهٔ محلی در ۱۰ و ارجاع به حافظهٔ راه دور در ۱۰۰ نانوثانیه انجام می‌شود. نرخ ساعت این پردازنده ۴ گیگاهرتز است و CPI پایه (زمانی که همه ارجاع‌ها در حافظهٔ نهان محلی قرار دارند) برابر با ۰/۵ است. می‌دانیم ۰/۲٪ از دستورات نیاز به ارجاع به حافظهٔ دور دارند و نرخ برخورد در حافظهٔ نهان محلی ۹۰٪ است. در این صورت CPI متوسط چقدر خواهد بود؟

$$CPI_{localMemAccess} = 0.998 \times (1 - 0.9) \times 10 \times 4 = 3.992$$

$$CPI_{remoteMemAccess} = 0.002 \times 100 \times 4 = 0.8$$

$$CPI_{avg} = CPI_{base} + CPI_{localMemAccess} + CPI_{remoteMemAccess} = 0.5 + 3.992 + 0.8 = 5.3$$

۳- (۲۰ نمره) ابتدا یک مدار رسم کنید که یک عدد چهار بیتی را با یک جمع کند. سپس مدار دیگری رسم کنید که یک عدد چهاربیتی را منهای یک کند. درنهایت این دو مدار را با هم طوری ترکیب کنید و مداری رسم کنید که یک ورودی کنترلی x دارد که اگر صفر باشد، عدد چهار بیتی ورودی با یک جمع می‌شود و اگر یک باشد، عدد چهار بیتی ورودی منهای یک می‌شود. هر سه مدار را با حداقل تعداد گیت‌های پایه بسازید.

پاسخ:

برای این که عددی را با یک جمع کنیم یا منهای یک کنیم، باید از یک نیم‌افزا (half adder) و سه تمام‌افزا (full adder) استفاده کنیم که یکی از ورودی‌های آن ۰۰۰۱ یا ۱۱۱۱ باشد. اگر بخواهیم مدار را با کمترین تعداد گیت‌های پایه بسازیم، کافی است داخل جمع‌کننده‌ها را ساده کنیم.

برای بیت اول داریم:

$$s_0 = a_0 \oplus 1 = a'_0, \quad c_1 = a_0.1 = a_0$$

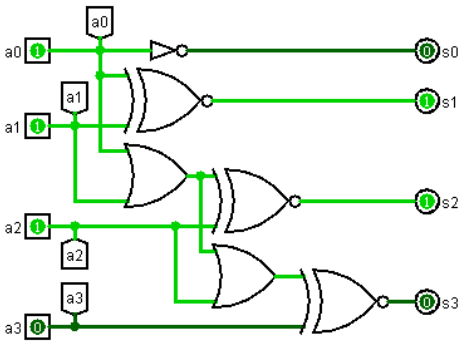
برای بیت دوم هنگام جمع با یک، داریم:

$$s_1 = a_1 \oplus 0 \oplus a_0 = a_1 \oplus a_0 \quad c_2 = a_1.0 + a_0.0 + a_1.a_0 = a_1.a_0$$

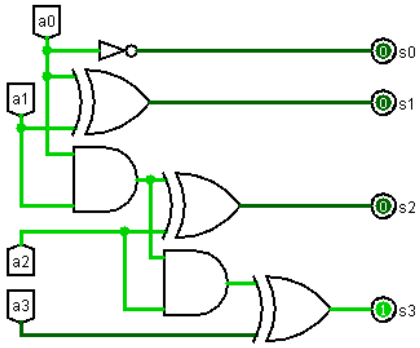
برای بیت دوم هنگام منهای یک، داریم:

$$s_1 = a_1 \oplus 1 \oplus a_0 = a_1 \odot a_0 \quad c_2 = a_1.1 + a_0.1 + a_1.a_0 = a_1 + a_0$$

بیت‌های بعدی نتیجه را هم به همین شکل ساده می‌کنیم و در نتیجه به مدارهای زیر می‌رسیم:



Decrementer



Incrementer

مدار سمت چپ را به شیوهٔ دیگری هم می‌توانیم بسازیم که چند گیت اضافه‌تر نیاز دارد اما در عوض ترکیب کردن دو مدار برای ساخت مدار نهایی ساده‌تر می‌کند. در این روش، می‌توانیم به جای جمع‌کننده از تفریق‌کننده استفاده کنیم. خروجی‌های این تفریق‌کننده را $d_0 \dots d_3$ و بیت‌های قرضی را $b_1 \dots b_3$ می‌نامیم.

برای بیت اول داریم:

$$d_0 = a_0 \oplus 1 = a'_0, \quad b_1 = a'_0 . 1 = a'_0$$

برای بیت دوم داریم:

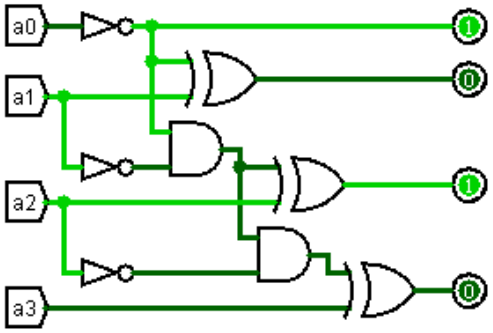
$$d_1 = a_1 \oplus 0 \oplus b_1 = a_1 \oplus a'_0 \quad b_2 = a'_1 . 0 + b_1 . 0 + a'_1 . b_1 = a'_1 . a'_0$$

برای بیت سوم داریم:

$$d_2 = a_2 \oplus 0 \oplus b_2 = a_2 \oplus b_2 \quad b_3 = a'_2 . 0 + b_2 . 0 + a'_2 . b_2 = a'_2 . b_2$$

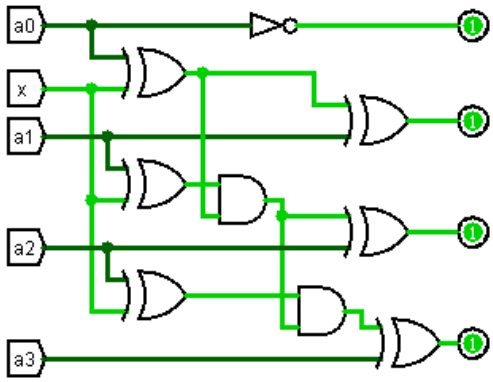
و بالاخره برای بیت چهارم:

$$d_3 = a_3 \oplus b_3$$



وقتی دو مدار را با هم ترکیب کنیم و از یک ورودی x برای انتخاب جمع یا تفریق استفاده کنیم، به این معنا است که می‌خواهیم بسته به مقدار x، عدد $a_3a_2a_1a_0$ را با یک جمع یا منهای یک کنیم. با توجه به این که نیازی به تولید بیت نقلی خروجی نداریم، این عمل را می‌توانیم طبق مدار زیر انجام دهیم.

در این مدار، اگر $x = 1$، مدار مثل یک تفریق‌کننده عمل می‌کند و اگر $x = 0$، مدار مثل جمع‌کننده عمل خواهد کرد.



۴- (۲۰ نمره) در یک پردازنده که با سرعت ۱٫۱ گیگاهرتز اجرا می‌شود، متوسط CPI بدون دسترسی به حافظه ۱٫۳۵ است. دسترسی به حافظهٔ داده فقط توسط دستورات load (۲۰٪ از تمام دستورات) و store (۱۰٪ از تمام دستورات) انجام می‌شود. سیستم حافظهٔ این کامپیوتر از دو حافظهٔ نهان L1 جداگانه برای داده و دستور تشکیل شده است که زمان دسترسی آن یک نانوثانیه است. ظرفیتِ هر دو حافظهٔ I-cache و D-cache ۳۲ کیلوبایت است و هر دو به صورت مستقیم نگاشت شده‌اند. نرخ فقدان در I-cache ۲٪ و بلوک‌های آن ۳۲ بایتی و سازوکار نوشتن آن write-back است. نرخ فقدان در D-cache ۵٪ و بلوک‌های آن ۱۶ بایتی است و سازوکار نوشتن آن write-through است. در D-cache یک بافر برای نوشتن وجود دارد که در ۹۵٪ موارد تأخیر نوشتن را حذف می‌کند. حافظهٔ نهان L2 یکپارچه است، ظرفیت آن ۵۱۲ کیلوبایت و دارای بلوک‌های ۶۴ بایتی و زمان دسترسی ۱۵ نانوثانیه است. این حافظه نهان L2 از طریق یک گذرگاه دادهٔ ۱۲۸ بیتی به حافظه نهان L1 متصل شده است که با سرعت ۲۶۶ مگاهرتز کار می‌کند و می‌تواند یک کلمه ۱۲۸ بیتی را در هر چرخه انتقال دهد. نرخ برخورد (hit rate) برای ارجاعات به حافظهٔ نهان L2 ۸۰٪ است. همچنین بیت dirty در ۵۰٪ از بلوک‌های جایگزین شده یک است. تأخیر دسترسی به حافظهٔ اصلی ۶۰ نانوثانیه است و پس از آن هر تعداد کلمهٔ ۱۲۸ بیتی می‌تواند با نرخ یک کلمه در هر چرخه از گذرگاه حافظه اصلی با نرخ ۱۳۳ مگاهرتز به حافظهٔ L2 منتقل شود.

الف- زمانِ متوسطِ دسترسی به دستورات چقدر است؟

ب- زمانِ متوسطِ خواندنِ داده‌ها چقدر است؟

ج- زمانِ متوسطِ نوشتنِ داده‌ها چقدر است؟

د- دسترسی به حافظه به طور متوسط چند چرخه طول می‌کشد؟

پاسخ:

گذرگاه L2 به L1 ۱۲۸ بیت معادل ۱۶ بایت است و نرخ آن ۲۶۶ مگاهرتز است. بنابراین انتقال هر ۱۶ بایت (برای D-cache) در این گذرگاه یک و انتقال هر ۳۲ بایت (برای I-cache) دو چرخهٔ $\frac{۱}{۲۶۶}$ میلی‌ثانیه‌ای طول می‌کشد.

$$L2ToIcache = L2Access + L2ToIcacheBus = 15 + 2 \times \frac{1000}{266} = 22.5\ ns$$

$$L2ToDcache = L2Access + L2ToDcacheBus = 15 + 1 \times \frac{1000}{266} = 18.8\ ns$$

وقتی در زمان خواندن از I-cache فقدان رخ می‌دهد، علاوه بر این که باید داده را از L2 بخوانیم، در ۵۰٪ از موارد باید دادهٔ جایگزین‌شده را در هم در L2 بنویسیم، چون سازوکار نوشتن در I-cache را write-back درنظر گرفته‌ایم، بنابراین:

$$L2ToIcacheMissPenalty = L2ToIcache + 0.5 \times L2ToIcache = 22.5 \times 1.5 = 33.75\ ns$$

گذرگاه حافظهٔ اصلی به L2 هم ۱۶ بایتی و نرخ آن ۱۳۳ مگاهرتز است، بنابراین انتقال هر ۶۴ بایت در این گذرگاه چهار چرخهٔ $\frac{۱}{۱۳۳}$ میلی‌ثانیه‌ای طول می‌کشد.

$$MemToL2 = MemAccess + MemToL2Bus = 60 + 4 \times \frac{1000}{133} = 90\ ns$$

حالا می‌توانیم به سوالات هر بند پاسخ دهیم:

الف-

$$access_{inst} = hitTime + missRateIcache \times (L2ToIcache + missRateL2 \times MemReadMissPenalty)$$
$$= 1 + 0.02 \times (33.75 + 0.20 \times 90) = 1 + 0.02 \times 51.75 = \mathbf{2.035\ ns}$$

ب-

$$access_{readdata} = hitTime + missRateDcache \times (L2ToDcache + missRateL2 \times MemReadMissPenalty)$$
$$= 1 + 0.05 \times (18.8 + 0.20 \times 90) = 1 + 0.05 \times 36.8 = \mathbf{2.84\ ns}$$

ج- وقتی روی حافظهٔ داده می‌نویسیم، چه داده در L1 باشد چه نباشد، یک بار هم باید آن را روی L2 بنویسیم. ولی چون بافر داریم، فقط در ۵٪ از موارد باید واقعا تاخیر ناشی از نوشتن روی L2 را تحمل کنیم. ضمنا در ۵٪ از موارد باید فقدان داده در L2 را هم بررسی کنیم.

$$access_{witedata} = hitTime + 0.05 \times L2ToDcache + 0.05 \times (L2ToDcache + missRateL2 \times MemMissPenalty)$$
$$= 1 + 0.05 \times 18.8 + 0.05 \times (18.8 + 0.20 \times 90) = 1 + 0.94 + 0.05 \times 36.8 = \mathbf{3.78\ ns}$$

د- دسترسی به حافظه برابر است با زمان واکشی دستورات (که برای همه دستورات لازم است)، زمان خواندن داده (که فقط برای دستورات load لازم است) و زمان نوشتن داده که فقط برای دستورات Store لازم است):

$$MemAccess = 2.035 + 0.20 \times 2.84 + 0.10 \times 3.78 = 2.981\ ns = \mathbf{3.28\ Clock\ Cycle}$$

اگر خواسته بودیم متوسط CPI را به‌دست بیاوریم، عدد بالا را با CPI پایه جمع می‌کردیم.

۵- (۲۰ نمره) فرض کنید درصد رخداد انواع پرش‌ها در برنامه‌هایی که روی یک پردازنده اجرا می‌شوند، طبق جدول زیر است:

پرش‌های غیرشرطی و فراخوانی زیرروال	٪۱
پرش‌های شرطی	٪۱۵
پرش‌های شرطی انجام‌شده	٪۶۰

الف- خط لولهٔ این پردازنده چهار مرحله دارد. محاسبهٔ مقصدِ پرش در مرحلهٔ دوم و محاسبهٔ شرطِ پرش در مرحلهٔ سوم انجام می‌شود، بنابراین پرش‌های شرطی در پایانِ مرحلهٔ دوم و پرش‌های غیرشرطی در پایانِ مرحلهٔ سوم کامل می‌شوند. اگر فرض کنید به طور خودکار و پیش از پردازشِ دستوراتِ پرش، دستورِ بلافاصله بعدی واردِ خط لوله می‌شود، متوسط CPI را حساب کنید.

پاسخ:

بنابر توضیح سوال، پرش‌های غیرشرطی و فراخوانی زیرروال به یک چرخه و پرش‌های شرطی انجام‌شده به دو چرخه تعلیق نیاز دارند و پرش‌های شرطی انجام‌نشده، بدون تعلیق اجرا می‌شوند.

$$CPI_{avg} = 1 + 0.01 \times 1 + 0.15 \times 0.6 \times 2 = 1.19$$

ب- این بار فرض کنید خط لولهٔ پردازنده ۱۵ مرحله دارد و پرش‌های غیرشرطی در پایانِ مرحلهٔ پنجم و پرش‌های شرطی در پایانِ مرحلهٔ دهم کامل می‌شود. متوسط CPI با این فرضِ جدید چقدر است؟ با مقایسهٔ پاسخ‌های دو بند الف و ب به چه نتیجه‌ای می‌رسید؟

پاسخ:

این بار، پرش‌های غیرشرطی و فراخوانی زیرروال به ۴ چرخه و پرش‌های شرطی انجام‌شده به ۹ چرخه تعلیق نیاز دارند.

$$CPI_{avg} = 1 + 0.01 \times 4 + 0.15 \times 0.6 \times 9 = 1.85$$

ج- فرض کنید خط لولهٔ این پردازنده این محدودیت را دارد که پس از وارد شدن یک دستورِ پرش در آن (چه شرطی و چه غیرشرطی)، فقط مرحلهٔ اولِ خطِ لوله برای اجرای دستورِ بعدی آزاد می‌شود. با این فرض، یک بار دیگر به سوالاتِ بندهای الف و ب پاسخ دهید.

پاسخ:

در این صورت، وقتی پرش شرطی انجام نمی‌شود، فقط یک دستور پس از آن می‌تواند وارد خط لوله شود، بنابراین فقط یکی از چرخه‌های تعلیق به طور مفید استفاده می‌شود.

4-stage pipeline: $CPI_{avg} = 1 + 0.01 \times 1 + 0.15 \times 0.6 \times 2 + 0.15 \times 0.4 \times 1 = 1.24$

15-stage pipeline: $CPI_{avg} = 1 + 0.01 \times 4 + 0.15 \times 0.6 \times 9 + 0.15 \times 0.4 \times 8 = 2.33$