به نام خدا

دانشکده مهندسی کامپیوتر

درس معماری کامپیوتر
نیم‌سال دوم ۰۲-۰۳
استاد: دکتر حسین اسدی

تمرین سری دهم

- پرسش‌های خود را در صفحه quera مربوط به تمرین مطرح نمایید.
- سوالات نظری را حتماً به صورت انفرادی و سوالات عملی را می‌توانید در گروه‌های دو نفر تحویل دهید.
- پاسخ‌ها را به صورت تایپی بنویسید.
- اسکرین‌شات‌ها، عکس‌ها و فایل‌های مربوط به سوال عملی را در فایل فشرده مربوطه در cw و quera قرار دهید. هر گونه عدم تطابق بین دو تمرین آپلود شده در دو سایت منجر به از دست رفتن نمره تمرین مربوطه می‌شود.
- پی دی اف قسمت تئوری را در سامانه cw و quera بارگذاری کنید.
- هر دانشجو می‌تواند حداکثر سه تمرین را با دو روز تأخیر بدون کاهش نمره ارسال نماید.

تمارین تئوری

۱. فرض کنید ماشینی با دو حالت آدرس‌دهی داریم که ویژگی‌های زیر را دارد:

۱. حافظه اصلی: با اندازه ۶۴ کلمه که هر کلمه آن ۸ بیت است.
۲. حافظه نهان: با اندازه ۱۶ کلمه که هر بلاک حافظه نهان ۸ کلمه دارد و سیاست آن نگاشت مستقیم[1] است.
۳. ۴ ثبات عمومی[2] با نام‌های R0, R1, R2, R3 دارد.
۴. ذخیره‌سازی اعداد به‌صورت مکمل ۲[3] است.
۵. پشتیبانی از حالات آدرس‌دهی مستقیم[4] و فوری[5] را دارد.

دو مدل دستور زیر را درنظر بگیرید:

Opcode	Address
2 bits	6 bits

Opcode	register	Opr2
3 bits	2 bits	3 bits

و فرض کنید دستورات ماشین ما به صورت زیر است:

Opcode	Mnemonic	Operation
01	jmp address	PC <- Address
10	jnz address	if (R0) != 0 then PC <- Address
11	jz address	if (R0) = 0 then PC <- Address
000	add r1,data	r1 <- (r1) + data; t =1
000	add r1,r2	r1 <- (r1) + (r2); t =0
001	sub r1,data	r1 <- (r1) - data; t =1
001	sub r1,r2	r1 <- (r1) - (r2); t = 0

در دستورات add ردیف چهارم و sub ردیف ششم، عملگر سوم ۳ بیتی است که اگر پرارزش‌ترین بیت آن (t) برابر ۰ باشد یعنی یک ثبات مدنظر است و اگر ۱ باشد یعنی یک داده فوری[6] است. حال با توجه به ماشین داده شده به سوالات زیر پاسخ دهید:

آ) مطابق ساختار حافظه نهان در اسلایدها، ساختار حافظه نهان این کامپیوتر را رسم کنید و مشخص کنید که هر قسمت چند بیت است.

[1] direct-mapped
[2] General Purpose
[3] 2s-complement
[4] Direct
[5] Immediate
[6] Immediate Data

ب) نرخ برخورد[7] برنامه زیر را حساب کنید (شروع برنامه از آدرس 14h حافظه اصلی است)
20h, 26h, 0, 0, 0, 5Eh, 2Eh, 28h, BFh, BFh, 25h, 99h, 0

۲. فرض کنید که پردازنده‌ای در اختیار داریم که چهار هسته دارد و از پروتکل MESI برای مدیریت حافظه‌ی نهان بین هسته‌ها استفاده می‌کند. هر هسته یک حافظه‌ی نهان ۲۵۶ بایتی با سیاست نگاشت مستقیم و سیاست write back با بلوک‌های 64 بایتی دارد. همچنین آدرس‌های حافظه‌ی معتبر در سیستم بین 0x10000000 و 0x1FFFFFFF است.

دستورات زیر در هسته‌های مختلف به ترتیب اجرا می‌شوند: (دستور ld برای بار کردن داده حافظه در ثبات فایل و دستور st برای ذخیره‌سازی داده یک ثبات در محلی از حافظه استفاده می‌شوند)

```
1        ld R1, 0x110000c0 \\ A memory instruction from core 1
2        st R2, 0x11000080 \\ A memory instruction from core 1
3        st R3, 0x1FFFFF40 \\ A memory instruction from core 0
4        ld R4, 0x1FFFFF00 \\ A memory instruction from core 1
5        st R5, 0x110000c0 \\ A memory instruction from core 1
```

بعد از اجرای دستورات بالا مقادیر کنترلی حافظه‌ی نهان به صورت زیر است:

Final State

Cache 0		
	Tag	MESI state
Set 0	0x1FFFFF	S
Set 1	0x1FFFFF	M
Set 2	0x110000	I
Set 3	0x110000	I

Cache 1		
	Tag	MESI state
Set 0	0x1FFFFF	S
Set 1	0x1FFFFF	I
Set 2	0x110000	M
Set 3	0x110000	E

Cache 2		
	Tag	MESI state
Set 0	0x10FFFF	I
Set 1	0x1FFFFF	I
Set 2	0x10FFFF	M
Set 3	0x10FFFF	M

Cache 3		
	Tag	MESI state
Set 0	0x133333	E
Set 1	0x000000	I
Set 2	0x000000	I
Set 3	0x10FFFF	I

حال با توجه به حالت‌های بالا، جدول زیر که نشان دهنده‌ی حالت اولیه مقادیر کنترلی حافظه‌ی نهان قبل از اجرای دستورات بالا است را تکمیل کنید. در صورتی که در ابتدا آدرس tag نامشخص بود، مقدار X را وارد کنید. برای حالت MESI نیز یکی از حروف M یا E یا S یا I را بنویسید.

Initial State

Cache 0		
	Tag	MESI state
Set 0		
Set 1		
Set 2		
Set 3		

Cache 1		
	Tag	MESI state
Set 0		
Set 1		
Set 2		
Set 3		

[7] hit rate

Cache 2		
	Tag	MESI state
Set 0		
Set 1		
Set 2		
Set 3		

Cache 3		
	Tag	MESI state
Set 0		
Set 1		
Set 2		
Set 3		

۳. در درس با مفهوم پیش واکشی[8] آشنا شدید. برای یافتن آدرس‌هایی که باید پیش واکشی شوند روش‌های گوناگونی وجود دارد که سه مدل پایه‌ای آن به شرح زیر هستند:

- Stride Prefetching
- Stream Prefetching
- Runahead Execution Prefetching

- در مورد این سه روش تحقیق کنید و توضیح دهید هر کدام چگونه تشخیص می‌دهند یک آدرس برای پیش‌واکشی مناسب است یا خیر.
- حال سه برنامه زیر را در نظر بگیرید و توضیح دهید کدام الگوریتم پیش واکشی برای آن‌ها مناسب‌تر از سایرین است:

(آ) اجرای یک الگوریتم گراف که نیازمند چند مرتبه پایش گراف هدف است.
(ب) ضرب داخلی دو ماتریس بزرگ که در حافظه قرار دارند.
(ج) پردازش پرسش و پاسخ‌های متوالی که به یک پایگاه داده ارسال می‌شوند.

۴. نتیجه‌ی اجرای قطعه کدهای پایتون زیر برای کسانی که با IEEE 754 آشنا نیستند غیرمنتظره خواهد بود:

آ) با استفاده از ماشین حساب‌های دارای قابلیت محاسبه ممیزشناور، نمایش ممیزشناور اعداد 0.1 و 0.2 را در 64 بیت به دست آورید و با جمع آن‌ها نشان دهید که این خطای محاسباتی چه طور به وجود آمده است.

```
print(0.1 + 0.2)                # prints 0.30000000000000004
print(0.1 + 0.2 == 0.3)         # prints False
```

ب) برای اعداد زیر نیز با نوشتن نمایش ۶۴ بیتی عددهای داده شده و انجام جمع نشان دهید که چرا نمی‌توان عدد 9007199254740993 را در استاندارد IEEE 754 نشان داد.

```
print(9007199254740992.0 + 1.0) # prints 9007199254740992.0
```

۵. فرض کنید که یک نوع نمایش عدد اعشاری وجود دارد که به صورت زیر تعریف می‌شود:

علامت	نما	مانتیس
۱ بیت	۵ بیت	۱۰ بیت

با این توصیف به سوالات زیر جواب دهید:

۱. مقدار بایاس را پیدا کنید.
۲. مقدار $+\infty$ را در این توصیف نمایش دهید.
۳. کوچکترین و بزرگترین عدد نرمال‌شده که می‌توان با این نمایش نشان داد، چه عددی است.
۴. کوچکترین و بزرگترین عدد نرمال‌نشده که می‌توان با این نمایش نشان داد، چه عددی است.

[8] Prefetching

۶. با توجه به اینکه در استاندارد IEE754 ، یک بیت برای علامت ، ۱۱ بیت برای نما و ۵۲ بیت برای بخش کسری یک عدد ۶۴ بیتی ممیزشناور اختصاص داده شده است،

آ) بزرگ ترین عدد مثبت نرمال و غیر نرمال که قابل نمایش با این فرمت است، چیست؟

ب) کوچکترین عدد مثبت نرمال و غیر نرمال که قابل نمایش با این فرمت است، چیست؟

تمارین عملی

۱. **تمرین عملی اول**

در این تمرین قصد داریم یک برنامه محک[9] را با سیاست‌های مختلف را روی ابزار Gem5 شبیه‌سازی کنیم و نتایج بدست آمده از شبیه‌سازی را تحلیل کنیم.

برای شبیه‌سازی از دو سیاست FIFO و LRU استفاده کنید. برای پیکربندی سیستم شبیه‌سازی نیز از پیکربندی پیش‌فرض در آدرس زیر استفاده کنید و به صورت زیر با قابلیت‌های زیر اجرا کنید.

```
./build/X86/gem5.opt configs/deprecated/example/se.py -c [bench] --
   caches --l2cache --l2_size=4kB --mem-type=DDR4_2400_16x4 --
   cacheline_size 128
```

پس از شبیه‌سازی به واسطه راهنمایی داده شده، دنباله[10] دسترسی حافظه توسط شبیه‌ساز را بدست آورید.

حال با توجه به این دنباله دسترسی به حافظه، به واسطه یک اسکریپت پایتون، بدست آورید که در صورتی که سیاست حافظه نهان به صورت بهینه عمل می‌کرد، مقدار نرخ برخورد به چه صورت تغییر می‌کرد.

نکته: فرض کنید ساختار حافظه نهان به صورت 2-way set-associative است.

راهنمایی(اضافه کردن قابلیت سیاست حافظه نهان)

برای اضافه کردن قابلیت سیاست جایگزینی[11] هنگام اجرای شبیه‌سازی مراحل زیر را اجرا کنید. ابتدا در فایل موجود در مسیر configs/common/ObjectList خط زیر را اضافه کنید:

```
repl_list = ObjectList(getattr(m5.objects, 'BaseReplacementPolicy',
   None))
```

سپس قابلیت‌های زیر را به عنوان قابلیت‌های شبیه‌سازی، در فایل configs/common/Options.py به تابع addNoISAOptions اضافه کنید:

```
parser.add_argument("--l2_repl", default="LRURP",
choices=ObjectList.repl_list.get_names(),
help = "replacement policy for l2")
```

نهایتا سه خط زیر را به انتهای تابع __get__cache__opts در فایل موجود در مسیر configs/common/CacheConfig.py اضافه کنید:

```
replacement_policy_attr = f"{level}_repl"
if hasattr(options, replacement_policy_attr):
opts["replacement_policy"] = ObjectList.repl_list.get(getattr(options,
   replacement_policy_attr))()
```

حال می‌توانید با استفاده از قابلیت l2__repl– نوع سیاست جایگزینی را هنگام شبیه‌سازی برای حافظه نهان لایه دوم تعیین کنید.

راهنمایی(بدست آوردن ترتیب دسترسی به حافظه)

برای بدست آوردن دنباله دسترسی به حافظه، باید فایل configs/common/CacheConfig.py را تغییر دهیم. در تابع config__cache و در زیر شرط if options.l2cache ابتدا دو خط زیر را کامنت کنید:

```
system.l2.cpu_side = system.tol2bus.mem_side_ports
system.l2.mem_side = system.membus.cpu_side_ports
```

[9] Benchmark
[10] sequence
[11] Replacement Policy

سپس درست زیر این دو خط کامنت شده، قطعه کد زیر را اضافه کنید:

```
        system.monitor2 = CommMonitor()
        system.monitor2.trace = MemTraceProbe(trace_file = "CT_mon2.trc.gz")
        system.monitor2.slave = system.l2.mem_side

        system.membus.slave = system.monitor2.master
        system.l2.cpu_side = system.tol2bus.master
```

با اضافه کردن این قطعه کد شما پکت‌هایی را که روی حافظه‌نهان لایه دوم جابه‌جا می‌شوند، مانیتور می‌کنید. با اجرای شبیه‌سازی، یک فایل فشرده در آدرس m5out/CT_mon2.trc.gz ایجاد می‌شود که دارای یک فایل، حاوی اطلاعات مانیتور شده روی حافظه‌نهان لایه دوم است. شما به واسطه اسکریپت موجود در مخزن gem5 می‌توانید اطلاعات موجود در فایل مدنظر را decode کنید.

```
        python3 util/decode_packet_trace.py CT_mon2.trc result.csv
```

فایل خروجی تولید شده فرمتی به صورت زیر دارد:

```
        5,u,217728,128,256,0
        6,u,331264,128,98,1500
        5,u,221184,128,256,3000
        6,u,331136,128,98,11000
        6,u,323072,128,131171,15500
        6,u,322176,128,99,18500
        6,u,318080,128,99,22500
        6,u,319104,128,99,23500
        5,u,221312,128,256,24500
        6,u,322304,128,99,25000
```

در هر خط یک Cache Access نمایش داده شده است که مقدار سوم آدرس خانه مدنظر از حافظه را نمایش می‌دهد. شما به واسطه این داده‌ها می‌توانید نرخ برخورد حالت بهینه را بدست آورید.

۲. تمرین عملی دوم

یکی از مباحث تحقیقاتی در آزمایشگاه Data Storage, Networks, and Processing (DSN)، تحلیل رفتار کاربردها به منظور بهینه‌سازی معماری‌های مرتبط با حافظه‌های نهان[۱۲] است. در این تمرین قصد داریم با قسمت اول این شاخه تحقیقاتی آشنا شویم. همان‌طور که در مباحث درس نیز اشاره شد، تمامی سیاست‌های معماری حافظه نهان برای تمامی کاربردها مناسب نیستند. در صورتی که این سیاست‌ها برای کاربردها درست انتخاب نشود، نه تنها بهبودی در کارایی مشاهده نمی‌شود بلکه می‌تواند منجر به کاهش کارایی نیز گردد. بدین منظور باید این کاربردها شناسایی شوند و سیاست‌های مناسب برای حافظه‌های نهان انتخاب گردد. در این تمرین می‌خواهیم مفاهیم مورد بحث در تمرین سری هشتم و نهم را تکمیل نماییم. در دو تمرین سری قبل یعنی تمرینات سری هشتم و نهم با مفاهیم محلیت زمانی[۱۳] و محلیت فضایی[۱۴] آشنا شدید. در این تمرین قصد داریم نتایجی که قبلاً گرفته بودید را به ازای درخواست‌های نوشتن و خواندن جدا کنیم و تحلیل نماییم.

قسمت اول، محلیت زمانی: در این تمرین باید یک کد به زبان C++ یا Python بنویسید که محلیت زمانی چند فایل ردگیری[۱۵] از فضای ابری Alibaba که در صفحه درس در CW قرار داده شده است را به ازای درخواست‌های نوشتن و خواندن به طور جداگانه محاسبه کند.

[12] Cache
[13] Temporal locality
[14] Special Locality
[15] Trace

قسمت دوم، محلیت فضایی: در این تمرین باید یک کد به زبان C++ یا Python بنویسید که محلیت فضایی چند فایل ردگیری از فضای ابری Alibaba که در صفحه درس در CW قرار داده شده است را به ازای درخواست‌های نوشتن و خواندن به طور جداگانه محاسبه کند.

راهنمایی

۱. برای راحتی بهتر است یکبار فایل‌های ردگیری را به ازای درخواست‌های نوشتن و سپس به ازای درخواست‌های خواندن بررسی کنید.

۲. برای راهنمایی بیشتر به صورت تمرین‌های سری هشتم و نهم مراجعه کنید.

فرمت فایل Alibaba

فرمت هر خط فایل‌های ردگیری Alibaba به صورت زیر است:

Time Stamp(ns), Response Time(ns), Offset(Byte), Request Size (Byte),
Request Type(Read/Write), Process ID, Major Disk Number, Minor Disk Number

گزارش

۱. محلیت زمانی: در گزارش خود باید نمودار مربوط به محلیت زمانی این چهار فایل ردگیری را برای پنجره‌های ۱ دقیقه، ۱ ساعت، ۱۲ ساعت و ۲۴ ساعت یکبار به ازای درخواست‌های نوشتن و یکبار به ازای درخواست‌های خواندن رسم کنید.
سپس نمودارهای این تمرین را با تمرین سری هشتم مقایسه کنید.

۲. محلیت فضایی: در این گزارش باید ۸ نمودار به ازای درخواست‌های نوشتن و ۸ نمودار به ازای درخواست‌های خواندن برای چهار بارکاری Alibaba رسم کنید، یکبار با طول صف ۶۴ و بار دیگر با طول صف ۱۲۸ این نمودارها را رسم کنید. هریک از این نمودارها باید شامل ۴ میله باشد که درصد ۴ کلاس مختلف را نشان دهد.
سپس نمودارهای این تمرین را با تمرین سری نهم مقایسه کنید.

۳. کدهای خود را نیز باید به همراه گزارش بارگذاری کنید.

مطالعه بیشتر

در صورتی که علاقمند به مطالعه بیشتر در این شاخه پژوهشی هستید می‌توانید مقالات زیر را مطالعه نمایید:

Ebrahimi, Shahriar, Reza Salkhordeh, Seyed Ali Osia, Ali Taheri, Hamid R. Rabiee, and Hossen Asadi. "Rc-rnn: Reconfigurable cache architecture for storage systems using recurrent neural networks." IEEE Transactions on Emerging Topics in Computing 10, no. 3 (2021): 1492-1506.