به نام خدا

دانشکده مهندسی کامپیوتر

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

پروژه

- پروژه در گروه‌های **چهار نفری** انجام می‌شود. نحوه گروه‌بندی در CW اطلاع‌رسانی می‌شود.
- همه موارد قابل تحویل برای پروژه را در یک فایل Zip با نام Arch-Project-STDID1-STDID2-STDID3-STDID4.zip جمع‌آوری نموده و در سامانه CW و Quera بارگذاری نمایید (از هر گروه تنها یک نفر پروژه را بارگذاری نماید).
- هر کدام از پروژه‌ها را تنها ۳ گروه می‌توانند انتخاب کنند و اولویت با گروه‌هایی خواهد بود که پروژه مورد نظر را زودتر انتخاب نماید.
- در صورت هرگونه سوال یا اشکال، آن را در تالار مربوط به پروژه موردنظر در صفحه درس در CW یا Quera مطرح نمایید.
- توصیه می‌شود شروع پروژه را به روزهای آخر موکول نفرمایید و در اسرع وقت کارهای اولیه پروژه را شروع نمایید.
- هر گروه باید حداکثر تا تاریخ **چهار خرداد** تیم پروژه و توصیف مختصر پروژه را در قالب یک فایل pdf تک صفحه‌ای در صفحه درس بارگذاری نماید.
- موعد انجام پروژه روز **پنجم تیر** خواهد بود.
- پروژه‌ها به صورت حضوری به دستیاران آموزشی تحویل داده می‌شود. همه اعضای گروه باید برای این منظور حضور یافته و به همه‌ی قسمت‌های پروژه تسلط داشته باشند.
- گزارش پروژه باید در فرمت لاتک و در سامانه لاتک آنلاین دانشگاه نوشته شود. لذا یکی از نفرات پروژه باید قالب گزارش سمینار را از لینک ذیل انتخاب کرده و در این قالب گزارش تیم پروژه را ایجاد نماید. پروژه باید بین اعضای گروه به اشتراک گذاشته شود و تمامی اعضای گروه باید در نوشتار مشارکت نمایند. دقت شود تاریخچه مشارکت اعضای گروه، توسط دستیار آموزشی در سامانه قابل رویت خواهد بود.
  آدرس ورود به سامانه لاتک دانشگاه: Login Latex
  قالب گزارش پروژه: Template Latex
- گزارش پروژه یک باید با آقای مرادی در سامانه به اشتراک گذاشته شود.
- گزارش پروژه‌های دو، سه و چهار باید با آقایان کشوری، بهنام و مرادی در سامانه به اشتراک گذاشته شود.
- گزارش پروژه پنج و شش باید با آقایان معینی، قاسمی و مرادی در سامانه به اشتراک گذاشته شود.
- ایمیل هاTA برای اشتراک گذاری پروژه‌ها:

moradi: am.moradi@sharif.edu
behnam: hirbod.behnam@sharif.edu
keshvari: hooman.keshvari@sharif.edu
moeini: khosro.moeini81@sharif.edu
ghasemi: mohsen.ghasemi@sharif.edu

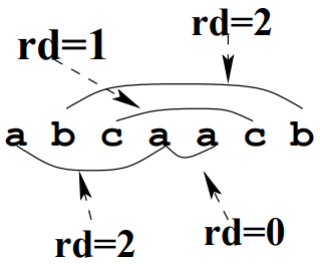
۱. پروژه اول: محاسبه فاصله استفاده مجدد

صورت پروژه

یکی از مباحث تحقیقاتی در آزمایشگاه Data Storage, Networks, and Processing (DSN)، تحلیل رفتار کاربردها به منظور بهینه‌سازی معماری‌های مرتبط با حافظه‌های نهان[1] 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 قصد داریم با یک مورد از این مؤلفه‌ها یعنی فاصله استفاده مجدد[2] بیشتر آشنا شویم.

فاصله استفاده مجدد

فاصله استفاده مجدد یک روش کلاسیک برای مشخص کردن محلیت داده است. فاصله استفاده مجدد برای یک دسترسی مثل آدرس A برابر با تعداد دسترسی‌های متمایز بین دو بار دسترسی به آدرس A تعریف می‌شود. برای روشن شدن این مطلب به مثالی که در شکل زیرآمده است، توجه کنید. در این مثال یک دنباله از دسترسی‌ها آورده شده است. که شامل دسترسی‌ها به آدرس a و b و c است. مثلاً اگر بخواهیم به دسترسی‌های آدرس a بپردازیم. در این دنباله سه بار آدرس a مورد دسترسی قرار گرفته است. بار اول بین دو دسترسی به آدرس a، آدرس‌های b و c قرار گرفته‌اند. در این حالت rd برابر ۲ اندازه‌گیری می‌شود. اما در حالت بعدی بین دو دسترسی به a هیچ آدرس دیگری مورد دسترسی قرار نگرفته است. از این رو rd برابر ۰ اندازه‌گیری می‌شود. پس میانگین rd برای آدرس a برابر ۱ خواهد بود. یک مثال دیگر دسترسی به آدرس b است. در این حالت دو آدرس a و c هرکدام بین دو بار دسترسی به آدرس b، دو بار مورد دسترسی قرار گرفته‌اند. از آنجایی که در تعریف rd آدرس‌های متمایز حساب می‌شوند، پس rd برابر ۲ اندازه‌گیری می‌شود.



گام‌های پروژه

۱. یکی از چالش‌های پیاده‌سازی فاصله استفاده مجدد، زمان اجرا و حجم حافظه مورد استفاده آن است. بنابراین باید این کد به صورت بهینه نوشته شود. باید زمان اجرا آن $O(nlogn)$ و حافظه آن نیز $O(n)$ باشد. برای مطالعه در رابطه با نحوه پیاده‌سازی به مقاله ۳ صفحه‌ای زیر مراجعه کنید:

Data Structures and Algorithms for Calculating Reuse Distance

۲. در این تمرین باید یک کد به زبان C++ یا Python بنویسید که مقدار میانگین فاصله استفاده مجدد را محاسبه کند.

۳. کد نوشته شده را روی چهار فایل ردگیری[3] از فضای ابری Alibaba که در صفحه درس در CW قرار داده شده است اجرا کنید و فاصله استفاده مجدد را برای آن‌ها محاسبه کنید.

راهنمایی

[1] Cache
[2] Reuse Distance
[3] Trace

۱. مقاله معرفی شده در بالا را حتماً مطالعه کنید.
۲. کد قرار داده شده در CW که به زبان C++ و زمان اجرای آن $O(n^2)$ و حافظه آن $O(n)$ است را بررسی نمایید.
۳. **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 و ارزیابی

۱. در گزارش خود باید مقدار میانگین فاصله استفاده مجدد را برای ۴ بارکاری Alibaba محاسبه کنید و تحلیل کنید.
۲. سپس تحلیلی بر تأثیر مقدار میانگین فاصله استفاده مجدد بر دو مؤلفه محلیت زمانی[۴] و فضایی[۵] ارائه دهید.

قسمت امتیازی

یکی از دو مورد زیر را برای گرفتن نمره امتیاز می‌توانید انتخاب کنید.

۱. کد قرار داده شده در CW که به زبان C++ است، علاوه بر میانگین فاصله مجدد مواردی همچون مقدار بیشنه یا کمینه و ... را محاسبه می‌کند. می‌توانید به عنوان نمره امتیازی محاسبه این نتایج را به کد خود اضافه کنید.
۲. می‌توانید کد Distance Reuse Useful که در مقاله ECI-Cache آمده را پیاده‌سازی کنید.

مطالعه بیشتر

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

Ahmadian, Saba, Onur Mutlu, and Hossein Asadi. "ECI-Cache: A high-endurance and cost-efficient I/O caching scheme for virtualized platforms." Proceedings of the ACM on Measurement and Analysis of Computing Systems 2, no. 1 (2018): 1-34.

[4] Temporal Locality
[5] Spatial Locality

۲. پروژه دوم: پیاده‌سازی حافظه نهان در پردازنده MIPS

صورت پروژه

در این پروژه قصد داریم به پردازنده‌ای که در طول ترم آن را کامل‌کرده‌اید، یک حافظه نهان اضافه کنیم. هدف نهایی ما از این پروژه تسریع برنامه‌ها در پردازنده است که این تسریع به صورت شفاف قابل بررسی باشد. حافظه اصلی موجود در پروژه‌ای که در طول ترم ساخته‌اید در یک چرخه پاسخ می‌دهد در صورتی که در عمل این چنین نیست و دسترسی‌ها به حافظه‌ی اصلی با تأخیر زیادی همراه هستند.

گام‌های پروژه

پروژه مربوطه را باید در چند قسمت پیاده‌سازی کنید:

۱. ابتدا باید مکانیزم تأخیر را در حافظه خود شبیه‌سازی کنید به صورتی که برای حاضر کردن جواب به چندین چرخه زمان نیاز داشته باشد و از حافظه دارای تأخیر استفاده کنید.

۲. در قسمت بعد باید پردازنده خود را به گونه‌ای تغییر دهید که بتواند در دستورات مربوط به حافظه چند چرخه متوقف شود تا عملیات به درستی انجام شود.

۳. سپس باید یک حافظه‌ی نهان سطح ۱ پیاده‌سازی کنید و توقف‌های به وجود آمده هنگام دسترسی به حافظه اصلی را به حداقل برسانید. ویژگی‌های این حافظه به این صورت است که:

(آ) از نوع write back است.

(ب) در مورد ظرفیت و اندازه بلوک‌ها آزاد هستید اما باید حداقل با ۳ اندازه مختلف این حافظه را بسازید و با ذکر علت بگویید کدامیک بهتر عمل می‌کند. لذا پیشنهاد می‌کنیم حافظه خود را به صورت پارامتری طراحی کنید.

(ج) مدل نگاشت حافظه نهان به صورت نگاشت مستقیم[۶] است.

گزارش و ارزیابی

۱. در گزارش خود باید هر مرحله و نحوه پیاده‌سازی خود را ذکر کنید.

۲. باید نحوه پیاده‌سازی هر قسمت را به همراه مستندات مربوط به درستی عملکرد حافظه نهان را ذکر کنید.

۳. همچنین باید حداقل ۳ برنامه با تعداد عملیات حافظه اصلی بالا طراحی کنید و با ۳ حافظه نهان طراحی شده مقایسه کنید. بدین منظور باید به ازای هر برنامه یک نمودار ارائه دهید که هر کدام شامل ۴ حالت یعنی یک حالت بدون حافظه نهان و ۳ حالت دارای حافظه نهان باشد. سپس این نمودارها را تحلیل و در گزارش نهایی خود بگذارید.

قسمت امتیازی

در این قسمت باید علاوه بر حافظه نهان سطح ۱ ، یک حافظه نهان سطح ۲ نیز طراحی کنید که اندازه این حافظه نهان سطح ۲ چهار برابر حافظه نهان سطح یک باشد. همچنین باید دو حافظه از سیاست نگاشت set-associative پیروی کنند.

[6] Direct Mapped

۳. پروژه سوم: پیاده‌سازی واحد محاسباتی اعداد ممیزشناور مبتنی بر استاندارد IEEE 754

صورت پروژه

در این پروژه قصد داریم به پردازنده‌ی MIPS که در تمرین‌های عملی درس طراحی کردید واحد FPU را اضافه کنیم تا بتواند عملیات ریاضی روی اعداد اعشاری را انجام دهد. در ابتدا کدگزاری دستورات جدید در زیر آورده شده است. دقت کنید که دستورات قبلی شما باید دست نخورده باقی بمانند. همان‌طور که از پردازنده‌ی MIPS به خاطر دارید یک سری دستورات R-Type داشتیم که با سه ثبات کار می‌کردند و به صورت زیر بودند:

opcode	rs	rt	rd	funct
4 bits	3 bits	3 bits	3 bits	3 bits

اگر به خاطر داشته باشید همیشه در این دستورات opcode برابر 0000 بود و funct مشخص می‌کرد که چه دستوری باید انجام شود. در این پروژه با سری دستورات جدیدی رو به رو هستیم که دقیقا همان فرمت R-Type را دارند با این تفاوت که مقدار opcode برابر 0001 است. جدول عملیاتی که باید FPU شما قادر به انجام آن باشد با توجه به مقدار funct آن در زیر آمده است.

Mnemonic	Operation	funct
FADD	rd ← rs + rt	000
FSUB	rd ← rs - rt	001
FMULT	rd ← rs * rt	010
FABS	rd ← |rs|	011
FSLT	rd ← 1 `if` rs < rt `else` 0	100

دستور FSLT یا Float Set Less Than دو ثبات rs و rt را با هم مقایسه می‌کند و در صورتی که rs بزرگتر از rt بود، مقدار ۱ را در ثبات rd قرار می‌دهد. در غیر این صورت مقدار ۰ در این ثبات قرار می‌گیرد.

همان طور که به یاد دارید ماشین ما ۸ بیتی است. پس باید اعداد اعشاری ما نیز ۸ بیتی باشند. فرمت اعداد اعشاری که ماشین ما می‌شناسد به صورت زیر است:

sign bit	exponent	significand
1 bit	4 bits	3 bits

این فرمت به Minifloat معروف است. دقت کنید که مانند خود float و double توان عدد ما به صورت 2^{x-7} است که x همان مقدار exponent است.

گام‌های پروژه

در این پروژه لازم است گام‌های زیر را انجام دهید:

۱. در Quartus و به صورت شماتیک یک FPU طراحی کنید که عملیات خواسته شده را انجام دهد.

۲. در ادامه باید control unit خود را طوری عوض کنید که بتواند دستورات جدید را انجام دهد.

۳. بدون استفاده از دستورات جدید برنامه‌ای با اسمبلی MIPS خودتان بنویسید که بتواند بدون استفاده از دستورات FPU دو عدد اعشاری ۸ بیتی را با هم جمع کند.

۴. دقت کنید که در این پروژه باید از اعداد خاص مانند NaN یا Inf پشتیبانی شود.

نحوه آزمون

۱. در نهایت بعد از زدن پروژه باید شما مدار خود را تست کنید.

۲. برای این کار برنامه‌ای بنویسید که بتواند به صورت تقریبی رادیکال یک عدد که در خانه‌ای در حافظه ذخیره شده است را پیدا کند.

۳. برای این کار کافی است که برنامه‌ای بنویسید که از ۰ شروع کند و عدد را به توان دو برساند. سپس با اضافه کردن 0.25 در هر مرحله به آن چک کند که آیا همچنان خطا کمتر می‌شود یا خیر و در صورتی که خطا به جای کمتر شدن بیشتر شد، عدد را به عنوان رادیکال عدد داده شده نمایش دهد.

گزارش و ارزیابی

۱. در نهایت تمامی کارهایی که کردید و کدهایی که نوشته‌اید را در گزارش خود نشان دهید.

۲. همچنین حتما از مدار و عملکرد آن اسکرین‌شات تهیه کنید.

۳. در نهایت نیز بررسی کنید که کدی که بدون دستورات FADD برای جمع زده بودید چه قدر کندتر از زمانی است که از دستورات FADD استفاده می‌کنید. برای این کار صرفا چک کردن تعداد چرخه‌ها کافی است.

قسمت امتیازی

شما باید تابع Fast Inverse Square Root را در پردازنده‌ی خود به صورت نرم‌افزاری پیاده‌سازی کنید. این تابع به صورت تقریبی مقدار $\frac{1}{\sqrt{x}}$ را حساب و خروجی را تولید می‌کند. برای پیاده‌سازی این تابع در ابتدا باید مقدار ثابتی که برای اعداد ممیزشناور ۳۲ بیتی مبتنی بر استاندارد IEEE 754 برابر 0x5f3759df است را برای اعداد ۸ بیتی پیدا کنید و سپس الگوریتم را پیاده‌سازی کنید. برای آشنایی با این الگوریتم در ابتدا می‌توانید به صفحه‌ی ویکیپدیا مراجعه کنید تا با کلیت الگوریتم و چگونگی آن آشنا شوید. در ادامه می‌توانید به این فیلم مراجعه کنید که در آن توضیح داده می‌شود عدد 0x5f3759df از کجا آمده است.

در نهایت بعد از نوشتن تابع آن را بررسی کنید که چه قدر خطا نسبت به جواب واقعی $\frac{1}{\sqrt{x}}$ دارد.

۴. پروژه چهارم: پیاده‌سازی branch predictor

صورت پروژه

در این پروژه قصد داریم به پردازنده‌ی MIPS که در تمرین‌های عملی درس طراحی شده بود، واحد branch prediction اضافه کنیم. همان طور که در درس نیز یاد گرفته بودید branch predictorها با توجه به کدی که در قبل اجرا شده است پیش‌بینی می‌کنند که دستورهایی را در خطلوله[۷] بیاورند که با فرض taken بودن آن اجرا می‌شوند یا اینکه دستورهایی را در خطلوله بیاورند که با فرض not taken بودن اجرا می‌شوند.

گام‌های پروژه

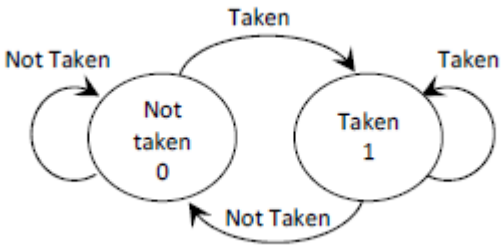
تغییر بخش خطلوله

اولین کاری که باید انجام دهید این است که به خطلوله‌ای که در گذشته در تمرین‌ها طراحی کرده بودید، قابلیت flush کردن را اضافه کنید. این بدین معنی است که خطلوله‌ی شما باید قابلیت این را داشته باشد که در صورتی که حدس زد یک پرش از نوع taken است ولی در واقعیت not taken بود بتواند دستوراتی که در خطلوله قرار دارند و به صورت اشتباه تا وسط اجرا شده‌اند را پاک کند.

دقت کنید که شما نیازی به ساختن واحد forwarding unit ندارید. کافی است که به صورت نرم‌افزاری و اضافه کردن NOP به کد خود مشکلات data dependency را برطرف کنید.

واحد branch prediction

در ادامه شما باید واحد branch prediction را طراحی کنید که بتواند taken یا not taken بودن پرش‌ها را پیش‌بینی کند. برای طراحی این branch predictor باید از یک بیت استفاده کنید که نشان می‌دهد که آیا در آخرین باری که پرش مذکور اتفاق افتاده است، taken بوده است یا خیر. همچنین این واحد باید ۴ مدخل داشته باشد که مربوط به آدرس دستورات پرش هستند. برای یادآوری یک branch predictor تک بیتی به صورت زیر کار می‌کند:



در ابتدا که پردازنده شروع به کار می‌کند مدخل‌های این branch predictor در حالت invalid قرار دارند (مثل حافظه‌ی نهان) و با اجرا شدن پرش‌ها پر می‌شوند. حال نکته‌ای که در این پروژه حائز اهمیت است سیاست جایگزاری مدخل‌ها است. شما در این پروژه باید دو نوع سیاست جایگزاری FIFO و LRU را برای branch predictor خود پیاده‌سازی کنید.

حالت پیش‌فرض برای پرش‌های جدید را taken در نظر بگیرید.

گزارش و ارزیابی

۱. زمانی که پیاده‌سازی شما تمام شد، در ابتدا برنامه‌ای بنویسید که دو حلقه‌ی تو در تو داشته باشد و آن را با سه حالت بدون branch predictor، با branch prediction همراه با سیاست جایگزاری FIFO و همچنین با branch prediction همراه با سیاست جایگزاری LRU اجرا کند. سپس تعداد چرخه‌هایی که طول می‌کشد که برنامه و حلقه‌ها انجام شوند را در هر سه حالت مقایسه و تحلیل کنید.

[7] pipeline

۲. در ادامه برنامه‌ای بنویسید که branch predictor شما در یک حلقه یک شرط را به صورت دائم اشتباه حدس بزند. سپس آن را با سه حالت بدون branch predictor، با branch prediction همراه با سیاست جایگزاری FIFO و همچنین با branch prediction همراه با سیاست جایگزاری LRU اجرا کند. سپس تعداد چرخه‌هایی که طول می‌کشد که برنامه انجام شود را در هر سه حالت مقایسه و تحلیل کنید.

۳. تمامی کدها و مشاهده‌ها و تحلیل‌های خود را به صورت گزارش در بیاورید. حتما در گزارشی که در سامانه آپلود می‌کنید عکس‌هایی از کارکرد مدار و شکل موج‌ها[8] قرار دهید.

قسمت امتیازی

برای گرفتن نمره‌ی امتیازی این پروژه کافی است که دو کار را انجام دهید. در ابتدا شما باید به جای یک بیت از دو بیت در واحد branch prediction استفاده کنید. در ادامه شما باید الگوریتم جایگزاری Pseudo-LRU را برای مدخل‌ها به کار گیرید. برای مطالعه درباره این الگوریتم می‌توانید از این لینک استفاده کنید.

[8] waveform

۵. پروژه پنجم: پیاده‌سازی perceptron branch predictor در Gem5

صورت پروژه

در این پروژه قصد داریم مکانیزم معروف perceptron branch predictor را به Gem5 اضافه کنیم و عملکرد آن را با سایر branch predictorهای Gem5 مقایسه کنیم.

این روش در سال ۲۰۰۱ توسط Daniel A. Jiménez مطرح شد. می‌توانید مقاله مربوط به این روش را از این لینک دانلود کنید. این روش در ادامه به طور مختصر شرح داده شده است اما پیشنهاد می‌کنیم قبل از شروع پروژه مقاله را مطالعه کنید.

توضیح مختصر روش مقاله

global branch history

global branch history یک عدد n بیتی است که بیت iام نشان می‌دهد نتیجه iامین پرش قبلی چه بوده است. اگر مقدار آن ۱ باشد یعنی taken و اگر ۰ باشد یعنی not taken بوده است.

perceptron

هر perceptron یک بردار از وزن‌های $w_0 ... w_n$ است که در آن n تعداد بیت global branch history است. w_0 مربوط به bias است و سایر w_iها متناظر بیت iام global branch history هستند.

پیش‌بینی

برای پیش‌بینی یک پرش با استفاده از perceptron ابتدا در نظر می‌گیریم $y = w_0$ است و در هر مرحله در صورتی که بیت iام branch history ۱ باشد مقدار w_i را به آن اضافه می‌کنیم و در غیر این صورت کم می‌کنیم. در نهایت اگر مقدار y بزرگتر از ۰ باشد taken و در غیر این صورت not taken پیش‌بینی می‌شود.

آموزش

در صورتی که بعد از مشخص شدن نتیجه واقعی پرش مقدار پیش‌بینی شده اشتباه بوده باشد باید وزن‌ها را بروزرسانی کنیم. مقدار واقعی را با t نشان می‌دهیم که اگر ۱ باشد یعنی taken و اگر ۱- باشد یعنی not taken بوده است. برای بروزرسانی وزن‌ها و train کردن یا همان آموزش perceptron در صورتی که بیت iام global branch history ۱ باشد مقدار t را به w_i اضافه می‌کنیم و در غیر این صورت از آن کم می‌کنیم. برای w_0 همیشه مقدار t را به آن اضافه می‌کنیم.

perceptron branch predictor

در ادامه نحوه کار perceptron branch predictor را توضیح می‌دهیم.

این مکانیزم از یک جدول با اندازه N استفاده می‌کند که هر سطر آن مربوط به یک perceptron است. زمانی که یک دستور پرش توسط پردازنده واکشی[9] می‌شود، برای پیش‌بینی مراحل زیر طی می‌شوند.

۱. آدرس پرش هش می‌شود تا به سطر iام از جدول یعنی یکی از اعداد 0 تا $N-1$ نگاشت شود.
۲. perceptron مربوط به سطر iام واکشی می‌شود.
۳. پرش با روشی که پیش‌تر توضیح دادیم، پیش‌بینی می‌شود.
۴. زمانی که نتیجه واقعی پرش مشخص شود perceptron با روشی که پیش‌تر توضیح دادیم train می‌شود.
۵. وزن‌های جدید perceptron در سطر iام جدول نوشته می‌شوند.

[9] Fetch

گام‌های پروژه

۱. کلاس Perceptron را تعریف کنید و عملیات‌های پیش‌بینی و آموزش را پیاده‌سازی کنید.

۲. کلاس PerceptronBP را با ارث‌بری از کلاس BPredUnit تعریف کنید.

راهنمایی : branch predictor ها در Gem5 در مسیر "./src/cpu/pred/" تعریف شده‌اند. شما نیز پیاده‌سازی خود را در این مسیر اضافه کنید. برای راهنمایی می‌توانید کدهای بقیه branch predictor ها را مطالعه کنید.

۳. با شبیه‌سازی با برنامه‌های محک[10] که پیش‌تر در تمرین از آن‌ها استفاده کردید درصد پیش‌بینی درست پیاده‌سازی خود را به ازای حالت‌های زیر بدست آورید.

History Length (n)	Table Rows (N)
15	8
31	8
31	16

۴. به ازای محک‌ها و مقادیر n قسمت قبل درصد پیش‌بینی صحیح را برای روش bi-mode که در Gem5 تعریف شده است بدست آورید.

گزارش و ارزیابی

۱. در یک مقدمه کوتاه درباره اهمیت branch predictorها بحث کنید.

۲. سپس به طور مختصر در مورد اهمیت این روش و تاثیر آن بحث کنید.

۳. در ادامه گزارش نحوه پیاده‌سازی خود را به صورت مختصر شرح دهید.

۴. نمودار درصد پیش‌بینی صحیح را به ازای برنامه‌های محک و حالت‌های مختلف هر روش رسم کنید.

۵. به ازای هر یک از این حالات اندازه جدول مورد نیاز را محاسبه کرده و در گزارش خود بیاورید. فرض کنید هر وزن perceptron ۸ بیت است.

قسمت امتیازی

پارامتر θ که در مقاله آمده است را به کد خود اضافه کنید. این پارامتر مربوط به آستانه یادگیری است. هدف از اضافه کردن این پارامتر کنترل وزن‌ها و جلوگیری از overflow است. آموزش زمانی اتفاق می‌افتد که قدر مطلق y کوچک‌تر از θ باشد. بنابراین مقدار هر یک از وزن‌ها کمتر از θ خواهد بود. (چرا؟) بنابراین تعداد بیت مورد نیاز برای نمایش هر وزن برابر $\lfloor log_2\theta \rfloor + 1$ خواهد بود و این تعداد بیت تضمین می‌کند هیچگاه overflow رخ ندهد.

[10] benchmark

۶. پروژه ششم: پیاده‌سازی Cache Replacement Imitating Belady's OPT Policy در Gem5

صورت پروژه

در این پروژه قصد داریم یک سیاست جایگزینی حافظه نهان را به Gem5 اضافه کنیم و با اجرای یک برنامه محک عملکرد آن را با سیاست‌های موجود مقایسه کنیم.

این روش در سال ۲۰۲۲ مطرح شده و جزئیات آن در این مقاله موجود است. شما برای پیاده‌سازی این پروژه لازم است که بخش III از این مقاله را مطالعه کنید. در ادامه نیز اجزای این سیاست جایگزینی به اختصار توضیح داده می‌شود اما توصیه می‌شود که قبل از پیاده‌سازی مقاله را حتما مطالعه کنید.

سیاست جایگزینی معرفی شده در مقاله

به طور کلی برای هر خط از حافظه نهان، زمان تقریبی رسیدن به آن خط یا ETA برابر جمع زمان کنونی و پارامتری به اسم Predicted reuse distance است. در هر اضافه کردن یک خط به حافظه نهان، خط با بالاترین ETA از حافظه نهان، evict می‌شود.

برای مطالعه جزئیات پیاده‌سازی به بخش III از مقاله ارائه شده در لینک مراجعه کنید.

گام‌های پروژه

۱. ابتدا بخش تعیین شده از مقاله را مطالعه کنید.
۲. سپس درک خود از نحوه چگونگی عملکرد این سیاست جایگزینی را در گزارش ارائه دهید.
۳. با راهنمایی‌های ارائه شده این سیاست را به سیاست‌های پیش‌فرض Gem5 اضافه کنید.
۴. با دانشی که از خواندن مقاله بدست آورده‌اید و تغییر فایل‌های منبع، سیاست مورد نظر را اضافه کنید.
۵. این سیاست را با سیاست‌های معروف موجود مقایسه کنید و سه برنامه محک مختلف را روی آن اجرا کنید و با شبیه‌سازی و تحلیل نتایج مشخص کنید کدام برنامه محک، عملکرد بهتری دارد.

راهنمایی

برای اضافه کردن یکی سیاست جدید به Gem5 لازم است که کد آن را تغییر دهید و سپس دوباره build کنید.

ابتدا در آدرس /src/mem/cache/replacement_policies/ دو دستور زیر را اجرا کنید:

```
cp lru_rp.cc [policy]_rp.cc
cp lru_rp.hh [policy]_rp.hh
```

و سپس فایل‌های جدید را با توجه به اسم سیاست جایگزینی تغییر دهید.

سپس به فایل ReplacementPolicies.py در آدرس /src/mem/cache/replacement_policies خطوط زیر را اضافه کنید:

```
class [POLICY]RP(BaseReplacementPolicy):
type = '[POLICY]]RP'
cxx_class = 'gem5::replacement_policy::[POLICY]'
cxx_header = "mem/cache/replacement_policies/[policy]_rp.hh"
```

نهایتا خط زیر را در فایل src/mem/cache/replacement_policies/SConscript اضافه کنید:

```
Source('[policy]_rp.cc')
```

همچنین [POLICY]RP را به آرایه SimObject در همان فایل اضافه کنید.

گزارش و ارزیابی

۱. در گزارشی که ارائه می‌دهید باید نحوه عملکرد این سیاست را همراه با دیاگرام‌های لازم به صورت مختصر توضیح دهید.

۲. سپس با اجرای سه برنامه محک مختلف روی این سیاست جایگزینی، سیاست LRU و سیاست LFU عملکرد این سیاست جایگزینی را به لحاظ پارامترهای مختلف مربوط به حافظه نهان مانند miss rate بررسی کنید.

۳. نهایتا به طور خلاصه در مورد پیچیدگی پیاده‌سازی این سیاست به صورت سخت‌افزاری و هزینه پیاده‌سازی آن بحث کنید.

قسمت امتیازی

یک اسکریپت پایتون بنویسید که ده برنامه محک مختلف را روی این سیاست و چهار سیاست معروف شبیه‌سازی کند و نتایج شبیه‌سازی را در قالب چند نمودار و یک فایل با فرمت csv ارائه دهد. این اسکریپت باید به صورت generic باشد و کارایی مناسبی داشته باشد. همچنین اگر بتواند چند شبیه‌سازی را به صورت موازی روی هسته‌های مختلف ماشین انجام دهد، امتیاز بالاتری به آن تعلق می‌گیرد.