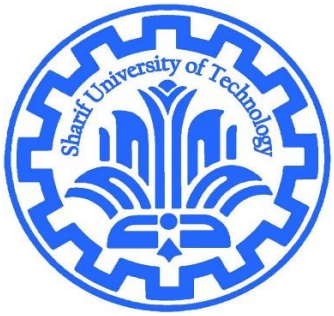
به نام خدا



عنوان پروژه:

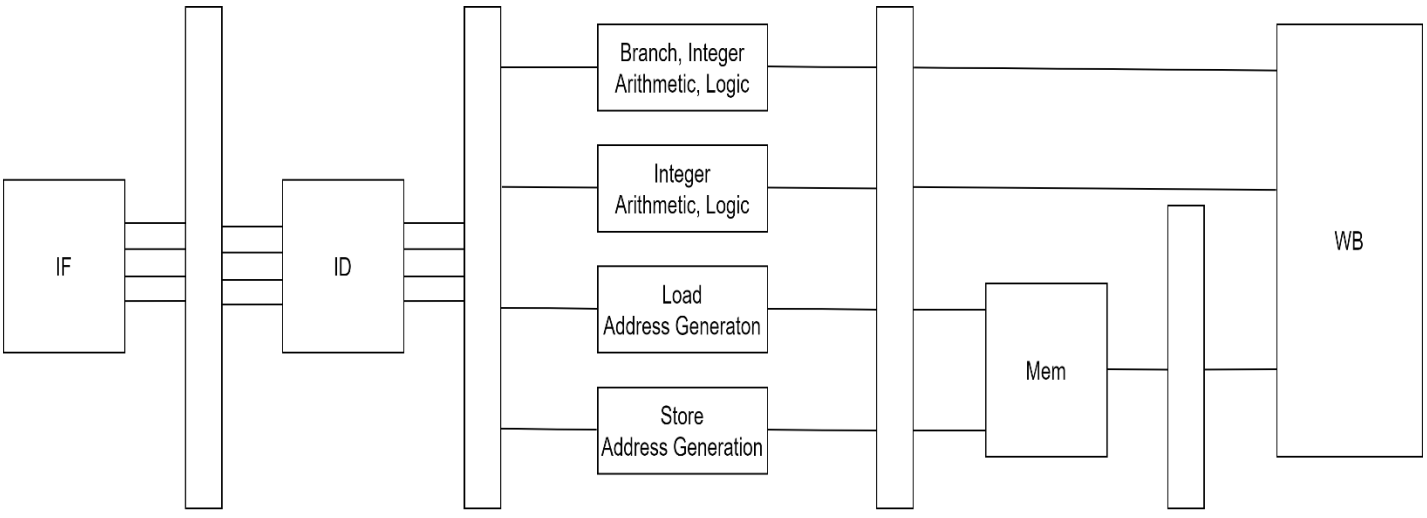
پیاده سازی پردازنده MIPS به صورت Four Issue

ساختار کامپیوتر و میکروپروسسور و آز

دکتر خسرو حاج‌صادقی

به موارد زیر قبل از انجام پروژه توجه بفرمایید:

- پروژه اول 70 درصد و پروژه دوم 100 درصد نمره کل امتیازی درس را دارد. شما می توانید پروژه ها را بصورت گروه های دو یا سه نفره نیز انجام دهید. برای پروژه اول، انجام پروژه تا دو نفر بدون جریمه است ولی انجام پروژه بصورت سه نفره 80 درصد نمره را به همراه خواهد داشت. همچنین برای پروژه دوم، شما مجاز به ایجاد گروه های سه نفره بدون جریمه هستید.
- توجه کنید که بصورت دقیق تشابه تمامی کدها چک می شود و درصورت کشف تقلب **نمره منفی** برای شما منظور خواهد شد. توجه کنید در بسیاری از قسمت های پروژه شما باید ایده ای را ایجاد و پیاده سازی کنید، بحث در مورد ایده ها با سایر گروه ها مانعی ندارد.
- اکثر سوالات طرح شده در پروژه ها به منظور ایجاد جهت فکری در پیاده سازی پروژه است و صرفا با در نظر گرفتن تست بنچی که در اختیارتان قرار می گیرد، آن ها را پاسخ دهید. حتما گزارش کاملی از پیاده سازی خودتان به همراه پاسخ سوالات مطرح شده تهیه نمایید، صرفا خروجی کافی نمی باشد.
- شما می توایند در بعضی از قسمت های پروژه برای مثال چک کردن خروجی های هر بخش، نتایج خود را با دستیار آموزشی بررسی بفرمایید.
- شماتیک کلی مربوط به پروژه اول در زیر آمده است.
- توجه کنید ایجاد تغییرات در تست بنچ به منظور همخوانی با ساختار طراحی شده توسط شما، وظیفه خودتان است.



پروژه اول

در این قسمت شما باید پردازنده خود را بصورت In-Order پیاده سازی کنید. برای اینکه چهارچوب مشخصی داشته باشیم تا ارزیابی درستی از تمام پروژه ها صورت بگیرد فرض های ساده کننده زیر را پیش از پیاده سازی در نظر بگیرید:

1. دستوراتی که در ارزیابی بررسی می شوند در زیر لیست شده اند:

R Format: add, sub, addu, subu, and, or, xor, nor, slt, sltu
I Format: addi, addiu, slti, sltiu, andi, ori, xori, lui, beg, bne, lw, sw

2. عمق پایپلاین واحدهای بخش Execute را برابر یک فرض کنید.
3. از تاخیر همه حافظه های موجود صرف نظر کنید.
4. تعداد پورت های نوشتن در رجیستر فایل برابر دو است و به تعداد دلخواه پورت خواندن دارد. تعداد commit ها در هر کلاک به انتخاب خودتان است.
5. از Exception موجود در دستورات صرف نظر کنید. (نیاز به پیاده سازی مسیری برای ISR [1]نیست)
6. طبق شماتیک صفحه اول در بخش Execute از واحدهای پردازشی زیر برخورداریم:
   - یک مسیر اجرای دستورات پرش و اعداد صحیح
   - یک مسیر اجرای دستورات اعداد صحیح
   - یک مسیر برای تولید آدرس و خواندن از حافظه (AG[2])
   - یک مسیر برای تولید آدرس و نوشتن در حافظه

بخش اول

در این بخش ابتدا فرض می کنیم مکانیزم تخمین پرش بصورت Static تعیین می شود. فرض کنید تمامی تخمین ها Always Not Taken هستند. در پیاده سازی خود به موارد زیر توجه کنید و به سوالات گفته شده پاسخ دهید:

- از آنجا که در مرحله decode ممکن است بعضی از دستورها به دلیل وابستگی به دستورات در حال اجرا و یا دستوراتی که در همان مرحله decode هستند اجازه ورود به بخش execute را نداشته باشند، نیاز است که بافری برای نگهداری آن ها طراحی شود. در مورد rob[3] و ساختار آن تحقیق کنید و بطور خلاصه آن را در گزارش خود توضیح دهید. همچنین بررسی کنید که افزایش سایز این بافر چه تاثیری در پیاده سازی و بهبود عملکرد خواهد داشت؟
- تمامی حالاتی که می توانیم با forwarding وابستگی بین دستورات را از بین ببریم را در گزارش خود ذکر کنید. (ذکر حالت کلی برای دستورات مشابه کافی است)
- همانطور که در شکل کلی ساختار ما می بینید در مرحله execute برای انجام عملیات بر روی اعداد صحیح دو عملگر لحاظ شده است که یکی از آن ها بطور مشترک محاسبات مربوط به branch را نیز انجام می دهد. آیا در مرحله decode تخصیصی از

[1] Interrupt Service Routine
[2] Address Generator
[3] Reorder buffer

دستورات به عملگرهای مرحله execute وجود دارد که بهینه باشد؟ برای پاسخ به این سوال ترتیب های مختلف از دستورات را در نظر بگیرید و حتما در پیاده سازی به آن توجه بفرمایید.

- بعد از پیاده سازی مقدار IPC[4] مدار خود را با استفاده از تست بنچ مورد نظر بدست آورید. تعداد کل دستورات را می توانید از زمان اجرای آزمایش Single Cycle بدست آورید. چرا؟
- اگر این پیاده سازی بصورت single issue انجام شود که خروجی آن همان pipeline موجود در درس است، با فرض کلاک 10 نانوثانیه، دارای زمان اجرای 344840 نانوثانیه خواهد بود. آیا پیاده سازی شما توانسته است بهبود قابل ملاحظه ای در زمان اجرا بدست آورد؟

بخش دوم

در این قسمت فرض مربوط به تخمین پرش Static را حذف می کنیم. در مورد الگوریتم های مختلف تخمین پرش تحقیق کنید و هر کدام را بطور خلاصه توضیح دهید. می توانید برای اینکار از اسلایدهای کمکی که در کنار فایل پروژه در اختیارتان قرار می گیرد استفاده بفرمایید. می خواهیم در این بخش بهینه ترین آن ها را برای پیاده سازی خودمان انتخاب کنیم. توجه کنید که معیار ما میزان miss-penalty کمتر بر روی تست بنچ نهایی است. نیازی به پیاده سازی سخت افزاری تمامی این روش ها به منظور یافتن بهینه ترین آن ها نیست بلکه شما می توانید با استفاده از کد isort32.ipynb که کد پایتون تست بنچ است، پیاده سازی های خود را بصورت نرم افزاری چک بفرمایید و بعد از یافتن بهینه ترین آن ها، پیاده سازی سخت افزاری را انجام دهید.
توجه کنید که شما باید جدولی مانند جدول زیر در گزارش خودتان داشته باشید که نشان می دهد با استفاده از این الگوریتم ها کمترین miss-penalty چقدر خواهد بود، همچنین ذکر کنید که این نتایج با در نظر گرفتن چه مشخصاتی در تخمینگر مورد نظر بدست آمده است.

Algorithm	Number of misses
Static: Not Taken	a
Dynamic: Last time predictor	b
Dynamic: Two-bit Counter based prediction	c
Dynamic: Global branch prediction	d
Dynamic: Gshare branch prediction	e
Dynamic: Local branch prediction	f

نتایج بالا با در نظر گرفتن تمامی حالت های ممکن باید بدست بیاید، برای مثال هم باید فرض کنید که سایز حافظه تخمینگر بصورتی است که collision در پرش ها اتفاق می افتد و نیز هم حالتی را در نظر بگیرید که collision اتفاق نمی افتد.

بعد از یافتن الگوریتم بهینه، کمترین مقدار حافظه برای آن را بدست بیاورید که با در نظر گرفتن پرش های مختلف در کد تست بنچ collision اتفاق نیفتد. این مقدار را در گزارش خود ذکر کنید.

بعد از پیاده سازی سخت افزاری، مقدار IPC را در این حالت محاسبه کنید و با قسمت قبل مقایسه بفرمایید.

[4] Instruction per clock

پروژه دوم

پروژه دوم در راستای پروژه اول تعریف می شود و بسیاری از قسمت های آن مشابه قسمت اول است. در این قسمت سعی داریم یکی دیگر از فرض های قسمت قبل را حذف کنیم. شما باید در این قسمت بتوانید پردازنده را به صورتی تغییر دهید که بتواند Out of order دستورات را اجرا کند. برای پیاده سازی این بخش موارد زیر را در نظر بگیرید:

- در مورد تمامی سوالاتی که در بخش قبل پرسیده شد مجدد تحقیق کنید. چه تغییراتی در rob باید برای پیاده سازی این قسمت داده شود؟ مجدد بررسی کنید آیا در این قسمت در نظر گرفتن سایز بالا برای بافر تاثیری بر روی عملکرد مدار ما خواهد گذاشت؟
- یکی از الگوریتم های مشهوری که برای اجرای out of order وجود دارد، الگوریتم tomasulo است. در مورد این الگوریتم تحقیق کنید و ساختار آن را بطور خلاصه در گزارش خود توضیح دهید. بطور خاص شما باید در مورد register renaming و ساختار Reservation Station ها تحقیق بفرمایید.
- در مورد hazard[5] WAR و hazard[6] WAW تحقیق بفرمایید و توضیح دهید که چگونه الگوریتم tomasulo این موارد را رفع می کند.
- در مورد ابعاد RS[7]ها بحث نمایید. چه field های نیازی است که در RS تعریف شود؟ بالا بردن ابعاد RS چه مشکلاتی به همراه خواهد داشت؟
- با توجه به ساختار پروژه ما، چند آدرس virtual برای register های جدید در register renaming لازم است؟ با در نظر گرفتن worst case این مقدار را بدست آورید، همچنین فرض کنید که نمی خواهیم به دلیل عدم وجود register خالی stall در مدار اتفاق بیفتد.
- یکی از اتفاقاتی که ممکن است در RS دیده شود این است که در یک کلاک مشخص، دو دستور آماده اجرا باشند و از آنجا که صرفا یک عملگر برای آن ها موجود است ما باید بین آن ها انتخاب کنیم. قاعدتا دستوری بهتر است انتخاب شود که زودتر به RS ارسال شده است. (چرا؟) برای اینکه بتوانیم این کار را انجام دهیم، چه روشی پیشنهاد می دهید؟ توصیه می کنم بر روی این سوال خوب فکر کنید و Selector در قسمت خروجی RS ها را با این دید پیاده سازی کنید.
- بررسی کنید که آیا افزایش تعداد پورت ها کمکی به عملکرد مدار می کند؟ (به منظور افزایش تعداد commit ها در هر کلاک)
- آیا در این ساختار forwarding معنا دارد؟ توضیح دهید.
- توصیه می شود اگر مایل به انجام پروژه دوم هستید، در ابتدا فرض کنید از تخمینگر ساده Static استفاده می کنیم و در انتها بخش دوم قسمت قبل را انجام دهید.
- مانند قسمت قبل IPC را برای مدار پیاده سازی شده بدست آورید و آن را در گزارش خود ثبت نمایید.

[5] Write After Read
[6] Write After Write
[7] Reservation Station