به نام خدا

Odd-Day Team

پروژهٔ درس معماری کامپیوتر

فاز سوم

اعضای تیم:

رضا عرفان‌آرانی

کیان عمومی

علیرضا مرادیان

آرش توانگر

دانشگاه صنعتی شریف

ترم بهار 1401

شرح فاز سوم:

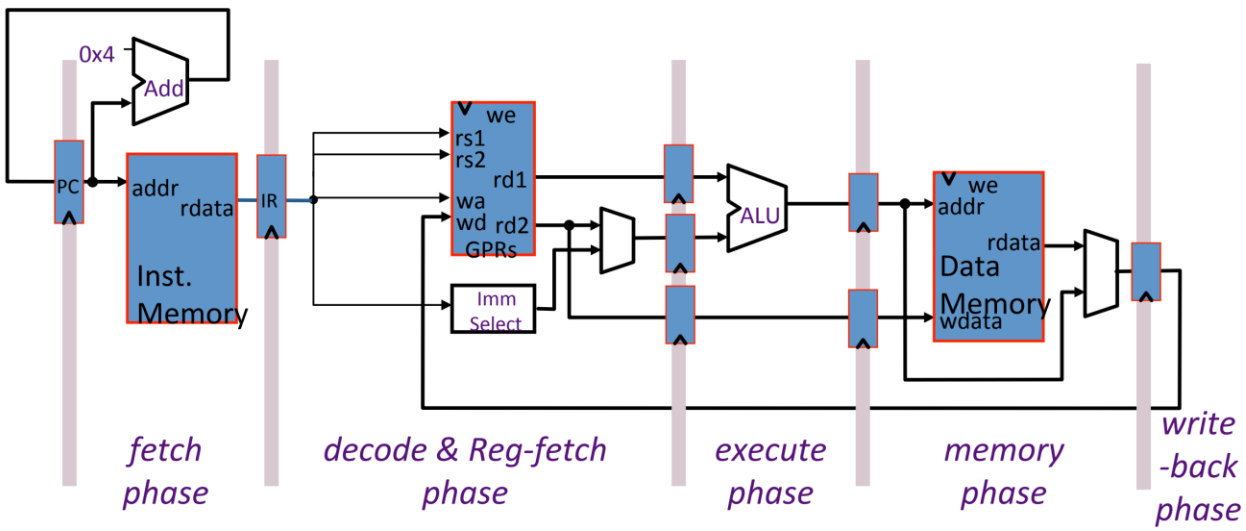
در این فاز ما به تغییر دادن معماری پردازندهٔ خود به صورت خط لوله‌ای پرداختیم.

برای انتخاب تعداد مراحل خط لوله ما طبق مطالب تدریس‌شده در درس، همان مدل 5 مرحله‌ای مشهور را پیاده‌سازی کردیم.

دلیل انتخاب خط لولهٔ 5 مرحله‌ای:

- توجه داریم که هر چه تعداد مراحلمان کمتر باشد، اولاً فرکانس clock کمتر می‌شود که این به این معناست که هر دستور به طور میانگین در تعداد کلاک بیشتری اجرا می‌شود و CPI بالاتری را نتیجه خواهد داد. از طرفی با این که تعداد مراحل خط لولهٔ بالا باعث افزایش فرکانس clock می‌شود، اما با زیاد بودن بیش از حد آن پر نگه داشتن خط لوله سخت‌تر می‌شود (به دلیل آن که ممکن است در میان آن bubble بیاید.) درحالیکه می‌دانیم زمانی خط لوله عملکرد پردازنده را بهبود می‌بخشد که در اکثر مواقع پر باشد. همچنین تعداد مراحل بالا در خط لوله منجر به افزایش سخت‌افزار مورد استفاده خواهد شد. تعداد مراحل بالای خط لوله با این که اجرای تعداد دستورات بیشتری را در یک لحظه پیش می‌برد اما همین موضوع ممکن است موجب بروز وابستگی بالاتری بین مراحل دستورات مختلف شود.
- حال با توجه به توضیحات بند قبل و جست جو در اینترنت و مطالعه در درس ما خط لوله 5 مرحله‌ای را انتخاب کردیم. چرا که خط لوله 5 مرحله‌ای به طور میانگین نه تعداد بسیار زیادی مرحله دارد که موجب اشکالات وابستگی و تاخیر به واسطه bubbleها شود و نه آنقدر کم مرحله دارد که طول clockمان کم باشد.

معماری کلی خط لولهٔ 5 مرحله‌ای برای پردازندهٔ Risc-V:



توضیحات مربوط به چگونگی عملکرد معماری فوق در درس داده شده است. در اینجا ما به توضیح در مورد نحوهٔ پیاده‌سازی این معماری در پروژهٔ خود می‌پردازیم:

- برای جداسازی مراحل فوق در معماری قبلیمان، ما نخست با استفاده از DFF سعی کردیم مراحل را به وجود بیاوریم. این دی-فلیپ-فلاپ‌ها را در شکل بالا می‌توانید قبل و بعد از هر مرحله روی مستطیل‌های صورتی مشاهده کنید. با این کار با هر کلاک ما به سراغ instruction بعدی می‌رویم و در عین حال مراحل decode & Reg-fetch، execute، و.... نیز برای دستورات قبلی اجرا می‌کنیم.
- در تست‌هایی که ما وابستگی داده مشاهده کردیم، با اضافه کردن دستور nop در محل‌های مناسب به فایل‌های mem. این مخاطرات را رفع کردیم و توانستیم تست‌های بدون branch و jump را به طور صحیح پاس کنیم.