**قسمت سوم آزمایش پیاده سازی MIPS (جلسه سوم):**

در جلسات گذشته، مراحل واکشی، کدگشایی، اجرا و نوشتن در رجیسترفایل(WB) پردازنده پیاده سازی گردید. در این جلسه، ابتدا مرحله‌ی حافظه و ماژول تشخصی مخاطره را پیاده سازی کنید، سپس کل پردازنده را تست کنید.

برای پیاده سازی حافظه نکات زیر را در نظر بگیرید:

۱- خواندن به صورت ترکیبی و نوشتن در حافظه به صورت ترتیبی انجام می‌شود. نوشتن در لبه بالارونده رونده‌ی کلاک صورت می‌گیرد.

۲- در این مرحله به کمک سیگنال MEM_W_EN داده‌ها در حافظه نوشته می شود. هرگاه این سیگنال مقدارش در سر بالارونده کلاک یک باشد، مقدار در در آدرس مشخص شده از حافظه، تغییر می‌کند.

۳- داده‌های درون حافظه به کمک سیگنال MEM_R_EN خوانده می‌شود.

۴- این حافظه تنها یک خط آدرس دارد که هم برای نوشتن و هم برای خواندن استفاده می شود. آدرس در حقیقت داده‌ی محاسبه شده تو سط ALU در مرحله‌ی قبل می‌باشد. این مقدار در رجیسترهای پایپ ذخیره شده است.

۵- برای پیاده‌سازی فضای حافظه می توانید از تعریف آرایه‌ای از رجیسترها استفاده کنید. مانند زیر:

Reg [31:0] memory [0:63];

۶- خواندن و نوشتن فقط از آدرس‌های مضرب ۴ (به دلیل ۳۲ بیتی بودن معماری) انجام می‌شود. به طور مثال: در ازای خواندن از آدرس‌های ۱۰۲۴، ۱۰۲۵، ۱۰۲۶ و ۱۰۲۷ نتایج یکسانی خوانده می‌شود یعنی ۴ بایت از آدرس ۱۰۲۴ خوانده می شود.

۷- فضای آدرس‌دهی حافظه داده از ۱۰۲۴ شروع می شود. در این صورت برای خواندن یا نوشتن آدرس ۱۰۲۴ مقادیر ۴ بایت اول آرایه memory یعنی memory[0] خوانده یا در آن نوشته می شود (همه آدرس های ورودی به این مرحله از ۱۰۲۴ کسر شود).

۸- پس از اضافه کردن واحد حافظه به پردازنده، یک کد برای تست پردازنده بنویسید که شامل هازارد نباشد (چون هنوز واحد تشخیص هازارد به پردازنده اضافه نشده است).

۹- ماژول Write Back را پیاده سازی نمایید، این ماژول بسیار ساده خواهد بود، این ماژول داده خوانده شده از حافظه و داده محاسبه شده توسط ALU را به همراه شماره رجیستر مقصد از طریق رجیسترهای پایپ‌لاین به عنوان ورودی دریافت می‌کند و با توجه به سیگنال‌های کنترلی محتوای رجیسترفایل را تغییر می‌دهد. در رجیسترهای پایپ‌لاین میان دو مرحله‌ی memory و write back ، سیگنال های کنترلی WB_EN و MEM_R_EN ذخیره شده‌اند. هرگاه WB_EN برابر یک باشد، با توجه به MEM_R_EN داده‌ی خوانده شده از حافظه یا داده‌ی محاسبه شده توسط ALU در رجیستر فایل نوشته می شود.

حال باید ماژول تشخصی مخاطره افزوده شود اما در این بخش به طور مختصر انواع مخاطره ها را ذکر می کنیم:
به طور کلی در پردازنده‌ها سه نوع مخاطره وجود دارد:

**الف) مخاطره ساختاری:** این مخاطره در بطن ساختار خط لوله وجود دارد به همین دلیل ساختاری نا گرفته است. این مخاطره بین مرحله WB و ID به دلیل همزمانی خواندن و نوشتن از ثبات های عمومی ناشی می شود. برای رفع این مخاطره نوشتن در ثبات‌های عمومی را به لبه پایین رونده منتقل کردیم. بنابراین این مخاطره در پردازنده رفع شده است.

**ب) مخاطره کنترلی:** این مخاطره ناشی از دستورات پرش است. در صورتی که دستور پرش وارد خط لوله شود به دلیل تأخیر در تشخیص و محاسبه آدرس پرش دو دستور به اشتباه وارد خط لوله می‌شود. برای رفع این مشکل سیگنال های Flush به پردازنده افزوده شد. بنابراین این محاطره نیز رفع شده است.

**ج) مخاطره داده ای:** مخاطره داده‌ای به صورت زیر دسته‌بندی می گردد:

۱- **خواندن پس از نوشتن (RAW):** مخاطره خواندن بعد از نوشتن در حالتی رخ می‌دهد که یک دستور، رجیستری که هنوز محاسبه یا ذخیره نشده است را فراخوانی می‌نماید. در این حالت می‌بایست دستورات جدید تا محاسبه یا ذخیره شدن آن رجیستر متوقف گردند. در مثال زیر دستور ۲ نیاز به رجیستر R2 دارد که دستور ۱ آنرا محاسبه می نماید، بنابراین تا ذخیره یا محاسبه شدن مقدار R2 دستور ۲ باید متوقف گردد.

1. SUB R2,R0,R1
2. AND R3,R2,R1

در این آزمایش سعی در برطرف نمودن این نوع هازارد را داریم.

۲- **نوشتن پس از خواندن (WAR):** در مثال زیر رجیستر R1 در حالاتی ممکن است قبل از خوانده شدن توسط دستور ۱ مقدار آن به وسیله دستور ۲ تغییر کند. به این رخداد هازارد داده‌ای از نوع WAR گفته می‌شود.

1. SUB R2,R0,R1
2. AND R1,R3,R4

این نوع هازارد در پردازنده‌های In-Order مانند پردازنده مورد بحث در این دستور کار رخ هرگز رخ نخواهد داد.

۳- **نوشتن پس از نوشتن(WAW):** هازارد داده‌ای نوشتن پس از نوشتن نیز همچون هازرد WAR در پردازنده‌های In-Order رخ نمی‌دهد. این نوع مخاطره ممکن است در پردزنده های Out-of-order رخ دهد و ترتیب نوشتن در رجیستر مقصد تغییر کند.

برای رفع هازارد RAW ماژول Hazard Detection Unit همانند شکل زیر به پردازنده ARM اضافه نمایید. در این واحد، منابع Src1 و Src2 در مرحله ID با مقصدهای مراحل EXE و MEM به صورت مجزا مقایسه می‌شود و در صورت برابر بودن یکی از منابع با مقصدها، سیگنال کنترلی Hazard_Detected_Signal را برابر ۱ قرار می‌دهد. این سیگنال باید دستورات درون IF و رجیسترهای پس از آن را متوقف نماید و حبابی را به خط لوله تزریق نماید. برای ایجاد حباب کافی است تمامی سیگنال های حیاتی پردازنده صفر گردد. برای پیاده سازی این مرحله به ترتیب زیر عمل کنید:

۱- واحد تشخیص هازارد با حداقل ورودی‌های زیر ایجاد نمایید.

```
module hazard_Detection_Unit(
    input [4:0]src1,
    input [4:0]src2,
    input [4:0] Exe_Dest,
    input Exe_WB_EN,
    input [4:0]Mem_Dest,
    input Mem_WB_EN,
    output hazard_Detected
);
```

۲- تمامی حالاتی که هازارد RAW رخ می‌دهد را در نظر بگیرید و خروجی واحد تشخیص هازارد را یک کنید. این حالت ها به شرح زیر است:

- برابری src1 با مقصد EXE در صورت یک بودن WB_EN در مرحله اجرا
- برابری src1 با مقصد MEM در صورت یک بودن WB_EN در مرحله حافظه
- برابری src2 با مقصد EXE در صورت یک بودن WB_EN در مرحله اجرا و دو منبعی بودن دستور
- برابری src2 با مقصد MEM در صورت یک بودن WB_EN در مرحله حافظه و دو منبعی بودن دستور

۳- به دستورات Immediate که دارای تنها یک منبع هستند توجه نمایید. سیگنال دارای منبع دوم بودن که براساس سیگنال فوری بودن (immediate) و دستور STR تولید شده است را نیز به هازارد اضافه کنید.

۴- ورودی‌های مورد نیاز واحد هازارد را به خروجی دیگر مراحل متصل نمایید.

۵- قابلیت Freeze را به رجیستر PC در مرحله IF و رجیسترهای بعد از IF اضافه نمایید و خروجی واحد تشخیص هازارد را به آن متصل نمایید.

۶- برای ایجاد حباب در خط لوله سیگنال‌های کنترلی خروجی از واحد کنترل را صفر نمایید. برای اینکار خروجی ماژول مخاطره را با خروجی بررسی شرط دستورات or کنید.

۷- دستورات برنامه محک را بدون تغییر اجرا نمایید.

۱۰- پردازنده خود را اشکال‌یابی نمایید و درستی نتایج را به کمک دستیار آموزشی بررسی نمایید.

۱۱- گزارش این بخش را به همراه خروجی SignalTap تکمیل نمایید.

۱۲- در بعضی از دستورات src1 وجود ندارد اما کد پیاده سازی شده آنها را تشخیص نمی‌دهد. در صورت رفع این مشکل ۵ درصد نمره اضافه منظور می‌شود. در صورت پیاده سازی تغییرات خود را در گزارش قید کنید.

موفق باشید

نصیحت‌کن