گزارش کار پروژه نهایی پردازنده‌ی RISC-V

درس: آزمایشگاه مدارهای منطقی و معماری کامپیوتر

استاد: مهندس محمد لالی

دانشجو: بهنیا فرهبد

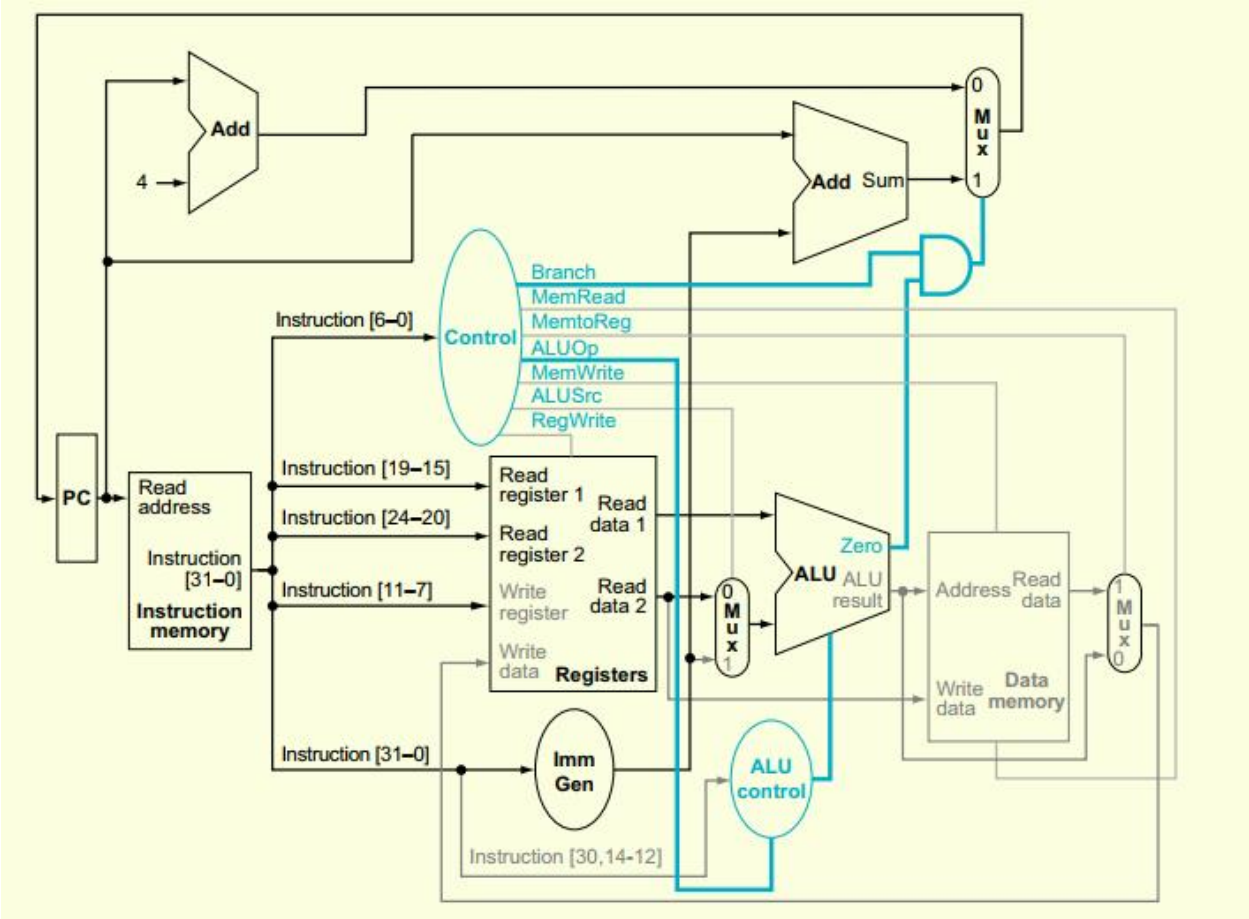
شماره‌ی دانشجویی: ۹۸۲۰۲۳۰۲۰

زمستان ۱۴۰۰

در این گزارش سعی میکنم تمام بخش هایی را که انجام داده‌ام به طور کامل شرح دهم. اما متاسفانه نتوانستم

بخش ۲ پروژه که پردازنده‌ی پایپ لاین شده بود را بسازم.

Data path ای که قصد پیاده سازی آن را داریم:



۱. پردازنده‌ی ما از PC یا همان Program counter شروع می‌شود. این ماژول برای دسترسی به آدرس دستوری که قرار است اجرا شود استفاده می‌شود. بدین گونه که با استفاده از یک Mux، هر بار یا ٤ بایت به مقدار آدرس قبلی خود اضافه می‌کند، یا اگر از دستورات jal, bne, beq و... استفاده شود، آن مقدار آدرس مخصوصی که به این ماژول ارسال می‌شود را می‌گیرد. عمل جمع توسط یک جمع کننده به نام adder.v انجام می شود.

۲. سپس داده‌ی خروجی PC به یک رجیستر ٦٤ بیتی که دستورالعمل خوانده شده از حافظه را در خود نگه می‌دارد ارسال می‌شود، تا دستور خوانده شود و بخش های مختلف دستورالعمل به واحد های مختلف ارسال شود و دستور بعد از دیکُد شدن اجرا شود.

۳. ۱. دستور به یک Register File که رجیستر های مختلفی در آن قرارداده شده ارسال می‌شود تا در انجام محاسبات و اجرای دستورات از آنها استفاده کنیم. همزمان با آن (صفحه‌ی بعد)

۲. بخشی از دستور دودویی شده به واحدControl که مغز متفکر سیستم ماست فرستاده می‌شود که قرار است به کمک دستورالعملی که بصورت ورودی دریافت می‌کند سیگنال های خروجی کنترل کننده را صادر کند . این سیگنال ها به بخش های مختلفی ورود پیدا کرده و مشخص می‌کنند هر قسمت به چه صورت و با چه ترتیبی کار کنند. در نهایت همزمان با دو مورد قبلی ->

۳. بخشی از دستور به واحد Sign extend یا Immediate Generator ارسال می‌شود. این واحد کمک می‌کند تا یک عدد را با حفظ ارزش اش، به تعداد بیت های بیشتری توسعه دهیم ، که در پروژه ما یک عدد ۳۲ بیتی را با حفظ ارزش به یک عدد ٦٤ بیتی تبدیل می‌کند.

٤. بعد از مرحله‌ی قبلی، داده‌ها به واحد ALU Control ارسال می‌شود تا مشخص شود که چه عملی روی داده‌های ورودی ALU باید صورت گیرد؛ که این عمل‌ها شامل جمع، تفریق،logical and، logical or است. خود واحد ALU

نیز دو ورودی (از واحد ALU Control) گرفته و طبق مقدار سیگنالی که از این واحد میآید ، پردازش موردنظر را روی دو داده ورودی انجام می دهد.

۵. سپس داده‌ی خروجی مرحله‌ی قبل به واحد Data Memory که می‌تواند تعدادی رجیستر در کنار پردازنده باشد(که همان کار RAM یا Cache را انجام می‌دهد)، ارسال می‌شود. برنامه های اسمبلی (کد ماشین) در این حافظه نوشته می‌شود، البته کاربرد دیگر آنن وشتن و خواندن داده در آن است که می توان اعدادی را در خانه‌های آن نوشت یا از آن خواند.

بخش سوم گزارش‌کار:

Hazard ها: در بین ۵ دستور داده شده، دو Hazard در خط‌های اول و دوم (استفاده از رجیستر x۲۰) و در خط‌های دوم و سوم (استفاده از رجیستر X۲۱) داریم.