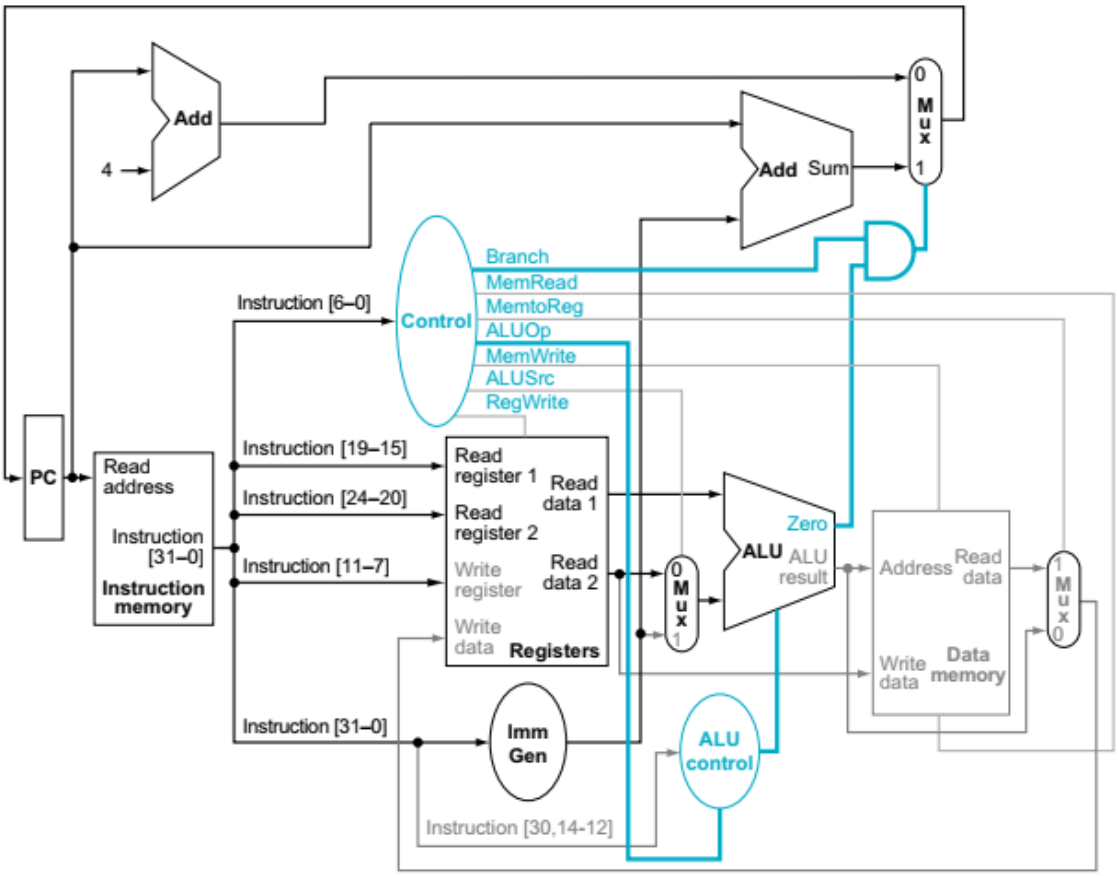
گزارش کار پروژه نهایی

نام و نام خانوادگی: بهار بهزادی پور

شماره دانشجویی: ۹۸۲۰۲۳۰۰۶

Part1 :

در پارت ۱ طبق شکل زیر یک cpu تک سایکل پیاده سازی کردم :



طی جلسات گذشته هر کدام از قسمت های شکل بالا را به صورت مجزا پیاده سازی کردم و سپس در فایل SS_CPU آنها را مطابق datapath به یکدیگر متصل کردم.

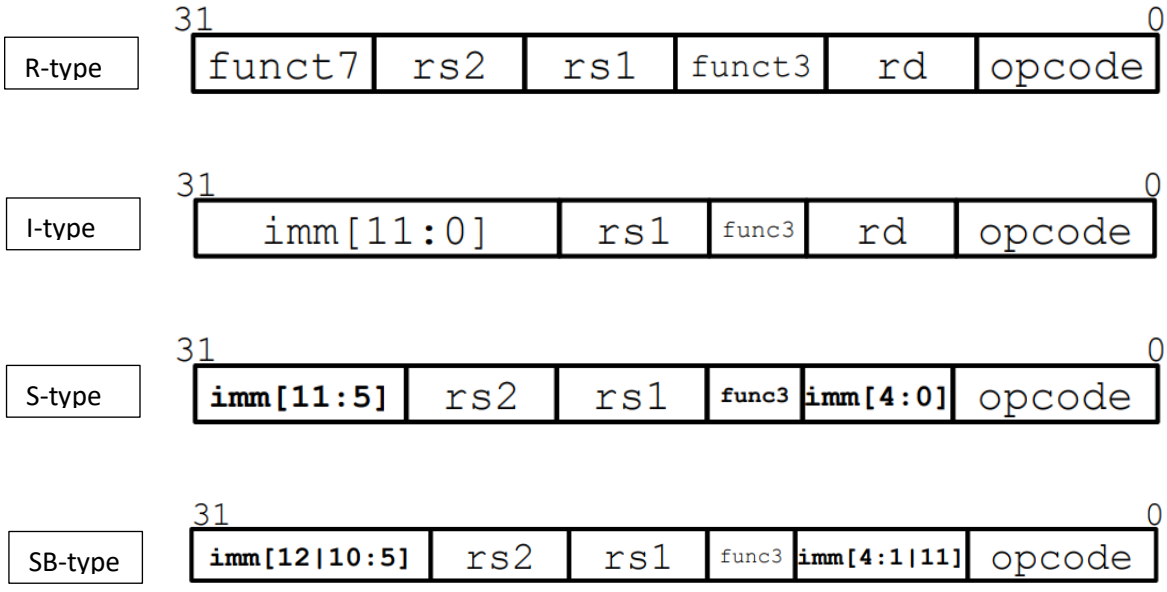
در ابتدا آدرس مورد نظر از طریق رجیستر pc به instruction memory ارسال میشود. در instruction memory به آدرس مورد نظر در memory رفته و instruction آن آدرس را استخراج میکند و ۷ بیت اول آن (opcode) را به control ، بیت های ۱۵ تا ۱۹ و ۲۰ تا ۲۴ را به عنوان read register به register_file ارسال میکند. ۵ بیت ۷ تا ۱۱ را به عنوان write register به register_file می فرستد. یک adder وجود دارد که ادرس خارج شده از pc را بعلاوه ۴ میکند تا یک خانه جلو برود و دستور بعدی را بخواند. در اینجا برای مالتی پلکسری که داده را به pc می رساند یک مالتیپلکسر جداگانه طراحی کردم که اولین بار آدرس را صفر در نظر گرفته و از دفعات بعدی بین داده های ورودی با توجه به مقدار s داده خروجی را انتخاب میکند. بعد از این تفکیک برای اینستراکشن ، مقادیر branch ، memRead ، memToReg ، AluOp ، MemWrite ، AluSrc ، RegWrite با توجه به opcode ورودی کنترل ، مقدار دهی میشوند و به سایر بخش ها ارسال می شوند.

پس از آن register file ، read data1 را به alu می فرستد و read data2 به عنوان ورودی به مالتیپلکسر ارسال می شود تا بین این داده و خروجی immgen با توجه به aluSrc که از کنترل آمده است یکی را انتخاب کرده و به alu ارسال کند. در نهایت alu با توجه به داده های ورودی و خروجی که از طرف alu control به آن ارسال شده دو خروجی zero و alu result را بدست آورده و alu result را به عنوان آدرس برای data memory میفرستد. Read data2 نیز که از register file خارج شده بود وارد data memory میشود. Read data را خروجی میدهد ، که همراه alu result به مالتیپلکسر می رود و با توجه به memToReg خروجی را تولید و به write data برای register file میفرستند. این چرخه تا اتمام دستورات ادامه پیدا خواهد کرد.

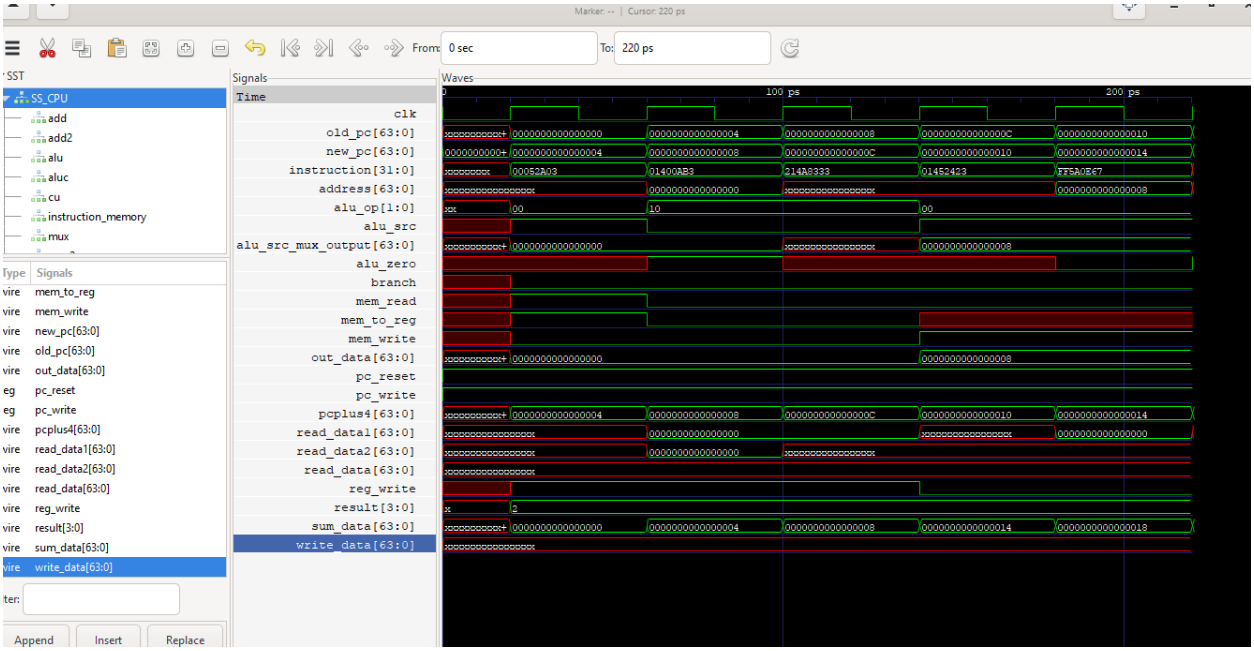
دستورات داده شده در دستورکار را به فرم باینری تبدیل کردم و سپس در instruction ، دستورات داده شده را به عنوان مقدار پیشفرض قرار دادم. دستورات عبارت اند از:

```
lw x20, 0(x10)      → 000000000000 01010 010 10100 0000011
add x21, 0, x20     → 0000000 10100 00000 000 10101 0110011
sub x6, x21, x20    → 010000 10100 10101 000 00110 0110011
sw x20, 8(x10)      → 0000000 10100 01010 010 01000 0100011
beq x20,x21,-4      → 1111111 10101 10100 000 11100 1100111
```

دستور اول از نوع I-type دستور دوم و سوم از نوع R-type و دستور چهارم از نوع S-type و دستور آخر از نوع SB-type می باشند. به کمک فرمت زیر این تبدیل را انجام دادم :



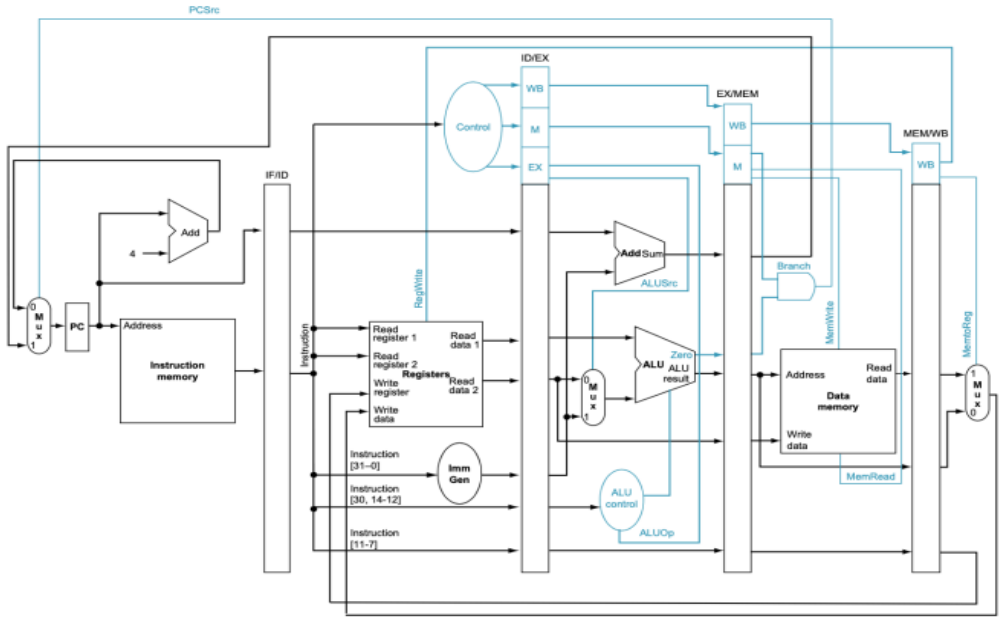
در نهایت خروجی برنامه پارت ۱ به شکل زیر شد:



همانطور که در تصویر فوق مشخص است هر ۵ instruction به درستی خوانده شده و مقادیر مرتبط به دست آمده اند.

Part2 :

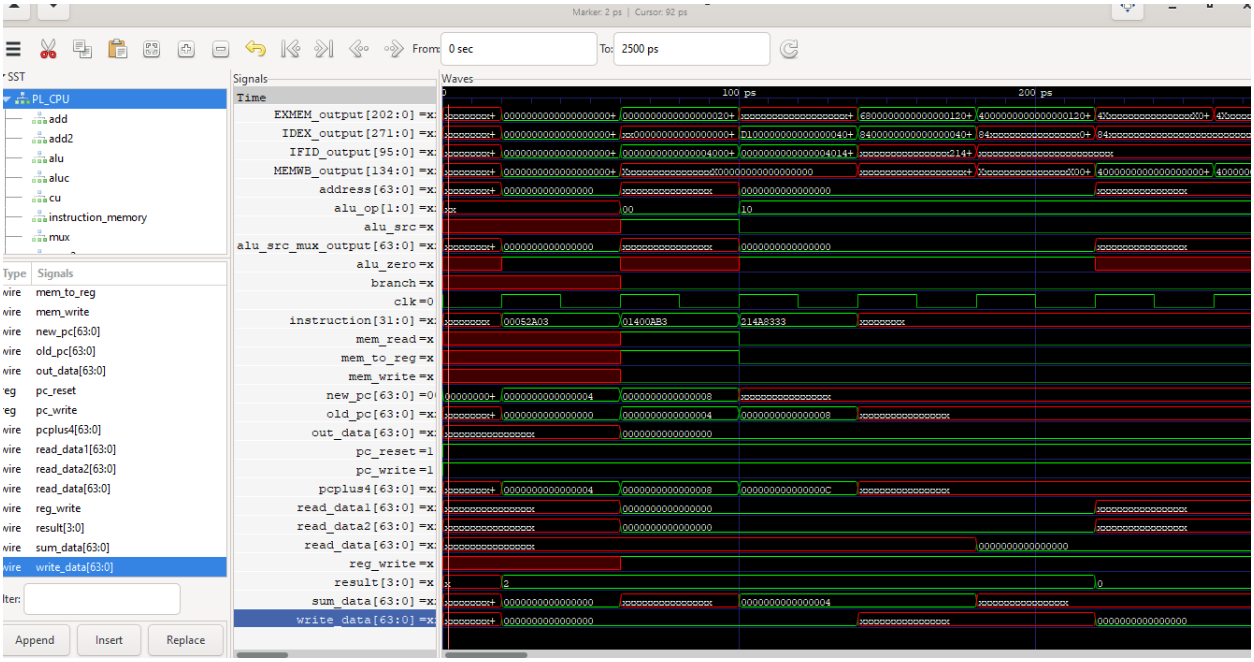
در پارت ۲ طبق شکل زیر یک cpu که با استفاده از pipeline کار میکند را پیاده سازی کردم :



برای پیاده سازی pipeline cpu از پارت ۱ استفاده میکنیم با این تفاوت که datapath به ۵ قسمت تقسیم می شود که آنها را استیج ۱ تا ۵ می نامیم. در بین هر دو قسمت یک رجیستر با نام های IF/ID ، ID/EX ، EX/MEM ،MEM/WB وجود دارد.

این رجیستر ها را مشابه PC پیاده سازی کردم با این تفاوت که IF/ID یک رجیستر ۹۶ بیتی است چون ۳۲ بیت از اینستراکشن میگیرد و ۶۴ بیت از خروجی PC که همان آدرس است و در مجموع میشود ۹۶ بیت. هنگام فراخوانی در PL_CPU ورودی را به صورت کانکت اینستراکشن و آدرس خروجی از PC در نظر گرفتم. خروجی هم ۹۶ بیتی است . ۷ بیت از آن را به کنترل اختصاص داده ، ۶۴ بیت برای رجیستر بعدی ، ۵ بیت برای read register1 ، ۵ بیت برای read register2 ، ۳۲ بیت برای immgen ، ۳ و ۵ بیت دیگر هم برای رجیستر بعدی اختصاص داده میشود. رجیستر بعدی ، ID/EX است که یک رجیستر ۲۷۲ بیتی است چون ۸ بیت از کنترل ، ۶۴ + ۳ + ۵ بیت از رجیستر IF/ID ، ۲تا ۶۴ بیت از register file و ۶۴ بیت از immgen میگیرد. پس دارای ورودی ۲۷۲ بیت و خروجی ۲۷۲ بیت می باشد. بخشی از خروجی را به رجیستر بعدی یعنی EX/MEM و بخشی را به addSum ، Alu ، Mux و Alu control ارسال میکند. رجیستر بعدی EX/MEM نام دارد که یک رجیستر ۲۰۳ بیتی است. این رجیستر در ورودی جمعا ۷۴ بیت از رجیستر قبلی یعنی ID/EX ، ۶۴ بیت از addSum ، ۶۵ بیت از Alu میگیرد. و به رجیستر بعدی و data memory خروجی می دهد. رجیستر آخر هم MEM/WB نام دارد و یک رجیستر ۱۳۵ بیتی است. ۷۱ بیت از رجیستر قبلی و ۶۴ بیت از Datat memory به عنوان ورودی دریافت میکند و یک خروجی ۱۳۵ بیتی تولید میکند که به مالتیپلکسر استیج آخر و register file ارسال میشود. در datapath اصلی یک forwarding unit وجود دارد که با توجه به دستورکار در این پروژه پیاده سازی نشد. این واحد نقش کنترل hazard ها را بر عهده دارد.

خروجی کار به صورت زیر است :



با توجه به دستورات این برنامه :

```
lw x20, 0(x10)
add x21, 0, x20
sub x6, x21, x20
sw x20, 8(x10)
beq x20,x21,-4
```

می توان گفت در خط ۱ و ۲ برای x20 یک hazard اتفاق می افتد. در خط ۲ و ۳ نیز برای x21 یک hazard داریم.