پروژه درس ساختار کامپیوتر و میکروپروسسور

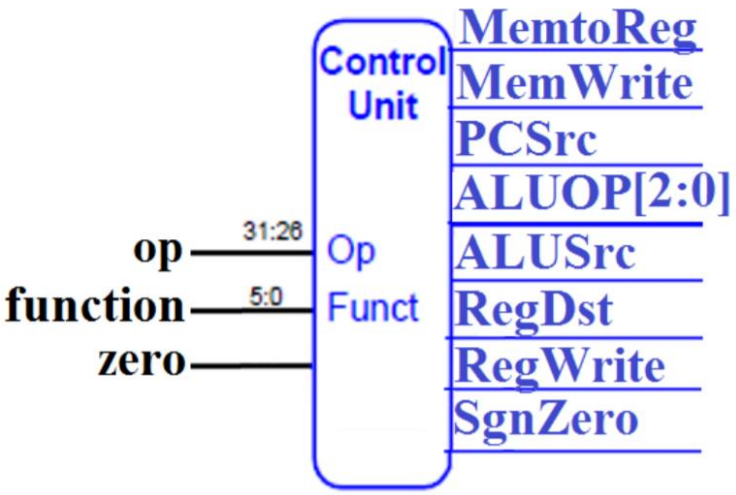
دکتر موحدیان

فاز دوم

طراحی پردازنده ی Single Cycle

بهار 99

قسمت اول: پیاده سازی ماژول Controller



ورودی‌ها:

1) ورودی شش بیتی برای شش بیت بالای دستور. (op)

2) ورودی شش بیتی برای شش بیت پایین دستور. (funct)

3) ورودی تک بیتی مطابق با صفر و یا یک بودن خروجیALU . (zero)

خروجی ها:

1) خروجی تک بیتی برای بیت کنترلی مالتی پلکسر در خروجی Memory Data برای تعیین آن که داده ورودی برای RegFile از خروجی حافظه Data دریافت شود و یا از خروجی ALU (MemToReg).

2) خروجی تک بیتی برای بیت کنترلی مربوط به فعالسازی نوشتن به Data Memory. (MemToReg)

** نکته : در این فاز پارامتر ND تاخیر DataMemory که در فاز قبل نوشته اید را صفر بگیرید. همچنین در این فاز استفاده ای از سیگنال MemReady این ماژول نخواهیم کرد اما آن را در پردازنده (top module) قرار دهید.

3) خروجی تک بیتی برای بیت کنترلی مالتی پلکسر در ورودی Instruction Memory.

برای تعیین آن که PC خط بعد دستورات به عنوان آدرس Memory Instruction قرار بگیرد و یا آدرس پرش. (PCSrc)

4) خروجی تک بیتی برای بیت کنترلی مالتی پلکسر در ورودی دوم ALU برای تعیین آن که ورودی دوم ALU خروجی دوم File Register باشد و یا Signe Extend Immediate . (ALUSrc)

5) خروجی تک بیتی برای بیت کنترلی مالتی پلکسر در ورودی File Register برای تعیین آن که آدرس رجیستر مقصد از کدام بازه دستور برداشت شود. (RegDst)

6) خروجی تک بیتی برای بیت کنترلی مربوط به فعالسازی نوشتن به Register File . (RegW)

7) خروجی تک بیتی برای بیت کنترلی مربوط به آن که واحدSign Extend ، ورودی اش را zero extend کند یا sign extend. (SgnZero)

8) خروجی سه بیتی برای مشخص کردن نوع عملیاتALU . (ALUOP)

هدف:

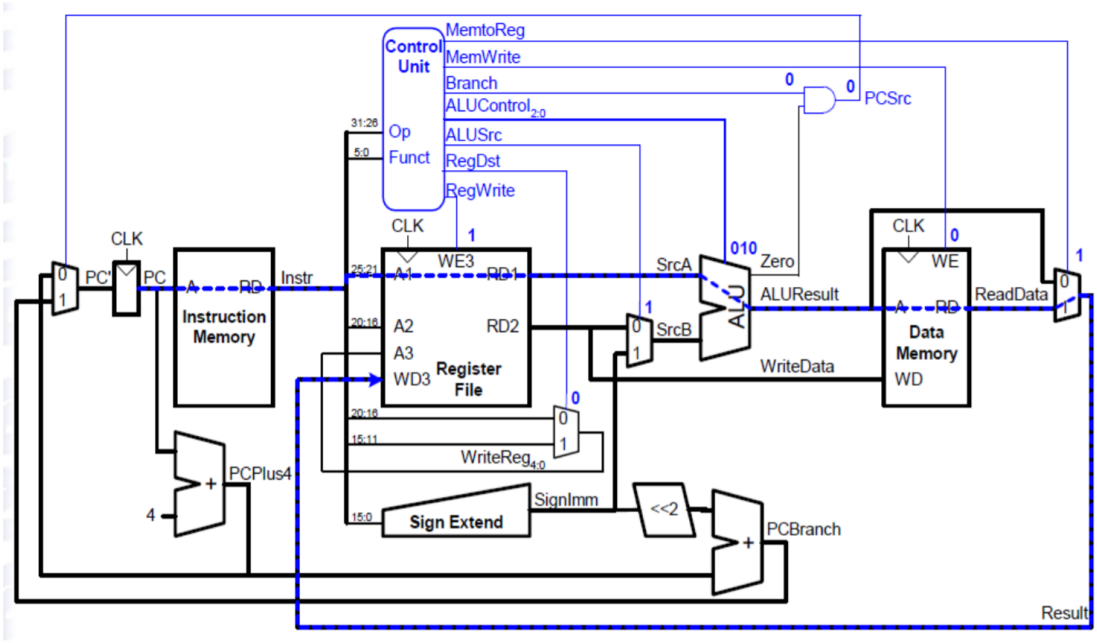
هدف طراحی ماژولی با ورودی و خروجی های ذکر شده می باشد که با دریافت شش بیت بالا و پایین دستور و ورودی تک بیتی zero مربوط به صفر و یک بودن خروجی ALU ،تمامی بیت های کنترلی را با توجه به نوع دستور مقدار دهی کند.

دستورات مورد نیاز برای پیاده سازی:

ADD - ADDU - SUB - SUBU - AND - OR - XOR - NOR - SLT - SLTU - LW - SW

BEQ - BNE - ANDI - ORI - XORI - ADDI - ADDIU - SLTI - SLTIU

قسمت دوم: پیاده سازی ماژول DataPath



هدف در این قسمت پیاده سازی ساختار بالا است که ساختار یک پردازنده‌ی Single Cycle است که در هر کلاک 1 دستور را از حافظه دستورالعمل خوانده و اجرا میکند.

برای پیاده سازی این قسمت لازم است:

1) تعدادی مالتی پلکسر و جمع کننده و رجیستر PC با توجه به ساختار بالا پیاده سازی شود.

2) پیاده سازی واحدSign Extend و واحد شیفت دهنده 2 تایی.

3) از ماژول های طراحی شده در قسمت قبل و فاز قبل instance گرفته شود و این ماژول ها به بقیه مدار متصل شوند.

تست عملکرد پروسسور:

در نهایت DataPath و Controller را به هم متصل کرده و پردازنده خود را کامل کنید. در این جا باید کارکرد پردازنده ی خود را تست کنید. پردازنده ی شما باید بتواند برنامه های مختلف (که با Instruction set پیاده شده در بخش های قبل نوشته شده باشند) را اجرا کند.

تست 1 : Sort

برای این بخش کد اسمبل شده ی اجرای الگوریتم Insertion Sort در اختیارتان قرار گرفته است. این برنامه 96 عدد نامرتب تولید کرده و آن ها را با الگوریتم مرتب سازی درجی مرتب کرده و در حافظه ذخیره می کند. پردازنده شما باید بتواند این برنامه را اجرا کرده و نتیجه مطلوب را بدهد. برای چک کردن خروجی برنامه از تست بنچی که در اختیارتان قرار داده شده استفاده کنید. نتیجه روی کنسول باید به این صورت باشد:

```
# ffff8a4f ff49a03e ed9232cf ed9232cf ec6c3298 ec6c3298 e6663185 e6663185 dddd8526 dbdb842d b6b6e9be b4b4e947 aac70a4f aaaaec60 a48e7be0 a48e7be0
# 9c7a4305 9c7a4305 99bd6c60 8bd654a6 8bd654a6 8894b185 878c0526 86b259c7 86b259c7 82846947 826e5d18 826e5d18 81da5ead 81da5ead 8183b298 8081042d
# 807f69be 802ab2cf 8019fbe0 8008c305 8007d4a6 8002d9c7 8001dd18 8000dead 8000203e 8000203e 7fff8a4f 7fff8a4f 7f49a03e 7f49a03e 6d9232cf 6c6c3298
# 66663185 5ddd8526 5ddd8526 5bdb842d 5bdb842d 36b6e9be 36b6e9be 34b4e947 34b4e947 2ac70a4f 2ac70a4f 2aaaec60 2aaaec60 248e7be0 1c7a4305 19bd6c60
# 19bd6c60 0bd654a6 0894b185 0894b185 078c0526 078c0526 06b259c7 02846947 02846947 026e5d18 01da5ead 0183b298 0183b298 0081042d 0081042d 007f69be
# 007f69be 002ab2cf 002ab2cf 0019fbe0 0019fbe0 0008c305 0008c305 0007d4a6 0007d4a6 0002d9c7 0002d9c7 0001dd18 0001dd18 0000dead 0000dead 0000203e
```

همچنین زمان اجرا و تعداد کلاک اجرای این برنامه را در گزارش خود بیاورید.

تست 2 : Fibonacci Sequence

در این بخش باید خودتان یک برنامه برای آزمایش پردازنده بنویسید و آن را اجرا کنید. با استفاده از دستورات mips (فقط دستوراتی که پردازنده ی شما پوشش میدهد) کدی بنویسید که اعداد 1 تا 15 دنباله ی فیبوناچی را حساب کرده و در خانه های 16 تا 30 حافظه دیتا ذخیره کند.

برای تبدیل کد خود به زبان ماشین میتوانید از یک mips assembler استفاده کنید یا خودتان دستی این کار را انجام دهید. برنامه ی نهایی زبان ماشین در یک فایل hex به نام fibtest ذخیره کرده (با فرمتی مشابه isort32.hex) و آن را در محل تست بنچ قرار دهید.

توجه کنید که در خط آخر برنامه ی خود یک لوپ بی نهایت به صورت BEQ $zero $zero 0xFFFF اضافه کنید که پردازنده در آن نقطه بماند. به تست بنچ fib_tb.v بروید و پارامتر end_pc مطابق محل آخرین خط

کد خود قرار دهید تا تست بنچ در آن نقطه تمام شود. توجه کنید که هر خط 4 بایت میباشد و باید تعداد خطوط کد خود را در 4 ضرب کنید.

در صورت درستی کد و پردازنده باید 15 عدد اول دنباله را ببینید. در این بخش نیز زمان اجرا و تعداد کلاک را در گزارش خود بیاورید. همچنین کد اسمبلی خود را نیز همراه فایل ها آپلود کنید.

خواسته های پروژه:

1- طراحی کد وریلاگ واحد های بالا و انجام تست های خواسته شده و ارائه نتایج صحیح.

2- گزارشی کوتاه در مورد نحوه نوشتن واحد ها و کارکرد کلی مدار و نتایج و زمان اجرای تست ها.

فایل های خود را به فرمت rar درآورید در غالب فایلی با اسم Modules_Student#.rar در cw آپلود کنید که منظور از student# شماره دانشجویی شما می‌باشد.