بسمه تعالی

درس ساختار کامپیوتر و میکروپروسسور

استاد : دکتر موحدیان عطار

دانشکده مهندسی برق



گزارش فاز دوم پروژه درس ( پیاده سازی پردازنده single cycle بر اساس دستور العمل mips)

امیرحسین صفری

97101994

واحد های سازنده پردازنده :

واحد ControlUnit :

در این واحد با پس از ورودی گرفتن ، سیگنال های op ( opcode ) ، funct ( function ) و zero ، سیگنال های کنترلی خروجی را به عنوان ورودی به ماژول datapath می دهیم ، برای این کار ابتدا با استفاده از opcode ، R\_type یا I\_type بودن دستورات چک شده و سپس با استفاده از switch ، سیگنال های کنترلی مختص هر دستور مشخص می شود ( برای دستورات R\_type ، فقط خروجی ALUop ، مختض هر دستور و در واقع بر اساس function ورودی تعیین می شوند و سایر سیگنال های کنترلی مثل هم می باشند ، اما در دستورات I\_type ، تمام سیگنال های کنترلی مختص هر دستور و بر اساس opcode ، مشخص می شوند)

واحد datapath :

در این واحد ، ابتدا clk و reset را ورودی می گیریم و در ادامه سیگنال های خروجی واحد ControlUnit می باشد به علاوه سیگنال instr ( instruction ) از واحد Instruction\_mem ( Instruction memory ) و هم چنین سیگنال readdata از واحد t\_DataMemory ( data memory ) را به عنوان ورودی می گیریم .

و هم چنین سیگنال های خروجی zero ، pc ، alu\_result ، writedata را برای ورودی دادن به ControlUnit و Instruction memory و data memory را تولید می کنیم .

برای ساخت این واحد از واحد های زیر استفاده شده است :( نام سیگنال ها بر اساس شکل صفحه بعد انتخاب شده اند )

1 ) ماژول reset\_ff : یک D\_flipflop که بر سر راه pc قرار می گیرد تا در صورت reset شدن مدار ، pc برابر با صفر وارد Instruction memory شود .

2 ) ماژول PCplus4 : برای ساختن pc بعدی ( سیگنال PCPlus4 ) این سیگنال در صورت اتفاق نیفتادن branch وارد Instruction memory می شود .

3 ) ماژول shift : از این ماژول برای در 4 ضرب کردن ( شیفت دادن به تعداد 2 به چپ ) سیگنال SignImm استفاده می شود .

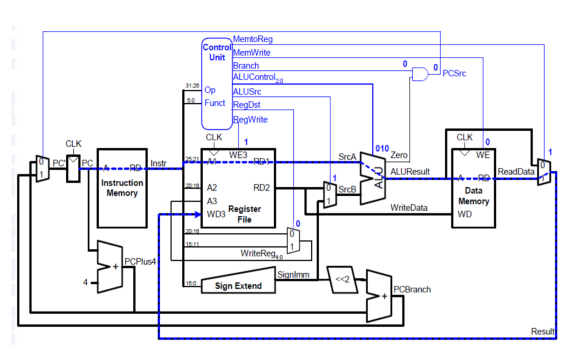
4 ) ماژول adder : با استفاده از این ماژول سیگنال PCPlus4 را با 4 برابر SignImm جمع کرده و سیگنال PCBranch را می سازد . ( این سیگنال در صورتی که branch رخ داده باشد ، وارد Instruction memory می شود )

5 ) ماژول MUX : از این ماژول در 4 مکان مختلف به شرح زیر استفاده می گردد :

1 ) مشخص کردن اینکه ورودی PCBranch یا PCPlus4 وارد Instruction memory شود .

2 ) برای اینکه سیگنال WriteReg را مشخص کند ، در واقع آدرس رجیستری که در آن داده ذخیره می شود را بر اساس اینکه دستور R\_type یا I\_type می باشد ، مشخص می کند.

3 ) برای مشخص کردن ورودی دوم (SrcB ) واحد ALU ، از بین داده ی موجود در رجیستر اگردستور R\_type باشد و یا داده ی immediate اگر دستور I\_type باشد .

4 ) و بالاخره muxای که مشخص کند که آیا داده ی خروجی data memory به عنوان سیگنال Result مشخص شود و یا خروجی ALU .

6 ) ماژول RegisterFile : برای کار با رجیستر های موجود و نوشتن داده و یا خواندن داده از آن.

7 ) ماژول signORzeroExtend : برای sign\_extend کردن مقدار immediate ( برای دستوراتی مثل addi و beq ) یا zero\_extend کردن آن ( برای دستوراتی مثل andi )

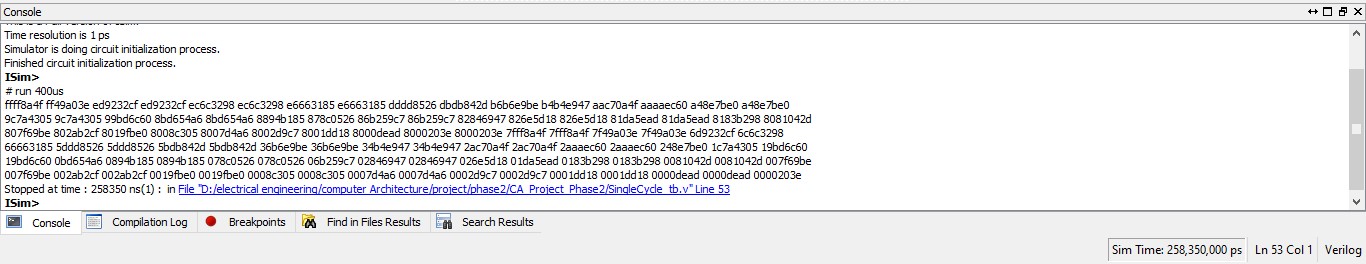
8 ) ماژول ALU : برای اینکه خروجی مورد نظر را بر اساس نوع دستور و در واقع سیگنال ALUcontrol .

حال با استفاده از دو ماژول ControlUnit و datapath ماژول mips را ساخته و در واقع در این مرحله پردازنده ساخته می شود ، سپس با اتصال کردن Instruction\_mem و t\_DataMemory به پردازنده ، ارتباط پردازنده نیز با حافظه برقرار می شود .

نتایج و زمان اجرای تست ها

1 ) تست 1 : sort

برای این تست ضمن اینکه تغییراتی جزیی مطابق با نام ماژول ها را در فایل SingleCycle\_tb اعمال می کنیم ، آن را اجرا کرده و نتیجه مطابق زیر می شود .





همانطور که در تصویر نیز مشخص است ، زمان اجرای کد 258350 ns می باشد .

2 ) تست Fibonacci Sequence

در ابتدا کد mips ای که 15 عدد اول دنباله ی فیبوناچی را محاسبه و در حافظه ذخیره می کند را در فایل fibtest.asm نوشته و سپس با استفاده از نرم افزار Mars ، فایل hex متناظر با آن را با نام fibtest.hex را تولید می کنیم و در ادامه مطابق آن چه که در دستور پروژه گفته شده فایل تست بنچ fib\_tb.v را تکمیل کرده و پس از اجرای آن ، خروجی را به صورت زیر مشاهده می کنیم : ( هم چنین زمان اجرا 1280 ns می باشد )

