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

۹۷۱۰۱۹۹۴

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)

۱) 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 .

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

۳) ماژول shift : از این ماژول برای در ۴ ضرب کردن (شیفت دادن به تعداد ۲ به چپ) سیگنال SignImm استفاده می شود .

۴) ماژول adder : با استفاده از این ماژول سیگنال PCPlus4 را با ۴ برابر SignImm جمع کرده و سیگنال PCBranch را می سازد . (این سیگنال در صورتی که branch رخ داده باشد ، وارد Instruction memory می شود)

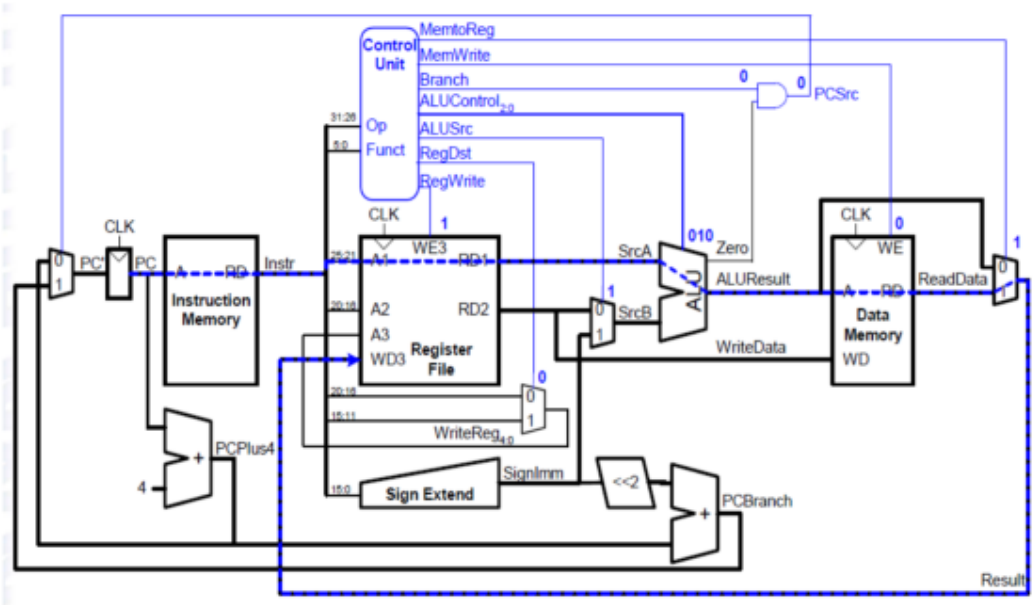
۵) ماژول MUX : از این ماژول در ۴ مکان مختلف به شرح زیر استفاده می گردد :

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

۲) 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.

۳) 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 .

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



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

۷) ماژول signORzeroExtend : برای sign_extend کردن مقدار immediate (برای دستوراتی مثل addi و beq) یا zero_extend کردن آن (برای دستوراتی مثل andi)

۸) 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

۱) تست ۱ : 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 .

۲) تست Fibonacci Sequence

در ابتدا کد mips ای که ۱۵ 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)