پاسخ تمرین شماره ۳ درس معماری کامپیوتر

امیر حسین عاصم یوسفی
۹۶۱۱۰۳۲۳

۱۴ اردیبهشت ۱۳۹۸

سوال ۱ :

برای این که متوجه شویم که چه اتفاقی بر اثر این خطا رخ می دهد یک دستور add را بر روی این پردازنده انجام می دهیم . به این صورت که ابتدا در رجیستر R31 مقدار صفر را قرار و مقدار یک را داخل رجیستر R30 قرار می دهیم و بعد دستور add R30 , R31 , R31 را اجرا می کنیم و انتظار داریم که در R30 مقدار صفر ذخیره شده باشد ولی چون مقدار Write Register به خاطر خطا برابر با یک می باشد بنابراین نتیجه در رجیستر R31 نوشته می شود و مقدار موجود در رجیستر R30 همان یک باقی می ماند . همان طور که می بینیم در صورتی که در دستور خود بخواهیم بر نتیجه بر روی رجیستر با اندیس زوج ذخیره کنیم به خاطر این خطا مقدار نتیجه در رجیستر فرد نوشته می شود . بنابراین ما باید برنامه را طوری تغییر دهیم که فقط از رجیستر ها با اندیس زوج استفاده کند .
با این خطا همواره مقدار حاصل از دستورات branch نتیجه واکشی دستورات اشتباه می باشد که با توجه به این موضوع دیگر نمی توانیم هیچ دستور branch ای را به صورت درست انجام دهیم پس پردازنده با این خطا غیر قابل استفاده است .

سوال۲ :

همان طور که میدانیم دستور LW نسبت به سایر دستورات زمانی بیشتری نیاز دارد .
با توجه به این که در صورت سوال قید نشده که تاخیر کدام بلوک را ۱۰ درصد کاهش می دهیم بنابراین SpeedUp را برای دستور گفته شده به ازای کاهش تاخیر تمام بلوک ها به دست می آوریم :
ابتدا باید مشخص کنیم که برای این دستور از کدام بلوک های موجود در مسیرداده باید عبور کنیم :

I-Mem

Regs

Mux : برای انتخاب کردن ورودی

ALU

D-Mem

Mux : برای انتخاب داده ای که باید در رجیستر نوشته شود

با توجه به بالا مقدار زمان برای دستور مورد نظر قبل از کاهش به صورت زیر است :

$$LW_{Time} = 200ps + 90ps + 20ps + 90ps + 250ps + 20ps = 670ps$$

I-Mem

حال ۱۰ درصد از تاخیر این بلوک کم می کنیم که برابر می شود با $Delay_{IMem} = 180$ پس مقدار تاخیر برای دستور مورد نظر به صورت زیر می باشد :

$$LW_{Time} = 180ps + 90ps + 20ps + 90ps + 250ps + 20ps = 650ps$$

بنابراین SpeedUp به صورت زیر می باشد :

$$\text{SpeedUp} = \frac{670}{650} = 1/0307$$

Add

با توجه به این که این واحد در مسیر این دستور قرار ندارد بنابراین کم کردن تاخیر این واحد هیچ تاثیری در سرعت این دستور ندارد بنابراین در این حالت SpeedUp ای وجود ندارد .

Mux

حال ۱۰ درصد از تاخیر این بلوک کم می کنیم که برابر می شود با $Delay_{Mux} = 18$ پس مقدار تاخیر برای دستور مورد نظر به صورت زیر می باشد :

$$LW_{Time} = 200ps + 90ps + 18ps + 90ps + 250ps + 18ps = 666ps$$

بنابراین SpeedUp به صورت زیر می باشد :

$$\text{SpeedUp} = \frac{670}{666} = 1/0060$$

ALU

حال ۱۰ درصد از تاخیر این بلوک کم می کنیم که برابر می شود با $Delay_{ALU} = 81$ پس مقدار تاخیر برای دستور مورد نظر به صورت زیر می باشد :

$$LW_{Time} = 200ps + 90ps + 20ps + 81ps + 250ps + 20ps = 661ps$$

بنابراین SpeedUp به صورت زیر می باشد :

$$\text{SpeedUp} = \frac{670}{661} = 1/0136$$

Regs

چون زمان این بلاک به اندازه ALU می باشد بنابراین مقدار SpeedUp آن هم یکسان است و برابر است با ۱/۰۱۳۶

D-Mem

حال ۱۰ درصد از تاخیر این بلوک کم می کنیم که برابر می شود با $Delay_{DMem} = 225$ پس مقدار تاخیر برای دستور مورد نظر به صورت زیر می باشد :

$$LW_{Time} = 200ps + 90ps + 20ps + 90ps + 225ps + 20ps = 645ps$$

بنابراین SpeedUp به صورت زیر می باشد :

$$\text{SpeedUp} = \frac{670}{645} = 1/0387$$

Sign-Extend

با توجه به این که این واحد در مسیر این دستور قرار ندارد بنابراین کم کردن تاخیر این واحد هیچ تاثیری در سرعت این دستور ندارد بنابراین در این حالت SpeedUp ای وجود ندارد .

Shift Left 2

با توجه به این که این واحد در مسیر این دستور قرار ندارد بنابراین کم کردن تاخیر این واحد هیچ تاثیری در سرعت این دستور ندارد بنابراین در این حالت SpeedUp ای وجود ندارد .

ALU Ctrl

با توجه به این که این واحد در مسیر این دستور قرار ندارد بنابراین کم کردن تاخیر این واحد هیچ تاثیری در سرعت این دستور ندارد بنابراین در این حالت SpeedUp ای وجود ندارد .

با توجه به مطالب بالا می توان دریافت که SpeedUp برای این دستور در بهترین حالت مقداری برابر با ۱/۰۳۸۷ دارد .

سوال ۳ :

الف

برای این قسمت جدول به صورت زیر می باشد :

RegWrite	*RegDst*	*Branch*	*MemRead*	*MemtoReg*	*MemWrite*	*ALUSrc*

ب

برای این قسمت جدول به صورت زیر می باشد :

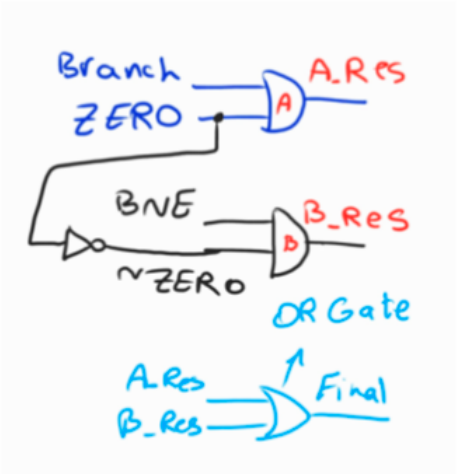
RegWrite	*RegDst*	*Branch*	*MemRead*	*MemtoReg*	*MemWrite*	*ALUSrc*

سوال ۴ :

برای این کار باید یک سیگنال کنترلی به اسم BNE به سیگنال های کنترلی خود اضافه کنیم در مرحله بعد باید یک گیت AND اضافه کنیم که ورودی های آن به شرح زیر می باشد :

۱- سیگنال BNE
۲- خروجی ZERO مربوط به ALU را NOT می کنیم و به گیت AND می دهیم . (علت این کار در ادامه آمده است)

برای این که بتوان دستور خواسته شده را پشتیبانی کرد باید به صورت زیر تغییرات را اعمال کنیم :



علت استفاده از گیت OR این می باشد که پردازنده بتواند بعد از اعمال تغییرات هنوز هم از دستور BEQ پشتیبانی کند .
علت این که خروجی ZERO مربوط به ALU را NOT می کنیم این است که وقتی دو مقادیر موجود در دو رجیستر با یک دیگر برابر نیستند این خروجی مقدار صفر را به خود می گیرد که باعث می شود خروجی A-Res همواره صفر شود و در این صورت چون سیگنال کنترلی Branch را هم خودمان برابر صفر قرار دادیم پس خروجی B-Res نیز برابر با صفر می شود و بنابراین همواره خروجی Final برابر با صفر می شود پس همواره فقط مقدار PC را به $PC + 4$ تغییر می دهیم که اشتباه است . به همین دلیل خروجی صفر ALU را NOT می کنیم . تا خروجی B-Res برابر با یک شود که باعث می شود خروجی Final یک شود و مالتی پلکسر مقدار PC را به $PC + 4 + (imm«2)$ تغییر می دهد و دستور BNE انجام می شود .
برای این که این دستور به درستی انجام شود مقادیر سیگنال های کنترلی به شرح زیر خواهد بود :

RegWrite	*RegDst*	*Branch*	*MemRead*	*MemtoReg*	*MemWrite*	*ALUSrc*

سوال ۵ :

برای این منظور تنها کافی است مالتی پلکسری که ورودی MemtoReg دارد را تغییر داده و به جای آن از یک مالتی پلکسر ۳ به یک استفاده کنیم که در ورودی دوم خود مقدار PC را می گیرد . حال برای این که مقدار PC را داخل رجیستر rt قرار دهد باید سیگنال های کنترلی را به صورت زیر تغییر دهیم :

RegWrite	*RegDst*	*Branch*	*MemRead*	*MemtoReg*	*MemWrite*	*ALUSrc*

سوال ۶ :

با توجه به این که دستور از نوع R-type می باشد بنابراین آدرس رجیستر مقصد در بیت های [15-11] می باشد پس $RegDst = 1$.
چون نتیجه را می خواهیم در بانک ثبات ذخیره کنیم پس $RegWrite = 1$
چون این یک دستور Branch نیست پس $Branch = 0$

با توجه به این که دو مقدار خوانده شده را به عنوان ورودی به ALU می دهیم پس $ALUSrc = 0$
چون در این دستور نه از حافظه می خوانیم و نه بر روی حافظه می نویسیم پس
$MemRead = 0$, $MemWrite = 0$
با توجه به این که مقدار خروجی از ALU را باید به بانک ثبات برگردانیم پس $MemtoReg = 0$
چون دستور add داریم بنابراین $ALUOp = 01$ (ADD)
همچنین چون دستور Branch نداریم مقدار PC را به PC+4 تغییر می دهیم بنابراین $PC = 0x0016$
مقدار گذرگاه ها به شرح زیر می باشد :

$$\$ \text{ rt} = \$ \ 10 = \text{Instruction } [20\text{-}16] = 01010$$
$$\$ \text{ rd} = \$20 = \text{Instruction } [15\text{-}11] = 10100$$
$$\$ \text{ rs} = \$9 = \text{Instruction } [25\text{-}21] = 01001$$

با توجه به مقادیر بالا داریم

$$\text{Read } \ \text{data1} = 1$$
$$\text{Read } \ \text{data2} = 2$$
$$\text{ALU } \ \text{Result} = 3$$
