# پاسخ تمرین شماره۴ درس معماری کامپیوتر

امیر حسین عاصم یوسفی
۹۶۱۱۰۳۲۳

۲۲ اردیبهشت ۱۳۹۸

## سوال ۱ :

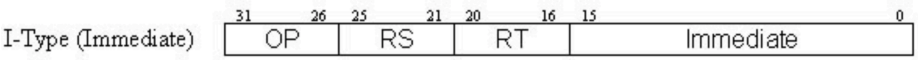
### الف

برای این قسمت جدول به شکل زیر می باشد :

| *Instruction* | *ALUSrc* | *MemRead* | *RegDst* | *RegWrite* | *ALUResult* |
|---|---|---|---|---|---|
| *addi $8, &9, -4* | 1 | 0 | 0 | 1 | 0x40 |
| *lw $t2, 8($at)* | 1 | 1 | 0 | 1 | 0x4C |

### ب

با توجه به این که ساختار دستورات I type به صورت زیر می باشد :




در صورتی که خطای کفته شده اتفاق بیافتد همیشه مقدار آدرس رجیستر مقصد را از بیت های ۱۱ تا ۱۵ می خواند که با توجه به شکل بالا این مقدار نامعتبر می باشد زیرا ۱۵ بیت اول نشان دهنده مقدار immediate می باشد و آدرس رجیستر داخل آن قرار ندارد بنابراین این دستور دیگر به درستی انجام نمی شود .

## سوال ۲ :

### الف

این دستور دستور beq می باشد . و علت استفاده از این بخش محاسبه کردن آدرس دستور بعدی می باشد اما با توجه به قرمول زیر

$$PC = PC + 4 + (offset << 2)$$

### ب

زمانی که PC را با ۴ جمع می کنیم در واقع ۴ بیت بالایی آدرس دستور بعدی مشخص می شود ولی چون آدرس بعدی در این نوع دستورات طبق فرمول گفته شده می باشد بنابراین باید ۲۸ بیت پایین PC را با offset«2 پر کنیم و چون offset ۲۶ بیت دارد با دو بار شیفت دادن به سمت چپ و تبدیل کردن آن به ۲۸ بیت آن را به جای ۲۸ بیت پایینی PC قرار می دهیم تا به این ترتیب ۳۲ بیت آدرس دستور بعدی محاسبه شود.

## پ

سیگنال ZERO زمانی یک می شود که مقدار موجود در دو رجیستر با یک دیگر برابر باشد و زمانی صفر می شود که مقدار موجود در دو رجیستر با یک دیگر برابر نباشند بنابراین از این سیگنال برای شناسایی برابر بودن مقدار داخل دو رجیستر استفاده می شود .
سیگنال کنترلی BRANCH برای کنترل کردن انجام عملیات از نوع BRANCH یعنی (beq و زمانی یک می شود که دستور ما به صورت پرش به یک دستور با شرایط باشد .
یکی از آن ها کافی نیست زیرا اگر فقط سیگنال ZERO داشته باشم آن گاه هر بار که مقدار داخل دو رجیستر با یک دیگر برابر باشد مقدار PC با $PC + 4 + (offset«2)$ جایگزین می شود که این می تواند اجرای دستورات مانند sub را دچار مشکل کند به این ترتیب که اگر دو عدد مساوی را از یکدیگر کم کنیم مقدار حاصل صفر می شود و به این ترتیب مقدار سیگنال ZERO برابر با یک می شود بنابراین دیگر پردازنده به دستور بعدی نمی رود و مقدار PC معتبر نخواهد بود .
از طرفی اگر فقط سیگنال BRANCH داشته باشیم بازهم نمی توانیم هر بار که این سیگنال یک می شود بدون توجه به برابری مقدار داخل دو رجیستر پرش انجام می شود . که این باعث درست اجرا نشد دستورات از نوع BRANCH میشود .

---

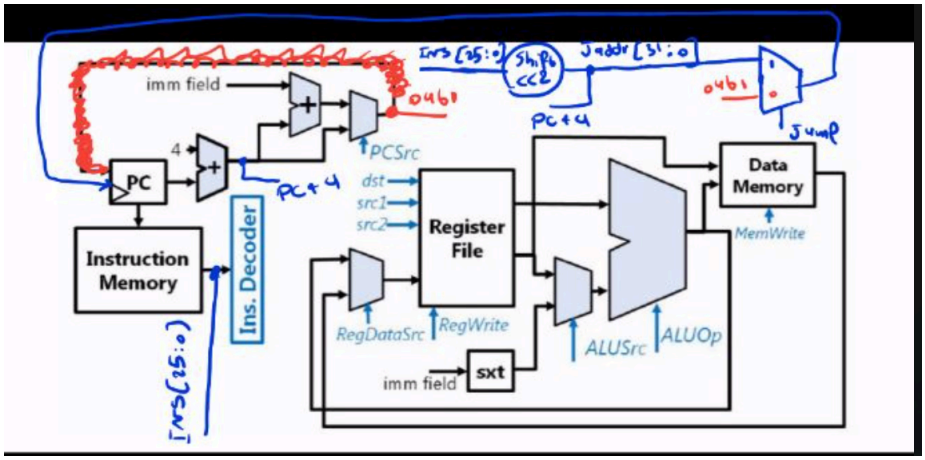
# سوال ۳

با توجه به این که ساختار دستورات J type به صورت زیر می باشد :

| J-format | op | target address |
|---|---|---|

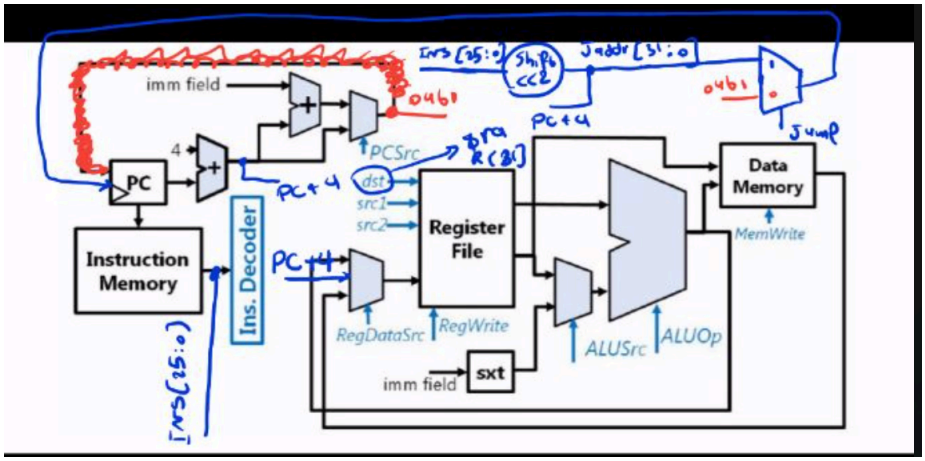
## jar

برای این که این دستور انجام شود باید یک واحد شیفت دهنده که مقدار موجود در Instruction[25:0] را دو بیت به سمت چپ شیفت می دهند . در مر حله بعد آن را با مقدار PC+4 که خروجی اولین adder می باشد ترکیب می کنیم به این صورت که خروجی واحد شیفت دهنده را در ۲۸ بیت پایینی PC قرار می دهیم و به این ترتیب آدرس پرش به دست می آید .
در مرحله بعد باید یک سیگنال کنترلی به اسم Jump اضافه کنیم .
در مرحله بعد باید یک مالتی پلکسر ۲ به ۱ قرار دهیم که سیگنال select آن همان سیگنال Jump میباشد و ورودی صفر آن ، خروجی مالتی پلکسری می باشد که سیگنال کنترلی PCSrc دارد . و ورودی یکم آن مقدار PC $+ 4 + (Instruction[25:0]«2)$ می باشد . بنابراین شکل نهایتا به صورت زیر خواهد بود :



### jal

برای این دستور باید علاوه بر تغییرات بخش قبل ، مالتی پلکسری که سیگنال select آن RegDataSrc می باشد را از ۲ به یک ، به یک مالتی پلکسر ۳ به یک تغییر دهیم و در ورودی دوم آن مقدار PC+4 را که خروجی adder اول می باشد را قرار دهیم و به سیگنال dst هم آدرس رجیستر مورد نظر برای کپی کردن را بدهیم ($ra) بنابراین نهایتا شکل به صورت زیر در می آید :




---

## سوال ۴

### الف

این data path مربوط به یک دستور R type میباشد .

### ب

تاخیر این دستور برابر است با :

$$\text{R type}_{delay} = \text{Instruction memory} + \text{Registers} + \text{ALU} + \text{Registers} = 2 + 1 + 2 + 1 = 6\text{ns}$$

### ج

با توجه به این که معادلات زمانی به صورت زیر می باشد :

$$\text{Minimum instruction delay} = \text{d}_{min} + \text{d}_{ff} > \text{hold\_time}$$
$$\text{Max instruction delay} + \text{setup\_time} = \text{d}_{max} + \text{d}_{ff} + \text{setup\_time} < \text{clock}$$

اما برای سادگی می توان به جای $\text{d}_{min} + \text{d}_{ff}$ مقدار مینیمم تاخیر دستورات را درنظر گرفت . برای به دست آوردن این مقدار تاخیر تمام دستورات را به دست می آوریم :

$$\text{delay}_{beq} = 2 + 1 + 2 = 5\text{ns}$$
$$\text{delay}_{sw} = 2 + 1 + 2 + 2 = 7\text{ns}$$
$$\text{delay}_{lw} = 2 + 1 + 2 + 2 + 1 = 8\text{ns}$$

بنابراین مینیمم تاخیر دستورات برابر با 5ns می باشد بنابراین داریم

$$hold\ \_\ time < 5ns$$

هم چنین باز برای سادگی می توان $d_{max} + d_{ff}$ را ماکسیمم تاخیر دستورات در نظر گرفت که برابر است با 8ns بنابراین برای کلاک داریم :

$$8ns + 0.3\ ns = 8.3ns$$

بنابراین کران پایین برای کلاک برابر است با 8.3ns

## سوال ۵

### الف

برای این بخش باید توجه کرد که اگر این اشکال به وجود بیاید تنها دستورات beq و sw به درستی کار می کنند زیرا در دستور beq نه از واحدهای data memeory استفاده می کنیم و نه از واحد Registers بنابراین اشکال به وجود آمده هیچ ایرادی در اجرای این دستور به وجود نمی آورد .
برای دستور sw چون آدرس رجیستر مقصد در بیت ها ۱۶ تا ۲۰ می باشد با همیشه صفر بودن سیگنال RegDst هیچ اشکالی در اجرای این دستور به وجود نمی آید .

### ب

با توجه به این که میانگین اجرای دستورات برای این پردازنده به شکل زیر می باشد :

$$0.48 * 6ns + 0.22 * 8ns + 0.11 * 7ns + 0.19 * 5ns = 6.36ns$$

بنابراین اگر بتوانیم سیگنال RegDst را درست کنیم حجم بیشتری از دستورات که شامل دستورات Arithmetic میباشد را می توان پوشش داد که مقرون به صرفه تر است .

### ج

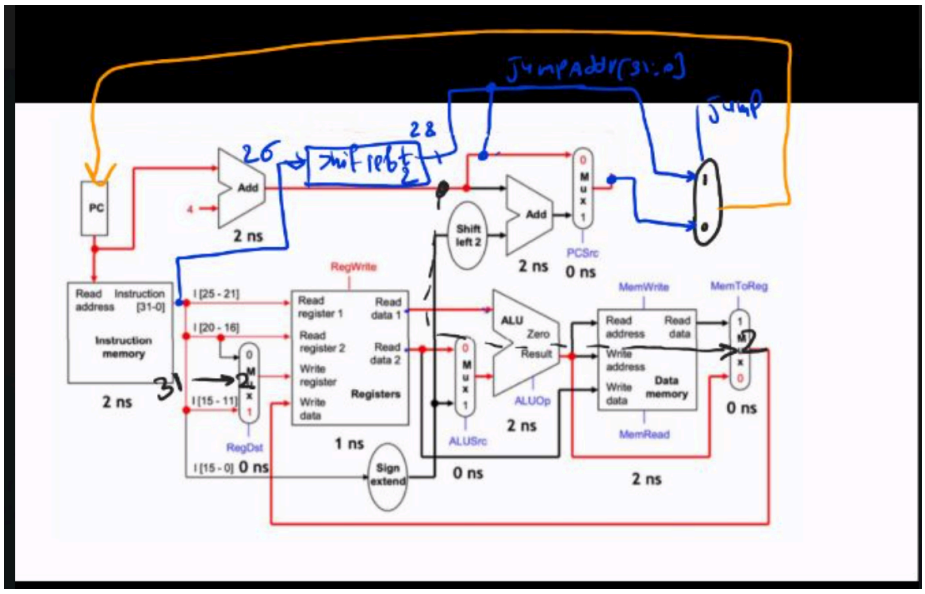
با توجه به این که فقط دو دستورات R type و دستور lw نیاز به سیگنال MemtoReg دارند و با توجه به مقدار سیگنال های کنترلی برای دستورات گفته شده که به صورت زیر می باشد

| Instruction | RegDst | ALUSrc | Memto-Reg | Reg Write | Mem Read | Mem Write | Branch | ALUOp1 | ALUp0 |
|---|---|---|---|---|---|---|---|---|---|
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

می توان دید که اگر مقدار دو سیگنال RegWrite و MemRead را با یکدیگر and کنیم و مقدار سیگنال MemtoReg را از آن بگیریم می توانیم این اشکال را بر طرف کرد .
انتخاب این دو سیگنال به این دلیل است که اگر سیگنال RegWrite داشته باشیم و نوشتن در حافظه نداشته باشیم یعنی دستور از نوع R type است بنابراین باید مقدار سیگنال MemtoReg برابر با صفر باشد که برابر حاصل and این دو سیگنال است . همچنین برای دستور این قانون صدق می کند .

## سوال ۶

باید مالتی پلکسری که سیگنال select آن MemtoReg و مالتی پلکسری که سیگنال select آن RegDst می باشد تبدیل به ماکس های ۳ به یک کنیم که برای ماکس اولی در ورودی دوم باید مقدار PC+4 و برای ماکس دوم در ورودی دم آدرس رجیستر مقصد را می دهیم .
همچنین باید یک سیگنال کنترلی به نام Jump اضافه کنیم و هم چنین باید یک واحد شیفت دهنده به چپ برای محاسبه ۲۸ بیت پایین PC هم باید اضافه کنیم و با توجه به تغییرات بالا شکل نهایی به صورت زیر است :




که مقدار سیگنال های کنترلی به شرح زیر می باشد :

| *RegWrite* | *RegDst* | *Branch* | *MemRead* | *MemtoReg* | *MemWrite* | *ALUSrc* | *Jump* |
|---|---|---|---|---|---|---|---|
| 1 | 10 | X | 0 | 10 | 0 | X | 1 |

## سوال عملی

در طراحی این مدار از واحد هایی برای شیفت دادن و همچنین sign extend کردن استفاده شده که جزئیات آن در طراحی آمده است .

**به دلیل این که در صورت سوال گفته نشده که مقدار اولیه PC را چند باید در نظر گرفت ، این مقدار صفر در نظر گرفته شده است .**