به نام خداوند بخشنده مهربان

معماری کامپیوتر

دانشکده مهندسی کامپیوتر

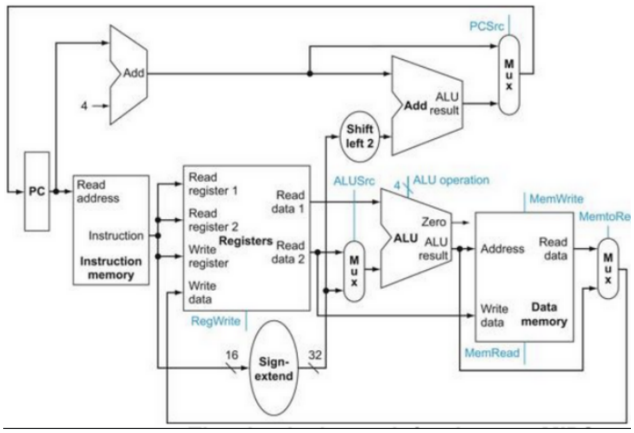
دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶

تمرین چهارم

سوال ۱

۱. جدول زیر که نشان‌دهنده‌ی مقدار سیگنال‌های کنترلی در طول اجرای دستورات است را با توجه به مسیر داده [۱] معماری پردازنده MIPS که در کلاس طراحی شد(شکل زیر)، تکمیل نمایید. دقت کنید که اگر سیگنال کنترلی MUX برابر ۰ باشد ورودی پایین آن انتخاب می‌شود و در صورتی که برابر ۱ باشد ورودی بالای آن انتخاب می‌شود.



Instruction	PCSrc	ALUSrc	MemWrite	MemToReg
`mult $t0, $t1`				
`bne $t0, $t1, 10`				
`lw $t0, 0($t1)`				
`sw $t0, 0($t1)`				

Instruction	PCSrc	ALUSrc	MemWrite	MemToReg
mult $t0, $t1	1	1	0	0
bne $t0, $t1, 10	$Zero_{ALU}$	1	0	don't care
lw $t0, 0($t1)	1	0	0	1
sw $t0, 0($t1)	1	0	1	don't care

سوال ۲

۲. طراحی مسیر داده برای پردازنده MIPS را در نظر بگیرید(شکل سوال ۱). می‌خواهیم با ایجاد تغییراتی، دو دستور زیر را به دستورات قابل پشتیبانی توسط این مسیر داده اضافه کنیم. ابتدا با ذکر دلیل نوع دستور مناسب برای هر کدام را مشخص کنید و سپس تغییرات لازم برای پشتیبانی از هر دستور را مشخص نمایید.

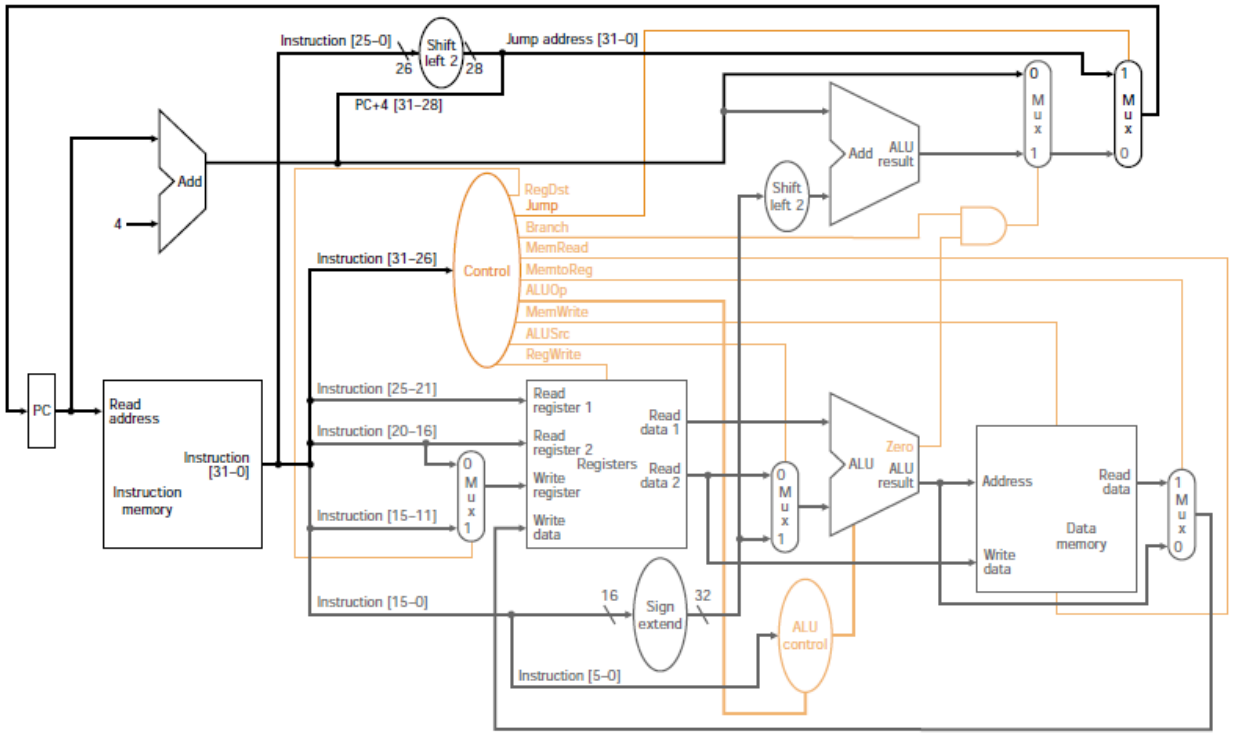
۱. `getpc $rd`: این دستورالعمل مقدار PC را در ثبات `rd` می‌ریزد.

۲. `jof $rt, imm($rs)`: این دستورالعمل ابتدا مقدار PC را در ثبات `rt` ذخیره می‌کند و سپس PC را برابر `imm + $rs` قرار می‌دهد.

۱. این دستور می تواند از نوع R type باشد، زیرا تنها چیزی که نیاز دارد گرفتن ۵ بیت $rd است که با این فرمت امکان پذیر است. برای انجام این کار از PC یک سیم به ماکسی که جلوی ورودی Write data قرار دارد وارد می کنیم و ورودی دیگر این ماکس را همان سیمی که از قبل به آن می آمده قرار می دهیم و سلکت آن سیگنال کنترلی ما می شود.

۲. برای این دستور از فرمت I type استفاده می کنیم. برای پشتیبانی این دستور نیز اول به بخش طراحی شده در قسمت ۱ نیاز داریم، سپس از خروجی ALU یک سیم و از خروجی ماکس PCSrc یک سیم دیگر به یک ماکس ۲ به ۱ وصل می کنیم و خروجی ماکس جدید را به ورودی PC وصل می کنیم. سلکت ماکس نیز سیگنال کنترلی ما می شود.

برای سوال های ۳ تا ۶ از شماتیک زیر به عنوان Datapath پردازنده ی MIPS استفاده شده است:



سوال ۳

۳. جدول تأخیرهای زیر را برای بخش‌های مختلف پردازنده MIPS در نظر بگیرید:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Control
200ps	70ps	20ps	90ps	90ps	250ps	40ps

فرض کنید ۴ درصد دستورات از نوع پرش، ۱۳ درصد Beq و ۱۴ درصد از نوع Load و ۱۷ درصد از نوع Store و بقیه از نوع R باشند. در صورتی که با تغییراتی بتوان تأخیر ALU را به ۷۵ و تأخیر D-Mem را به ۲۲۰ کاهش داد و تأخیر کنترل به ۵۰ برسد، میزان Speed Up چقدر خواهد بود؟

باید ابتدا زمان متوسط هر دستور را برای این برنامه حساب کنیم، سپس با اعمال تغییرات جدید، speed up را مشخص کنیم. برای اینکار ابتدا تاخیر هر دستور را محاسبه می کنیم، سپس میانگین وزن دار می گیریم. برای حساب کردن هر کدام از این تاخیر ها نیاز است تا مسیر داده ها را دنبال کنیم. max هایی که تعریف شده اند، هرکدام برای یک بخشی از مدار است که نیاز دارد ۲ یا تعداد بیشتری پردازش قبلش انجام شود تا بتواند پردازش خود را کامل کند، که در این مثال، عمدتا منظور mux ها و ALU ها است یا حالتی که PC به طور مستقلانه از کل مدار آپدیت می شود.

$$D_{\text{jump}} = \max_{\substack{\text{برای}\\ \text{ماکس پرش}}} \left(\text{ورودی که می‌خواهیم, سیگنال کنترلی}\right) + \text{مسیر تا PC}$$

$$D_{\text{jump}} = \max_{\substack{\text{برای}\\ \text{ماکس پرش}}} \left(D_{\text{Add}}(\text{PC جمع}), D_{\text{I-Mem}}(\text{jump خواندن آدرس}), D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}) \right) + D_{\text{Mux}}(\text{Mux تعیین پرش})$$

$$D_{\text{Beq}} = \max_{\text{برای ماکس برنچ}} \left(\text{سیگنال کنترلی, ورودی دوم, ورودی اول}\right) + \text{مسیر تا PC}$$

$$\text{ورودی دوم} = \max_{\text{برای ALU برنچ}} \left(\text{ورودی دوم, ورودی اول}\right), \quad \text{سیگنال کنترلی} = \max_{\text{برنچ AND}} \left(\text{ورودی دوم, سیگنال کنترلی}\right)$$

$$\text{AND ورودی دوم} = \text{سیگنال zero} = \max_{\text{اصلی ALU}} \left(\text{ورودی دوم, ورودی اول, سیگنال کنترلی}\right), \quad \text{داده از رجیستر} = \text{ورودی اول alu}$$

$$\text{ALU ورودی دوم} = \max_{\text{ماکس ALUSrc}} \left(\text{ورودی که میخواهیم, سیگنال کنترلی}\right)$$

$$D_{\text{Beq}} = \max_{\text{برای ماکس برنچ}} \left\{ D_{\text{Add}}(\text{PC جمع}), \max_{\text{برای ALU برنچ}} \left(D_{\text{Add}}(\text{PC جمع}), D_{\text{I-Mem}}(\text{خواندن Imm. دستور}) \right) + D_{\text{Add}}(\text{انجام جمع}), \right.$$
$$\max_{\text{برنچ AND}} \left[D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \max_{\text{اصلی ALU}} \left(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \right.\right.$$
$$D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Regs}}(\text{خواندن رجیستر ها}), \max_{\text{ماکس ALUSrc}} \left(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \right.$$
$$\left.\left.\left. D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Regs}}(\text{خواندن رجیستر ها})) + D_{\text{Mux}}(\text{آماده شدن خروجی}) \right) + D_{\text{ALU}}(\text{انجام محاسبه}) \right] \right\}$$
$$+ ۲D_{\text{Mux}}(\text{در مسیر برنچ})$$

$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \left(\text{PC مسیر, اجرای خود دستور}\right),$$
$$\text{PC مسیر} = \max_{\text{ماکس برنچ}} \left(\text{سیگنال کنترل, ورودی}\right) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ})$$
$$\text{اجرای خود دستور} = \max_{\text{نوشتن در RF}} \left(\text{داده, آدرس مقصد}\right),$$
$$\text{داده} = \text{مسیری از حافظه که خود از طریق آدرس حاصل از ALU مشخص می شود}$$

$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \left\{ \max_{\text{ماکس برنچ}} \left(D_{\text{Add}}(\text{PC جمع}), D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}) \right) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ}), \right.$$
$$\max_{\text{نوشتن در RF}} \left[D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \max_{\text{ماکس MemtoReg}} \left\{ D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \right.\right.$$
$$D_{\text{D-Mem}}(\text{خواندن داده}) + \max_{\text{اصلی ALU}} \left[D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \right.$$
$$\max_{\text{ماکس ALUSrc}} \left(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), D_{\text{I-Mem}}(\text{خواندن دستور}) \right) + D_{\text{Mux}}(\text{تاخیر ماکس}),$$
$$\left.\left. D_{\text{Regs}}(\text{خواندن رجیستر}) + D_{\text{I-Mem}}(\text{خواندن دستور}) \right] + D_{\text{ALU}}(\text{انجام محاسبه}) \right\} + D_{\text{Mux}}(\text{تاخیر ماکس}),$$
$$\left. \max_{\text{ماکس RegDst}} \left(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), D_{\text{I-Mem}}(\text{خواندن دستور}) \right) + D_{\text{Mux}}(\text{تاخیر ماکس}) \right]$$
$$\left. + D_{\text{Regs}}(\text{نوشتن رجیستر ها}) \right\}$$

$$D_{\text{store}} = \max_{\text{PC یا دستور}} \Big(\text{PC مسیر}, \text{اجرای خود دستور}\Big),$$

$$\text{PC مسیر} = \max_{\text{ماکس برنچ}} \Big(\text{سیگنال کنترل}, \text{ورودی}\Big) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ})$$

$$\text{اجرای خود دستور} = \max_{\text{حافظه}} \Big(\text{سیگنال کنترلی}, \text{داده}, \text{آدرس مقصد}\Big),$$

$$\text{داده} = \text{از RF گرفته می شود},$$

$$\text{آدرس} = \text{خروجی ALU}$$

$$D_{\text{store}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(D_{\text{Add}}(\text{PC جمع}), D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور})\Big) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ}),$$

$$\max_{\text{بلوک حافظه}} \Big[D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Regs}}(\text{خواندن رجیستر}), \max_{\text{ALU اصلی}} \Big(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}),$$

$$\max_{\text{ماکس ALUSrc}} \Big(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), D_{\text{I-Mem}}(\text{خواندن دستور})\Big) + D_{\text{Mux}}(\text{تاخیر ماکس}),$$

$$D_{\text{Regs}}(\text{خواندن رجیستر}) + D_{\text{I-Mem}}(\text{خواندن دستور})\Big) + D_{\text{ALU}}(\text{انجام محاسبه}),$$

$$D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور})\Big] + D_{\text{D-Mem}}(\text{نوشتن در حافظه})\Bigg\}$$

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \Big(\text{PC مسیر}, \text{اجرای خود دستور}\Big),$$

$$\text{PC مسیر} = \max_{\text{ماکس برنچ}} \Big(\text{سیگنال کنترل}, \text{ورودی}\Big) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ})$$

$$\text{اجرای خود دستور} = \max_{\text{نوشتن در RF}} \Big(\text{سیگنال کنترلی}, \text{داده}, \text{آدرس مقصد}\Big),$$

$$\text{داده} = \text{داده توسط ALU اصلی آماده می شود},$$

$$\text{آدرس} = \text{از دستور گرفته می شود}$$

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(D_{\text{Add}}(\text{PC جمع}), D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور})\Big) + ۲D_{\text{Mux}}(\text{ماکس های پرش و برنچ}),$$

$$\max_{\text{نوشتن در RF}} \Big[D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), \max_{\text{ALU اصلی}} \Big(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}),$$

$$\max_{\text{ماکس ALUSrc}} \Big(D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور}), D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Regs}}(\text{خواندن رجیستر})\Big) + D_{\text{Mux}}(\text{تاخیر ماکس}),$$

$$D_{\text{Regs}}(\text{خواندن رجیستر}) + D_{\text{I-Mem}}(\text{خواندن دستور})\Big) + D_{\text{ALU}}(\text{انجام محاسبه}) + D_{\text{Mux}}(\text{ماکس MemtoReg}),$$

$$D_{\text{I-Mem}}(\text{خواندن دستور}) + D_{\text{Control}}(\text{پردازش دستور})\Big] + D_{\text{Regs}}(\text{نوشتن رجیستر ها})\Bigg\}$$

در حالت اولیه مقدار تاخیرهای هر دستور به صورت زیر است:

$$D_{\text{jump}} = \max_{\substack{\text{برای}\\ \text{ماکس پرش}}} \Big(۷۰, ۲۰۰, ۲۰۰ + ۴۰\Big) + ۲۰ = ۲۶۰ps$$

$$D_{\text{Beq}} = \max_{\text{برای ماکس برنچ}} \Bigg\{ ۷۰, \max_{\text{برنچ ALU برای}} \Big(۷۰, ۲۰۰\Big) + ۷۰, \max_{\text{برنچ AND}} \Big[۲۰۰ + ۴۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۴۰, ۲۰۰ + ۹۰, \max_{\text{ماکس ALUSrc}} \big(۲۰۰ + ۴۰, ۲۰۰ + ۹۰\big) + ۲۰\Big) + ۹۰\Big] \Bigg\} + ۲ * ۲۰ = ۴۴۰ps$$

$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۴۰\Big) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \Big[۲۰۰ + ۴۰, \max_{\text{ماکس MemtoReg}} \Big\{ ۲۰۰ + ۴۰, ۲۵۰ + \max_{\text{اصلی ALU}} \Big[۲۰۰ + ۴۰, \max_{\text{ماکس ALUSrc}} \Big(۲۰۰ + ۴۰, ۲۰۰\Big) + ۲۰, ۹۰ + ۲۰۰\Big] + ۹۰ \Big\} + ۲۰, \max_{\text{ماکس RegDst}} \Big(۲۰۰ + ۴۰, ۲۰۰\Big) + ۲۰\Big] + ۹۰ \Bigg\} = ۷۴۰ps$$

$$D_{\text{store}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۴۰\Big) + ۲ * ۲۰, \max_{\text{بلوک حافظه}} \Big[۲۰۰ + ۹۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۴۰, \max_{\text{ماکس ALUSrc}} \Big(۲۰۰ + ۴۰, ۲۰۰\Big) + ۲۰, ۹۰ + ۲۰۰\Big) + ۹۰, ۲۰۰ + ۴۰\Big] + ۲۵۰ \Bigg\} = ۶۳۰ps$$

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۴۰\Big) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \Big[۲۰۰ + ۹۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۴۰, \max_{\text{ماکس ALUSrc}} \big(۲۰۰ + ۴۰, ۲۰۰ + ۹۰\big) + ۲۰, ۹۰ + ۲۰۰\Big) + ۹۰ + ۲۰, ۲۰۰ + ۴۰\Big] + ۹۰ \Bigg\} = ۵۱۰ps$$

حال متوسط تاخیر به ازای دستور در این برنامه را حساب می کنیم:

$$\frac{۴}{۱۰۰} * ۲۶۰ + \frac{۱۳}{۱۰۰} * ۴۴۰ + \frac{۱۴}{۱۰۰} * ۷۴۰ + \frac{۱۷}{۱۰۰} * ۶۳۰ + \frac{۵۲}{۱۰۰} * ۵۱۰ = ۵۴۳/۵ \text{ ps per instruction}$$

حال این عدد را بعد اعمال تغییرات محاسبه می کنیم:

$$D_{\text{jump}} = \max_{\substack{\text{برای}\\ \text{ماکس پرش}}} \Big(۷۰, ۲۰۰, ۲۰۰ + ۵۰\Big) + ۲۰ = ۲۷۰ps$$

$$D_{\text{Beq}} = \max_{\text{برای ماکس برنچ}} \Bigg\{۷۰, \max_{\text{برنچ ALU برای}} \Big(۷۰, ۲۰۰\Big) + ۷۰, \max_{\text{برنچ AND}} \Big[۲۰۰ + ۵۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۵۰, ۲۰۰ + ۹۰, \max_{\text{ماکس ALUSrc}} \big(۲۰۰ + ۵۰,$$
$$۲۰۰ + ۹۰\big) + ۲۰\Big) + ۷۵\Big]\Bigg\} + ۲ * ۲۰ = ۴۲۵ps$$

$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۵۰\Big) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \Big[۲۰۰ + ۵۰, \max_{\text{ماکس MemtoReg}} \Big\{۲۰۰ + ۵۰, ۲۲۰+$$
$$\max_{\text{اصلی ALU}} \Big[۲۰۰ + ۵۰, \max_{\text{ماکس ALUSrc}} \Big(۲۰۰ + ۵۰, ۲۰۰\Big) + ۲۰, ۹۰ + ۲۰۰\Big] + ۷۵\Big\} + ۲۰,$$
$$\max_{\text{ماکس RegDst}} \Big(۲۰۰ + ۵۰, ۲۰۰\Big) + ۲۰\Big] + ۹۰\Bigg\} = ۶۹۵ps$$

$$D_{\text{store}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۵۰\Big) + ۲ * ۲۰, \max_{\text{بلوک حافظه}} \Big[۲۰۰ + ۹۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۵۰, \max_{\text{ماکس ALUSrc}} \Big(۲۰۰ + ۵۰, ۲۰۰\Big) + ۲۰,$$
$$۹۰ + ۲۰۰\Big) + ۷۵, ۲۰۰ + ۵۰\Big] + ۲۲۰\Bigg\} = ۵۸۵ps$$

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \Bigg\{ \max_{\text{ماکس برنچ}} \Big(۷۰, ۲۰۰ + ۵۰\Big) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \Big[۲۰۰ + ۹۰, \max_{\text{اصلی ALU}} \Big(۲۰۰ + ۵۰,$$
$$\max_{\text{ماکس ALUSrc}} \Big(۲۰۰ + ۵۰, ۲۰۰ + ۹۰\Big) + ۲۰,$$
$$۹۰ + ۲۰۰\Big) + ۷۵ + ۲۰, ۲۰۰ + ۵۰\Big] + ۹۰\Bigg\} = ۴۹۵ps$$

حال متوسط تاخیر جدید را حساب می کنیم:

$$\frac{۴}{۱۰۰} * ۲۷۰ + \frac{۱۳}{۱۰۰} * ۴۲۵ + \frac{۱۴}{۱۰۰} * ۶۹۵ + \frac{۱۷}{۱۰۰} * ۵۸۵ + \frac{۵۲}{۱۰۰} * ۴۹۵ = ۵۲۰/۲ \text{ ps per instruction}$$

میزان speed up برنامه نیز به صورت زیر محاسبه می شود.

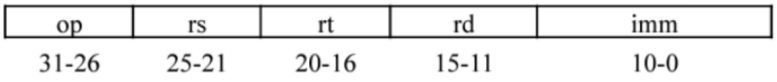
$$\text{speed up} = \frac{\text{runtime}_{\text{old}}}{\text{runtime}_{\text{new}}} = \frac{۵۴۳/۵ * C}{۵۲۰/۲ * C} \approx ۱/۰۴۴۸$$

سوال ۴

۴. با استفاده از دستورات پردازنده MIPS توانسته‌ایم CPU جدیدی را طراحی کنیم که به شکل single-cycle کار می‌کند و دستورات آن مطابق زیر است:

Instruction	Operation
`lw_add rd, (rs), rt`	rd = Mem[Reg[rs]] + Reg[rt]
`addi_st (rs), rs, imm`	Mem[Reg[rs]] = Reg[rs] + imm
`sll_add rd, rs, rt, imm`	rd = (Reg[rs] « imm) + Reg[rt]

و تمامی دستورات این پردازنده جدید ما در قالبی ۳۲ بیتی با شکل زیر در حافظه ذخیره می‌شوند:



حال تصور کنید تأخیر بخش‌های مختلف پردازنده جدید به شکل جدول زیر باشد:

Unit	Latency
Register File	2.5
Instruction Memory	4
Data Memory	6
ALU	5.5

حداقل زمان مورد نیاز برای انجام هر یک از دستورات پردازنده جدید را محاسبه کنید. (فرض کنید که می‌توان از دو ALU در طراحی استفاده کرد)

حداقل زمان مورد نیاز برای انجام این دستورات، همان زمان مسیر بحرانی در هر یک از این دستورات است. مسیر بحرانی هر یک از دستورات زیر به این صورت محاسبه می شود:

- lw_add

$$L_{\text{critical path}} = L_{\text{IM}}(\text{fetching instruction}) + L_{\text{RF}}(\text{reading registers}) + L_{\text{DM}}(\text{reading data memory}) + L_{\text{ALU}}(\text{calculating the sum}) + L_{\text{RF}}(\text{writing the result}) = 20.5$$

- addi_st

$$L_{\text{critical path}} = L_{\text{IM}}(\text{fetching instruction}) + L_{\text{RF}}(\text{reading registers}) + L_{\text{ALU}}(\text{calculating the sum}) + L_{\text{DM}}(\text{storing the result}) = 18$$

- sll_add

$$L_{\text{critical path}} = L_{\text{IM}}(\text{fetching instruction}) + L_{\text{RF}}(\text{reading registers}) + L_{\text{ALU}}(\text{calculating the shift}) + L_{\text{ALU}}(\text{calculating the sum}) + L_{\text{RF}}(\text{write back}) = 20$$

سوال ۵

۵. مسیرداده Single-Cycle پردازنده MIPS را در نظر بگیرید. با توجه به دو دستور زیر به سوالات پاسخ دهید:

1. AND Rd,Rs,Rt
2. SW Rt, Offset(Rs)

آ) مقادیر سیگنال‌های کنترلی برای هر یک از این دو دستور را مشخص کنید.

ب) از کدام یک از منابع (بلوک‌های موجود در مدار) در این دستورات استفاده می‌شود.

ج) کدام یک از منابع، خروجی مشخصی تولید می‌کنند ولی از آن‌ها در این دستورات استفاده نمی‌شود؟

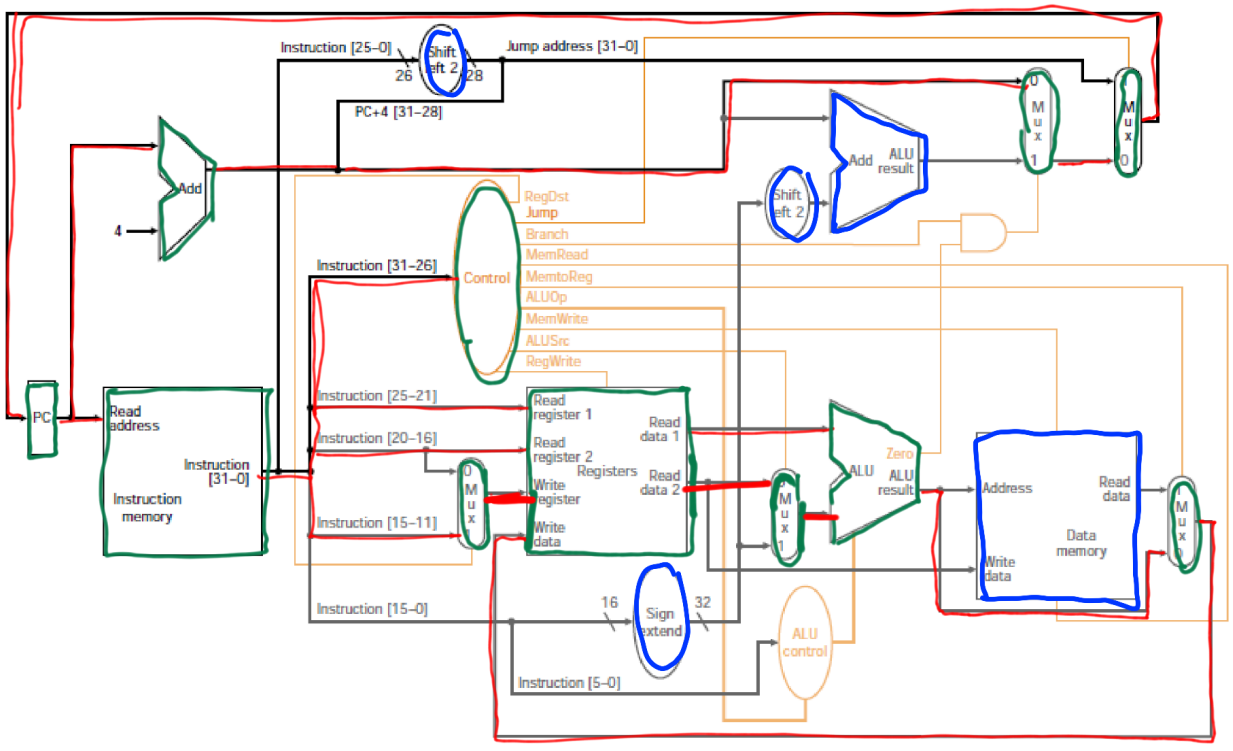
د) فرض کنید دو سیستم مختلف با تأخیرهای مشخص شده در مدار مطابق جدول زیر را داریم. مسیر بحرانی[۲] برای دستورات AND و load را مشخص کنید.

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Control
a	200ps	70ps	20ps	90ps	90ps	250ps	40ps
b	750ps	200ps	50ps	250ps	300ps	500ps	300ps

(آ)

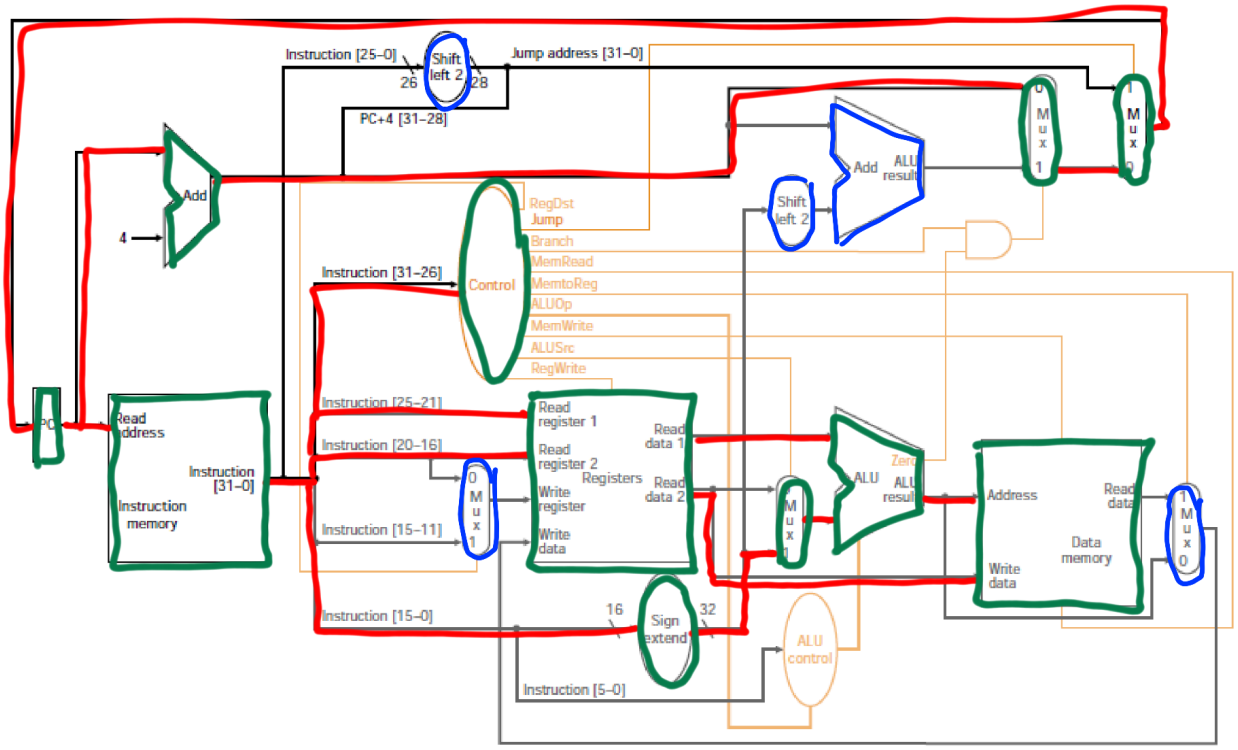
Instruction	RegDst	Jump	Branch	MemRead	MemtoReg	MemWrite	ALUSrc	RegWrite
AND Rd,Rd,Rt	1	0	0	0	0	0	0	1
SW Rt, Offset(Rs)	d	0	0	0	d	1	1	0

(ب) (ج)



۱.

در شکل بالا می توانید مسیر طی شونده توسط دستور AND را مشاهده کنید، در این تصویر، بلوک هایی که با رنگ سبز مشخص شده اند به این معنی است که از آنها در این دستور استفاده می شود. و نیز بلوک های آبی، بلوک هایی هستند که خروجی تولید می کنند ولی از آن خروجی بهره ای برده نمی شود.



.۲

در شکل بالا می توانید مسیر طی شونده توسط دستور AND را مشاهده کنید، در این تصویر، بلوک هایی که با رنگ سبز مشخص شده اند به این معنی است که از آنها در این دستور استفاده می شود. و نیز بلوک های آبی، بلوک هایی هستند که خروجی تولید می کنند ولی از آن خروجی بهره ای برده نمی شود.

(د) ۱. از همان فرمولی که برای سوال ۳ برای ماکسیمم تاخیر به دست آورده ایم استفاده می کنیم تا مسیر بحرانی هرکدام را مشخص کنیم.

i

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \left\{ \max_{\text{ماکس برنچ}} \left(۷۰, ۲۰۰ + ۴۰\right) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \left[۲۰۰ + ۹۰, \max_{\text{ALU اصلی}} \left(۲۰۰ + ۴۰, \right.\right.\right.$$

$$\left.\left.\left. \max_{\text{ماکس ALUSrc}} \left(۲۰۰ + ۴۰, ۲۰۰ + ۹۰\right) + ۲۰, ۹۰ + ۲۰۰ \right) + ۹۰ + ۲۰, ۲۰۰ + ۴۰ \right] + ۹۰ \right\} = ۵۱۰ps$$

مسیر بحرانی: مسیری است که با خوندن دستور(۲۰۰) شروع میشه، سپس رجیسترش مشخص می شود،(۹۰) بعد از آن از یک ماکس می گذرد(۲۰) سپس از ALU عبور می کند(۹۰) و پس از آن از ماکس ALUSrc می گذرد(۲۰) و نوشته می شود(۹۰).

ii

$$D_{\text{R-type}} = \max_{\text{PC یا دستور}} \left\{ \max_{\text{ماکس برنچ}} \left(۲۰۰, ۷۵۰ + ۳۰۰\right) + ۲ * ۵۰, \max_{\text{نوشتن در RF}} \left[۷۵۰ + ۳۰۰, \max_{\text{ALU اصلی}} \left(۷۵۰ + ۳۰۰, \right.\right.\right.$$

$$\left.\left.\left. \max_{\text{ماکس ALUSrc}} \left(۷۵۰ + ۳۰۰, ۷۵۰ + ۳۰۰\right) + ۵۰, ۳۰۰ + ۷۵۰ \right) + ۷۵ + ۵۰, ۷۵۰ + ۳۰۰ \right] + ۳۰۰ \right\} = ۱۷۰۰ps$$

مسیر بحرانی: مسیری است که با خوندن دستور(۷۵۰) شروع میشه، سپس رجیسترش مشخص می شود،(۳۰۰) بعد از آن از یک ماکس می گذرد(۵۰) سپس از ALU عبور می کند(۲۵۰) و پس از آن از ماکس ALUSrc می گذرد(۵۰) و نوشته می شود(۳۰۰).

۲. i

$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \left\{ \max_{\text{ماکس برنچ}} \left(۷۰, ۲۰۰ + ۴۰\right) + ۲ * ۲۰, \max_{\text{نوشتن در RF}} \left[۲۰۰ + ۴۰, \max_{\text{ماکس MemtoReg}} \left\{ ۲۰۰ + ۴۰, ۲۵۰ + \right.\right.\right.$$

$$\max_{\text{ALU اصلی}} \left[۲۰۰ + ۴۰, \max_{\text{ماکس ALUSrc}} \left(۲۰۰ + ۴۰, ۲۰۰\right) + ۲۰, ۹۰ + ۲۰۰ \right] + ۹۰ \right\} + ۲۰,$$

$$\left.\left.\left. \max_{\text{ماکس RegDst}} \left(۲۰۰ + ۴۰, ۲۰۰\right) + ۲۰ \right] + ۹۰ \right\} = ۷۴۰ps$$

مسیر بحرانی آدرس مموری است: مسیری است که با خوندن دستور(۲۰۰) شروع میشه، سپس رجیسترش مشخص می شود،(۹۰) سپس از ALU عبور می کند(۹۰) و بعد دیتا از D-Mem خوانده می شود(۲۵۰) سپس از ماکس عبور می کند(۲۰) و در RF نوشته می شود(۹۰)

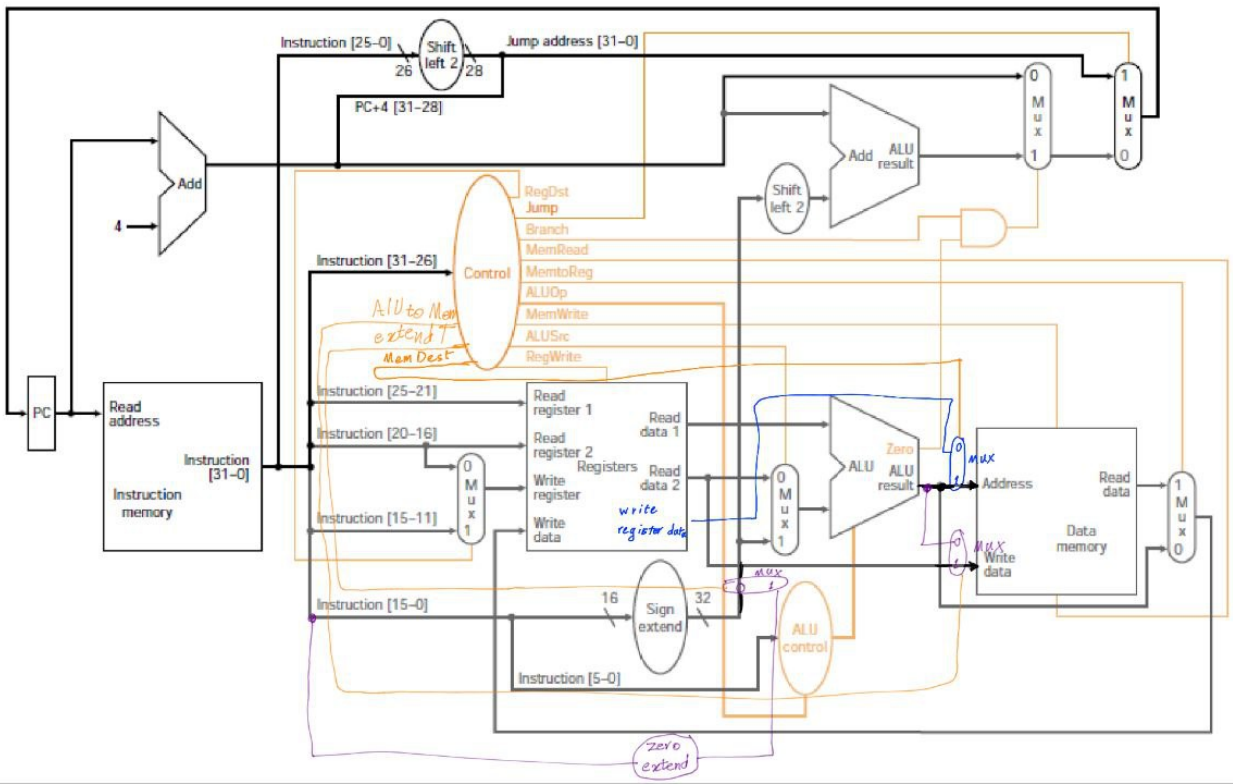
$$D_{\text{Load}} = \max_{\text{PC یا دستور}} \left\{ \max_{\text{ماکس برنچ}} \left(۲۰۰, ۷۵۰ + ۳۰۰ \right) + ۲ * ۵۰, \max_{\text{نوشتن در RF}} \left[۷۵۰ + ۳۰۰, \max_{\text{ماکس MemtoReg}} \left\{ ۷۵۰ + ۳۰۰, ۵۰۰ + \right. \right. \right.$$
$$\max_{\text{اصلی ALU}} \left[۷۵۰ + ۳۰۰, \max_{\text{ماکس ALUSrc}} \left(۷۵۰ + ۳۰۰, ۷۵۰ \right) + ۵۰, ۳۰۰ + ۷۵۰ \right] + ۲۵۰ \left. \right\} + ۵۰,$$
$$\left. \left. \max_{\text{ماکس RegDst}} \left(۷۵۰ + ۳۰۰, ۷۵۰ \right) + ۵۰ \right] + ۳۰۰ \right\} = ۲۲۰۰ps$$

مسیر بحرانی آدرس مموری است ولی دفعه ی قبل رجیستر آن زمان می برد ولی این بار آماده شدن offset طولانی است: مسیری است که با خوندن دستور(۷۵۰) شروع می شود. سپس پردازش می شود(۳۰۰) و سیگنال ALUSrc مشخص می شود. و از آن عبور می کنند(۵۰). سپس از ALU عبور می کند(۲۵۰) داده را از حافظه می خواند(۵۰۰) از ماکس عبور می دهد(۵۰) و در رجیستر می ریزد(۳۰۰)

سوال ۶

۶. مسیر داده و واحد کنترل پردازنده میپس را در نظر بگیرید، فرض کنید بخواهیم دو دستور زیر را اضافه کنیم، چه تغییراتی در شماتیک باید ایجاد کنیم؟مقدار سیگنال‌های خروجی از واحد کنترلی به جز ALUop را برای هر دستور مشخص کنید.

- Addm (rd),rs,rt : mem[rd] = rs+rt
- Xormi (rt),rs,i ; mem[rt] ← rs XOR zero_extend(i)



- برای اضافه کردن دستور Addm کافیست بخش آبی رنگ را به مدار اضافه کنیم. به RF یک خروجی جدید اضافه می کنیم که مقدار rd را نیز به نمایش بگذارد، سپس بر سر مسیر ورود آدرس به حافظه یک ماکس می گذاریم تا تشخیص دهد خروجی ALU آدرس است یا رجیستر .rd
- برای دستور Xormi نیز به تمام بخش های اضافه شده نیاز داریم. ابتدا نیاز است یک بلاک zero extend طراحی کنیم و سپس تصمیم بگیریم که Imm چگونه extend شود. سپس یک ماکس بر سر راه دیتای ورودی به حافظه می گذاریم تا تصمیم بگیرد که خروجی ALU وارد شود یا مقدار یک رجیستر.