تمرین ۵

سوال ۱

مسیر داده

تغییرات روی datapath به این شکل انجام شده که برای تهیه `rs1` و `rs2` نیاز به دو `clk` میباشد. سیگنال کنترلی `RegRead1:0` برای `enable` شدن نوشتن هر کدام از رجیسترهای مبدا بر روی register مربوطه تعبیه گردیده است. بیت 0 مربوط به `rs1` و بیت 1 مربوط به رجیستر `rs2` می‌باشد.

برای تعیین آدرس رجیسترها سیگنال `RegAddr1:0` در نظر گرفته شده تا بین `rs2` , `rs1` و `rd` انتخاب کند.

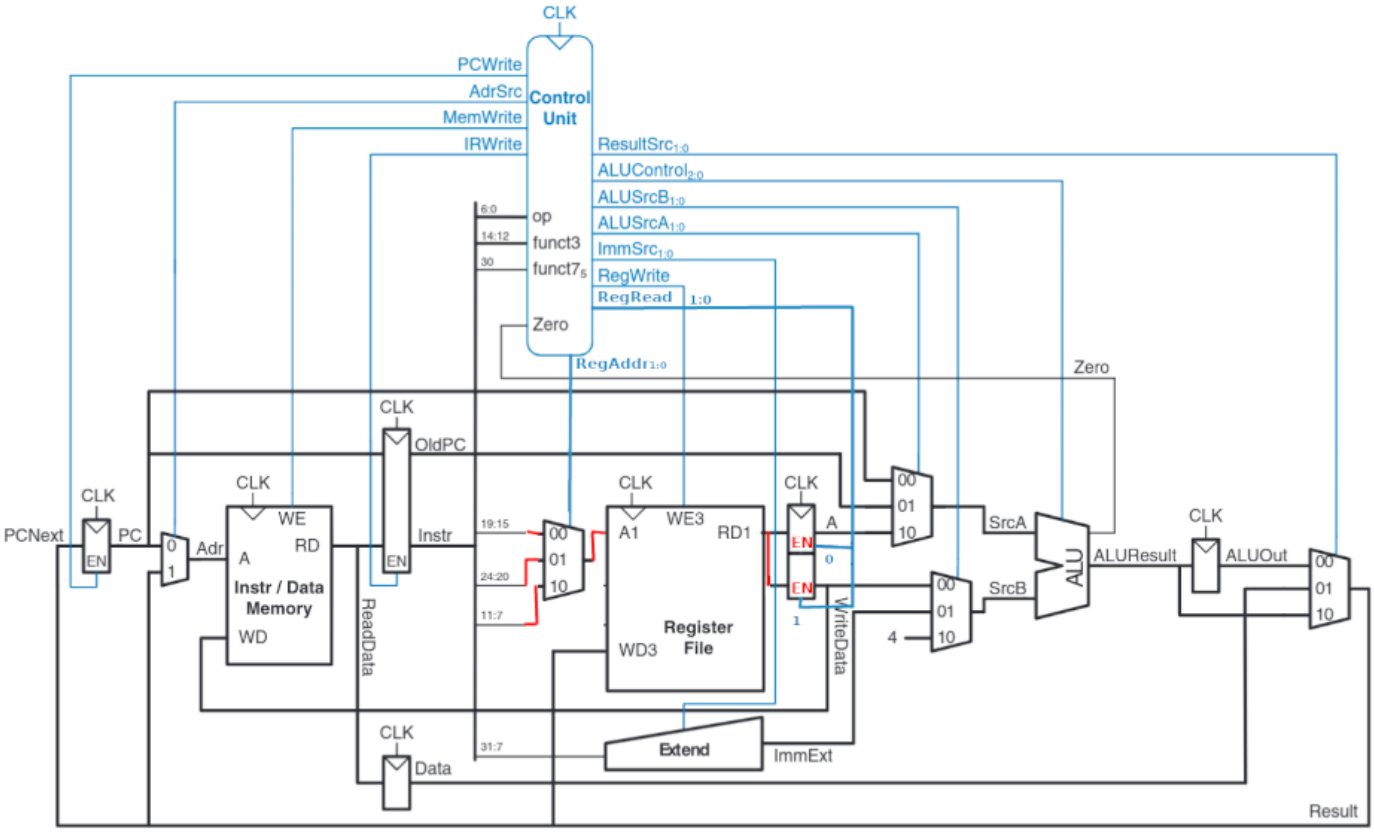
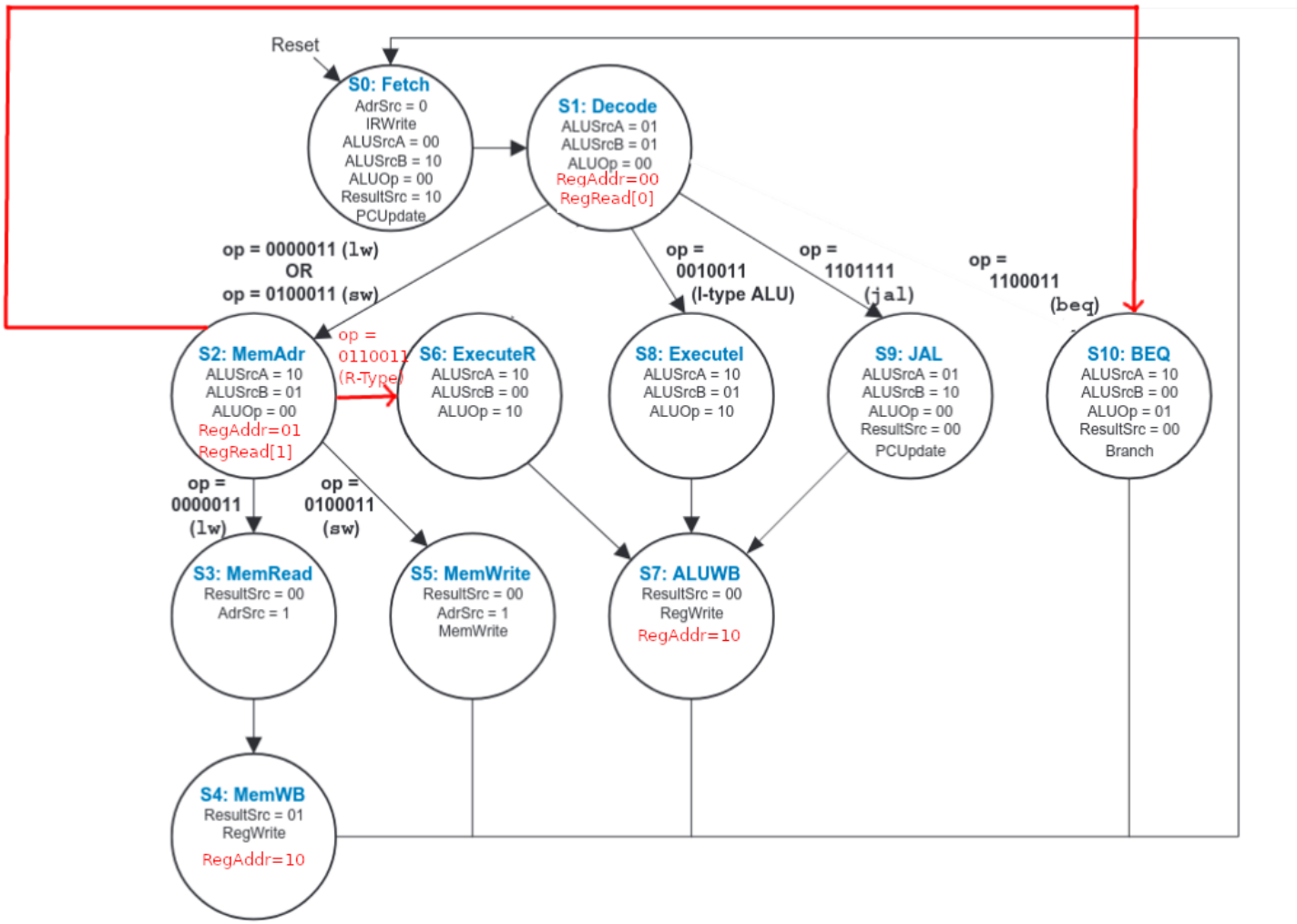


Figure 7.27 Complete multicycle processor

ماشین حالت

حالت جدیدی به ماشین حالت اضافه نشده‌است. اما دستوراتی `R-Tyte` و `Branch` هر کدام یک stage اضافه‌تر دارند. چرا که آماده‌سازی ورودی‌های ALU هنگامی که هر دو ورودی register باشند دو `clk` به طول می‌انجامد.



سوال ۲

در ۷ خط ابتدایی این قطعه کد، که شامل دستور `addi` و `sb` است که به ترتیب عمل `add immediate` و `store byte` را انجام میدهند.

که در انتهای این دستورات رجیستر فایل و حافظه ما بر اساس جدول زیر مقدار دهی شده است.

Register

Register	Value
t0 (x5)	0x00010010
t1 (x6)	0x00000007
t2 (x7)	0x00000000
t3 (x28)	0x00000001

Register	Value
t4 (x29)	0x00000000
t5 (x30)	0x00000001

Memory

Address	+0	+1	+2	+3
0x0001001c	00	00	00	00
0x00010018	00	00	00	00
0x00010014	00	00	00	00
0x00010010	00	01	00	00
0x0001000c	00	00	00	00

سپس وارد حلقه `loop` میشویم؛ در ابتدای هر بار اجرا شدن حلقه شرط برقراری شرط `t5 < t1` چک میشود و اگر این شرط برقرار بود به پایان برنامه یعنی، لیبل `end` میرویم و در غیر اینصورت دستورات حلقه اجرا میشوند.

در حلقه هر بار رجیستر `t5` یک واحد افزایش میابد که درواقع همان `counter` حلقه است.

سپس با اجرای دستور `add t4,t2,t3` ، مقدار `t2+t3` در رجیستر `t4` ذخیره میشود.

با اجرای خط بعدی، مقدار ذخیره شده در رجیستر `t0` با شمارنده حلقه که درواقع همان `t5` است جمع میشود و نتیجه در `t0` ذخیره میشود تا برای اجرای خط بعدی، آدرس حافطه مقصد را برای دستور `sb t4, 0 (t0)` به درستی داشته باشیم.

سپس مقدار `t4` در آدرس مقصد `t0` ذخیره میشود.

با اجرای دستور `mv t2,t3` ، مقدار ذخیره شده در رجیستر `t3` به رجیستر `t2` کپی میشود.

با اجرای دستور `mv t3,t4` ، مقدار ذخیره شده در رجیستر `t4` به رجیستر `t3` کپی میشود.

در خط بعدی نیز با اجرای دستور `sub t0,t0,t5` ، با تفریق شمارنده حلقه از رجیستر `t0` دوباره در رجیستر

`t0` همان مقدار اولیه `65552` ذخیره میشود.

و دوباره با جامپ به اول حلقه، جایی که شرط در آن چک میشود، برمیگردیم.

با اجرای این دستورات اعداد `{0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21}` در اجزای حافظه ذخیره میشوند که همان دنباله فیبوناچی است.

در نهایت، مقادیر حافظه مانند زیر خواهند بود. اعداد داخل حافظه بر مبنای `Hex` نوشته شده اند.

Memory

Address	+0	+1	+2	+3
0x0001001c	00	00	00	00
0x00010018	15	00	00	00
0x00010014	03	05	08	0D
0x00010010	00	01	01	02
0x0001000c	00	00	00	00

در کد داده شده، هفت خط کد بالای حلقه یکبار اجرا میشوند و حلقه از مقدار `t5 = 1` تا `t5 < 8` اجرا میشود. در نهایت شرط حلقه زمانی که `t5 = 8` است برقرار نخواهد بود و به پایان کد میرسیم.

جمعا خواهیم داشت:

`71 = 1 + (9 * 7) + 7`

عملیات در این قطعه کد انجام میشود.

در کدی که ما داریم، اولا میدانیم که دستور `mv` یک دستور `I-Type` بوده و در چهار سیکل انجام میشود؛ همچنین دستور `sb` نیز، از آنجایی که دستور ذخیره سازی است هم در چهار سیکل انجام میگیرد. که در اینجا جمعا `56` دستور از این نوع داریم.

دستور `bgt` و `j` هر کدام در سه سیکل انجام میشوند. در اینجا نیز `15` نوع از این دو دستور داریم.

با محاسبه درصد تکرار هر کدام در این قطعه کد، مقدار `CPI` معادل را بدست میاوریم

CPI `(0.7887)(4) + (0.2112)(3) = 3.7884`

سوال ۳

op	funct3	funct7	Type	Instruction	Description	Operation
0010011 (19)	101	0100000	I	srai rd, rs1, uimm	shift right arithmetic imm.	rd = rs1 >>> uim

تغییرات مورد نیاز:

- پشتیبانی ALU از srai

در extend unit تغییری ایجاد نمیکنیم و همان sign extend I-Type برای این مورد جوابگوست, چرا که تنها ۵ بیت از imm در ALU مورد استفاده قرار خواهد گرفت(به این دلیل که رجیسترها ۳۲ بیتی اند و نمیتوان بیشتر از ۳۲ بیت آنهارا شیفت داد) و sign extend روی بیت ۱۲ ام اتفاق می‌افتد. پس ورودی ALU در دسترس می‌باشد.

Table 7.3

ALUOp	funct3	{op5, funct75}	ALUControl	Instruction	
00	x	x	000 (add)	lw, sw,	
01	x	x	001 (subtract)	beq	
10	000	00, 01, 10	000 (add)	add	
	000	11	001 (subtract)	sub	
	010	x	101 (set less than)	slt	
	110	x	011 (or)	or	
	111	x	010 (and)	and	
	101	x	111 (srai)	srai	

- واحد کنترل و مسیر داده نیازی به تغییر ندارند. درست مثل دیگر دستورالعمل‌های I-Type دیگر دستور اجرا می‌شود و سیگنال‌های کنترلی تفاوتی نمی‌کنند.