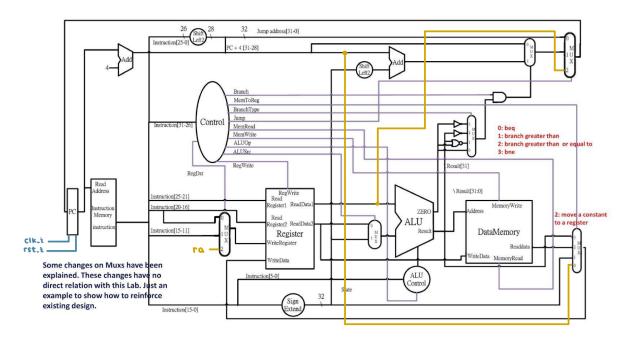
Computer Organization Lab3

Name: 莊婕妤 ID: 109550182

Architecture Diagram



Hardware Module Analysis

這次的 Lab 因為要多實作 jal、jr 兩個比較特別的指令,所以我多拉了上圖中的三條黃色的線當做各自的 MUX 的 input,像是多拉了 ReadData1 到 PC source MUX 讓 jr 可以跳到指定的 register,以及多拉了 ra 到 Register File MUX 和 PC + 4 到 Memory to Register MUX 讓 jal 較好實作。

為了處理更多樣的指令, 我也加了很多的 control signal 給我的 decoder, 也多設計了 3-to-1 MUX 跟 4-to-1 MUX 的 module。

Finished Part

Test Data 1

															50 000 ns	
Name	Value	0.000 ns		5.000 ns	10.000 ns	15.00	ns 20.0	00 ns 2	5.000 ns	30.000 ns	35.000	ns 40.0	00 ns 45	.000 ns	50.000 ns	. i ⁵
^¼ Clk	1															
^¼ Start	1															
end_count[31:0]	XXXXXXXX								xxxxx	xxx						
₩ pc_in[31:0]	0000003c		000000	04 00000000	00000000	0000001	0 00000014	00000020	000000024	00000028	0000002c	00000030	00000034	00000038	00000003с	0
₩ pc_out[31:0]	00000038		оооооо	00000000	00000008	0000000	c 00000010	00000014	00000020	00000024	00000028	0000002c	00000030	00000034	00000038	0
₩ next_pc[31:0]	0000003c		ооооо	04 00000000	0000000c	0000001	0 00000014	00000018	00000024	000000028	00000002c	00000030	00000034	00000038	0000003c	0
™ next_pc_1[31:0]	0000003c		000000	08 00000010	00000018	0000002	0 00000028	00000038	00000024	00000038	0000002c	00000030	000100Ь4	00000048	0000003c	0
™ instr[31:0]	00000000	<u> </u>	200100	01 20020002	20030003	2004000	4 20050005	0800000	ac010000	ac020004	8c060000	8c870000	00234020	8c090004	ококо	ю
WriteReg[4:0]	00	XX	01	02	03	04	05	Ŷ	00	т	06	07	08	09	00	
₩ rs[31:0]	00000000	ļ			Т		оооооо)				00000004	00000001	0	000000	
₩ rt[31:0]	00000000	(00 (.)				0000000	0		00000001	000000002			00000003	0	000000	
₩ WriteData[31:0]	00000000	 	ококок	01 000000002	00000003	0000000	4 00000000	000	00000	00000004	000000001	00000002	00000004	00000002	000000	ю
RegWrite	1			^	1	^	^			1	-			^		
1 Branch	0							_			-					
- Station								1					1			
C = 128						_			_							
	1,	2		0,		0,	0,		0,	0,		0				
ata Memory =	0,),	0,		0,	0,		0,	0,		0				
Data Memory = Data Memory =	0,	0		0,		0,	0,		0,	0, 0.		0				
Mata Memory =	0,	0	١,	0,	(0,	0,		0,	υ,		U				
	R1 =	1	R2 =		R3 =		3 R4	_	4 R	5 -	5	R6 =		1. R7 =		
8 = 4.		,		. (, ,		- /		,		0,			0. R15 =	=	
,	R17 =	-,	R18 =	(,		0, R20			21 =		R22 =		0. R23 =		
	R25 =			(, ,		0, R28		. ,		- ,	R30 =		0, R31 =		

Test Data 2



Finished Part Explanation

上面 Test Data 1 & Test Data 2 波形圖跑出的結果跟我自己 trace 過整個 assembly code 的結果一模一樣,因此我認為我的電路圖實作是正確的。

Problems You Met and Solutions

在實作的過程中,我遇到的問題主要是不太熟悉 Jump、Jump and Link、Jump Register 是如何運作的,以及他們相對應的 control signals 該如何去設定。但在理解他們的 Instructions,並一條線一條線 trace 過整個電路,讓我在設計 decoder 時更加順手。

Summary

這次的 Lab 讓我更了解一個能夠處理如此多種指令的 CPU 內部長什麼樣子,透過各個 Module 的設計,再整個接起來合成一個 Simple Single CPU,幫助我更熟悉他們背後運作的原理,像是如何運用 control signals 讓其可以執行許多不一樣的 Instructions。經過此次作業後,我對於整個 CPU 的觀念都更加清楚了。