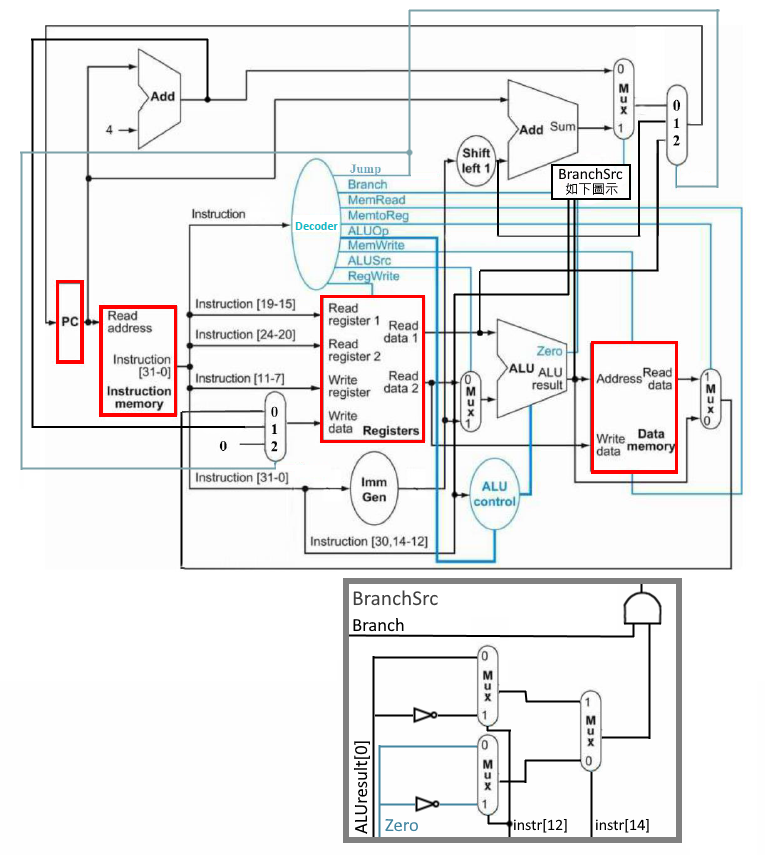
**Computer Organization**

**0716214 江岳勳**

**0716222 黃偉傑**

**Architecture diagram:**

****

**Detailed description of the implementation:**

有許多module都跟Lab3的一模一樣，所以這裡只提到有更動過的實作細節

* ALU\_Ctrl.v

因為我們上次都已經把lw跟sw等指令分配完成了，所以這次要新增處理的只有blt跟bge而已。blt和bge都是執行slt指令，再由外部top module來區分兩者。

因此這裡的ALU\_Ctrl\_o為4’b0111：



* Decoder.v

多判斷指令是否為jal，若是的話Instr\_field為4。

查閱真值表，逐一設定指令的MemtoReg, RegWrite, MemRead, MemWrite, Jump (參見後表)

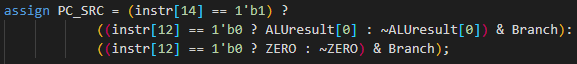
* MUX\_3to1.v

assign data\_o = (select\_i == 2'b00) ? data0\_i : ((select\_i == 2'b01 ? data1\_i : data2\_i));

* Simple\_Single\_CPU.v

將多新增的module彼此之間的wire接上：Data\_Memory, Mux\_MemToRegSrc, Mux\_WBSrc, Mux\_PCJumpSrc

處理新的Branch指令：原本的PC\_Src只有考慮beq和bne(只處理zero)，新增了blt和bge後，查看真值表可以發現這四個可以先用instr[14]分類，再進行處理。所以新的PC\_Src為：



下面是我們建立的真值表：

ALU\_Ctrl (不會進ALU 的 instruction : jal, jalr)

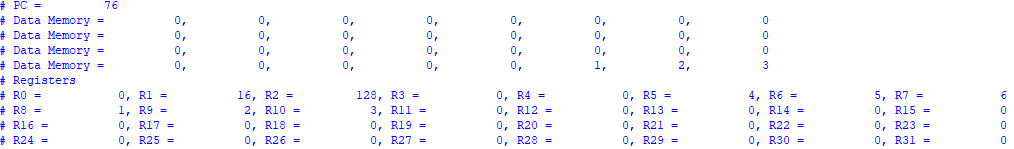
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **instruction** | **ALUOp[1 : 0]** | **instr [3 : 0]** | **ALU action** | **ALU\_Ctrl\_o** |
| lw | 00 | 010X | add | 0010 |
| sw | 00 | 010X | add | 0010 |
| add | 10 | 0000 | add | 0010 |
| sub | 10 | 0001 | sub | 0110 |
| and | 10 | 1110 | and | 0000 |
| or | 10 | 1100 | or | 0001 |
| xor | 10 | 1000 | xor | 0011 |
| slt | 10 | 0100 | slt | 0111 |
| sll | 10 | 0010 | sll | 0100 |
| sra | 10 | 1011 | sra | 0101 |
| beq | 01 | 000X | sub | 0110 |
| bne | 01 | 001X | sub | 0110 |
| blt | 01 | 100X | slt | 0111 |
| bge | 01 | 101X | slt | 0111 |
| addi | 11 | 000X | add | 0010 |
| andi | 11 | 111X | and | 0000 |
| ori | 11 | 110X | or | 0001 |
| xori | 11 | 100X | xor | 0011 |
| slti | 11 | 010X | slt | 0111 |
| slli | 11 | 0010 | sll | 0100 |
| srli | 11 | 1010 | srli | 1111 |

Decoder

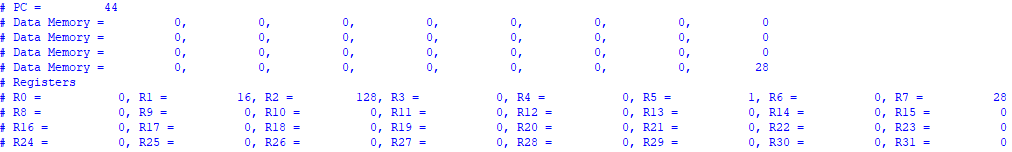
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Field(Opcode)** | **M2R** | **RegW** | **MemR** | **MemW** | **Jump[2]** | **ALUSrc** | **Branch** | **ALUOp[2]** |
| R-type | 0 | 0 | 1 | 0 | 0 | 00 | 0 | 0 | 10 |
| LW | 1(0000011) | 1 | 1 | 1 | 0 | 00 | 1 | 0 | 00 |
| JALR | 1(1100111) | 0 | 1 | 0 | 0 | 10 | 1 | 0 | 11 |
| I-type | 1(else) | 0 | 1 | 0 | 0 | 00 | 1 | 0 | 11 |
| S-type | 2 | 0 | 0 | 0 | 1 | 00 | 1 | 0 | 00 |
| B-type | 3 | 0 | 0 | 0 | 0 | 00 | 0 | 1 | 01 |
| JAL | 4 | 0 | 1 | 0 | 0 | 01 | 0 | 0 | 00 |

**Implementation results:**

CO\_test\_data1.txt：



CO\_test\_data2.txt：



**Problems encountered and solutions:**

1. 在做blt和bge的時候，一開始不是很清楚要怎麼讓Branch的PC\_Src知道現在示做blt/bge/beq/bne

* 將真值表建出來之後，很輕鬆地就可以看出當instr[14]為1時是blt/bge，了解之後再改動PC\_src便沒遇到什麼困難。

1. 在測試時sw怎麼樣都不會動，解讀了一下之後發現sw的目標為位置為-12的記憶體，想了很久怎麼想都覺得不合理

* 東看西看找很久才發現Lab3和Lab4的Reg\_File不一樣，Lab4的Reg\_Filer2的初始值為128，因為我們是直接使用Lab3的結果下去時做Lab4的需求，所以沒改到這裡。

**Comment:**

0716214 江岳勳：

我覺得這次的作業其實非常簡單，因為我們上次都已經把DataMemory有關的功能都寫好了，只需要將線接上，再讓ALU\_Ctrl能夠多判斷blt和bge就結束了，是一份相對輕鬆的作業。

0716222黃偉傑：

有了上次Lab3協作與建構Simplfied CPU的經驗，我認為對降低這次Lab的難度有顯著的幫助。感覺得出來上次跟這次Lab應該原本要合在一起，只是這樣一來會讓loading 更繁重，感謝助教的貼心哈哈。

經過這次經驗，讓我也對於Single Cycle 的架構更加了解了，讓我覺得挑戰pipelined CPU或許也不是件難事，希望我不要被打臉太嚴重。t