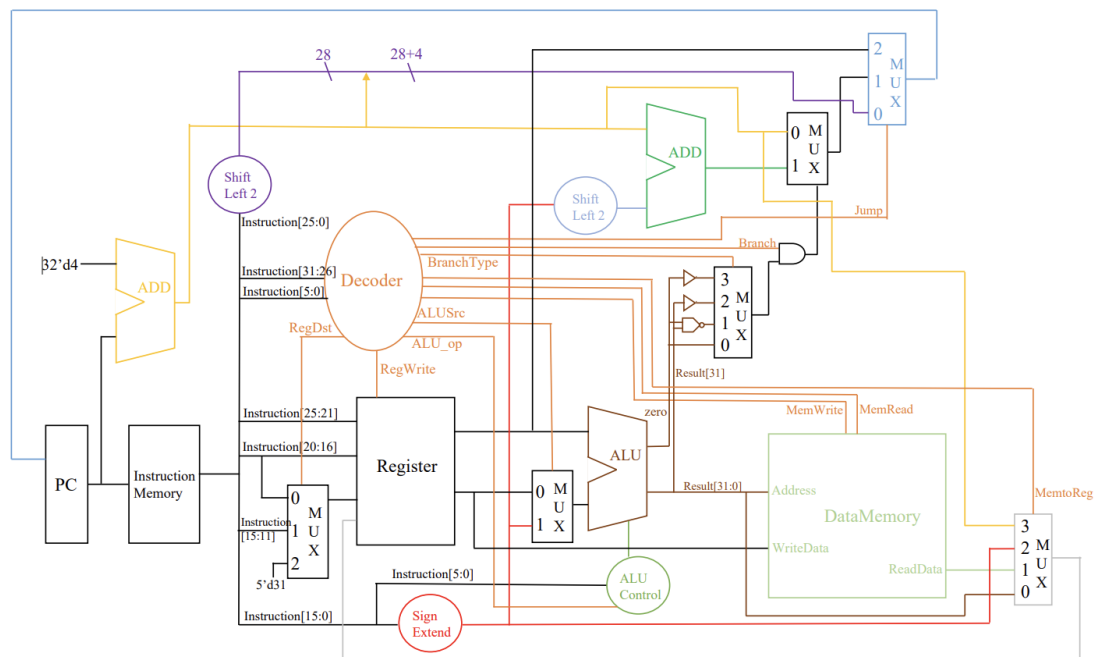


Lab3: Single Cycle CPU - Complete Edition

student ID:110550093 Name: 蔡師睿

1. Architecture diagrams



Single cycle CPU

2. Hardware module analysis

Program Counter

當 reset 不為 0 時，輸出輸入的訊號；若為 0，則輸出 0。

Instruction Memory

讀取.txt 檔中的指令。

Register File

讀取輸入的暫存器的地址並輸出其儲存的值。另外根據 RegWrite 判斷，是否需要對暫存器寫入新的值。

Data Memory

存放、讀取、寫入記憶體的地方。

Adder

將輸入的兩個 32-bit 數字相加輸出。

Decoder

解碼指令的地方。

| Instruction | RegWrite | ALU_op | ALUSrc | RegDst | Branch |
|------------------------|----------|--------|--------|--------|--------|
| and, or, add, sub, slt | 1 | 10 | 0 | 01 | 0 |
| jr | 0 | xx | x | xx | x |
| jump | 0 | xx | x | xx | x |
| jal | 1 | xx | x | 10 | x |
| beq, bne | 0 | 01 | 0 | xx | 1 |
| addi | 1 | 00 | 1 | 00 | 0 |
| slti | 1 | 11 | 1 | 00 | 0 |
| lw | 1 | 00 | 1 | 00 | 0 |
| sw | 0 | 00 | 1 | xx | 0 |
| else | 0 | 00 | 0 | 00 | 0 |

| Instruction | Jump | MemtoReg | MemRead | MemWrite | BranchType |
|------------------------|------|----------|---------|----------|------------|
| and, or, add, sub, slt | 01 | 00 | 0 | 0 | 00 |
| jr | 10 | xx | 0 | 0 | xx |
| jump | 00 | xx | 0 | 0 | xx |
| jal | 00 | 11 | 0 | 0 | xx |
| beq | 01 | xx | 0 | 0 | 00 |
| bne | 01 | xx | 0 | 0 | 11 |
| addi | 01 | 00 | 0 | 0 | xx |
| slti | 01 | 00 | 0 | 0 | xx |
| lw | 01 | 01 | 1 | 0 | xx |
| sw | 01 | xx | 0 | 1 | xx |
| else | 01 | 00 | 0 | 0 | 00 |

MUX 2 to 1

若 select 爲 1，輸出 data1；若爲 0，輸出 data0。

MUX 3 to 1

select 爲 0、1、2 時，分別輸出 data0、data1、data2。

MUX 4 to 1

select 爲 0、1、2、3 時，分別輸出 data0、data1、data2、data3。

Shift left two

將輸入的 32-bit 數字往左移兩位補 0 再輸出。

Sign Extended

輸入的 16-bit 二進位數字的最高位若爲 1，在此數字前補 16 個 1，若爲 0，則在前補 16 個 0，最後將補完位滿足 32-bit 的數字輸出。

ALU Control

根據由 Decoder 傳入的 ALUOp 和 Instruction 第 5 到 0 bit 的值，輸出相對應的 ALU Control 值。

| {ALUOp, Instruction[5:0]} | ALU Control |
|---------------------------|-------------|
| 9'b010_100100 | 4'b0000 |
| 9'b010_100101 | 4'b0001 |
| 9'b000_xxxxxxx | 4'b0010 |
| 9'b010_100000 | 4'b0010 |
| 9'b001_xxxxxxx | 4'b0110 |
| 9'b010_100010 | 4'b0110 |
| 9'b010_101010 | 4'b0111 |
| 9'b011_xxxxxxx | 4'b0111 |

ALU

根據 ALU Control 傳進來的值做相對應的 And、Or、Add、Sub、Slt 或 Nor 操作。

| ALU action | ALU input control ={A_invert, B_invert, Operation} |
|------------|---|
| And | 4'b0000 |
| Or | 4'b0001 |
| Add | 4'b0010 |
| Sub | 4'b0110 |
| Nor | 4'b1100 |
| Slt | 4'b0111 |

Single cycle CPU

根據 Architecture diagram 和以上的 modules 組出正確的 Single-cycle CPU。

3. Finished part

Result of data1

```

Data Memory =      1,      2,      0,      0,      0,      0,      0,      0
Data Memory =      0,      0,      0,      0,      0,      0,      0,      0
Data Memory =      0,      0,      0,      0,      0,      0,      0,      0
Data Memory =      0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =      0, R1 =      1, R2 =      2, R3 =      3, R4 =      4, R5 =      5, R6 =      1, R7 =      2
R8 =      4, R9 =      2, R10 =      0, R11 =      0, R12 =      0, R13 =      0, R14 =      0, R15 =      0
R16 =      0, R17 =      0, R18 =      0, R19 =      0, R20 =      0, R21 =      0, R22 =      0, R23 =      0
R24 =      0, R25 =      0, R26 =      0, R27 =      0, R28 =      0, R29 =      0, R30 =      0, R31 =      0

```

Result of data2

```

Data Memory =      0,      0,      0,      0,      0,      0,      0,      0
Data Memory =      0,      0,      0,      0,      0,      0,      0,      0
Data Memory =      0,      0,      0,      0,      68,      2,      1,      68
Data Memory =      2,      1,      68,      4,      3,      16,      0,      0
Registers
R0 =      0, R1 =      0, R2 =      5, R3 =      0, R4 =      0, R5 =      0, R6 =      0, R7 =      0
R8 =      0, R9 =      1, R10 =      0, R11 =      0, R12 =      0, R13 =      0, R14 =      0, R15 =      0
R16 =      0, R17 =      0, R18 =      0, R19 =      0, R20 =      0, R21 =      0, R22 =      0, R23 =      0
R24 =      0, R25 =      0, R26 =      0, R27 =      0, R28 =      0, R29 =      0, R30 =      0, R31 =      16

```

註：我的輸出使用 testbench.v 中被註解掉的那一段，因此格式稍有不同。

4. Problems and solutions

1. 在 vivado 中 simulation 時發現我的整個 CPU 都跑不動，一個指令都跑不進去，後來檢查了很久才發現是在 Simple_Single_CPU.v 的 program counter 接錯線路了，導致沒辦法讀入任何指令。
2. 後來又發現 ALU 沒有運作，又再次經過長時間的 Debug 後，發現犯了低級錯誤，ALU_Ctrl.v 中因為我的寫法，所以應該要使用 casex，但我少打了一個 x。

5. Summary

比起 Lab2 雖然只增加了幾個指令，但實作起來更為困難，且整個 CPU 結構也更為複雜，即使我很快完成整體 CPU 的設計，但大部分的時間都花在 Debug 上，也讓我感覺 Verilog debug 的困難程度遠超過 C/C++ 和 Python。