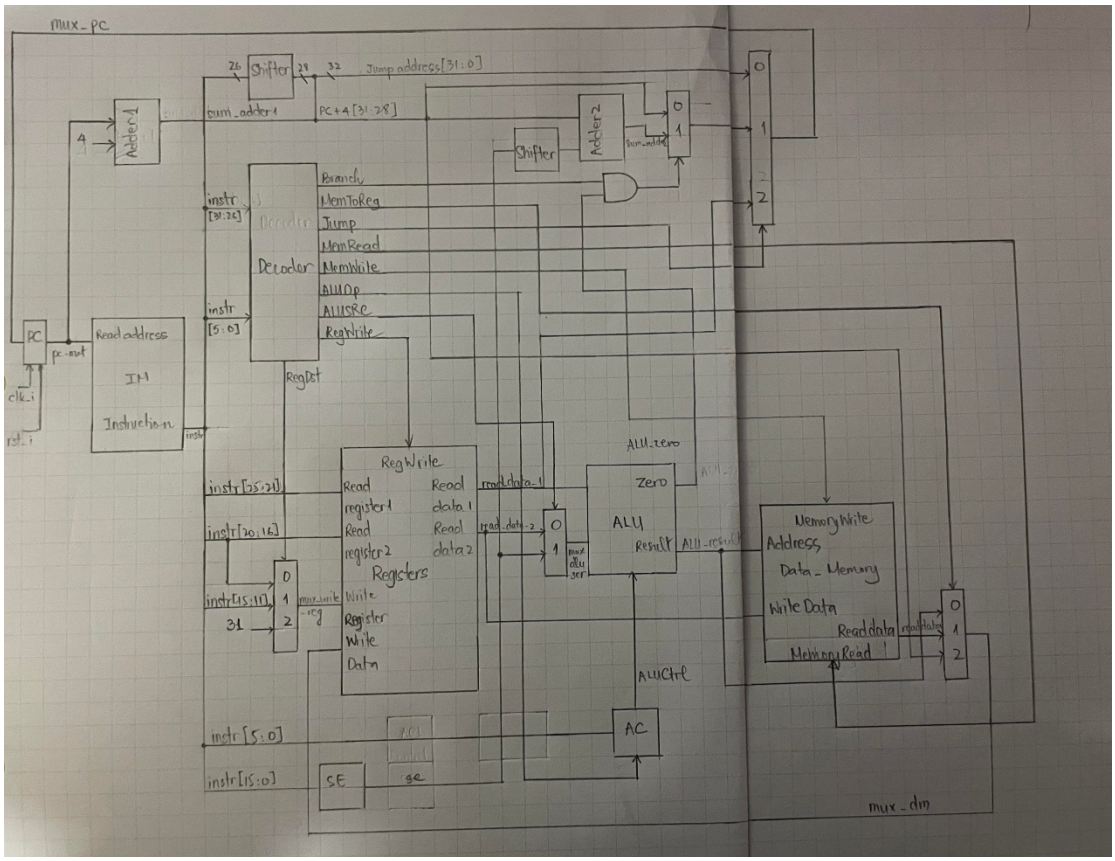
Computer Organization Lab3

Name: 郭大寧

ID: 109550184

Architecture diagrams:



Hardware module analysis:

(explain how the design work and its pros and cons)

rst_i 變成 1 開始執行程式。每一個 clock edge 執行新指令，前指令結束。讀取 instruction，使用 decoder 來分析需要執行的計算，取得各個 control signals 來控制。

在 ALU 部分，只針對 add、addi、sub、and、or、slt、slti、beq、lw、sw 取得 operation，分成 5 種：and，or，add、addi、lw、sw，sub、beq，slt、slti 來計算。

增加 Data Memory 部分來處理 lw 和 sw，利用 decoder 取到的 MemWrite 和 MemRead 來分別需要執行的指令。

使用 MUX 3x1 來選擇適合的 write register，write data，下一個 PC。RegDst 會決定 write register，MemToReg 會決定 write data，Jump 會決定下一個 PC。

Finished part:

(show the screenshot of the simulation result and waveform, and explain it)

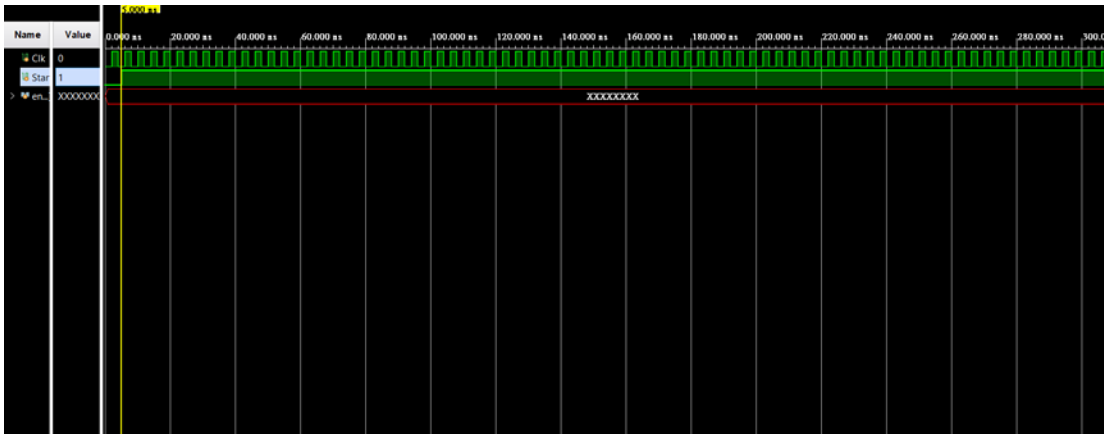
Test1:

```
PC =          x
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 =        128, R31 =          0
```

Test2:

```
PC =          x
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 =        128, R31 =         16
```

Waveform:



Explain:

Test1 部分：有做到 j 指令，在 PC=36 可以看到 m0 變成 1。

Test2 部分：在 PC=20 有做到 jal 指令，而在 PC=24 有做到 fib 裡第一個指令。在 PC=52 結束後又呼叫 jal 指令回到 PC=20。通過 test2 可以看到 recursion algorithm。

Pros:

利用 decoder 來取得 control signals 控制。

Cons:

為了做到更多 opcode，control signals 開報容量需要增加。

Problems you met and solutions:

增加 jal 時，write register 和 write data 部分不對。增加 RegDst 和 MemToReg 容量來調整。增加 address 5’b11111 和 PC+4 部分。

Summary:

實作 single cycle cpu，每一個 clock edge 代表執行一個新指令，結束前指令。利用 decoder 來分析 instruction 取得各個 control signals 來控制 cpu 需執行的動作。跟 lab2 比，多增加 lw、sw、jump、jal、jr 指令。使用 MUX 加上 control signals 來選出需要的參數傳達。