計算機組織 Final Project

Pipelined CPU Design

老師：朱守禮老師

第 10 組

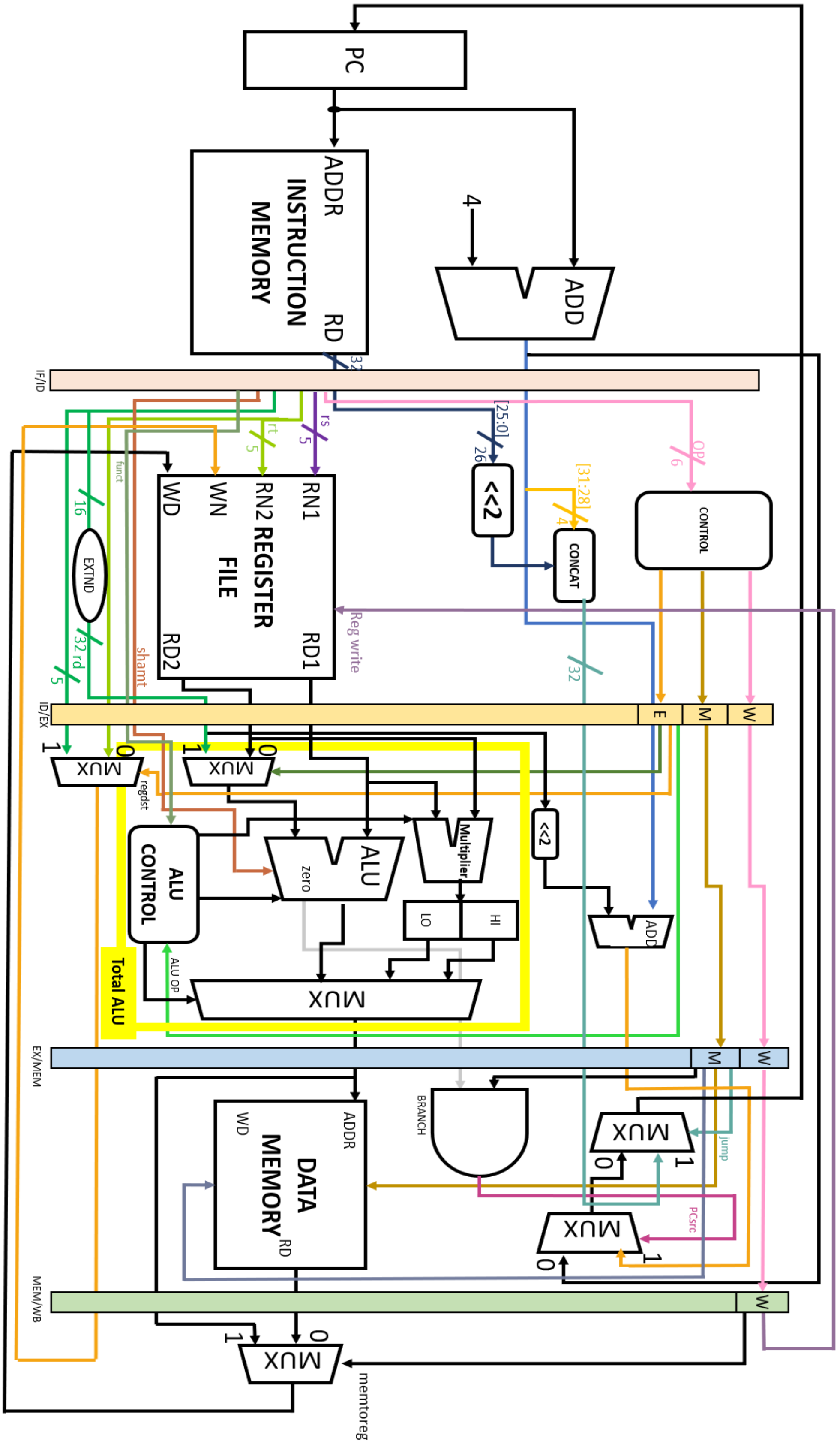
11027112 陳逸駿

11027140 陳芃睿

11027147 邱峻彥

11027149 游婕歆

1. 架構圖



PC
ADDR
INSTRUCTION MEMORY
RD
4
ADD
IF/ID
32
[25:0]
26
<<2
[31:28]
4
CONCAT
OP
6
CONTROL
rs
5
rt
5
funct
RN1
RN2
WN
WD
REGISTER FILE
RD1
RD2
Reg write
shamt
16
EXTND
32 rd
5
32
ID/EX
E
M
W
MUX
1
0
regdst
ALU CONTROL
ALU
zero
Multiplier
LO
HI
ALU OP
Total ALU
<<2
ADD
EX/MEM
BRANCH
ADDR
WD
DATA MEMORY
RD
jump
PCsrc
MEM/WB
memtoreg

2. 背景

使用 verilog 撰寫並運用 Modelsim 進行模擬，以 Midterm Project 所設計之 ALU Design 為基礎，參考課本 Chapter 4 與課程講義之 Pipelined Datapath ，設計一個 Pipelined MIPS Lite CPU 。將原本的 ALU 換成期中 project 所撰寫的 ALU 後要維持運作正常，最後在符合老師的規定下，完成老師要求的 16 道指令。

3. 設計重點說明

參考了課本後面的 Datapath and Control Unit ，使用 single_cycle_CPU 用 5 個暫存器(分別是 IF/ID、ID/EX、EX/MEM、MEM/WB)切成五個部分

- 讀取指令
- 做 pc=pc+4

-------------------------------IF/ID---------------------------

- 解碼指令
- 指令擴充
- 取 pc+4 後的指令(32bits)的前面 6 位元的後 4 位元和 jump 指令中的 address（26 位元）向左偏移 2 位元的組合

---------------------------------ID/EX--------------------------

- 處理指令運算

-------------------------------EX/MEM-------------------------

- 看需不需要執行 jump 或是 branch
- 運算結果存入 data memory

-------------------------------MEM/WB -------------------------

- 決定要寫回 WD 的要選 data memory 中的值還是 TotalALU 算好的值

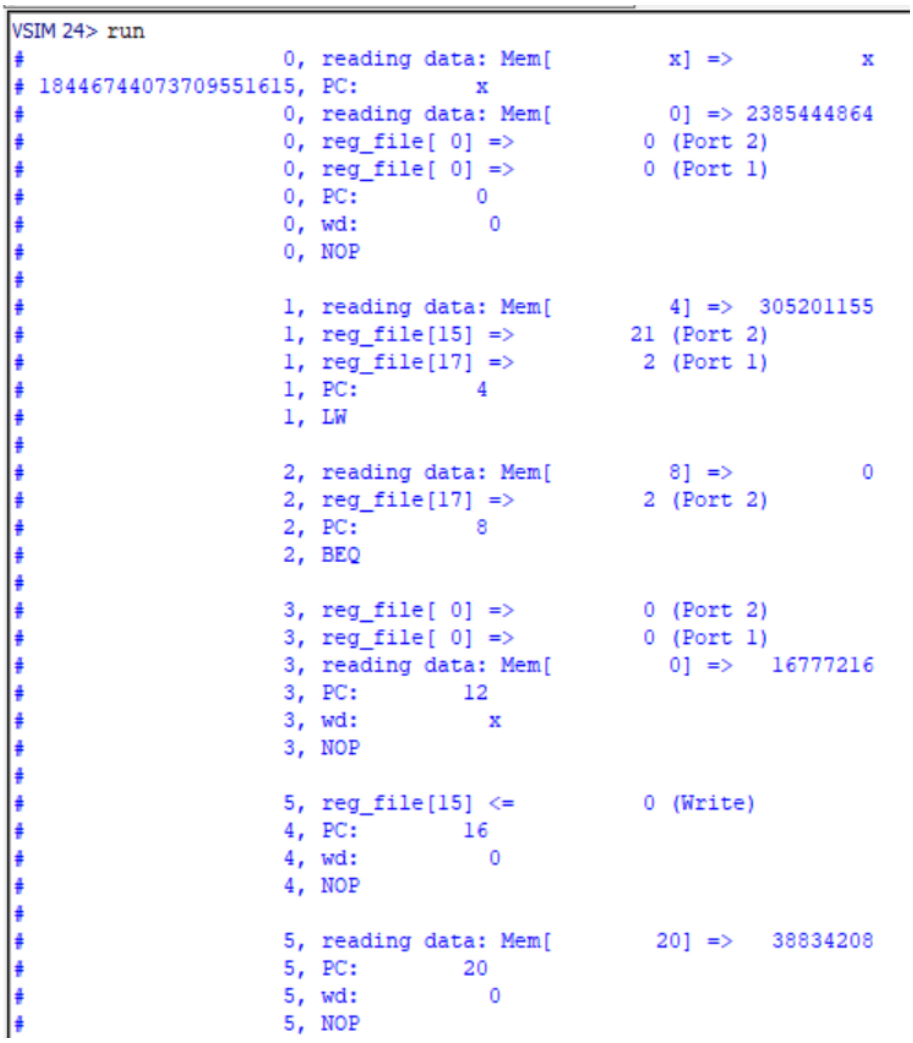
4.modelsim 執行結果與討論

✔ 驗證結果&Waveform

1.

Beq 和 j 都會造成 DataHazard 所以要接 3 個 nop 解決

IF	ID	EX	MEM	WB	
nop					
lw	nop				
beq	lw	nop			beq 會造成 DataHazard
nop	nop	nop	beq	lw	所以要接 3 個 nop 解決 DataHazard
add	nop	nop	nop	beq	beq 之後就跳去 add 指令了

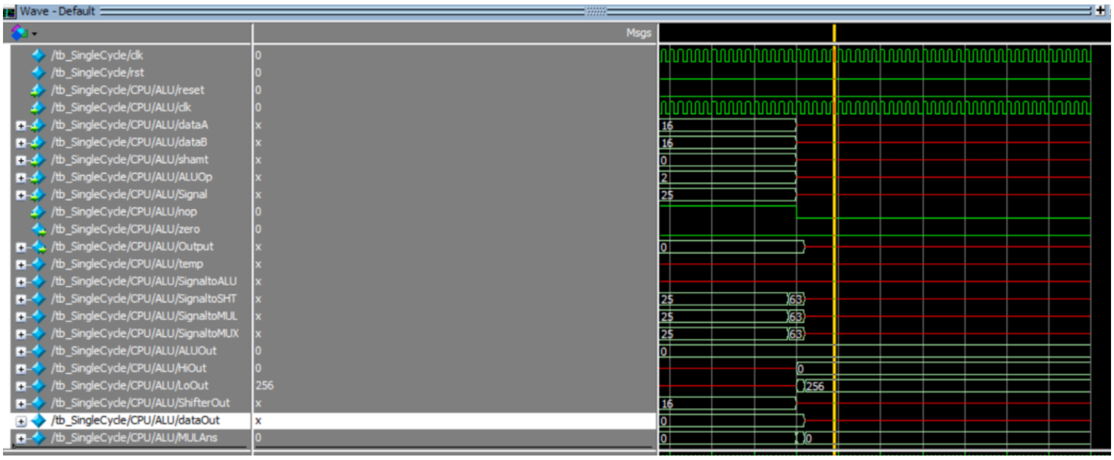


```
VSIM 24> run
#                     0, reading data: Mem[         x] =>          x
# 18446744073709551615, PC:          x
#                     0, reading data: Mem[         0] => 2385444864
#                     0, reg_file[ 0] =>          0 (Port 2)
#                     0, reg_file[ 0] =>          0 (Port 1)
#                     0, PC:          0
#                     0, wd:           0
#                     0, NOP
#
#                     1, reading data: Mem[         4] =>  305201155
#                     1, reg_file[15] =>         21 (Port 2)
#                     1, reg_file[17] =>          2 (Port 1)
#                     1, PC:          4
#                     1, LW
#
#                     2, reading data: Mem[         8] =>          0
#                     2, reg_file[17] =>          2 (Port 2)
#                     2, PC:          8
#                     2, BEQ
#
#                     3, reg_file[ 0] =>          0 (Port 2)
#                     3, reg_file[ 0] =>          0 (Port 1)
#                     3, reading data: Mem[         0] =>   16777216
#                     3, PC:         12
#                     3, wd:           x
#                     3, NOP
#
#                     5, reg_file[15] <=          0 (Write)
#                     4, PC:         16
#                     4, wd:           0
#                     4, NOP
#
#                     5, reading data: Mem[        20] =>   38834208
#                     5, PC:         20
#                     5, wd:           0
#                     5, NOP
```

2.

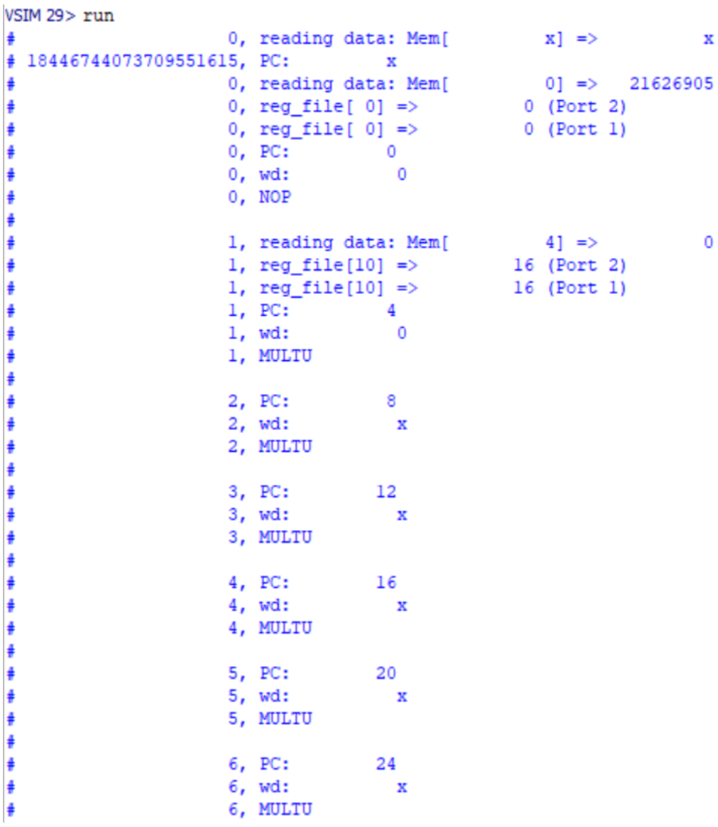
- 遇到 multu 的時候乘法要做 32 回合才做完，要再一回合寫給 HILO 所以後面需要加 33 個 nop

- maddu 乘加器因為要多一回合做加法所以後面需要加 34 個 nop

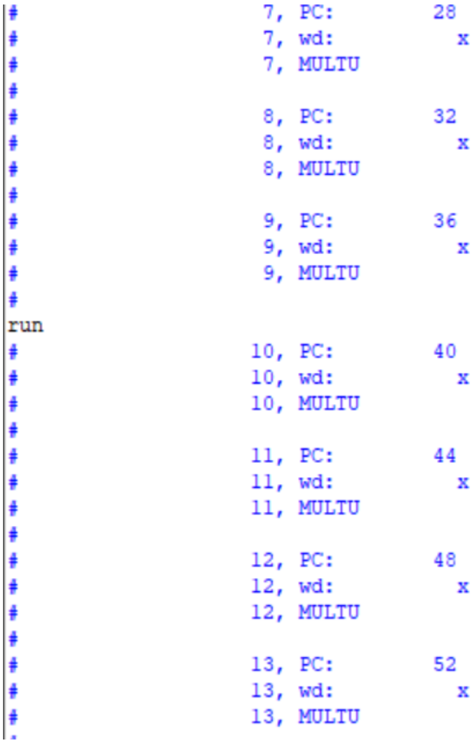


Wave - Default

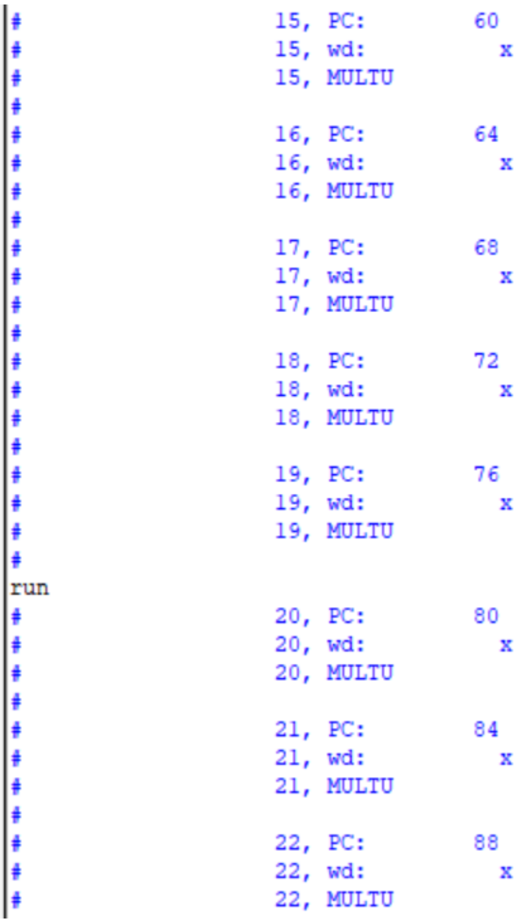
```
Msgs
/tb_SingleCycle/clk                  0
/tb_SingleCycle/rst                  0
/tb_SingleCycle/CPU/ALU/reset        0
/tb_SingleCycle/CPU/ALU/clk          0
/tb_SingleCycle/CPU/ALU/dataA        x
/tb_SingleCycle/CPU/ALU/dataB        x
/tb_SingleCycle/CPU/ALU/shamt        x
/tb_SingleCycle/CPU/ALU/ALUOp        x
/tb_SingleCycle/CPU/ALU/Signal       x
/tb_SingleCycle/CPU/ALU/nop          0
/tb_SingleCycle/CPU/ALU/zero         0
/tb_SingleCycle/CPU/ALU/Output       x
/tb_SingleCycle/CPU/ALU/temp         x
/tb_SingleCycle/CPU/ALU/SignaltoALU  x
/tb_SingleCycle/CPU/ALU/SignaltoSHT  x
/tb_SingleCycle/CPU/ALU/SignaltoMUL  x
/tb_SingleCycle/CPU/ALU/SignaltoMUX  x
/tb_SingleCycle/CPU/ALU/ALUOut       0
/tb_SingleCycle/CPU/ALU/HiOut        0
/tb_SingleCycle/CPU/ALU/LoOut        256
/tb_SingleCycle/CPU/ALU/ShifterOut   x
/tb_SingleCycle/CPU/ALU/dataOut      x
/tb_SingleCycle/CPU/ALU/MULAns       0
```



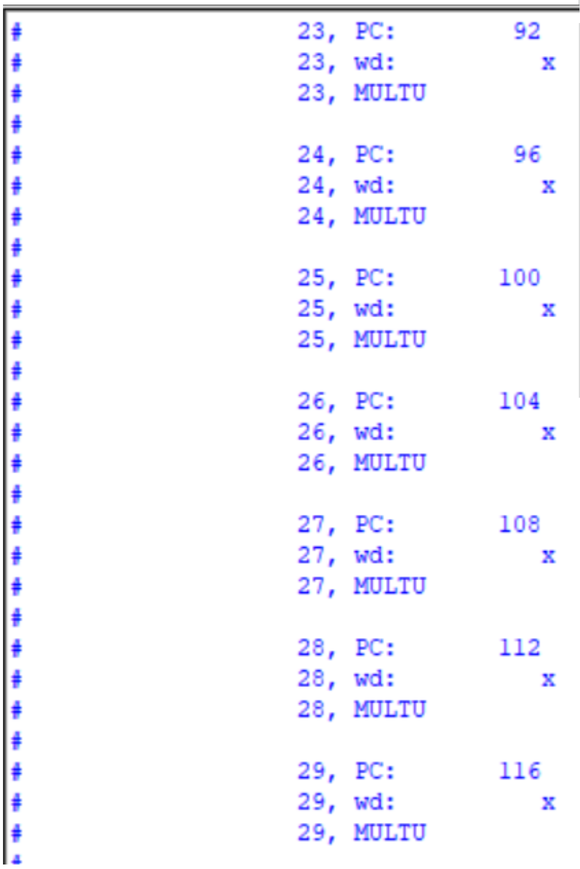
```
VSIM 29> run
#                     0, reading data: Mem[         x] =>          x
# 18446744073709551615, PC:          x
#                     0, reading data: Mem[         0] =>   21626905
#                     0, reg_file[ 0] =>          0 (Port 2)
#                     0, reg_file[ 0] =>          0 (Port 1)
#                     0, PC:          0
#                     0, wd:           0
#                     0, NOP
#
#                     1, reading data: Mem[         4] =>          0
#                     1, reg_file[10] =>         16 (Port 2)
#                     1, reg_file[10] =>         16 (Port 1)
#                     1, PC:          4
#                     1, wd:           0
#                     1, MULTU
#
#                     2, PC:          8
#                     2, wd:           x
#                     2, MULTU
#
#                     3, PC:         12
#                     3, wd:           x
#                     3, MULTU
#
#                     4, PC:         16
#                     4, wd:           x
#                     4, MULTU
#
#                     5, PC:         20
#                     5, wd:           x
#                     5, MULTU
#
#                     6, PC:         24
#                     6, wd:           x
#                     6, MULTU
```



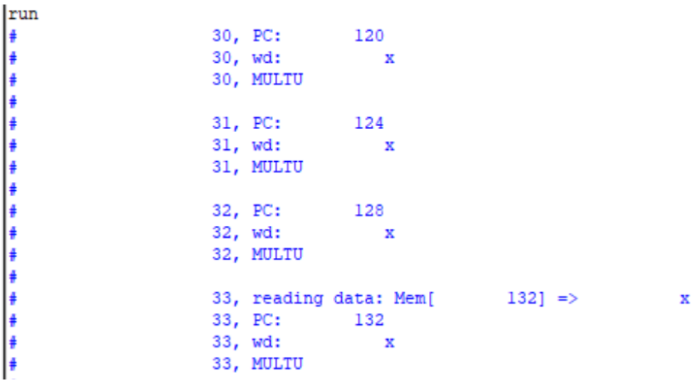
```
#                    7, PC:         28
#                    7, wd:           x
#                    7, MULTU
#
#                    8, PC:         32
#                    8, wd:           x
#                    8, MULTU
#
#                    9, PC:         36
#                    9, wd:           x
#                    9, MULTU
#
run
#                   10, PC:         40
#                   10, wd:           x
#                   10, MULTU
#
#                   11, PC:         44
#                   11, wd:           x
#                   11, MULTU
#
#                   12, PC:         48
#                   12, wd:           x
#                   12, MULTU
#
#                   13, PC:         52
#                   13, wd:           x
#                   13, MULTU
```



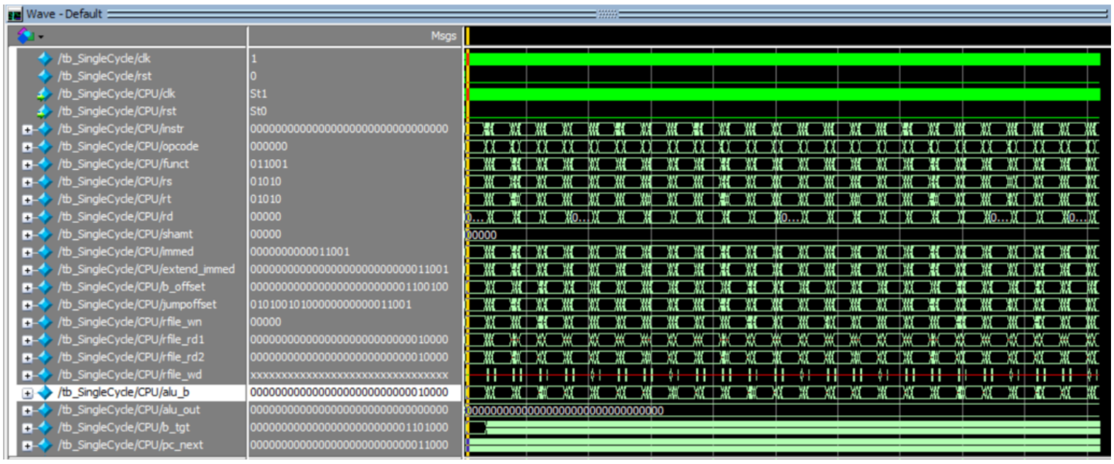
```
#                   15, PC:         60
#                   15, wd:           x
#                   15, MULTU
#
#                   16, PC:         64
#                   16, wd:           x
#                   16, MULTU
#
#                   17, PC:         68
#                   17, wd:           x
#                   17, MULTU
#
#                   18, PC:         72
#                   18, wd:           x
#                   18, MULTU
#
#                   19, PC:         76
#                   19, wd:           x
#                   19, MULTU
#
run
#                   20, PC:         80
#                   20, wd:           x
#                   20, MULTU
#
#                   21, PC:         84
#                   21, wd:           x
#                   21, MULTU
#
#                   22, PC:         88
#                   22, wd:           x
#                   22, MULTU
```



```
#                   23, PC:         92
#                   23, wd:           x
#                   23, MULTU
#
#                   24, PC:         96
#                   24, wd:           x
#                   24, MULTU
#
#                   25, PC:        100
#                   25, wd:           x
#                   25, MULTU
#
#                   26, PC:        104
#                   26, wd:           x
#                   26, MULTU
#
#                   27, PC:        108
#                   27, wd:           x
#                   27, MULTU
#
#                   28, PC:        112
#                   28, wd:           x
#                   28, MULTU
#
#                   29, PC:        116
#                   29, wd:           x
#                   29, MULTU
```

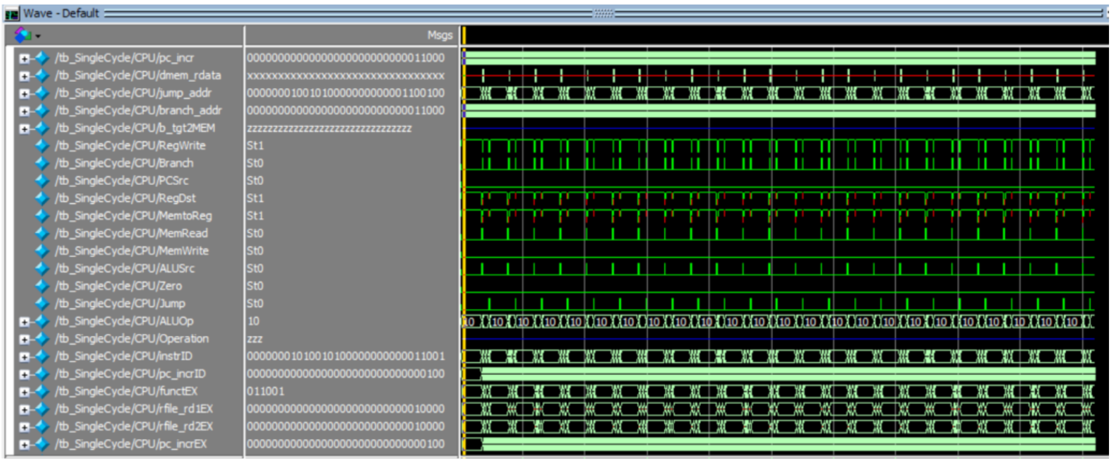


```
run
#                   30, PC:        120
#                   30, wd:           x
#                   30, MULTU
#
#                   31, PC:        124
#                   31, wd:           x
#                   31, MULTU
#
#                   32, PC:        128
#                   32, wd:           x
#                   32, MULTU
#
#                   33, reading data: Mem[        132] =>           x
#                   33, PC:        132
#                   33, wd:           x
#                   33, MULTU
```



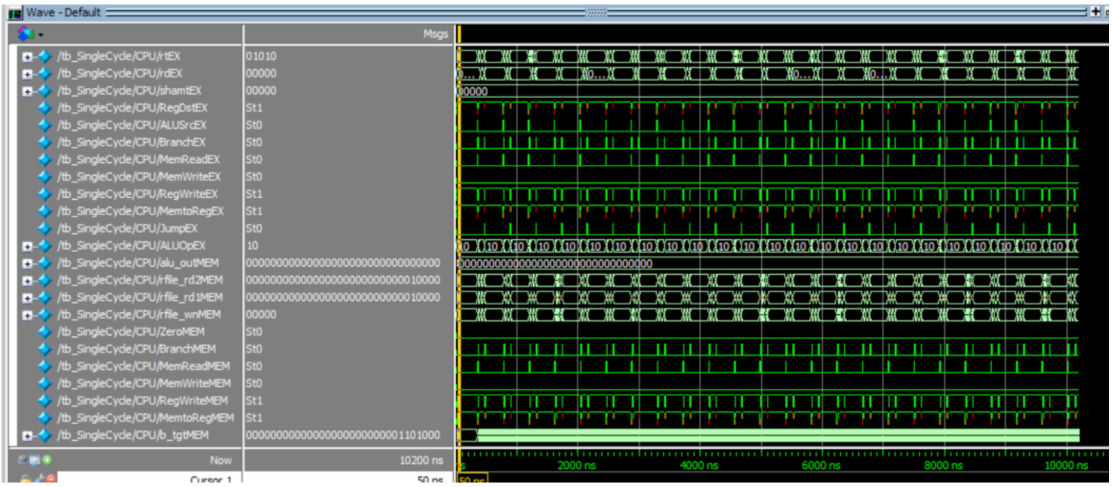
Wave - Default

	Msgs
/tb_SingleCycle/clk	1
/tb_SingleCycle/rst	0
/tb_SingleCycle/CPU/clk	St1
/tb_SingleCycle/CPU/rst	St0
/tb_SingleCycle/CPU/instr	00000000000000000000000000000000
/tb_SingleCycle/CPU/opcode	000000
/tb_SingleCycle/CPU/funct	011001
/tb_SingleCycle/CPU/rs	01010
/tb_SingleCycle/CPU/rt	01010
/tb_SingleCycle/CPU/rd	00000
/tb_SingleCycle/CPU/shamt	00000
/tb_SingleCycle/CPU/immed	0000000000011001
/tb_SingleCycle/CPU/extend_immed	00000000000000000000000000011001
/tb_SingleCycle/CPU/b_offset	00000000000000000000000001100100
/tb_SingleCycle/CPU/jumpoffset	01010010100000000000011001
/tb_SingleCycle/CPU/rfile_wn	00000
/tb_SingleCycle/CPU/rfile_rd1	00000000000000000000000000010000
/tb_SingleCycle/CPU/rfile_rd2	00000000000000000000000000010000
/tb_SingleCycle/CPU/rfile_wd	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
/tb_SingleCycle/CPU/alu_b	00000000000000000000000000010000
/tb_SingleCycle/CPU/alu_out	00000000000000000000000000000000
/tb_SingleCycle/CPU/b_tgt	00000000000000000000000001101000
/tb_SingleCycle/CPU/pc_next	00000000000000000000000000011000



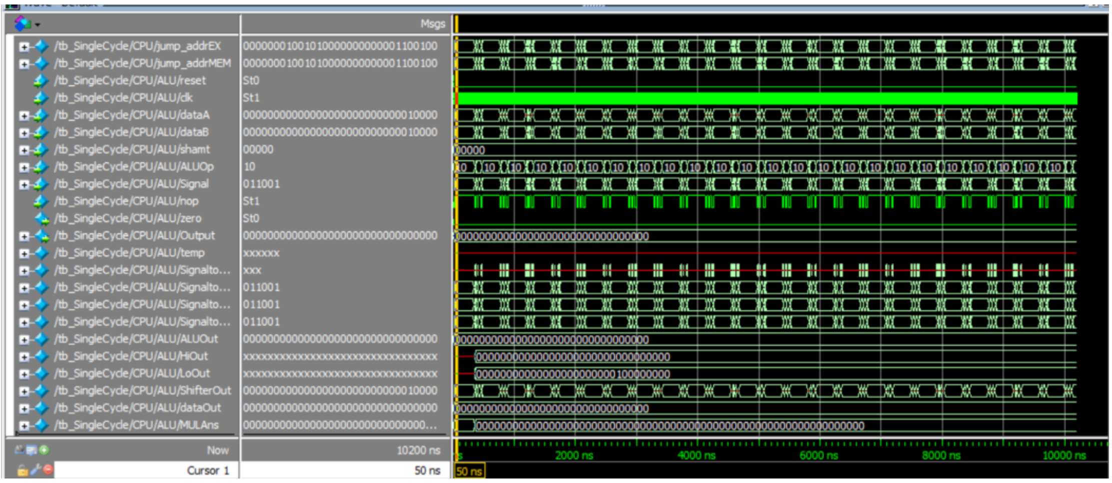
Wave - Default

	Msgs
/tb_SingleCycle/CPU/pc_incr	00000000000000000000000000011000
/tb_SingleCycle/CPU/dmem_rdata	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
/tb_SingleCycle/CPU/jump_addr	00000001001010000000000001100100
/tb_SingleCycle/CPU/branch_addr	00000000000000000000000000011000
/tb_SingleCycle/CPU/b_tgt2MEM	zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
/tb_SingleCycle/CPU/RegWrite	St1
/tb_SingleCycle/CPU/Branch	St0
/tb_SingleCycle/CPU/PCSrc	St0
/tb_SingleCycle/CPU/RegDst	St1
/tb_SingleCycle/CPU/MemtoReg	St1
/tb_SingleCycle/CPU/MemRead	St0
/tb_SingleCycle/CPU/MemWrite	St0
/tb_SingleCycle/CPU/ALUSrc	St0
/tb_SingleCycle/CPU/Zero	St0
/tb_SingleCycle/CPU/Jump	St0
/tb_SingleCycle/CPU/ALUOp	10
/tb_SingleCycle/CPU/Operation	zzz
/tb_SingleCycle/CPU/instrID	00000001010010100000000000011001
/tb_SingleCycle/CPU/pc_incrID	00000000000000000000000000000100
/tb_SingleCycle/CPU/functEX	011001
/tb_SingleCycle/CPU/rfile_rd1EX	00000000000000000000000000010000
/tb_SingleCycle/CPU/rfile_rd2EX	00000000000000000000000000010000
/tb_SingleCycle/CPU/pc_incrEX	00000000000000000000000000000100



Wave - Default

	Msgs
/tb_SingleCycle/CPU/rtEX	01010
/tb_SingleCycle/CPU/rdEX	00000
/tb_SingleCycle/CPU/shamtEX	00000
/tb_SingleCycle/CPU/RegDstEX	St1
/tb_SingleCycle/CPU/ALUSrcEX	St0
/tb_SingleCycle/CPU/BranchEX	St0
/tb_SingleCycle/CPU/MemReadEX	St0
/tb_SingleCycle/CPU/MemWriteEX	St0
/tb_SingleCycle/CPU/RegWriteEX	St1
/tb_SingleCycle/CPU/MemtoRegEX	St1
/tb_SingleCycle/CPU/JumpEX	St0
/tb_SingleCycle/CPU/ALUOpEX	10
/tb_SingleCycle/CPU/alu_outMEM	00000000000000000000000000000000
/tb_SingleCycle/CPU/rfile_rd2MEM	00000000000000000000000000010000
/tb_SingleCycle/CPU/rfile_rd1MEM	00000000000000000000000000010000
/tb_SingleCycle/CPU/rfile_wnMEM	00000
/tb_SingleCycle/CPU/ZeroMEM	St0
/tb_SingleCycle/CPU/BranchMEM	St0
/tb_SingleCycle/CPU/MemReadMEM	St0
/tb_SingleCycle/CPU/MemWriteMEM	St0
/tb_SingleCycle/CPU/RegWriteMEM	St1
/tb_SingleCycle/CPU/MemtoRegMEM	St1
/tb_SingleCycle/CPU/b_tgtMEM	00000000000000000000000001101000
Now	10200 ns
Cursor 1	50 ns



```
                                         Msgs
/tb_SingleCycle/CPU/jump_addrEX          00000001001010000000000001100100
/tb_SingleCycle/CPU/jump_addrMEM         00000001001010000000000001100100
/tb_SingleCycle/CPU/ALU/reset            St0
/tb_SingleCycle/CPU/ALU/clk              St1
/tb_SingleCycle/CPU/ALU/dataA            00000000000000000000000000010000
/tb_SingleCycle/CPU/ALU/dataB            00000000000000000000000000010000
/tb_SingleCycle/CPU/ALU/shamt            00000
/tb_SingleCycle/CPU/ALU/ALUOp            10
/tb_SingleCycle/CPU/ALU/Signal           011001
/tb_SingleCycle/CPU/ALU/nop              St1
/tb_SingleCycle/CPU/ALU/zero             St0
/tb_SingleCycle/CPU/ALU/Output           00000000000000000000000000000000
/tb_SingleCycle/CPU/ALU/temp             xxxxxx
/tb_SingleCycle/CPU/ALU/Signalto...      xxx
/tb_SingleCycle/CPU/ALU/Signalto...      011001
/tb_SingleCycle/CPU/ALU/Signalto...      011001
/tb_SingleCycle/CPU/ALU/Signalto...      011001
/tb_SingleCycle/CPU/ALU/ALUOut           00000000000000000000000000000000
/tb_SingleCycle/CPU/ALU/HiOut            xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
/tb_SingleCycle/CPU/ALU/LoOut            xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
/tb_SingleCycle/CPU/ALU/ShifterOut       00000000000000000000000000010000
/tb_SingleCycle/CPU/ALU/dataOut          00000000000000000000000000000000
/tb_SingleCycle/CPU/ALU/MULAns           00000000000000000000000000000000...
Now                                      10200 ns
Cursor 1                                 50 ns
```

5. 結論

這一次的 project 還是要求我們使用 modelsim 來熟悉如何使用硬體描述語言並且能夠在期中 project 做出的 ALU 的基礎下完成一個切 pipeline 的 CPU。在切 pipeline 時我們可以藉由 wave 圖清楚看到重疊執行，也能思考如何接線，了解後會讓我們之後更清楚如何去切更多階的處理器和對 pipeline 更加熟悉然後在替換 ALU 和新增老師指定的指令後，完成一個小型處理器並且因為有了這些過程讓我們能更了解上課時所不清楚的部分，相信我們能夠在計算機組織這堂課結束後，具備獨自製作簡單的處理器能力。最後，看到我們完成的小型處理器完成時跑出的 wave 圖，真的覺得很開心努力了那麼久終於有那麼一點成果，雖然過程中真的很辛苦很煩線一直接錯都覺得要寫不出來完蛋了。

6.心得感想

芃睿：這次的 final project 比上次的 midterm project 難了不少，在一開始還搞不太懂時根本不知道要從何開始做起，單純翻課本也得不出設計面的靈感。幸好老師在上課時有講了很多可能我們那裡容易犯錯、那裡容易出現問題，也跟其他進度比我們快的同學互相討論，最後瞭解了這次的 final project 到底要做甚麼，清楚了之後做起來就順手許多。

峻彥：這次的 final project 原本以為我們在擁有 midterm project 的經驗下、受過 midterm project 的洗禮，我們會好寫很多，直到開始要著手時，腦袋只有一片"我是誰我在哪"，根本不知道要幹嘛，老師給的題目說明也沒有很清楚說明該做什麼,不像期中 Project 的時候有給架構圖可以稍微參考一下知道線大概要怎麼接、還需要再撰寫哪些部分，所以真的是一頭霧水。一開始根本照著課本上做，結果發現 哇 一直出包。真的是花了很多時間去了解 NOP 等東西才把這份 project 生出來。

婕歆：這次的 final project 真的太難了！！！一開始根本毫無頭緒感覺根本沒有因為寫過期中 project 就覺得 final 比較好理解，像一隻無頭蒼蠅在到處打轉。幸好周遭很多比較厲害的同學都很願意分享自己的做法，同組的組員們也都很認真地把 project 生出來，而且在一些我可能比較搞不懂的地方也都很細心的教導我，最後在理解完整個 project 跟把圖畫出來後，一切就都豁然開朗了！！這次的圖真的有太多線拉一拉都不知道自己拉到哪去了，非常眼花撩亂，常常拉到一半忘記自己拉到哪了又重頭來一次，真的用道很暴躁是近視感覺增加了幾百度。

逸駿：難 真的難 真的好難，我除了難想不到其他的形容詞。一開始我覺得好像

跟 midterm project 有點關係，我應該可以，結果實際去開始看之後才發現是我太高估自己了，我根本不知道自己要幹嘛。可能是我在這之前的基礎相較於組員們比較不好，所以在一開始我光理解就花了很多時間，但他們還是不厭其煩有耐心的讓我搞懂，這次的設計經驗真的是一個很寶貴的經驗！

7. 分工

撰寫程式碼+書面報告:共同分工完成