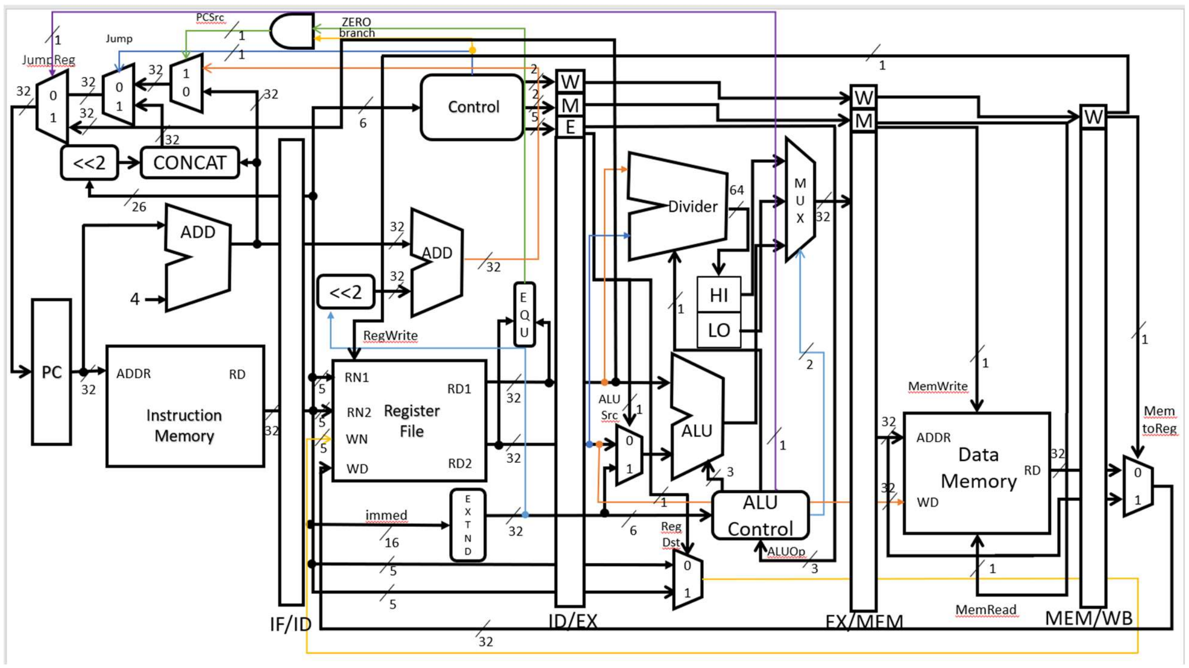
計算機組織期末報告

108 學年度第二學期
第 9 組
老師：朱守禮　老師
學生： 10727214 洪友祥
10727215 沈家丞
10727221 邱晨凱
10727248 鄭珮慈

一、背景

使用 Verilog HDL 與 Modelsim 模擬器，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 。

二、架構圖



三、方法

(1)ALU

沿用 Midterm Project 的設計，以 32 個 1-bit ALU 組成 32-bits ALU，具有 32-bits AND、OR、ADD、SUB、SLT 的功能，並加入 Shifter 來滿足 SRL 指令，而從 alu_crl 接收到的 crl 訊號共三位元，最低兩個位元當作 alu_one 的 sel，而最高為元為 binvert。

(2)Divider

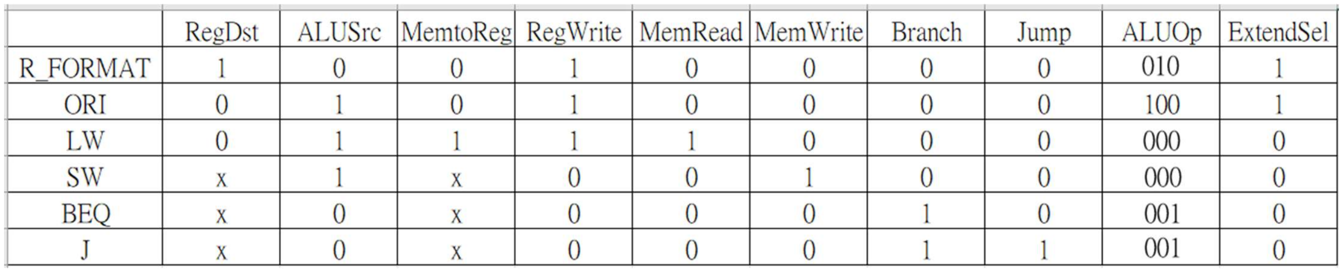
設計一個 32-bits 無號數除法器，設計時設置餘數和商數為 32 位元，除數為 64 位元，最一開始程式會先將記數的變數 i 設置為十進位的 0 ，再來重設商數，並將餘數設為被除數，除數向前移動 32 位元，並且 i 會+1，這是 i = 0 的狀況，其他在 i 還沒到 32 之前的動作都是，i 先+1，餘數-除數，接著判斷餘數是正是負，若是正就將商左移一位最後一位設 1，反之則左移一位最後一位設 0，並且餘數=餘數+除數。兩種情況之後除數都會又移 1，如跑完 33 次，i=33，就將商數存是後 32 位元的 dataOut，餘數存入前 32 位元的 dataOut。

(3)alu_crl

判斷 ALUOp 決定要值行的指令是 Add、Sub、Or、And、Slt，並將對應訊號輸入電路中。

(4)control_pipelined

利用輸入的指令代號 opcode，產生對應的控制訊號 ，並於內部做好暫存器 EX、MEM、WB 之設定。
控制訊號如下表所示 :



	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	Jump	ALUOp	ExtendSel
R_FORMAT	1	0	0	1	0	0	0	0	010	1
ORI	0	1	0	1	0	0	0	0	100	1
LW	0	1	1	1	1	0	0	0	000	0
SW	x	1	x	0	0	1	0	0	000	0
BEQ	x	0	x	0	0	0	1	0	001	0
J	x	0	x	0	0	0	1	1	001	0

(5)add32

實現 32 位元加法，此模組用於 PC 及 branch 的位址加法。

(6)branch_equ

將原本在 ALU 中做判斷的 beq 指令拉出，當 opcode 為 beq 且 Register File 兩輸出值相等時，便將 ZERO 設為 1，若否，則設為 0。

(7)memory

用於 Instruction Memory 與 Data Memory，主要根據 MemRead 和 MemWrite 去決定對指定記憶體位置的讀取、寫入。

(8)mips_pipelined

整合所有 module 以及處理訊號傳送。

(9)IF/ID

為 IF（Instruction Fetch）與 ID（Instruction Decode）之間的暫存器。當 clock 正緣觸發，若 rst 為 true，將輸出值設為 0；若 rst 為 false，將輸入值傳送給輸出值，藉此完成訊號傳遞。

(10)ID/EX

為 ID（Instruction Decode）與 EX（Execute Calculation）之間的暫存器。當 clock 正緣觸發，若 rst 為 true，將輸出值皆設為 0；若 rst 為 false，將輸入值傳送給輸出值，藉此完成訊號傳遞。

(11)EX/MEM

為 EX（Execute Calculation）與 MEM（Memory Access）之間的暫存器。當 clock 正緣觸發，若 rst 為 true，將輸出值皆設為 0；若 rst 為 false，將輸入值傳送給輸出值，藉此完成訊號傳遞。

(12)MEM/WB

為 MEM（Memory Access）與 WB（Write Back）之間的暫存器。當 clock 正緣觸發，若 rst 為 true，將輸出值皆設為 0；若 rst 為 false，將輸入值傳送給輸出值，藉此完成訊號傳遞。

(13)mux2

二對一多工器，用於選擇 ALU 的 operand2 輸入、PC 要 fetch 下一道指令的位址、欲寫回的暫存器以及內容。

(14)mux3

三對一多工器，用於選擇 EX 階的運算結果(Hi/Lo/ALU)。

(15)reg_file

用於 Register File，主要根據 RegWrite 決定暫存器是否寫入。

(16)reg32

32 位元暫存器，在此當作 PC 使用。

(17)sign extend

將 16 位元的值進行有號數擴充成 32 位元。

(18)unsign_extend

將 16 位元的值進行無號數擴充成 32 位元。

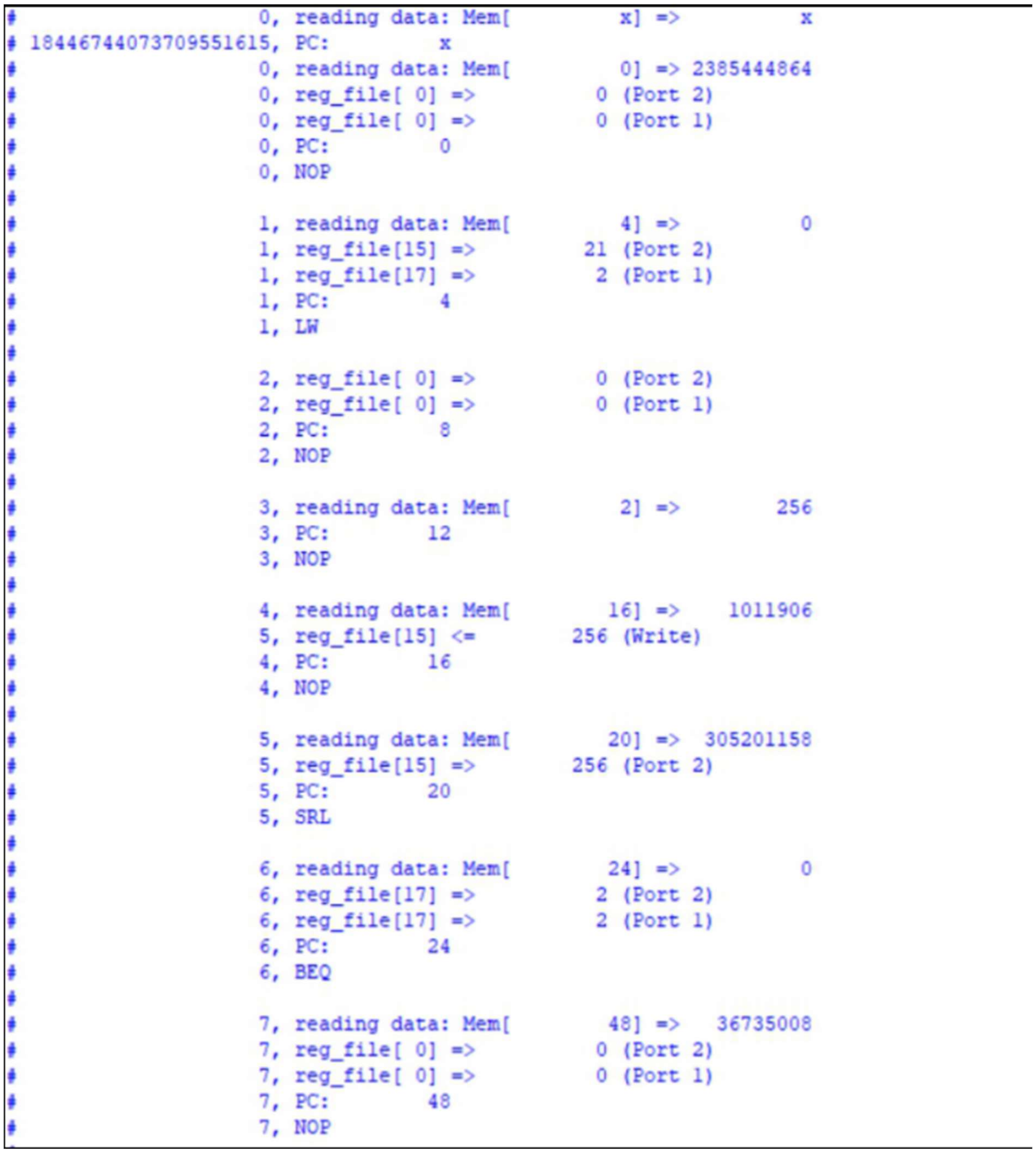
(19)tb_Pipelined

根據時脈不斷地執行程式碼，而當 CPU.funct 在不同值的時候，分別執行不同的指令。

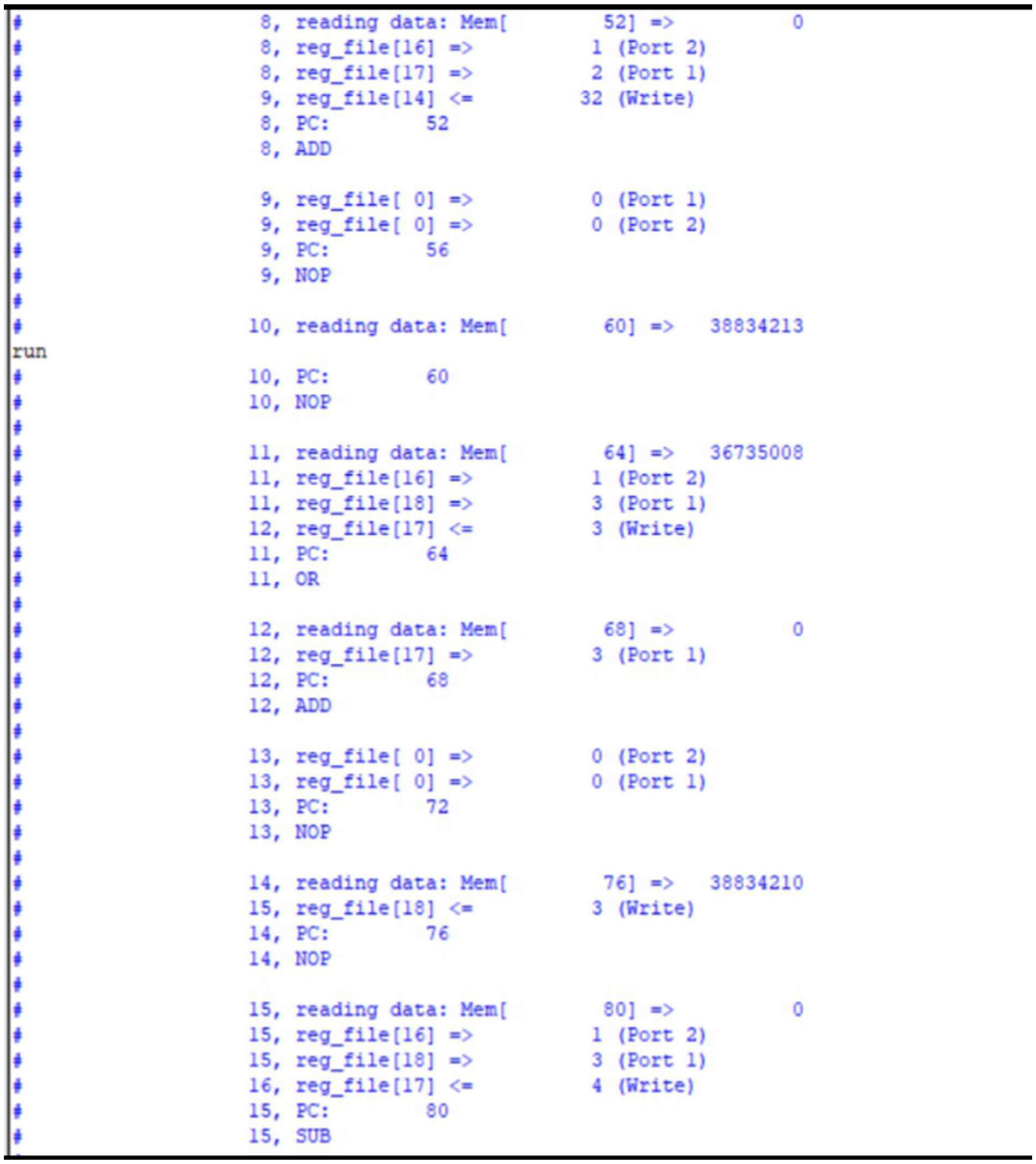
四、結果

我們根據以下指令來做測試

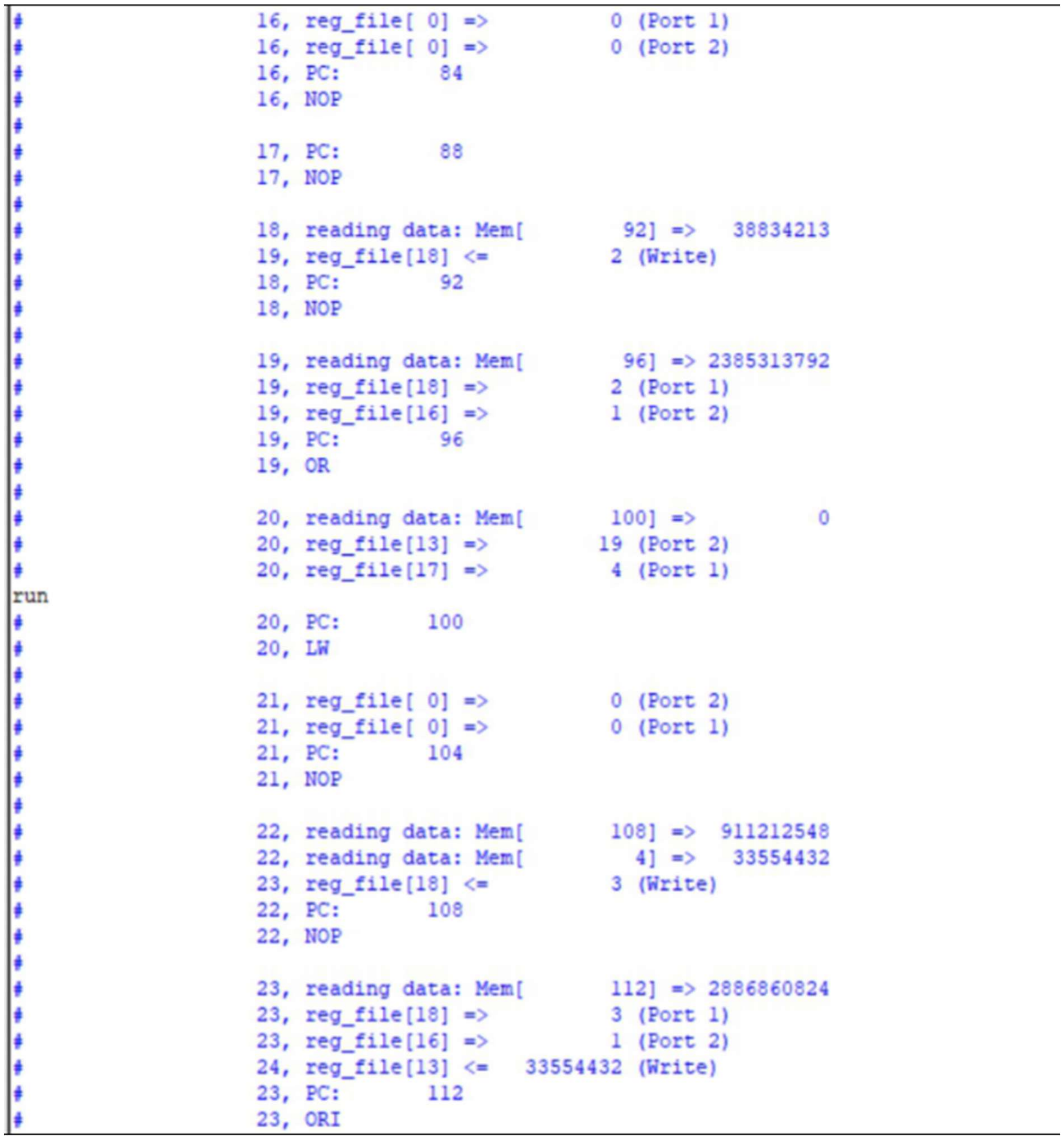
(1) lw $s1, $t7, 0
(2) srl $t7, $t6, 3
(3) beq $s1, $s2, 6
(4) add $s2, $s0, $s2
(5) sub $s2, $s0, $s2
(6) add $s1, $s0, $s1
(7) or $s2, $s0, $s2
(8) add $s1, $s0, $s1
(9) sub $s2, $s0, $s2
(10) or $s2, $s0, $s2
(11) lw $s1, $t5, 0
(12) ori $s2, $s0, 4
(13) sw $zero, $s2, 24
(14) divu $t7, $v1
(15) mfhi $s1
(16) mflo $s0



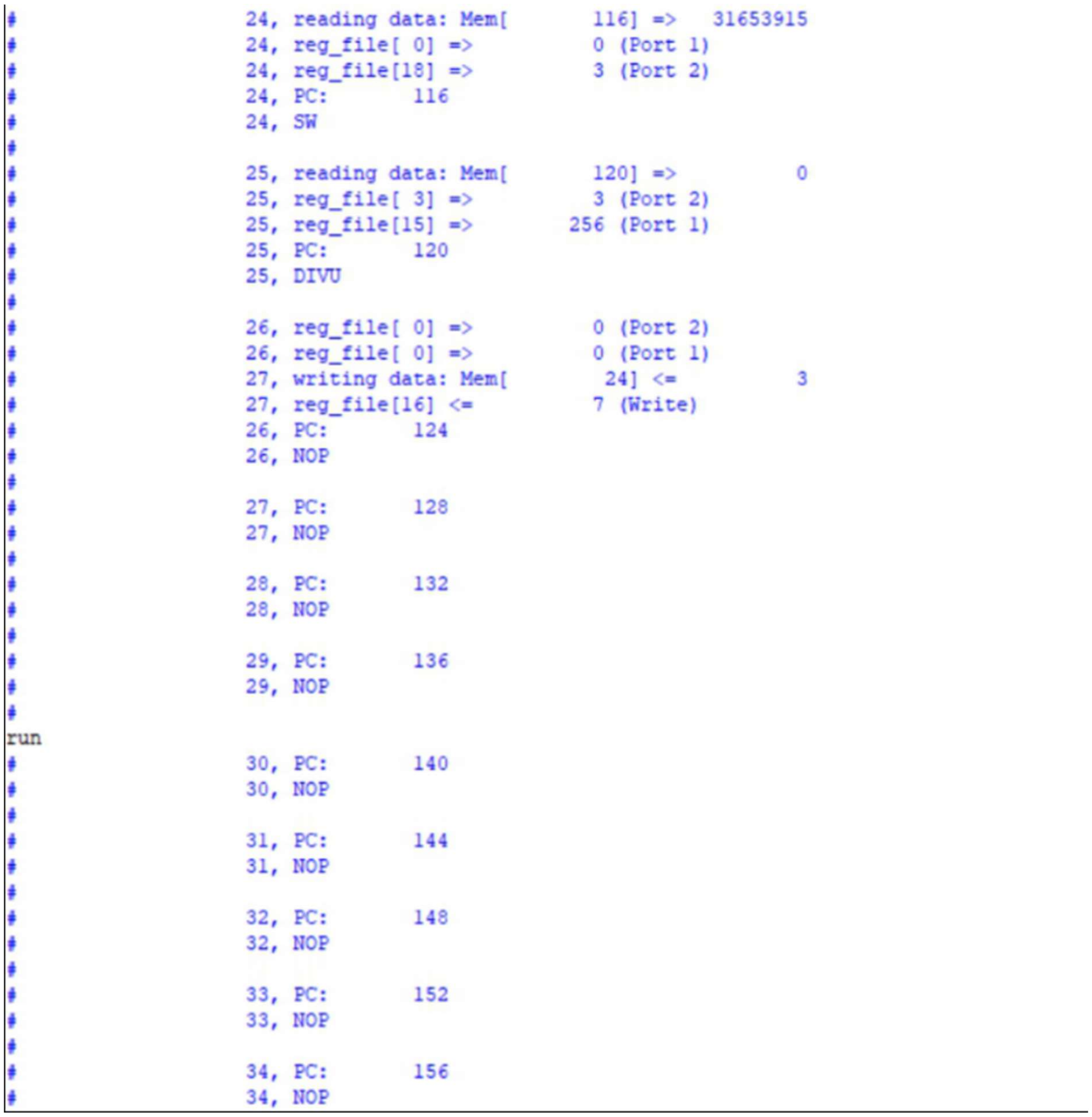
```
#                    0, reading data: Mem[         x] =>          x
# 18446744073709551615, PC:          x
#                    0, reading data: Mem[         0] => 2385444864
#                    0, reg_file[ 0] =>          0 (Port 2)
#                    0, reg_file[ 0] =>          0 (Port 1)
#                    0, PC:          0
#                    0, NOP
#
#                    1, reading data: Mem[         4] =>          0
#                    1, reg_file[15] =>         21 (Port 2)
#                    1, reg_file[17] =>          2 (Port 1)
#                    1, PC:          4
#                    1, LW
#
#                    2, reg_file[ 0] =>          0 (Port 2)
#                    2, reg_file[ 0] =>          0 (Port 1)
#                    2, PC:          8
#                    2, NOP
#
#                    3, reading data: Mem[         2] =>        256
#                    3, PC:         12
#                    3, NOP
#
#                    4, reading data: Mem[        16] =>    1011906
#                    5, reg_file[15] <=        256 (Write)
#                    4, PC:         16
#                    4, NOP
#
#                    5, reading data: Mem[        20] =>  305201158
#                    5, reg_file[15] =>        256 (Port 2)
#                    5, PC:         20
#                    5, SRL
#
#                    6, reading data: Mem[        24] =>          0
#                    6, reg_file[17] =>          2 (Port 2)
#                    6, reg_file[17] =>          2 (Port 1)
#                    6, PC:         24
#                    6, BEQ
#
#                    7, reading data: Mem[        48] =>   36735008
#                    7, reg_file[ 0] =>          0 (Port 2)
#                    7, reg_file[ 0] =>          0 (Port 1)
#                    7, PC:         48
#                    7, NOP
```



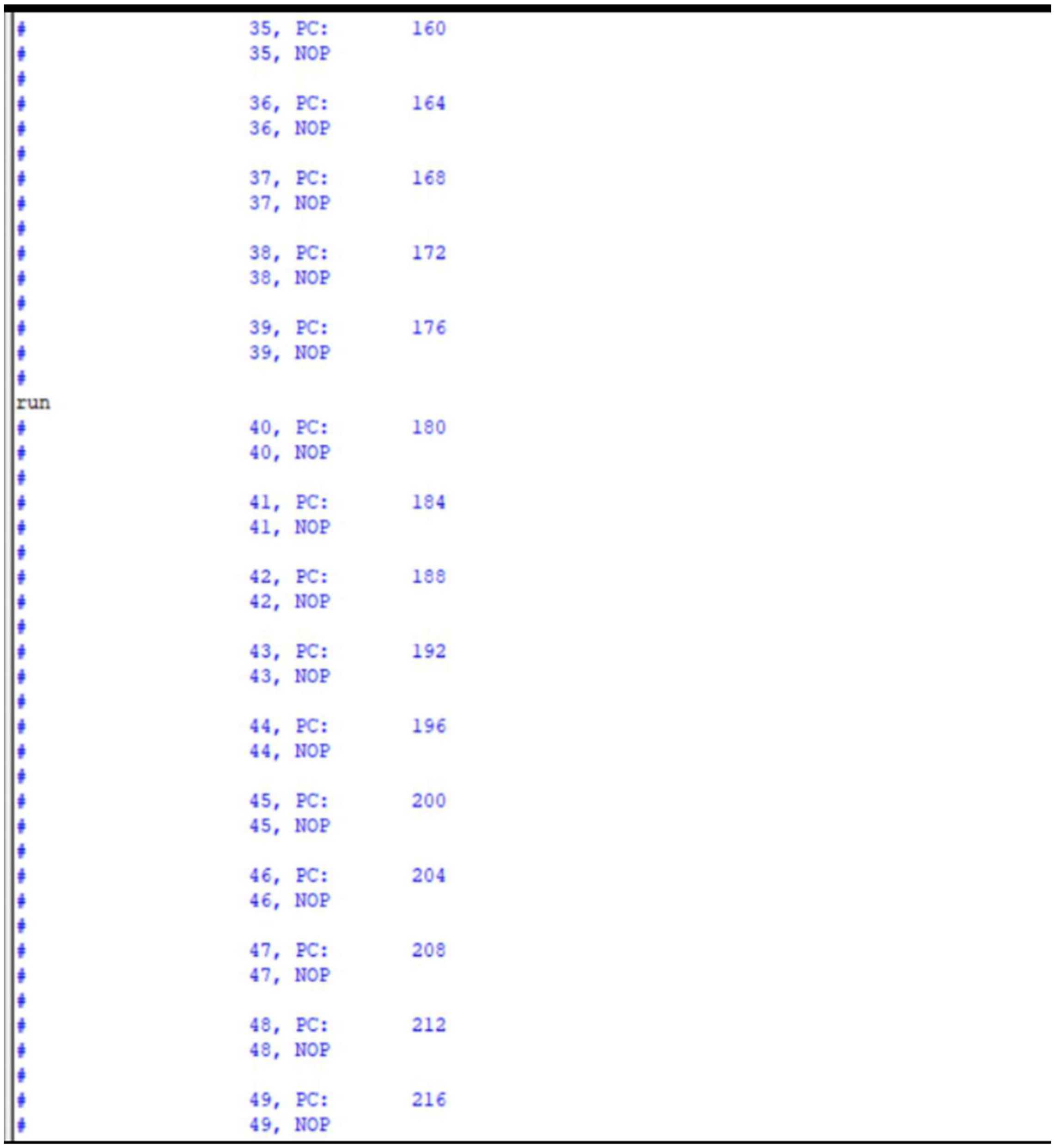
```
#                    8, reading data: Mem[        52] =>          0
#                    8, reg_file[16] =>          1 (Port 2)
#                    8, reg_file[17] =>          2 (Port 1)
#                    9, reg_file[14] <=         32 (Write)
#                    8, PC:        52
#                    8, ADD
#
#                    9, reg_file[ 0] =>          0 (Port 1)
#                    9, reg_file[ 0] =>          0 (Port 2)
#                    9, PC:        56
#                    9, NOP
#
#                   10, reading data: Mem[        60] =>   38834213
run
#                   10, PC:        60
#                   10, NOP
#
#                   11, reading data: Mem[        64] =>   36735008
#                   11, reg_file[16] =>          1 (Port 2)
#                   11, reg_file[18] =>          3 (Port 1)
#                   12, reg_file[17] <=          3 (Write)
#                   11, PC:        64
#                   11, OR
#
#                   12, reading data: Mem[        68] =>          0
#                   12, reg_file[17] =>          3 (Port 1)
#                   12, PC:        68
#                   12, ADD
#
#                   13, reg_file[ 0] =>          0 (Port 2)
#                   13, reg_file[ 0] =>          0 (Port 1)
#                   13, PC:        72
#                   13, NOP
#
#                   14, reading data: Mem[        76] =>   38834210
#                   15, reg_file[18] <=          3 (Write)
#                   14, PC:        76
#                   14, NOP
#
#                   15, reading data: Mem[        80] =>          0
#                   15, reg_file[16] =>          1 (Port 2)
#                   15, reg_file[18] =>          3 (Port 1)
#                   16, reg_file[17] <=          4 (Write)
#                   15, PC:        80
#                   15, SUB
```



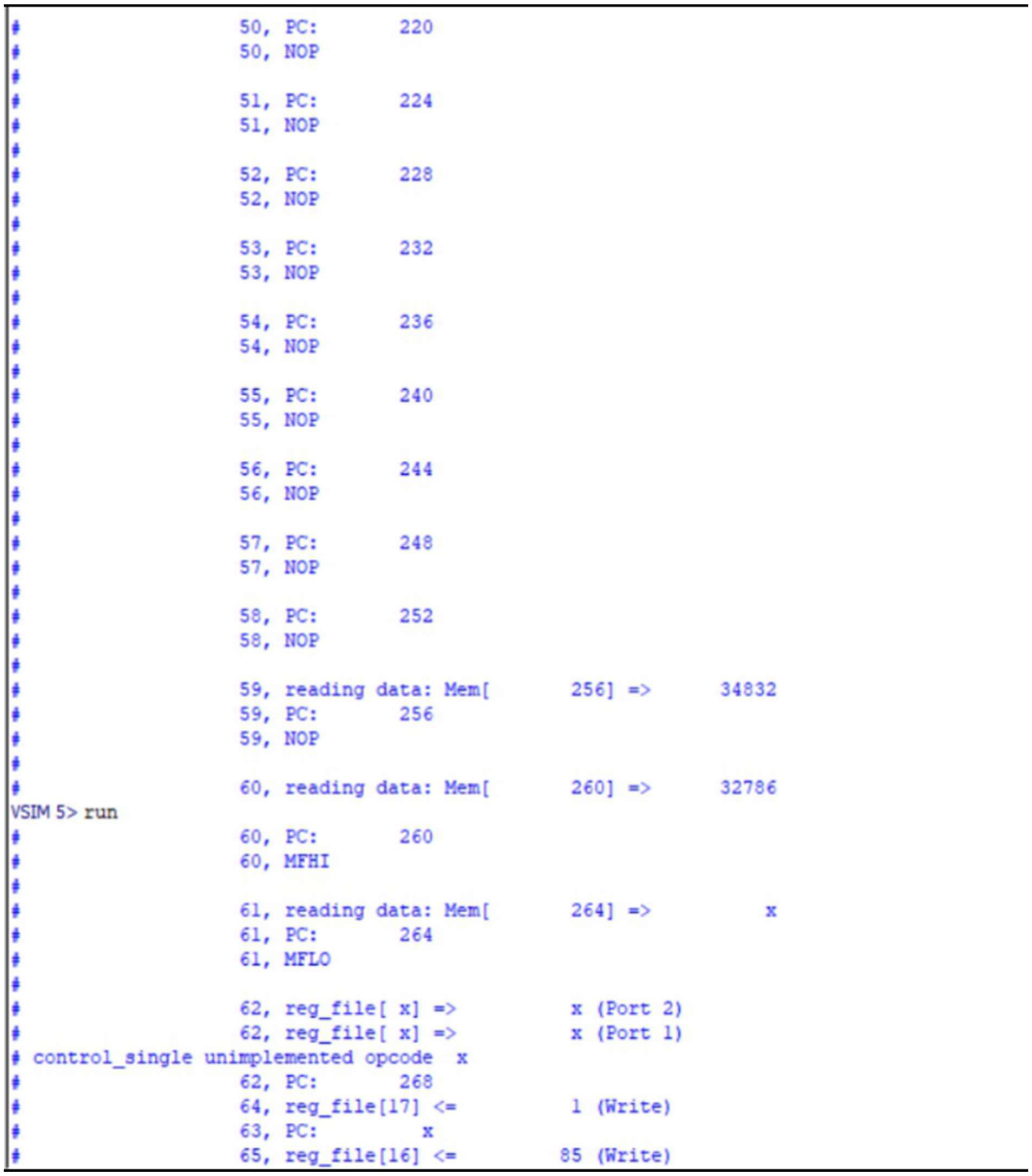
```
#                    16, reg_file[ 0] =>          0 (Port 1)
#                    16, reg_file[ 0] =>          0 (Port 2)
#                    16, PC:        84
#                    16, NOP
#
#                    17, PC:        88
#                    17, NOP
#
#                    18, reading data: Mem[        92] =>   38834213
#                    19, reg_file[18] <=          2 (Write)
#                    18, PC:        92
#                    18, NOP
#
#                    19, reading data: Mem[        96] => 2385313792
#                    19, reg_file[18] =>          2 (Port 1)
#                    19, reg_file[16] =>          1 (Port 2)
#                    19, PC:        96
#                    19, OR
#
#                    20, reading data: Mem[       100] =>          0
#                    20, reg_file[13] =>         19 (Port 2)
#                    20, reg_file[17] =>          4 (Port 1)
run
#                    20, PC:       100
#                    20, LW
#
#                    21, reg_file[ 0] =>          0 (Port 2)
#                    21, reg_file[ 0] =>          0 (Port 1)
#                    21, PC:       104
#                    21, NOP
#
#                    22, reading data: Mem[       108] =>  911212548
#                    22, reading data: Mem[         4] =>   33554432
#                    23, reg_file[18] <=          3 (Write)
#                    22, PC:       108
#                    22, NOP
#
#                    23, reading data: Mem[       112] => 2886860824
#                    23, reg_file[18] =>          3 (Port 1)
#                    23, reg_file[16] =>          1 (Port 2)
#                    24, reg_file[13] <=   33554432 (Write)
#                    23, PC:       112
#                    23, ORI
```



```
#                    24, reading data: Mem[       116] =>   31653915
#                    24, reg_file[ 0] =>          0 (Port 1)
#                    24, reg_file[18] =>          3 (Port 2)
#                    24, PC:       116
#                    24, SW
#
#                    25, reading data: Mem[       120] =>          0
#                    25, reg_file[ 3] =>          3 (Port 2)
#                    25, reg_file[15] =>        256 (Port 1)
#                    25, PC:       120
#                    25, DIVU
#
#                    26, reg_file[ 0] =>          0 (Port 2)
#                    26, reg_file[ 0] =>          0 (Port 1)
#                    27, writing data: Mem[        24] <=          3
#                    27, reg_file[16] <=          7 (Write)
#                    26, PC:       124
#                    26, NOP
#
#                    27, PC:       128
#                    27, NOP
#
#                    28, PC:       132
#                    28, NOP
#
#                    29, PC:       136
#                    29, NOP
#
run
#                    30, PC:       140
#                    30, NOP
#
#                    31, PC:       144
#                    31, NOP
#
#                    32, PC:       148
#                    32, NOP
#
#                    33, PC:       152
#                    33, NOP
#
#                    34, PC:       156
#                    34, NOP
```

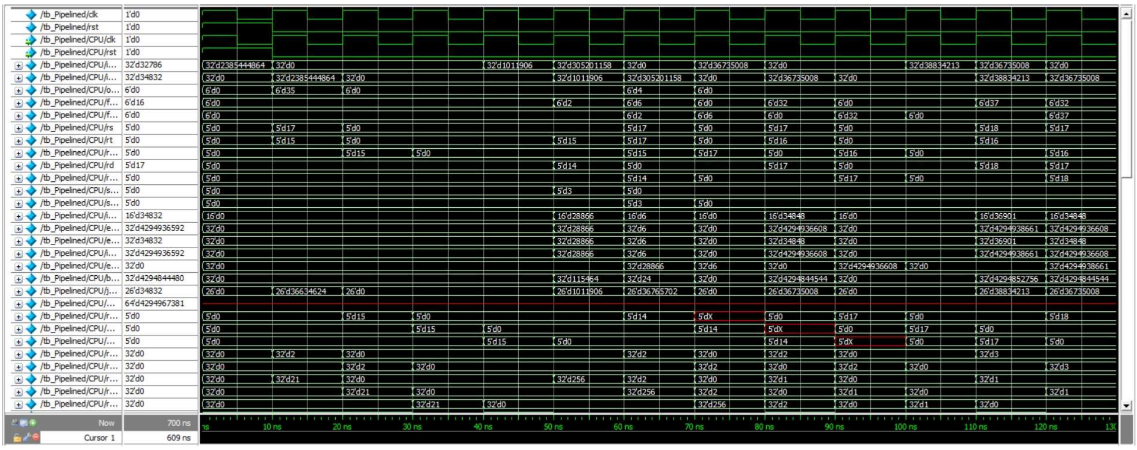


```
#                    35, PC:       160
#                    35, NOP
#
#                    36, PC:       164
#                    36, NOP
#
#                    37, PC:       168
#                    37, NOP
#
#                    38, PC:       172
#                    38, NOP
#
#                    39, PC:       176
#                    39, NOP
#
run
#                    40, PC:       180
#                    40, NOP
#
#                    41, PC:       184
#                    41, NOP
#
#                    42, PC:       188
#                    42, NOP
#
#                    43, PC:       192
#                    43, NOP
#
#                    44, PC:       196
#                    44, NOP
#
#                    45, PC:       200
#                    45, NOP
#
#                    46, PC:       204
#                    46, NOP
#
#                    47, PC:       208
#                    47, NOP
#
#                    48, PC:       212
#                    48, NOP
#
#                    49, PC:       216
#                    49, NOP
```

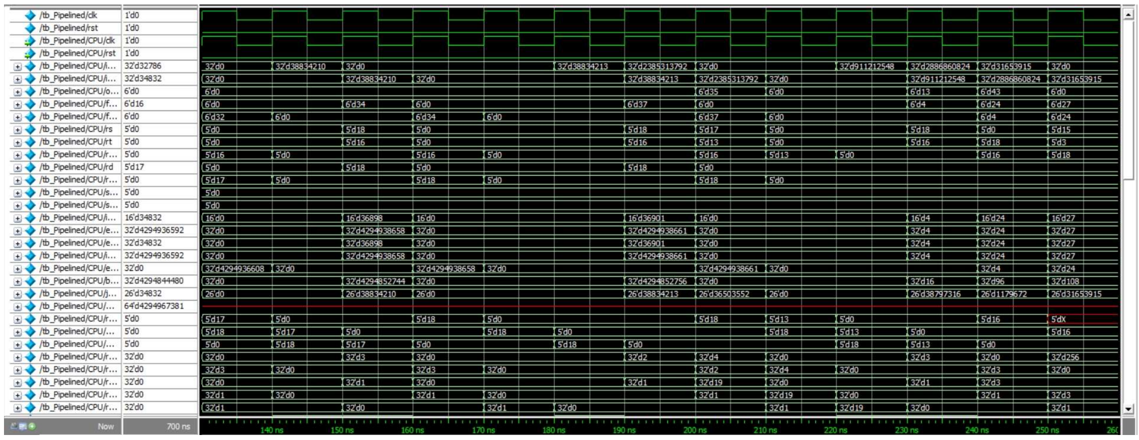


```
#                    50, PC:       220
#                    50, NOP
#
#                    51, PC:       224
#                    51, NOP
#
#                    52, PC:       228
#                    52, NOP
#
#                    53, PC:       232
#                    53, NOP
#
#                    54, PC:       236
#                    54, NOP
#
#                    55, PC:       240
#                    55, NOP
#
#                    56, PC:       244
#                    56, NOP
#
#                    57, PC:       248
#                    57, NOP
#
#                    58, PC:       252
#                    58, NOP
#
#                    59, reading data: Mem[       256] =>      34832
#                    59, PC:       256
#                    59, NOP
#
#                    60, reading data: Mem[       260] =>      32786
VSIM 5> run
#                    60, PC:       260
#                    60, MFHI
#
#                    61, reading data: Mem[       264] =>          x
#                    61, PC:       264
#                    61, MFLO
#
#                    62, reg_file[ x] =>          x (Port 2)
#                    62, reg_file[ x] =>          x (Port 1)
# control_single unimplemented opcode  x
#                    62, PC:       268
#                    64, reg_file[17] <=          1 (Write)
#                    63, PC:         x
#                    65, reg_file[16] <=         85 (Write)
```

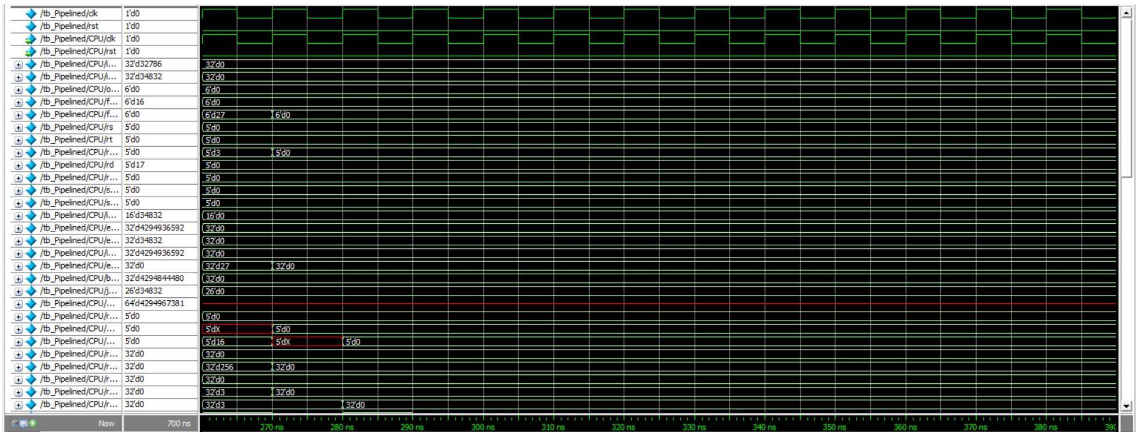
上方為 16 道指令於 terminal 輸出的結果，能觀察出指令之暫存器數值與輸出結果相符合。下方為 16 道指令顯示之 waveform 圖形。



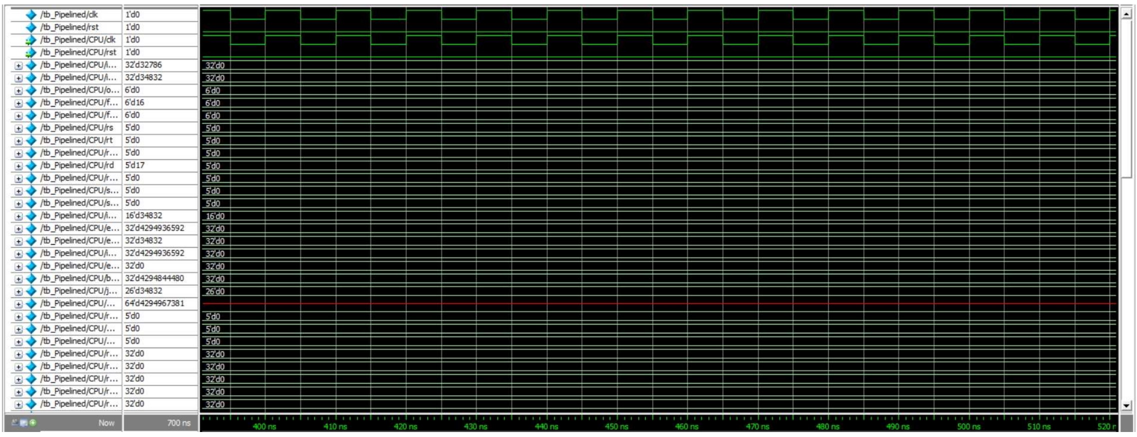
/tb_Pipelined/clk	1'd0
/tb_Pipelined/rst	1'd0
/tb_Pipelined/CPU/clk	1'd0
/tb_Pipelined/CPU/rst	1'd0
/tb_Pipelined/CPU/i...	32'd32786
/tb_Pipelined/CPU/i...	32'd34832
/tb_Pipelined/CPU/o...	6'd0
/tb_Pipelined/CPU/f...	6'd16
/tb_Pipelined/CPU/f...	6'd0
/tb_Pipelined/CPU/rs	5'd0
/tb_Pipelined/CPU/rt	5'd0
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/rd	5'd17
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/i...	16'd34832
/tb_Pipelined/CPU/e...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd34832
/tb_Pipelined/CPU/i...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd0
/tb_Pipelined/CPU/b...	32'd4294844480
/tb_Pipelined/CPU/j...	26'd34832
/tb_Pipelined/CPU/...	64'd4294967381
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
Now	700 ns
Cursor 1	609 ns



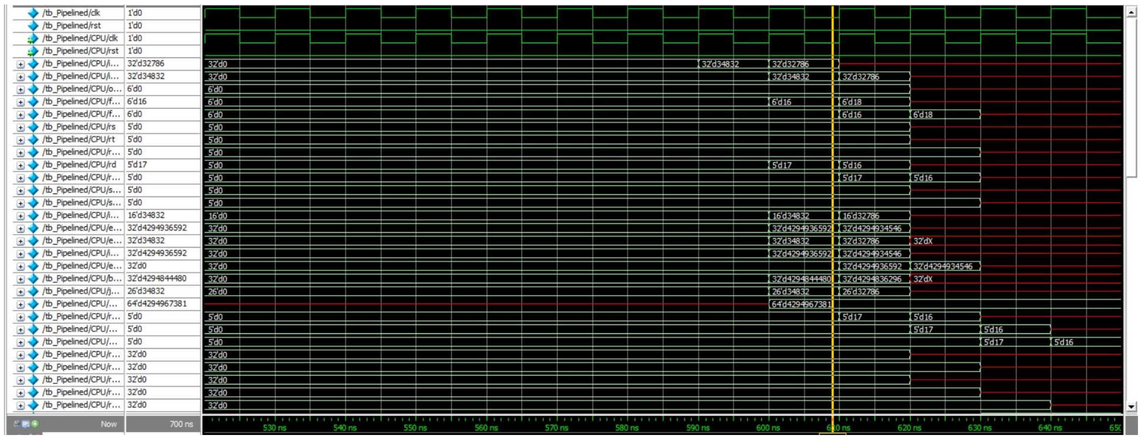
/tb_Pipelined/clk	1'd0
/tb_Pipelined/rst	1'd0
/tb_Pipelined/CPU/clk	1'd0
/tb_Pipelined/CPU/rst	1'd0
/tb_Pipelined/CPU/i...	32'd32786
/tb_Pipelined/CPU/i...	32'd34832
/tb_Pipelined/CPU/o...	6'd0
/tb_Pipelined/CPU/f...	6'd16
/tb_Pipelined/CPU/f...	6'd0
/tb_Pipelined/CPU/rs	5'd0
/tb_Pipelined/CPU/rt	5'd0
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/rd	5'd17
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/i...	16'd34832
/tb_Pipelined/CPU/e...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd34832
/tb_Pipelined/CPU/i...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd0
/tb_Pipelined/CPU/b...	32'd4294844480
/tb_Pipelined/CPU/j...	26'd34832
/tb_Pipelined/CPU/...	64'd4294967381
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
Now	700 ns



/tb_Pipelined/clk	1'd0
/tb_Pipelined/rst	1'd0
/tb_Pipelined/CPU/clk	1'd0
/tb_Pipelined/CPU/rst	1'd0
/tb_Pipelined/CPU/i...	32'd32786
/tb_Pipelined/CPU/i...	32'd34832
/tb_Pipelined/CPU/o...	6'd0
/tb_Pipelined/CPU/f...	6'd16
/tb_Pipelined/CPU/f...	6'd0
/tb_Pipelined/CPU/rs	5'd0
/tb_Pipelined/CPU/rt	5'd0
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/rd	5'd17
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/i...	16'd34832
/tb_Pipelined/CPU/e...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd34832
/tb_Pipelined/CPU/i...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd0
/tb_Pipelined/CPU/b...	32'd4294844480
/tb_Pipelined/CPU/j...	26'd34832
/tb_Pipelined/CPU/...	64'd4294967381
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
Now	700 ns



/tb_Pipelined/clk	1'd0
/tb_Pipelined/rst	1'd0
/tb_Pipelined/CPU/clk	1'd0
/tb_Pipelined/CPU/rst	1'd0
/tb_Pipelined/CPU/i...	32'd32786
/tb_Pipelined/CPU/i...	32'd34832
/tb_Pipelined/CPU/o...	6'd0
/tb_Pipelined/CPU/f...	6'd16
/tb_Pipelined/CPU/f...	6'd0
/tb_Pipelined/CPU/rs	5'd0
/tb_Pipelined/CPU/rt	5'd0
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/rd	5'd17
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/s...	5'd0
/tb_Pipelined/CPU/i...	16'd34832
/tb_Pipelined/CPU/e...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd34832
/tb_Pipelined/CPU/i...	32'd4294936592
/tb_Pipelined/CPU/e...	32'd0
/tb_Pipelined/CPU/b...	32'd4294844480
/tb_Pipelined/CPU/j...	26'd34832
/tb_Pipelined/CPU/...	64'd4294967381
/tb_Pipelined/CPU/r...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/...	5'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
/tb_Pipelined/CPU/r...	32'd0
Now	700 ns



五、討論

Q1.　如何在做 branch 指令的運算時，下一道指令能夠 fetch 到正確的?

Ans: 在 register file 的兩個輸出 rs、rt 連接一個判斷是否相等的模組，在 ID 階就對立即值做有號數擴充，將是否做 branch 跳躍的判斷提前至 ID 階，stall 一個 cycle 不 fetch 指令，就能達到正確運算 branch 指令的效果。

Q2. 如何測試結果正不正確?

Ans: 必須將 Register file、Data memory 以及 instruction memory 設定完成後，一道一道檢驗輸出結果，追蹤該指令的 datapath，將訊號顯示於 waveform 後，可曉得在哪一個模組中的訊號不如預期，再針對該模組做修正。

Q3. 除法器做運算時，該如何處理 hazard 發生的問題 ?

Ans: 我們在做除法運算時，會在後方加入 34 個 nop 指令，使指令推遲避免發生 hazard 。

六、結論(心得)

這次的 Final project，基本上是統整整個學期的東西了，延續了期中 project 較為簡單的除法器、位移器、ALU 等部分。而這次最困難的不外乎就是切 Pipeline 了，要去把所有指令，去切成五個部分，來達成同步執行指令的事情，以加速程式的執行。而這次任務主要的東西，基本上都是在切 Pipeline，所以後來我們發現分工其實很多人分到的程式碼部分，是可以不用做更改的，可能是因為我們一開始以為分工可以跟期中 Project 一樣，只要更改程式碼達成教授的要求，那就可以了，結果後來才發現全然不同，需要自行設計一個架構圖，來執行老師所要求需要的所有指令、規範，我們最後也終於在幾乎快壓哨的時候完成了我們的 Final project。

七、未來展望

計算機組織這門課，我們從一開始陸續接觸了電腦系統組織，再來了解些 MIPS 有關的電腦語言，也因為經過之前的組合語言、嵌入式系統，更容易掌握這門課，接著陸續的 ALU 乘法器除法器，再到整個 CPU ，我們其實學得挺多的，希望未來的我們能學到更多 MIPS 指令，在硬體方面上能夠更熟悉。