Final Project

班別:資工二甲

組別:40

10727124 劉宇廷

10727131 陳彥綸

10727153 呂宜鴻

一.設計重點說明

add32

32-bit加法。

ALU

設定暫存器temp暫時儲存答案，首先判斷如果signal=ADD，設定ctr=0，否則ctr=1，以Ripple-Carry的進位方式，依序呼叫ALU\_bit連接31個bit，最後一個位元執行ALU\_Set，目的是增加SLT的判斷，最後將結果temp傳給dataOut，並利用dataOut設定zero的值。

ALU\_bit

設定3個wire(tempAND/tempOR/temp)，tempAND儲存a&b的結果，tempOR儲存a|b的結果，temp儲存FA處理的結果，最後判斷signal=AND，tempAND傳給sum；signal=OR，tempOR傳給sum；signal=SLT，less傳給sum；Signal=ADD/SUB，temp傳給sum。

FA

利用gate-level的方式，將輸入的位元做full-adder的處理，減法也可利用相同概念，利用A-B=A+(-B)做相同處理。

alu\_ctl

分別設定兩暫存器(temp/counter)，當Funct訊號為AND/OR/ADD/SUB/SLT/SRL時，將指令對應的值傳給temp；特別處理當signal訊號為DIV時，首先將counter歸零，判斷當posedge時counter++，並處理當counter=33時，temp設為6'b111111，代表開啟HiLo暫存器，給除法器放值進去。

ALU\_Set

功能與ALU\_bit差不多，特別處理最後一個位元，當signal=SLT，將temp值傳給set，在ALU中最後一個位元的set會更改第一個位元的set，完成dataA<dataB時，結果為{31'b0,1'b1}。

Control\_single

根據輸入的指令opcode，判斷指令類型(R\_FORMAT、ORI、LW、SW、BEQ、J)，並產生對應的控制訊號(RegDst、ALUSrc、MemtoReg、RegWrite、MemRead、MemWrite、Branch、Jump、ALUOp)。

Divider

當signal為DIVU時，rem設定為32'b0與dataA的組合，divr為dataB與32'b0的組合，之後當posedge時，判斷如果reset=1，初始化暫存結果的temp和儲存商數的quot，其餘情況先將rem減去divr，並判斷rem的最高位元是否為0，若是，代表rem>=0，quot左移一位並設定LSB=1；若否，代表rem<0，quot左移一位並設定LSB=0，將divr持續往右移一位，最後判斷Signal=OUT時，代表除法器執行結束，將temp值更改為quot與rem[31:0]的結合，代表前32位元儲存商數，後32位元儲存餘數，最後將temp值傳給dataOut。

HiLO

主要功能是將64位元的除法器結果DivAns，分成HiOut/LoOut，前者是

DivAns[31:0]代表餘數，後者是DivAns[63:32]代表商數 。

shifter

將輸入的dataA右移dataB字元，分別建立temp1~5表示每移一階所暫存的結果，首先判斷如果reset=1，dataOut歸零，其餘情況從第一階的dataA[0]開始，依序判斷如果dataB[0]為1，值為dataA[1]，否則維持原值，依序執行將結果存至temp1；第二階從temp1[0]開始，依序判斷如果dataB[1]為1，值為temp[2]，否則維持原值，依序執行並將結果存至temp2，以此類推，最後判斷如果dataB[4]為1，temp4[0]值為temp4[16]，否則維持原值，並將結果存至temp5，再將temp5傳給dataOut完成移位。

MUX

完成上述功能後的結果，利用MUX做最後的輸出判斷，如果signal=AND/OR/ADD/SUB/SLT，dataOut為ALUOut；signal=SRL，dataOut為Shifter；signal=MFHI，dataOut為HiOut；Signal=MFLO，dataOut為LoOut。

mux2

根據sel判斷，如果sel=1，將b的值傳給y；如果sel=0，將a的值傳給y。

sign\_extend

將讀入的16-bit input延伸為32-bit做輸出，第17~32 bit為input的第16-bit位元的延伸。

reg32

作為pc使用，首先判斷如果rst為1，就將d\_out設為32’b0；若rst為0且en\_reg為1，即代表暫存器可寫入，就將d\_in寫進d\_out。

reg\_file

32-bit register file，用RegWrite判斷暫存器是否可寫入，RN1、RN2分別為rs、rt欲讀取的暫存器編號，WN為欲寫入的暫存器編號，WD為寫入暫存器的資料，RD1為rs暫存器所讀取的資料，RD2為 rt暫存器所讀取的資料，將file\_array的RN1、RN2位置中的資料放入RD1、RD2中，若RegWrite為1且WN⁠≠0，就將WD寫入file\_array的WN位置。

memory

先建立1KB的記憶體大小，分別利用MemRead控制記憶體是否可讀取、MemWrite控制記憶體是否可寫入，若MemRead=1，至指定記憶體位置讀取資料；若MemWrite=1，將資料寫入指定記憶體位置。

IF\_ID

介於stage1到stage2之間的pipeline register，將stage1讀入的pc\_incr傳至IF\_ID\_Reg\_pc\_incr代表program counter，RD傳至IF\_ID\_Reg\_RD代表讀入的指令內容。

ID\_EX

介於stage2到stage3之間的pipeline register，將stage2得到的資料輸出到暫存器中，內容包含Control訊號:ALUOp、ALUSrc、Branch、Jump、MemRead、MemWrite、MemtoReg、RegDst、RegWrite，以及指令內容rd、rt、RD1、RD2，還有pc\_incr。

EX\_MEM

介於stage3到stage4之間的pipeline register，將stage3得到的資料輸出到暫存器中，內容包含Control訊號:Branch、Jump、MemRead、MemWrite、MemtoReg、RegWrite，以及ALU計算結果值、WD、WN，還有zero值。

MEM\_WB

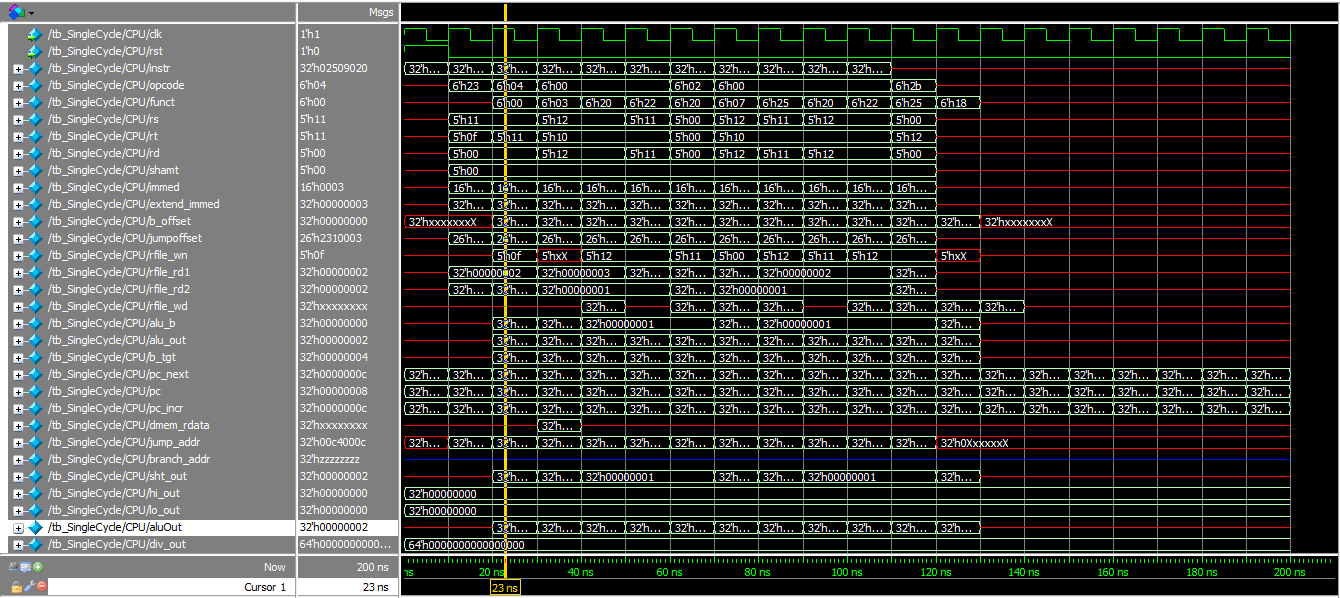
介於stage4到stage5之間的pipeline register，將stage4得到的資料輸出到暫存器中，內容包含Control訊號: MemtoReg、RegWrite，以及ALU計算結果值、RD、WN。

mip\_test

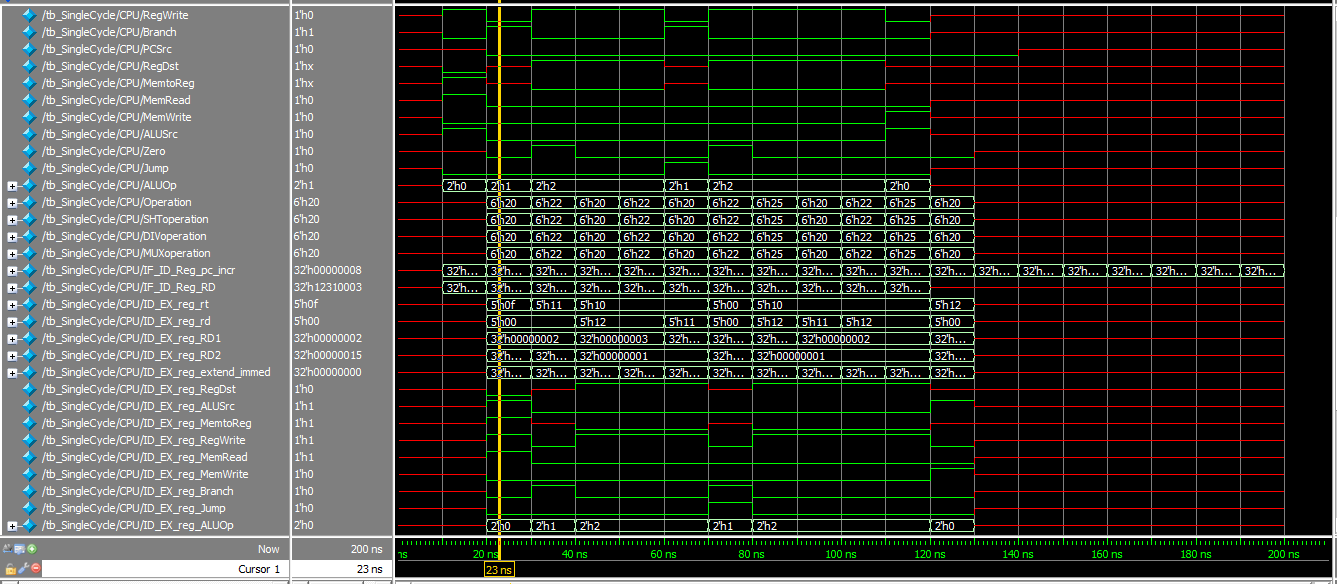
用來建立所需的module，並定義各種接線，使用於各module中。

二.Modelsim驗證結果與Waveform輸出圖形

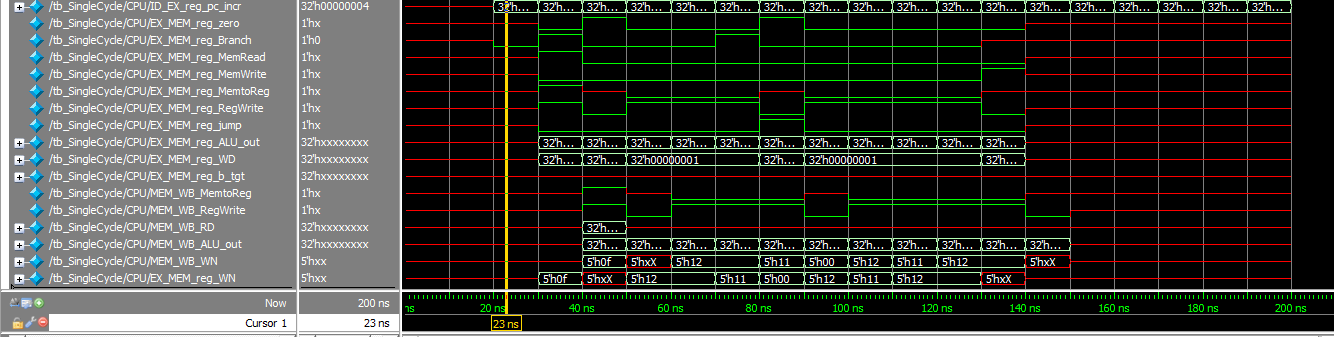
CPU(Waveform):



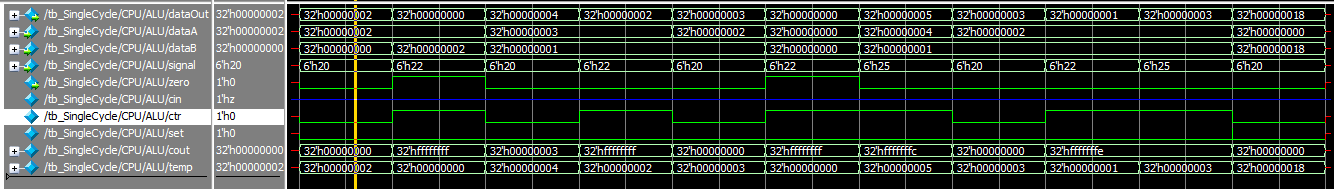
圖(一)



圖(二)

圖(三)

在CPU中，我們整合了所有的datapath的執行，從instructiion file取出指令，分割成opcode、rs、rd、rt、shamt、funct、immed、jumpoffset等部分，然後控制四個pipeline register來傳遞R-Type、I-Type、J-Type等指令所需的資料，圖(一)到圖(三)為我們執行lw、beq、add、sub、add、j、or、add、sub、or、sw這十一道指令所產生的CPU waveform圖。

ALU(waveform)

dataOut共分11個區塊:

第一區塊: lw ― 輸出為一位址和零相加

第二區塊: beq ― 輸出為零，並把Zero設為 1

第三區塊: add ― 輸出為 3 + 1 = 4

第四區塊: sub ― 輸出為 3 – 1 = 2

第五區塊: add ― 輸出為 2 + 1 = 3

第六區塊: j ― 輸出為零，並把Zero設為 1

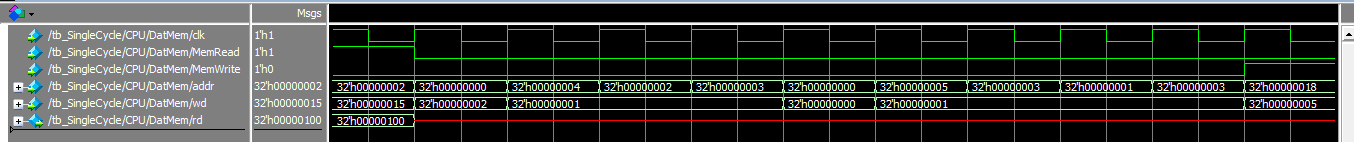
第七區塊: or ― 輸出為 4 || 1 = 5

第八區塊: add ― 輸出為 2 + 1 = 3

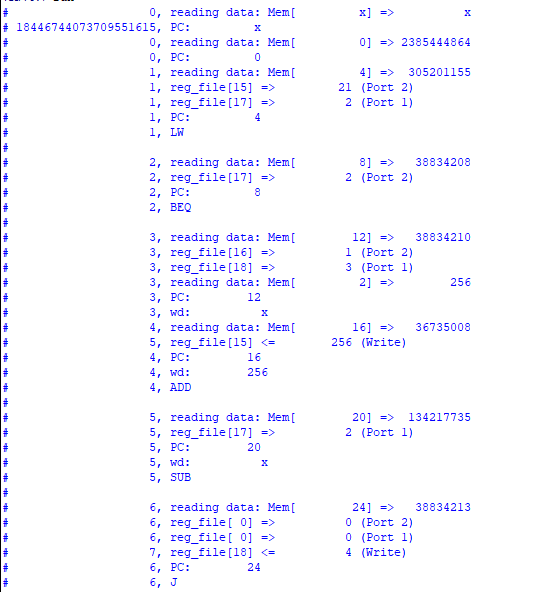
第九區塊: sub ― 輸出為 2 – 1 = 1

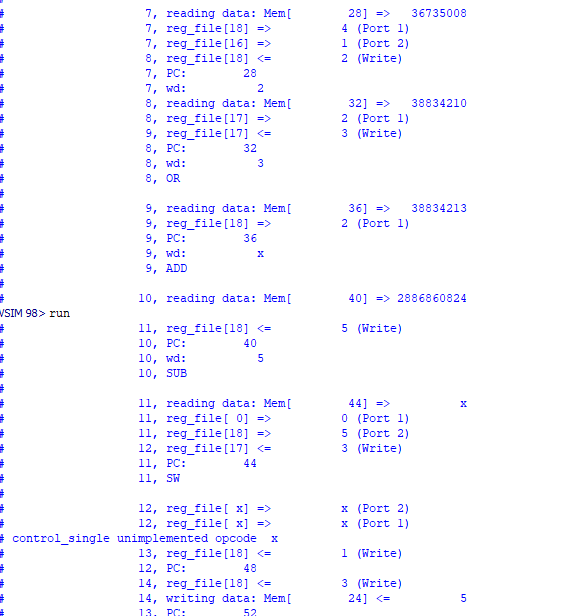
第十區塊: or ― 輸出為 2 || 1 = 3

第十一區塊: sw ― 輸出為一位址和零相加

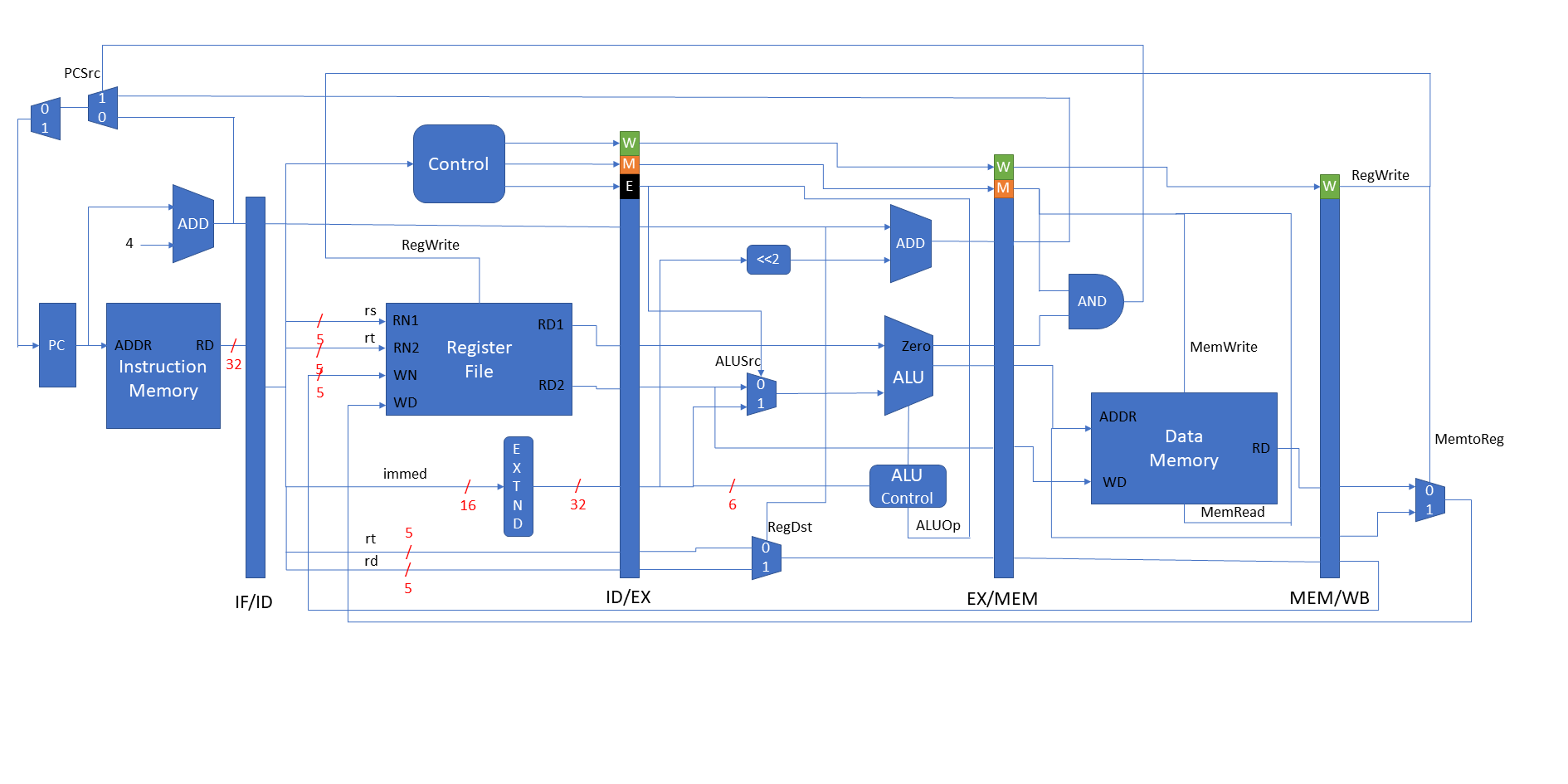
dataMemory(Waveform):

從圖中可看出在第一個clk時因為指令是lw，所以MemRead為1並且會輸出rd，而在最後一個clk時因為指令是sw，因此MemWrite為1。

驗證結果:

以上兩圖為執行11道指令的modelsim驗證結果。

三.Datapath 與詳細架構圖



四.心得感想

這次的CPU pipeline設計難度非常高，經過一番研究後設計出初步的CPU，能成功執行R-Type、I-Type多道指令，包含加減法、AND、OR、邏輯右移還有slt的判斷，可惜當遇到branch、jump、除法指令時，程式會遇到hazard的情形，導致資料內容有誤，最終執行結果出現異常。

五.各組員分工方式與負責項目

程式設計->劉宇廷、陳彥綸、呂宜鴻

設計重點說明->劉宇廷、呂宜鴻

Datapath架構圖->劉宇廷

Modelsim驗證結果與Waveform輸出圖形->陳彥綸

心得感想->呂宜鴻