計算機組織 Project2

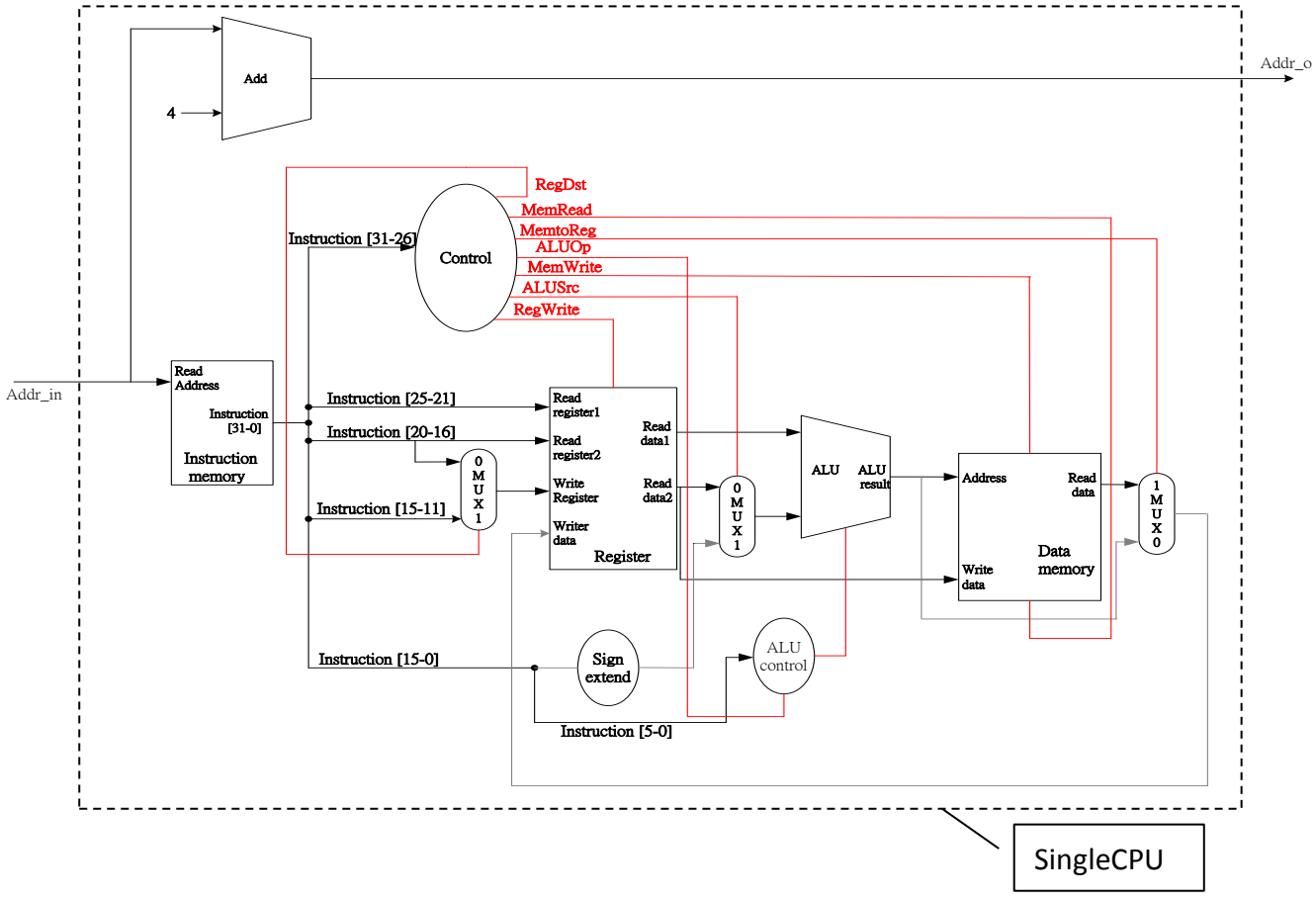
◎作業目的：

實作single cycle processor

◎階段 1：實做 R-type 指令(35%)

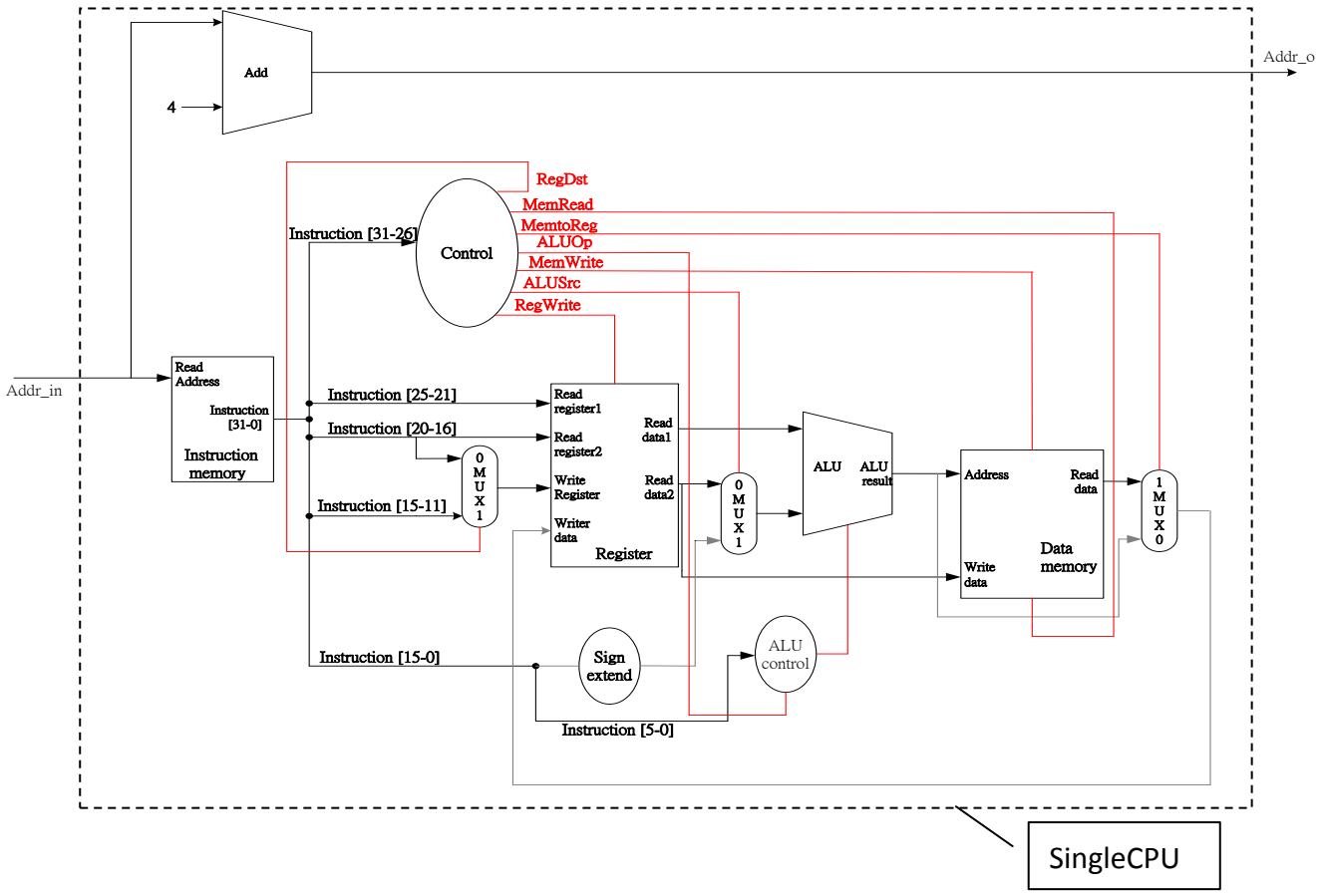
需完成 ADD、SUB、AND 以及 OR 四種功能

Data path: (參照第三版課本 p.309 FIGURE 5.19 或者 第四版課本 p.310 FIGURE 4.19)



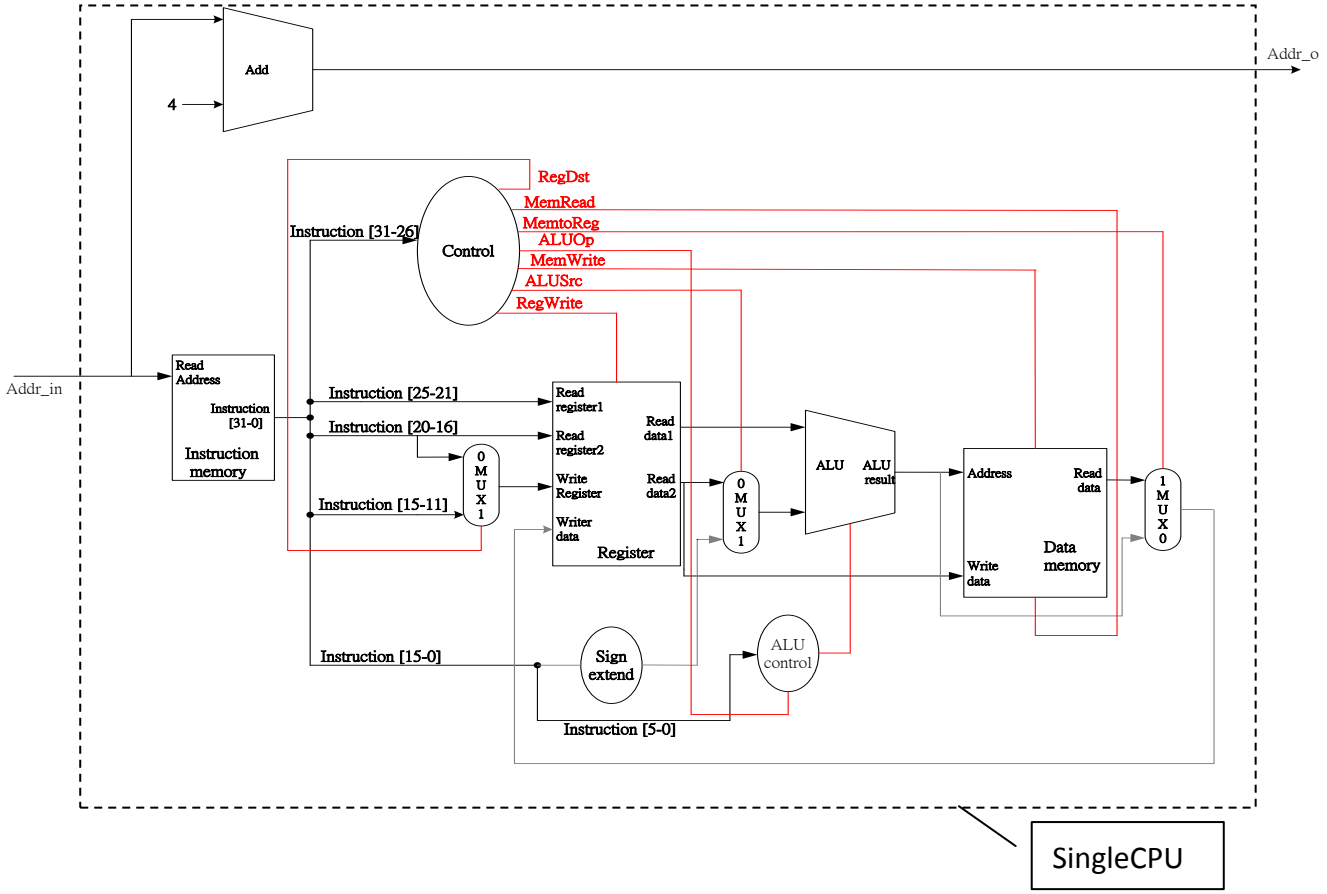
◎階段 2：結合階段 1，實做 lw、sw 指令(30%)

Data path: (參照第三版課本 p.310 FIGURE 5.20　或者　第四版課本 p.311 FIGURE 4.20)



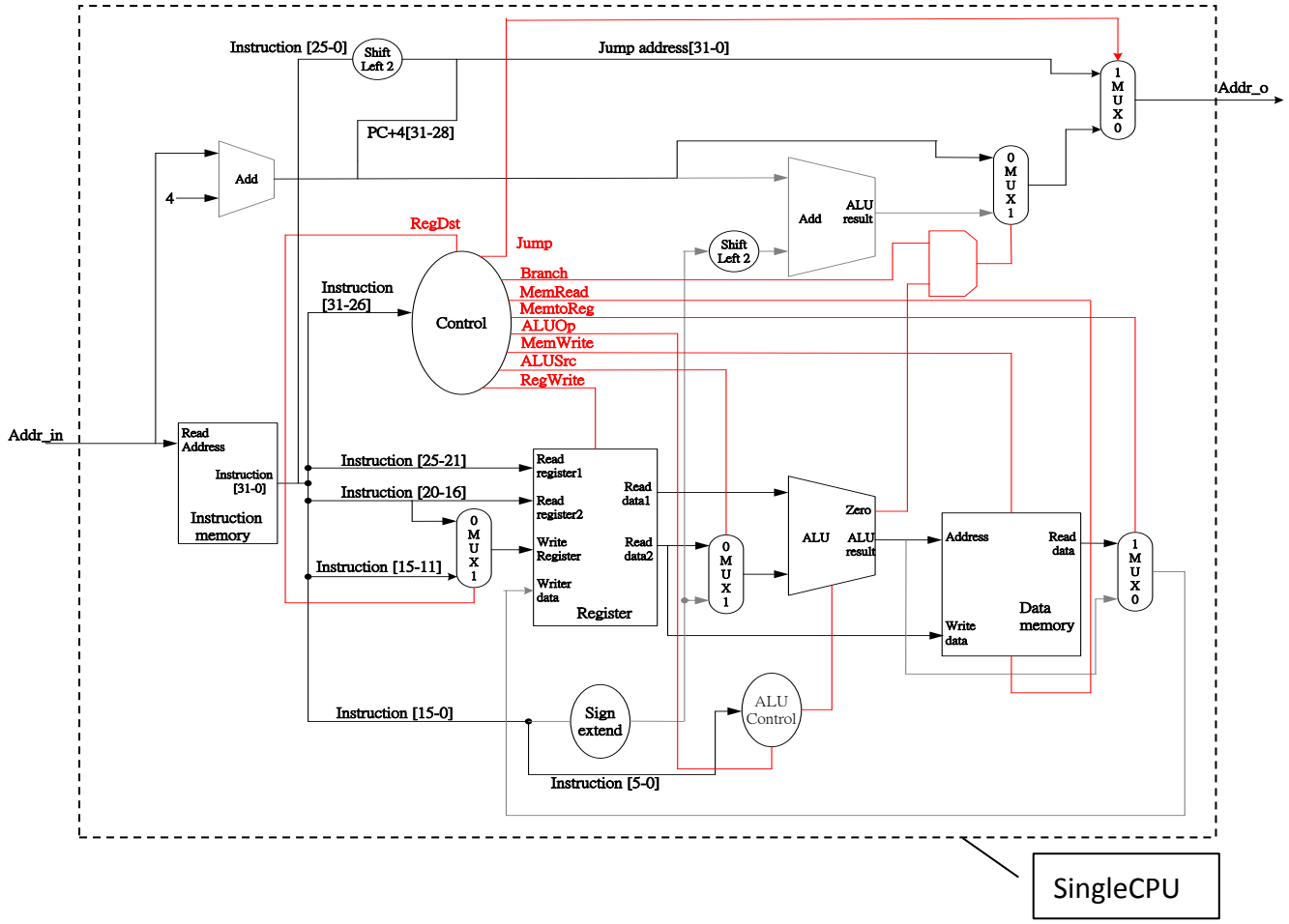
◎階段 3：結合階段 1、2，實做 addi、subi 指令(20%)

Data path: (參照第三版課本 p.310 FIGURE 5.20　或者　第四版課本 p.311 FIGURE 4.20)



◎階段 4：結合階段 1、2、3，實做 beq、bne、j 等指令(加分題 10%)

Data path:　(參照第三版課本 p.314 FIGURE 5.24　或者　第四版課本 p.315 FIGURE 4.24)



暫存器的編號請參照下表:

Name	Register number	Usage
$zero	0	the constant value 0
$v0-$v1	2-3	values for results and expression evaluation
$a0-$a3	4-7	arguments
$t0-$t7	8-15	temporaries
$s0-$s7	16-23	saved
$t8-$t9	24-25	more temporaries
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

指令編號：(Control)

指令	Op[31:26]
Add	010100(20)
Sub	010100(20)
And	010100(20)
Or	010100(20)
Sw	101001(41)
Lw	101010(42)
Addi	100111(39)
Subi	101000(40)
Beq	011100(25)
J	011011(26)

Function 編號：(ALU Control input)

指令	Funct[5:0]
Add	001001(9)
Sub	010010(18)
And	011011(27)
Or	100100(36)

ALU operation 功能:(Project1 ALU opcode)

指令	Opcode[5:0]
Add	000111(7)
Sub	001110(14)
And	010101(21)
Or	011100(28)

ALUOp (Control output)

功能	Branch	ALUOp1	ALUOp0
	ALUOp		
R-format	0	1	0
lw / sw / addi	0	0	0
subi	0	0	1
beq / bne	1	0	1

◎module interface:

```
 module SingleCPU(
      Addr_in,
      clk,
      Addr_o);
module Instruction_Memory(
      Addr_in,
      instr);
module Registers(
      clk,
      RSaddr,        //Read register1
      RTaddr,        //Read register2
      RDaddr,        //Write register
      RDdata,        // Read data1
      RTdata,        // Read data2
      RegWrite,      //Control signal
      src1);         //Write data
module ALU(
      src1,
      src2,
      operation,
      result,
      zero);
module Control(
      Op,
      RegDst,
      Jump,          //針對有做J-type指令的同學
      Branch,        //針對有做J-type指令的同學
      MemRead,
      MemtoReg,
```

```
     ALUOp,          //output [2:0]
     MemWrite,
     ALUSrc,
     RegWrite);
module Data_Memory(
     clk,
     addr,
     data,
     MemRead,
     MemWrite,
     DM_data);
module ALU_Control(
     funct,
     ALUOp,
     operation);
module Sign_Extend(
     data_i,
     data_o);
module ShiftLeftTwo32(
     data_i,
     data_o);
module ShiftLeftTwo26(       //針對有做J-type指令的同學
     data_i,
     data_o);
module Adder(
     data1,
     data2,
     data_o);
module MUX32_2to1(
     data1,
     data2,
     select,
     data_o);
module MUX5(
     data1,
     data2,
     select,
     data_o);
```

◎Instruction Memory須存放的指令順序為:

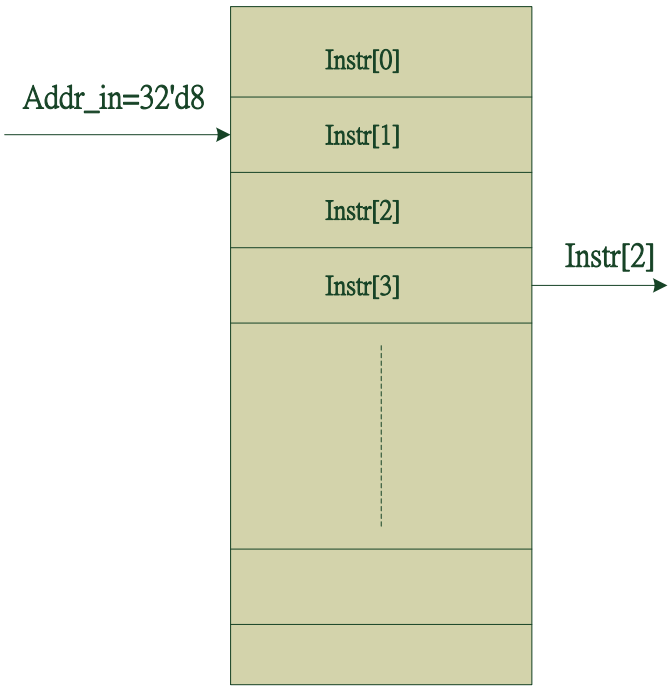
Addr_in	指令
32’ d0	add $t0, $t1, $t2 //{6’d20, 5’d9, 5’d10, 5’d8, 5’d0, 6’d7}
32’ d4	sub $t1, $t1, $t2
32’ d8	and $t3, $t4, $t5
32’ d12	or $t5, $t6, $t7
32’ d16	sw $t0, 2($t3)
32’ d20	lw $s3, 2($t3)
32’ d24	lw $s4, 3($t3)
30’ d28	sw $t0, 2($t2)
32’ d32	sw $s4, 4($t2)
32’ d36	addi $s5, $s4, 40
32’ d40	addi $s6, $s5, 22
32’ d44	subi $s3, $s6, 8
32’ d48	subi $s7, $s0, 2
32’ d52	beq $t4, $t6, 4 //以下針對有做J-type指令的同學
32’ d56	beq $t4, $t8, 4
32’ d60	beq $s3, $s5, 4
32’ d64	beq $t8, $t9, 4
32’ d68	j 110

Instruction_Memory實作與動作說明

1.有一指令:add $t0, $t1, $t2

=> 32'b010100_01001_01010_01000_00000_000111

2.動作說:(Hint:將Addr_in除以4)



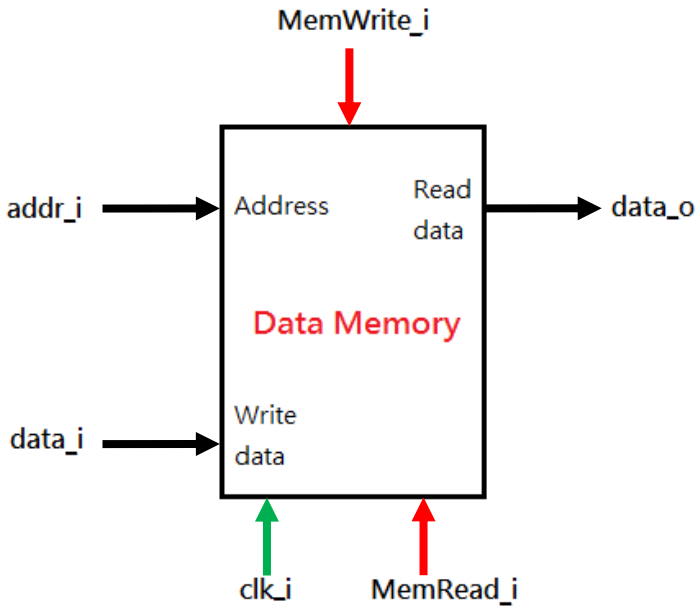
注意:這個範例是以word為單位

Registers預設值(Hint:用initial)

REGISTER [0]= 0
REGISTER [1]= 75
REGISTER [2]= 543
REGISTER [3]= 74
REGISTER [4]= 57
REGISTER [5]= 29
REGISTER [6]= 77
REGISTER [7]= 83
REGISTER [8]= 117
REGISTER [9]= 82
REGISTER [10]= 37
REGISTER [11]= 53
REGISTER [12]= 43
REGISTER [13]= 107
REGISTER [14]= 122
REGISTER [15]= 305
REGISTER [16]= 123
REGISTER [17]= 171
REGISTER [18]= 307
REGISTER [19]= 24
REGISTER [20]= 32
REGISTER [21]= 75
REGISTER [22]= 369
REGISTER [23]= 374
REGISTER [24]= 75
REGISTER [25]= 75
REGISTER [26]= 74
REGISTER [27]= 369
REGISTER [28]= 22
REGISTER [29]= 500
REGISTER [30]= 1000
REGISTER [31]= 21

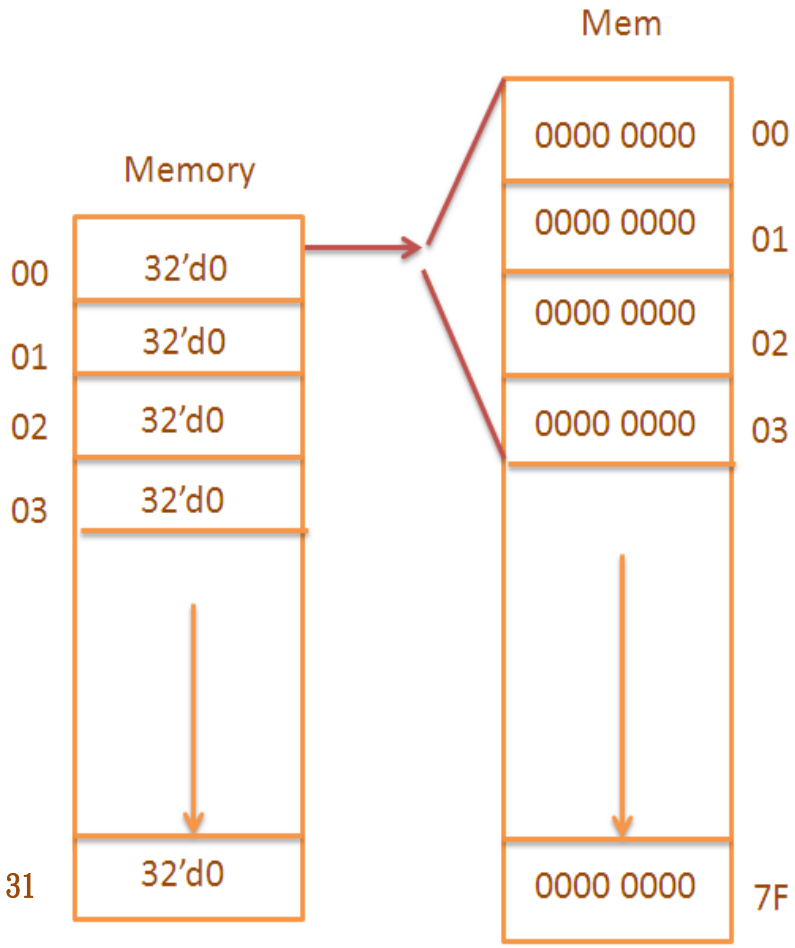
Data Memory設計格式

```
module Data_Memory(
                //input
                clk,
                addr,//Address line 32bits
                data,//Data line 32bits
                MemRead,//Control line
                MemWrite,// //Control line
                //output
               DM_data //32bits
    );
```



Address:0x00~0x80 bytes

以Word為設計，每8bits做儲存與寫入



初始化

```
Mem[i] = 8'b0;
```

寫入

Clk觸發& MemWrite觸發

```
        Mem[addr]   <= data[31:24];
        Mem[addr+1] <= data[23:16];
        Mem[addr+2] <= data[15:8];
        Mem[addr+3] <= data[7:0];
```

讀取

MemRead觸發

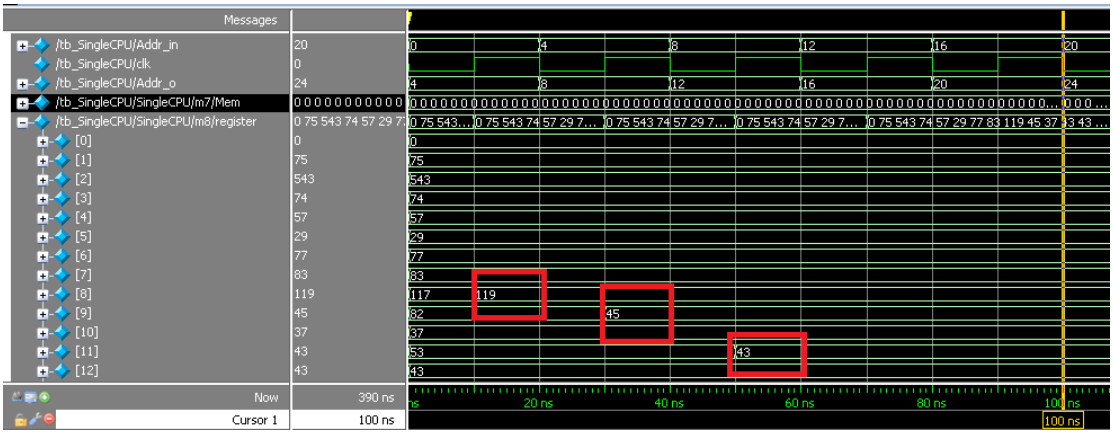
```
data = {Mem[addr], Mem[addr+1], Mem[addr+2], Mem[addr+3]};
```

Example

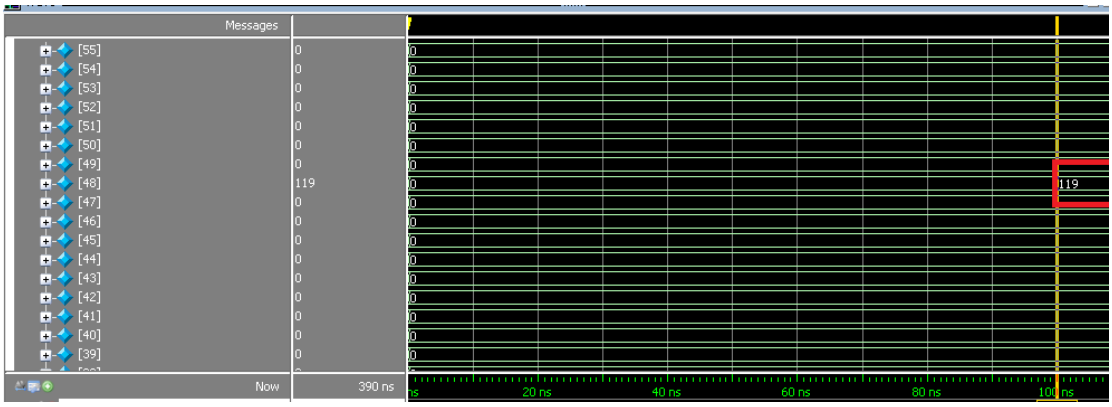
```
#20 MemRead=1'd0;MemWrite=1'd1;data=32'd10;addr=32'd10;
Memory[02] = {00000000, 00001010, 00000000, 00000000,};
00000000101000000000000000000=655360
```

◎模擬結果:(此為示意，故部分省略，請繳交完整結果)

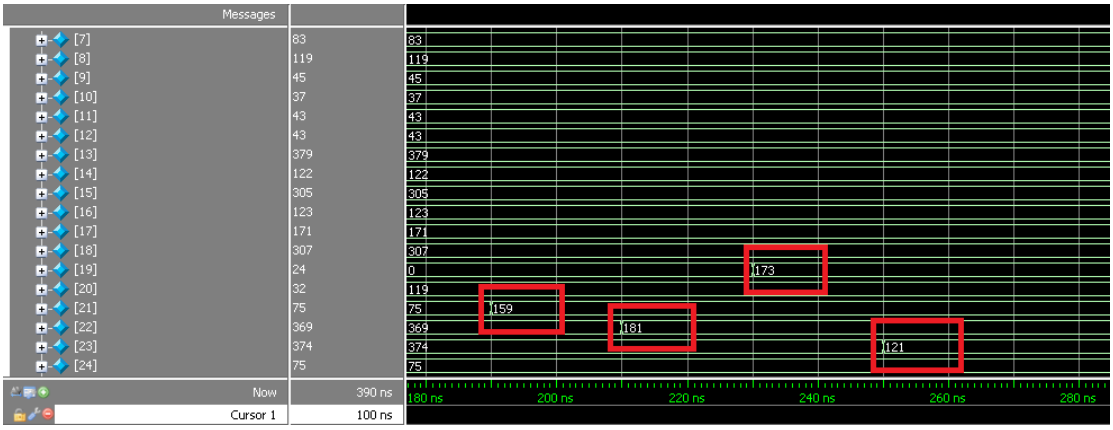
- Register



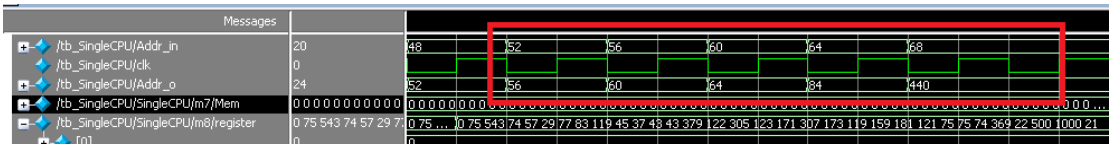
- Data Memory



- Addi & Subi



- Branch/Jump



◎需完成的.v檔:

ALU_Control.v、ALU.v(project1)、Instruction_memory.v、Control.v、SingleCPU.v、Registers.v

◎評分標準：

1. 依完成階段給分，**依提供的test bench為準**。
2. 第四階段為加分題(10分)
3. 其餘15分，依據報告與datapath、control path(其包含 ALU module、register module、data memory module、control unit module、ALUcontrol module、sign extend module、MUX)完整程度斟酌給分。

◎繳交檔案：

1. report.doc 報告word 檔：
   - i 整個cpu 的設計想法，各module 的設計方式，每個元件作用。
   - ii. Verilog module 架構圖 & 流程圖 & 模擬結果圖
   - iv. **遇到的困難 & 解決辦法** & 個人心得
2. Verilog code(.v 檔)，其中**包含提供的 module、test bench** 和同學自己的 module(包括外層的 module)
3. 抄襲者零分。

- 繳交時間: 即日起至 2016.05.9 的早上 10:00，請將檔案上傳至 **Moodle**，遲交請寄至 M10407414@mail.ntust.edu.tw。
- 檔案命名：將程式放在資料夾後壓縮成檔案 b1xxxxxxx_project2_v1.zip (v1 為繳交版本，會以最晚的版本評分)。

請勿抄襲。