

# Computer Organization Lab3

Name: 陳子祈

ID: 0819823

## Architecture diagrams:

R type						
Instruction set	Op code	rs	rt	rd	shamt	funct
Instr location	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]
add    \$rd, \$rs, \$rt	000000 (0)				00000	100000 (32)
sub    \$rd, \$rs, \$rt	000000 (0)				00000	100010 (34)
and    \$rd, \$rs, \$rt	000000 (0)				00000	100100 (36)
or     \$rd, \$rs, \$rt	000000 (0)				00000	100101 (37)
slt    \$rd, \$rs, \$rt	000000 (0)				00000	101010 (42)
jr    \$rs	000000 (0)		00000	00000	00000	001000 (8)
I type						
Instruction set	Op code	rs	rt	immediate		
Instr location	[31:26]	[25:21]	[20:16]	[15:0]		
addi   \$rt, \$rs, imm	001000 (8)					
slti   \$rt, \$rs, imm	001010 (10)					
beq    \$rt, \$rs, imm	000100 (4)					
lw     \$rt, \$rs, imm	100011 (35)					
sw     \$rt, \$rs, imm	101011 (43)					
Jump type						
Instruction set	Op code	Address				
Instr location	[31:26]	[25:0]				
j addr	000010 (2)					
jal addr	000011 (3)					

## Hardware module analysis:

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



ALU\_op\_o : 2bit output for ALU\_Ctrl to determine operation type

ALUSrc\_o : 1bit output determine ALU source

RegDst\_o : 1 bit output determine Read reg2 is rt or rd

Branch\_o : 1bit output the instruction is branch type or not

Jump\_o : 1bit output to determine jump or not

MemRead\_o : 1bit output for Data memory to determine read memory data or not

MemWrite\_o : 1bit output for Data memory to determine write memory data or not

MemtoReg\_o : 2bit output to determine where Register write data is from

Instr_op [31:26]		Instruction	RegDst [1:0]	ALUSrc	Mem toReg [1:0]	Reg Write	Mem Read	Mem Write	Branch	ALU Op [1:0]	Jump
[31:29]	[28:26]										
000	000	R-type	01	0	00	1	0	0	0	00	0
	010	j		0		0	0	0	1	10	1
	011	jal	10		11	1					1
	100	beq				0					0
I-type											
001	000	addi	00	1	00	1	0	0	0	01	0
	010	slti								11	
100		lw	00	1	01	1	1	0	0	01	0
101		sw	00	1		0	0	1	0	01	0

## 2) ALU\_Ctrl

功能：將ALU\_op及function code轉成ALU所需的ALUCtrl，決定ALU的動作及控制其他MUX、Shifter。

Port description：

funct\_i : 6bit input function code

ALUOp\_i : 2bit input for ALU\_Ctrl to determine operation type

ALUCtrl\_o : 4bit output to ALU control

jr\_o : 1bit output to determine whether use jr or not

ALUOp_i	funct_i	operation	ALUCtrl	jr_o
R type				
00	001000	jr	0000	1
	100000	add	0010	0
	100010	sub	0110	
	100100	and	0000	
	100101	or	0001	

	101010	slt	0111	
I type				
01		addi lw sw	0010	0
10		beq	0110	
11		slti	0111	
其他				
10		j	XXXX	0
10		jal	XXXX	

### 3) ALU

功能：32bit運算邏輯單位，參考課本附錄程式，可做add、sub、or、and、slt。

Port description：

src1\_i：32bit input data

src2\_i：32bit input data

ctrl\_i：4bit ALU\_Control

result\_o：32bit result for ALU

zero\_o：1 bit when the output is 0, zero must be set

### 4) Adder

功能：輸入兩個data輸出其相加結果。

Port description：

src1\_i：32bit input data

src2\_i：32bit input data

sum\_o：32bit output sum

### 5) Sign\_Extend

功能：將輸入data做Sign Extend，data\_i複製到data\_o低16位，data\_i最高位bit複製到data\_o高16位。

Port description：

data\_i：16bit input data

data\_o：32bit output data

### 6) Shift\_Left\_Two\_32

功能：將input data左移兩個bit。

Port description :

data\_i : 32bit input data

data\_o : 32bit output data

#### 7) MUX\_2to1

功能：如果 select\_i = 0 則輸出 data0\_i；select\_i = 1 則輸出data1\_i。

Port description :

data0\_i : 32bit input data

data1\_i : 32bit input data

select\_i : 1bit select for MUX

data\_o : 32bit output data

#### 8) MUX\_4to1

功能：如果 select\_i = 00 則輸出 data0\_i；select\_i = 01 則輸出data1\_i；select\_i = 10 則輸出data2\_i；select\_i = 11 則輸出data3\_i

Port description :

data0\_i : 32bit input data

data1\_i : 32bit input data

data2\_i : 32bit input data

data3\_i : 32bit input data

select\_i : 2bit select for MUX

data\_o : 32bit output data

#### 9) Simple\_Single\_CPU

功能：將上述所提到之 Module 依照 Architecture diagram 的附圖做連接，完成 Simplified Single-cycle CPU。

### Finished part:

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

Instruction set	Op code	rs	rt	rd	shamt	funct
Instr location	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]
add \$r0,\$r0,\$r0	000000 (0)	00000	00000	00000	00000	100000 (32)

**Description:****r0 = Reg[0] = r0+r0 = 0+0 = 0****PC = 0**

```

PC =          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
Data Memory =    0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =      0, R1 =      0, R2 =      0, R3 =      0, R4 =      0, R5 =      0, R6 =      0, R7 =      0
R8 =      0, R9 =      0, 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 =      128, R30 =      0, R31 =      0

```

Instruction set	Op code	rs	rt	immediate
<b>Instr location</b>	[31:26]	[25:21]	[20:16]	[15:0]
addi \$a0, zero, 4	001000 (8)	00000	00100	00000000000000100

**Description:****a0 = Reg[4] = zero+4 = 4****PC = 4**

```

PC =          4
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
Data Memory =    0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =      0, R1 =      0, R2 =      0, R3 =      0, R4 =      0, R5 =      0, R6 =      0, R7 =      0
R8 =      0, R9 =      0, 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 =      128, R30 =      0, R31 =      0

```

Instruction set	Op code	rs	rt	immediate
<b>Instr location</b>	[31:26]	[25:21]	[20:16]	[15:0]
addi \$t1, zero, 1	001000 (8)	00000	01001	00000000000000001

**Description:****t1 = zero+1 = 1****PC = 8**

```

PC =          8
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
Data Memory =    0,      0,      0,      0,      0,      0,      0,      0
Registers
R0 =      0, R1 =      0, R2 =      0, R3 =      0, R4 =      4, R5 =      0, R6 =      0, R7 =      0
R8 =      0, R9 =      0, 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 =      128, R30 =      0, R31 =      0

```

Instruction set	Op code	Address
<b>Instr location</b>	[31:26]	[25:0]
jal fib	000011 (3)	000000000000000000000000101 (5)

**Description:****Reg[31] = PC+4 = 16**

**PC = 12 (pcnext = 5\*4 = 20)**

PC = 12  
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  
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0  
Registers  
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, 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 = 128, R30 = 0, R31 = 0

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
addi \$sp, \$sp, -12	001000 (8)	11101	11101	1111111111110100

**Description:**

**sp = Reg[29] = sp-12 = 128-12 = 116**

**PC = 20**

PC = 20  
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  
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0  
Registers  
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, 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 = 128, R30 = 0, R31 = 16

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
sw \$ra, \$sp, 0	101011 (43)	11101	11111	0000000000000000

**Description:**

**Mem[Rs+imm] = Mem[116] = Reg[rt] = Reg[31] = 16 (ra = Reg[31])**

**PC = 24**

PC = 24  
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  
Data Memory = 0, 0, 0, 0, 0, 0, 0, 0  
Registers  
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, 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 = 116, R30 = 0, R31 = 16

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
sw \$s0, \$sp, 4	101011 (43)	11101	10000	0000000000000100

**Description:**

**Mem[Rs+imm] = Mem[120] = Reg[rt] = Reg[16] = 0 (s0 = Reg[16])**

**PC = 28**

PC = 28

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
Data Memory =	0,	0,	0,	0,	0,	16,	0,	0

Registers

R0 =	0,	R1 =	0,	R2 =	0,	R3 =	0,	R4 =	4,	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 =	116,	R30 =	0,	R31 =	16

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
sw \$s1, \$sp, 8	101011 (43)	11101	10001	00000000000001000

**Description:**

**Mem[Rs+imm] = Mem[124] = Reg[rt] = Reg[17] = 0 (s1 = Reg[17])**

**PC = 32**

PC = 32

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
Data Memory =	0,	0,	0,	0,	0,	16,	0,	0

Registers

R0 =	0,	R1 =	0,	R2 =	0,	R3 =	0,	R4 =	4,	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 =	116,	R30 =	0,	R31 =	16

Instruction set	Op code	rs	rt	rd	shamt	funct
Instr location	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]
add \$s0, \$a0, zero	000000 (0)	00100	00000	10000	00000	100000 (32)

**Description:**

**s0 = Reg[16] = a0+zero = Reg[4]+Reg[0] = 4**

**PC = 36**

PC = 36

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
Data Memory =	0,	0,	0,	0,	0,	16,	0,	0

Registers

R0 =	0,	R1 =	0,	R2 =	0,	R3 =	0,	R4 =	4,	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 =	116,	R30 =	0,	R31 =	16

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
beq \$s0, zero, re1	000100 (4)	10000	00000	00000000000001100 (12)

**Description:**

**If s0(Reg[16]) == 0, branch to re1. Now s0 == 4 => sequential**

**PC = 40**



```

PC = 40
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
Data Memory = 0, 0, 0, 0, 0, 16, 0, 0
Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, R5 = 0, R6 = 0, R7 = 0
R8 = 0, R9 = 1, R10 = 0, R11 = 0, R12 = 0, R13 = 0, R14 = 0, R15 = 0
R16 = 4, 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 = 116, R30 = 0, R31 = 16

```

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
beq \$s0, \$t1, re1	000100 (4)	10000	01001	00000000000001100 (12)

If s0(Reg[16]) == t1(Reg[9]), branch to re1. Now s0 != t1 => sequential

PC = 44

```

PC = 44
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
Data Memory = 0, 0, 0, 0, 0, 16, 0, 0
Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, R5 = 0, R6 = 0, R7 = 0
R8 = 0, R9 = 1, R10 = 0, R11 = 0, R12 = 0, R13 = 0, R14 = 0, R15 = 0
R16 = 4, 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 = 116, R30 = 0, R31 = 16

```

Instruction set	Op code	rs	rt	immediate
Instr location	[31:26]	[25:21]	[20:16]	[15:0]
addi \$a0, \$s0, -1	001000 (8)	10000	00100	1111111111111111

Description:

a0 = Reg[4] = s0-1 = Reg[16]-1 = 4-1 = 3

PC = 48

```

PC = 48
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
Data Memory = 0, 0, 0, 0, 0, 16, 0, 0
Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 4, R5 = 0, R6 = 0, R7 = 0
R8 = 0, R9 = 1, R10 = 0, R11 = 0, R12 = 0, R13 = 0, R14 = 0, R15 = 0
R16 = 4, 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 = 116, R30 = 0, R31 = 16

```

Instruction set	Op code	Address
Instr location	[31:26]	[25:0]
jal fib	000011 (3)	000000000000000000000000101 (5)

Description:

Reg[31] = PC+4 = 56

PC = 52 (pcnext = 5\*4 = 20)

```

PC = 52
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
Data Memory = 0, 0, 0, 0, 0, 16, 0, 0
Registers
R0 = 0, R1 = 0, R2 = 0, R3 = 0, R4 = 3, R5 = 0, R6 = 0, R7 = 0
R8 = 0, R9 = 1, R10 = 0, R11 = 0, R12 = 0, R13 = 0, R14 = 0, R15 = 0
R16 = 4, 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 = 116, R30 = 0, R31 = 16

```

礙於篇幅，中間過程就不詳細追蹤，直接看 final 結果:

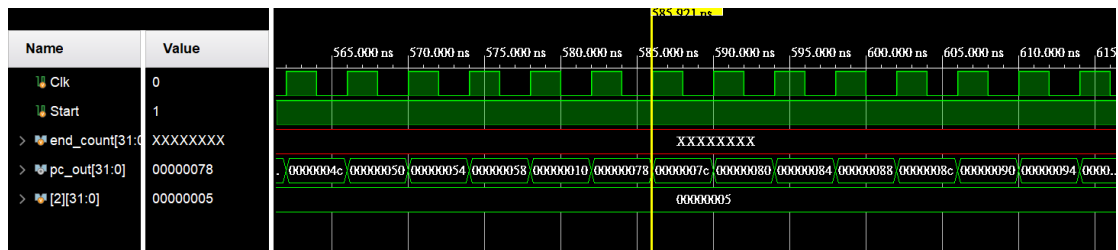
```

PC = 120
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 = 128, R30 = 0, R31 = 16

```

r2=2

波形圖也顯示當 PC 到 120 之後 r2 都是 2:



## Problems you met and solutions:

### 1. j、jal、jr 不知道要如何設計

雖然助教有提供參考設計圖，不過上面沒有列出 jal 改變 Reg[31]及 jr 改變 PC 的部分，所以我另外加了多工器或輸入進去，整個 module 就比較完整了。

### 2. 控制訊號寫錯

這次又與上次犯同樣的錯，但有鑑於上次的經驗，檢查接線都沒問題之後，就確定應該是控制訊號寫錯，改回來就好了。

## Summary:

雖然這次只是修改上一次 Lab 的內容，但是因為控制訊號變多，模組設計變複雜，建模組的過程中仍然遇到不少問題，當我把所有的控制訊號整理成表格之後，就很好設計所有控制訊號。感覺這次 Lab 讓我更了解 single cycle MIPS CPU 的 jump、load、store 等指令的運作流程。希望以後可以更快發現問題並解決問題，我也希望可以出多一點作業複習上課內容並增進寫 verilog 程式的能力。