

Lab4

Single cycle CPU

TA-黃炫峰

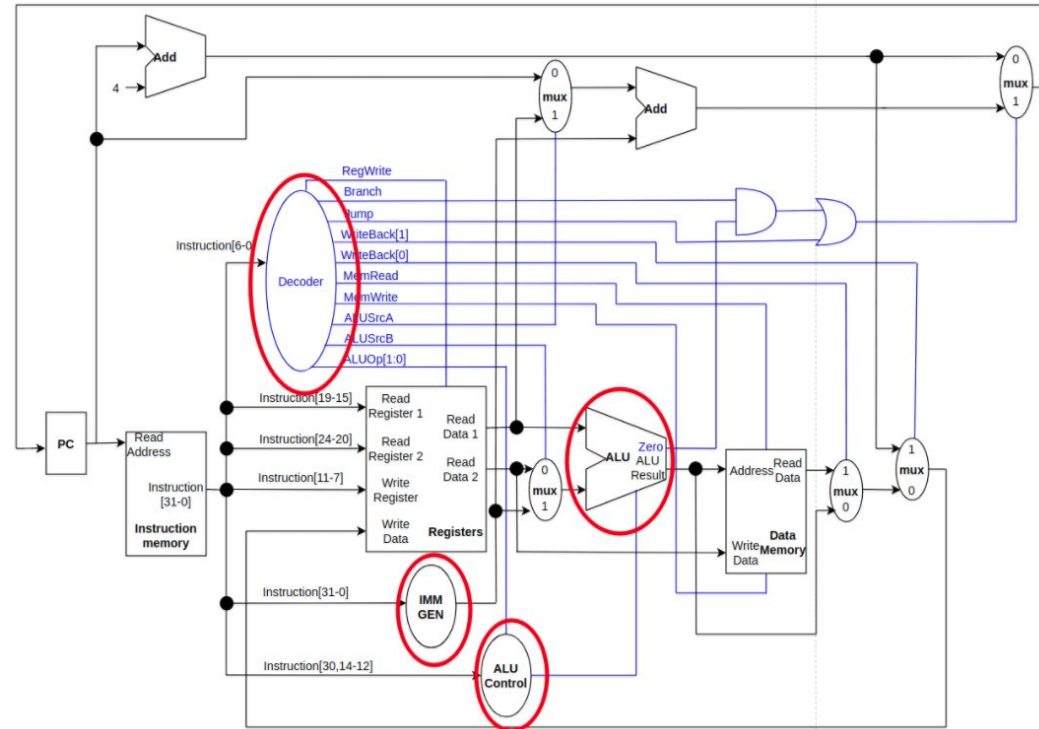
ma880521@gmail.com

Object

- To realize how to set the control signal in different instruction type.(Decoder & ALU Controller)
- To clarify how sign-extend work.
- Connect all datapath to form a single cycle CPU.

Implement

- You have to implement following unit:
 - Decoder
 - Sign-extend Unit
 - ALU Controller
 - ALU
 - Single Cycle CPU(Connect all unit)



Attached file

- **TODO**
 - **Lab4_release**
 - Decoder.v
 - Imm_Gen.v
 - ALU_Ctrl.v
 - alu.v
 - Simple_Single_CPU.v
- **Validate the correction of your implementation**
 - \$./demo.sh
- **Debug**
 - result.txt
 - answer.txt

answer.txt

[illegible]

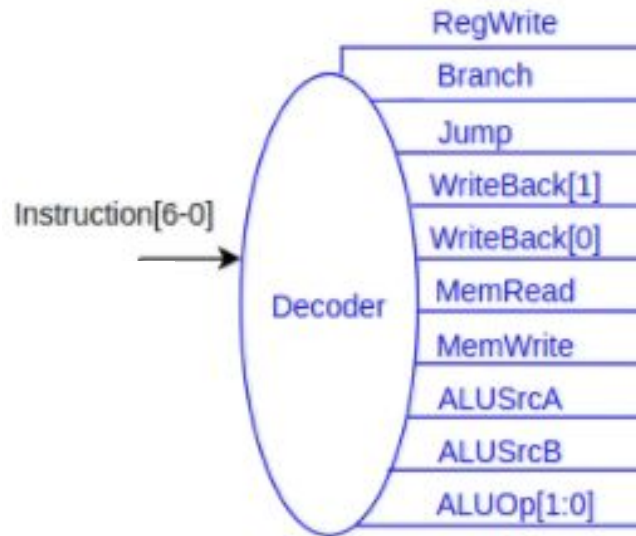
Implement instructions

- R-type
 - add
 - slt
- I-type
 - addi
- lw
- sw
- beq
- jal & jalr

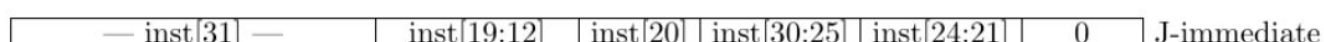
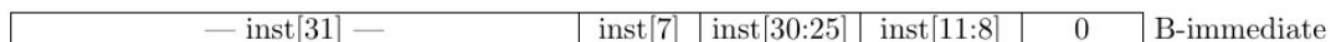
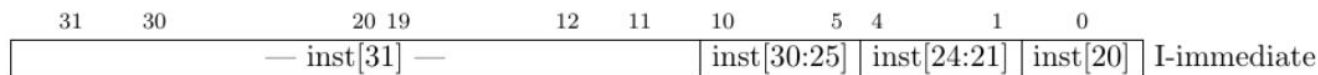
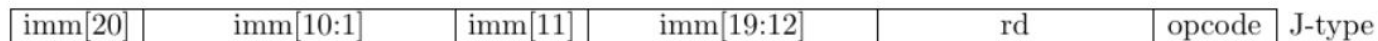
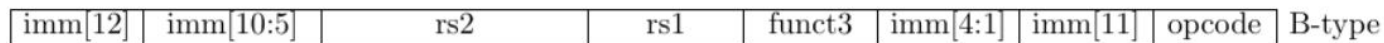
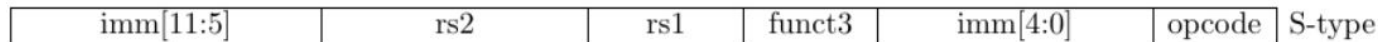
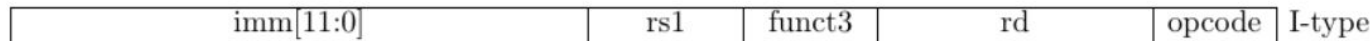
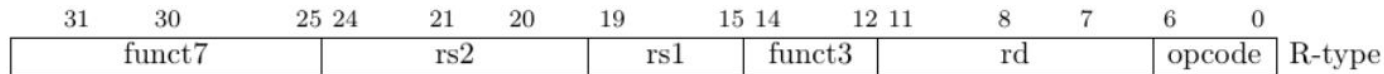
[illegible]

Decoder

- Generate the corresponding control signal according to the instruction
 - RegWrite = ?
 - Branch = ?
 - Jump = ?
 - WriteBack[1] = ?
 - WriteBack[0] = ?
 - MemRead = ?
 - MemWrite = ?
 - ALUSrcA = ?
 - ALUSrcB = ?
 - ALUOp[1:0] = ?



Sign-extend



Imm_Gen.v

```
`timescale 1ns/1ps

module Imm_Gen(
    | input [31:0] instr_i,
    | output reg[31:0] Imm_Gen_o
);

//Internal Signals
wire [7:1:0] opcode;
wire [2:0] func3;
wire [3:1:0] Instr_field;

assign opcode = instr_i[6:0];
assign func3 = instr_i[14:12];

/* Write your code HERE */

endmodule
```

Simple Single CPU

- Combine all the above parts to build a CPU.
- You need to complete Simple_Single_CPU.v all the parts inputs and output.

```
Imm_Gen ImmGen(  
    .instr_i(instr),  
    .Imm_Gen_o(Imm_Gen_o)  
);  
  
ALU_Ctrl ALU_Ctrl(  
    .instr(),  
    .ALUOp(),  
    .ALU_Ctrl_o()  
);  
  
MUX_2to1 MUX_ALUSrcA(  
    .data0_i(),  
    .data1_i(),  
    .select_i(),  
    .data_o()  
);  
  
Adder Adder_PCReg(  
    .src1_i(),  
    .src2_i(),  
    .sum_o()  
);
```

```
MUX_2to1 MUX_ALUSrcB(  
    .data0_i(),  
    .data1_i(),  
    .select_i(),  
    .data_o()  
);  
  
alu alu(  
    .rst_n(rst_i),  
    .src1(),  
    .src2(),  
    .ALU_control(),  
    .Zero(),  
    .result()  
);  
  
Data_Memory Data_Memory(  
    .clk_i(clk_i),  
    .addr_i(),  
    .data_i(),  
    .MemRead_i(),  
    .MemWrite_i(),  
    .data_o()  
);
```

.....

Test data

- A simple machine code which recursively add numbers from 1 to 10.

1				//00000000000000000000000000000000
2	addi	x10	x0 10	//00000000101000000000010100010011
3	jal	x1	8	//00000000100000000000000011101111
4	jal	x0	72	//00000100100000000000000001101111
5	addi	x2	x2 -16	//11111111000000010000000100010011
6	sw	x1	8(x2)	//00000000000100010010010000100011
7	sw	x10	0(x2)	//00000000101000010010000000100011
8	addi	x5	x10 -1	//11111111111101010000001010010011
9	slt	x7	x5 x0	//00000000000000101010001110110011
10	beq	x7	x0 16	//00000000000000111000100001100011
11	addi	x10	x0 0	//000000000000000000000010100010011
12	addi	x2	x2 16	//00000001000000010000000100010011
13	jalr	x0	x1 0	//00000000000000001000000001100111
14	addi	x10	x10 -1	//11111111111101010000010100010011
15	jal	x1	-40	//11111011001111111110000111011111
16	addi	x6	x10 0	//00000000000001010000001100010011
17	lw	x10	0(x2)	//00000000000000010010010100000011
18	lw	x1	8(x2)	//00000000100000010010000010000011
19	addi	x2	x2 16	//00000001000000010000000100010011
20	add	x10	x10 x6	//00000000011001010000010100110011
21	jalr	x0	x1 0	//00000000000000001000000001100111
22				//00000000000000000000000000000000

Testbench

- **This script cannot run in Windows**
- Put your.v file in Lab4_release
- \$./demo

Correct

```

CONGRATULATION!!
MMMMMMMMMMMMMMMMMMMMMMMMWXXK00kk000XWMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMW0dc:::ccllllllcc:::co0NMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWkw::lk0NNNNNNNNNNNNNNNNNNKko::dNMMMMMMMMMMMM
MMMMMMMMMMMMMO'c0NNNNNNNNNNNNNNNNNNNNNNNNNNNNKl'xWMMMMMMMMMM
MMMMMMMMMMnc'0NNNNNNNNNNKXNNNNNNNNNNXXNNNNNNNN0';XMMMMMMMMMM
MMMMMMMMW;;,KNNNNNNNNN0.xNNNNNNNNNNl;NNNNNNNNNNXc'NMMMMMMM
MMMMMWkL';dXNN0LNNNd.KNNNNNNNNNx.KNNdkNNNxc:.KwMMMMMM
MMXl.cl...0NNd:::;:::;:::;:::;.:::cNNX'.cc;cKMM
Wo'oXNNk.;NNNd'wwwwwwwwwwwwwwwwwwc:NNNc.dNNNx'cN
0;;.ONX.;.NNNo,Wk:NwwwwwwwwwwwwWooWl;NNNo.'@NK.;;k
MM0.oX.c.cNNNo,Wl.XwwwwwwwwwwwwW;;,Wl,KNNx.;oK'xMM
MMX.;.;0.LNNNo,Wo'XwwwwwwwwwwwwW::Wl'ONN0.do.;.0MM
MMMMNN.oX.oXNN0,wwwwwwwwwwwwwwwwwWl,kKNX.lk.0NMMMM
MMMM0.kl.xKNN0'wwwwwwWNcoocKwwwwwwWWl,kONN;.K.dMMMM
MMMMd.K.;xoKNd'wwwwwwWXX0kKwwwwwwWWc,kxxKo'N;.MMMM
MMMM;;N'.'. '@d;.codxxxkk0000kkxdoc;;d'..Kd.WMMM
MMMMN.xNkOxk;'.k0kxd..looo'.oxk00'...;xN0xX0.0MMM
MMMX'XNNNNNO....;ox00'.dx';k0xo;...'.kNNNNNN.LMMM
MMW;lNXOK0x.....;;;'.....';;;;'.o0K0XNx.NMM
MM0.cc'.ox'.'.;;;.....';;;;'.dx...l.dMM
MMN@OKl'x,;;;.....';;;;'.d;.X0ONMM
MMMMMMW';;.0ko,.....';;;;'.ox0';.KMMMMMM
MMMMMMWxoc,;lx,.....';;;;'.ccc;.xo;.odXMMMMMM
MMMMMMMMMk.0k,.....';;;;'.cccc;.dK,LMMMMMMMMMM
MMMMMMMMMMX'.....';;;;'.cccccc....OMMMMMMMMM

```

Notice

- [Lab4討論區](#)
- [Lab4-6分組表](#)

Appendix

R-type slt

- **Assembly**

```
1 slt rd, rs1, rs2
```

- **Semantics**

```
1 if( GPR[rs1] < GPR[rs2] )  
2     GPR[rd] = 1  
3 else  
4     GPR[rd] = 0
```

R-type slt

- **Assembly**

```
1 slt rd, rs1, rs2
```

- **Semantics**

```
1 if( GPR[rs1] < GPR[rs2] )  
2     GPR[rd] = 1  
3 else  
4     GPR[rd] = 0
```

Load

- **Assembly**

```
1 lw rd, imm(rs1)
```

- **Semantics**

```
1 target = GPR[rs1] + sign-extend(imm_{12})  
2 GPR[rd] = DM[target]
```

Store

- **Assembly**

```
1 sw rs2, imm(rs1)
```

- **Semantics**

```
1 target = GPR[rs1] + sign-extend(imm_{12})  
2 DM[target] = GPR[rs2]
```


Branch

- **Assembly**

```
1 beq rs1, rs2, imm
```

- **Semantics**

```
1 target = PC + sign-extend(imm_{13})  
2 if (GPR[rs1] == GPR[rs2])  
3     PC = target  
4 else  
5     PC = PC + 4
```

Jump and link

- **Assembly**

```
1 jal rd imm
```

- **Semantics**

```
1 target = PC + sign_extend(imm_{21})  
2 GPR[rd] = PC + 4  
3 PC = target
```

Jump indirect

- **Assembly**

```
1 jalr rd, rs1, imm
```

- **Semantics**

```
1 target = GPR[rs1] + sign-extend(imm_{12})  
2 GPR[rd] = PC + 4  
3 PC = target
```