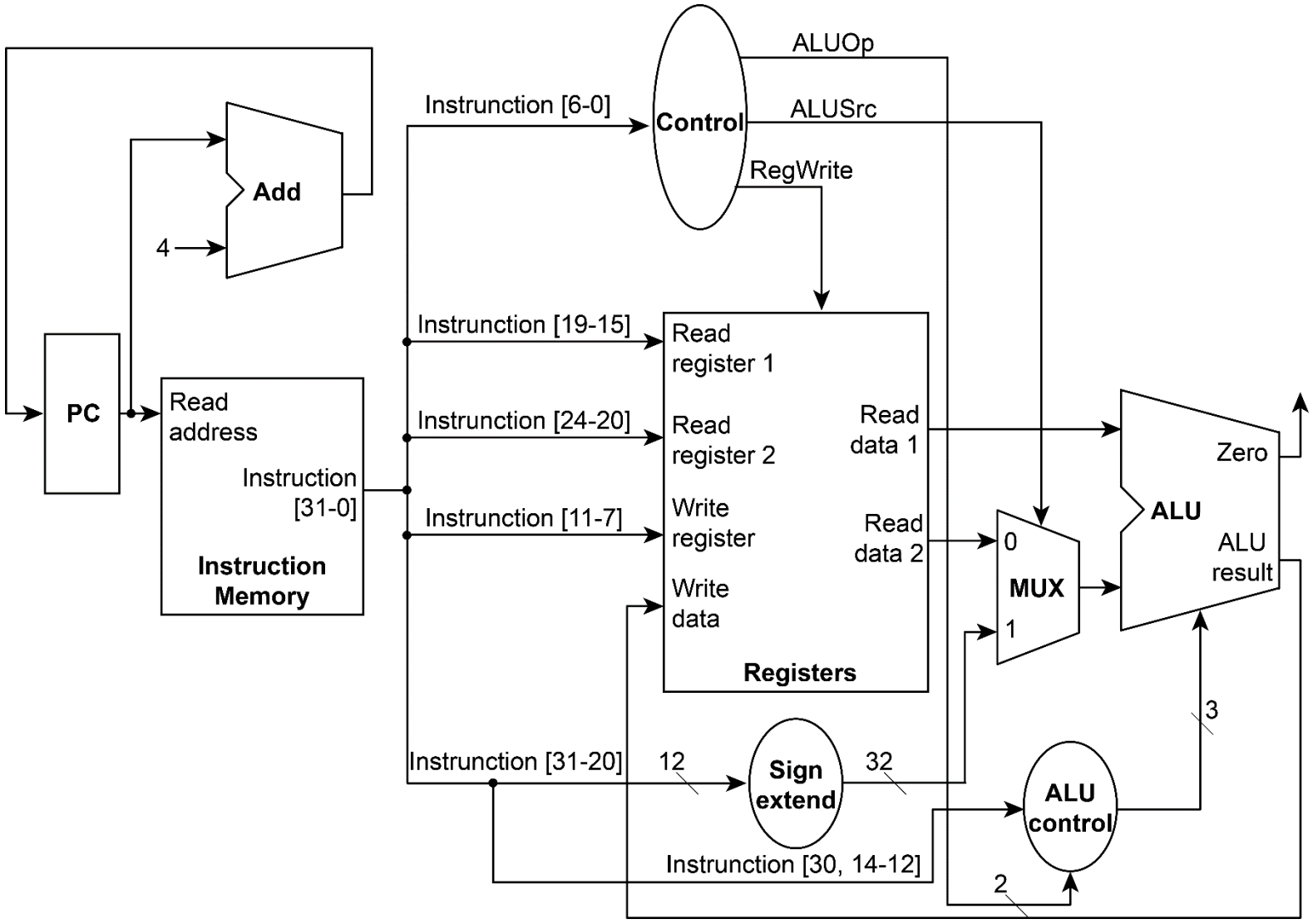
Homework 3

A Single Cycle CPU by Verilog

TA: 蔡承佑
Email: r07922182@ntu.edu.tw

Data Path

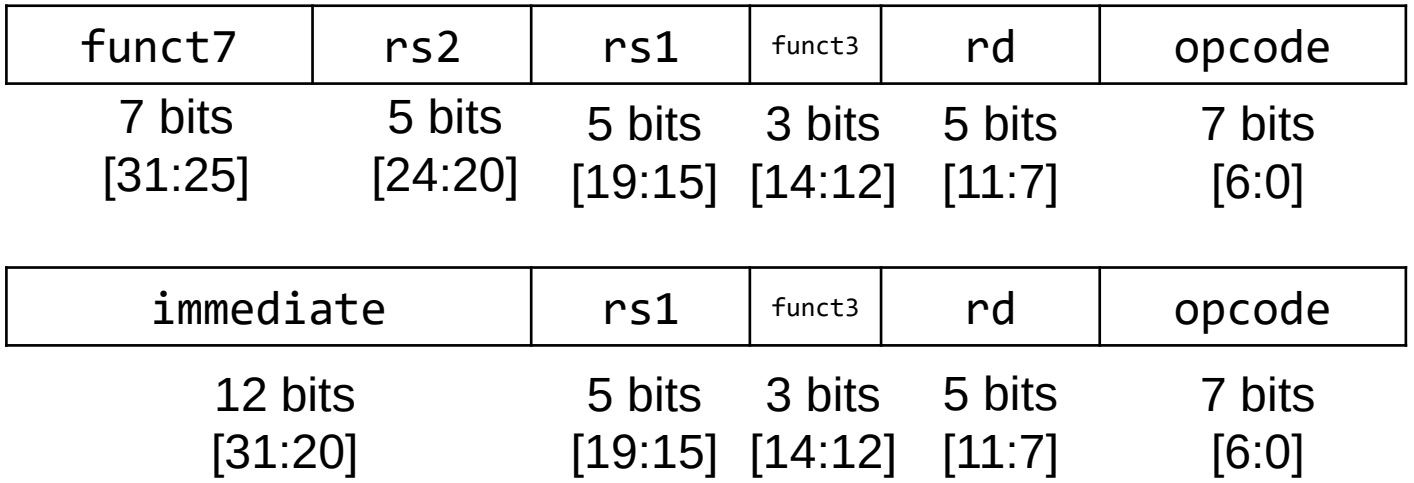

ALUOp
Instrunction [6-0]
Control
ALUSrc
RegWrite
Add
4
Instrunction [19-15]
Read register 1
PC
Read address
Instrunction [24-20]
Read register 2
Read data 1
Zero
ALU
Instruction [31-0]
Instrunction [11-7]
Write register
Read data 2
0
MUX
1
ALU result
Instruction Memory
Write data
Registers
3
Instrunction [31-20]
12
Sign extend
32
ALU control
Instrunction [30, 14-12]
2

Requirement #1

- Required Instruction Set
  - and
  - or
  - add
  - sub
  - mul
  - addi

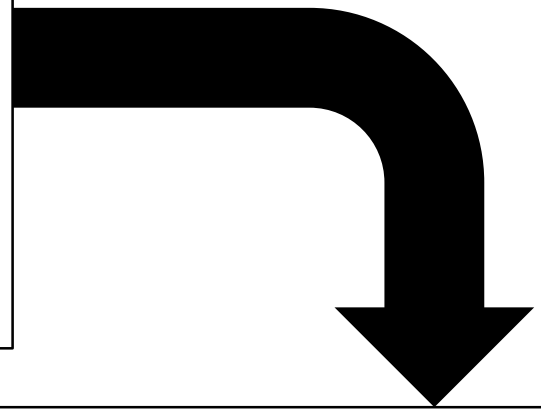
Requirement #2

- Translate the assembly code into machine code
- Register file: 32 registers
- Instruction Memory: 1KB
- Machine code:



Instruction Translation

```
add  $t0,$0,$0
addi $t1,$0,10
addi $t2,$0,13
mul  $t3,$t1,$t1
addi $t1,$t1,1
sub  $t2,$t2,$t1
and  $t3,$t1,$t2
or   $t4,$t2,$t3
```



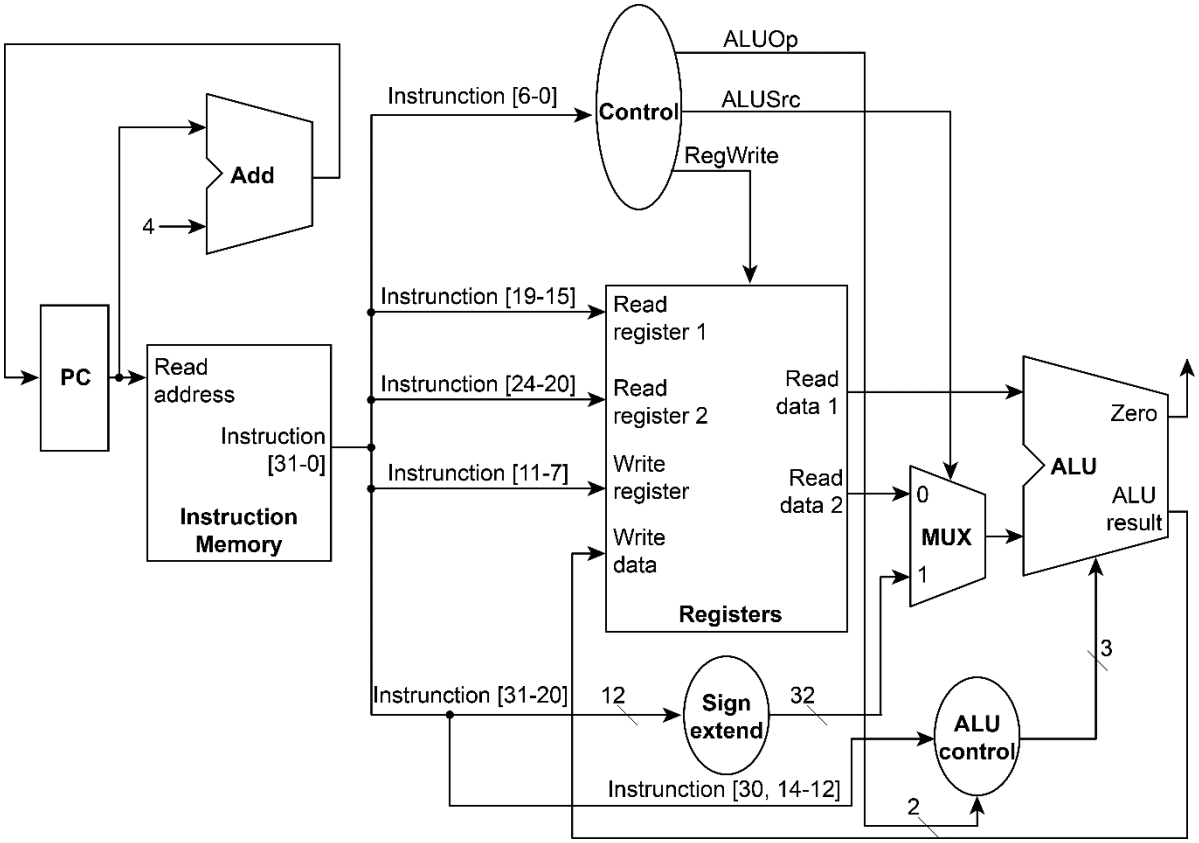
```
0000000_00000_00000_000_01000_0110011 //add  $t0,$0,$0
000000001010_00000_000_01001_0010011  //addi $t1,$0,10
000000001101_00000_000_01010_0010011  //addi $t2,$0,13
0000001_01001_01001_000_01011_0110011 //mul  $t3,$t1,$t1
000000000001_01001_000_01001_0010011  //addi $t1,$t1,1
0100000_01001_01010_000_01010_0110011 //sub  $t2,$t2,$t1
0000000_01010_01001_111_01011_0110011 //and  $t3,$t1,$t2
0000000_01011_01010_110_01100_0110011 //or   $t4,$t2,$t3
```

Modules

testbench.v

CPU.v

- Control.v
- Adder.v
- PC.v
- Instruction_Memory.v
- Registers.v
- MUX32.v
- MUX5.v
- Sign_Extend.v
- ALU.v
- ALU_Control.v



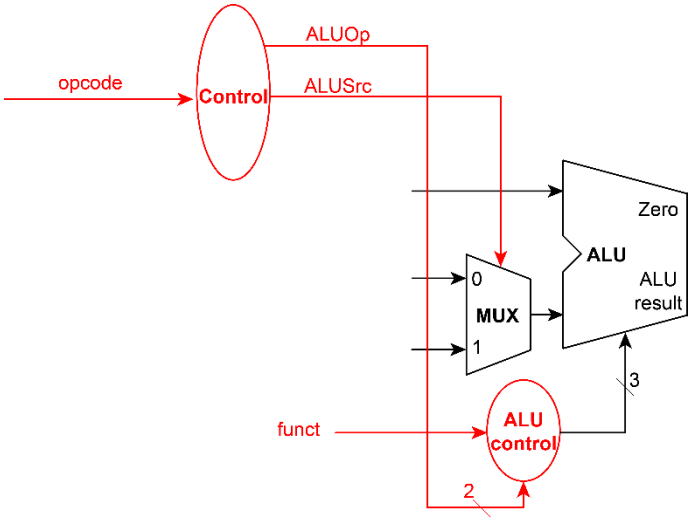
testbench.v

```
`define CYCLE_TIME 50

module TestBench;

reg                 Clk;
reg                 Reset;
reg                 Start;
integer             i, outfile, counter;

always #(`CYCLE_TIME/2) Clk = ~Clk;

CPU CPU(
    .clk_i  (Clk),
    .rst_i  (Reset),
    .start_i(Start)
);

initial begin
    counter = 0;

    // initialize instruction memory
    for(i=0; i<256; i=i+1) begin
        CPU.Instruction_Memory.memory[i] = 32'b0;
    end


    // initialize Register File
    for(i=0; i<32; i=i+1) begin
        CPU.Registers.register[i] = 32'b0;
    end

    // Load instructions into instruction memory
    $readmemb("instruction.txt", CPU.Instruction_Memory.memory);

    // Open output file
    outfile = $fopen("output.txt") | 1;

    Clk = 0;
    Reset = 0;
    Start = 0;

    #(`CYCLE_TIME/4)
    Reset = 1;
    Start = 1;
end
```

CPU.v

```
module CPU
(
    clk_i,
    rst_i,
    start_i
);

// Ports
input               clk_i;
input               rst_i;
input               start_i;

/*
Control Control(
    .Op_i       (),
    .RegDst_o   (),
    .ALUOp_o    (),
    .ALUSrc_o   (),
    .RegWrite_o ()
);
*/

/*
Adder Add_PC(
    .data1_in   (),
    .data2_in   (),
    .data_o     ()
);
*/

PC PC(
    .clk_i      (),
    .rst_i      (),
    .start_i    (),
    .pc_i       (),
    .pc_o       ()
);
```

Control.v / ALU_Control.v



funct7	rs2	rs1	funct3	rd	opcode	function
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
imm[11:0]		rs1	000	rd	0010011	ADDI
0000001	rs2	rs1	000	rd	0110011	MUL

Submission Rule

- Requirement #1
  - Source codes (`*.v` files)
    - `testbench.v`
    - `PC.v`
    - `Register.v`
    - `Instruction_Memory.v`
    - `CPU.v`
    - `Adder.v`
    - `Control.v`
    - `ALU_Control.v`
    - `Sign_Extend.v`
    - `ALU.v`
    - `MUX32.v`
    - `MUX5.v`
- Requirement #2
  - Machine Code text file
    - `instruction.txt` (no need to modify)
    - There is no need to submit "`output.txt`"
  - Report (`hw3_<student_ID>.pdf`)
    - Coding environment
    - Module implementation explanation
    - Either English or Chinese is fine
    - No more than 2 pages

Submission Rule

- Submission format
  - (dir) `hw3_<student_ID>`
    - `hw3_<student_ID>/hw3_<student_ID>.pdf`
    - `hw3_<student_ID>/code/*.v`
  - Pack the folder into a .zip file
  - Case sensitive
- Deadline: 2019/11/20(Wed.) 14:20
- Upload to NTU COOL

Evaluation Criteria

- Report: 15%
- Code: 85%
  - Correctness: 36%
  - Module correct implementation: 49%

- Wrong format: -10%
- Compilation error: coding 0%
  - Please make sure your code can be compiled before submitting
- 10% off per day for late submission

Project Grouping

- 1~3 persons in a group
- Fill your student ID(s) into the form

- This homework is individual work. The grouping is for upcoming projects, not for this.