



2016 Architecture

# Project 2 – Single Cycle CPU

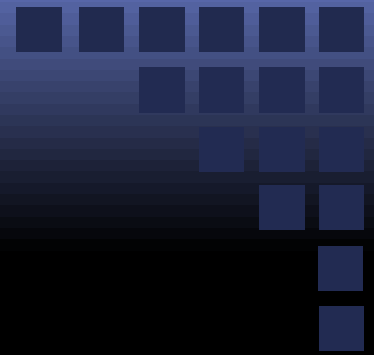


# Outline

- Goal
- Grade



# Outline



## □ Goal

- Instructions
- Project structure
- Initial status

## □ Grade



# Goal

- Implement instructions in behavior model of single cycle CPU
  - Given several components, try to combine them

# R-type instruction

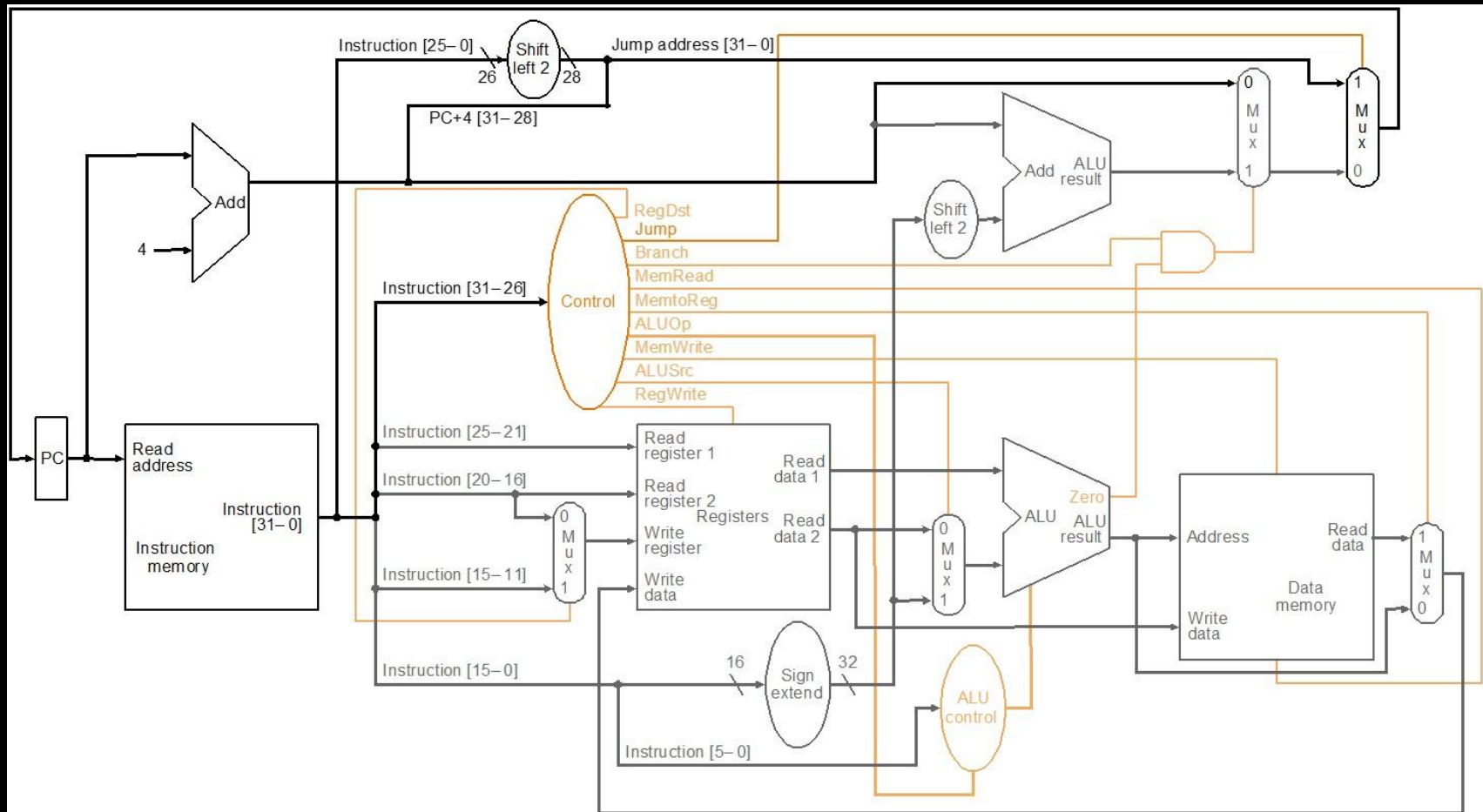
Instruction	Example	Meaning	Op field	Function field
<b>ADD</b> (Addition)	add r1, r2, r3	$r1 = r2 + r3$	0x00	32(0x20)
<b>SUB</b> (Subtraction)	sub r1, r2, r3	$r1 = r2 - r3$	0x00	34(0x22)
<b>AND</b> (Logic And)	and r1, r2, r3	$r1 = r2 \& r3$	0x00	36(0x24)
<b>OR</b> (Logic Or)	or r1, r2, r3	$r1 = r2   r3$	0x00	37(0x25)
<b>SLT</b> (Set on Less Than) signed comparison	slt r1, r2, r3	if ( $r2 < r3$ ) $r1 = 1$ else $r1 = 0$	0x00	42(0x2a)

# I-type instruction

Instruction	Example	Meaning	Op field
<b>ADDI</b> (Add Immediate)	addi r1, r2, 100	$r1 = r2 + 100$	0x08
<b>LW</b>	lw r1, 12(r2)	$r1 = \text{Memory}[r2+12]$	0x23
<b>SW</b>	sw r1, 12(r2)	$\text{Memory}[r2+12] = r1$	0x2B
<b>SLTI</b> (Set on Less Than Immediate)	slti r1, r2, 10	if( $r2 < 10$ ) $r1 = 1$ else $r1 = 0$	0x0A
<b>BEQ</b> (Branch On Equal)	beq r1, r2, 25	if ( $r1 == r2$ ) go to $PC+4+100$	0x04

Immediate is signed for these instructions

# Project Structure



# Project Structure

## ▣ Good coding style

- Only one module in one .v file
- Module name has to be the same as filename
  - ▣ Ex: module MUX() ... endmodule in MUX.v

## ▣ Modules are created and connected in one top-module(Simple\_Single\_CPU.v)

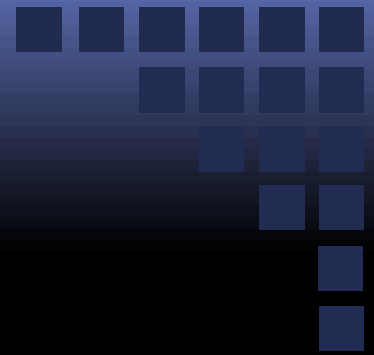
- Some wires in Simple\_Single\_CPU.v (around the Reg\_File) has been connected yet. **DO NOT modify them.**



# Initial status

- Data memory and all registers should be set to 0 at the beginning
- PC start with 0
- Be able to reset program by rst\_i

# Outline



## □ Goal

## □ Grade

- Test case
- List of given source
- Strict rule
- Notice

# Test Case

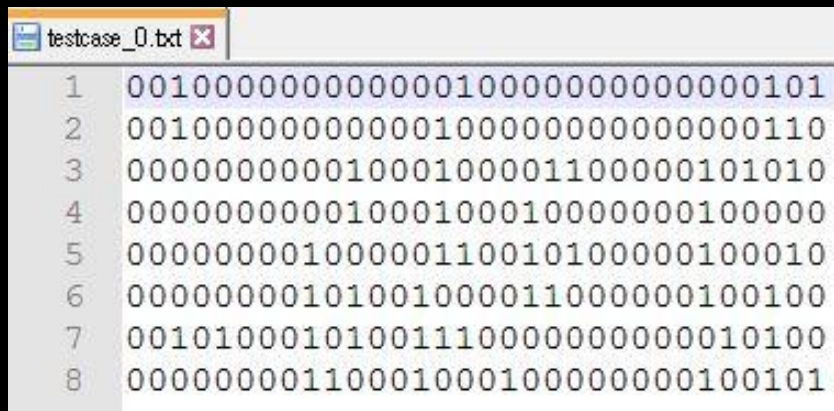
- Three kinds of test case
  - Completely correct or wrong

Test case	Points
ADD, SUB, AND, OR, SLT ADDI, SLTI	60%
+BEQ	20%
+LW,SW	20%

# Example

## ■ Testcase 0

- Format : Each line occupied by one assembly code



testcase\_0.txt

```
1 00100000000000010000000000000101
2 00100000000000010000000000000110
3 00000000001000100001100000101010
4 00000000001000100010000000100000
5 00000000100000110010100000100010
6 00000000101001000011000000100100
7 00101000101001110000000000010100
8 00000000110001000100000000100101
```

### Testcase\_0.txt

```
addi $1, $0, 5
addi $2, $0, 6
slt $3, $1, $2
add $4, $1, $2
sub $5, $4, $3
and $6, $5, $4
slti $7, $5, 20
or $8, $6, $4
```

### Result

```
$1=5, $2=6, $3=1,
$4=11, $5=10, $6=10
$7=1, $8=11
```

# Example

## ▣ Testcase 1

### Testcase\_1.txt

```
addi $1, $0, 19
addi $2, $0, 8
beq $0, $0, 2
slt $3, $2, $1
add $4, $1, $2
sub $5, $4, $3
and $6, $5, $4
slti $7, $5, 20
or $8, $1, $2
```

### Result

```
$1=19,$2=8,$3=0,
$4=0,$5=0,$6=0
$7=1,$8=27
```

# Example

## ▣ Testcase 2

### Testcase\_2.txt

```
addi $1, $0, 19
addi $2, $0, 8
sw $1, 0($10)
beq $0, $0, 1
slt $3, $2, $1
add $4, $1, $2
sub $5, $4, $3
beq $2, $0, 3
and $6, $5, $4
slti $7, $5, 20
lw $8, 0($10)
or $9, $1, $3
addi $1, $1, -4
```

### Result

\$1=15, \$2=8, \$3=0,  
\$4=27, \$5=27, \$6=27  
\$7=0, \$8=19, \$9=19

# Source

- We will provide total 13 .v files
  - Adder.v
  - ALU.v
  - ALU\_Ctrl.v
  - Data\_Memory.v
  - Decoder.v
  - Instr\_Memory.v (do not modify)
  - MUX\_2to1.v
  - ProgramCounter.v
  - Reg\_File.v (do not modify)
  - Shift\_Left\_Two\_32.v
  - Sign\_Extend.v
  - Simple\_Single\_CPU.v (the top-module)
  - Test\_bench.v (do not modify)
- We provide a Makefile for you to write project. You don't have to upload it to iLMS.

# Run simulation with makefile

- Put the makefile in your file directory
  - 1. type “make”
  - 2. makefile will do *\$ ncverilog all .v file you need*
- By using the makefile, you don't have to type “ncverilog ...” all the time



# Strict rule

- 1. Put all your .v file under directory “singleCycle” and compress as “singleCycle\_學號.tar.gz” (No file I/O)
- 2. Do not modify Instr\_Memory.v, Test\_Bench.v, Reg\_File.v. TA will replace it with our default version
- 3. Do not modify the wires that has been connected in Simple\_Single\_CPU.v

# Strict rule

- 4. Do not add/modify any clock and rst settings in your program. (ex: #delay)
- 5. Do not change any file name of the given .v files.
- 6. There shouldn't be \$stop in your code

# Compress Your Folder

- Your target folder name must be `singleCycle_學號`
- `tar -zcvf singleCycle_學號.tar.gz singleCycle_學號`

# Notice

- Deadline : 5/2, 23:59
- Please upload “singleCycle\_學號.tar.gz” to iLMS.
- Do NOT mail your code to TA. We will use your submitted file on iLMS to grade your project.

# Notice

- Violation of each strict rule **-30 points**
- Program not implemented by connected modules **Get 0 point**
- Copy ----- **Get 0 point**
- This project is related to Project 3, please work on it as soon as possible.