COMPUTER ARCHITECTURE HW4

B04902021 陳弘梵

CPU.v

- Read in "clock" and connect all of the small modules
- 2 WIREs: Inst_addr for Program Counter output, inst for instruction output

PC.v

- React to "clock" to fetch new instructions

Adder.v

- PC + 4 then output back to PC

Instruction_Memory.v

- Testing machine code will be written in reg, fetch it by input address

MUX5.v

- Choose the register ready for written by Boolean selection

Registers.v

- RDaddr from MUX5, RDdata from ALU, RegWrite from Control.v

Sign_Extend.v

- Fill the front 16 bits with the 15th bit of input

Control.v

- Truth table:

type	R-type	lw	sw	beq
Op	000000	100011	101011	000100
output	1001000010	0111100000	x1x0010000	x0x0001001

type	jump	Addi	Default
Op	000010	001000	
output	0000000100	0101000000	xxxxxxxxxx

```
RegDst_o   = output [9]
ALUSrc_o   = output [8]
RegWrite_o = output [6]
ALUOp_o[1] = output [1]
ALUOp_o[0] = output [0]
```

MUX32.v

- Choose the data from RTaddr or after sign_extend from Control.v

ALU_Control.v

- Truth table:

ALUOp_input[2]	ALU_Ctrl_output
00	010
01	110
10	funct[1], ~funct[2], funct[0]||funct[3]
default	xxx

Implementation:

For most of the wire connections, I used the "**assign**" syntax with continuous judge statement (a==b? a:b) to hand over the value instead of using new regs to store them. Also using some **direct truth table** checking instead of PLA-like logic computation.