Course Design of Computer Organization

Wang Zonghui

College of Computer Science & Technology,
Zhejiang University
March, 2010

topics

- 1. Verilog and Xilinx ISE
- 2. Basic logic component of datapath design
- 3. ALU and the ALU controller design
- 4. R-type instruction processor design
- 5. CPU controller design
- 6. Single-cycle clock CPU design
- 7. Multiple-cycle clock CPU design
- 8. Microprogrammed CPU controller unit design
- 9. Design of datapath of Microprogrammed CPU

Outline

- Experiment Purpose
- Experiment Task
- Basic Principle
- Operating Procedures

Experiment Purpose

- Understand the principles of Multiple-cycle clock CPU
- Master design methods of Multiple-cycle clock CPU supporting 7 common MIPS Instruction
- Master methods of program verification of CPU

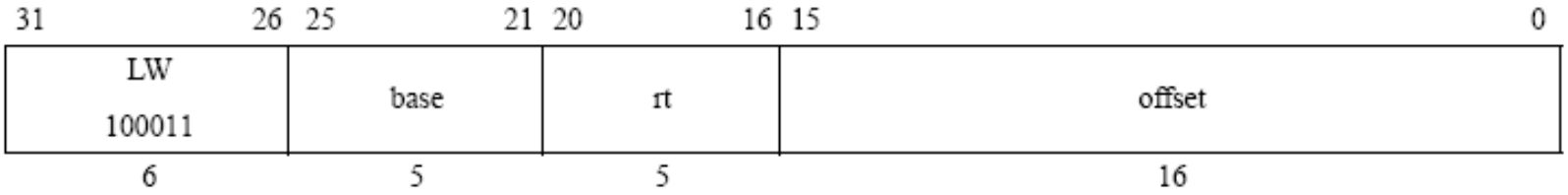
Experiment Task

- Design the Datapath, bring together the basic units into Multiple-cycle clock CPU
- Verify the CPU with program and observe the execution of program

Basic Principle

- Instructions
- Circuit diagram of Multiple-cycle clock CPU.
- Constitution
- Basic components
- IP Core for Memory

Instruction - LW



Format: `LW rt, offset(base)` MIPS32

Purpose:

To load a word from memory as a signed value

Description: `rt ← memory[base+offset]`

The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched, sign-extended to the GPR register length if necessary, and placed in GPR *rt*. The 16-bit signed *offset* is added to the contents of GPR *base* to form the effective address.

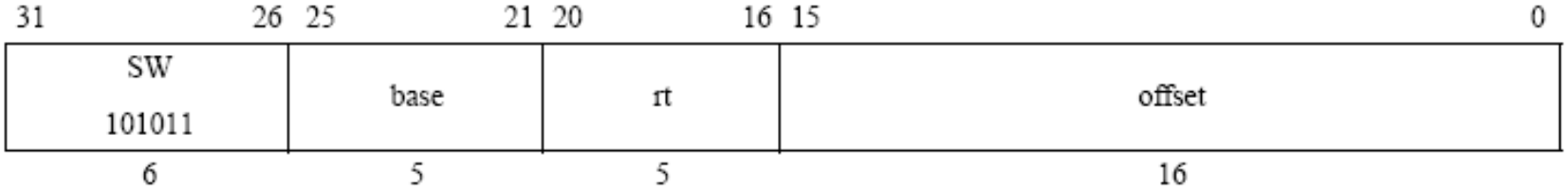
Restrictions:

The effective address must be naturally-aligned. If either of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

Operation:

```
vAddr ← sign_extend(offset) + GPR[base]
if vAddr1..0 ≠ 0^2 then
   SignalException(AddressError)
endif
(pAddr, CCA)← AddressTranslation (vAddr, DATA, LOAD)
memword← LoadMemory (CCA, WORD, pAddr, vAddr, DATA)
GPR[rt]← memword
```

Instruction - SW



Format: `SW rt, offset(base)` MIPS32

Purpose:

To store a word to memory

Description: `memory[base+offset] ← rt`

The least-significant 32-bit word of register *rt* is stored in memory at the location specified by the aligned effective address. The 16-bit signed *offset* is added to the contents of GPR *base* to form the effective address.

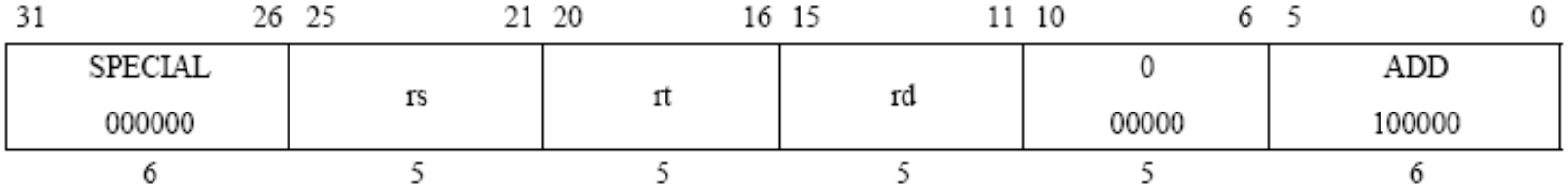
Restrictions:

The effective address must be naturally-aligned. If either of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

Operation:

```
vAddr ← sign_extend(offset) + GPR[base]
if vAddr1..0 ≠ 0^2 then
    SignalException(AddressError)
endif
(pAddr, CCA)← AddressTranslation (vAddr, DATA, STORE)
dataword← GPR[rt]
StoreMemory (CCA, WORD, dataword, pAddr, vAddr, DATA)
```

Instruction - ADD



Format: `ADD rd, rs, rt` MIPS32

Purpose:

To add 32-bit integers. If an overflow occurs, then trap.

Description: `rd ← rs + rt`

The 32-bit word value in GPR *rt* is added to the 32-bit value in GPR *rs* to produce a 32-bit result.

- If the addition results in 32-bit 2's complement arithmetic overflow, the destination register is not modified and an Integer Overflow exception occurs.
- If the addition does not overflow, the 32-bit result is placed into GPR *rd*.

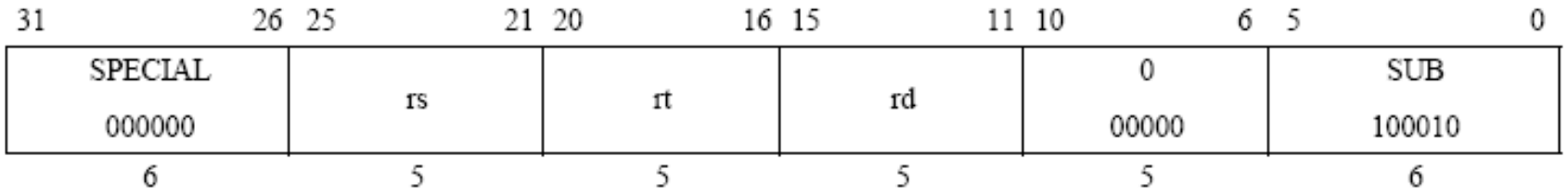
Restrictions:

None

Operation:

```
temp ← (GPR[rs]31||GPR[rs]31..0) + (GPR[rt]31||GPR[rt]31..0)
if temp32 ≠ temp31 then
   SignalException(IntegerOverflow)
else
   GPR[rd] ← temp
endif
```

Instruction - SUB



Format: `SUB rd, rs, rt` MIPS32

Purpose:

To subtract 32-bit integers. If overflow occurs, then trap

Description: `rd ← rs - rt`

The 32-bit word value in GPR *rt* is subtracted from the 32-bit value in GPR *rs* to produce a 32-bit result. If the subtraction results in 32-bit 2's complement arithmetic overflow, then the destination register is not modified and an Integer Overflow exception occurs. If it does not overflow, the 32-bit result is placed into GPR *rd*.

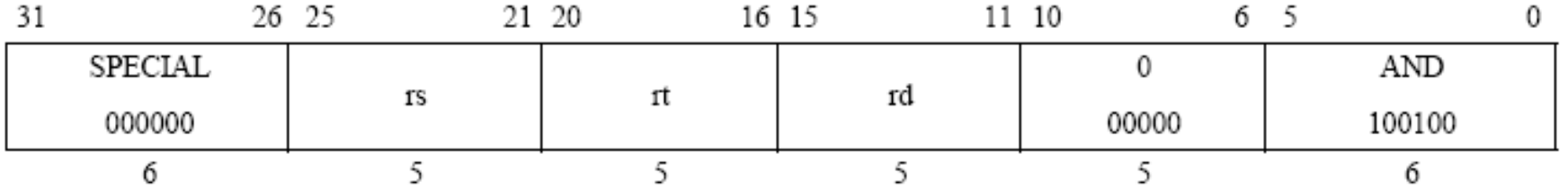
Restrictions:

None

Operation:

```
temp ← (GPR[rs]31||GPR[rs]31..0) - (GPR[rt]31||GPR[rt]31..0)
if temp32 ≠ temp31 then
   SignalException(IntegerOverflow)
else
   GPR[rd] ← temp31..0
endif
```

Instruction - AND



Format: `AND rd, rs, rt` MIPS32

Purpose:

To do a bitwise logical AND

Description: `rd ← rs AND rt`

The contents of GPR *rs* are combined with the contents of GPR *rt* in a bitwise logical AND operation. The result is placed into GPR *rd*.

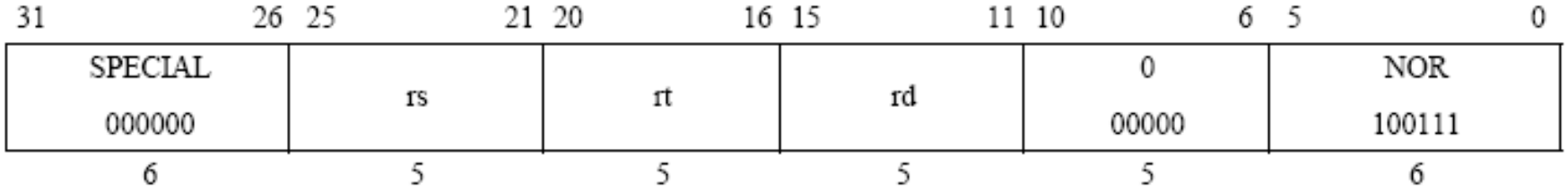
Restrictions:

None

Operation:

```
GPR[rd] ← GPR[rs] and GPR[rt]
```

Instruction - NOR



Format: `NOR rd, rs, rt` MIPS32

Purpose:

To do a bitwise logical NOT OR

Description: `rd ← rs NOR rt`

The contents of GPR *rs* are combined with the contents of GPR *rt* in a bitwise logical NOR operation. The result is placed into GPR *rd*.

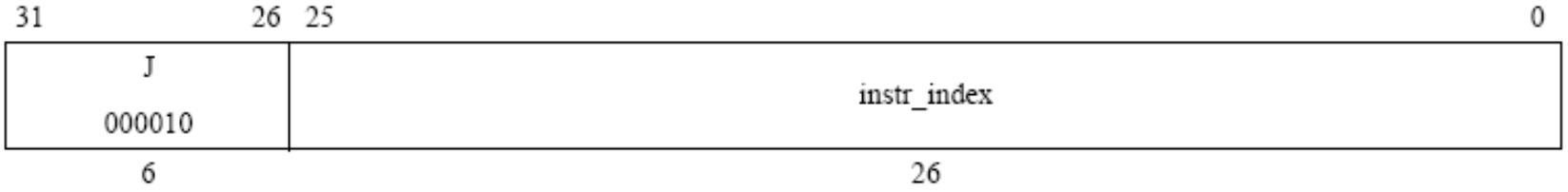
Restrictions:

None

Operation:

```
GPR[rd] ← GPR[rs] nor GPR[rt]
```

Instruction - JMP



Format: `J target` **MIPS32**

Purpose:

To branch within the current 256 MB-aligned region

Description:

This is a PC-region branch (not PC-relative); the effective target address is in the "current" 256 MB-aligned region. The low 28 bits of the target address is the *instr_index* field shifted left 2 bits. The remaining upper bits are the corresponding bits of the address of the instruction in the delay slot (not the branch itself).

Jump to the effective target address. Execute the instruction that follows the jump, in the branch delay slot, before executing the jump itself.

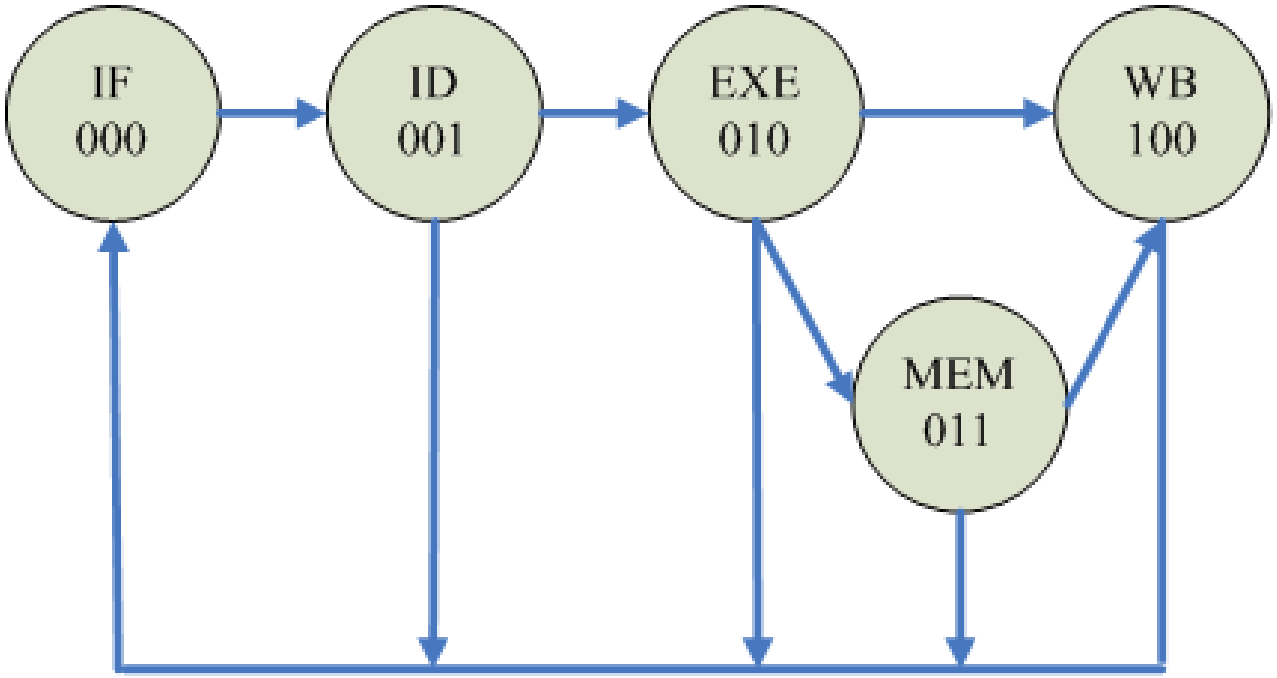
Restrictions:

Processor operation is **UNPREDICTABLE** if a branch, jump, ERET, DERET, or WAIT instruction is placed in the delay slot of a branch or jump.

Operation:

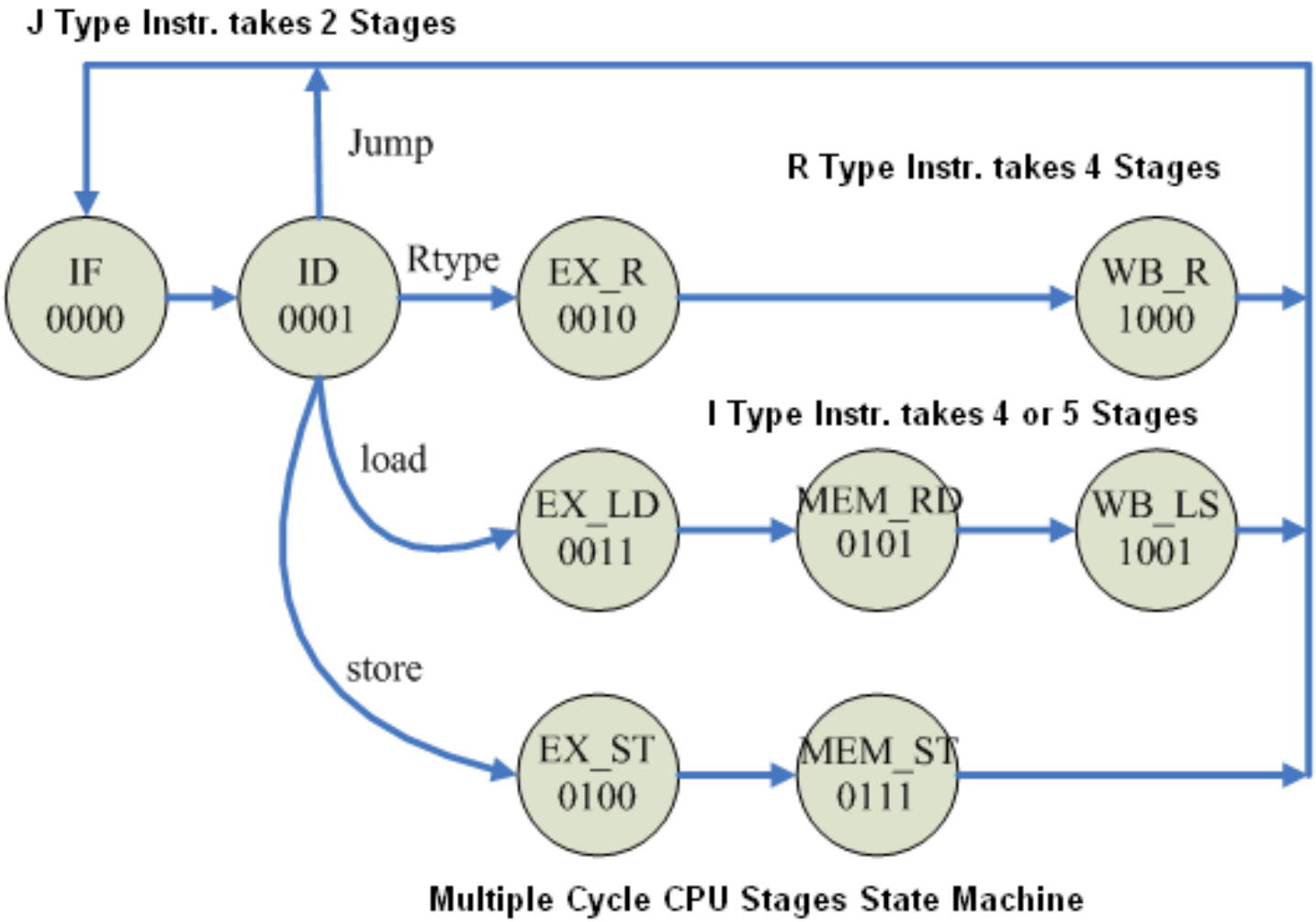
```
I:
I+1:PC ← PC_GPRLEN-1..28 || instr_index || 0^2
```

The principle of CPU Controller(1)

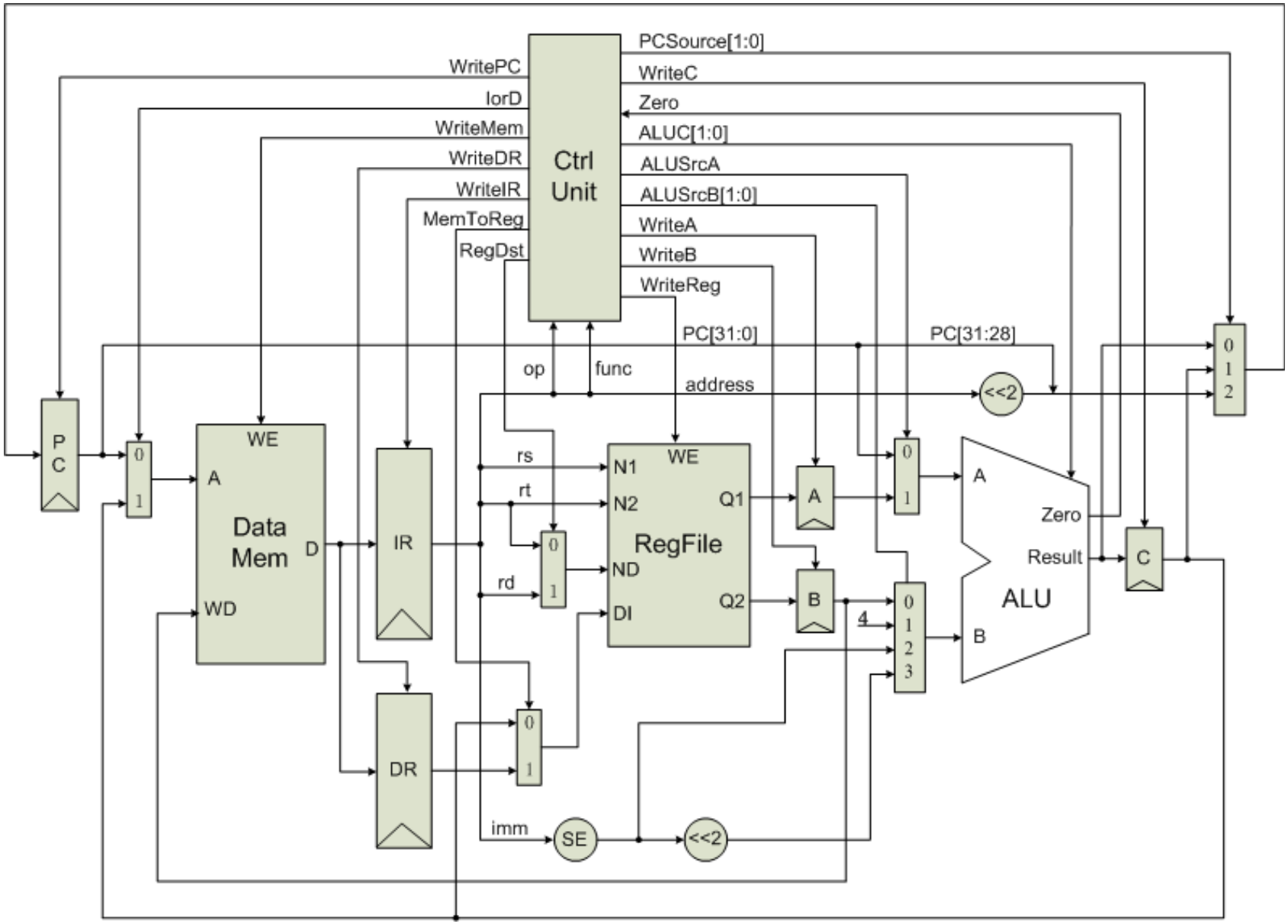


Stages of Multiple-Cycle Execution of Typical MIPS CPU

The principle of CPU Controller(2)



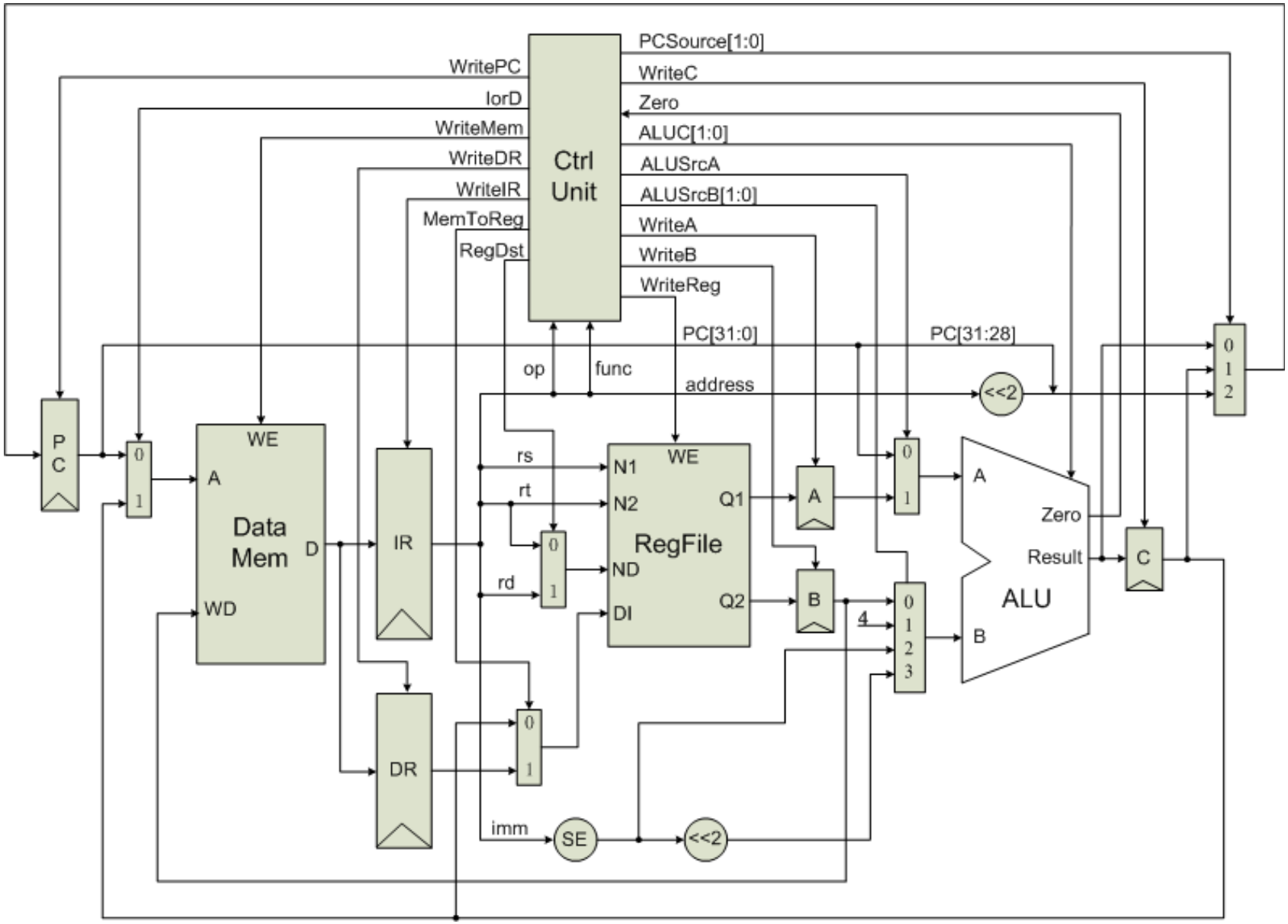
The principle of CPU Controller



CPU Controller Output

	Output Signal	Meaning When 1	Meaning When 0
1	PCSrc[1:0]	00: PC + 4;01: Branch Instr.;10: jump Instr	
2	WritePC	Write PC	Not Write PC
3	IorD	Instruction Addr	Data Addr.
4	WriteMem	Write Mem.	Not Write Mem.
5	Write DR	Write Data. Reg	Not Write Data. Reg
6	Write IR	Write Instr. Reg	Not Write Instr. Reg
7	MemToReg	From Mem. To Reg	From ALUOut To Reg
8	RegDest	rd	rt
9	ALUC	ALU Controller Op	
10	ALUSrcA	Register rs	PC
11	ALUSrcB	Selection:00:Reg rt; 01:4; 10:Imm.; 11: branch Address	
12	WriteA	Write A Reg.	Not Write A Reg.
13	WriteB	Write B Reg.	Not Write B Reg.
14	WriteC	Write C Reg.	Not Write C Reg.
15	WriteReg	Write Reg.	Not Write Reg.

The Datapath of CPU Controller



Basic Units of Multiple-cycle CPU

- CPU Controller
- ALU and ALU Controller
- Register file
- Mem. (Instruction and Data Mem.).
- others： Register, Sign-extend Unit, Shifter, Multiplexor

Memory

- Memory
  - Dual Port Block Memory
  - Port A: Read Only, Width: 32, Depth: 512
  - Port B: Read and Write, Read After Write
  - Rising Edge Triggered

Experiment Content

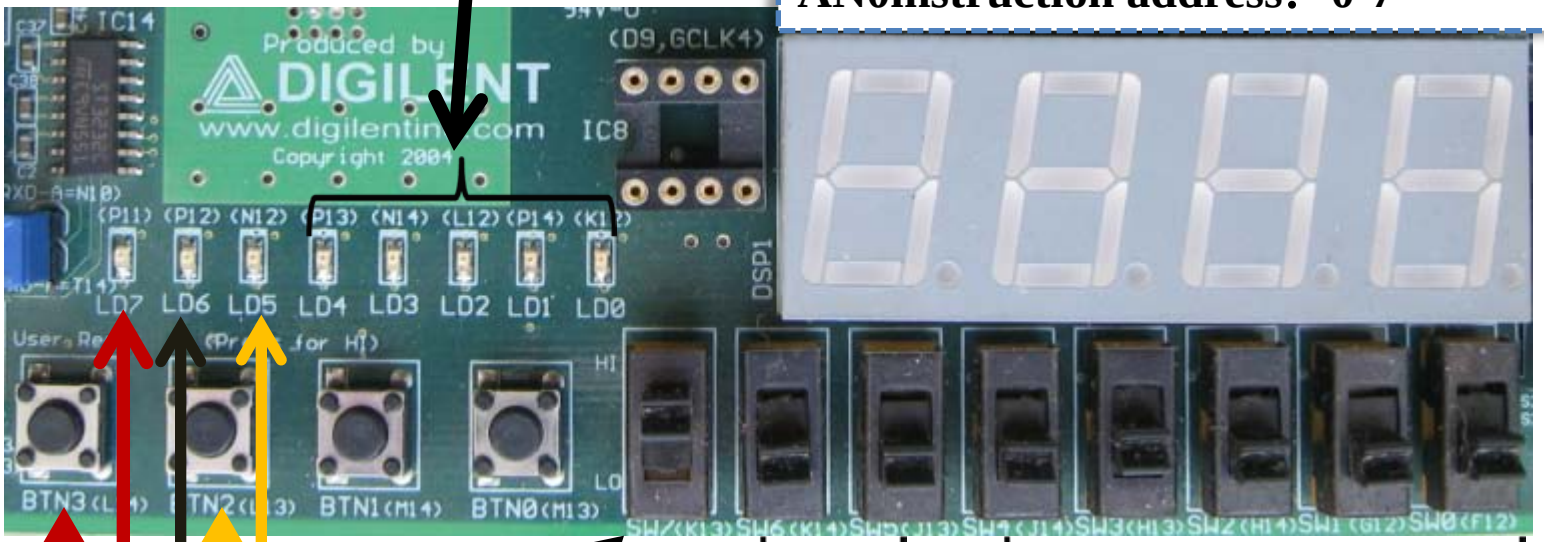
- 1. Create project and write code.
- 2. Generate Mem. (Instr. and Data).
- 3. Write UCF file.
- 4. Synthesize /Implement.
- 5. Program the bit file and verify it.

Top Module

```
memory x_memory(.addra(raddr),.addrb(waddr),.clka(clk),
    .clkb(clk),.dinb(b_data),.douta(mem_data),.web(write_mem));

ctrl x_ctrl(clk, rst, ir_data, zero,write_pc, iord, write_mem, write_dr, write_ir,
    memtoreg, regdst, pcsource, write_c, alu_ctrl, alu_srcA, alu_srcB, write_a,
    write_b, write_reg, state_out, insn_type, insn_code, insn_stage);

pcm x_pcm(clk, rst, alu_out, c_data, ir_data, pcsource, write_pc,pc);

alu_wrapper x_alu_wrapper(a_data, b_data, ir_data, pc, alu_srcA, alu_srcB,
    alu_ctrl, zero, alu_out);

reg_wrapper x_reg_wrapper(clk, rst, ir_data, dr_data, c_data, memtoreg,
    regdst, write_reg, rdata_A, rdata_B, r6out);
```

The stage of current instruction
（Repeat digital tube display）

States of current instruction
AN3 clock counter：0,1,2,3,4,5,6, ...
AN2instruction type：1-3（R, I, J）
AN1instruction stage：0,1,[2,[3,[4]]]
AN0instruction address：0-7



Manual Clock

Auto clock

Reset signal

CPU mode
L: manual mod
H: auto mode

Seven-Segment Digital Tube Display options

Register No. Setting (Total 5)

input	display
00_xxxxx	Register xxxxx low 16 bits
01_xxxxx	Registerxxxxx high 16 bits
10_xxxxx	States of current instruction
11_xxxxx	Customize

- Auto mode does not accept the manual clock
- Manual mode does not accept the auto clock
- LED to display the stage

Program for verification

<0> lw r1, $20(r0); 0x8c01_0014 State:0,1,3,5,9 Type:3 Code:1 (LD)

<1> lw r2, $21(r0); 0x8c02_0015 State:0,1,3,5,9 Type:3 Code:1 (LD)

<2> add r3, r1, r2; 0x0022_1820 State:0,1,2,8 Type:1 Code:3 (AD)

<3> sub r4, r1, r2; 0x0022_2022 State:0,1,2,8 Type:1 Code:4 (SU)

<4> and r5, r3, r4; 0x0064_2824 State:0,1,2,8 Type:1 Code:5 (AN)

<5> nor r6, r4, r5; 0x0085_3027 State:0,1,2,8 Type:1 Code: 6 (NO)

<6> sw r6, $22(r0); 0x ac06_0016 State:0,1,4,7 Type:3 Code: 2 (ST)

<7> J 0; 0x0800_0000 State:0,1 Type:2 Code:7 (JP)

DataMem(20) = 0xbeef_0000 ;

DataMem(21) =0x0000_beef ;

Thanks!