VE370 RC (Midterm) T4 & 6

2020/10/17

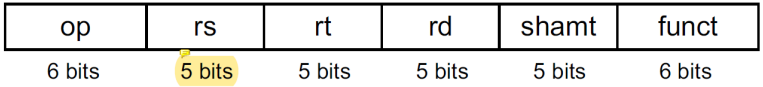
Outline

- Instruction Coding
- Single Cycle Processor

Outline

- Instruction Coding
- Single Cycle Processor

R-format



- Instruction fields
  - op: operation code (opcode)
  - rs: first source register number
  - rt: second source register number
  - rd: destination register number
  - shamt: shift amount (00000 for now)
  - funct: function code (extends opcode)

- op, funct: check the MIPS Reference Card
- shamt: only used in shift operations

R-format Example

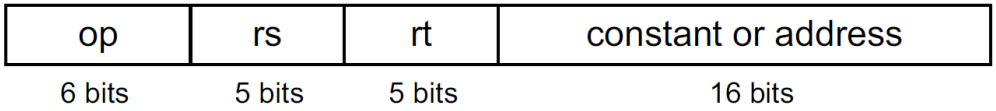
```
sll $t1, $s3, 2
```

op (6 bits)	rs (5 bits)	rt (5 bits)	rd (5 bits)	shamt (5 bits)	funct (6 bits)
0	0	$s3	$t1	2	0
0	0	19	9	2	0
000000	00000	10011	01001	00010	000000

- Result:

  $0000\ 0000\ 0001\ 0011\ 0100\ 1000\ 1000\ 0000_2 = 00134880_{16}$

I-format



16-bit immediate number or address

- rs: source or base address register
- rt: destination or source register
- Constant: -2^{15} to $+2^{15} - 1$
- Address: offset added to base address in rs

- Note the field for immediate number only has 16 bits

I-format Examples (T4 pp. 8 – 10)

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

- `addi $t0, $s0, 4`

op	$s0	$t0	4

- `lw $t0, 4($s0)`

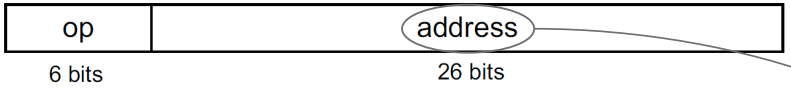
op	$s0	$t0	4

- `beq $s0, $t0, LOOP`

op	$s0	$t0	Relative Address

  - LOOP (Target address) = PC + 4 + Relative Address (RA) × 4
  - Equivalently, RA = (Target address – PC – 4)/4

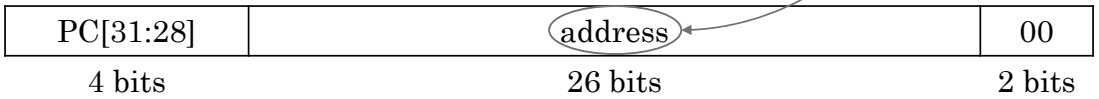
J-format



Encode full address in instruction

(Pseudo) Direct jump

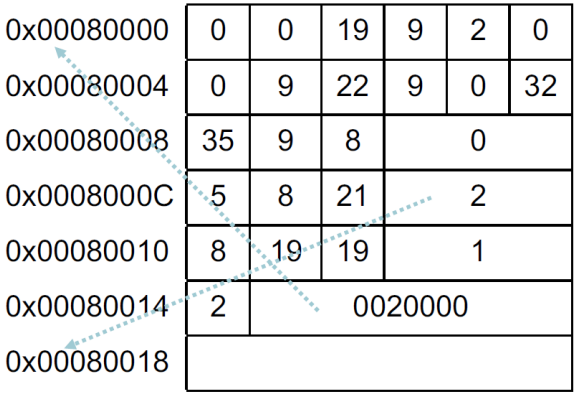
- Target address = PC[31:28] : (address × 4)



- The unchanged first 4 bits set the limitation for jump
- 00 at the end of target address: each instruction is a word

Example (T4 pp. 13)

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```



- bne address field: (0x00080018 – 0x0008000C – 4)/4 = 2

  Target addr. PC

- j address field: 0x0080000/4 = 0020000
  - Target address: 0x0(0020000 × 4) = 0x00080000
  - 0: PC[31:28]

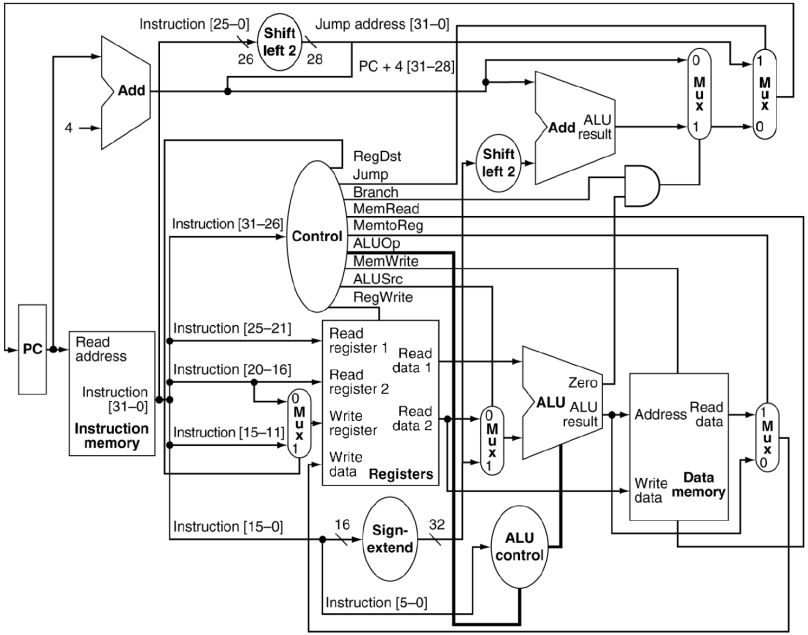
MIPS Addressing Mode (T4 pp. 15-20)

- Immediate Addressing
  - In I-type like `addi $t0, $s0, -1`
- Register Addressing
  - In R-type and I-type like `beq $s0, $s1, FUNCTION`
- **Base Addressing**
  - In I-type (memory-related) like `lw $t0, 32($s0)`
- **PC-relative addressing**
  - In near branch like `beq $s0, $s1, FUNCTION`
- **Pseudodirect addressing**
  - In J-type instructions like `j` and `jal`

Outline

- Instruction Coding
- Single Cycle Processor

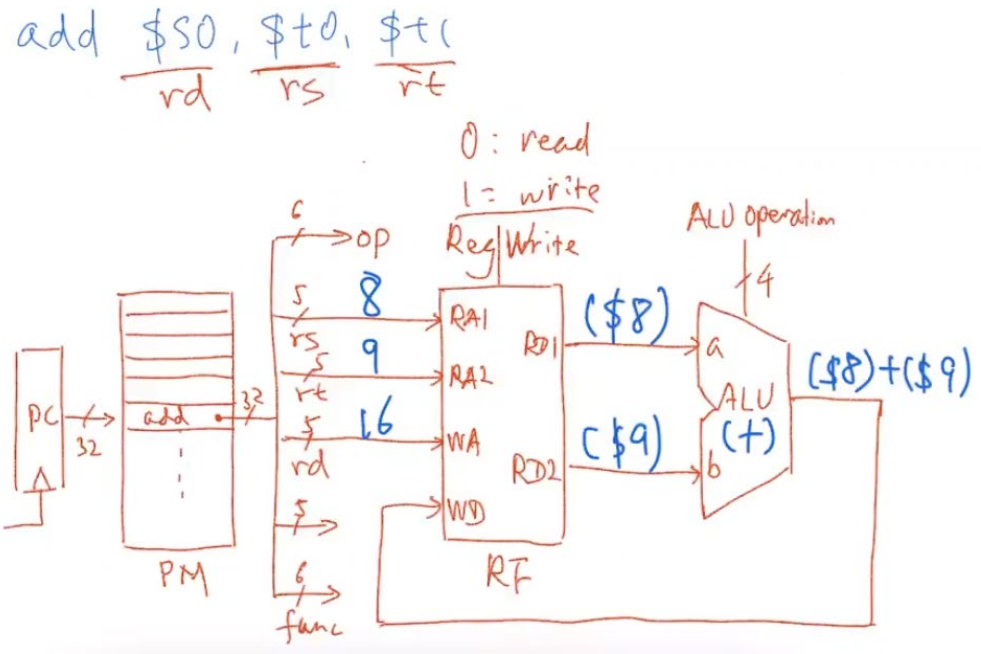
Single Cycle Processor



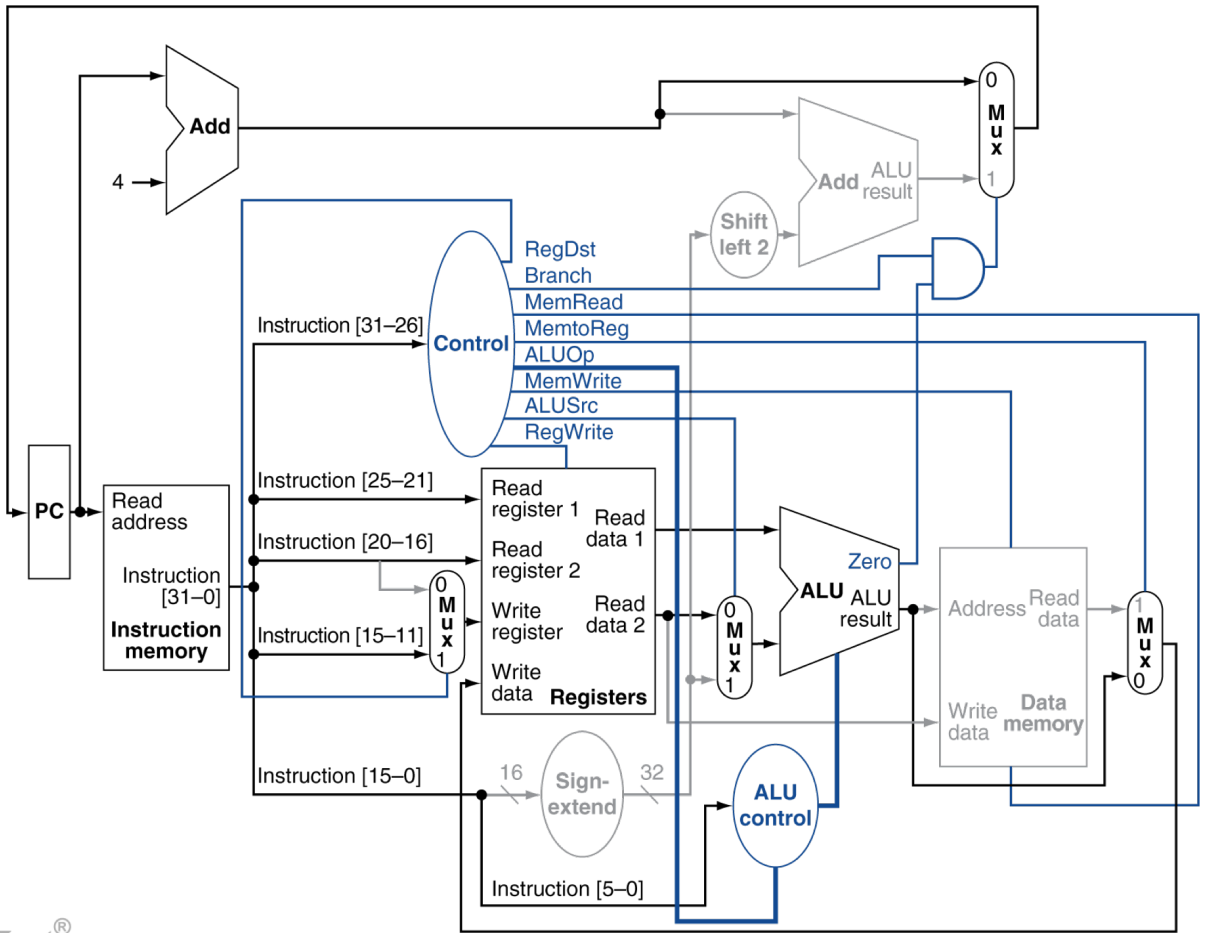
R-Format Instructions

- Instruction fetch
- Read from register file
- ALU calculation
- Write back to register file

Example: `add $s0, $t0, $t1`



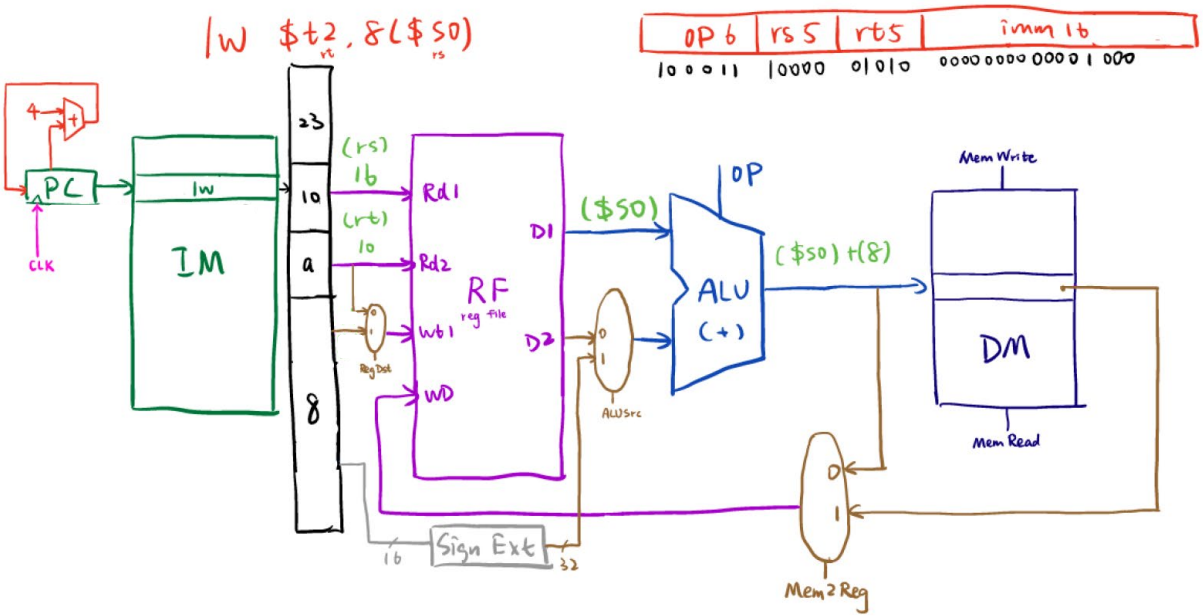
R-Type Instruction



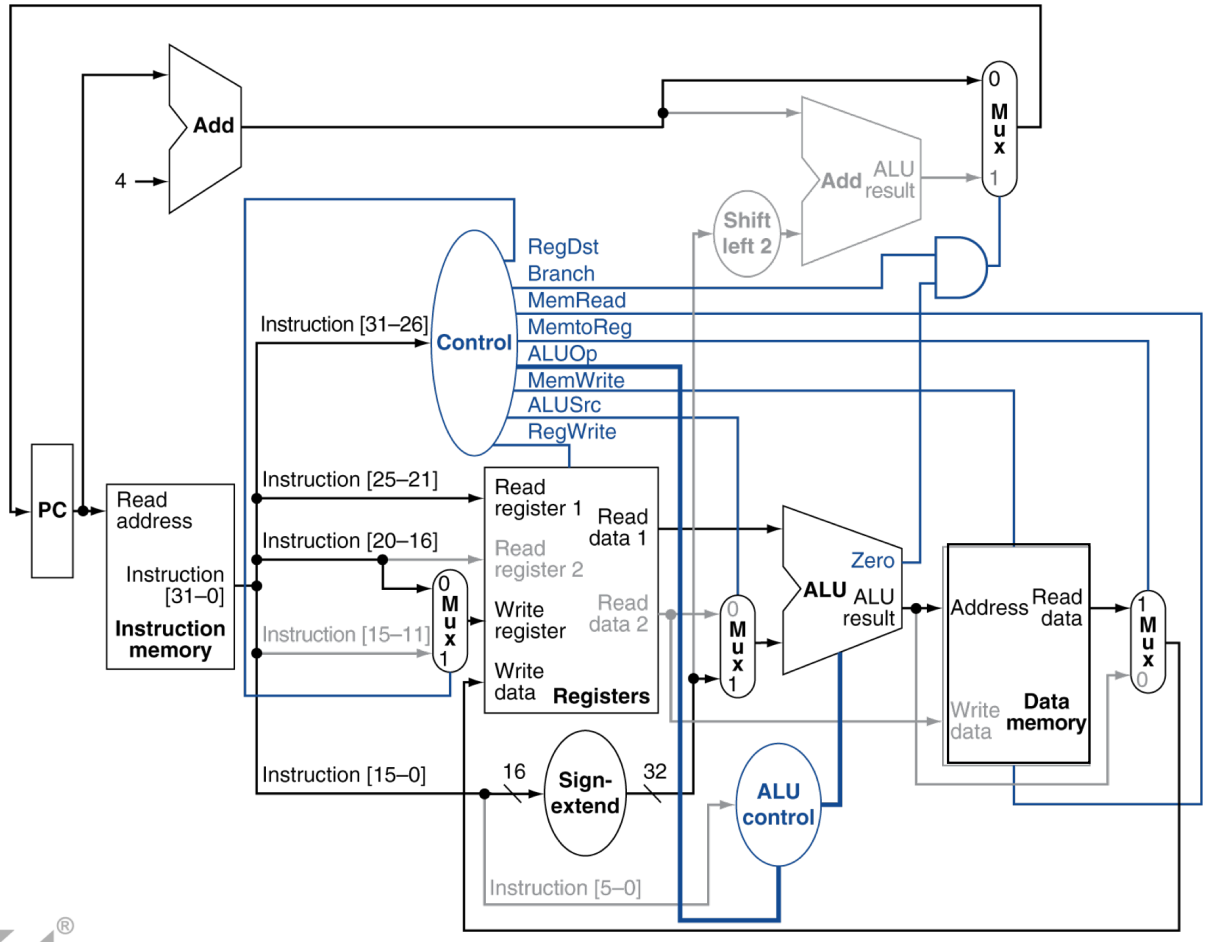
Load/Store Instructions

- Instruction fetch
- Read from reg file
- **Sign extend immediate number**
- ALU calculation
- **Load: Read data memory and update register**
- **Store: Write register value to memory**

Example: `lw $t2, 8($s0)`



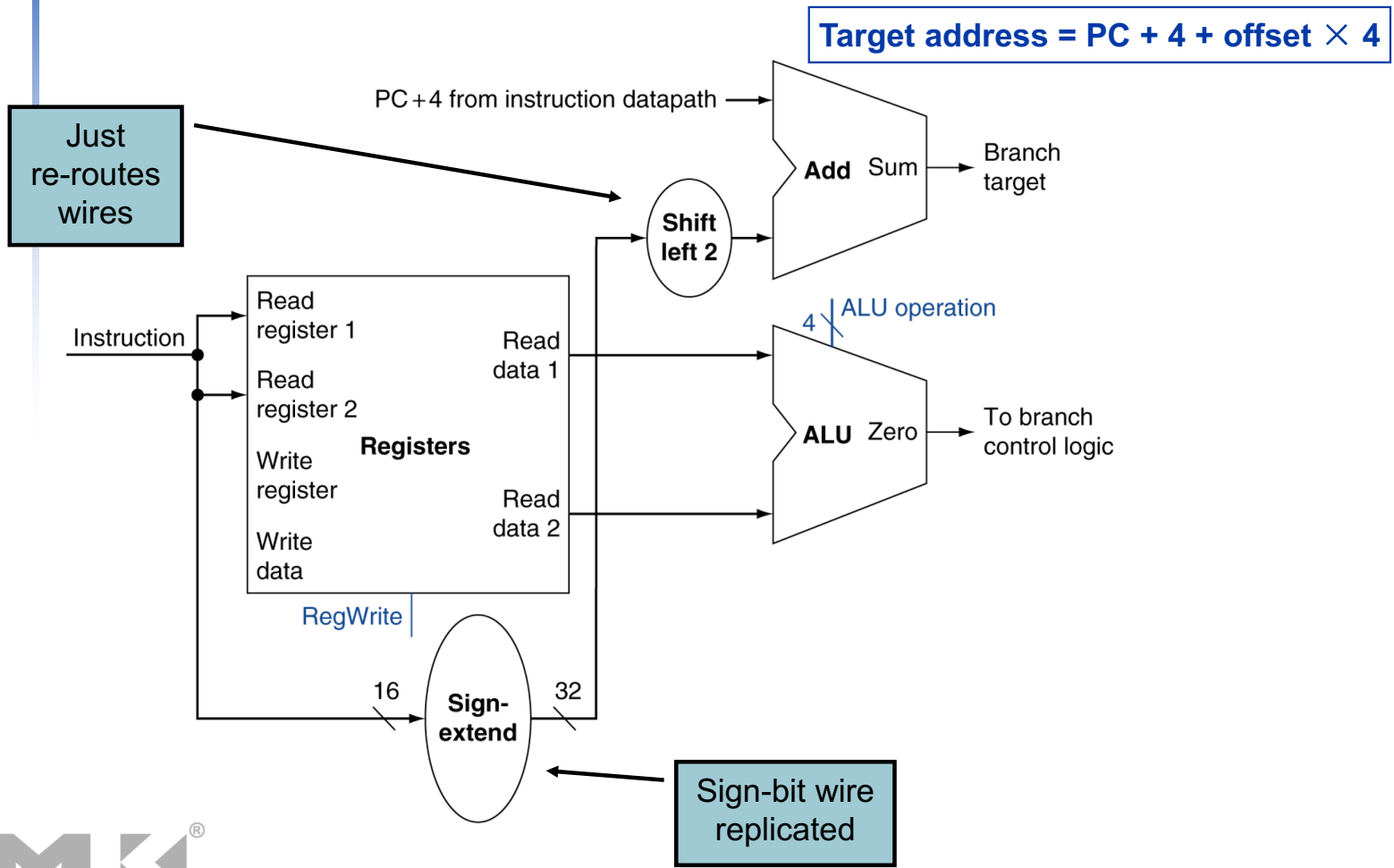
Load Instruction



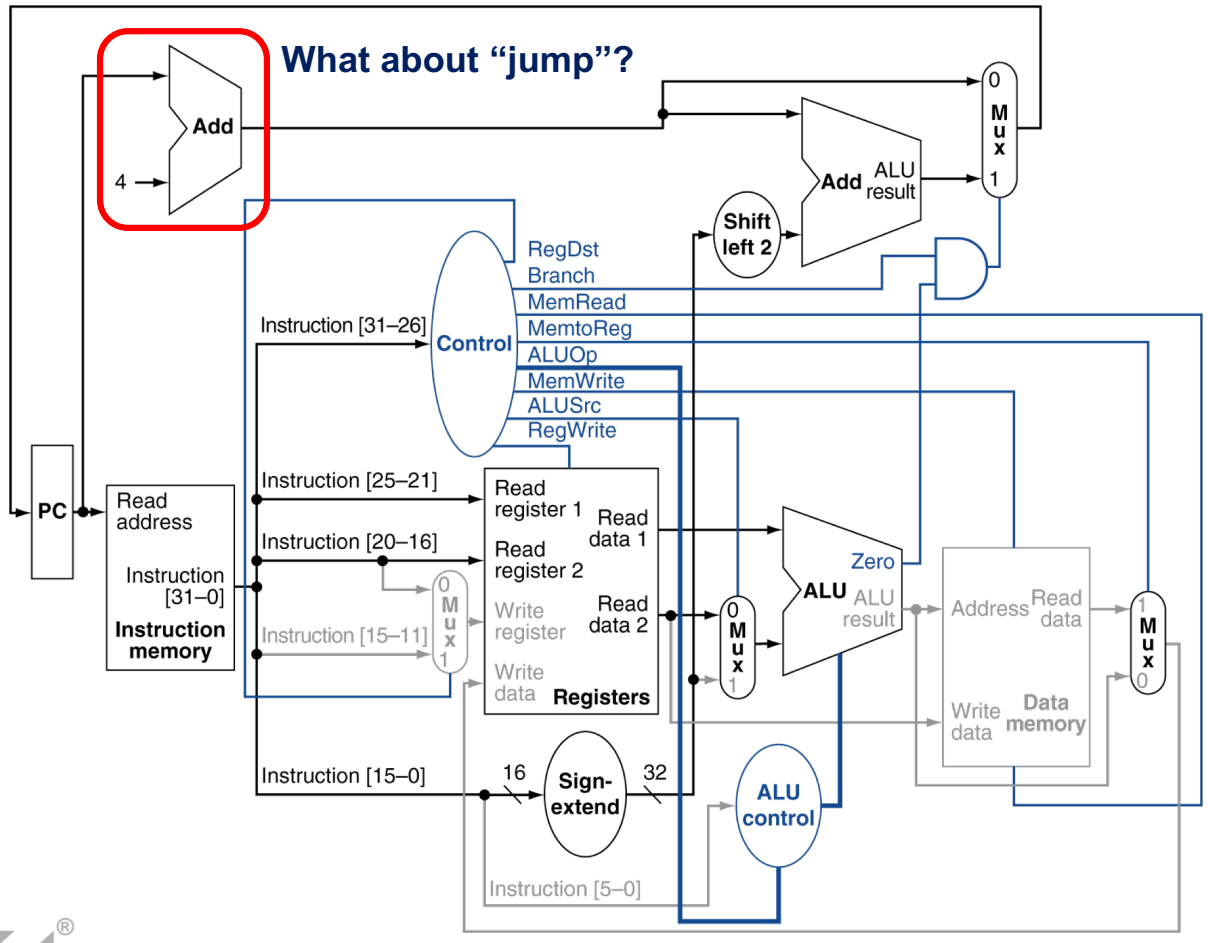
Branch Instructions

- Instruction fetch
- Read from reg file
- Calculate target address
  - Sign-extend the immediate relative address
  - Shift left 2 places
  - Add to PC + 4
- Compare operands
  - ALU calculation, subtract and check Zero output

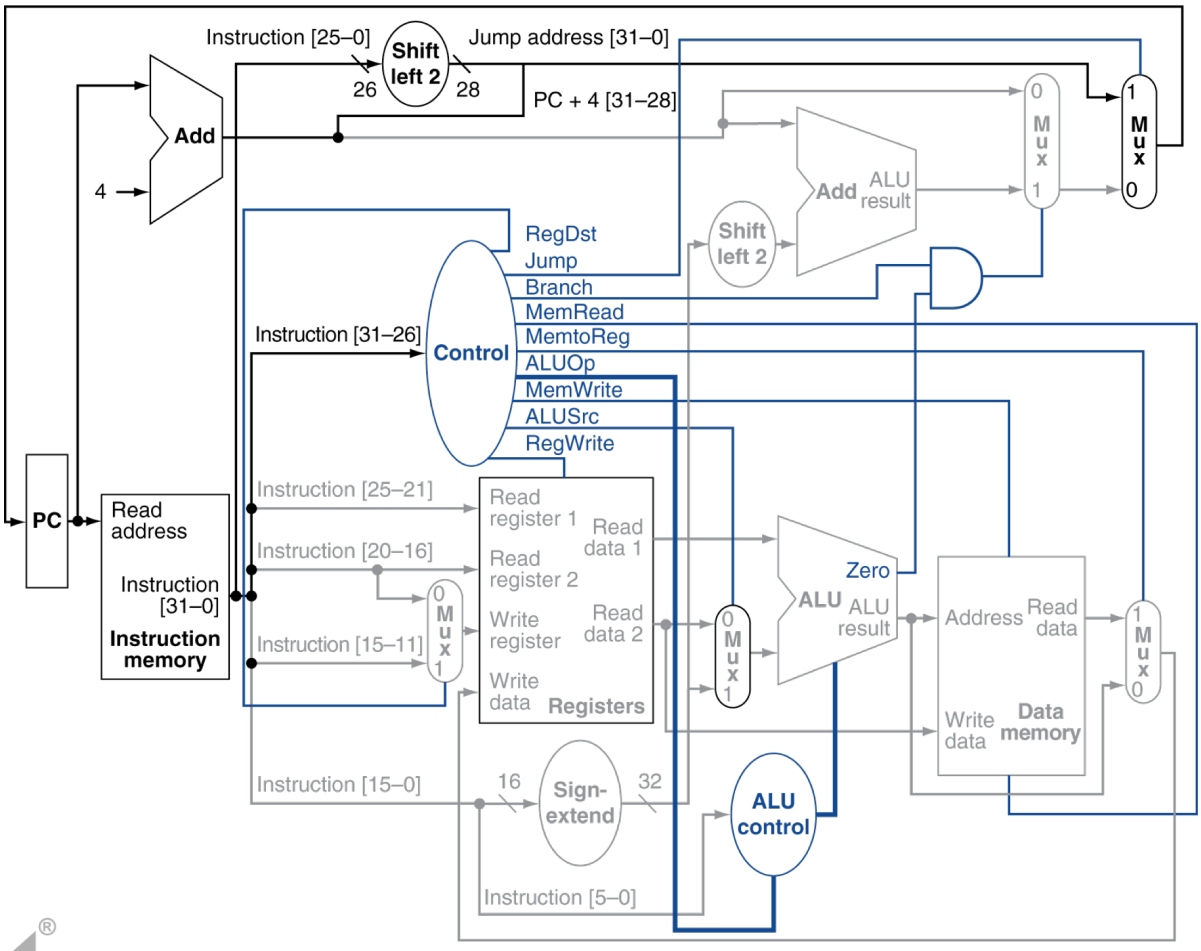
Branch Instructions



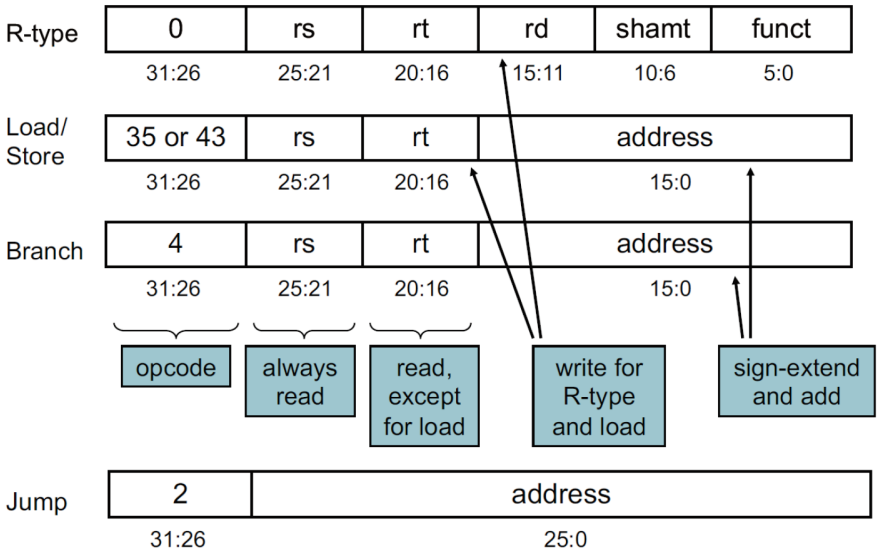
Branch-on-Equal Instruction



Datapath With Jump Supported



How Operands are Used?

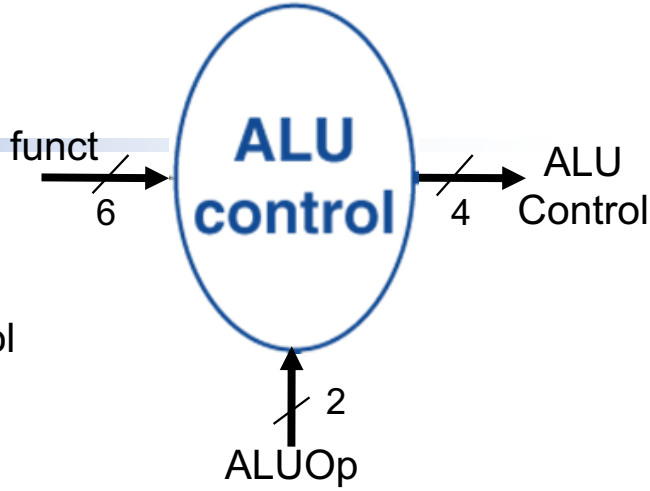


Control Signals

- All control signals derived from 6-bit Opcode of an instruction
- ALUSrc: Calculating reg data or immediate number?
- RegDst: Writing to rt or rd?
- ALUOp: see next page, funct is only used here
- MemWrite: Writing to data memory?
- MemRead: Read from data memory?
- Mem2Reg: Write back calculation result or data memory?
- Branch: Branch or not?
- RegWrite: Writing to reg file?

ALU Control Unit

- Assume 2-bit ALUOp generated by CPU controller
  - Combinational logic derives ALU control
  - With inputs ALUOp and funct – 8 bits



opcode	Operation	ALUOp	funct	ALU Control	ALU function
lw	load word	00	XXXXXX	0010	add
sw	store word				
beq	branch equal	01	XXXXXX	0110	subtract
R-type	add	10	100000	0010	add
	subtract		100010	0110	subtract
	AND		100100	0000	AND
	OR		100101	0001	OR
	set-on-less-than		101010	0111	set-on-less-than

Control Signals

Inst.	ALUSrc	RegDst	ALUOp	MemWrite	MemRead	Mem2Reg	Branch	RegWrite
add	0	1	10	0	0	0	0	1
addi	1	0	??	0	0	0	0	1
lw	1	0	00	0	1	1	0	1
sw	1	X	00	1	0	X	0	0
beq	0	X	01	0	0	X	1	0

Clock Period

Longest delay determines clock period

- Critical path: the path having longest delay
  - load instruction

    Instruction memory → register file → ALU → data memory → register file (plus MUXes)

- Same clock time for different instructions

Some Tips

- Please pay attention to details!
  - Try to avoid calculation mistakes
  - Keep in mind the correct order of the instruction fields
  - …
- Review the lecture notes!
- Good luck!