

# Lab 2 preview

Hung-Wei Tseng

# Announcement

- Lab 1 due tomorrow
  - Submit it through the “lab submission section” in TritonED
- Form your group from now!
  - We need 2 or 3 persons to make a processor
  - 4 or 5 are too more, we are not making burgers

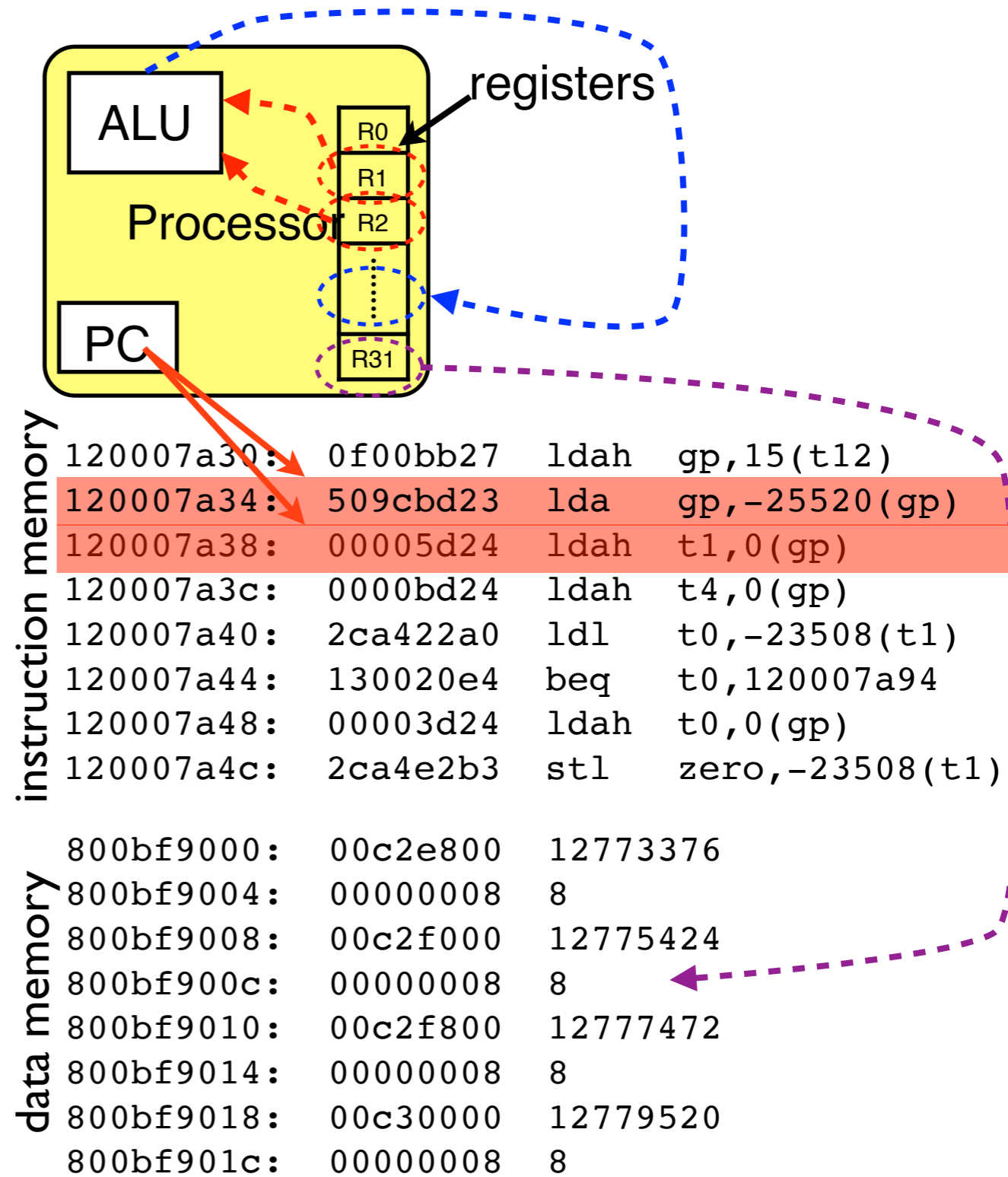


# In Lab 2...

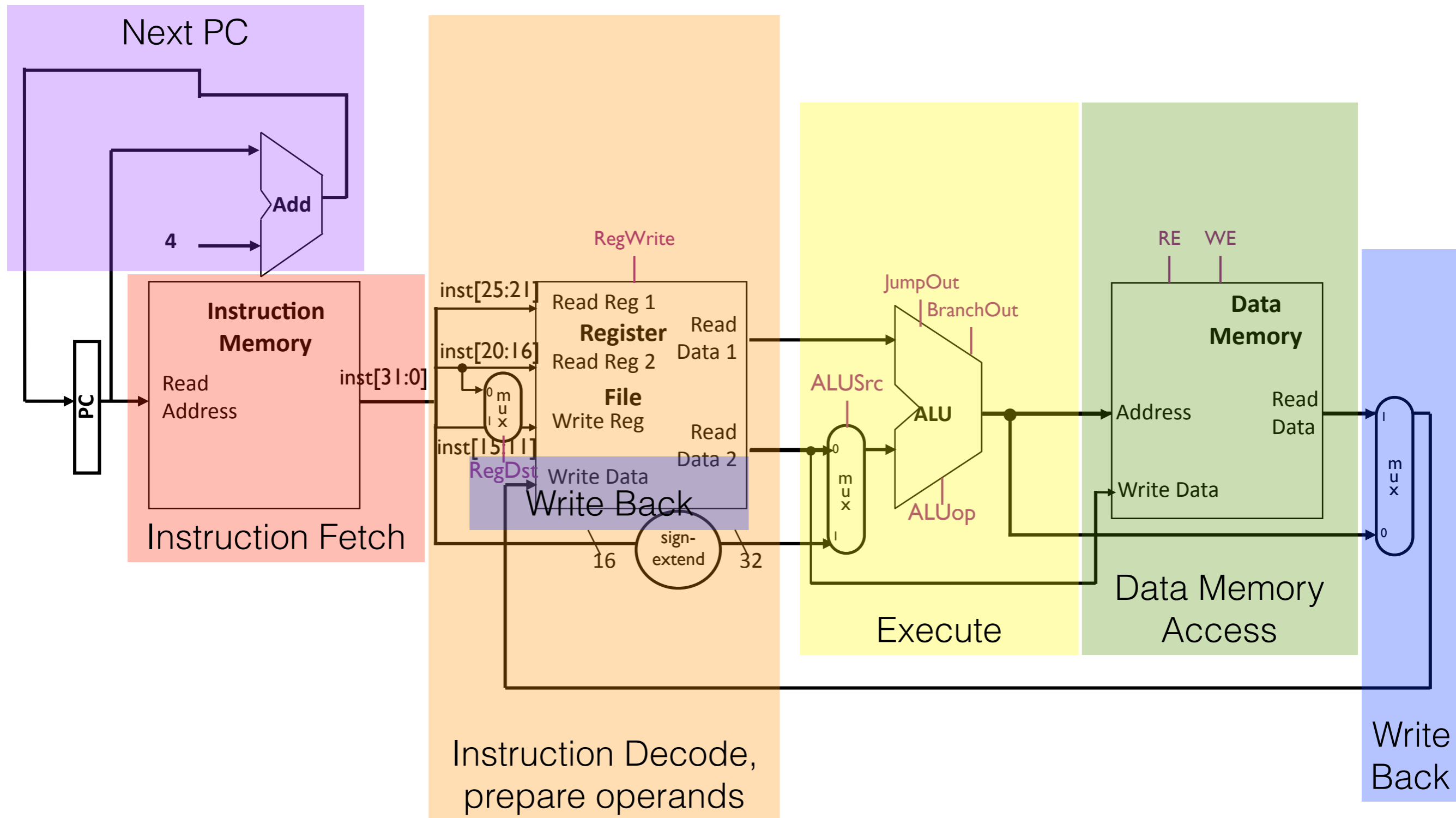
- You will be implementing datapath elements for a single cycle processor
  - We provide ALU, data memory, instruction
  - You need to implement the rest
- Figure out the schematic of your processor first
  - What are the elements required for a single cycle processor
  - How to test each component
  - How to connect them together

# Basic steps of execution

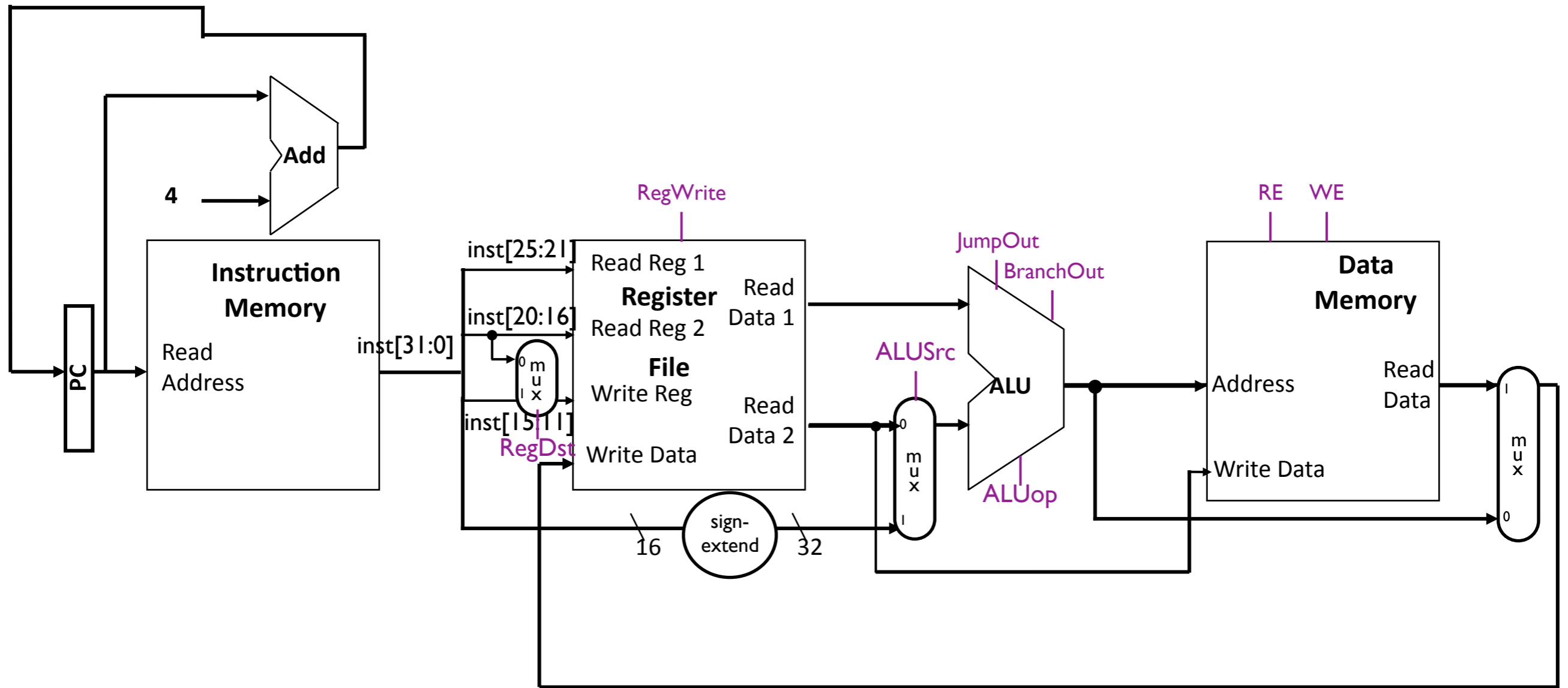
- Instruction fetch: where?  
**instruction memory**
- Decode:
  - What's the instruction?
  - Where are the operands?  
**registers**
- Execute **ALUs**
- Memory access
  - Where is my data?  
**data memory**
- Write back
  - Where to put the result  
**registers**
- Determine the next PC



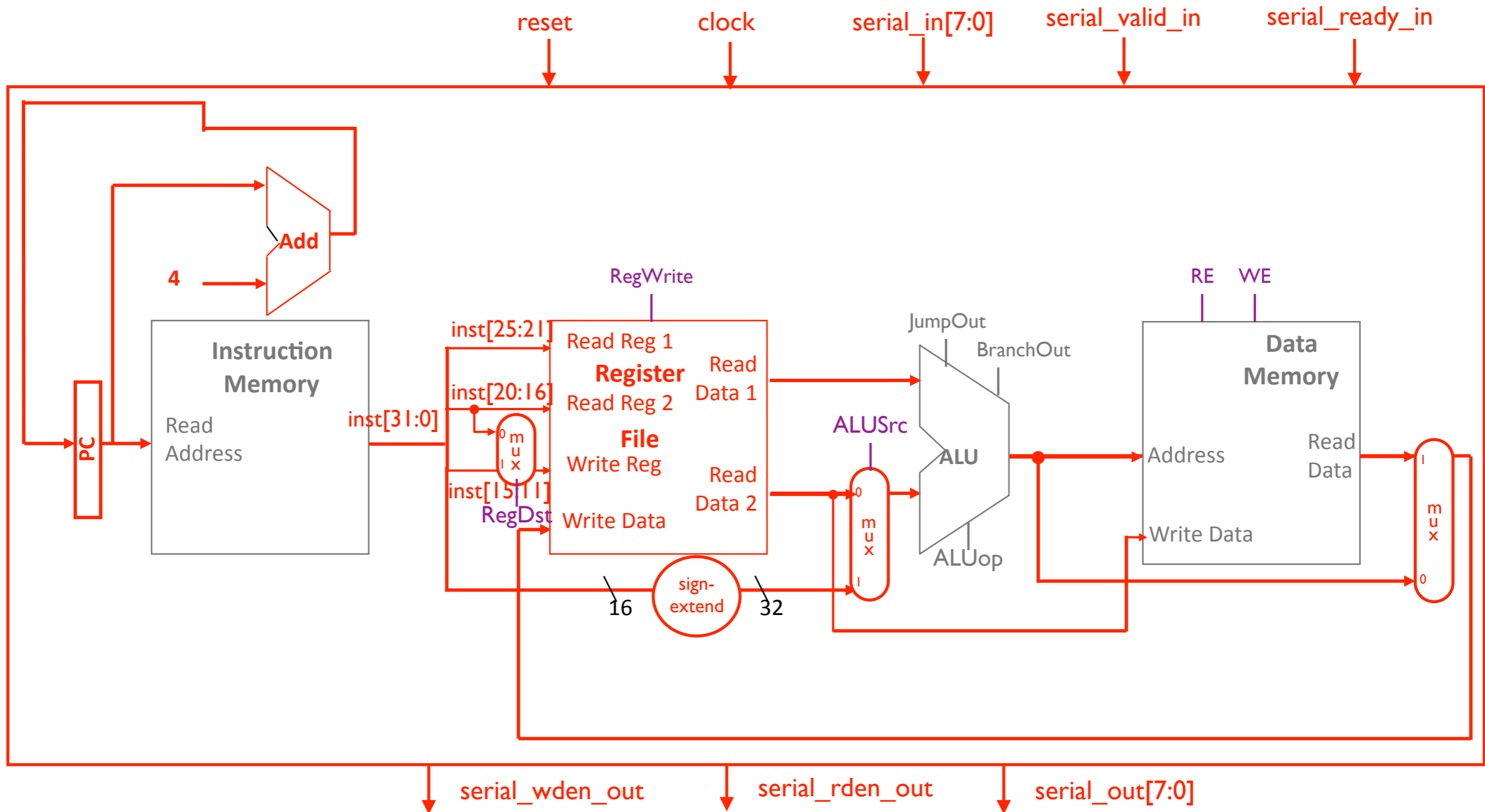
# Lab 2: single-cycle datapath



# Lab 2: single-cycle datapath

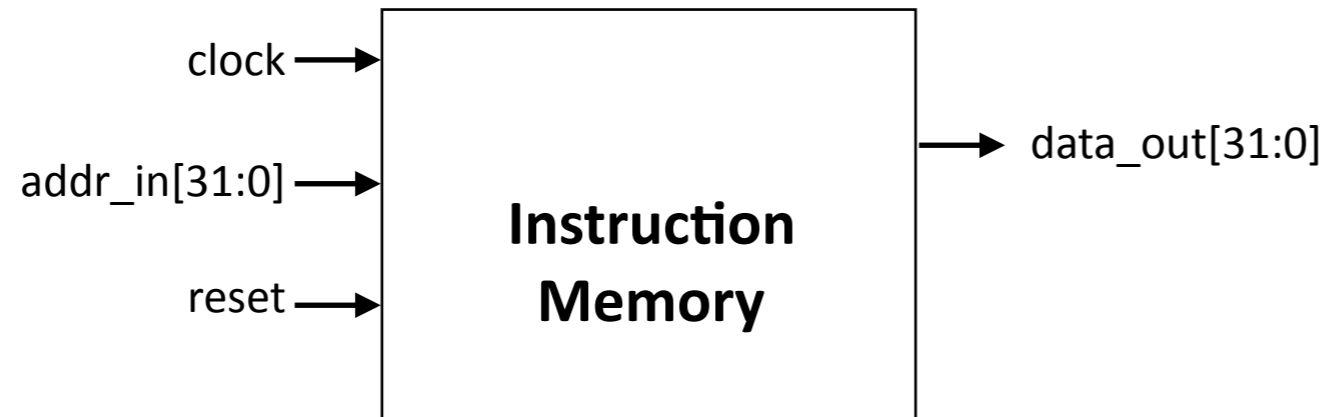


# You have to implement the processor



You have to implement the **red** parts!

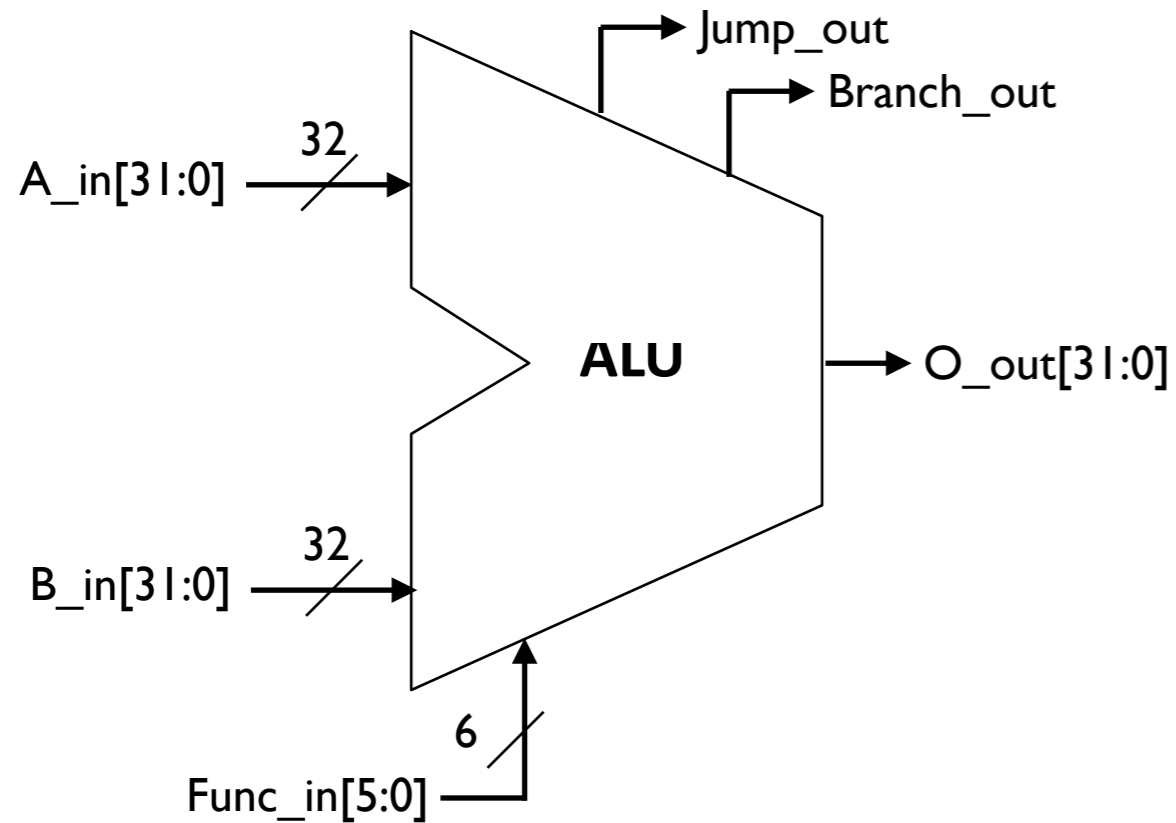
# Interface of instruction memory



```
module inst_rom(  
    input clock,  
    input reset,  
  
    input [31:0] addr_in, //Connect to PC_next  
    output [31:0] data_out //Fetched instruction  
);  
parameter INIT_PROGRAM="c:/myfiles/blank.memh";  
endmodule
```

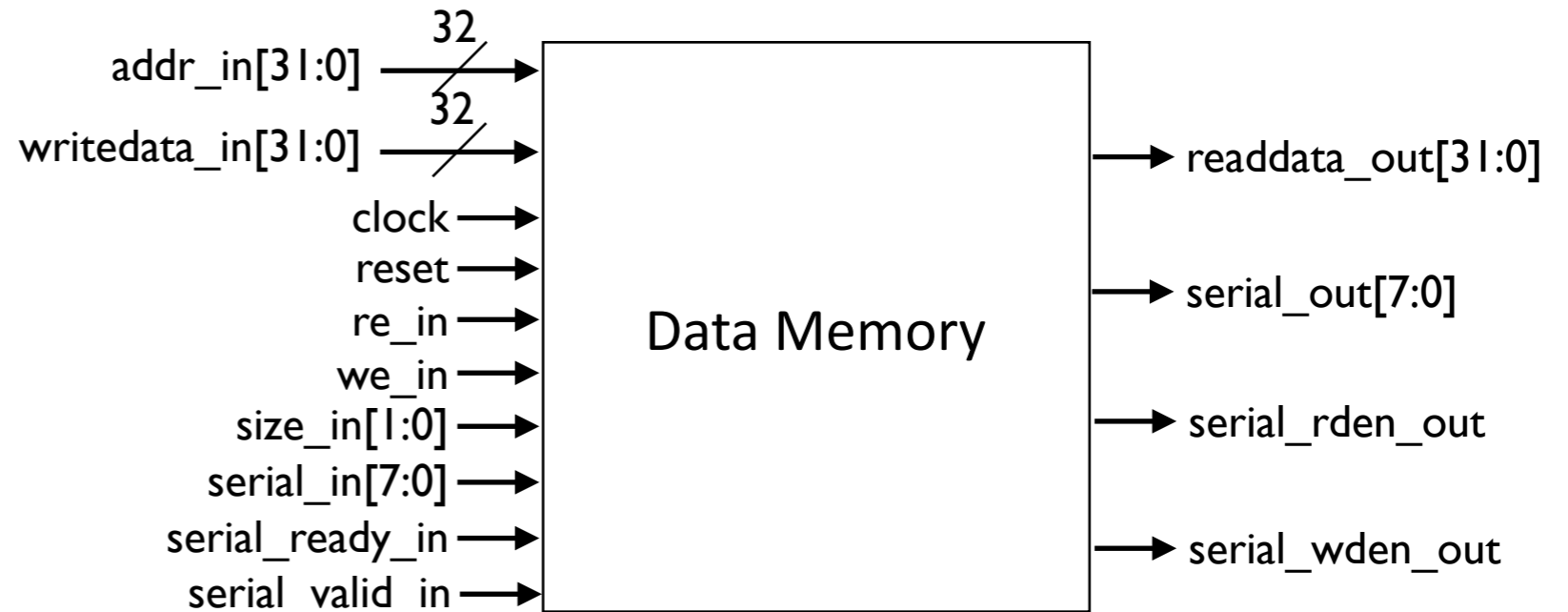


# Interface of ALU



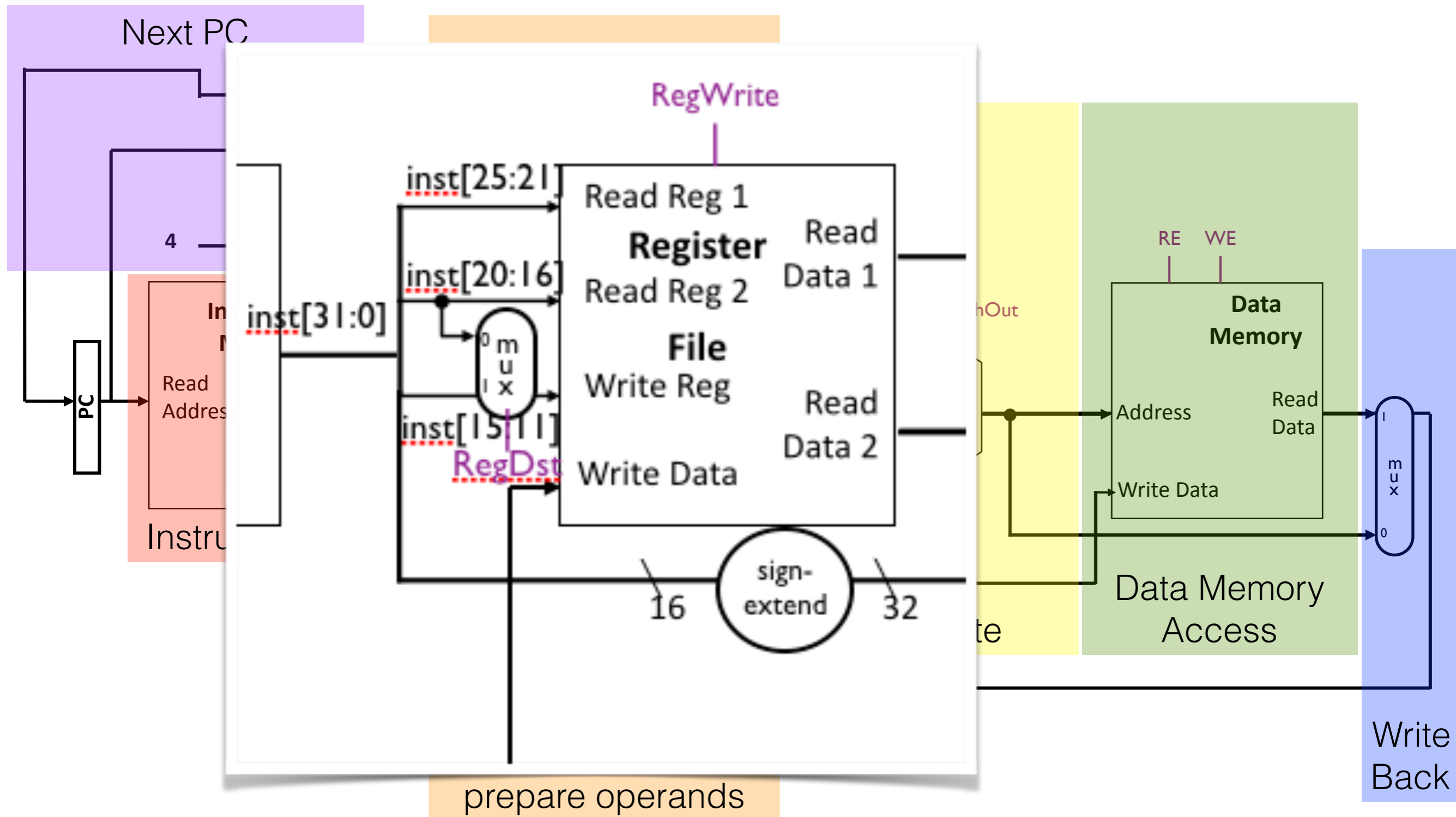
```
module alu(  
    input [5:0] Func_in,  
    input [31:0] A_in,  
    input [31:0] B_in,  
    output [31:0] O_out,  
    output Branch_out,  
    output Jump_out  
);
```

# Interface of data memory

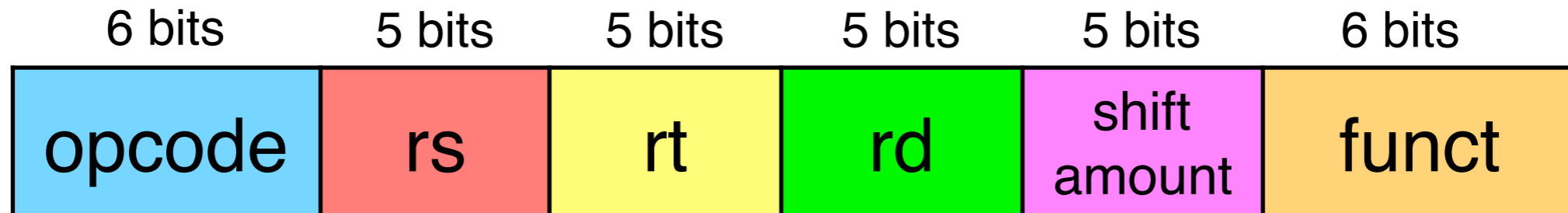


```
module data_memory(  
    input clock,  
    input reset,  
    input [31:0] addr_in, //Read/Write address  
    input [31:0] writedata_in, //Data to write to memory  
    input re_in, //Read Enable - set high when reading from memory  
    input we_in, //Write Enable - set high when writing to memory  
    output [31:0] readdata_out, //Data output for reads from memory  
    input [1:0] size_in, //Not used yet - hardwire to 2'b11  
    input [7:0] serial_in,  
    input serial_ready_in,  
    input serial_valid_in,  
    output [7:0] serial_out,  
    output serial_rden_out,  
    output serial_wren_out  
);  
parameter INIT_PROGRAM0="c:/myfiles/blank.memh";  
parameter INIT_PROGRAM1="c:/myfiles/blank.memh";  
parameter INIT_PROGRAM2="c:/myfiles/blank.memh";  
parameter INIT_PROGRAM3="c:/myfiles/blank.memh";
```

# Lab 2: single-cycle datapath

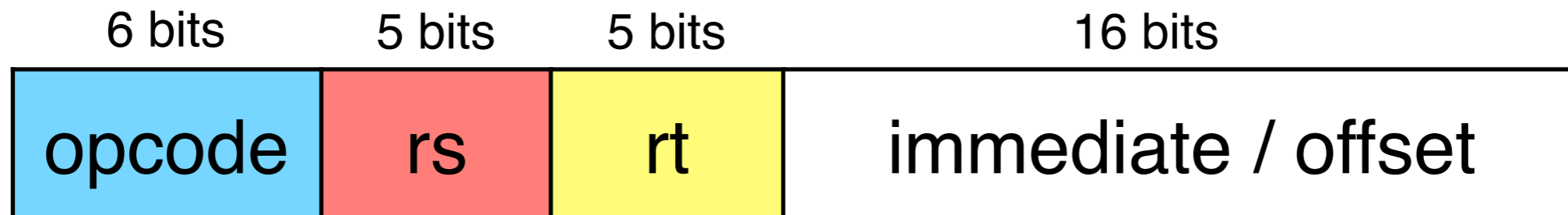


# R-type



- `op $rd, $rs, $rt`
  - 3 regs.: `add, addu, and, nor, or, sltu, sub, subu`
  - 2 regs.: `sll, srl`
  - 1 reg.: `jr`
- 1 arithmetic operation, 1 I-memory access
- Example:
  - `add $v0, $a1, $a2: R[2] = R[5] + R[6]`  
opcode = 0x0, funct = 0x20
  - `sll $t0, $t1, 8: R[8] = R[9] << 8`  
opcode = 0x0, shamt = 0x8, funct = 0x0

# I-type



- op \$rt, \$rs, **immediate**
  - addi, addiu, andi, beq, bne, ori, slti, sltiu
- op \$rt, **offset(\$rs)**
  - lw, lbu, lhu, ll, lui, sw, sb, sc, sh

only two  
addressing  
modes

- 1 arithmetic op, 1 I-memory and 1 D-memory access

- Example:

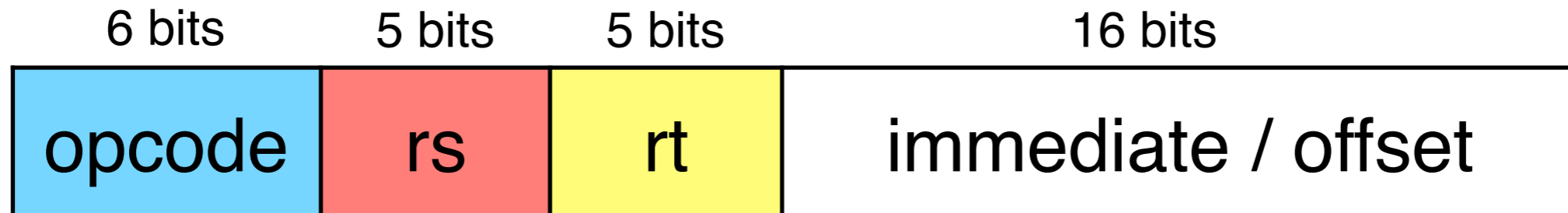
- `lw $s0, 4($s2):`  
 $R[16] = \text{mem}[R[18]+4]$

~~lw \$s0, \$s2(\$s1)~~

add \$s2, \$s2, \$s1

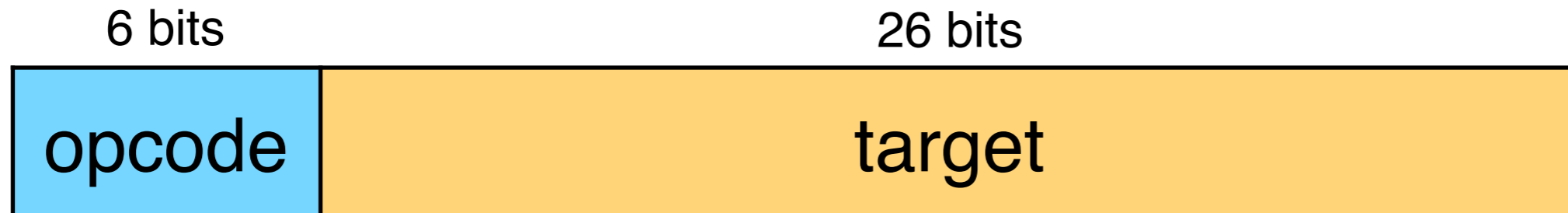
lw \$s0, 0(\$s2)

# I-type (cont.)



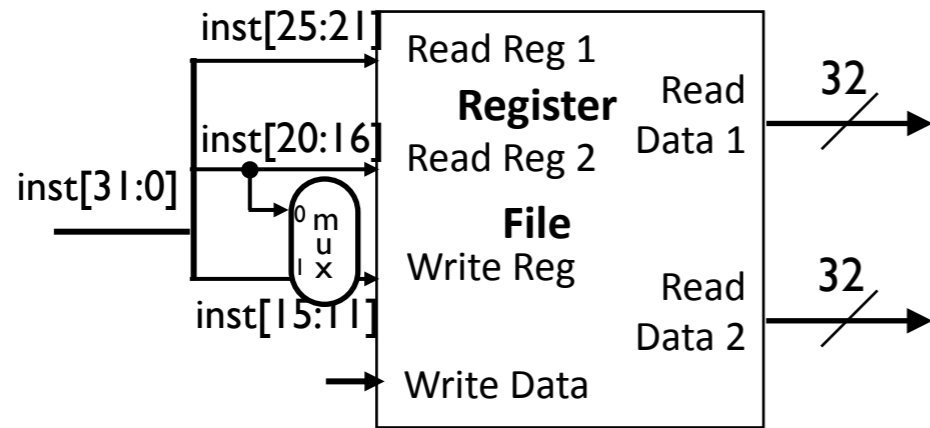
- op \$rt, \$rs, immediate
  - addi, addiu, andi, beq, bne, ori, slti, sltiu
- op \$rt, offset(\$rs)
  - lw, lbu, lhu, ll, lui, sw, sb, sc, sh
- 1 arithmetic op, 1 I-memory and 1 D-memory access
- Example:
  - `beq $t0, $t1, -40`  
`if (R[8] == R[9]) PC = PC + 4 + 4 * (-40)`

# J-type



- op immediate
  - j, jal
- 1 instruction memory access, 1 arithmetic op
- Example:
  - `jal quicksort`  
`R[31] = PC + 4`  
`PC = quicksort`

# Your register file

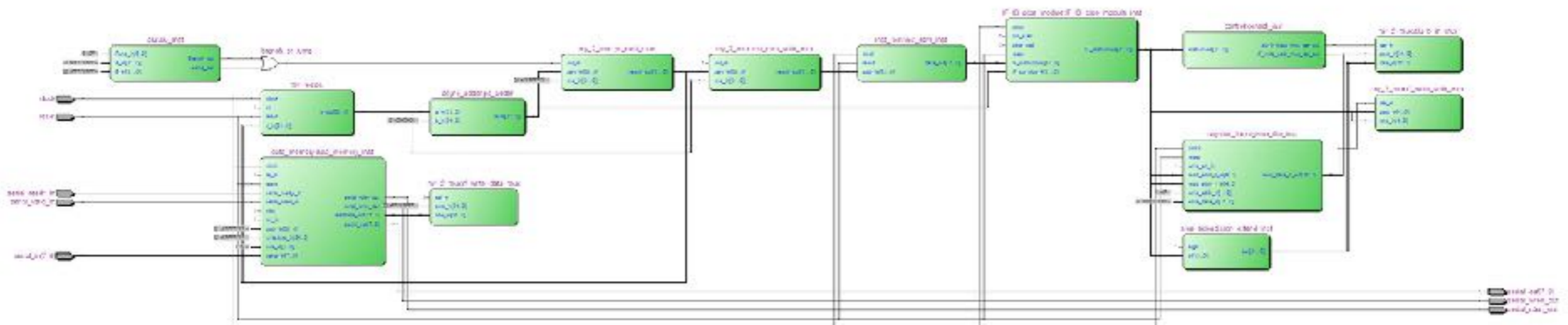


- Should contain at least the above inputs/outputs
  - Does every instruction write back data?
- Is it clocked?
- Support reset
- Deal with \$zero



# When you interview with us

- Show us it's working
- Show us your processor.v
- Show us your register\_file.v
  - Demonstrate it works: you need to design a testbench for it.
- Show us your schematic
  - You may output it through the RTL viewer under analysis tool in Quartus II



# Announcement

- No lecture next Monday — I will be at lab 6:00p-7:30p
- Lab #1 due next Monday

# Q & A