Computer Organization and Design
EGRE 426

Sherif Abdelwahed

Lab 4 Guide

Department of Electrical and Computer Engineering
Virginia Commonwealth University

Lab 4

Goal: Design a 16-bit single cycle RISC processor and simulate it. 2 students per team

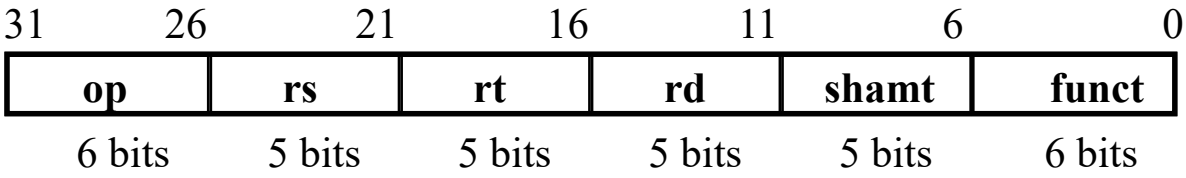
Main Steps:

1. Design Instruction Sets
2. Clocking Methodology and Datapath Components
3. Assemble Datapath
4. Analyze the implementation of each instruction
5. Design Control Unit
6. Write a simulator in VHDL (default) or C/C++, Java, Python (if I gave you a permission) and run the test program.

Step 1 (Proposal)

ADD and SUB

- `add rd, rs, rt`
- `sub rd, rs, rt`

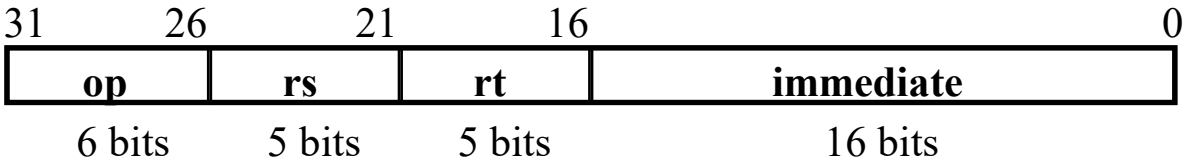


OR immediate:

- `ori rt, rs, im16`

Set less than immediate:

- `slti  rt, rs, im16`



LOAD and STORE word

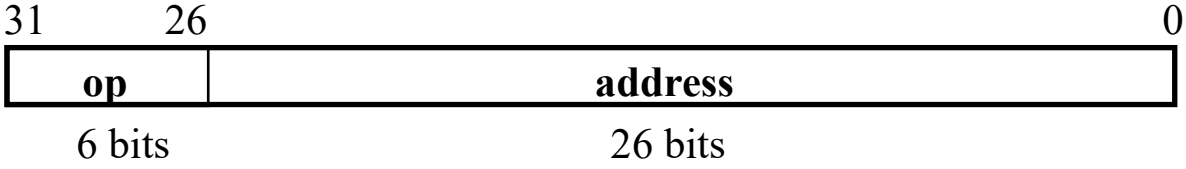
- `lw rt, im16(rs)`
- `sw rt, im16(rs)`

Branch

- `beq rs, rt, im16`

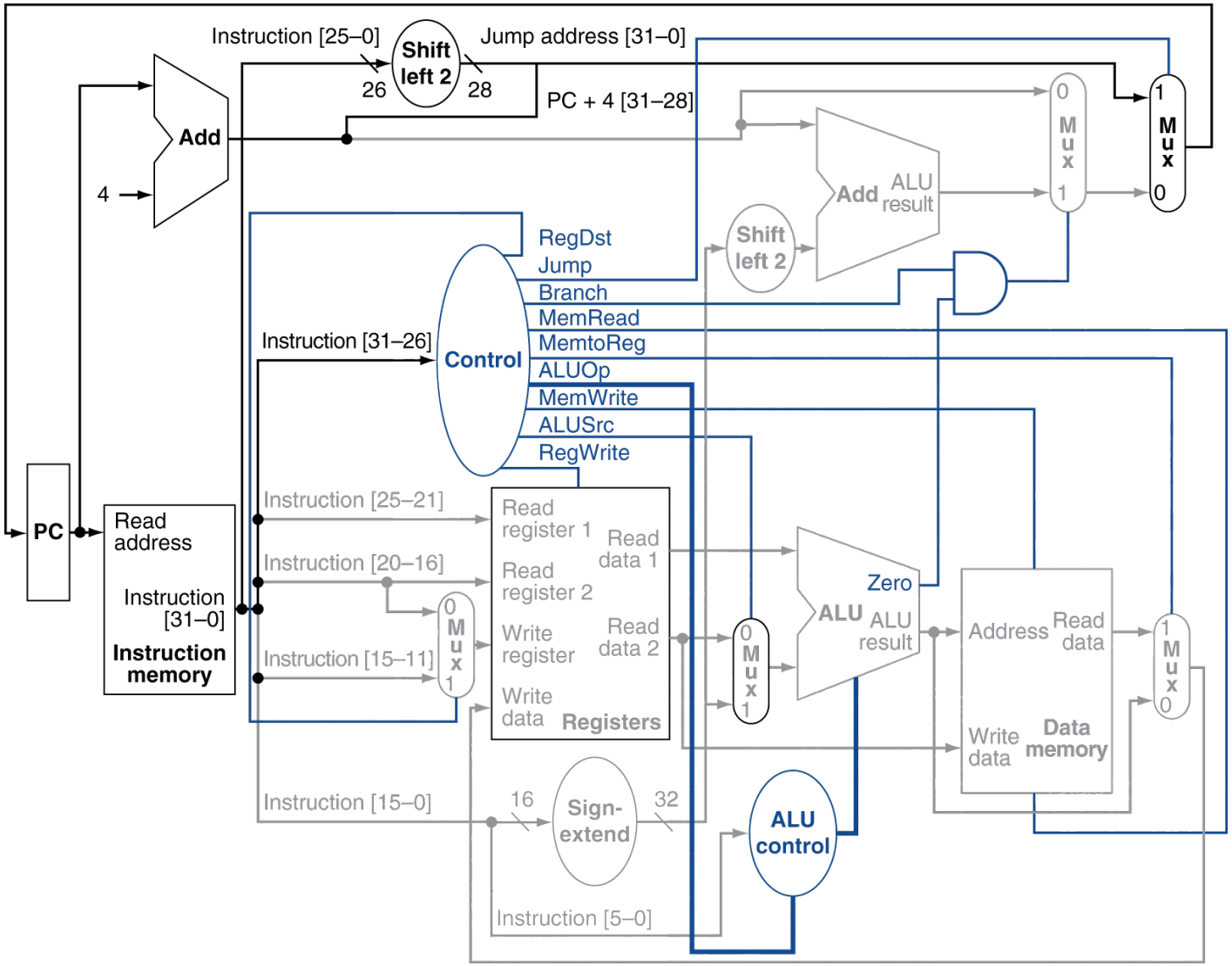
Jump

- `J add`$_{26}$



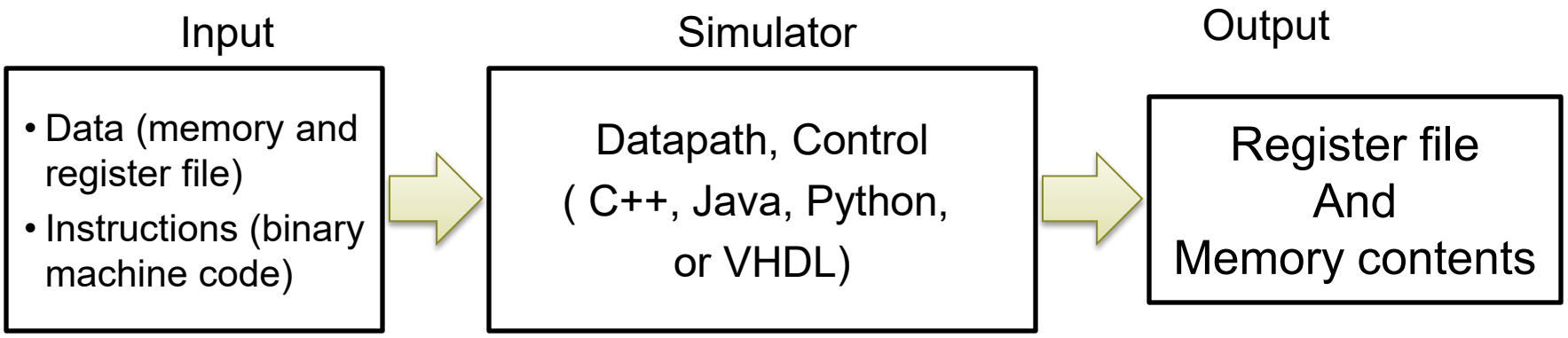
Minimum 15 instructions. Bonus if you add new instructions

Step 2-6


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]

Step 6.1

Write a Simulator and test a Pseudocode with your assembly



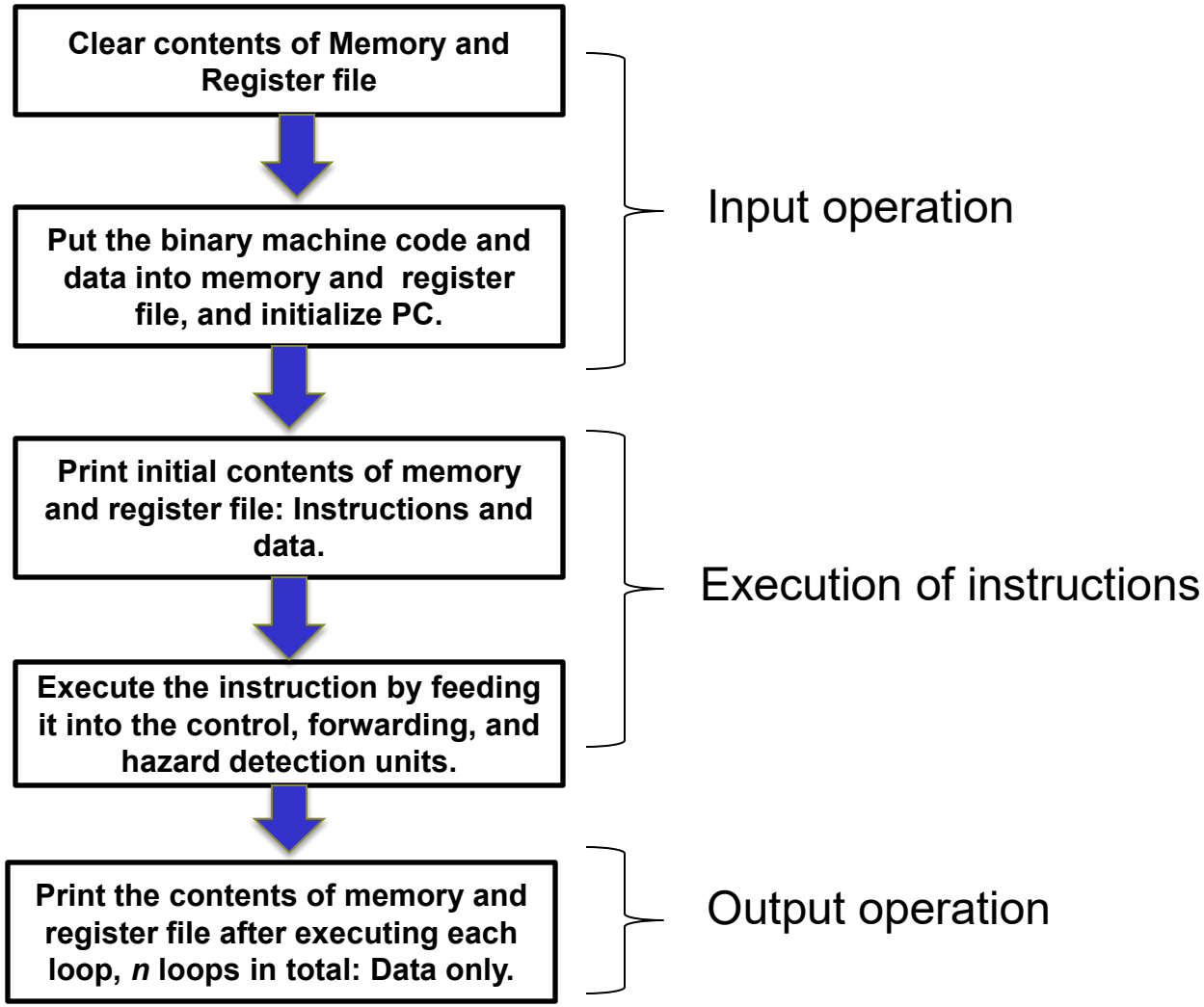
The input to the simulator:

- Translate the pseudo code (the test program) into your assembly language and binary machine code (By hand. Do not implement a compiler);
- Put the binary machine code into your memory (instructions); and
- Put the provided data into your memory and register file.

The output of the simulator:

- Initial contents: Instructions and data of the memory and register file before program running.
- Contents after each loop, n loops in total: data of the memory and register file (not instructions).

Step 6.2


Clear contents of Memory and Register file
Put the binary machine code and data into memory and register file, and initialize PC.
Input operation
Print initial contents of memory and register file: Instructions and data.
Execute the instruction by feeding it into the control, forwarding, and hazard detection units.
Execution of instructions
Print the contents of memory and register file after executing each loop, n loops in total: Data only.
Output operation

Step 6.3

Check the sample code for the simulator

Code the FSM in or VHDL (default) or C/C++, Java, Python. Sample code (C++) is shown below for the ALU as an example.

```
void ALU() {
   switch(opcode)  {
      case 0x0001: //add
          ALUOut = A + B;
          break;
      case 0x0003: //and
          ALUOut = A & B;
          break;
      case 0x0005: //mult
          ALUOut = high(A,B);
          break;
      case 0x000a://slt
          if (A < B)
              ALUOut = 1;
          else
             ALUOut = 0;
          break;
```

```
case 0x000a://sltu
   if ((unsigned word)A < (unsigned word)B)
       ALUOut = 1;
   else
       ALUOut = 0;
   break;
case 0x000b://addi
   ALUOut = A + sign_extend5to16((IR & 0x001F));
   break;
case 0x000c://subi
   ALUOut = A - sign_extend5to16((IR & 0x001F));
   break;
case 0x0010://sw
   ALUOut = A + sign_extend5to16((IR & 0x001F));
   break;
default:
   break;
   }
}
```