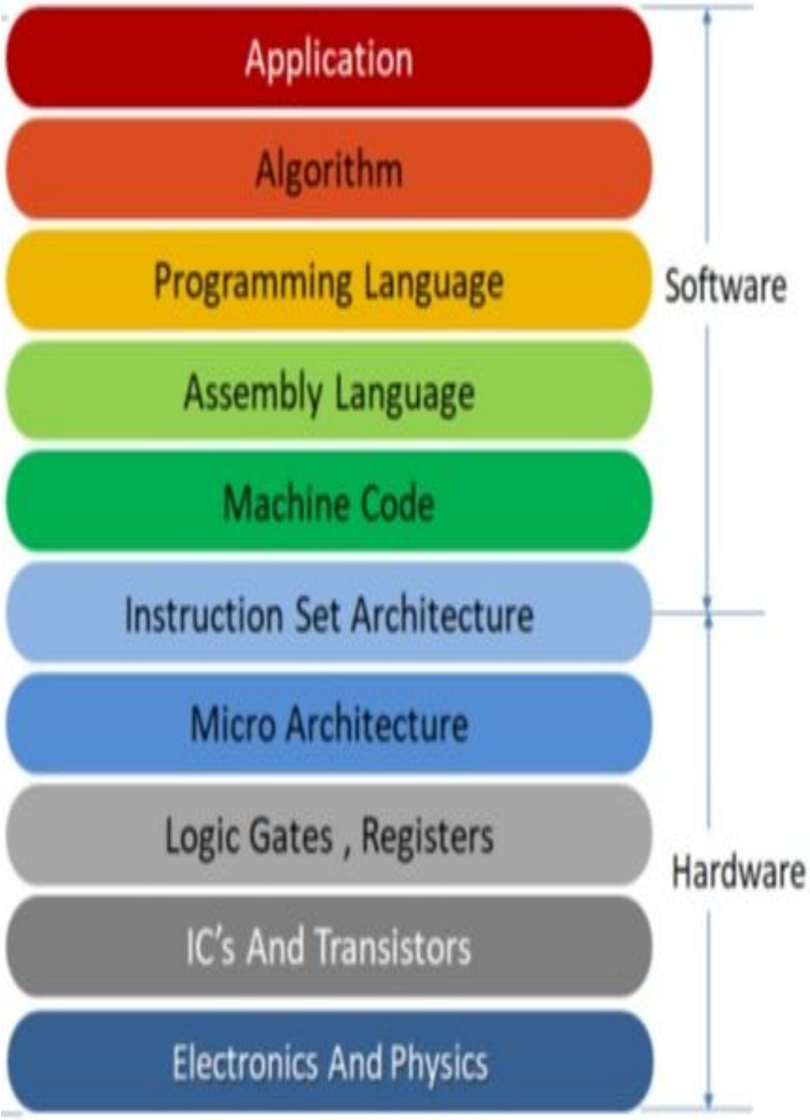
# Instruction Set Architecture

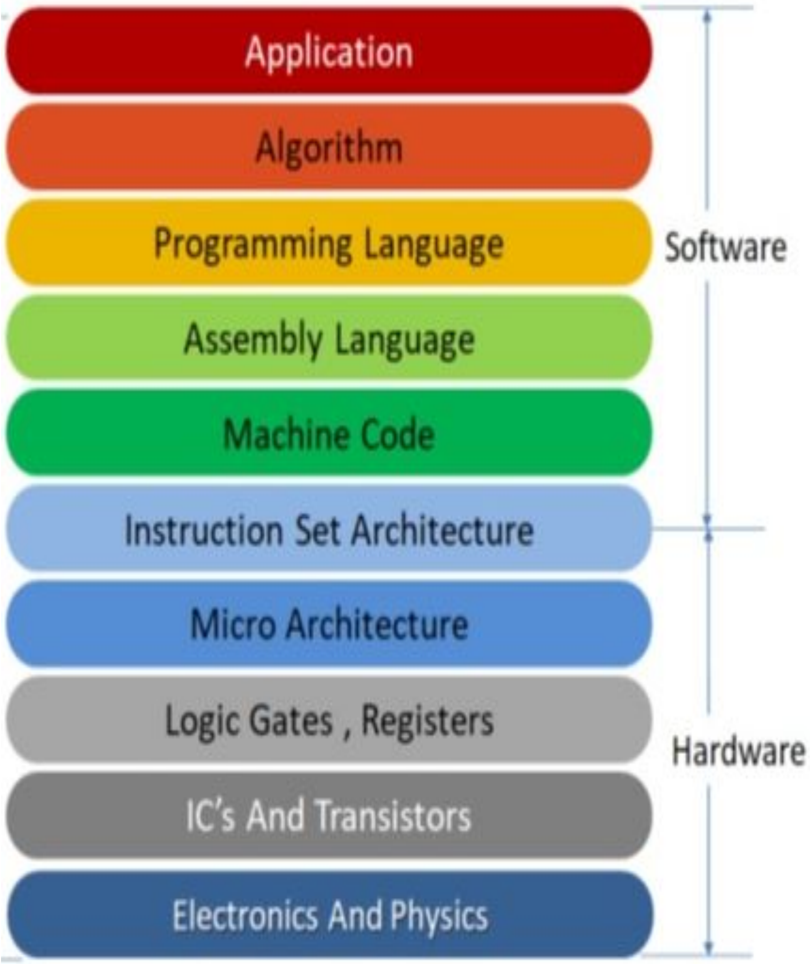
- Instructions: Commands for basic operations
- Types and formats of instructions
- Addressing modes of instructions
- Machine codes of instructions
- Data representation
- Registers and uses
- Memory address

Computer Engineers/Hardware Engineers must understand ISA to design/implement hardware details of a computer: Microarchitecture and it’s design using ICs, logic gates and associated components.

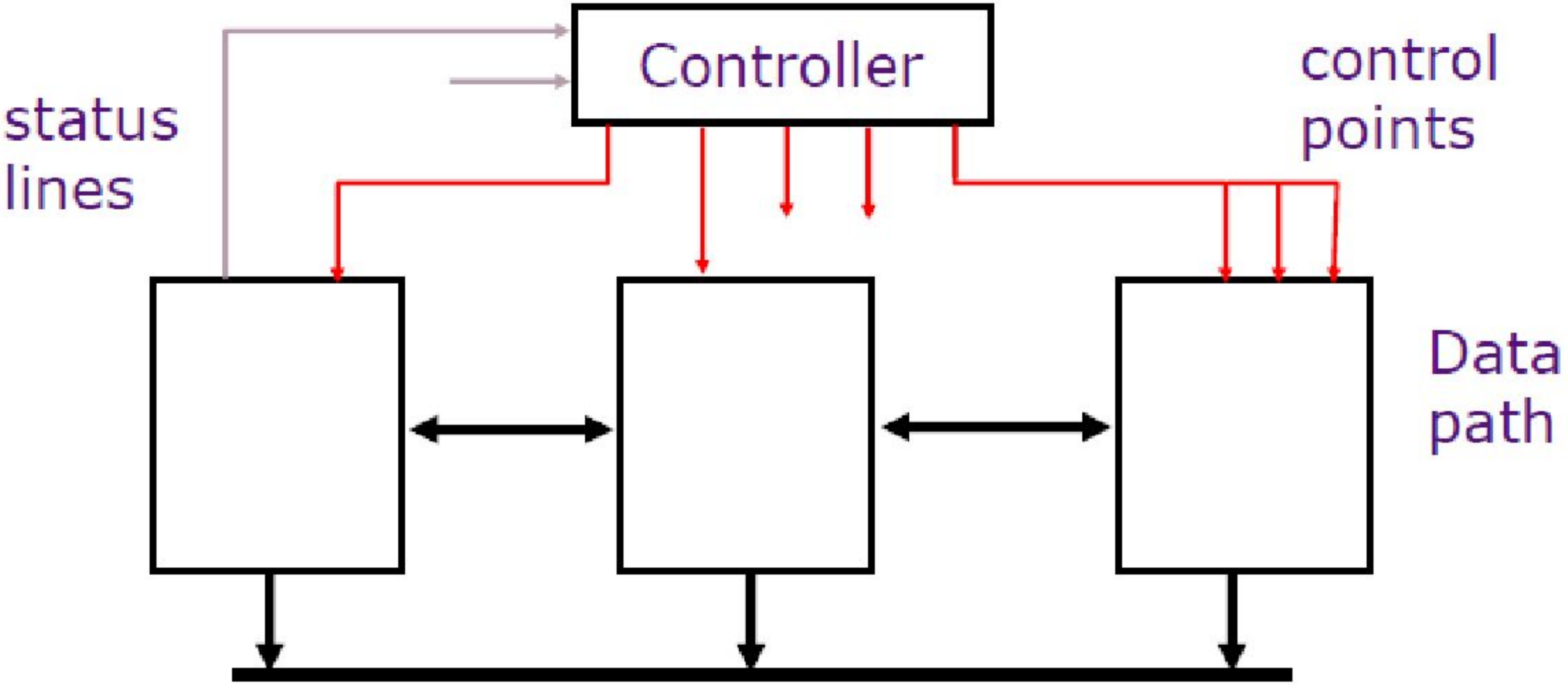


# Micro-architecture

- The level above the digital logic level is the **microarchitecture level**.
- Its job is to implement the ISA (Instruction Set Architecture) level above it,
- The design of the microarchitecture level depends on the ISA being implemented, as well as the cost and performance goals of the computer.
- Many modern ISAs, particularly RISC designs, have simple instructions that can usually be executed in a single clock cycle.
- More complex ISAs may require many cycles to execute a single instruction.
- Executing an instruction may require locating the operands in memory, reading them, and storing results back into memory.



# Microarchitecture: *Implementation of an ISA*

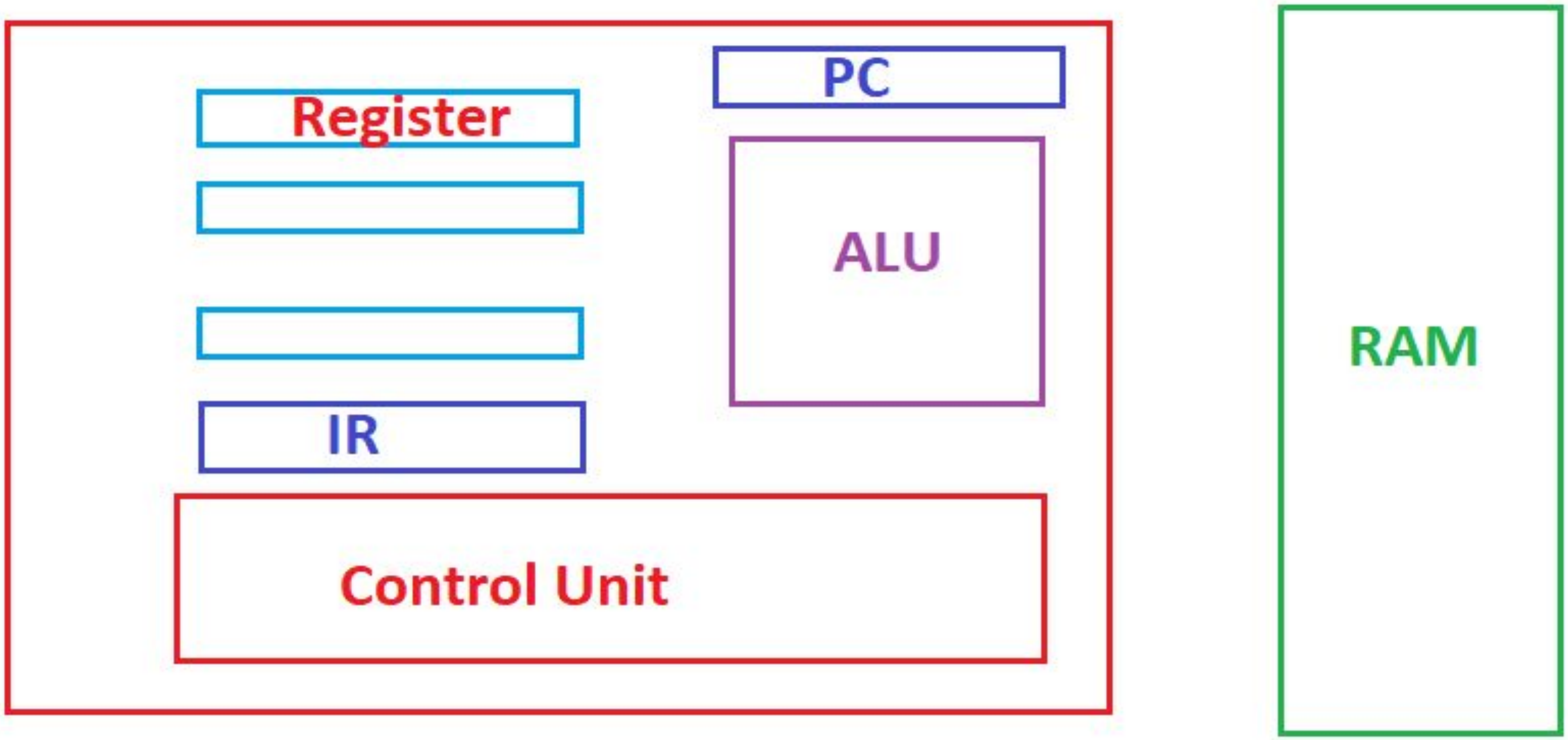


*Structure:* How components are connected.
*Static*

*Behavior:* How data moves between components
*Dynamic*

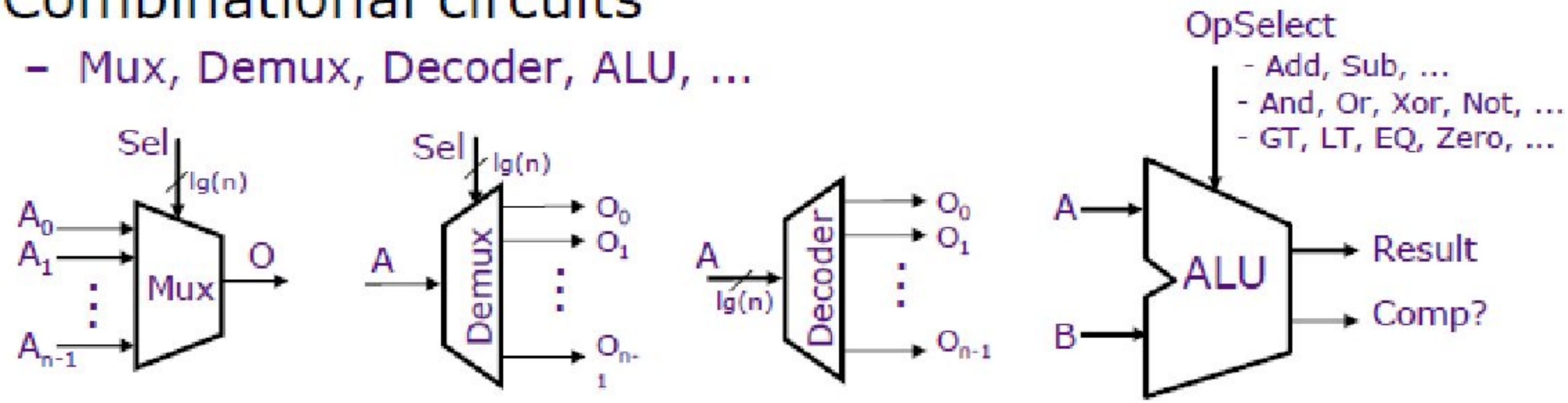
# Microarchitecture

- Design of Registers, ALU, Control Unit
- How to interconnect these and how to control/flow information among these
- How to select particular operation in ALU
- How to select particular registers (read/write)
- How to access memory (read/write)

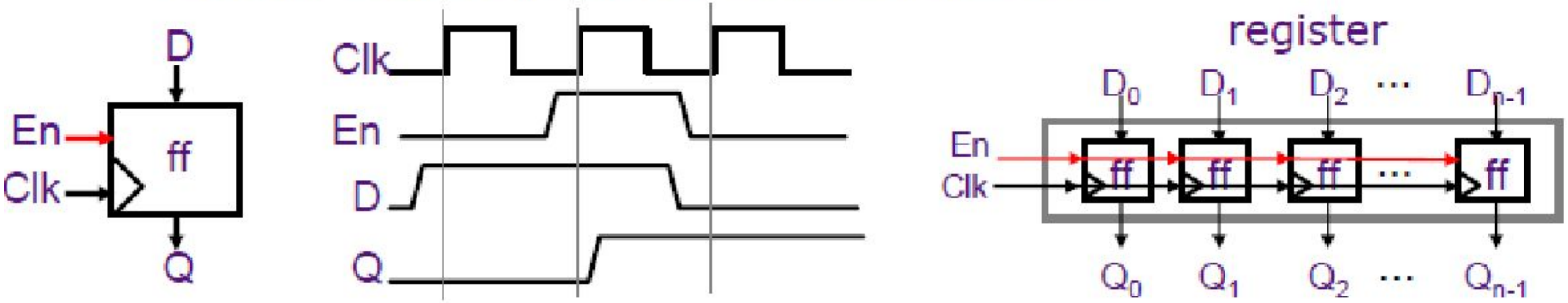


# Hardware Elements

- Combinational circuits
  - Mux, Demux, Decoder, ALU, ...

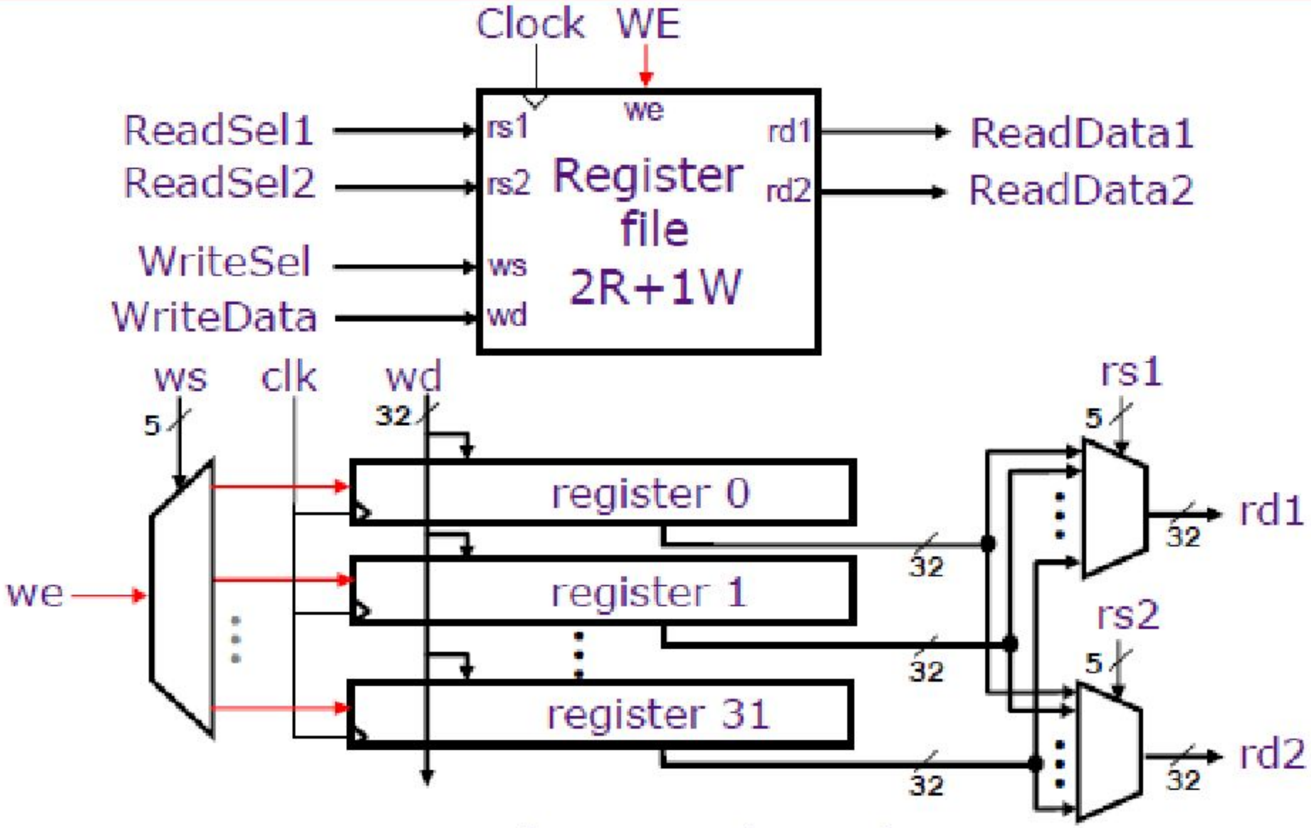


- Synchronous state elements
  - Flipflop, Register, Register file, SRAM, DRAM

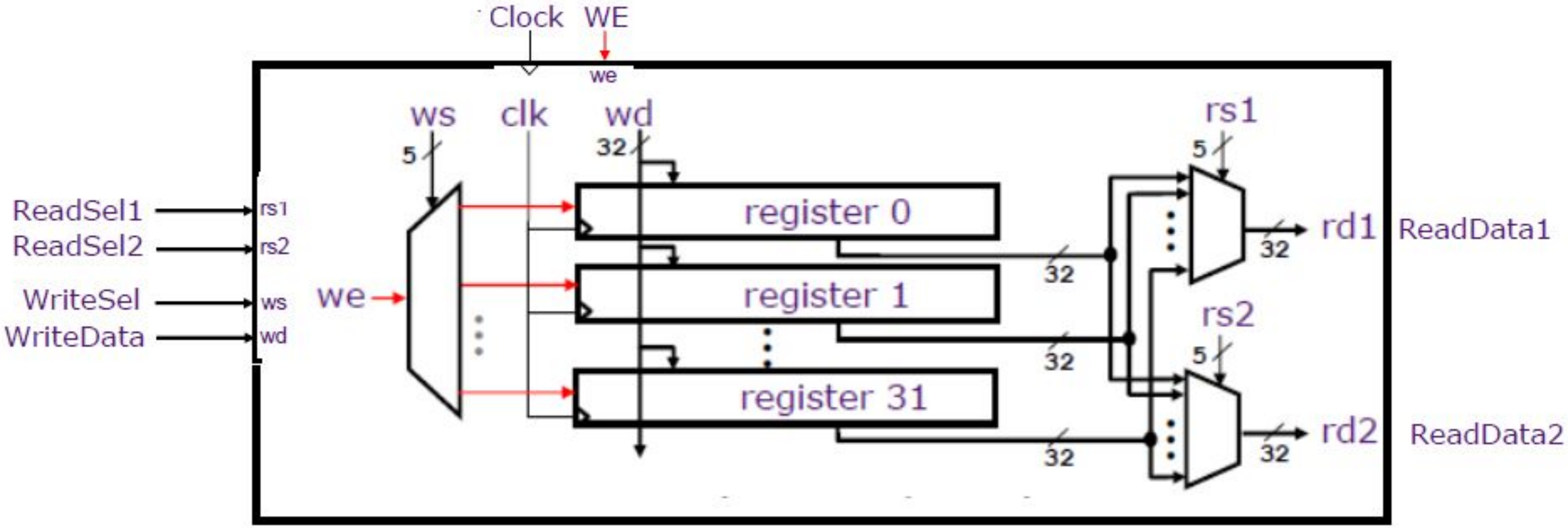


*Edge-triggered: Data is sampled at the rising edge*

# Register Files



- No timing issues in reading a selected register
- Register files with a large number of ports are difficult to design
  - *Intel's Itanium, GPR File has 128 registers with 8 read ports and 4 write ports!!!*


Clock
WE
we
ws
clk
wd
5
32
ReadSel1
ReadSel2
WriteSel
WriteData
rs1
rs2
ws
wd
we
register 0
register 1
register 31
32
32
32
rs1
5
rs2
5
rd1
ReadData1
32
rd2
ReadData2
32

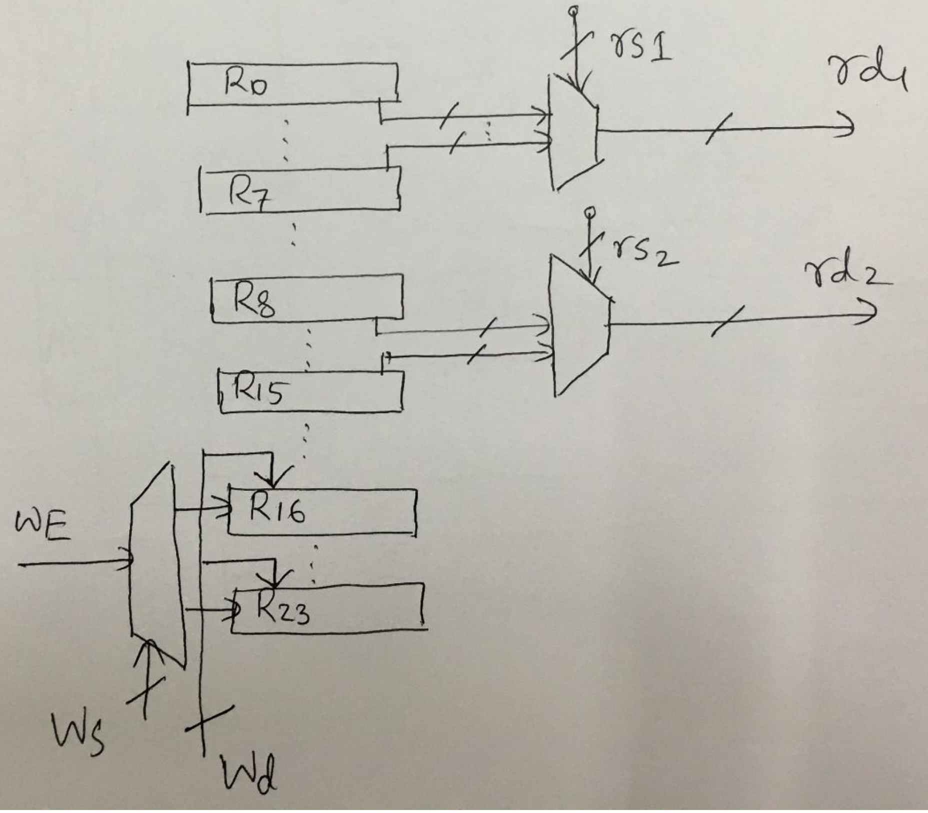
# Design Register Architecture:

2R + 1W

1st operand : $R_0$ – $R_7$

2nd " : $R_8$ – $R_{15}$

Result field : $R_{16}$ – $R_{23}$



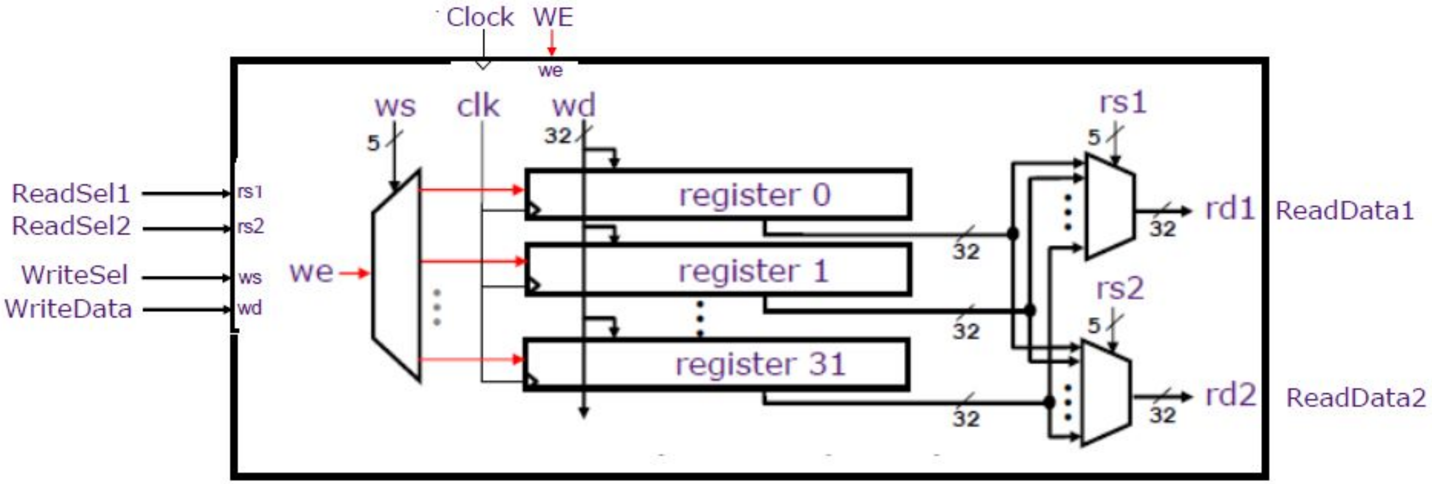
# Instruction Execution

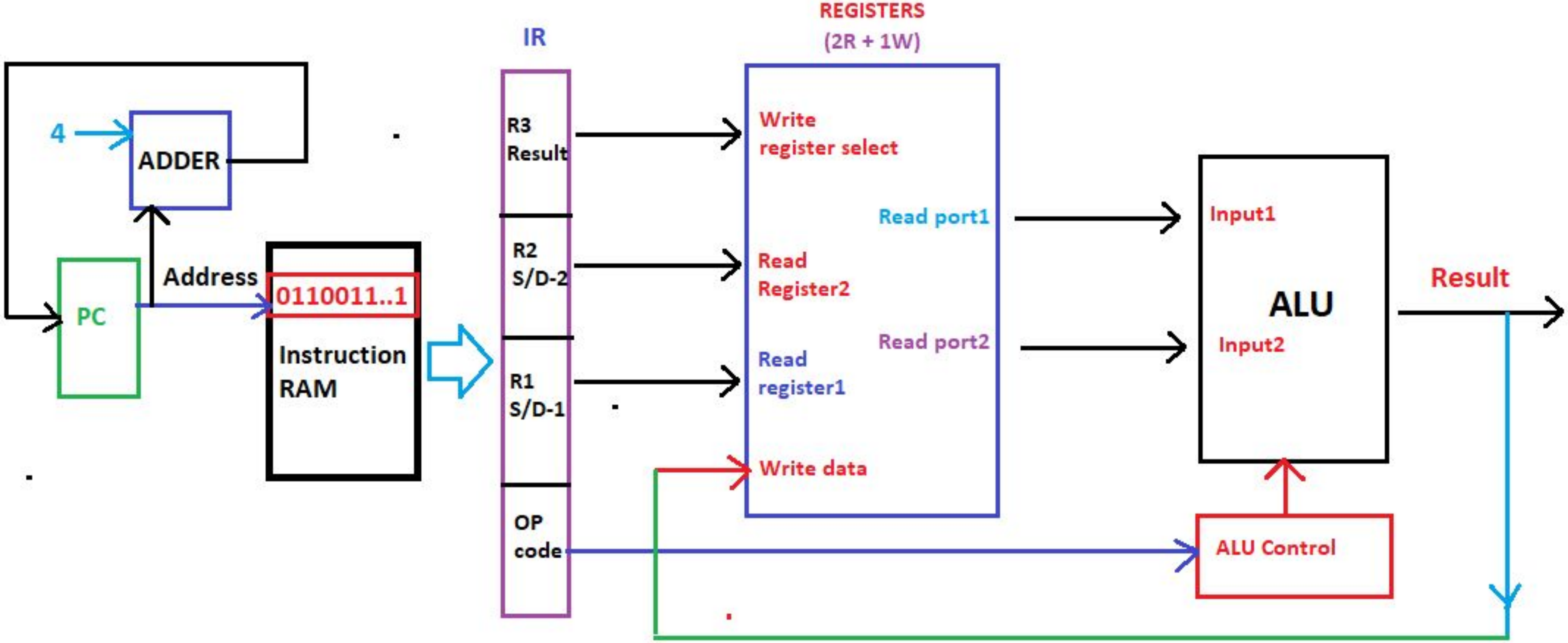
Execution of an instruction involves

1. instruction fetch
2. decode and register fetch
3. ALU operation
4. memory operation (optional)
5. write back

and the computation of the address of the *next instruction*

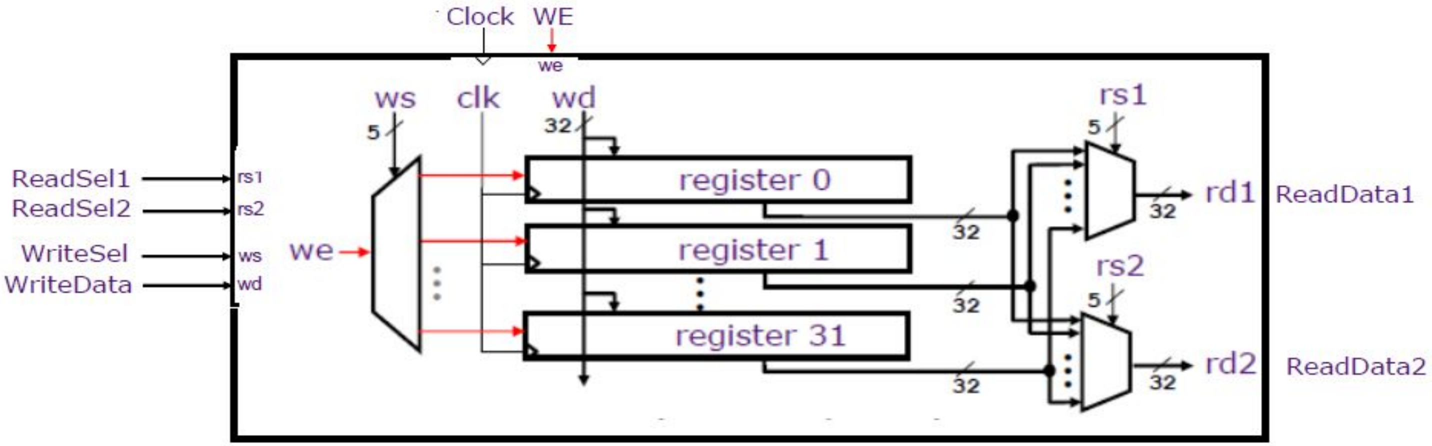
| | | | |
|---|---|---|---|
| Opcode | R1 (data-1) | R2(data-2) | R3(result) |

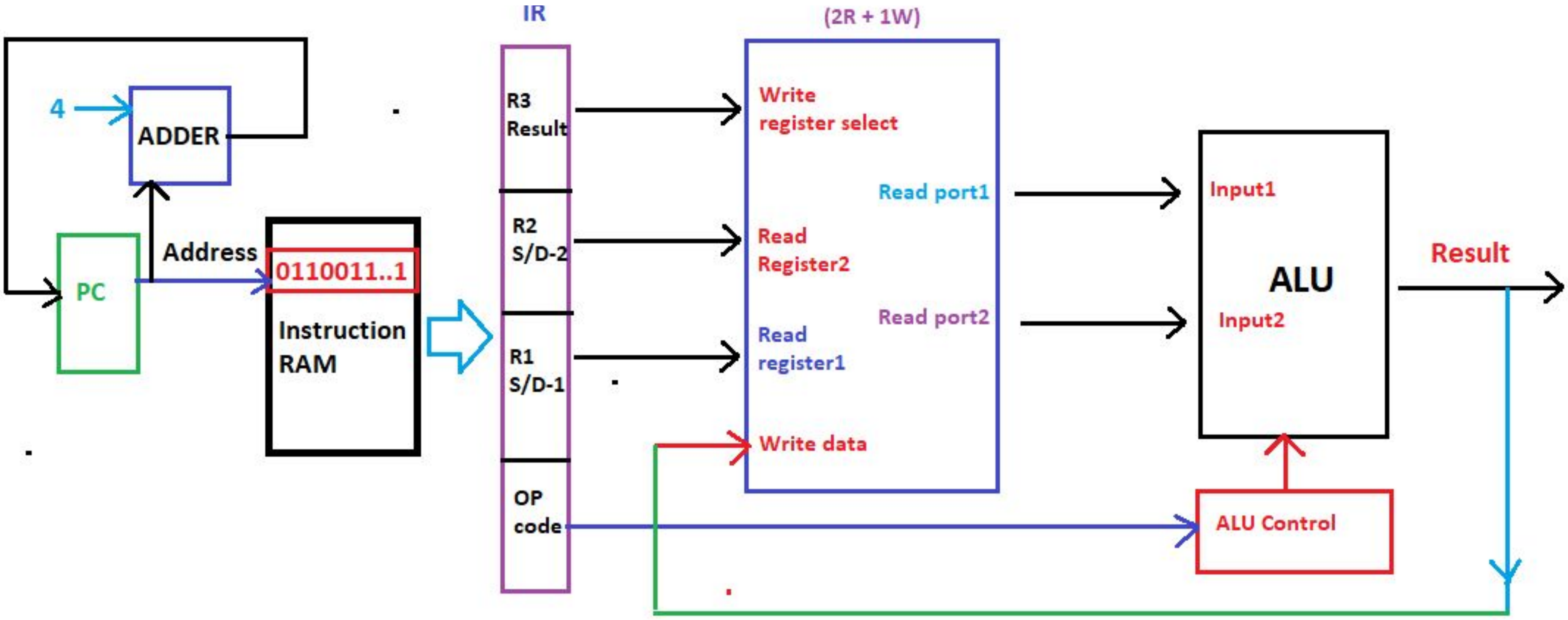




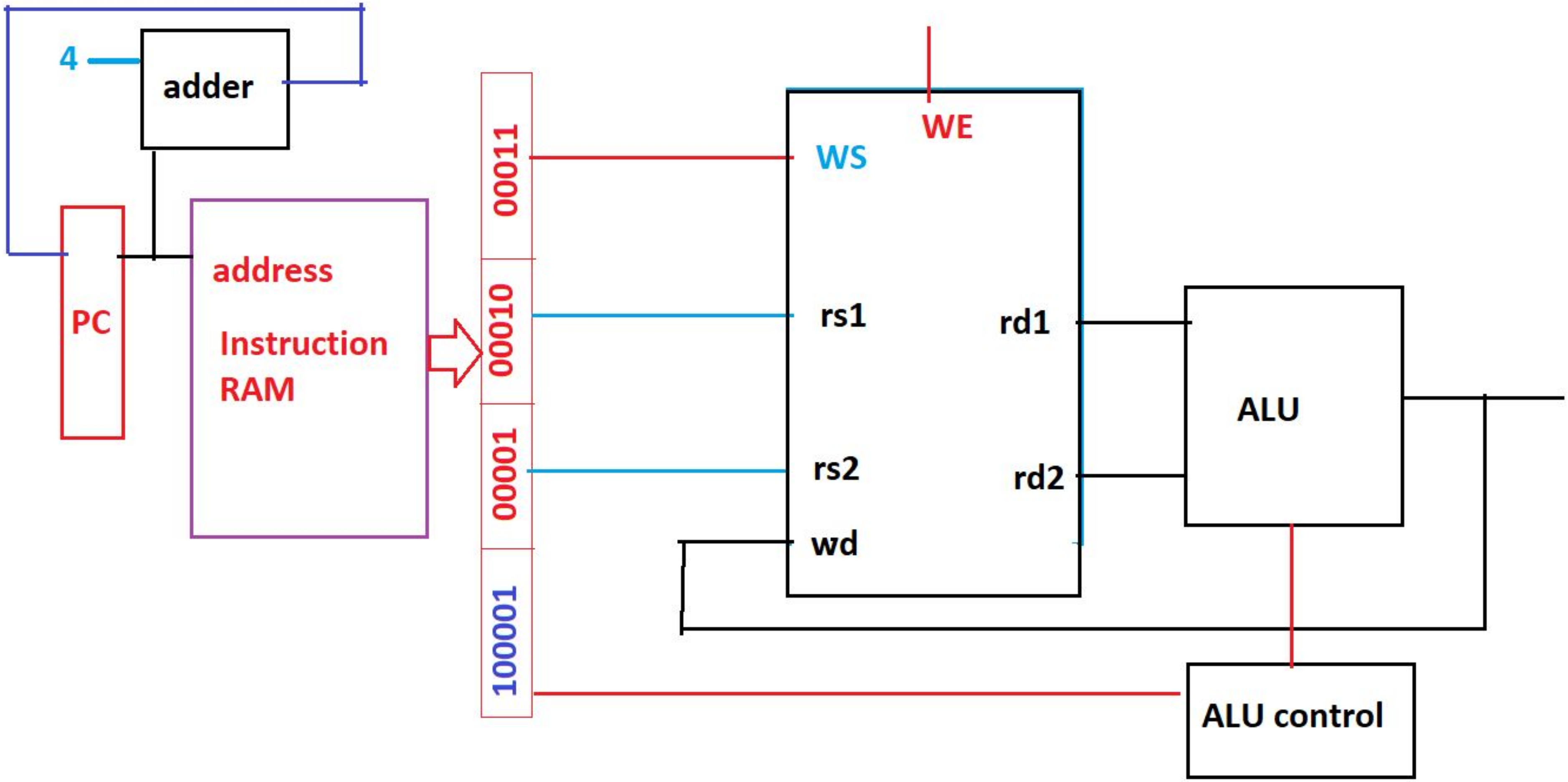
# Register Mode

| | | | |
|---|---|---|---|
| Opcode | R1 (data-1) | R2(data-2) | R3(result) |



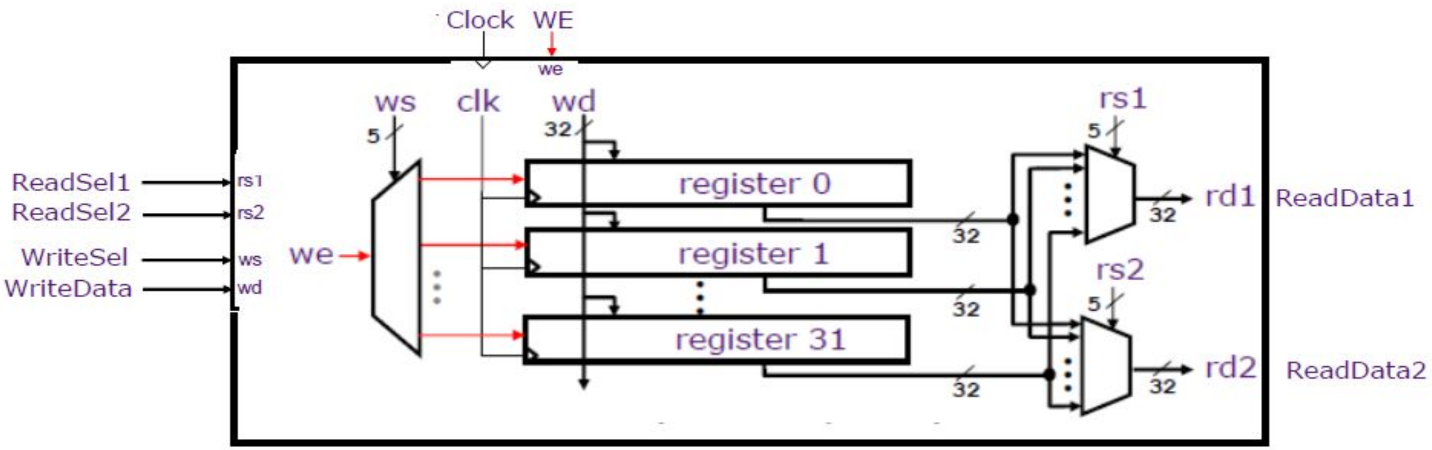


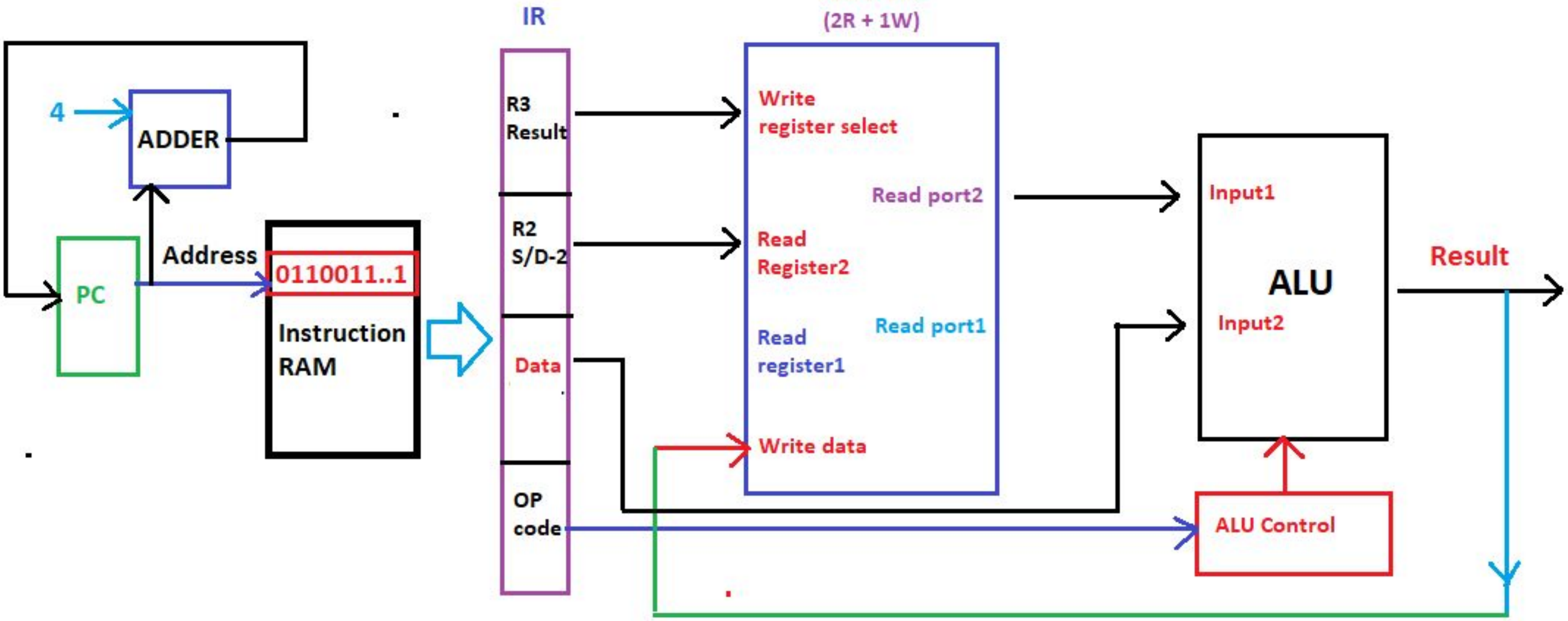
| ADD | R1 | R2 | R3 |
|---|---|---|---|
| 100001 | 00001 | 00010 | 00011 |

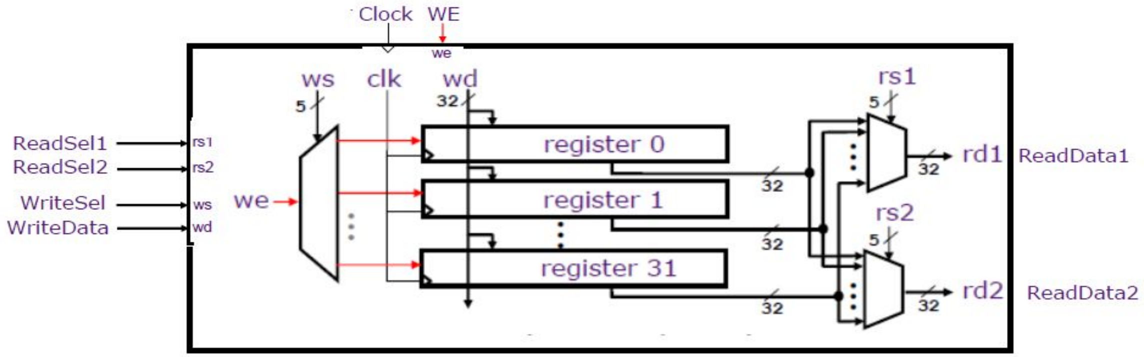


# Immediate Mode

| Opcode | data-1 | R2(data-2) | R3(result) |
|---|---|---|---|



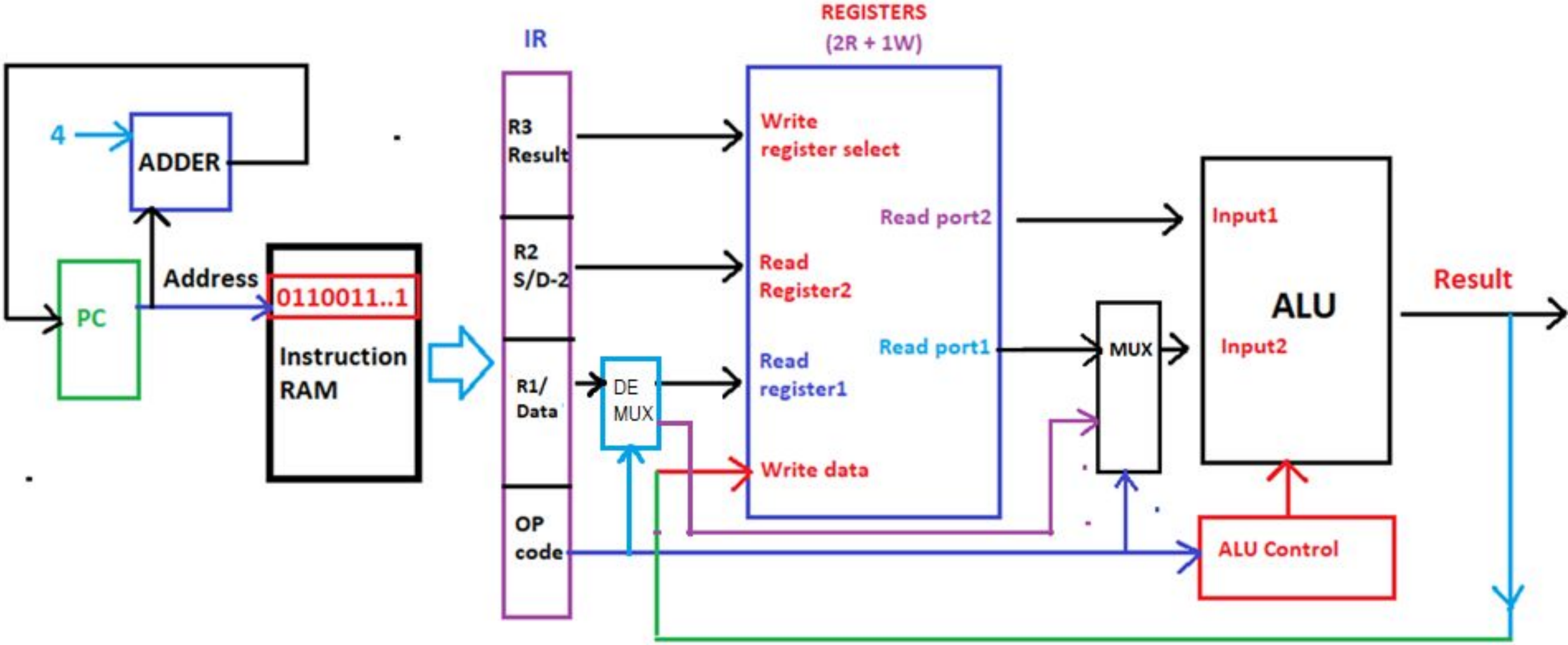




| Opcode | R1 (data-1) | R2(data-2) | R3(result) |
|---|---|---|---|

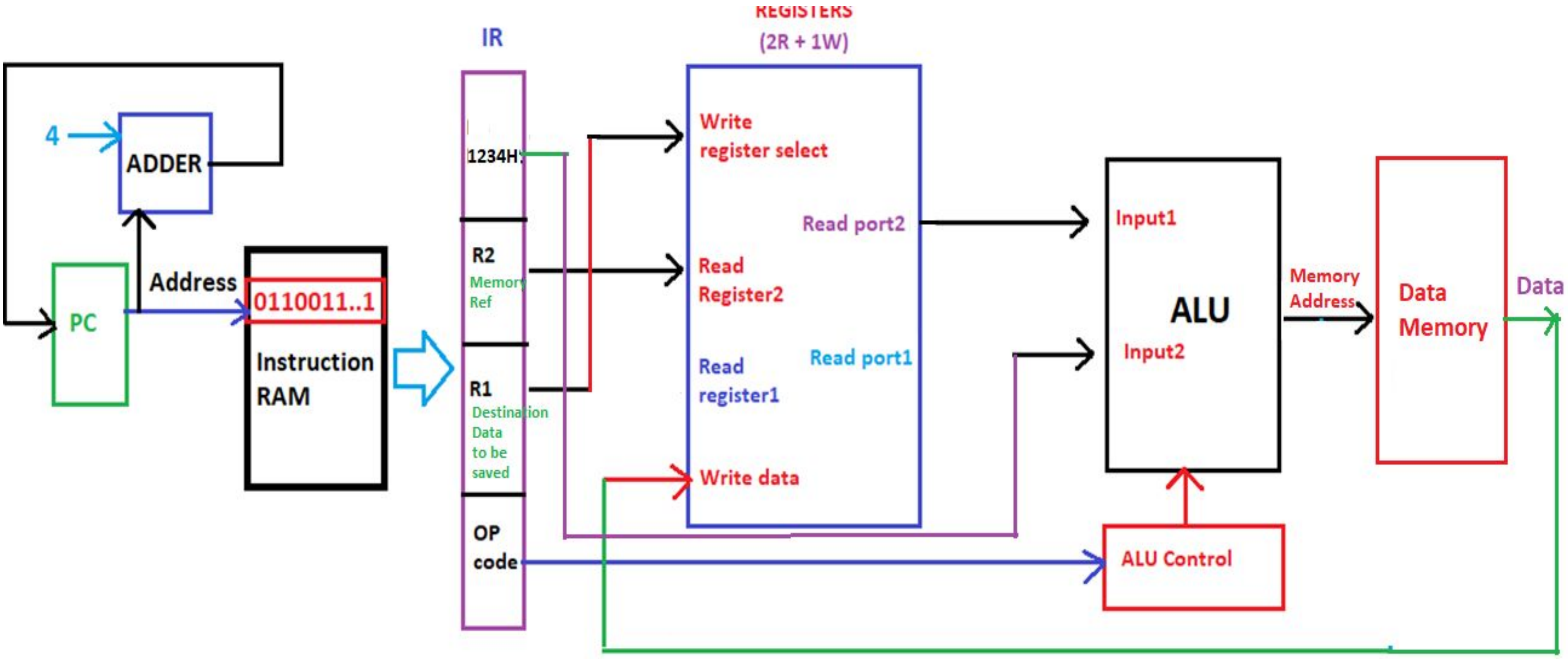
| Opcode | data-1 | R2(data-2) | R3(result) |
|---|---|---|---|

Immediate Mode & Register Mode

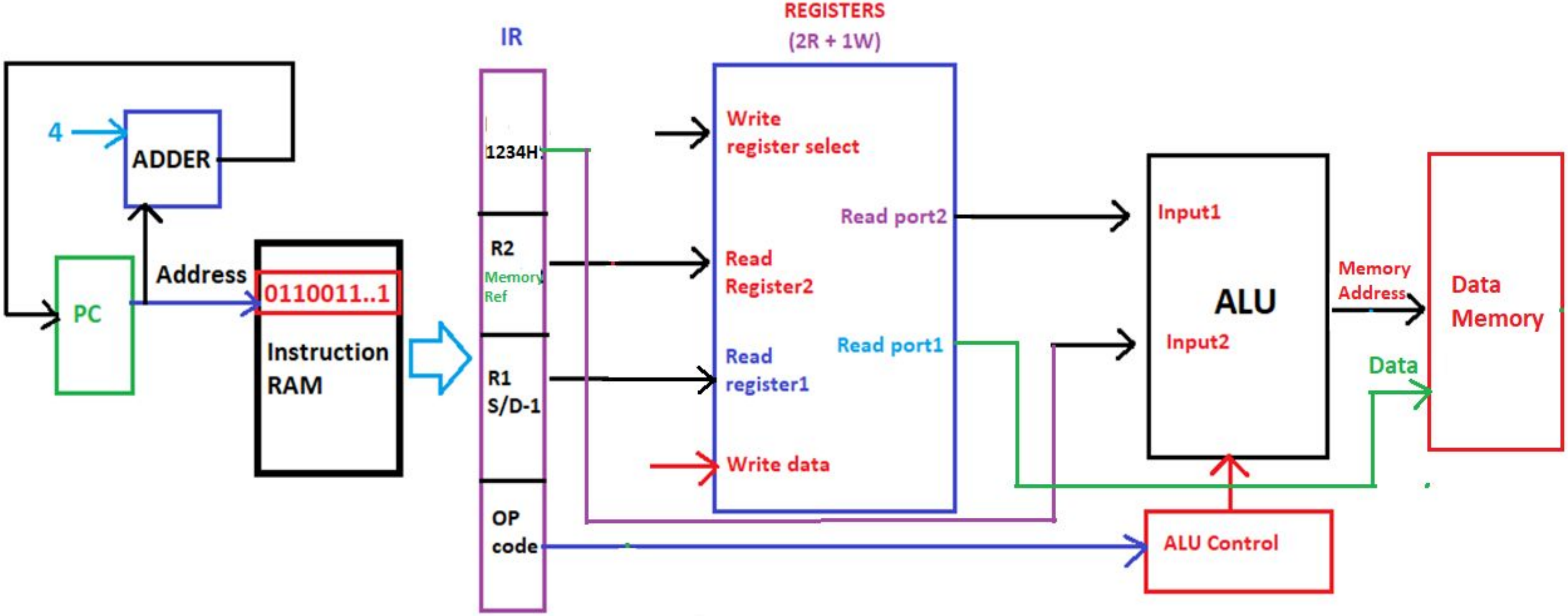


# Memory Indirect mode

| | | | |
|---|---|---|---|
| LOAD | R1(destination) | R2(base) | 124(offset) |



|  |  |  |  |
|---|---|---|---|
| STORE | R1 (source) | R2(base) | 124(offset) |


REGISTERS
(2R + 1W)
IR
4
ADDER
PC
Address
0110011..1
Instruction
RAM
1234H
R2
Memory
Ref
R1
S/D-1
OP
code
Write
register select
Read port2
Read
Register2
Read
register1
Read port1
Write data
Input1
ALU
Input2
ALU Control
Memory
Address
Data
Memory
Data