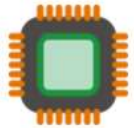
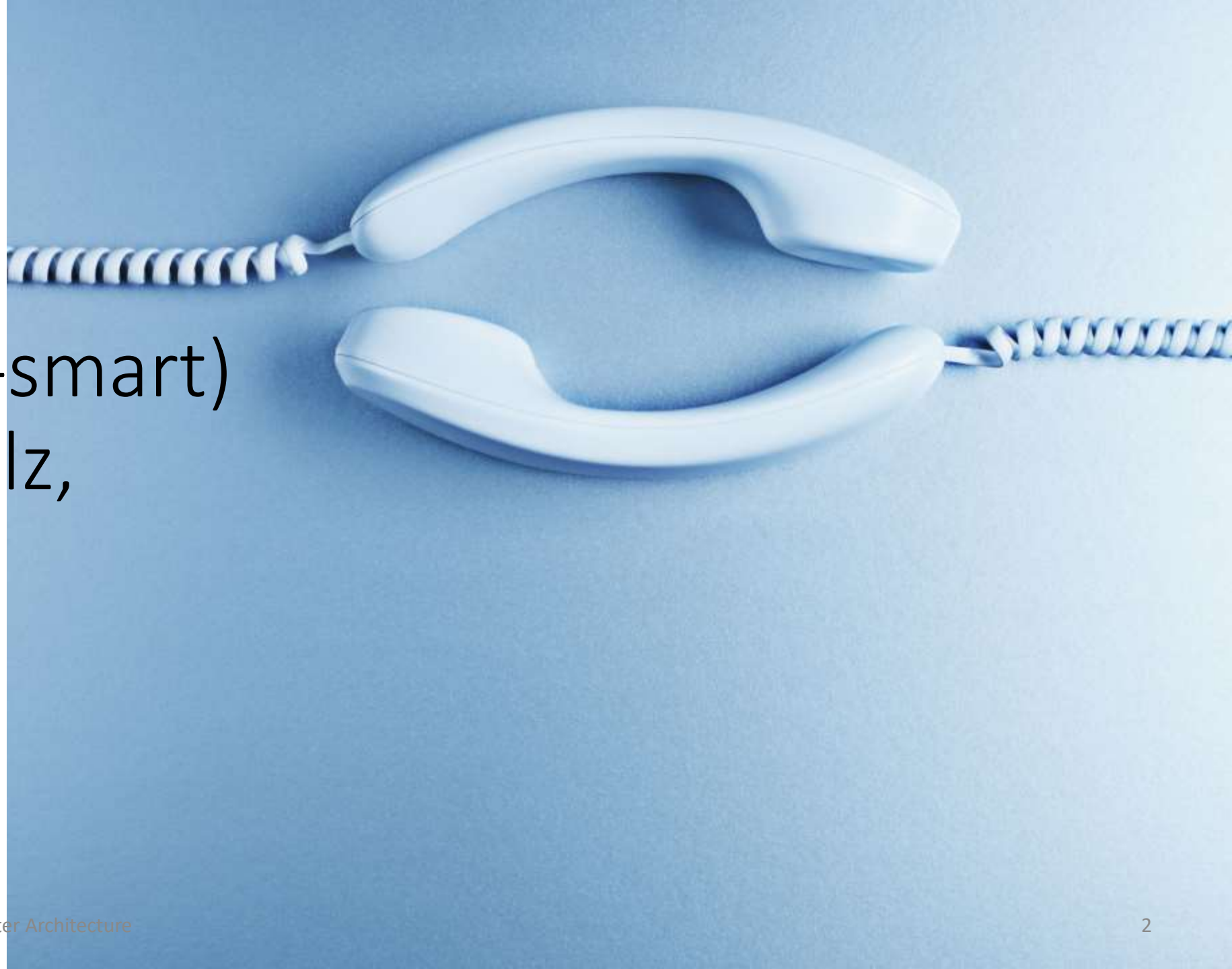



# CS230: Digital Logic Design and Computer Architecture

## Lecture 11: Intro. to Single cycle CPU

https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html

Phones (smart/non-smart) on silence plz, Thanks



# Single Cycle Processor

- All operations – single cycle ☺
- Clock cycle (unit of time) will be defined based on the longest instruction.
- Two paths of interest: datapath and control. Control tells datapath what to do.
- Do not forget the stored program concept.

# Clock Cycle

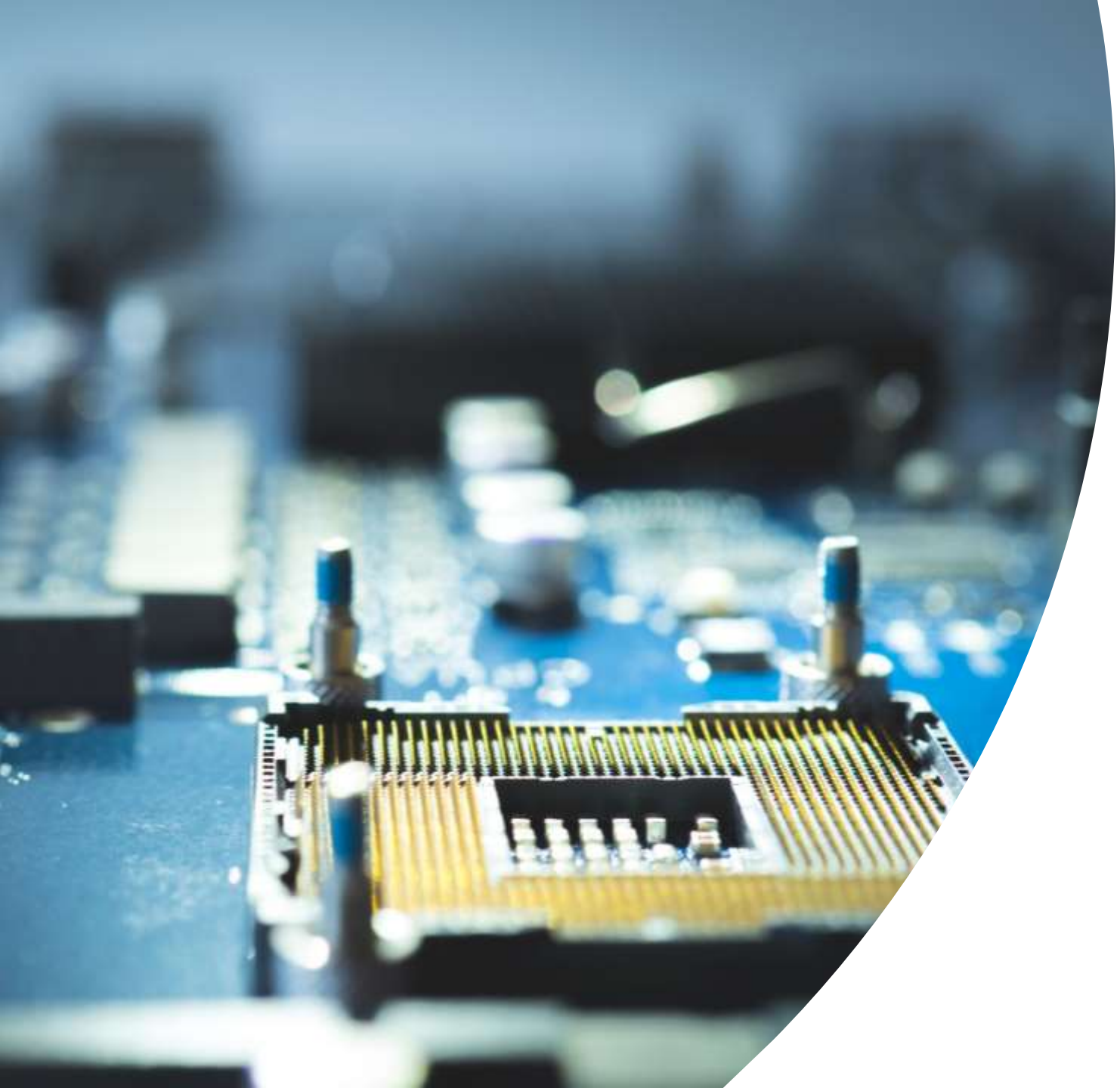
Tick, clock tick, clock period, clock, clock cycle, or cycle

Discrete time intervals

Based on processor frequency (clock rate)

1GHz processor, clock cycle = 1ns

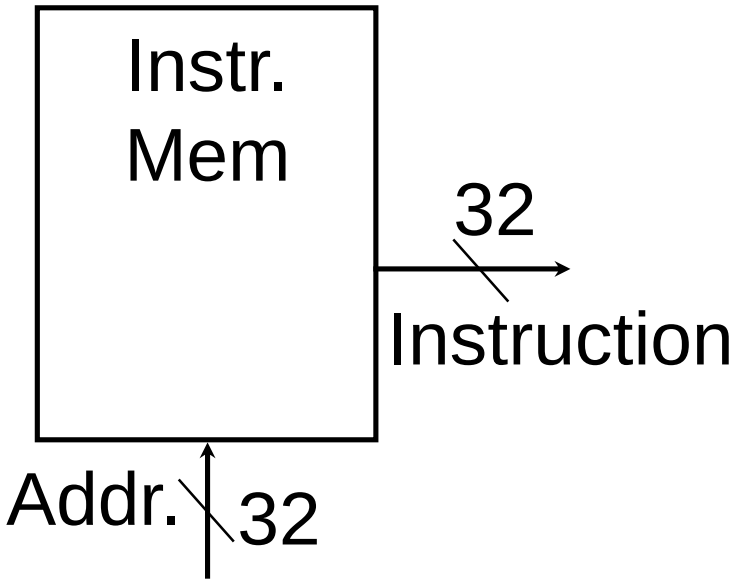
4GHz processor, clock cycle = 0.25ns



# Let’s start with the datapath

Anything that stores data or operates on data, within a processor

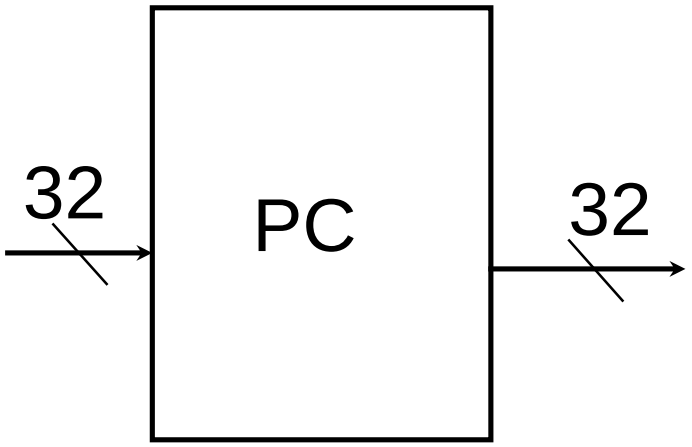
# Instruction Memory




Remember: No writes to instruction memory ☺

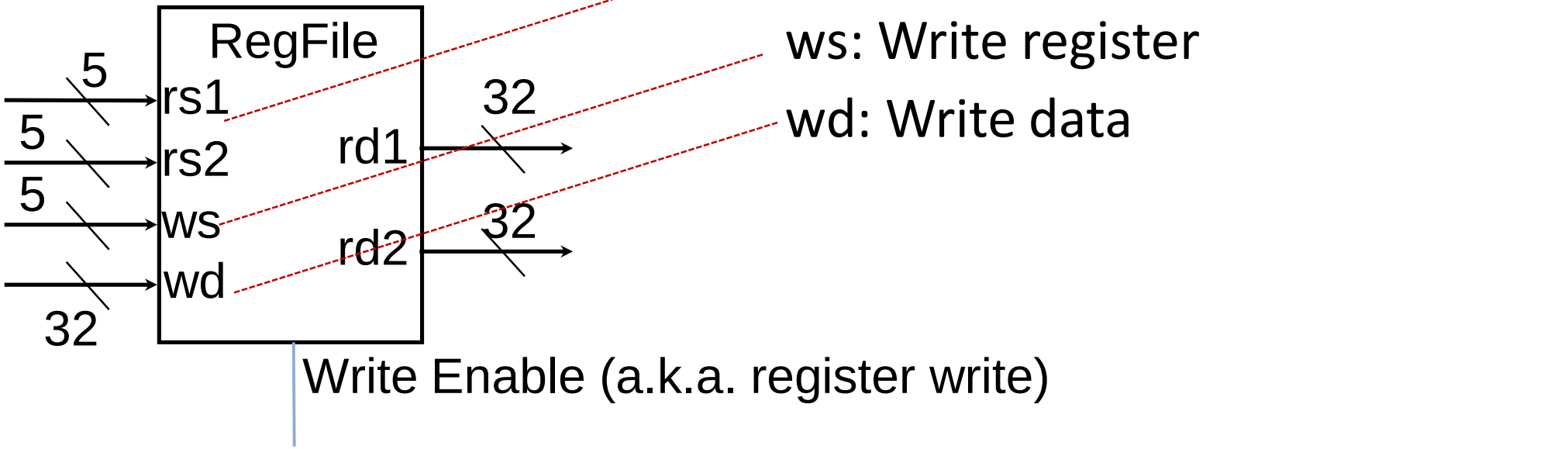
Not concerned about how programs are loaded into this memory.

# Program Counter

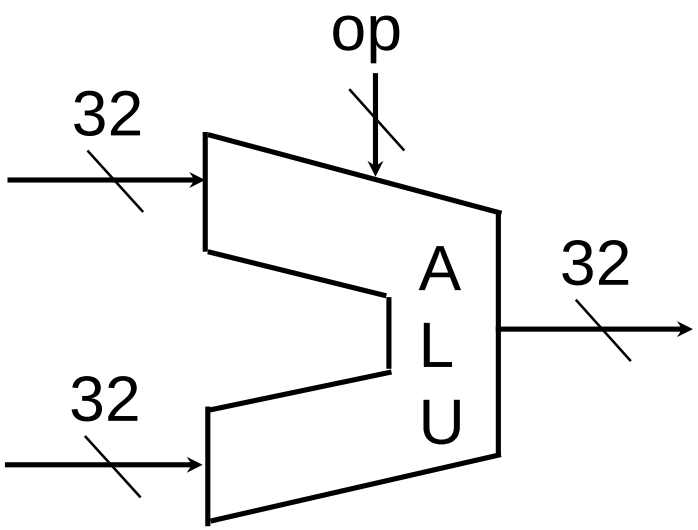



Remember: No writes to instruction memory ☺

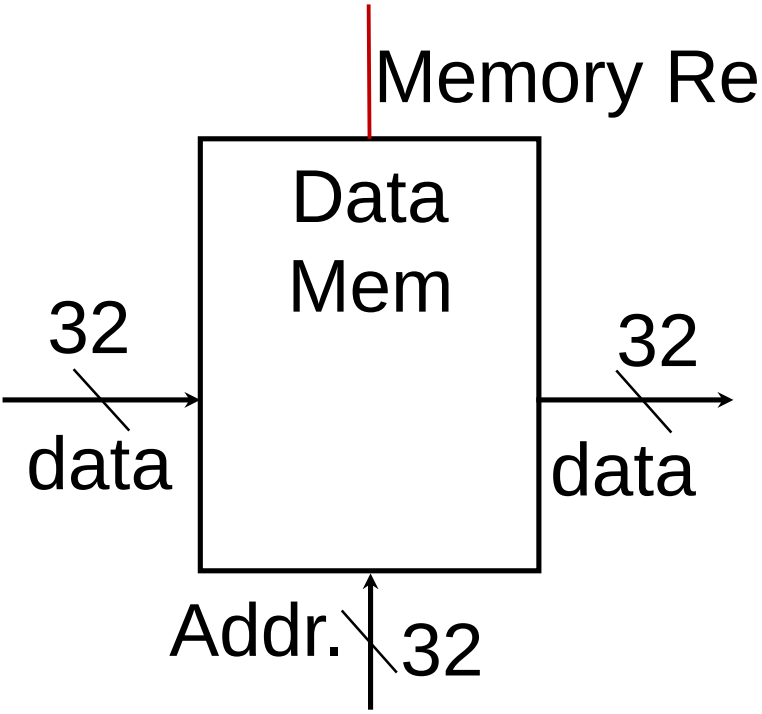
# Register File



# The ALU



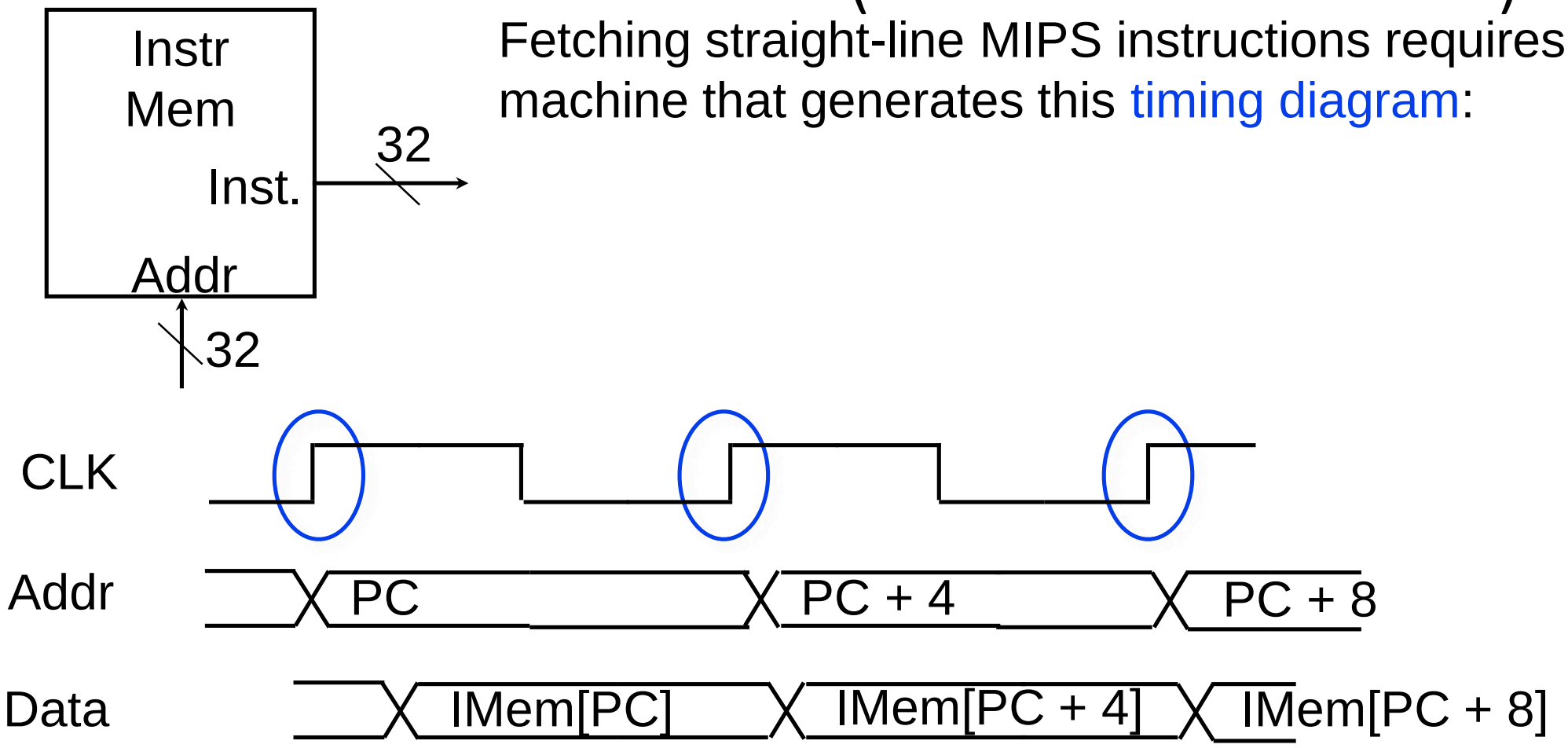
# Data Memory




Why data and instruction memory and not one memory?
Later in the course ☺

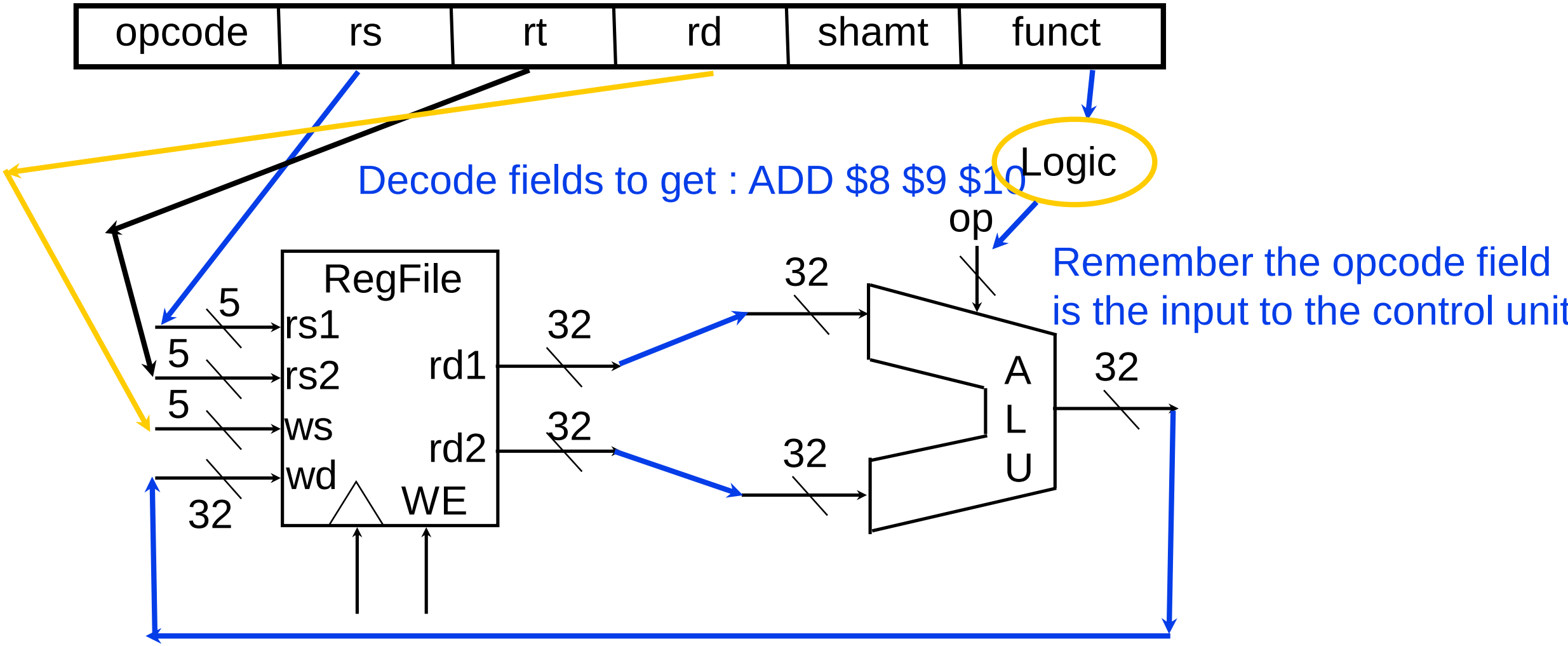
# Address and Data Bus (Instruction Fetch)

Fetching straight-line MIPS instructions requires a machine that generates this timing diagram:

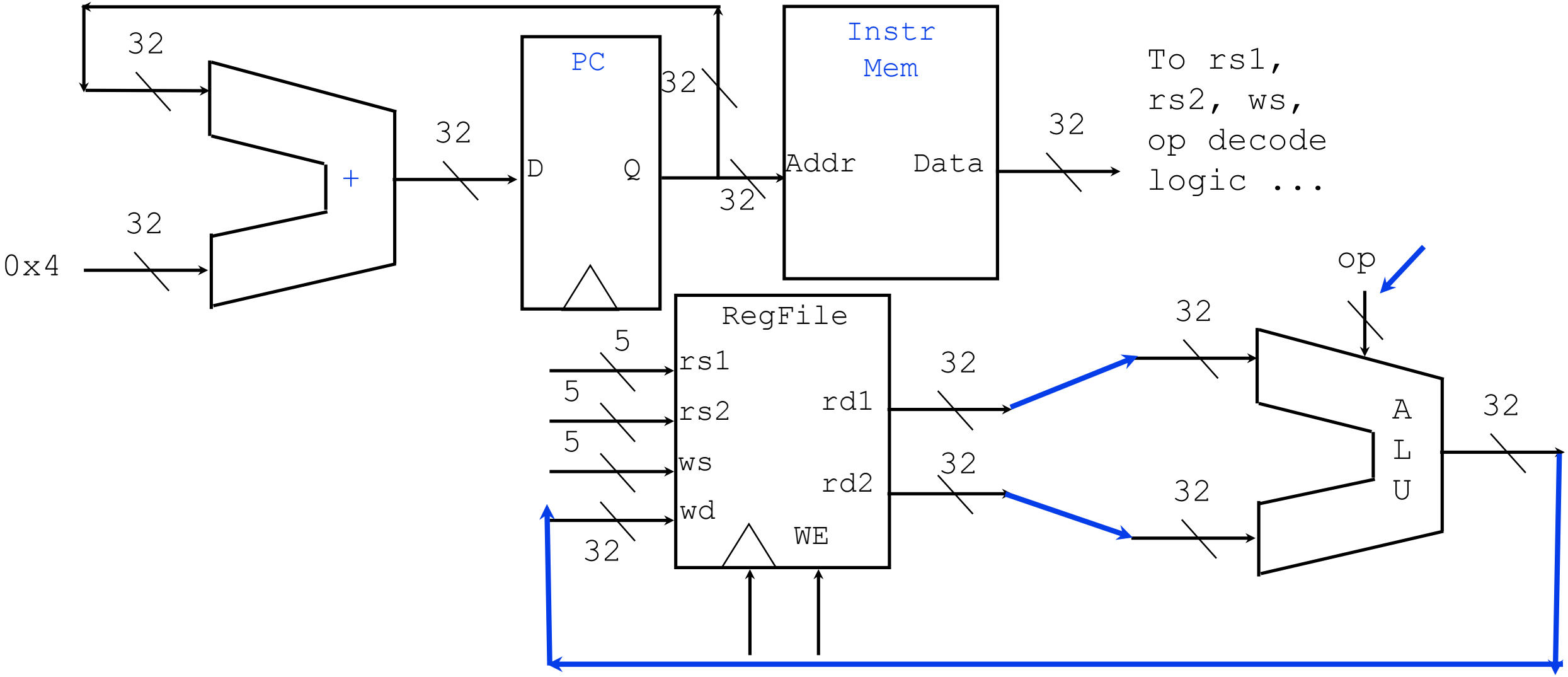



PC == Program Counter, points to next instruction.

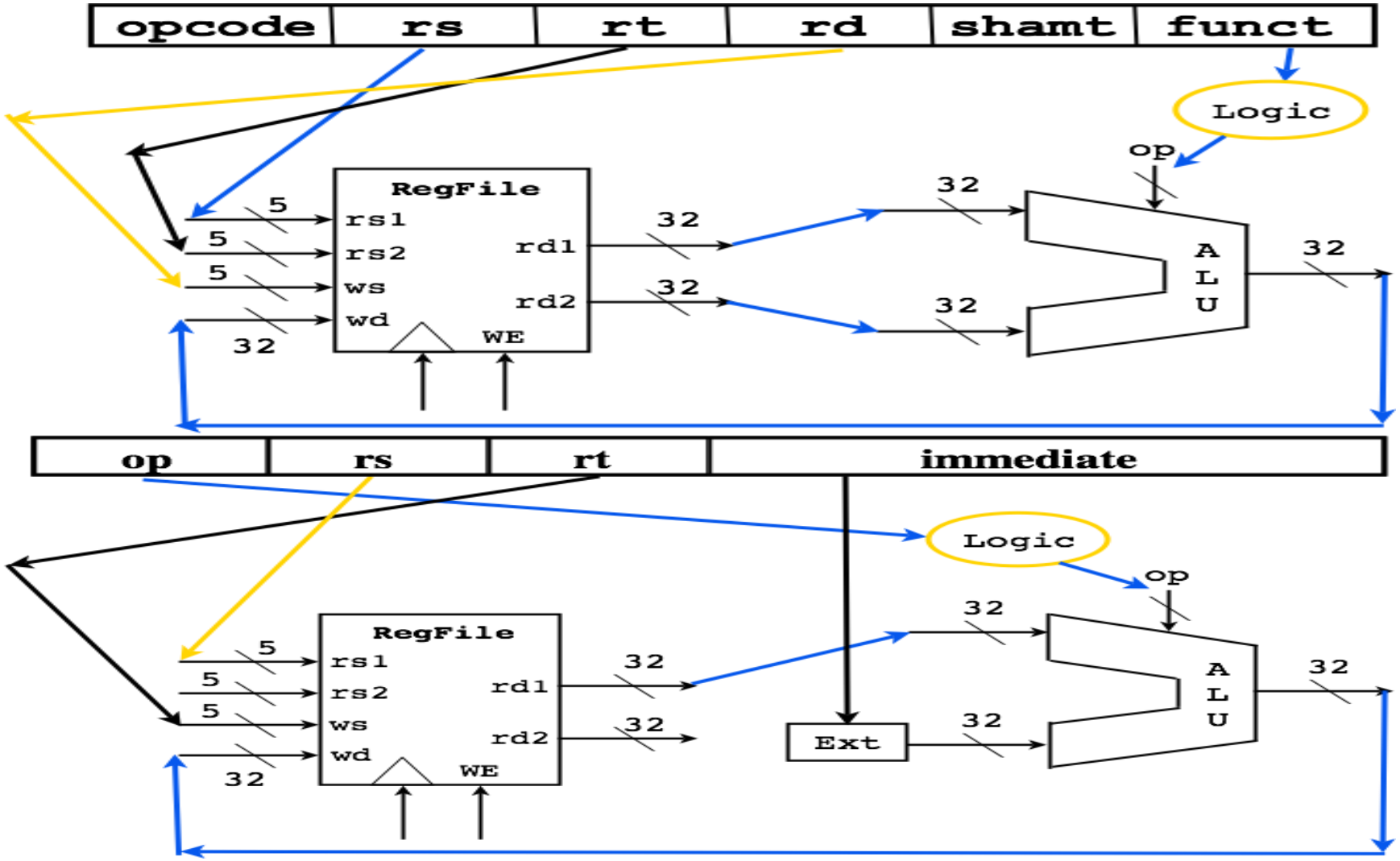
# Decode and Execute



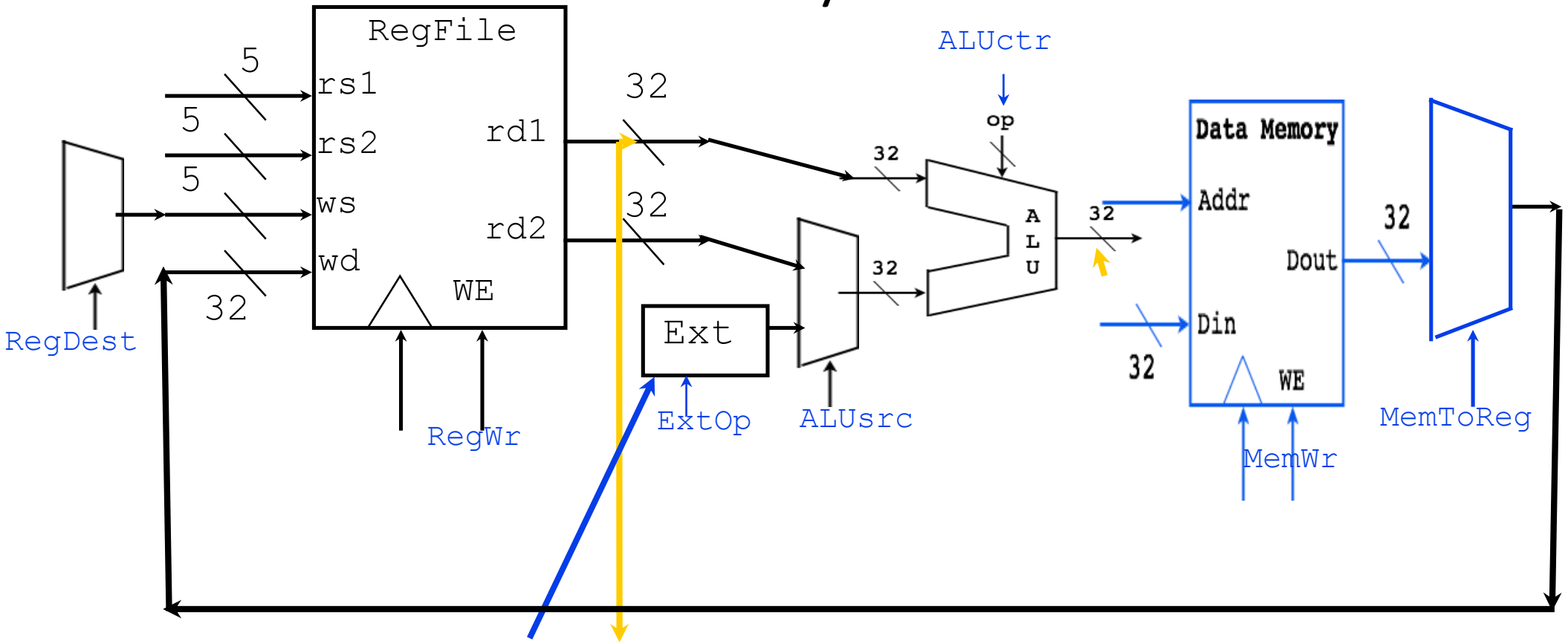
# All in one go


32
32
0x4
+
32
PC
D
Q
32
32
Instr
Mem
Addr
Data
32
To rs1,
rs2, ws,
op decode
logic ...
op
RegFile
5
5
5
32
rs1
rs2
ws
wd
rd1
rd2
WE
32
32
32
32
A
L
U
32

# What about I format?



# Loads from Memory

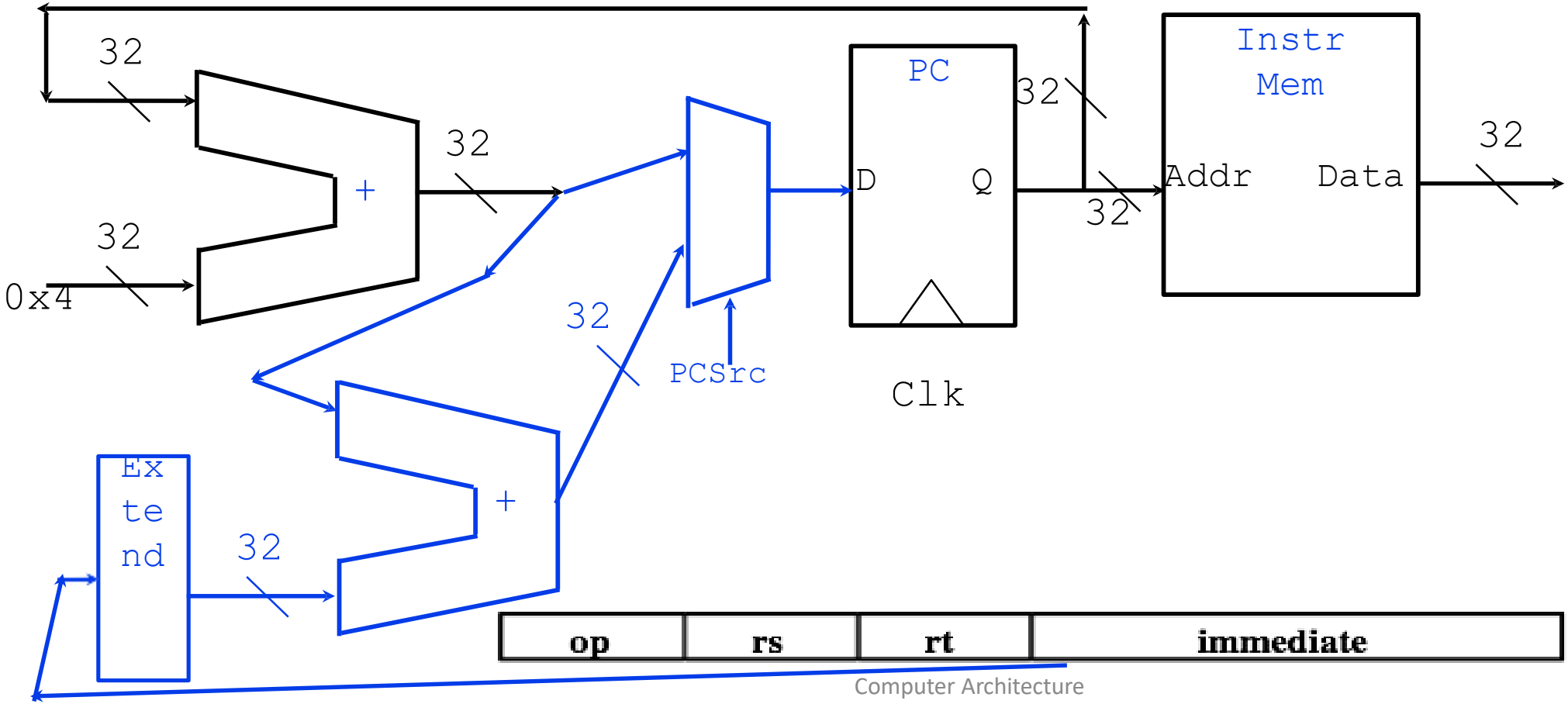



Syntax: LW $1, 32($2)

Action: $1 = M[$2 + 32]

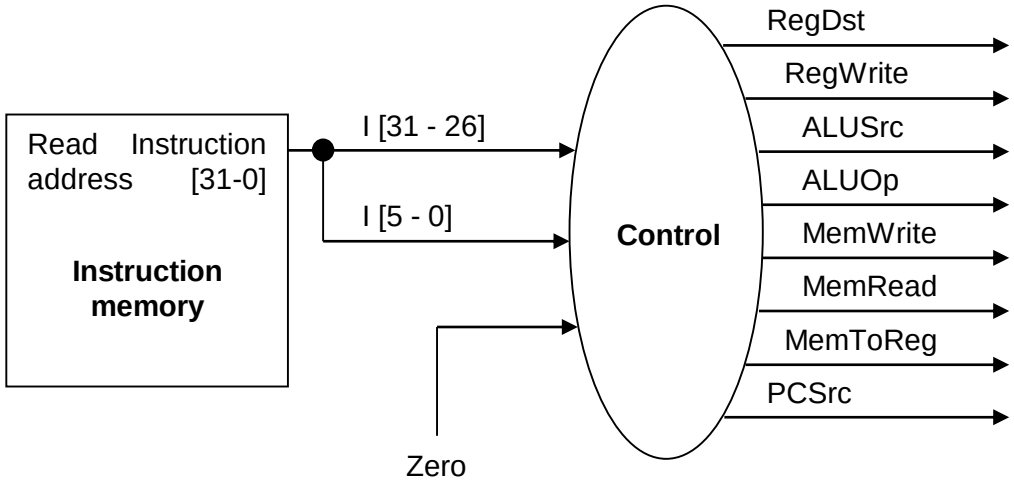
# Branch Instructions

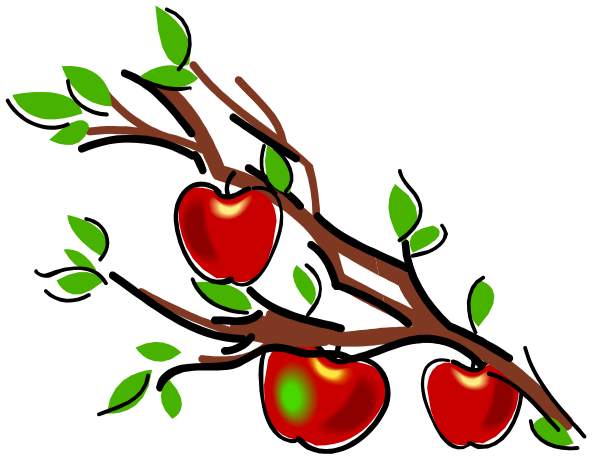
```
Syntax: BEQ $1, $2, 12
Action: If ($1 != $2), PC = PC + 4
Action: If ($1 == $2), PC = PC + 4 + 48
```



# Control Signals So far

- MemRead
- MemWrite
- RegWrite
- MemtoReg
- RegDst
- ALUop, ALUSrc
- PCSrc (we have not discussed about the branch)



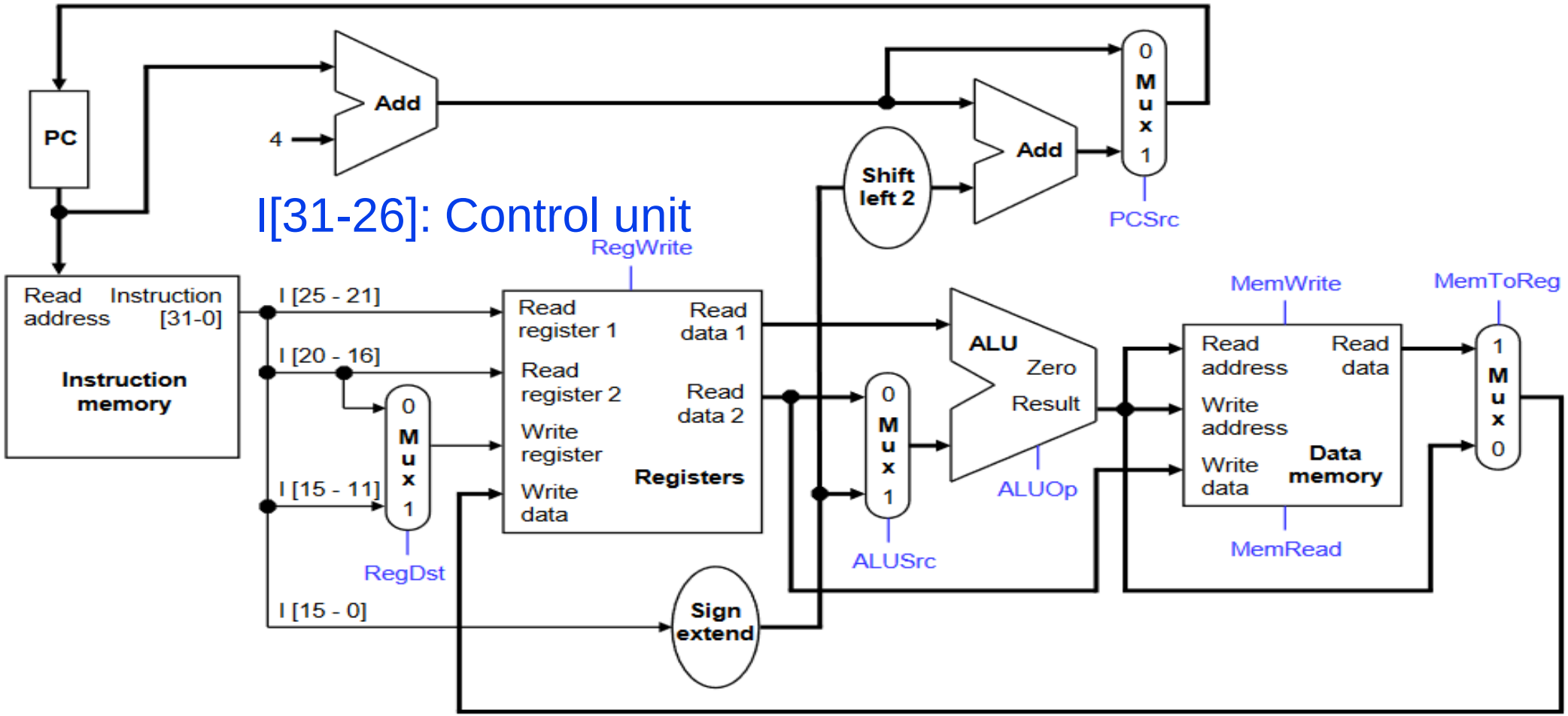
# In detail

- MemRead: Read from memory when assert
- MemWrite: Write into the memory when assert
- RegWrite: Reg. on Write register updated with the input, on assert
- MemtoReg: On assert, memory to register, on deassert, ALU to register
- RegDst: On assert, use rd field, on deassert use rt field
- ALUSrc: On assert, lower 16 bits of an inst., on deassert from the second register
- PCSrc: On assert, branch target, deassert, PC+4

# Control Signal Table

| Operation | RegDst | RegWrite | ALUSrc | ALUOp | MemWrite | MemRead | MemToReg |
|---|---|---|---|---|---|---|---|
| add | 1 | 1 | 0 | 010 | 0 | 0 | 0 |
| sub | 1 | 1 | 0 | 110 | 0 | 0 | 0 |
| and | 1 | 1 | 0 | 000 | 0 | 0 | 0 |
| or | 1 | 1 | 0 | 001 | 0 | 0 | 0 |
| slt | 1 | 1 | 0 | 111 | 0 | 0 | 0 |
| lw | 0 | 1 | 1 | 010 | 0 | 1 | 1 |
| sw | X | 0 | 1 | 010 | 1 | 0 | X |
| beq | X | 0 | 0 | 110 | 0 | 0 | X |

# The Complete Picture



# Why not single cycle?

- The longest possible datapath is the clock cycle time.

What does it mean?

# Why not single cycle?

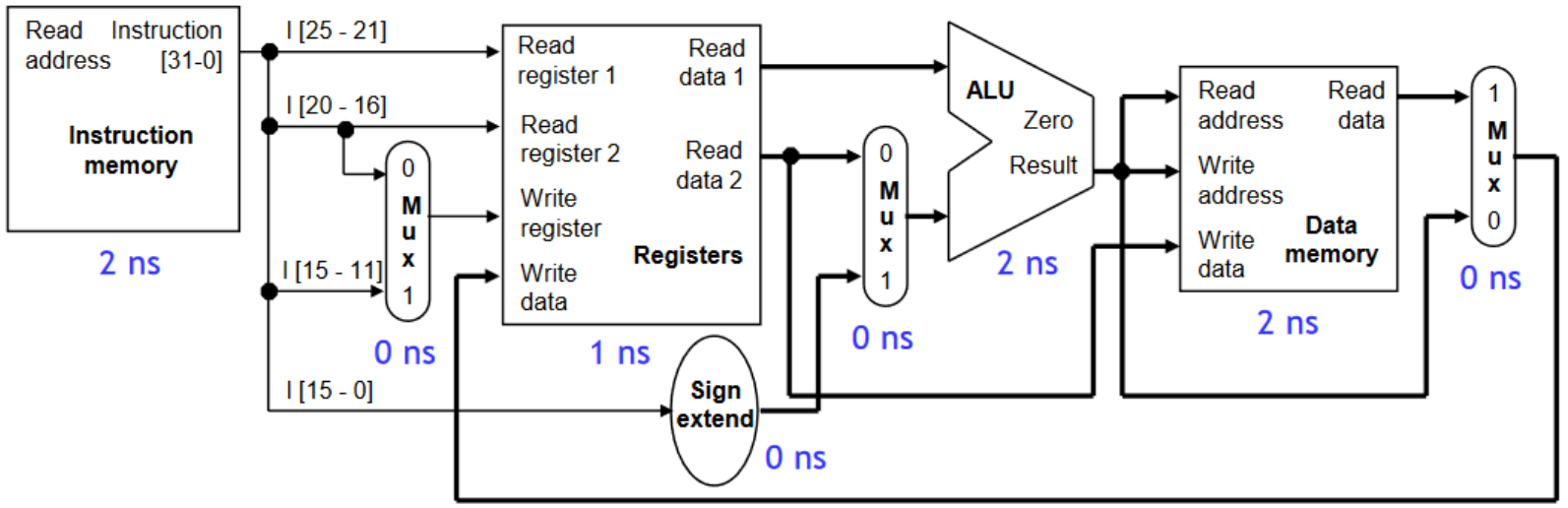
- For example, lw $t0, -4($sp) needs 8ns, assuming the delays shown here.

| | | |
|---|---|---|
| reading the instruction memory | 2ns | 8ns |
| reading the base register $sp | 1ns | |
| computing memory address $sp-4 | 2ns | |
| reading the data memory | 2ns | |
| storing data back to $t0 | 1ns | |

one clock cycle: 8ns

Processor frequency: 125MHz

Cycle per Instruction (CPI): 1



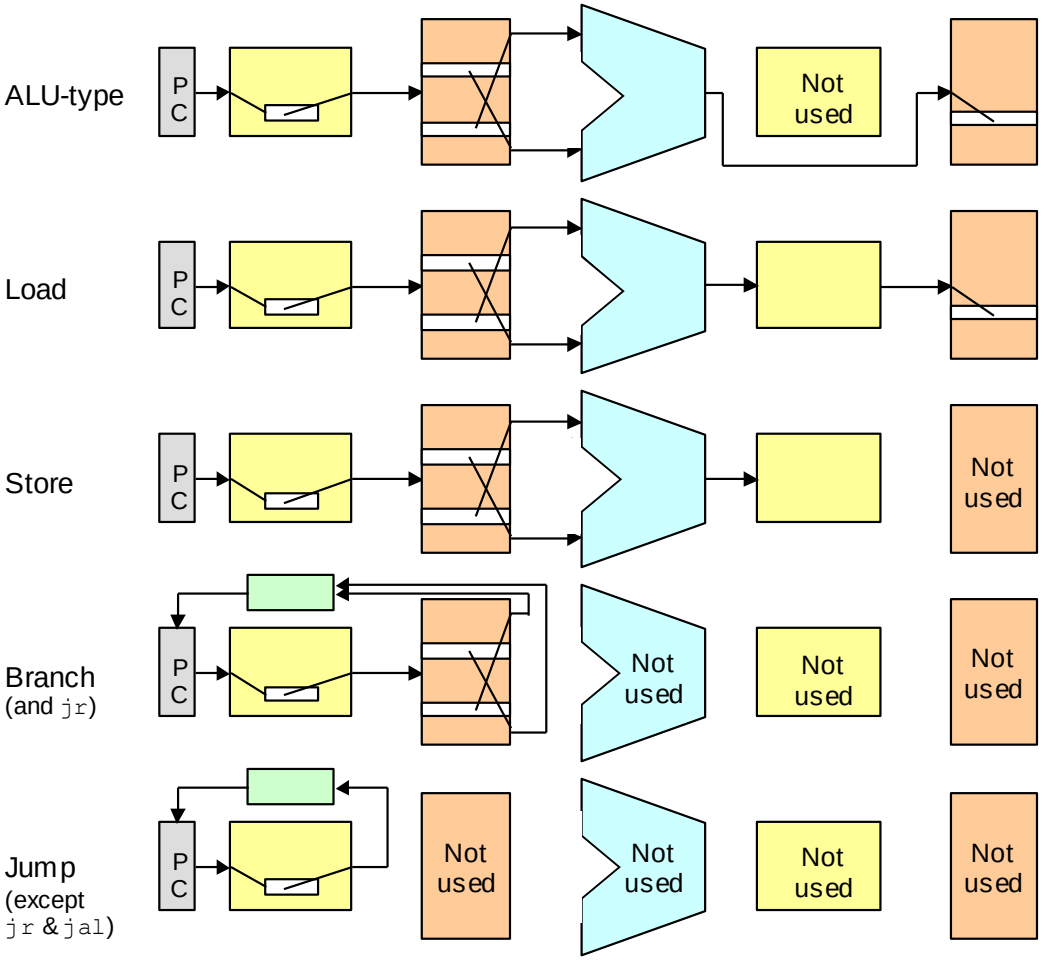

An add instruction: no need of 8ns

# Why not single cycle?

- The longest possible datapath is the clock cycle time.

Violating *common case fast (Confucius says)*

# Oh No! Such a bad design



# Single to Multi Cycle