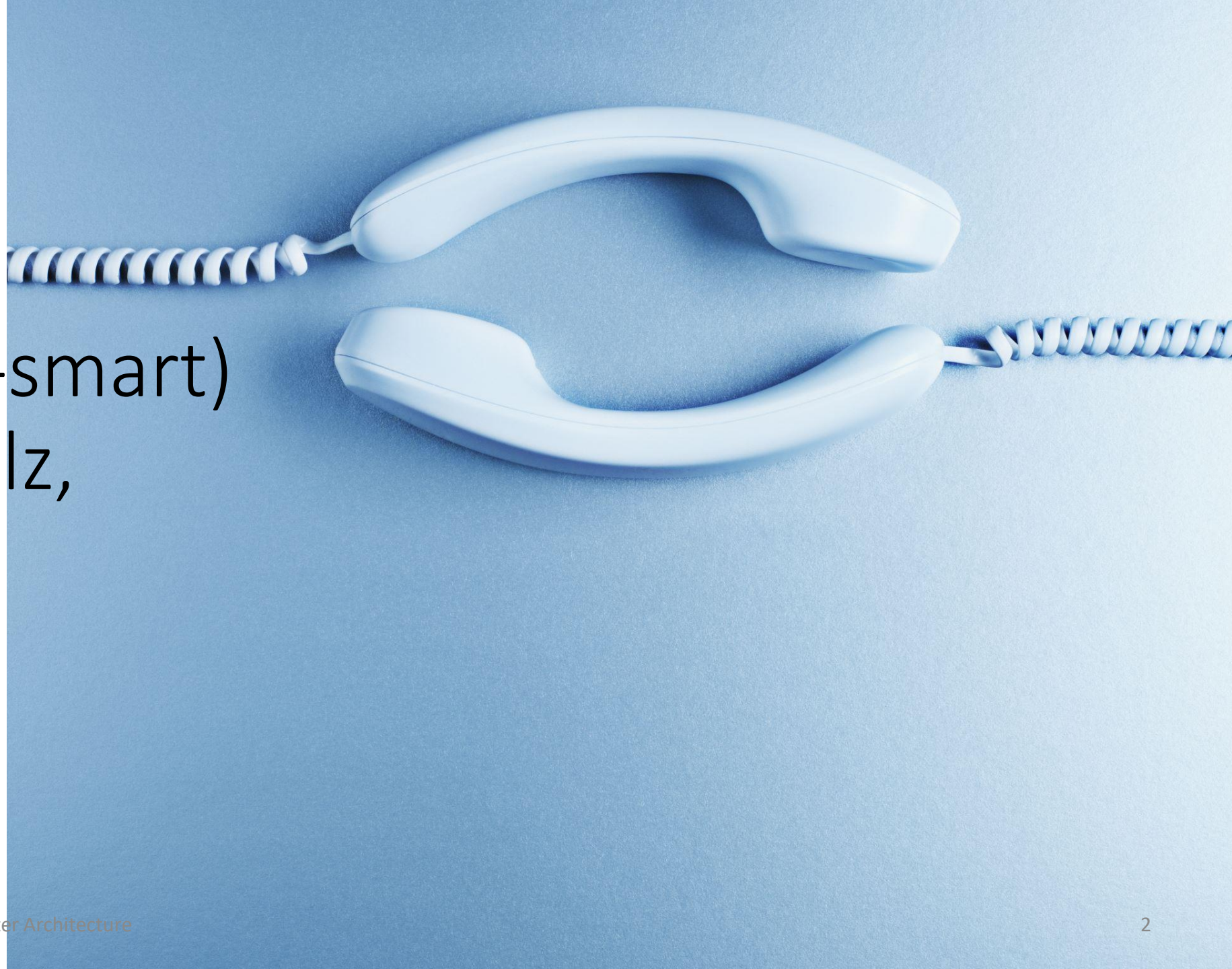
CASPER

CS230: Digital Logic Design and Computer Architecture

Lecture 10: Intro. to Single cycle CPU

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

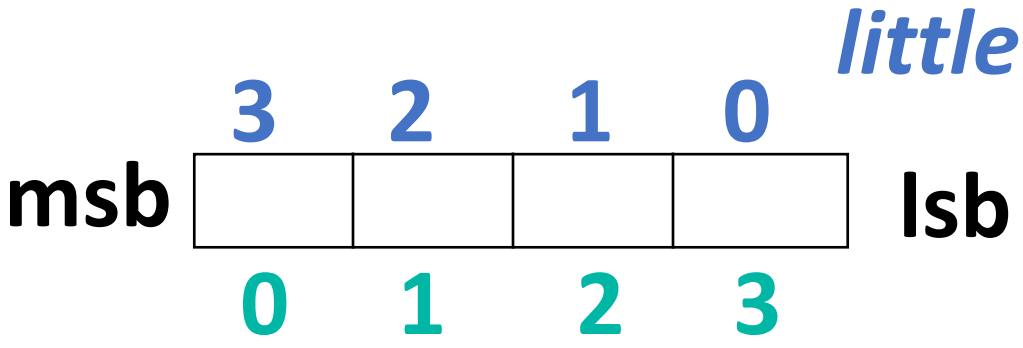
Phones (smart/non-smart) on silence plz, Thanks



Endianness (Byte ordering within a word)

- Big Endian: address of most significant byte = word address (xx00 = Big end of word), MIPS
- Little Endian: address of least significant byte = word address (xx00 = Little end of word), x86

Think about an egg ☺



Just for an example, do not take it for granted …

```
unsigned int i = 1;
char *c = (char*)&i;  // reading the LSB
Printf (“%d”, *c);
```

```
unsigned int i = 12345678;
char *c = (char*)&i;
Printf (“%d”, *c);
```

```
unsigned int i = 1;
char *c = (char*)&i;  // reading the LSB
Printf (“%d”, *c);
```

Little endian: 1

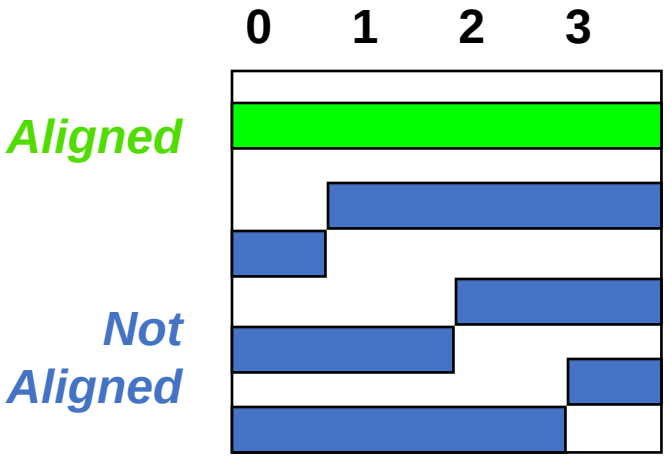
Big endian: 0

```
unsigned int i = 12345678;
char *c = (char*)&i;
Printf (“%d”, *c);
```

Little endian: 78

Big endian: 12

Instruction Alignment: Why we need it?



Aligned:

x-byte access starting from an address y: y % x must be zero.

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

MIPS vs X86

MIPS does not allow unaligned accesses

x86 does not enforce alignment ☺

Whose job is to generate aligned/unaligned accesses?

Compiler

Let’s go a bit deeper

Object of size s bytes at byte add. A is aligned if A mod s = 0

Alignment for faster transfer of data ?

Why fast ??

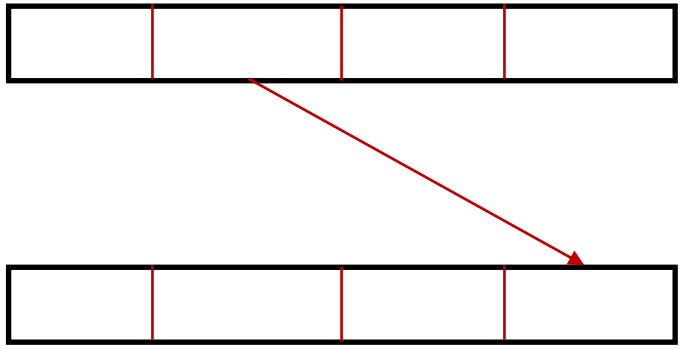
Think about memory (caches if you know).

Memory operations and alignment network

LOADs and STOREs need an alignment network that makes sure data loaded/written are aligned.

lb R1, 1($s3)

4-byte chunk



Register R1

For the Curious ones

https://lemire.me/blog/2012/05/31/data-alignment-for-speed-myth-or-reality/

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

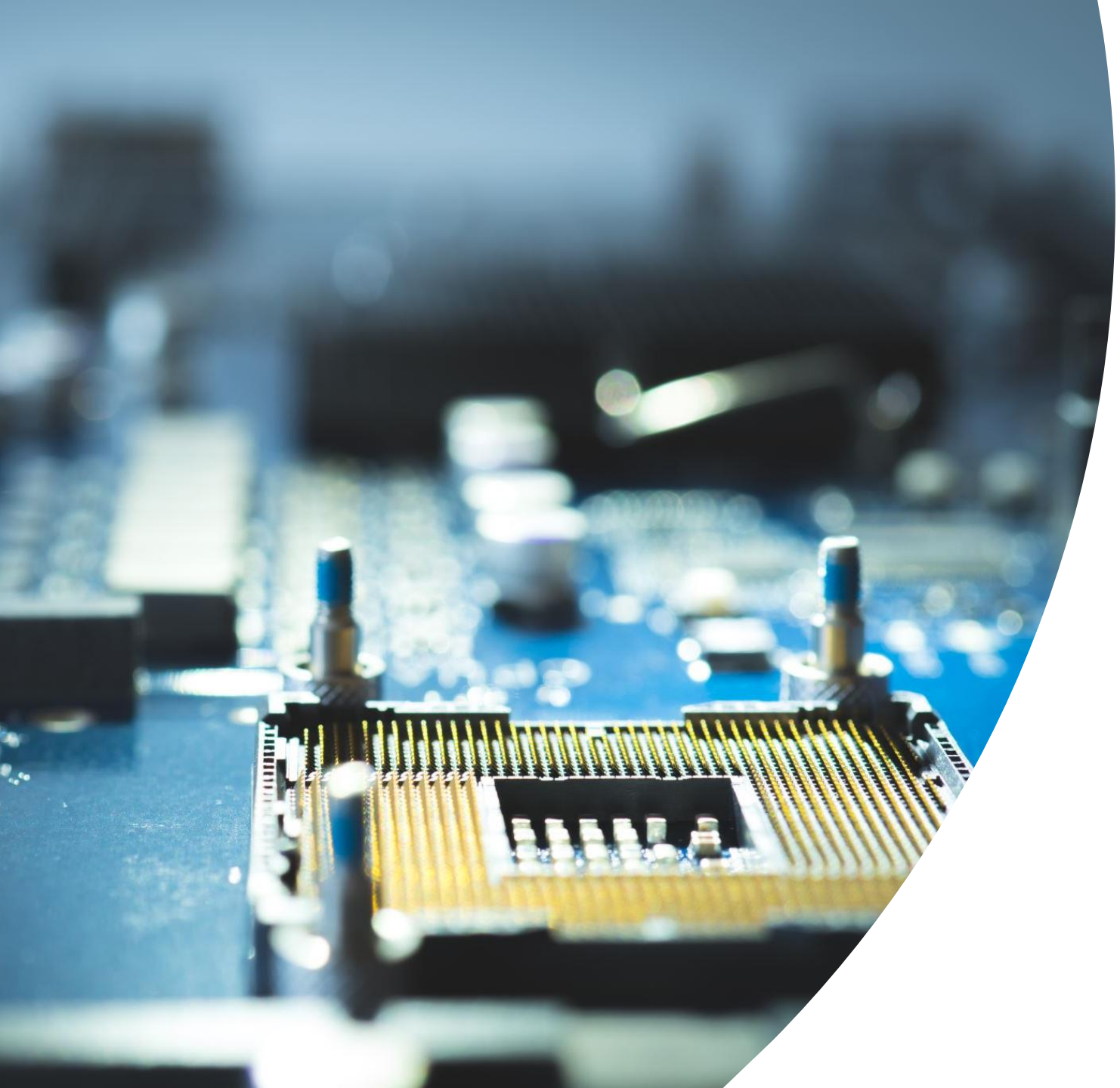
Tick, clock tick, clock period, clock, clock cycle, or cycle

Discrete time intervals

Based on processor frequency (clock rate)

1GHz processor, clock cycle = 1ns

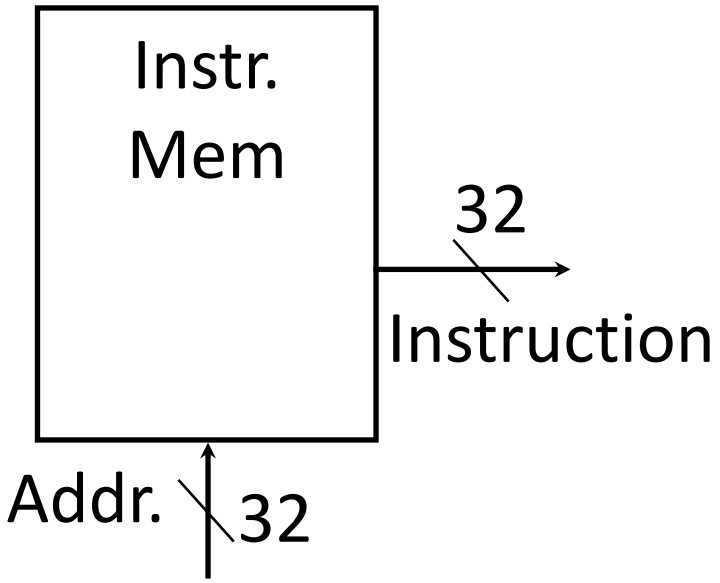
4GHz processor, clock cycle = 0.25ns



Let’s start with the datapath

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

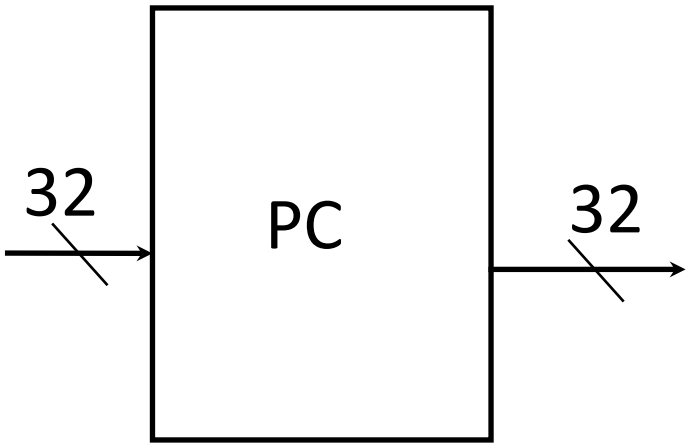
Instruction Memory



Remember: No writes to instruction memory ☺

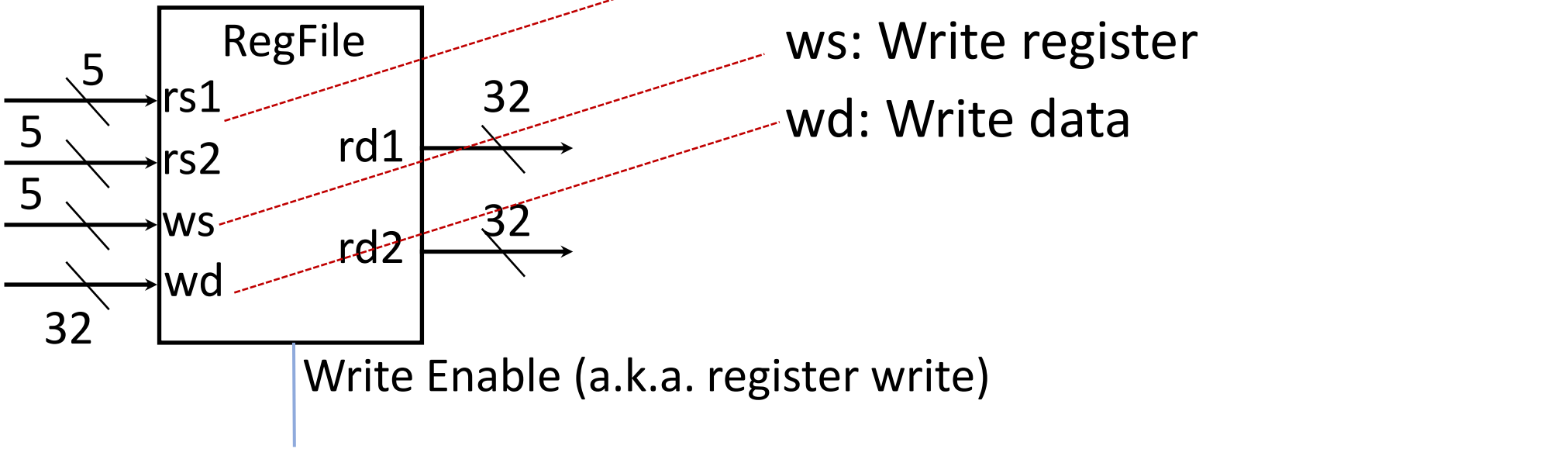
Not concerned about how programs are loaded into this memory.

Program Counter

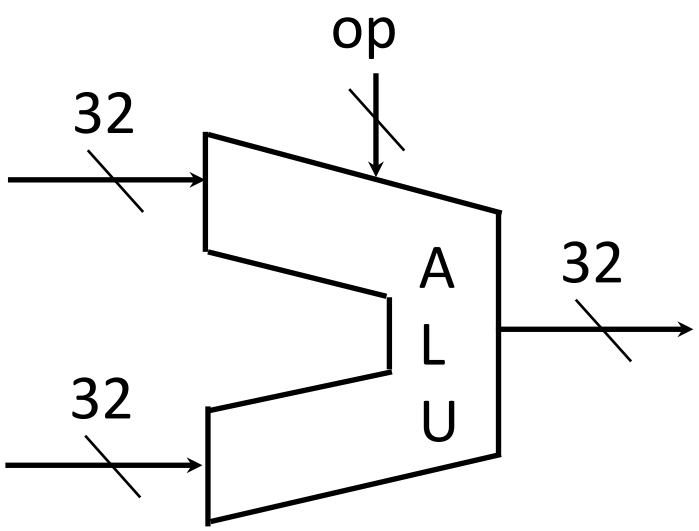


Remember: No writes to instruction memory ☺

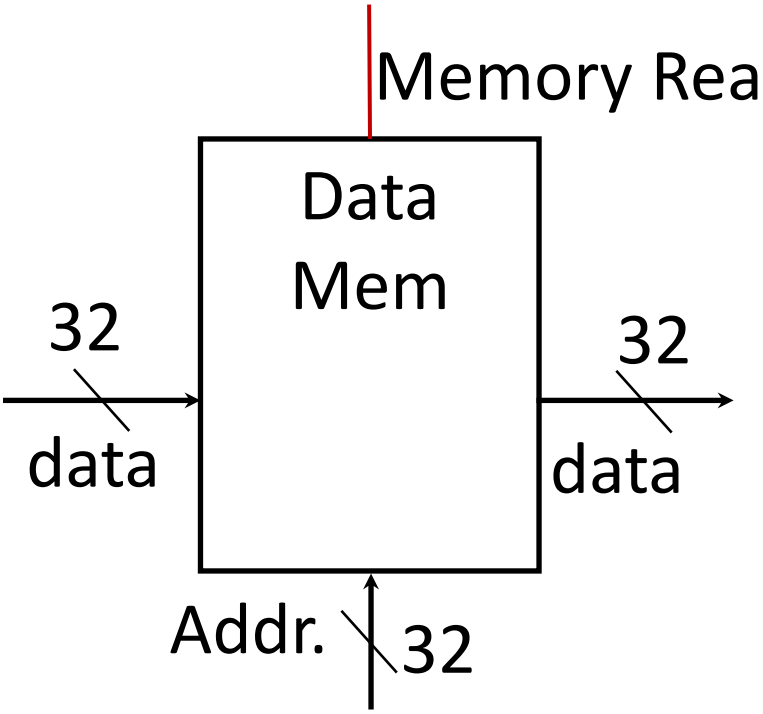
Register File



The ALU



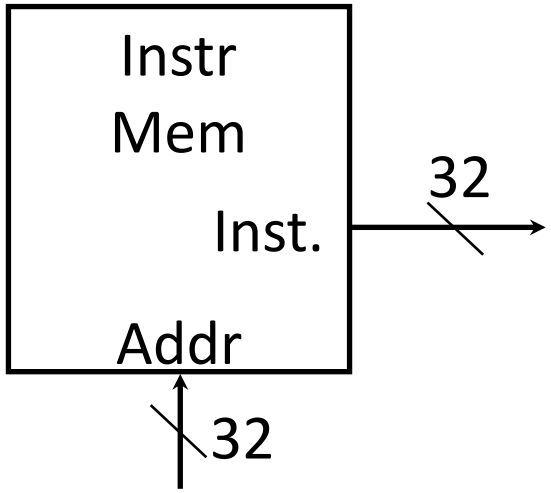
Data Memory



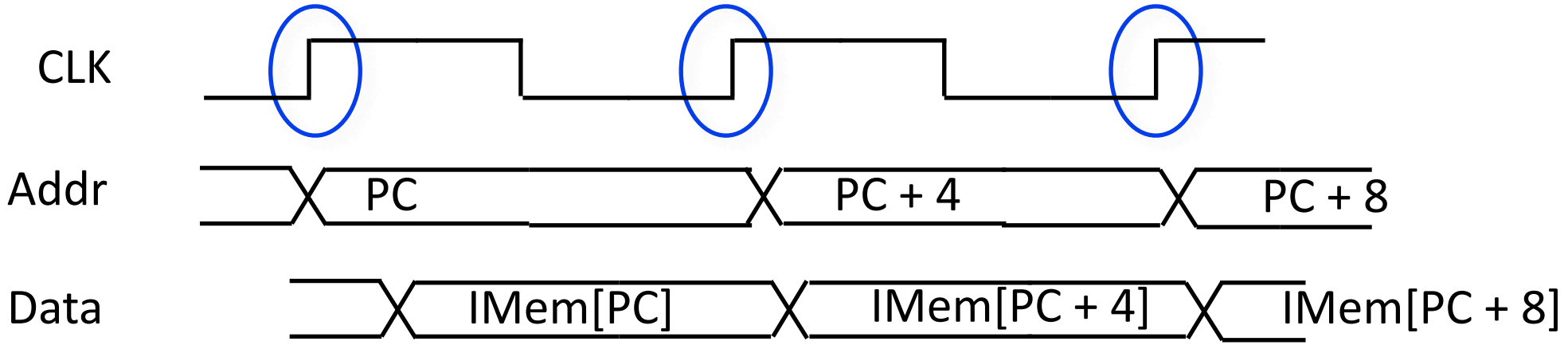
Why data and instruction memory and not one memory?

Later in the course ☺

Address and Data Bus (Instruction Fetch)

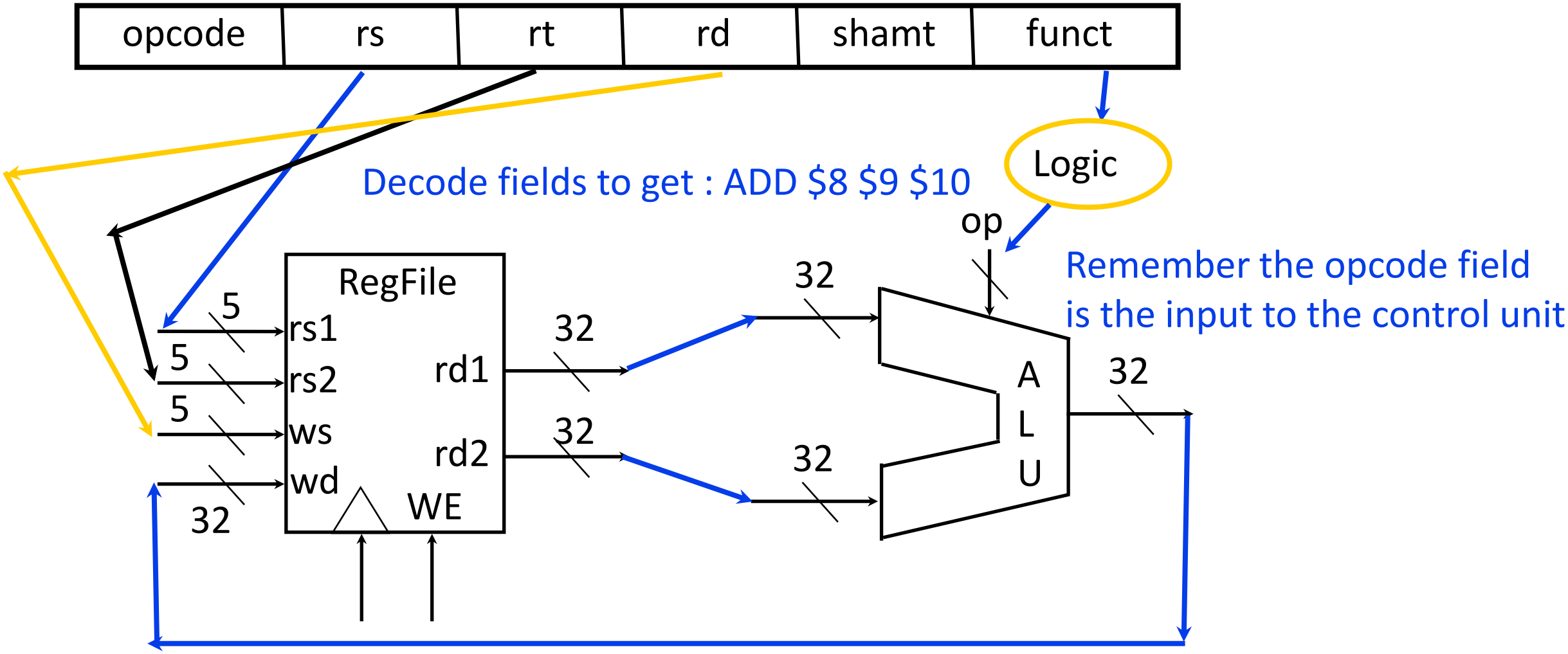


Fetching straight-line MIPS instructions requires a machine that generates this timing diagram:

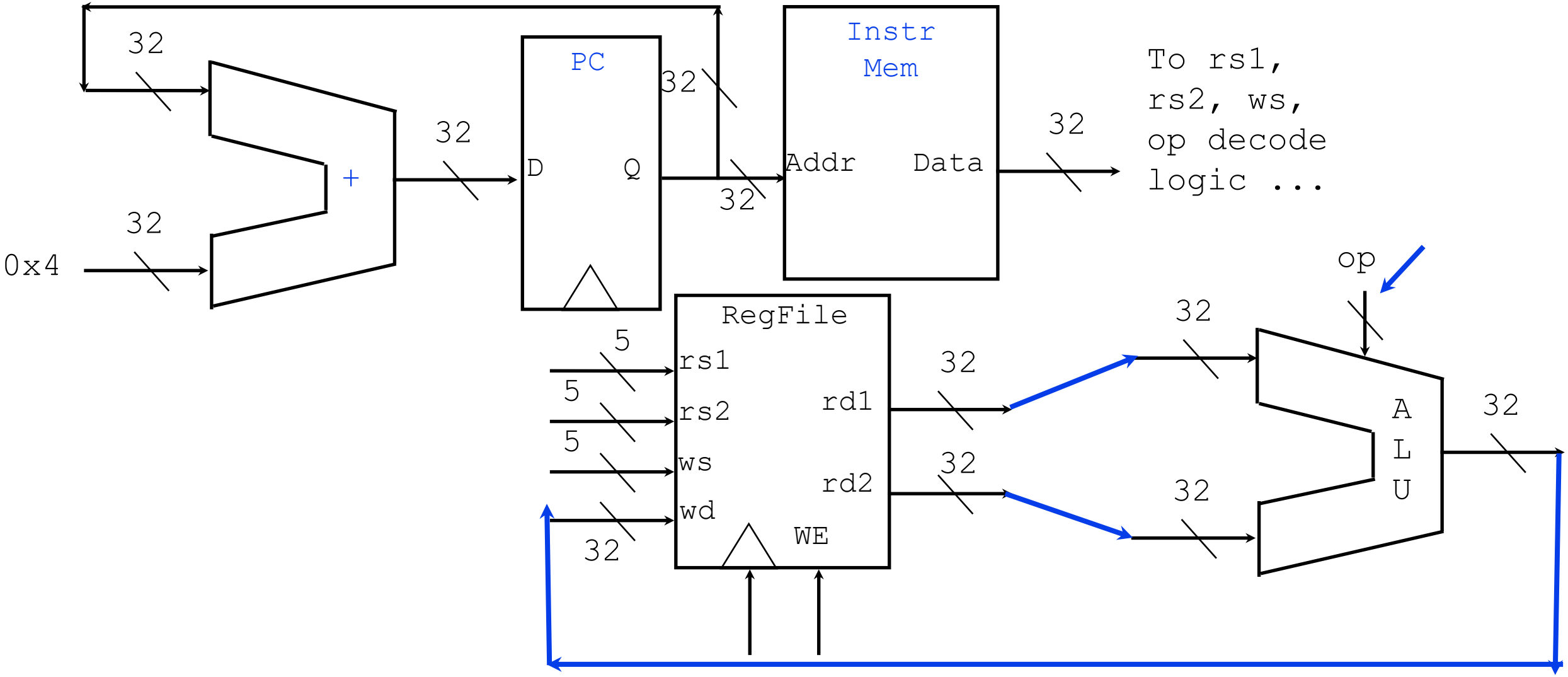


PC == Program Counter, points to next instruction.

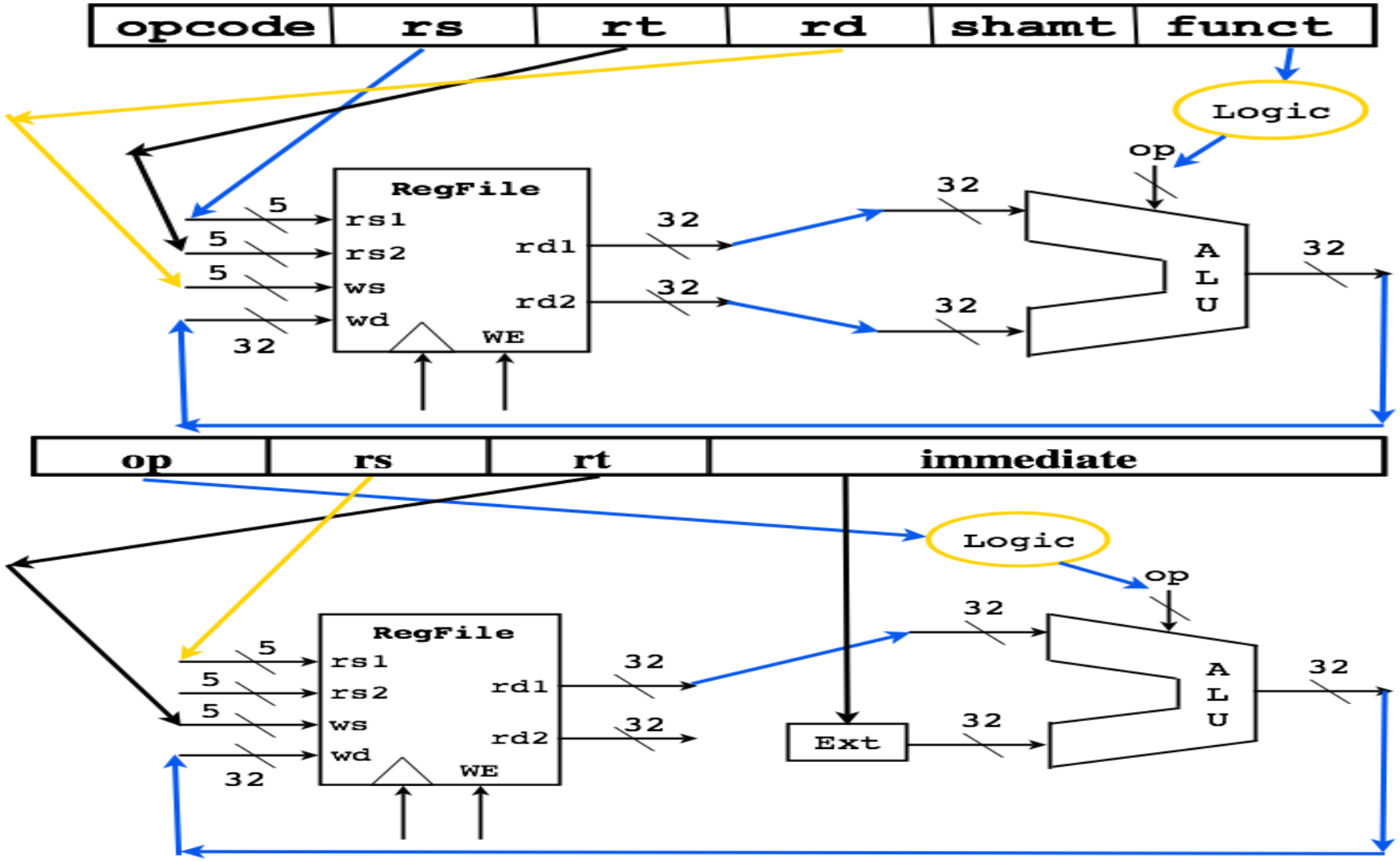
Decode and Execute



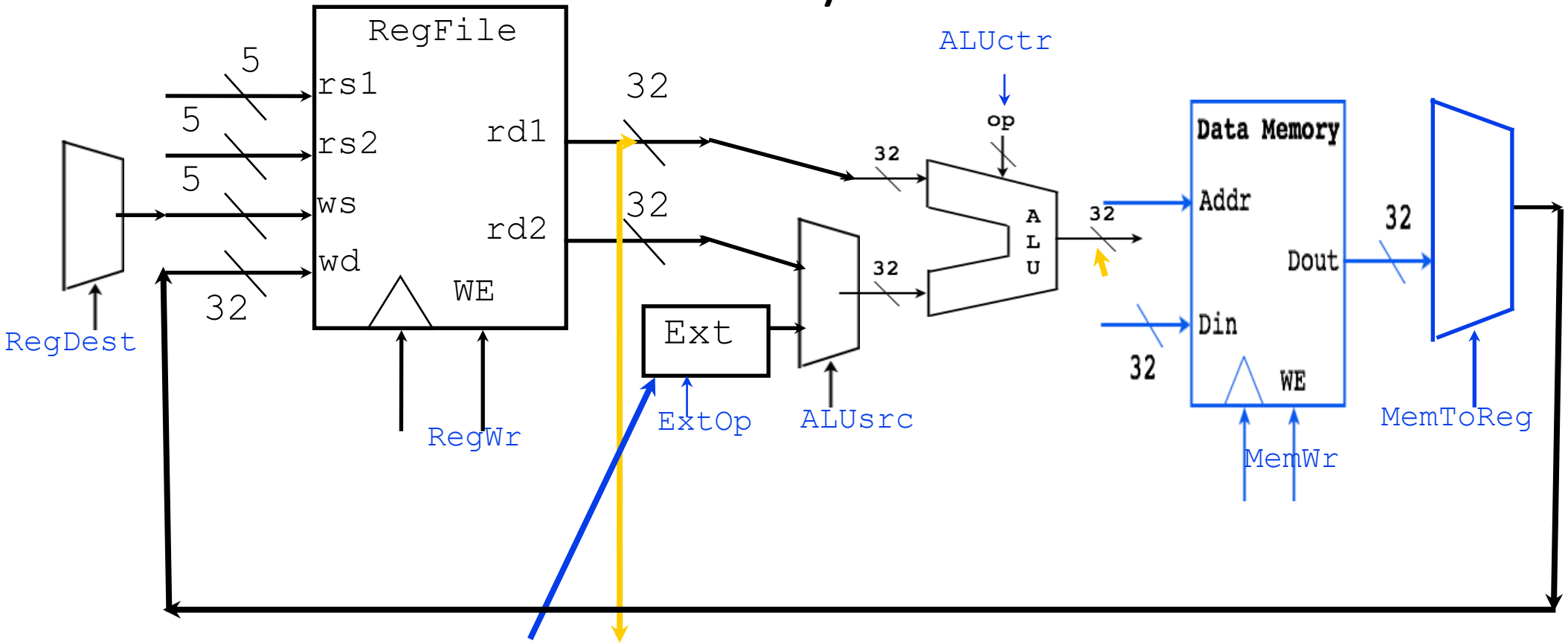
All in one go


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
WE
rd1
rd2
32
32
32
32
A
L
U
32

What about I format?



Loads from Memory

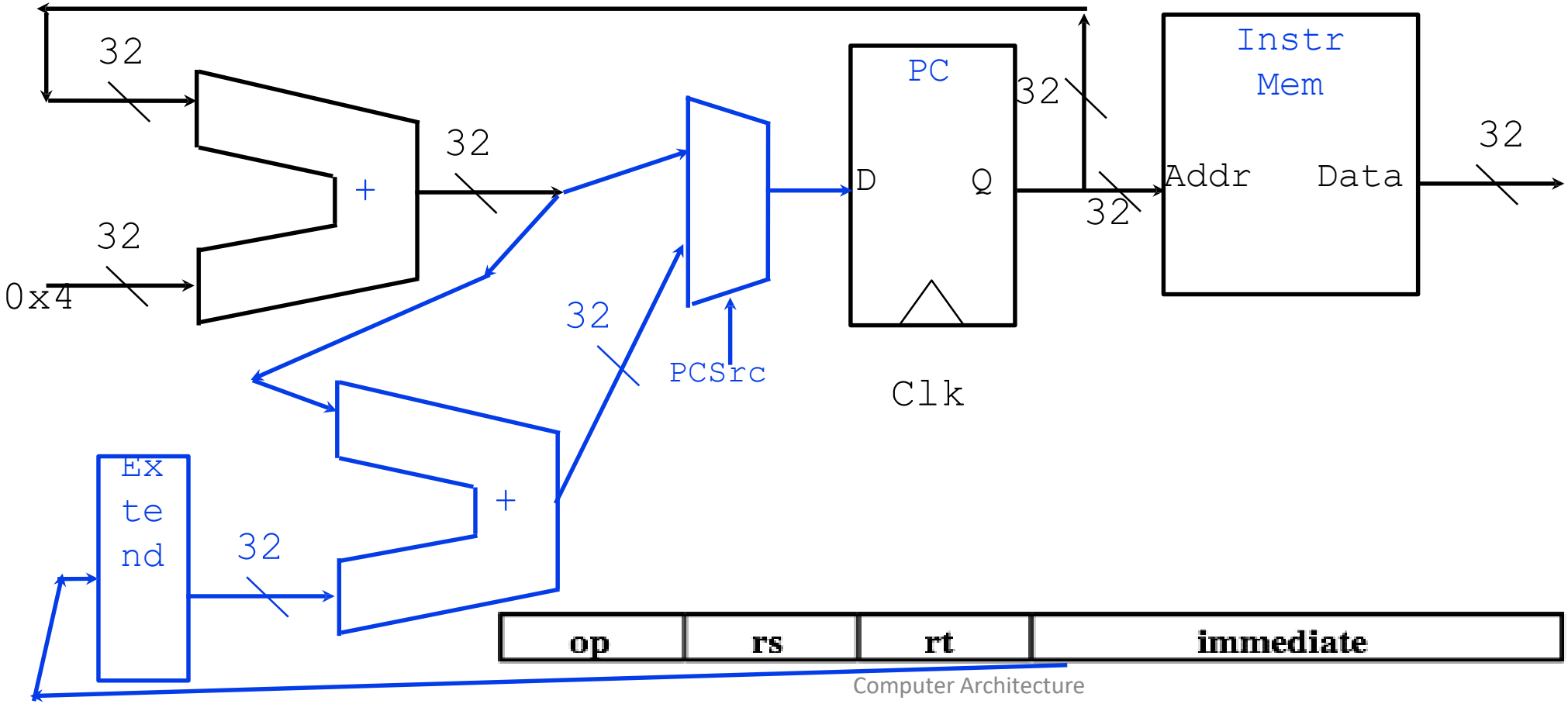


Syntax: LW $1, 32($2)

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

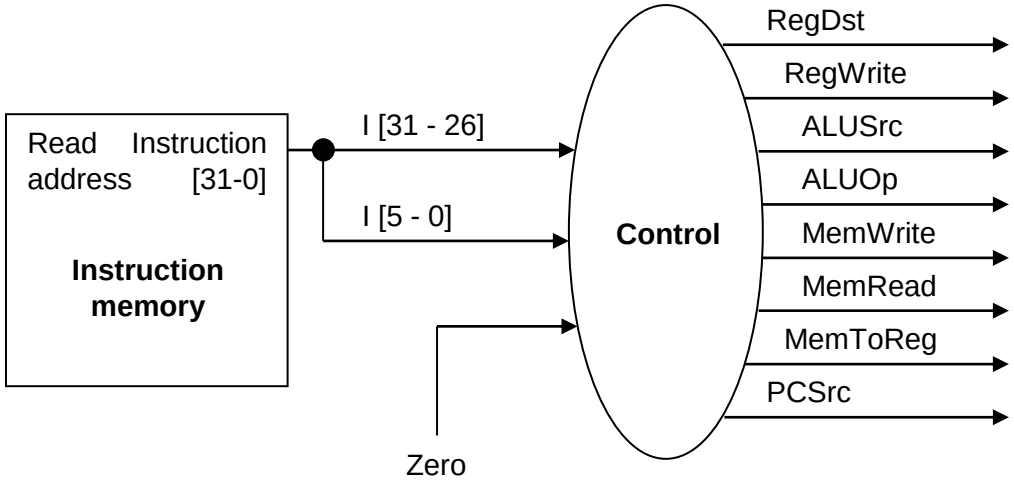
Branch Instructions

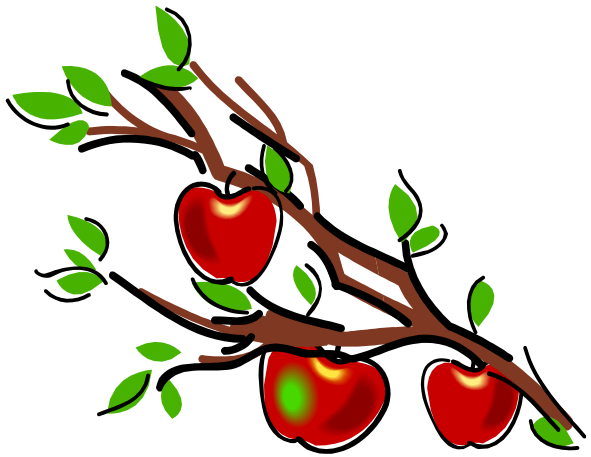
```
Syntax: BEQ $1, $2, 12
Action: If ($1 != $2), PC = PC + 4
Action: If ($1 == $2), PC = PC + 4 + 48
```



Control Signals So far

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





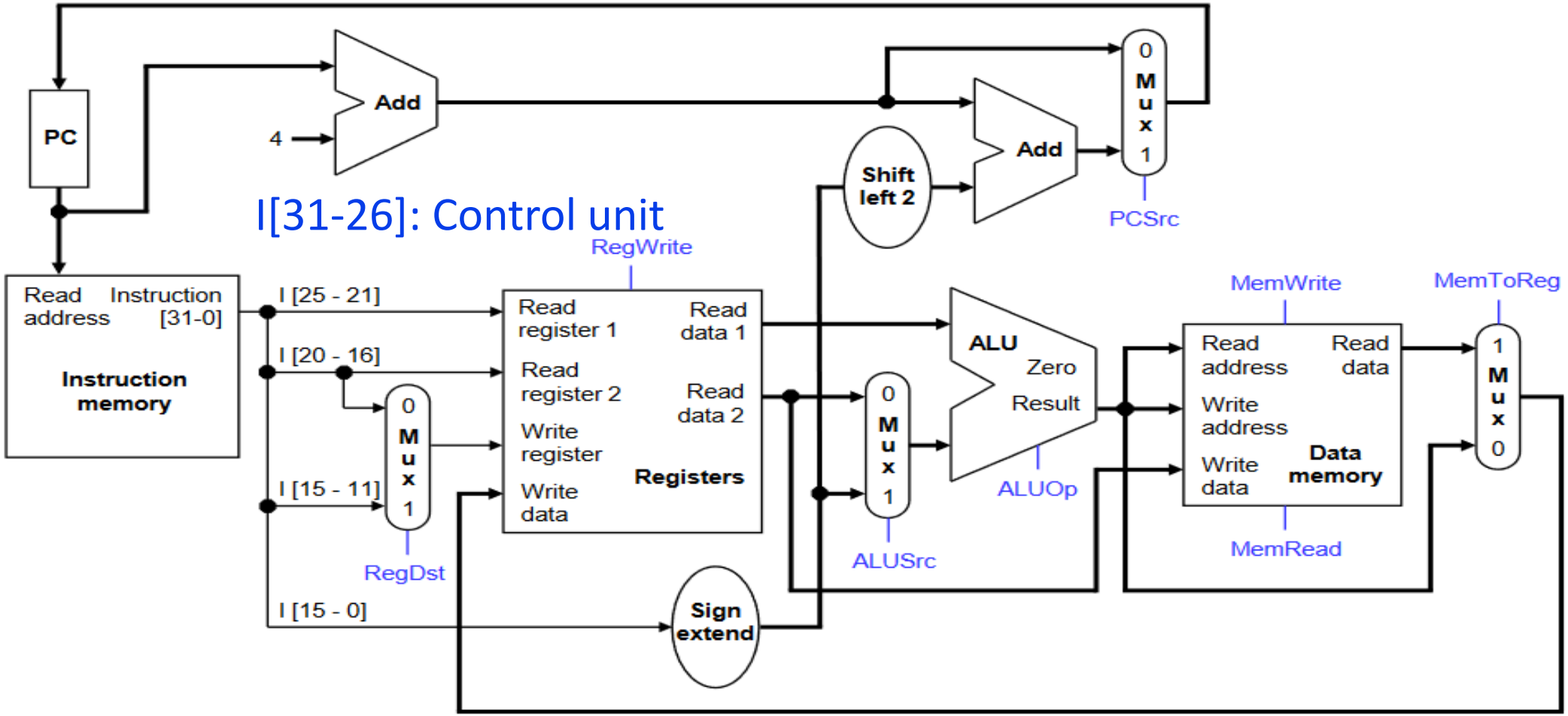
In detail

- 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?

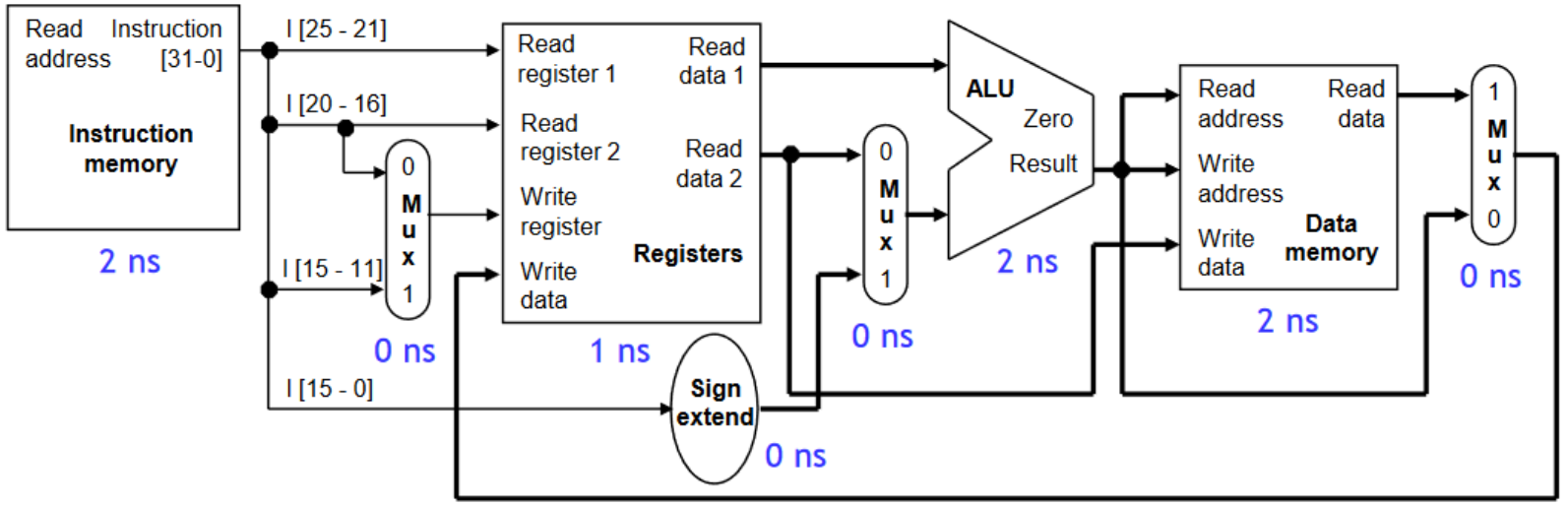
- For example, lw $t0, -4($sp) needs 8ns, assuming the delays shown here.

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



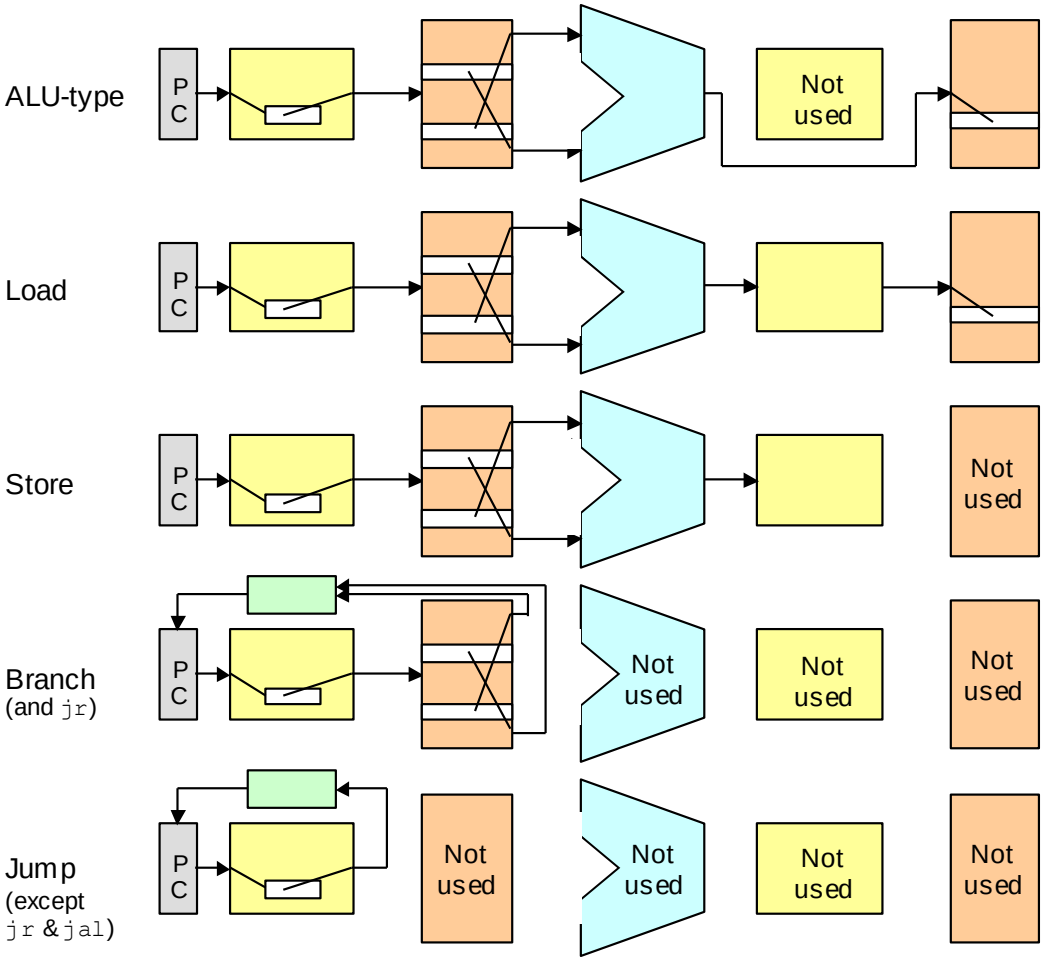
An add instruction: no need of 8ns

Why not single cycle?

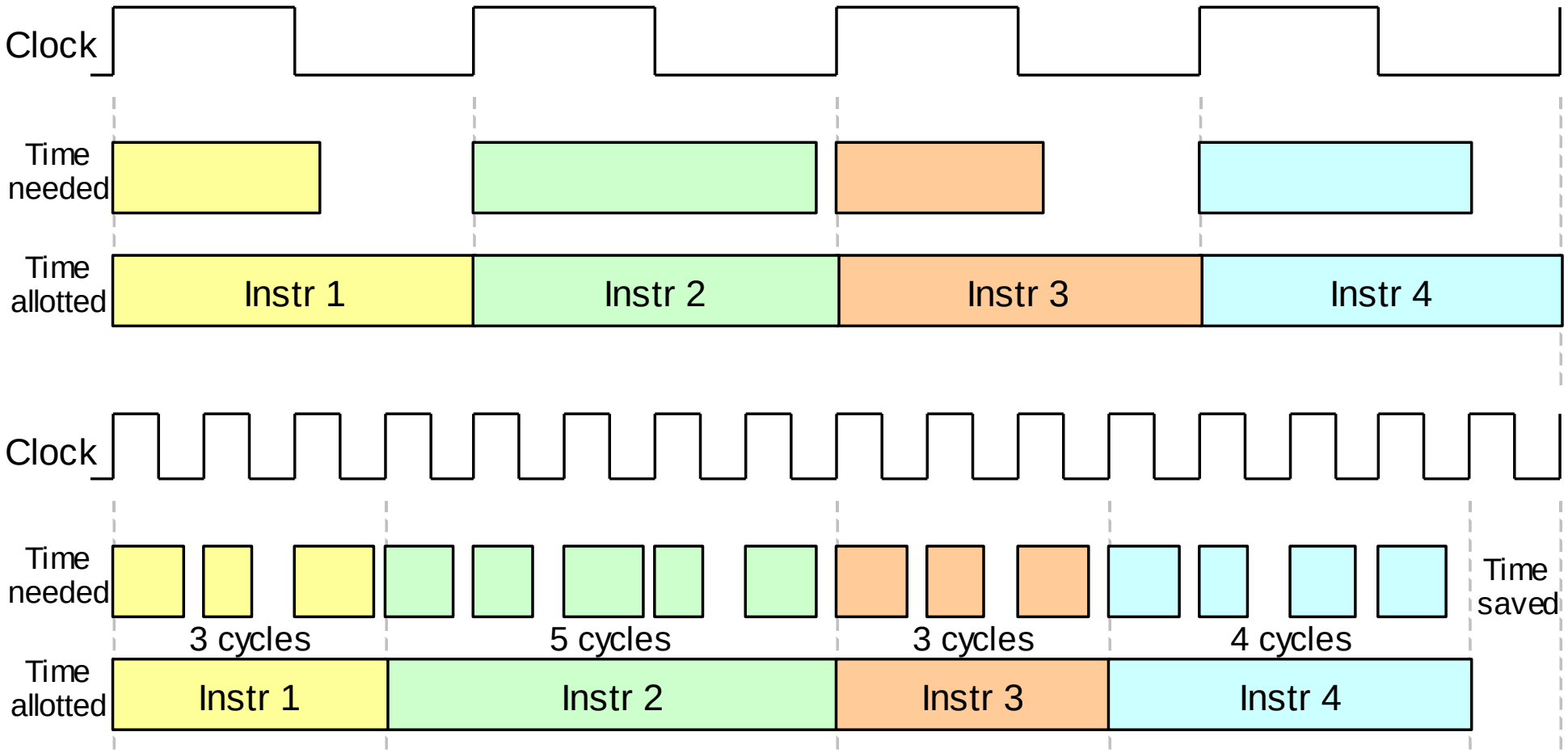
- The longest possible datapath is the clock cycle time.

Violating *common case fast (Confucius says)*

Oh No! Such a bad design



Single to Multi Cycle



que tenga un buen día