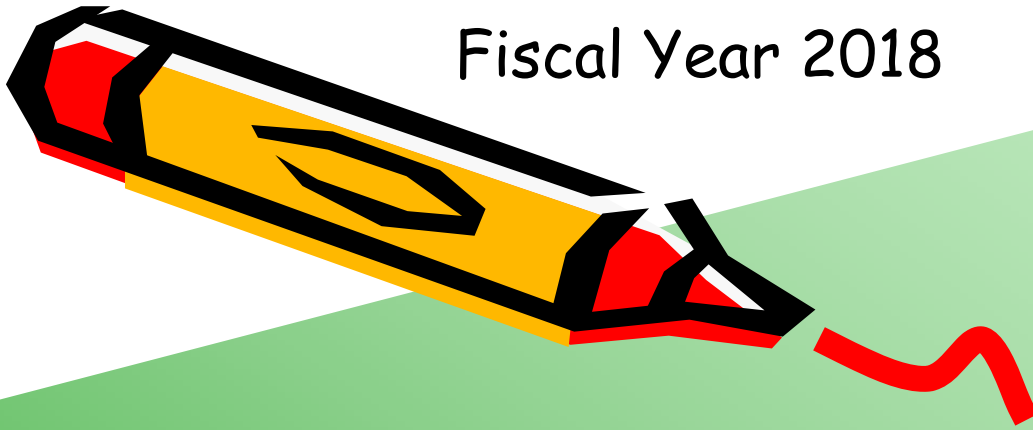


Fiscal Year 2018

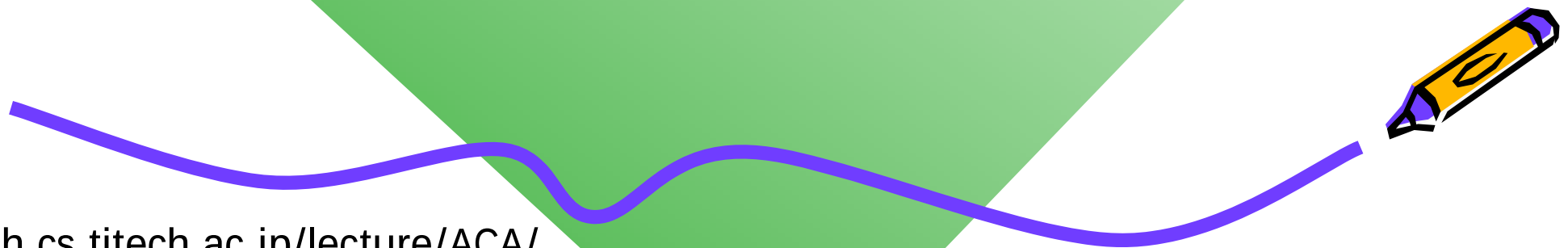
Ver. 2018-12-13a



Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

4. Pipelining



www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 13:20-14:50, Thr 13:20-14:50

Kenji Kise, Department of Computer Science
kise_at_c.titech.ac.jp

Signed Integer Representation

- Signed integer representation by **two's complement**
 - Negative numbers are represented by adding one after bit inversion of the positive numbers.
 - Eight-bit signed integers can range from -128 to 127

$$0000\ 0000_2 = 0_{10}$$

$$0000\ 0001_2 = +1_{10}$$

$$0000\ 0010_2 = +2_{10}$$

...

$$0111\ 1101_2 = +125_{10}$$

$$0111\ 1110_2 = +126_{10}$$

$$0111\ 1111_2 = +127_{10}$$

from 0 to 127
zero and positive numbers

$$1111\ 1111_2 = -0_{10}$$

$$1111\ 1110_2 = -1_{10}$$

$$1111\ 1101_2 = -2_{10}$$

...

$$1000\ 0010_2 = -125_{10}$$

$$1000\ 0001_2 = -126_{10}$$

$$1000\ 0000_2 = -127_{10}$$

Bit inversion of 0~127
(one's complement)

$$\underline{1111\ 1111_2} = -1_{10}$$

$$1111\ 1110_2 = -2_{10}$$

...

$$1000\ 0011_2 = -125_{10}$$

$$1000\ 0010_2 = -126_{10}$$

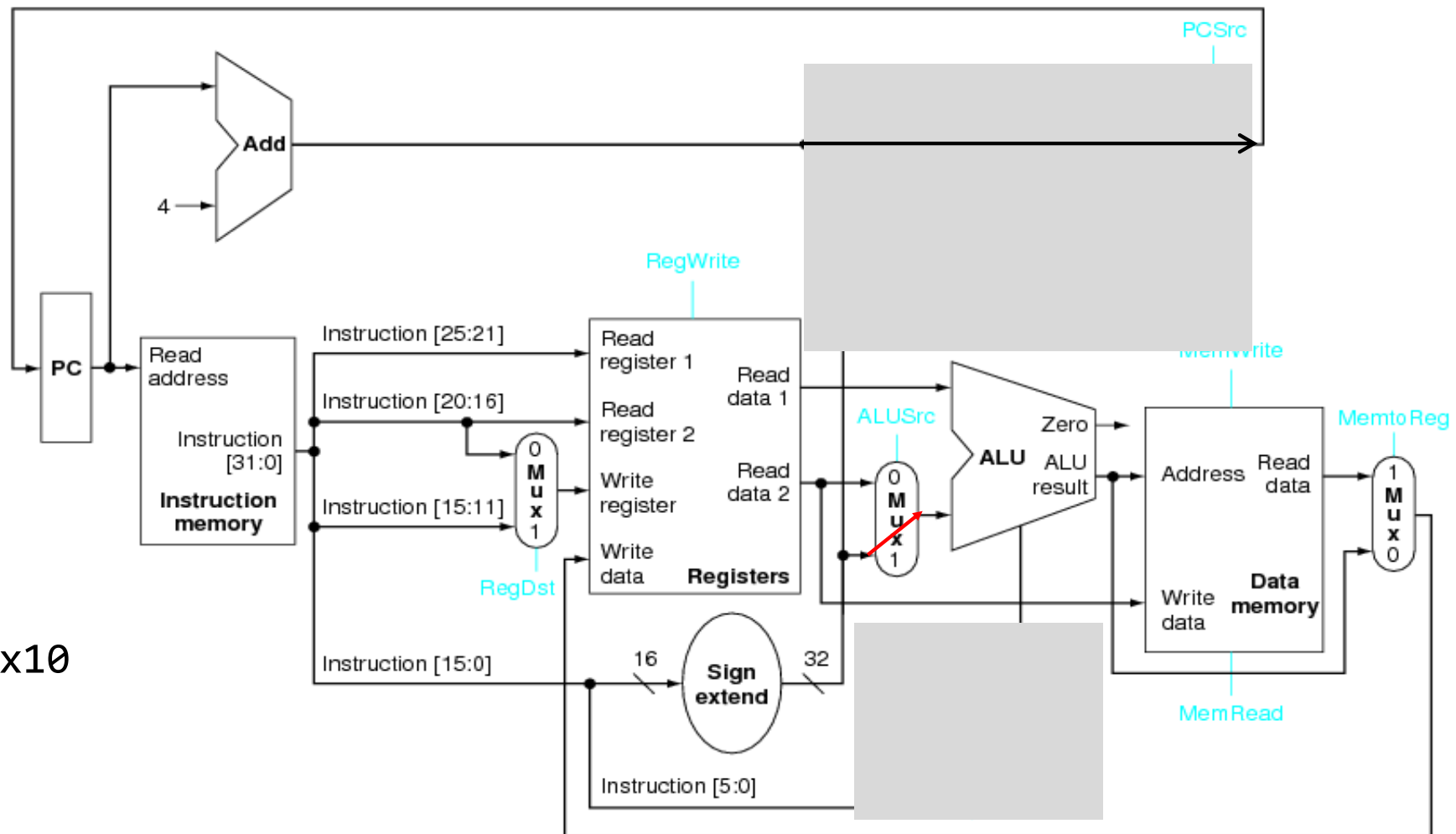
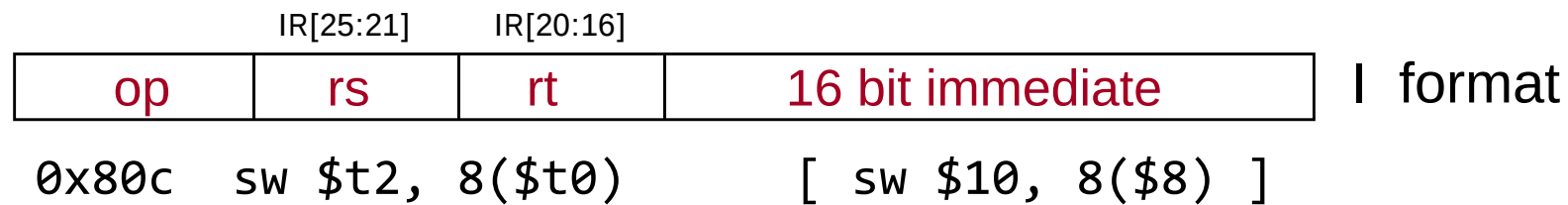
$$1000\ 0001_2 = -127_{10}$$

$$1000\ 0000_2 = -128_{10}$$

Adding 1 to the bit inversion
will be the negative number
(two's complement)
from -128 to -1



Datapath of processor supporting **ADD, ADDI, LW, SW**



\$8 = 0x10
\$10 = 2

MIPS Control Flow Instructions

- MIPS **conditional branch** instructions:

bne \$s0, \$s1, Lbl # go to Lbl if \$s0≠\$s1

beq \$s0, \$s1, Lbl # go to Lbl if \$s0=\$s1

- Ex: **if (i==j) h = i + j;**

bne \$s0, \$s1, Lbl1

add \$s3, \$s0, \$s1

Lbl1: ...

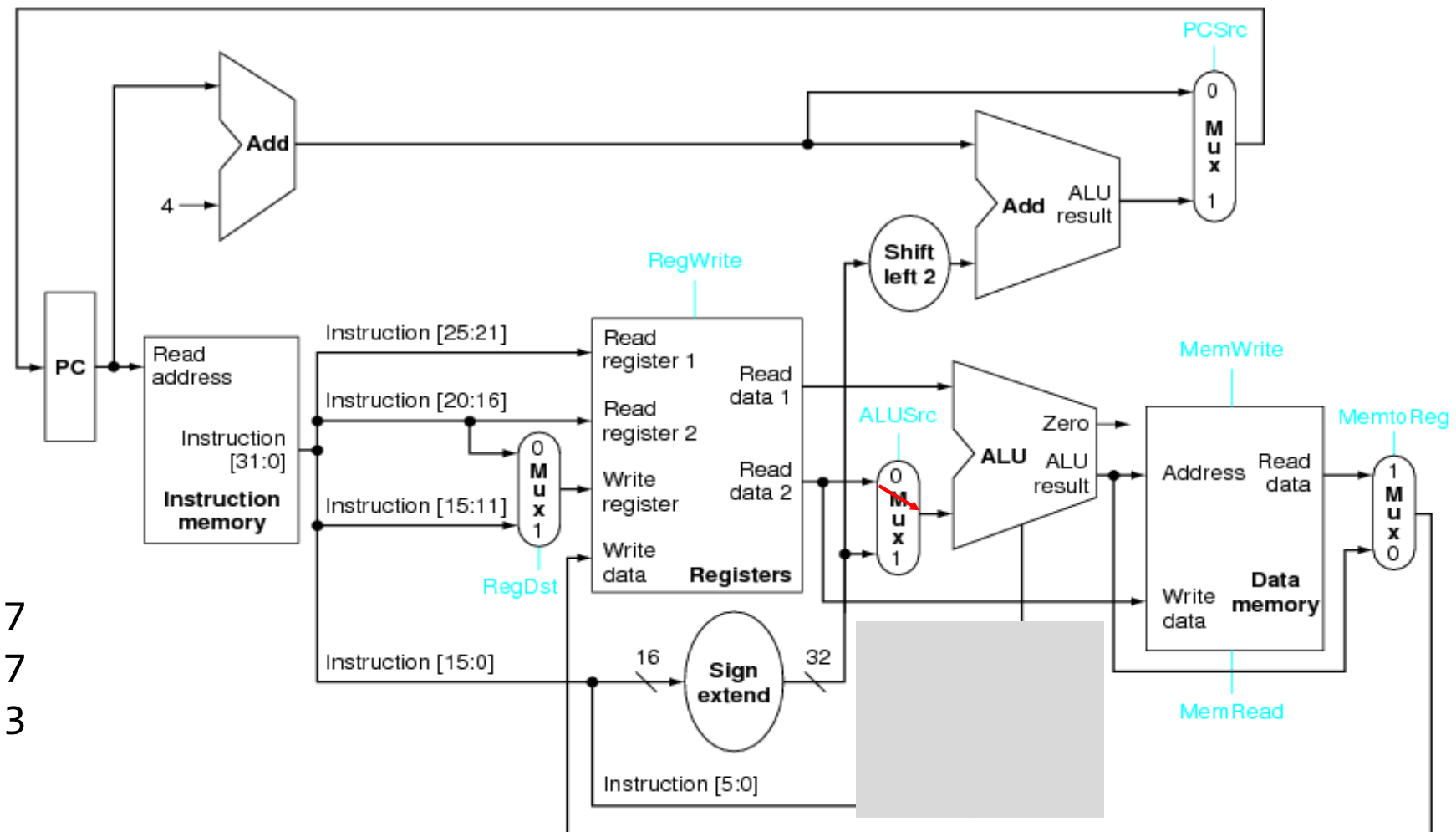
- Instruction Format (**I** format):



- How is the branch destination address specified?



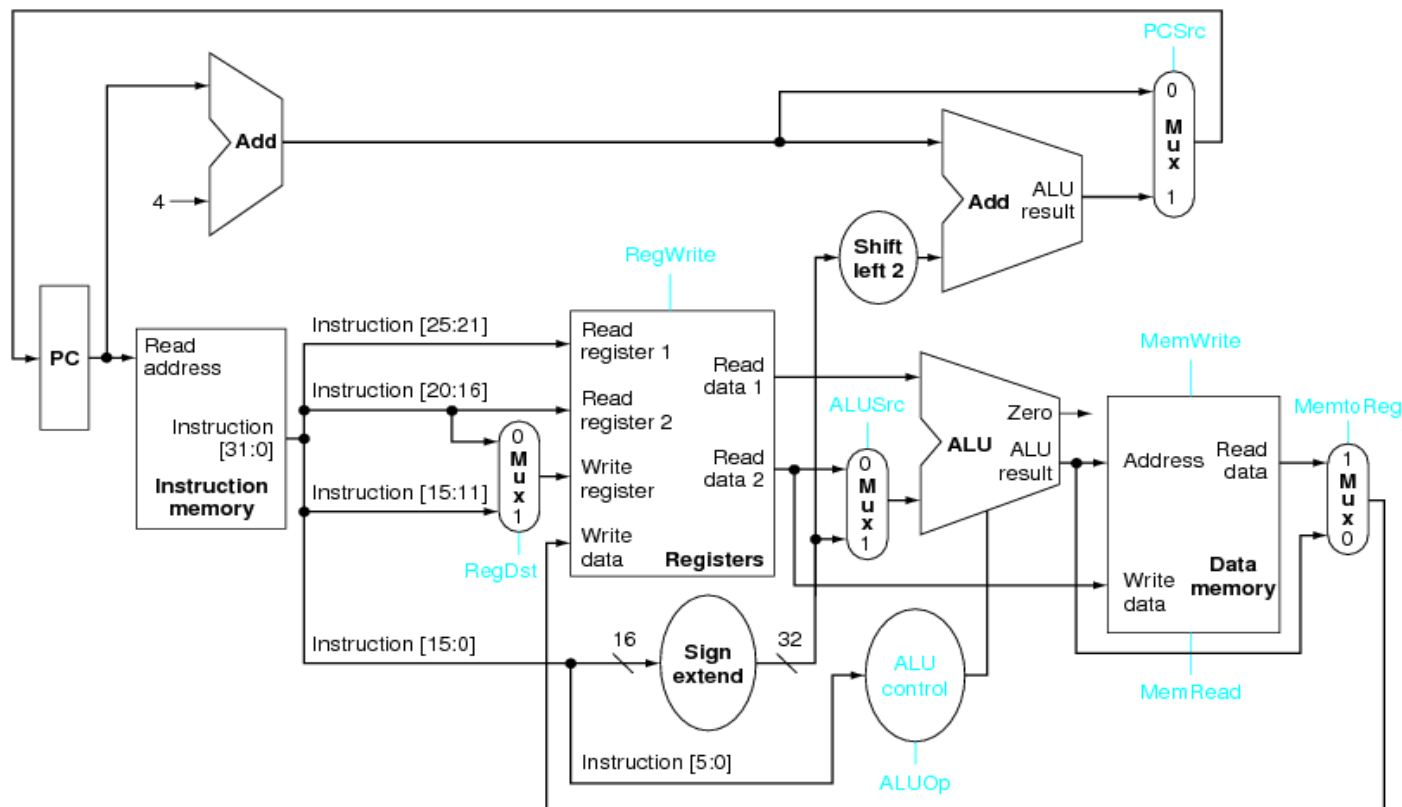
Datapath of processor supporting **ADD, ADDI, LW, SW, BNE, BEQ**



\$8 = 7
\$9 = 7
imm = -3

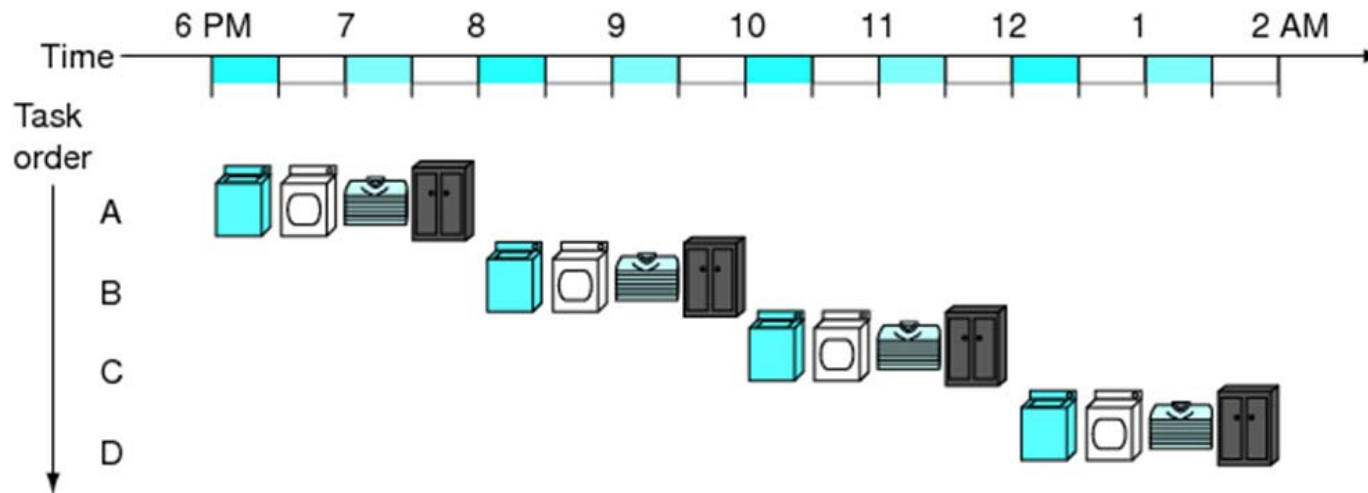
Single-cycle implementation of processors

- Single-cycle implementation also called **single clock cycle implementation** is the implementation in which an instruction is executed in one clock cycle. While easy to understand, it is too slow to be practical.



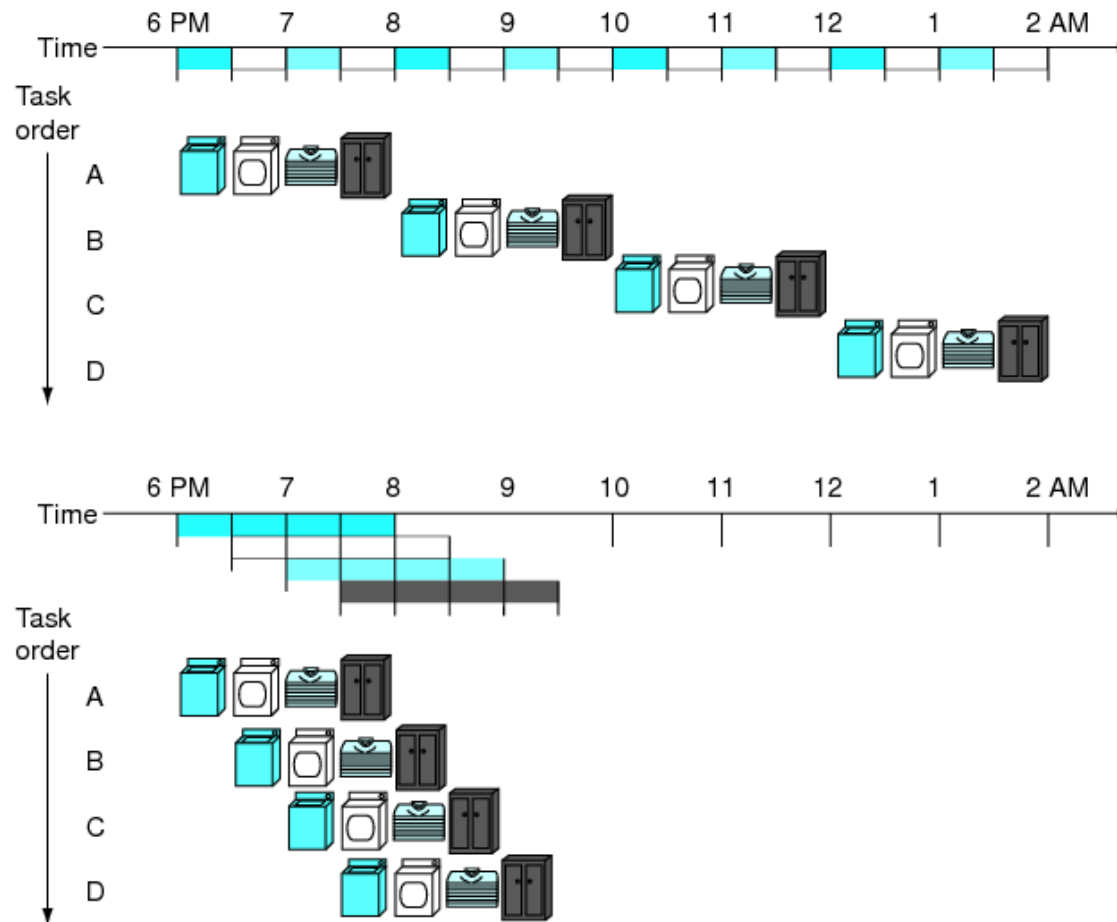
Single-cycle implementation of laundry

- (A) Ann, (B) Brian, (C) Cathy, and (D) Don each have dirty clothes to be *washed, dried, folded, and put away* where each takes 30 minutes.
- Cycle time is 2 hours.
- Sequential laundry takes 8 hours for 4 loads.



Single-cycle implementation and **pipelining**

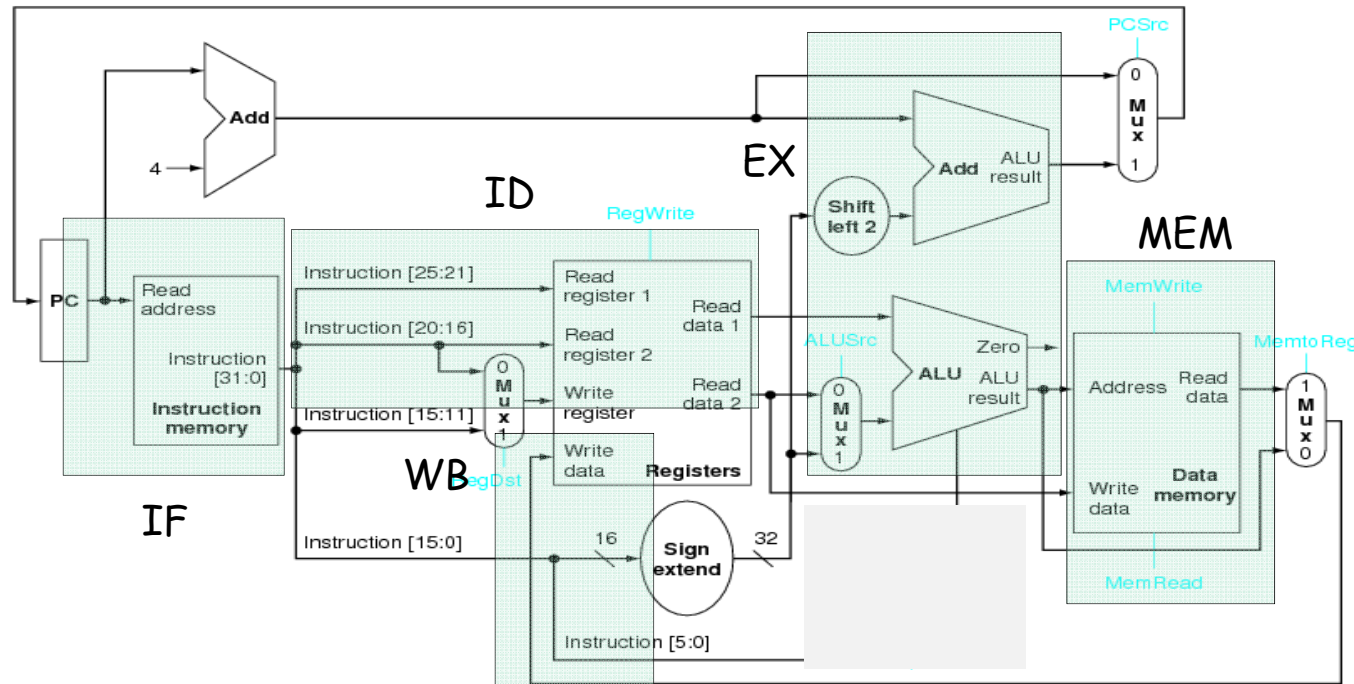
- Pipelined laundry takes 3.5 hours just using the same hardware resources. Cycle time is 30 minutes.
- What is the latency of each load?





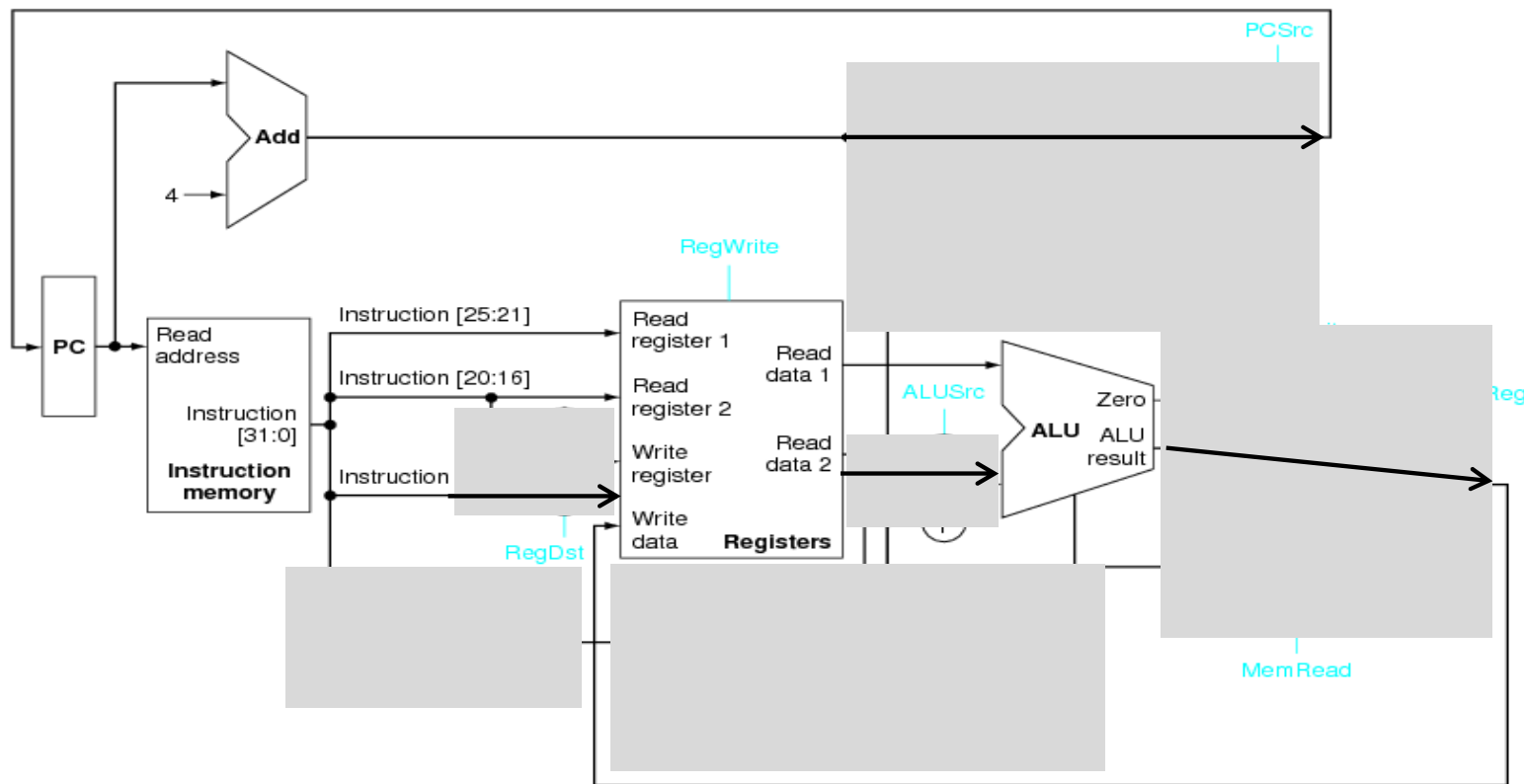
Conventional five steps (stages) of MIPS

- **IF**: Instruction Fetch from instruction memory
- **ID**: Instruction Decode and operand fetch from regfile (register file)
- **EX**: EXecute operation or calculate address for load/store or calculate branch condition and target address
- **MEM** (MA): MEMory access for load/store
- **WB**: Write result Back to regfile



Towards four stage pipelined one supporting ADD

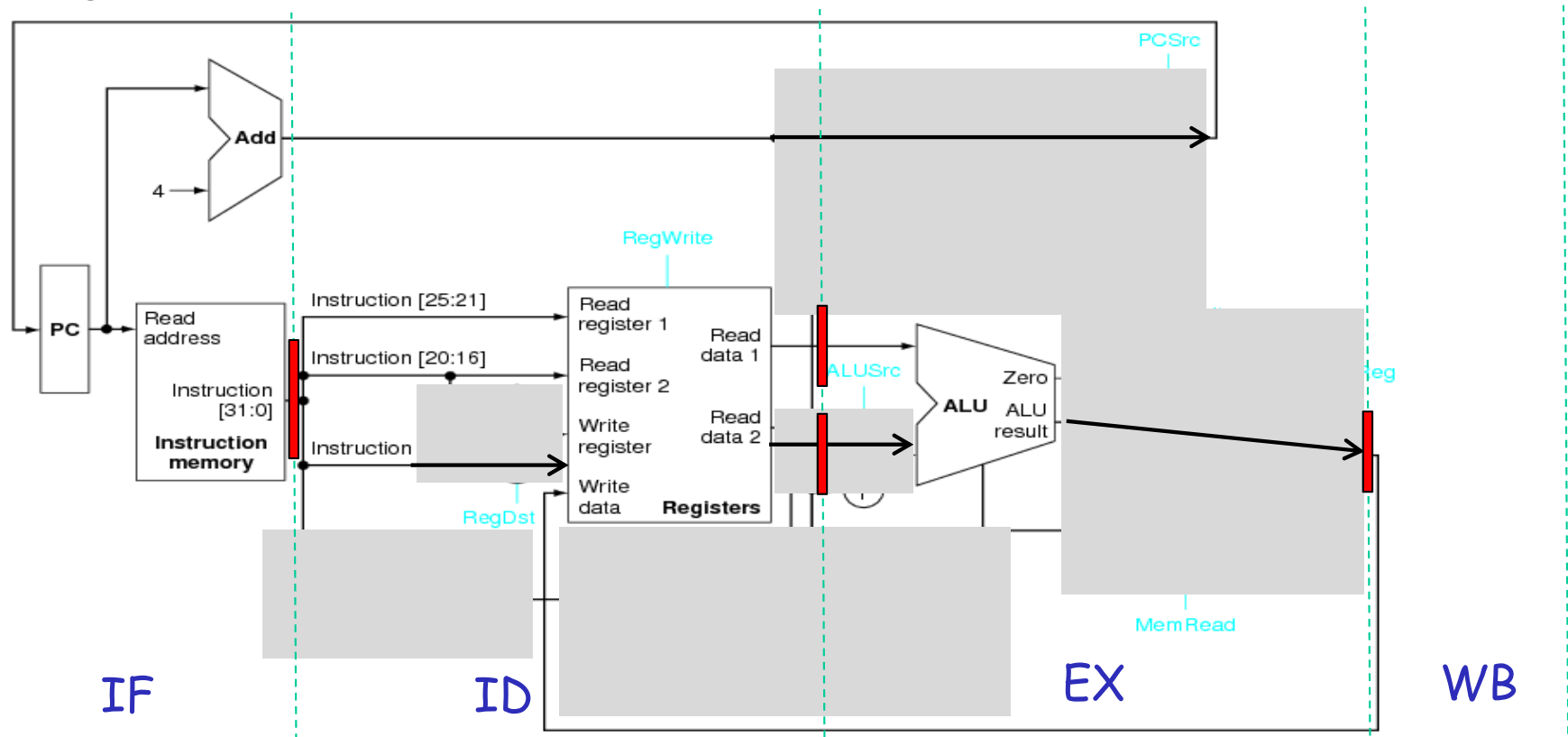
- **IF**: Instruction Fetch from instruction memory
- **ID**: Instruction Decode and operand fetch from regfile
- **EX**: EXecute operation
- **WB**: Write result Back to regfile



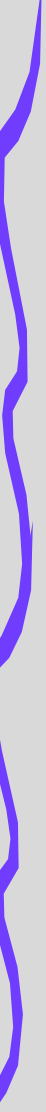
Pipeline registers

- add \$0, \$0, \$0 # NOP, $\$0 \leq 0 + 0$
- add \$1, \$1, \$1 # $\$1 \leq 22 + 22$
- add \$2, \$2, \$2 # $\$2 \leq 33 + 33$
- add \$0, \$0, \$0 # NOP
- add \$0, \$0, \$0 # NOP
- add \$0, \$0, \$0 # NOP

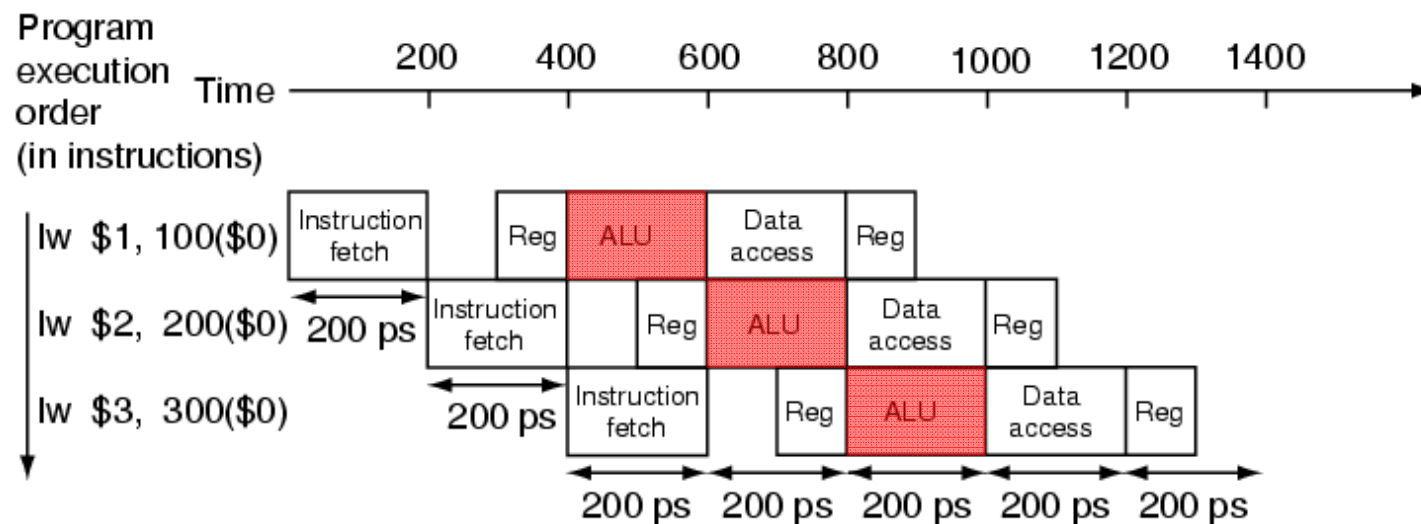
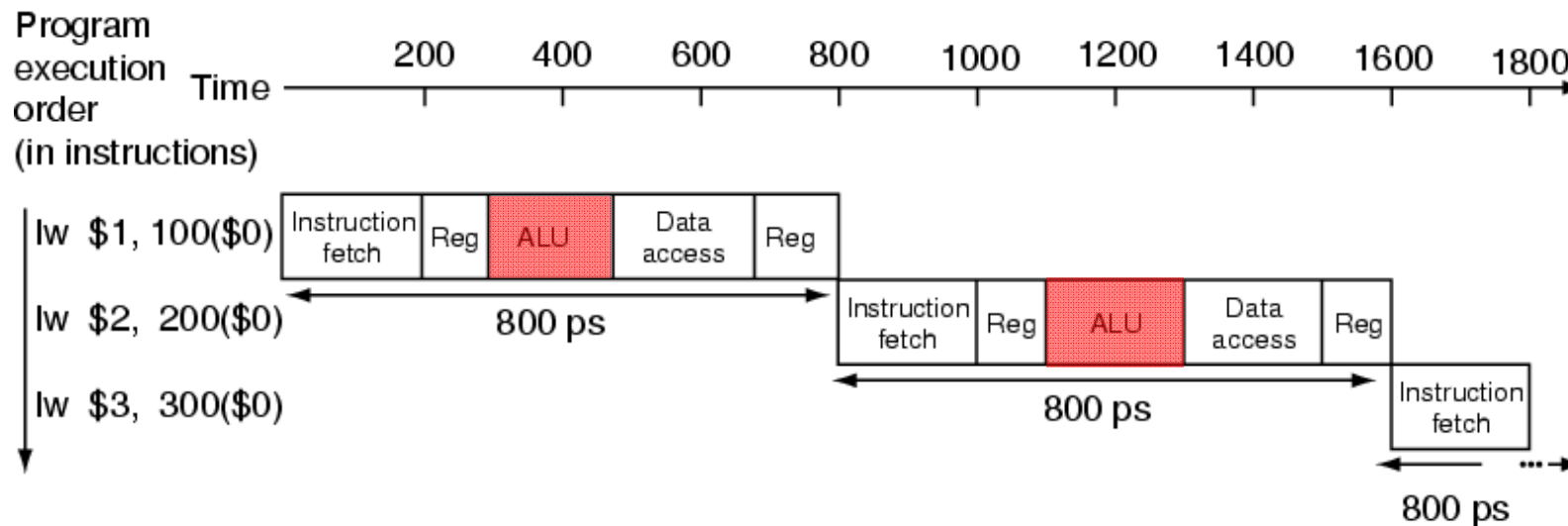
assuming initial values of $r[1]=22$ and $r[2]=33$



Exercise: Correcting the pipeline datapath



Single-cycle and pipelined processors

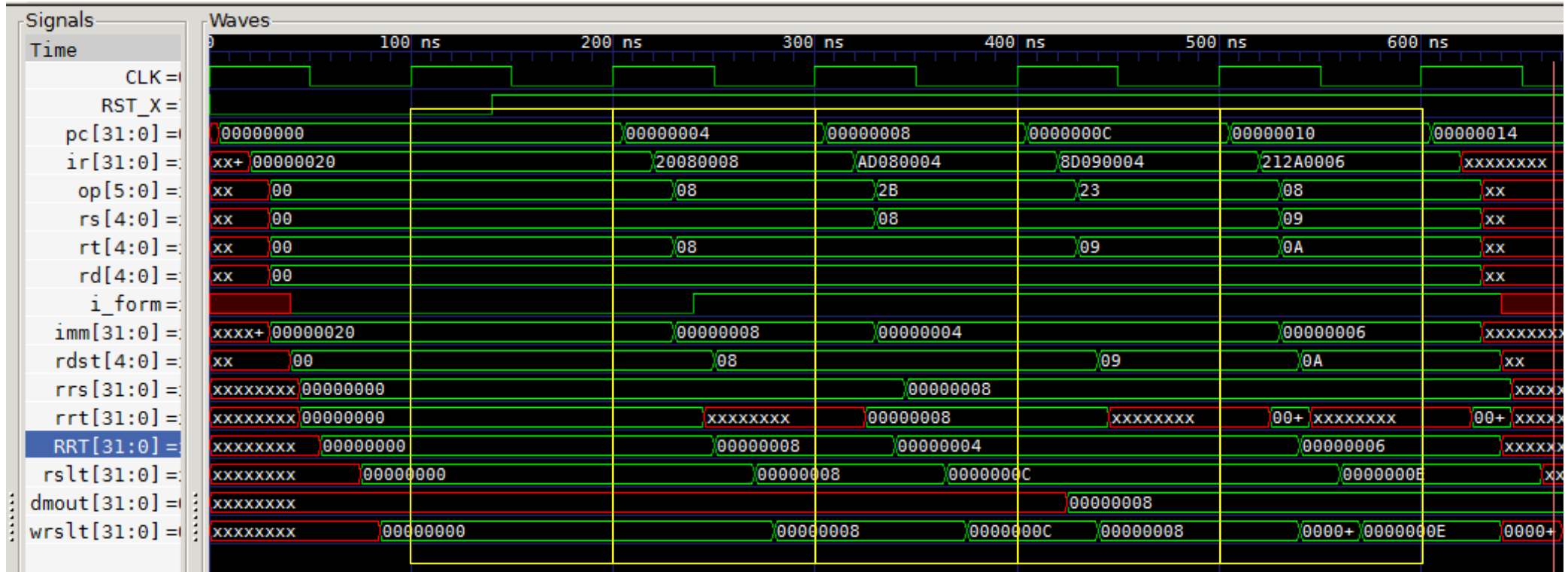


Homework 4

1. Design a four stage pipelined processor supporting MIPS `add` instructions in Verilog HDL. Please download `proc01.v` from the support page and refer it.
2. Verify the behavior of designed processor using following assembly code assuming initial values of `r[1]=22`, `r[2]=33`, `r[3]=44`, and `r[4]=55`
 - `add $0, $0, $0 # NOP {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20}`
 - `add $1, $1, $1 #`
 - `add $2, $2, $2 #`
 - `add $3, $3, $3 #`
 - `add $4, $4, $4 #`
3. Submit **a report printed on A4 paper** at the beginning of the next lecture.
 - The report should include a block diagram, a source code in Verilog HDL, and obtained waveforms of your design.



Waveform of proc04 (Homework 3)



- add \$0, \$0, \$0 # NOP {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20}
- addi \$t0, \$zero, 8 # {6'h8, 5'd0, 5'd8, 16'd8} \$t0 = 8
- sw \$t0, 4(\$t0) # {6'h2b, 5'd8, 5'd8, 16'd4} mem[12] = 8
- lw \$t1, 4(\$t0) # {6'h23, 5'd8, 5'd9, 16'd4} \$t1 = mem[12] = 8
- addi \$t2, \$t1, 6 # {6'h8, 5'd9, 5'd10, 16'd6} \$t2 = 8 + 6