Fiscal Year 2018

Ver. 2018-12-12a

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

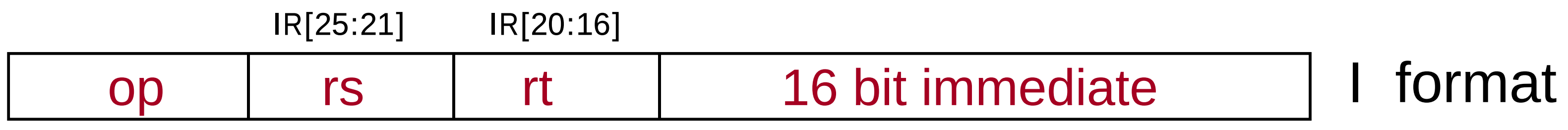
Advanced Computer Architecture

4. Pipelining

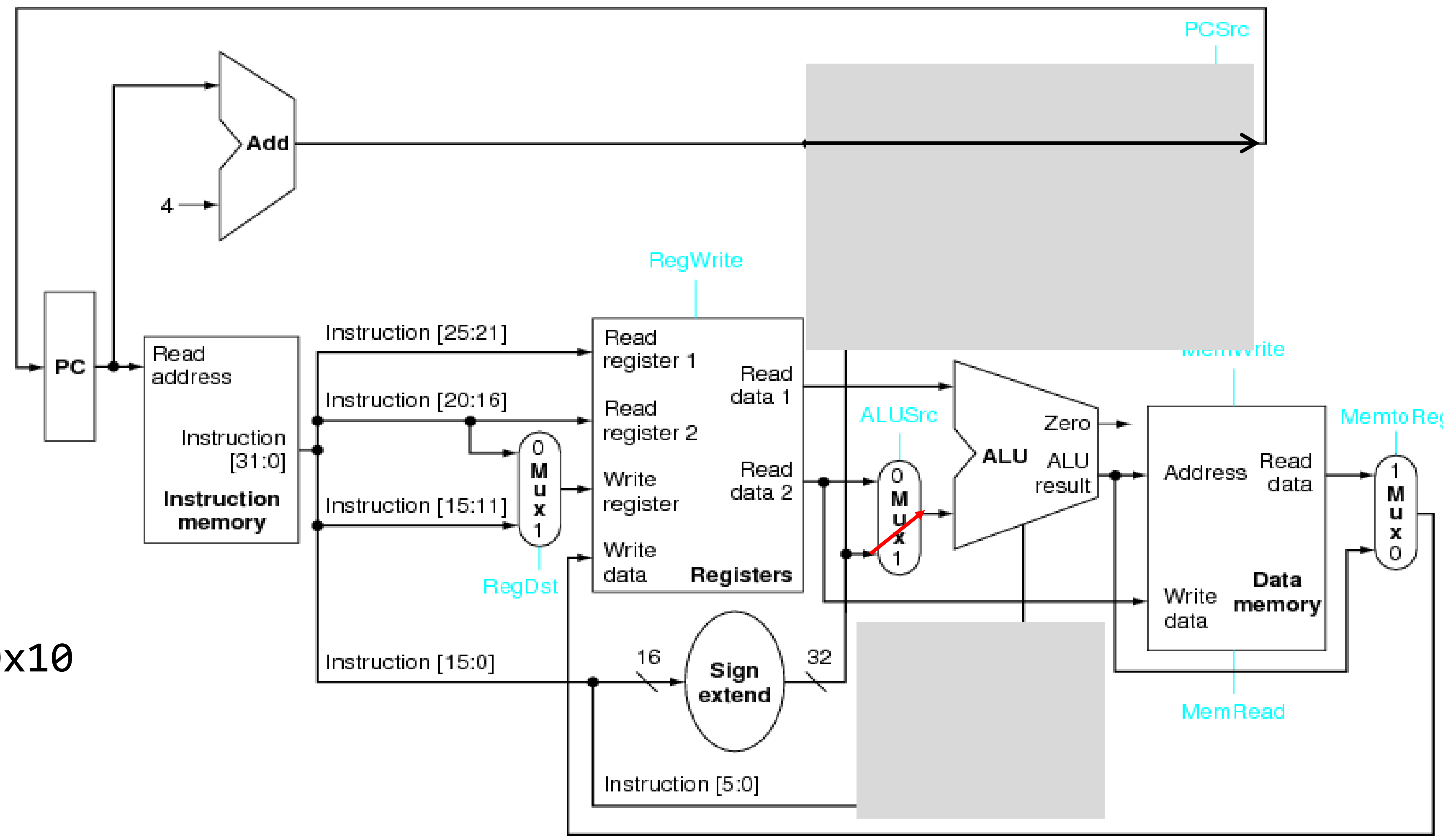
www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W936
Mon 13:20-14:50, Thr 13:20-14:50

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Datapath of processor supporting ADD, ADDI, LW, SW



```
0x80c   sw $t2, 8($t0)       [ sw $10, 8($8) ]
```



```
$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):

op	rs	rt	16 bit offset

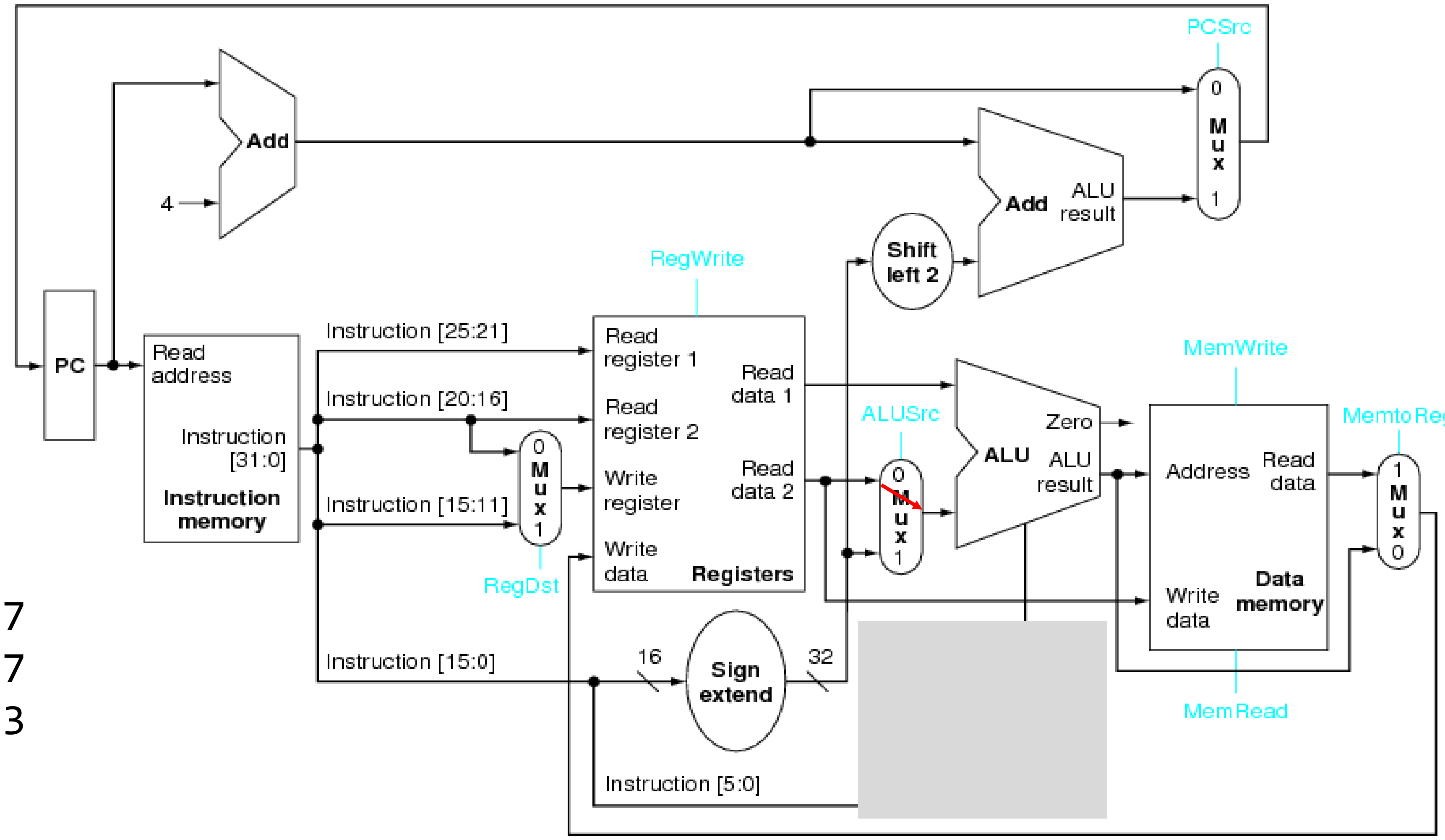
- How is the branch destination address specified?

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

	IR[25:21]	IR[20:16]	
op	rs	rt	16 bit immediate

I format

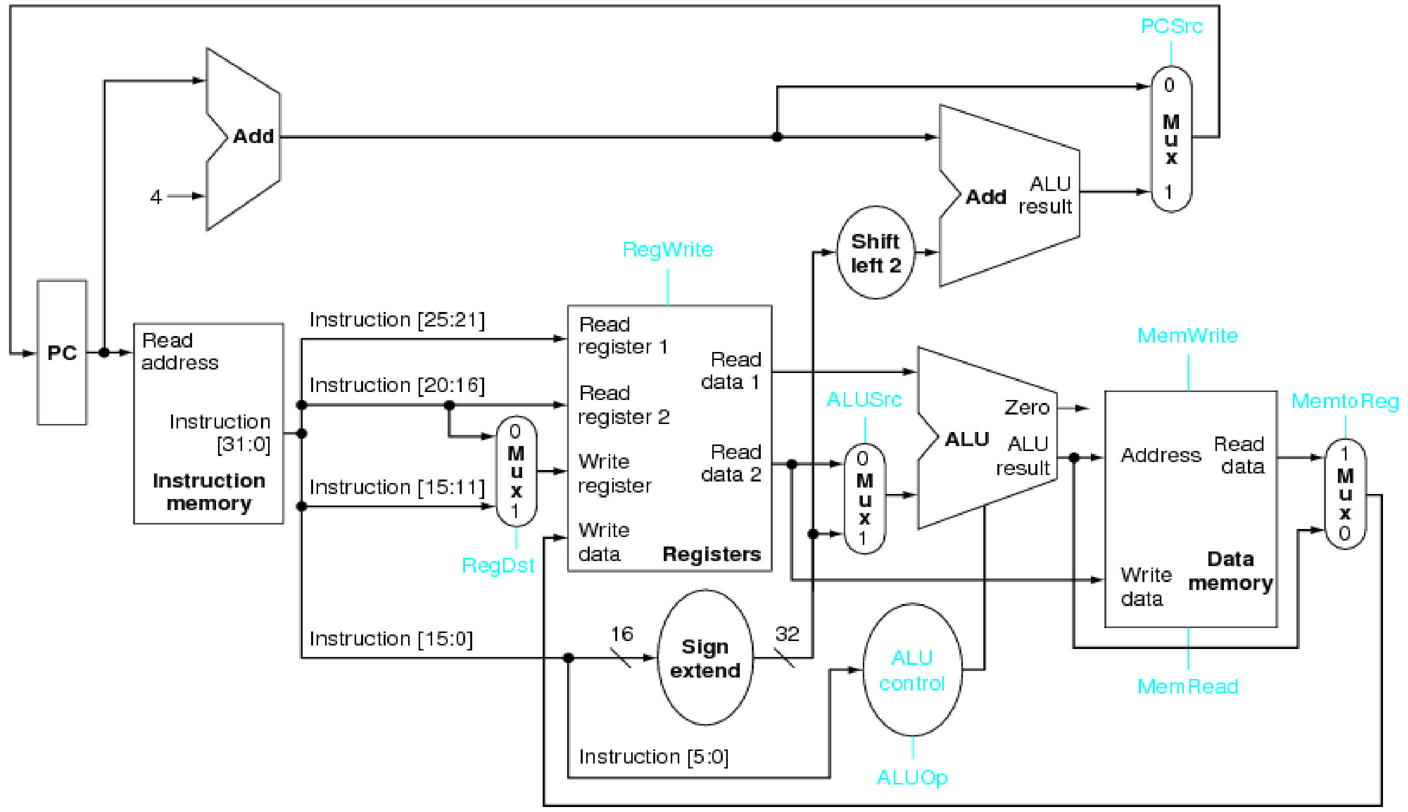
```
0x810  beq $t0, $t1, Lb      [ beq $8, $9, Lb ]
```



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

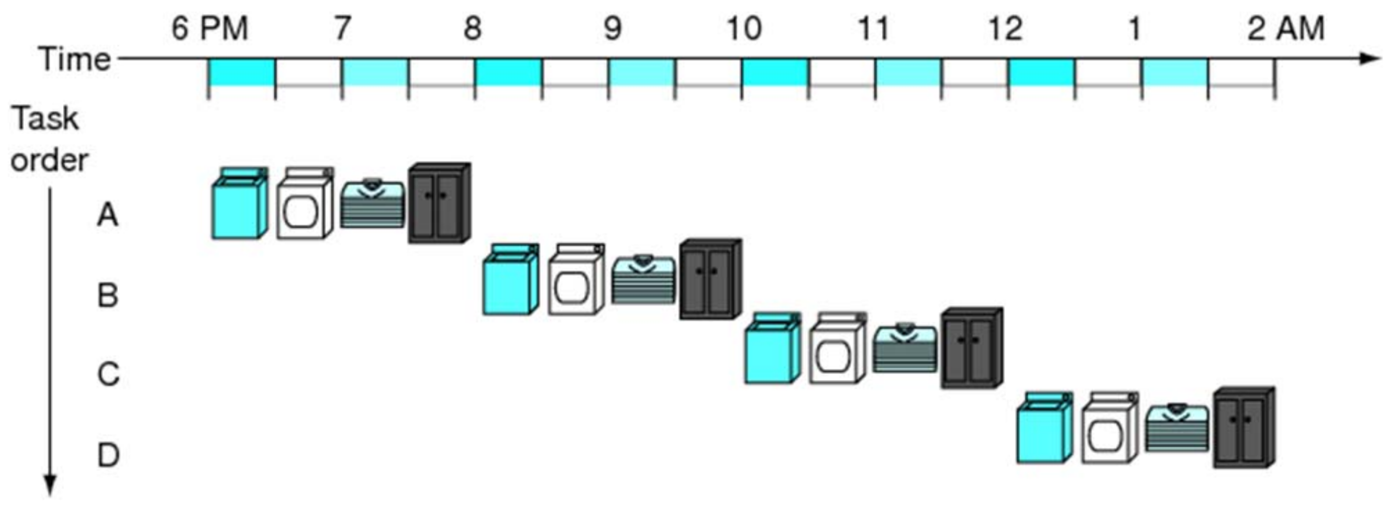
Single-cycle implementation of processors

- 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.



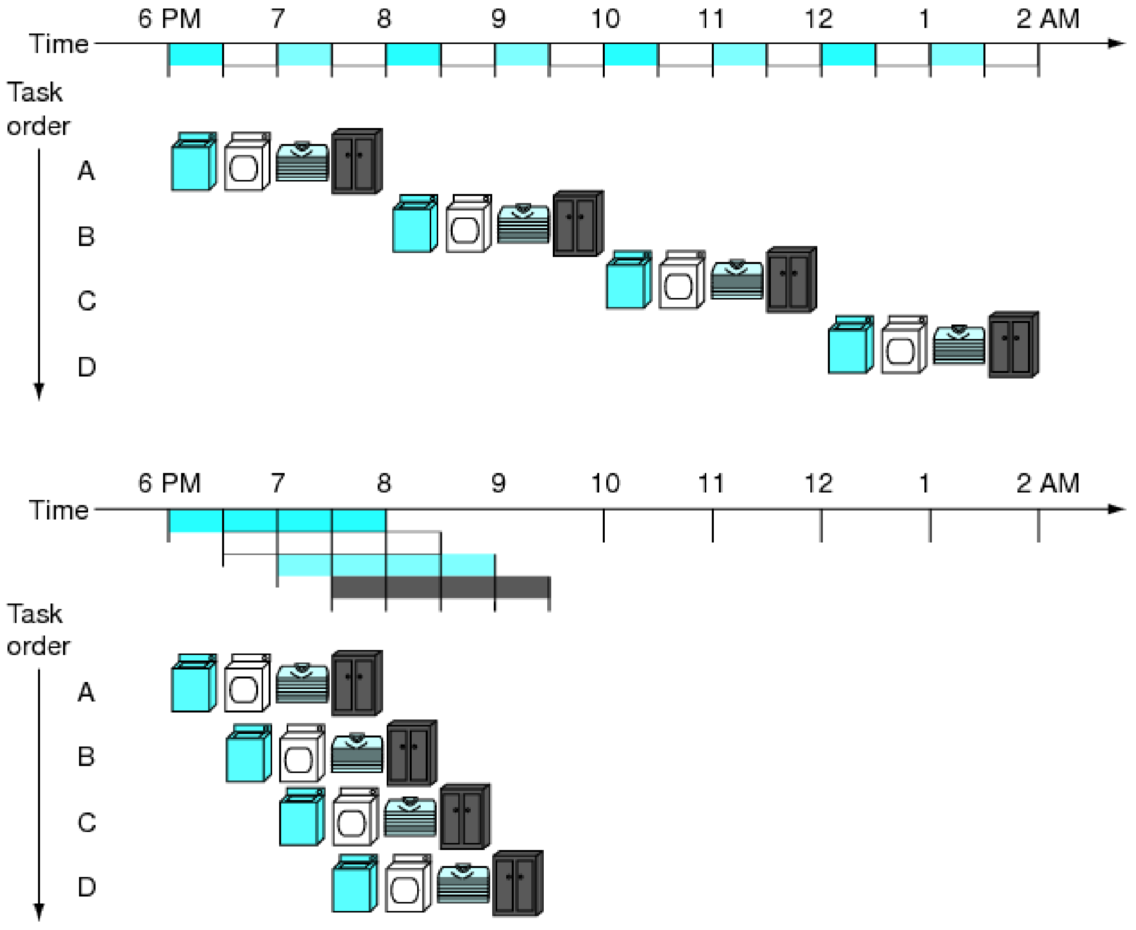
Single-cycle implementation of laundry

- (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.

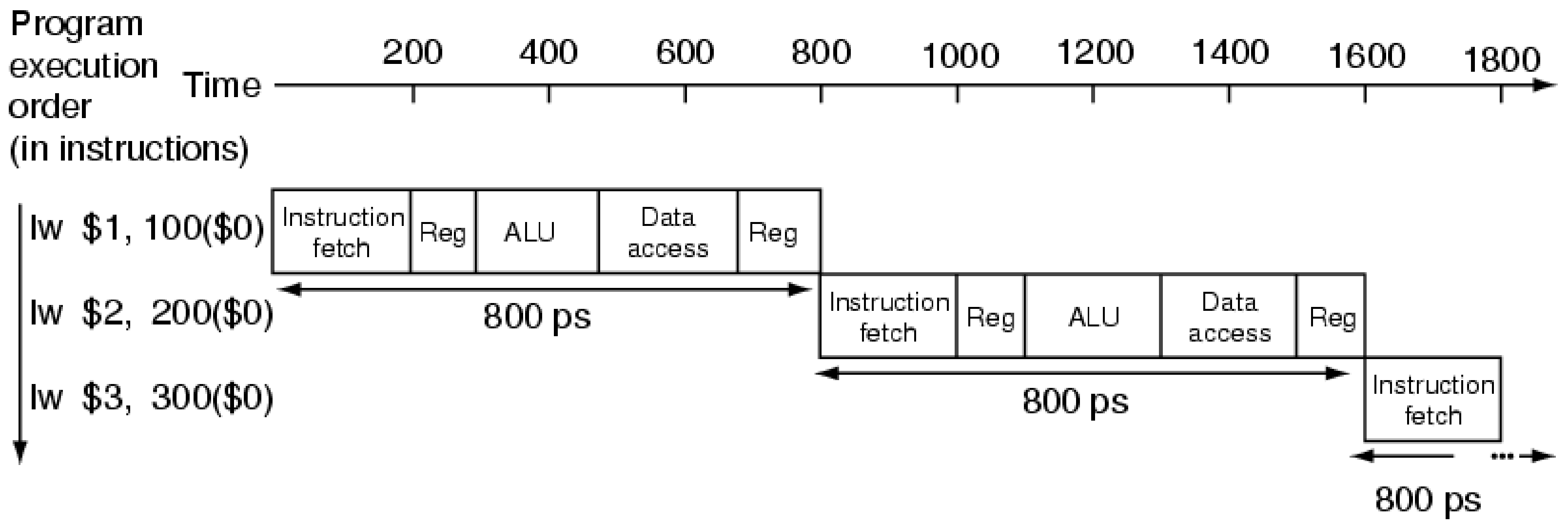


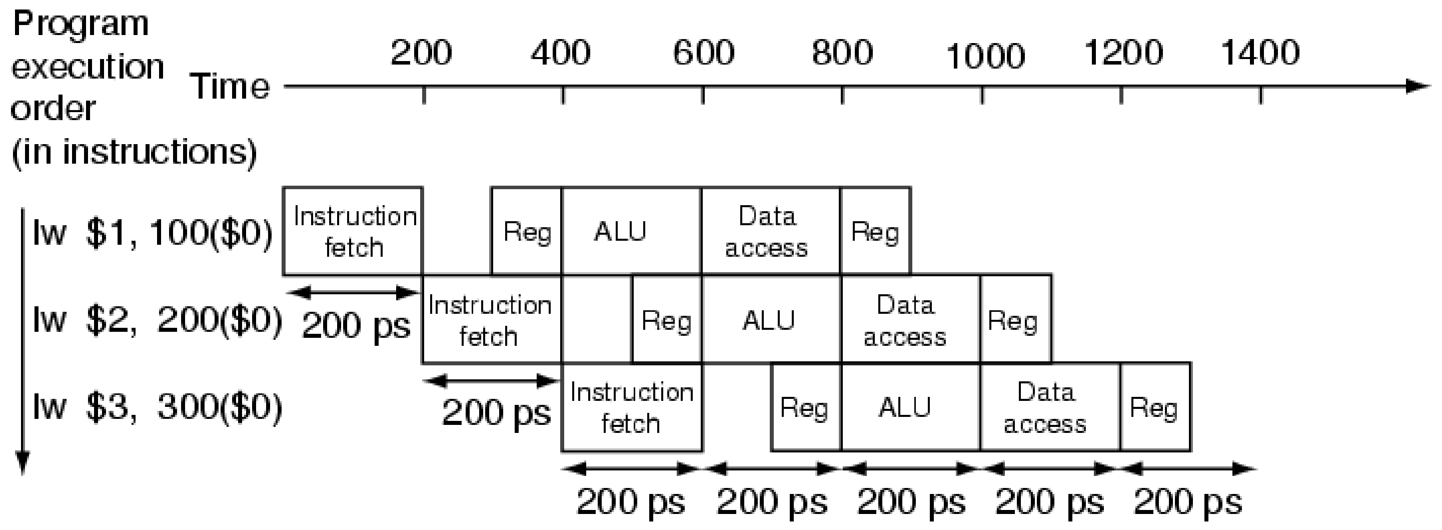
Single-cycle implementation and pipelining

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



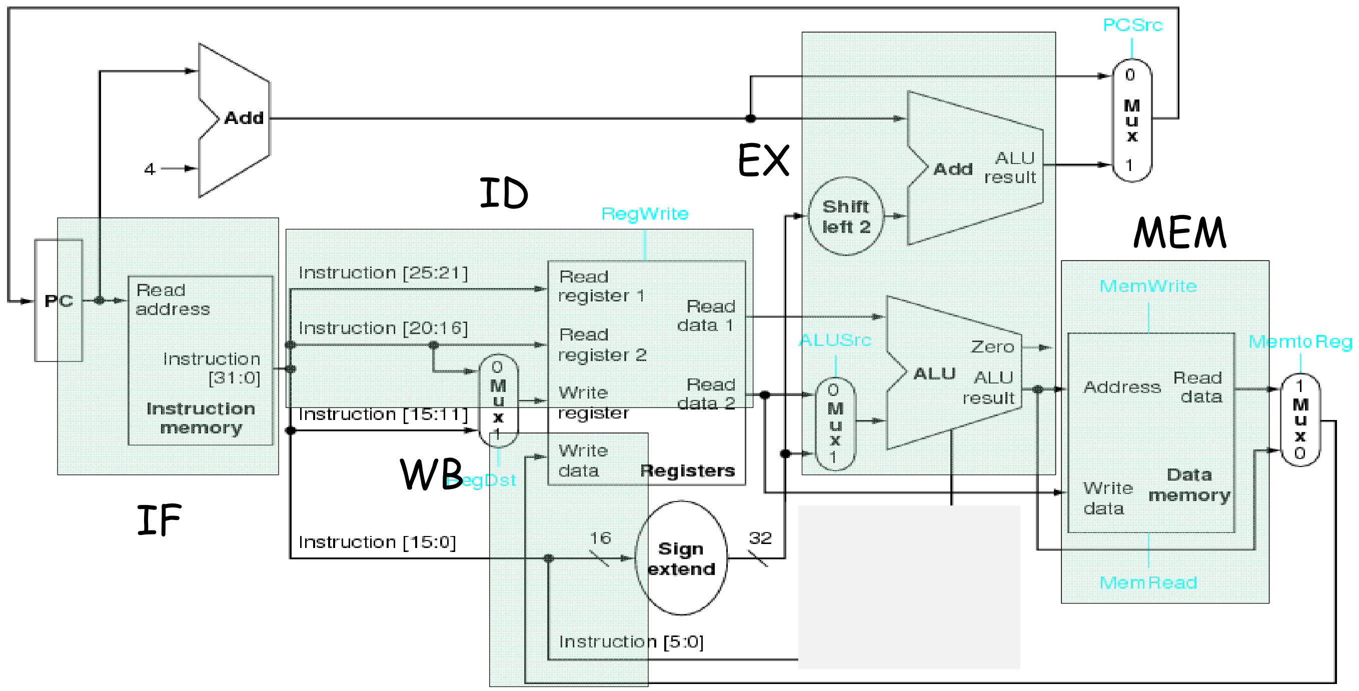
Single-cycle and pipelined processors





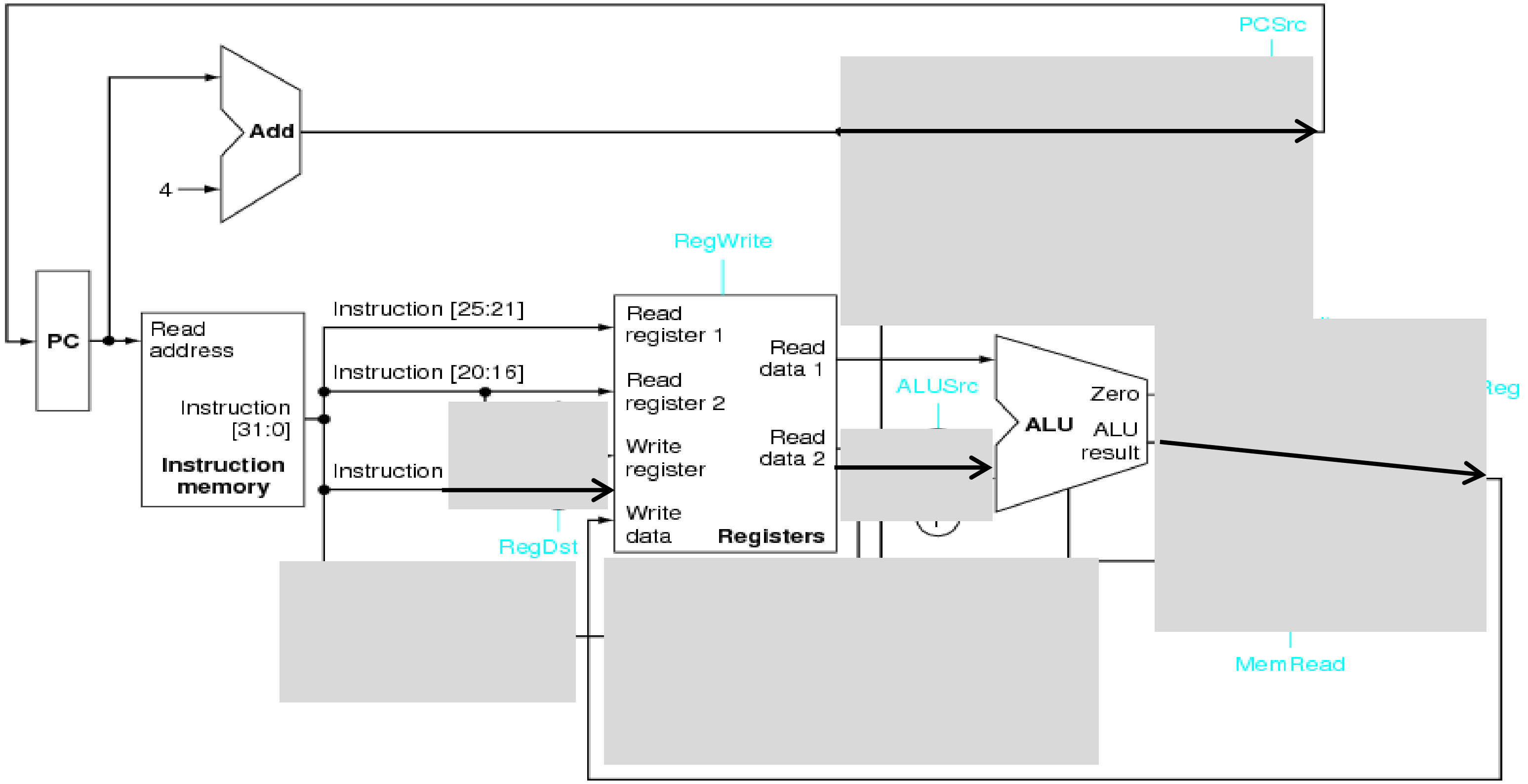
Conventional five steps (stages) of MIPS

- 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

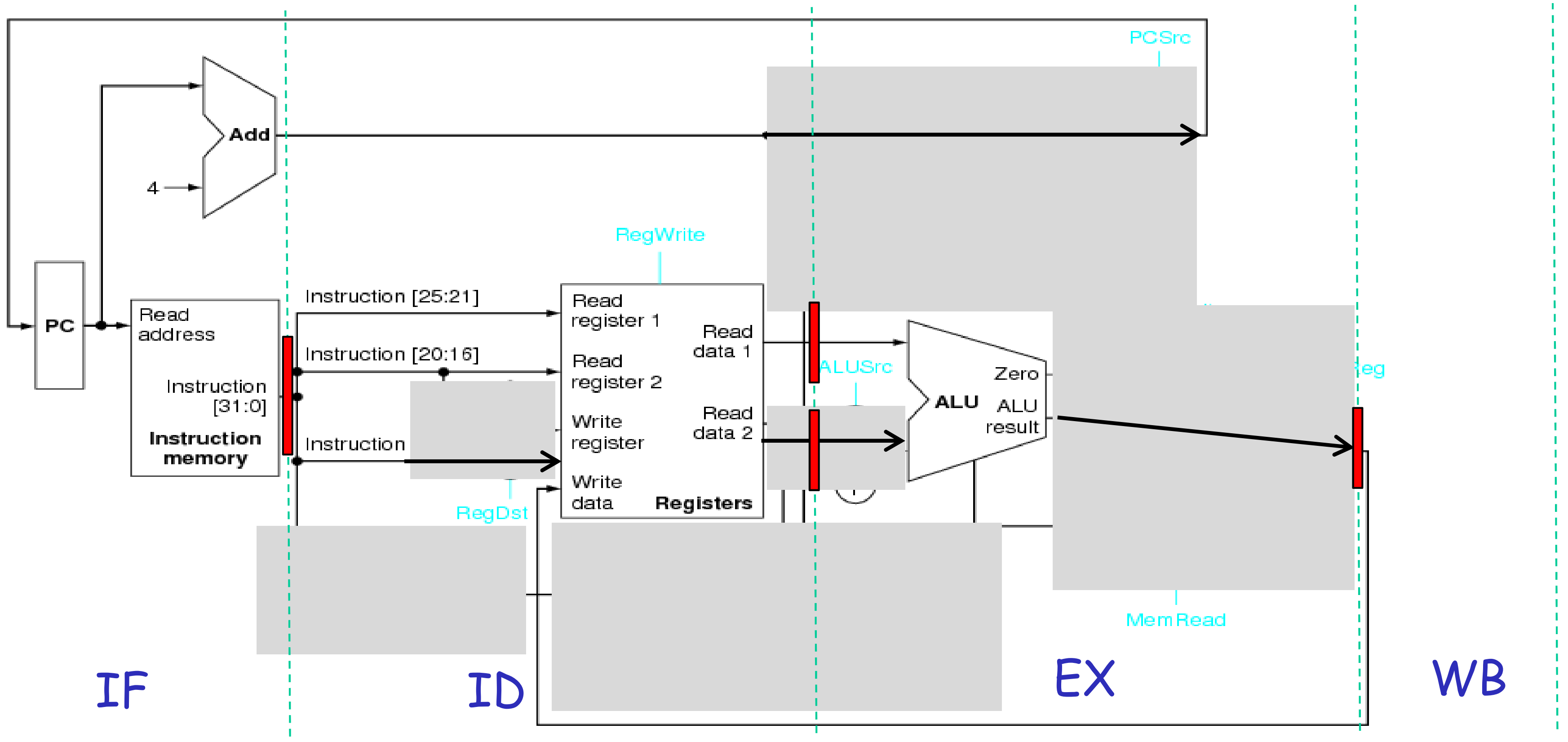
- IF: Instruction Fetch from instruction memory
- ID: Instruction Decode and operand fetch from regfile
- EX: EXecute operation
- WB: Write result Back to regfile



Pipeline registers

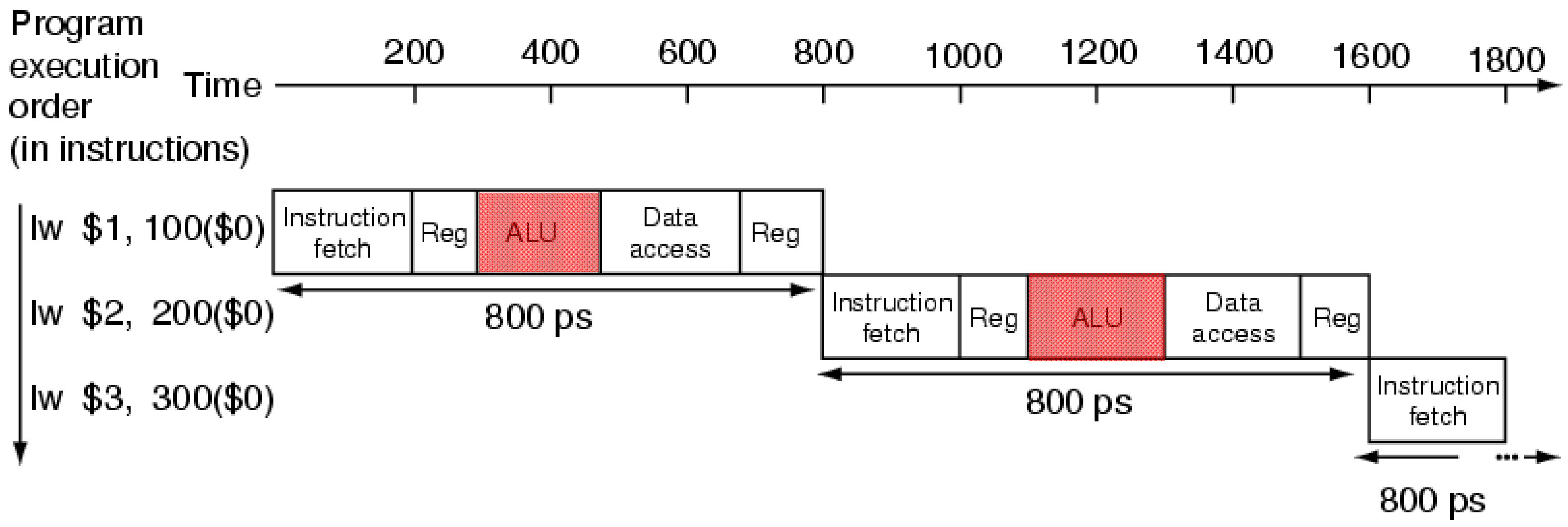
```
•    add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
•    add  $1,  $1,  $1   # $1 <= 22 + 22
•    add  $2,  $2,  $2   # $2 <= 33 + 33
•    add  $0,  $0,  $0   # NOP
•    add  $0,  $0,  $0   # NOP
•    add  $0,  $0,  $0   # NOP
```

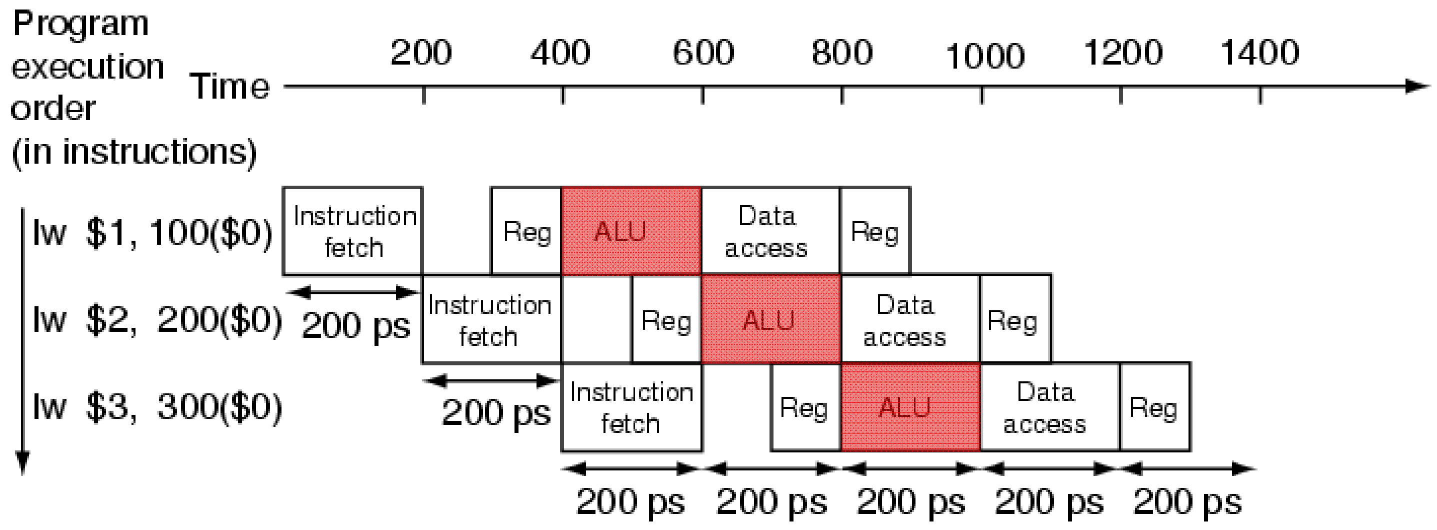
assuming initial values of r[1]=22 and r[2]=33



Exercise: Correcting the pipeline datapath

Single-cycle and pipelined processors



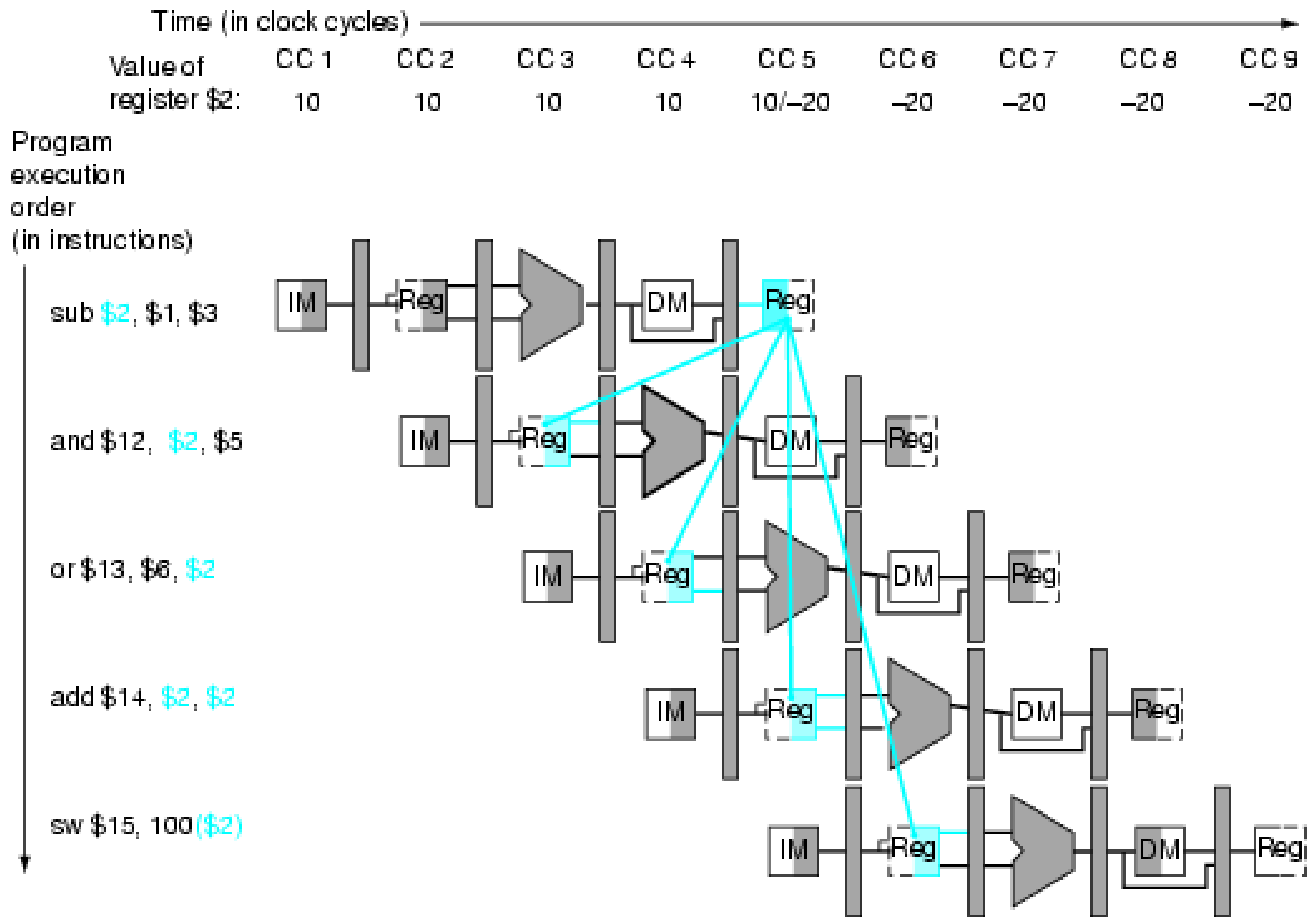


Homework 4

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.

Data hazard caused by data dependency

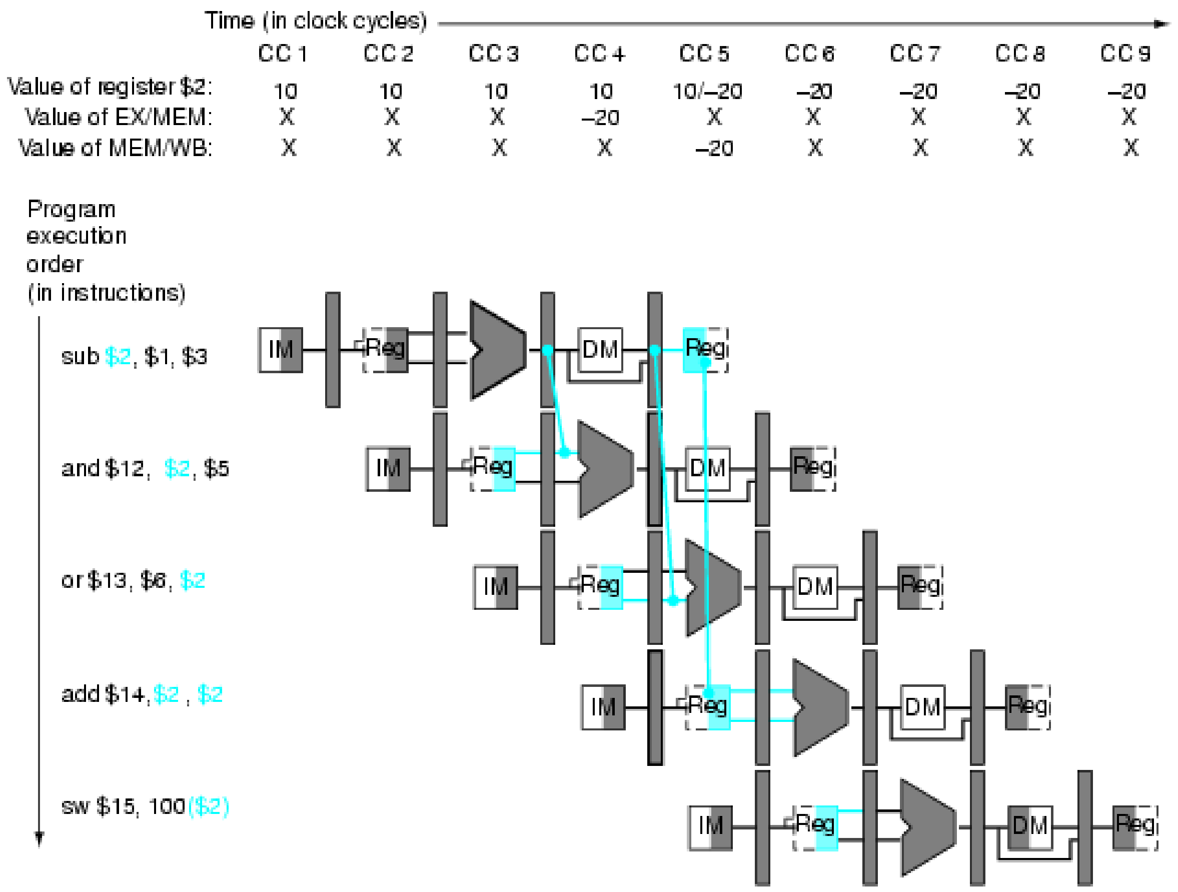
- The register value $2 for AND instruction is not written back to register file yet.



pipeline diagram

Data forwarding

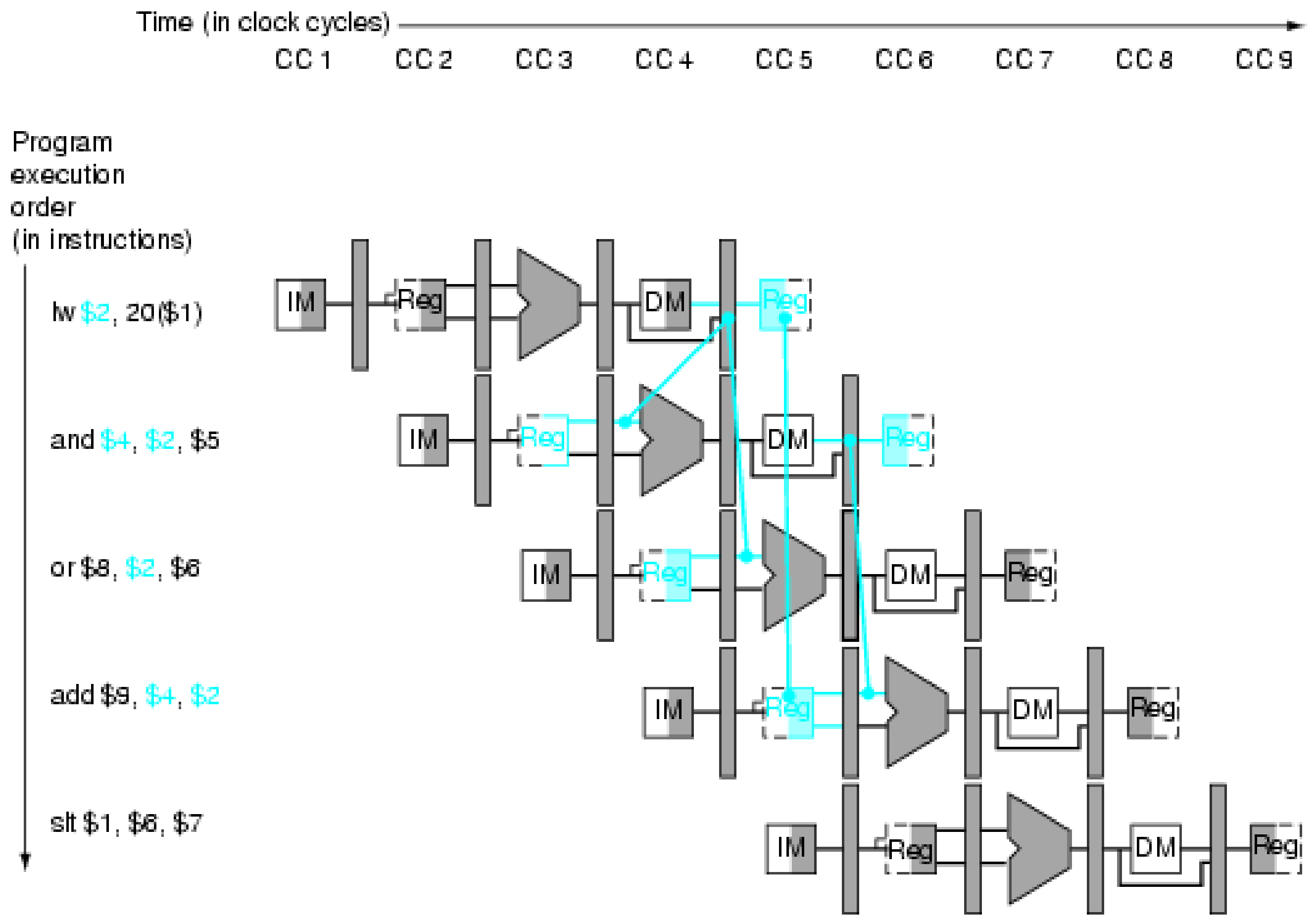
- Do not wait for data to be written to register file. Sending the data directly to where needed (consumers).
- In these cases, pipeline registers provide the operand.



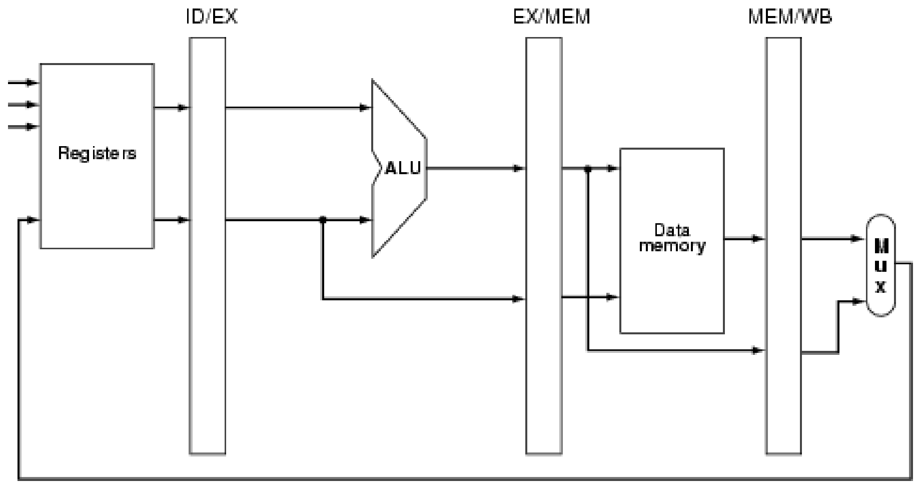
Time (in clock cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
CC 8
CC 9
Value of register $2: 10 10 10 10 10/–20 –20 –20 –20 –20
Value of EX/MEM: X X X –20 X X X X X
Value of MEM/WB: X X X X –20 X X X X
Program execution order (in instructions)
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2 , $2
sw $15, 100($2)
IM
Reg
DM

Stalling the processor by data hazard

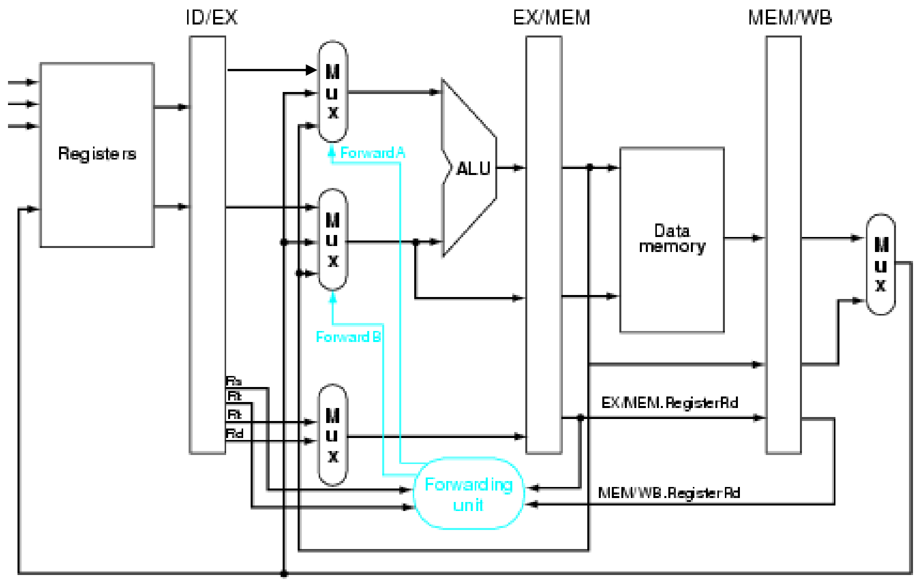
- Stall needed only when …



Adding path and mux for data forwarding



a. No forwarding



b. With forwarding