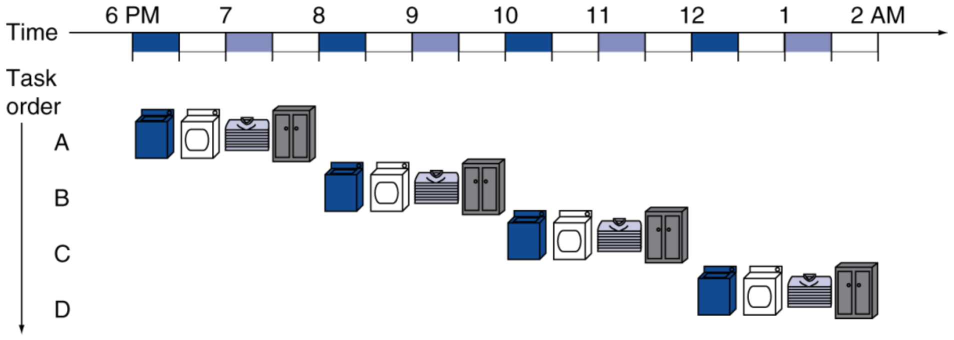
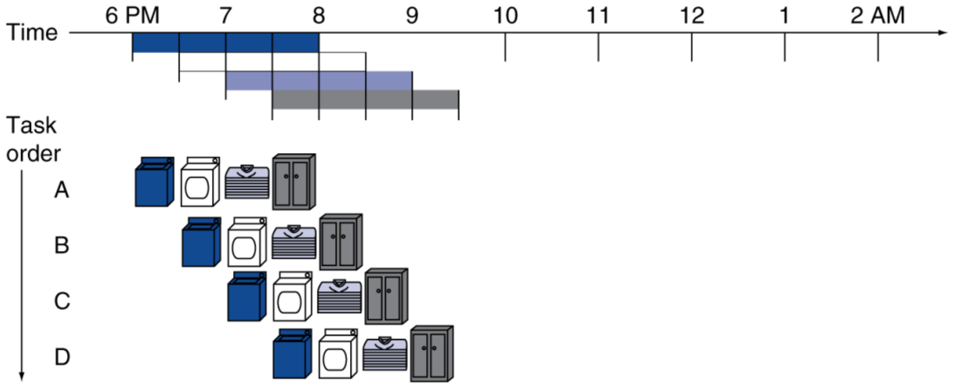
Pipelining Analogy

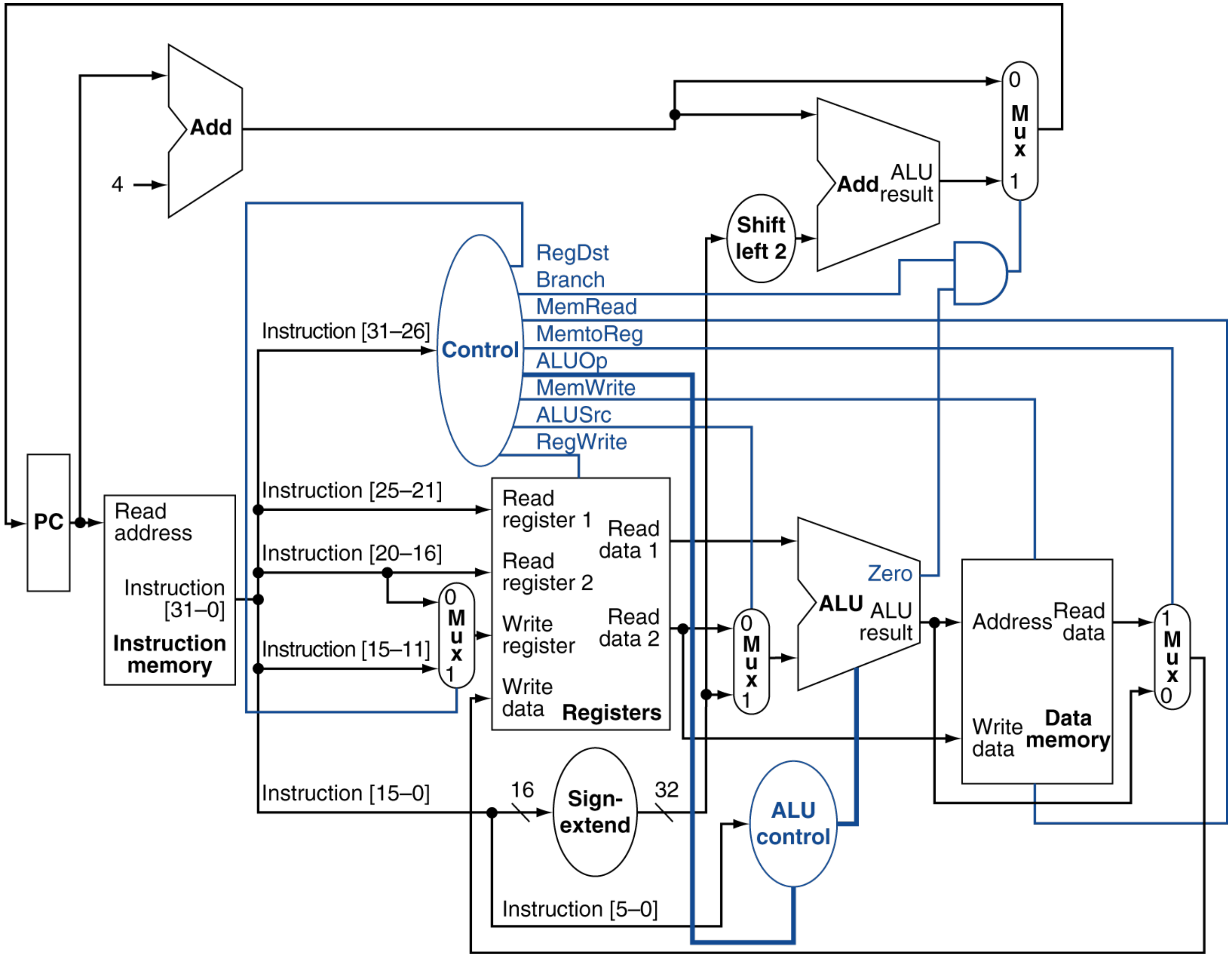
- Pipelined laundry: overlapping execution
  - Parallelism improves performance





- Four loads:
  - Speedup
    = 8/3.5 = 2.3
- Infinite loads:
  - Speedup
    = 4n/(3+n) ≈ 4
    = number of stages

Datapath With Control



Pipelining 을 하려면

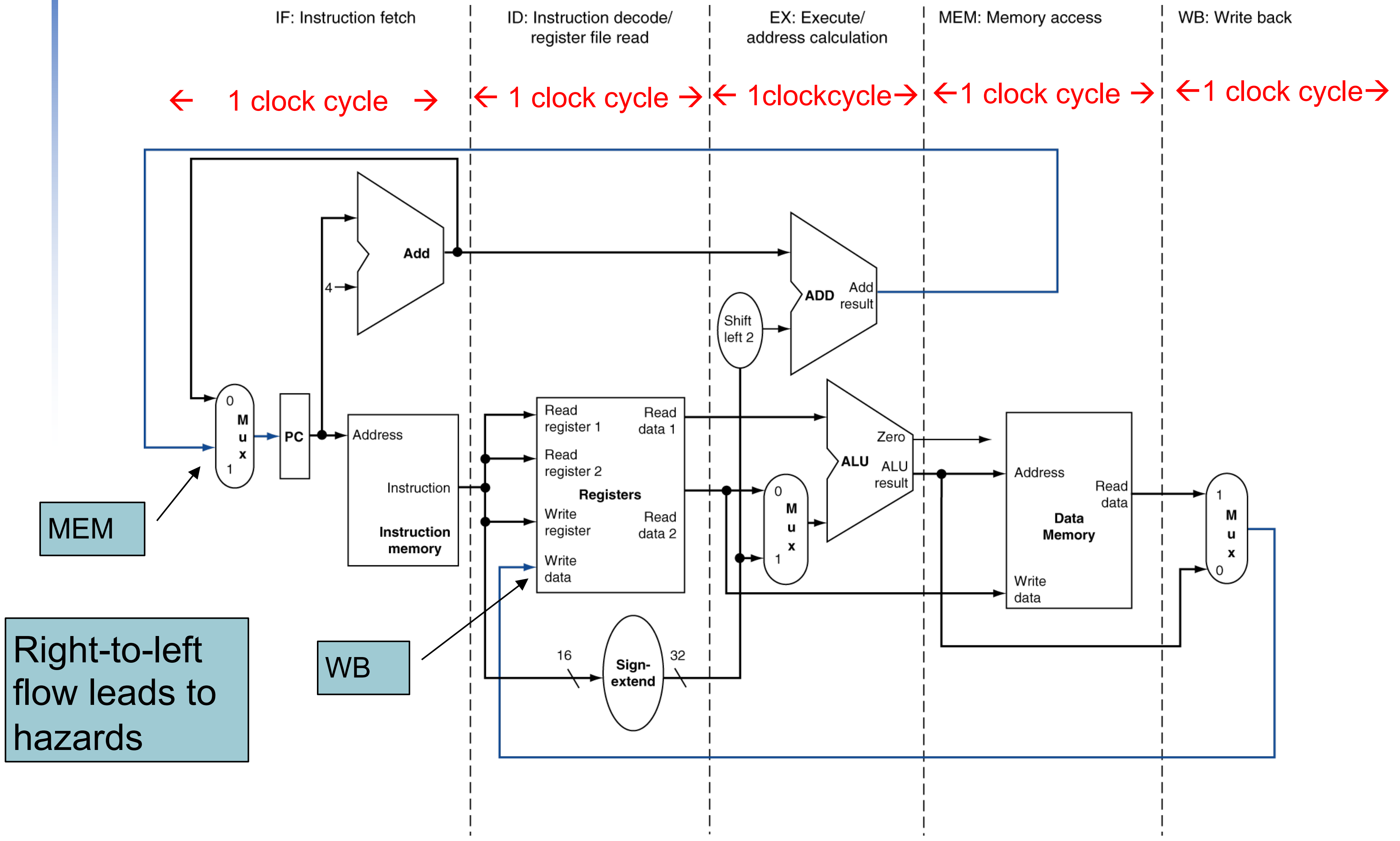
- 수행할 task 를 여러 개의 sub-task 로 나누어야 함
- 각 sub-task 는 1-clock cycle 에 수행된다.

→ 즉 명령어 한 개는 여러 clock cycles 에 수행된다.

MIPS Pipeline

- Five stages, one step per stage
  1. IF: Instruction fetch from memory
  2. ID: Instruction decode & register read
  3. EX: Execute operation or calculate address
  4. MEM: Access memory operand
  5. WB: Write result back to register

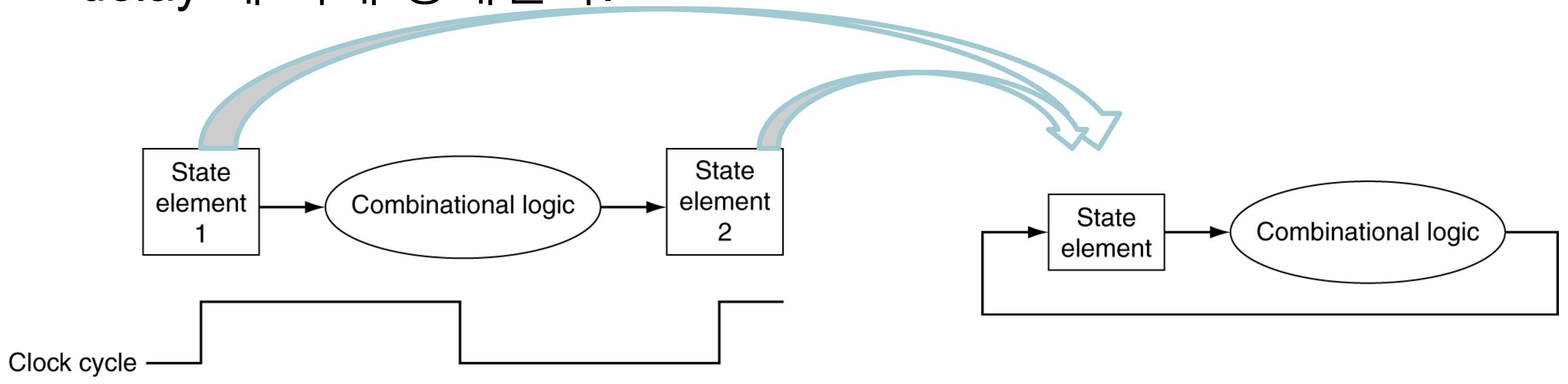
MIPS Pipelined Datapath



Right-to-left flow leads to hazards

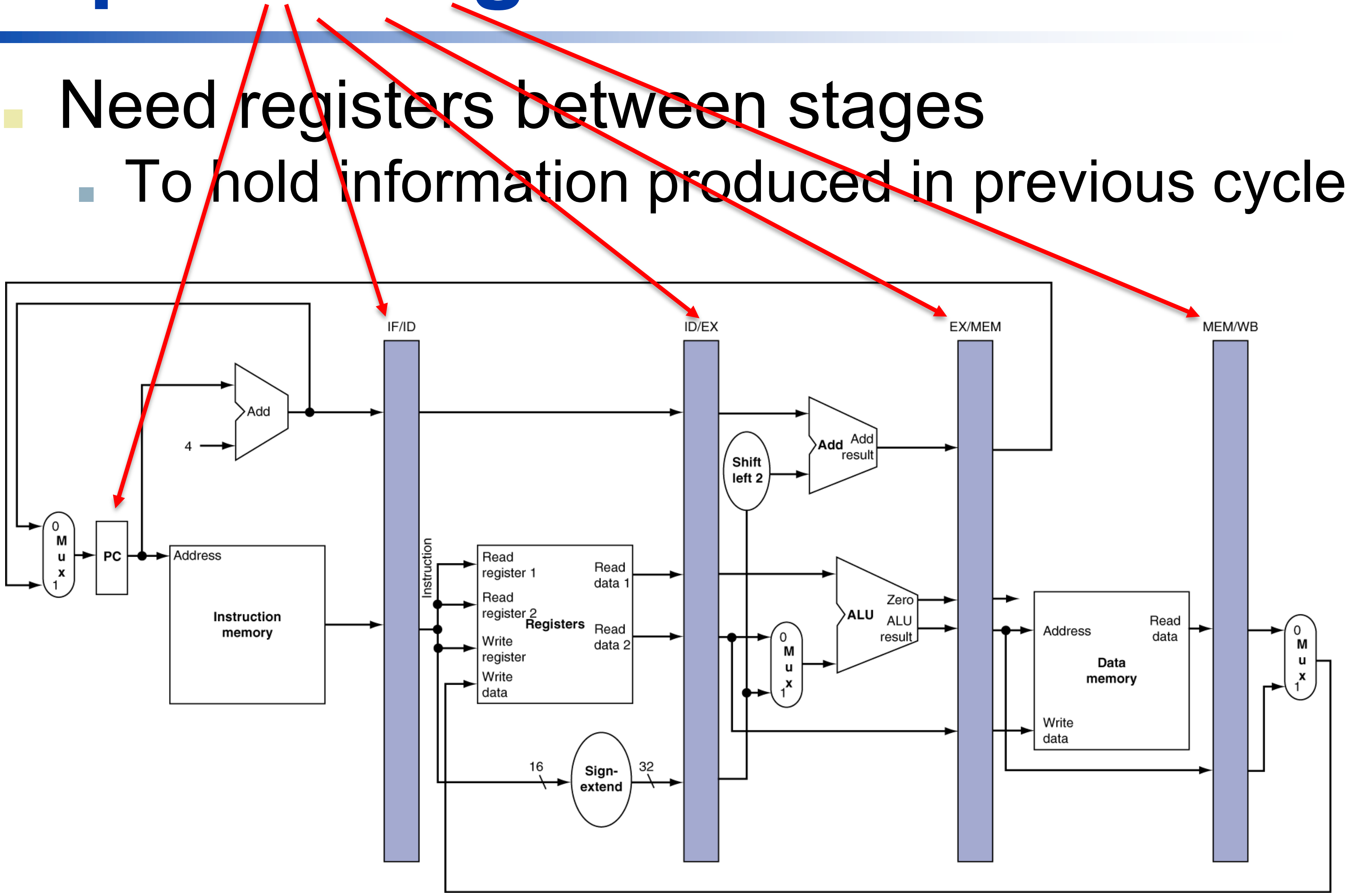
Synchronous Digital 회로의 동작

- 클락 사이클 동안(Between clock edges) 에 Combinational 회로에서 입력 신호에 대한 출력 신호를 만든다
- combinational 회로의 input은 state elements의 output이다.
- combinational 회로의 output은 state elements의 input이다.
- clock period (clock edge 간의 간격) 은 combinational 회로의 longest delay 에 의해 정해진다.



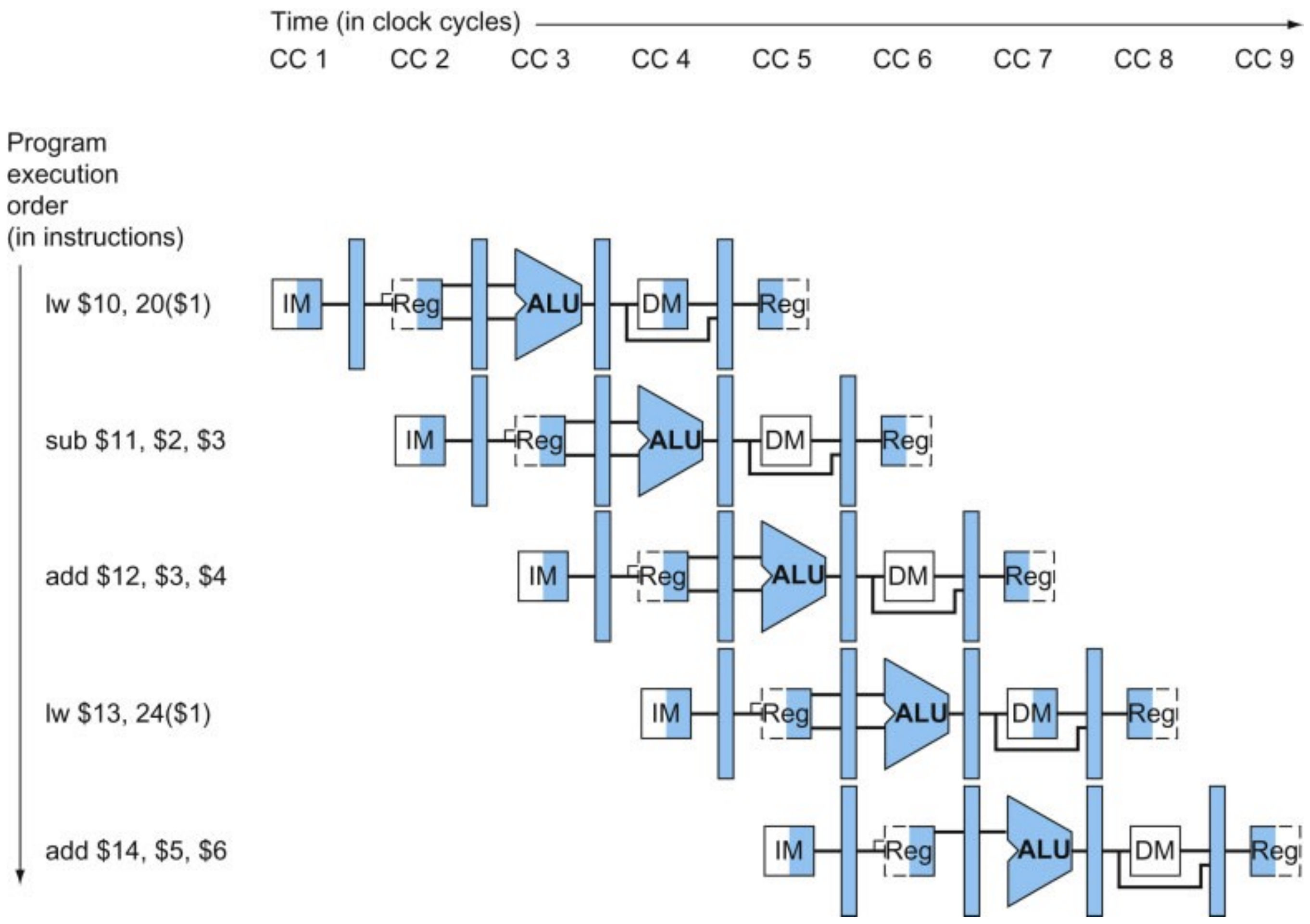
Pipeline registers

- Need registers between stages
  - To hold information produced in previous cycle

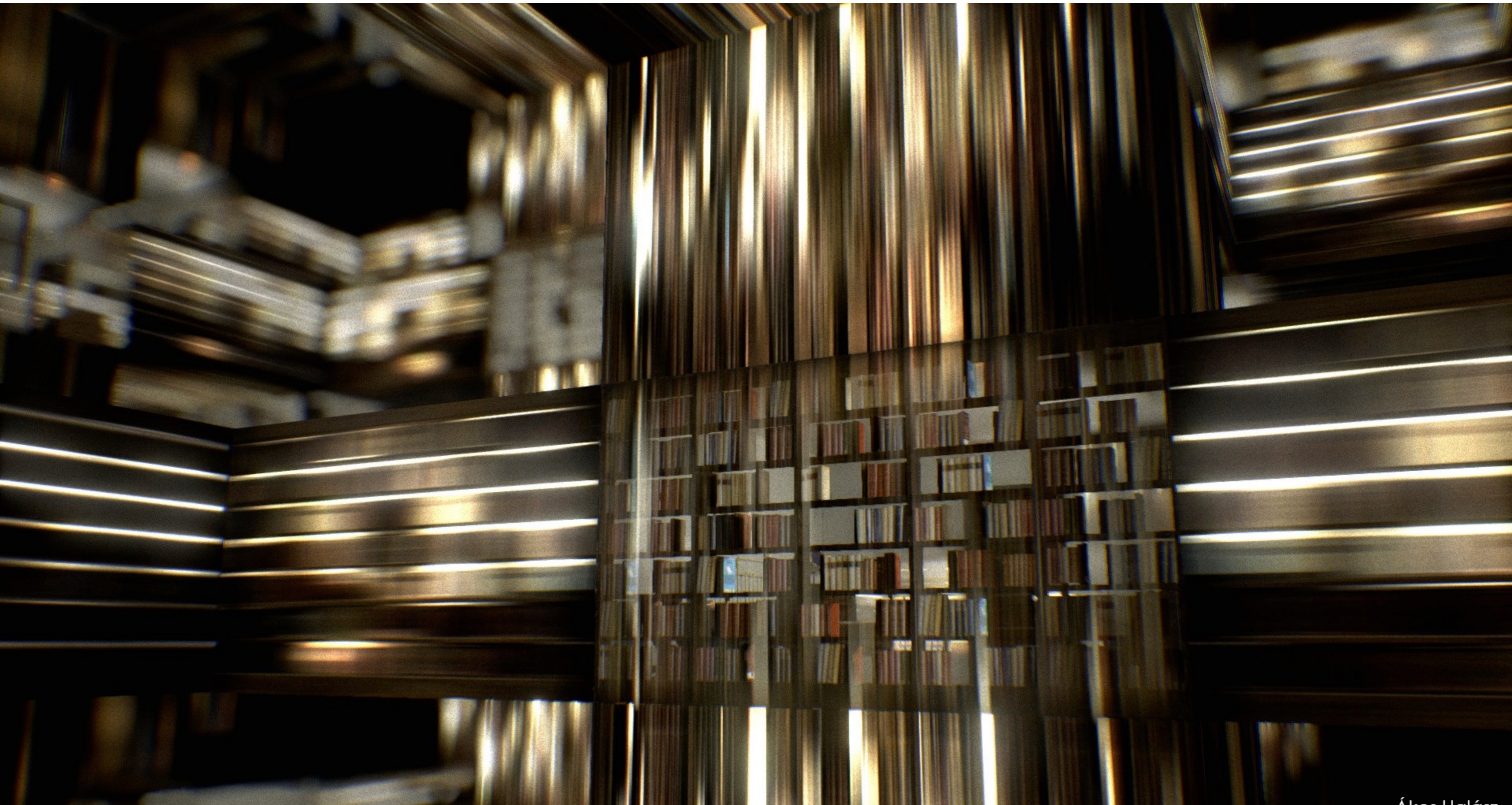


write signals to pipeline registers are always 1

Multi-Cycle Pipeline Diagram

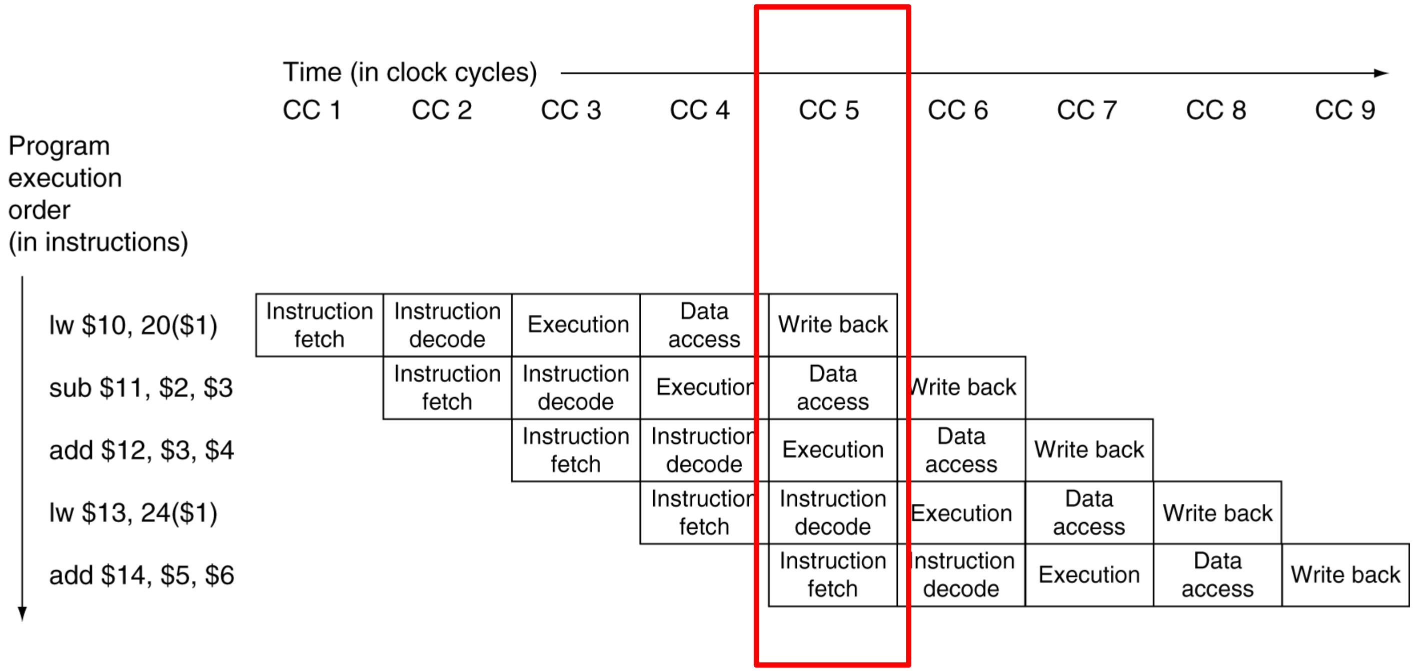


tesseract from movie “interstellar”(2014)



Multi-Cycle Pipeline Diagram

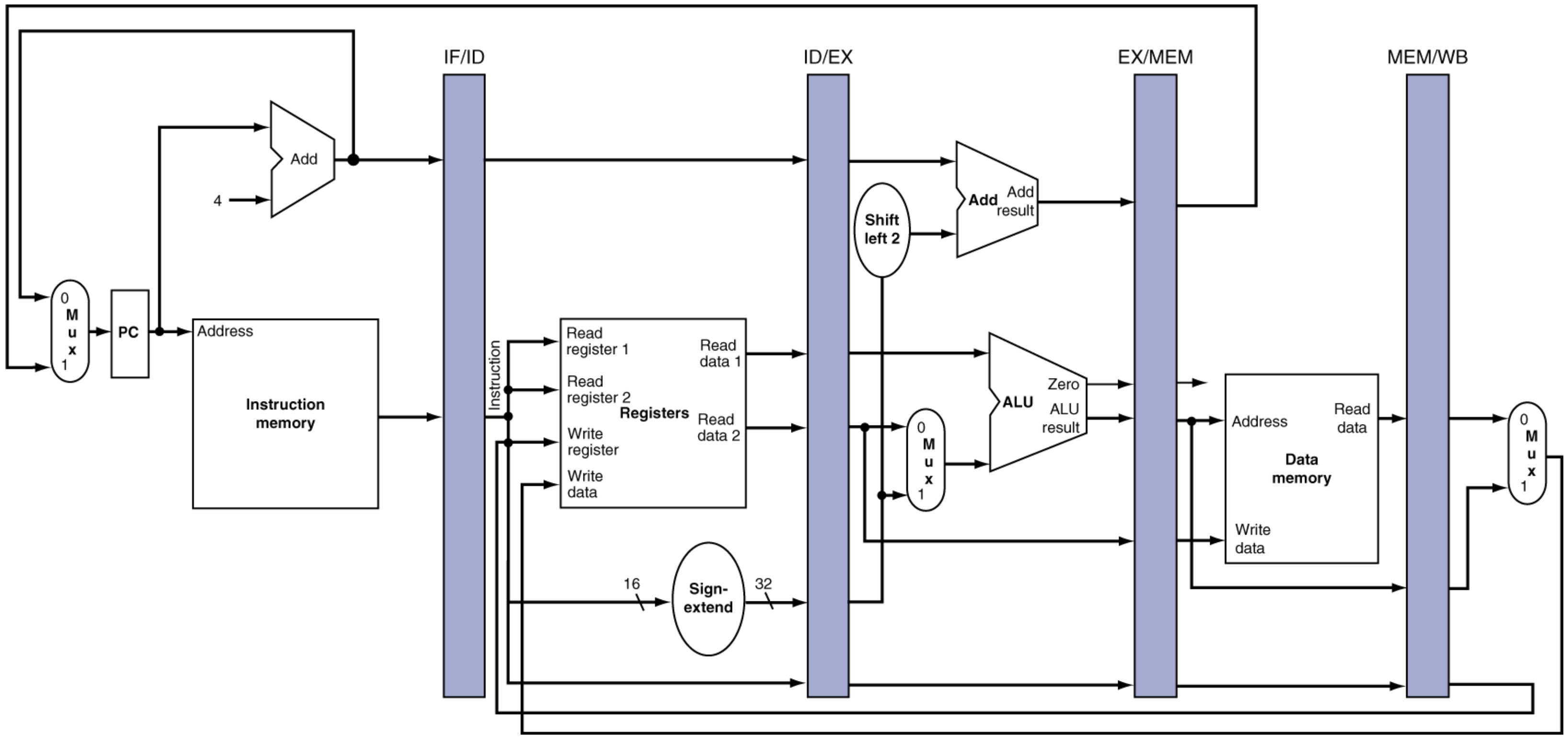
- Traditional form



Single-Cycle Pipeline Diagram

- State of pipeline in a given cycle (CC5)

add $14, $5, $6	lw $13, 24 ($1)	add $12, $3, $4	sub $11, $2, $3	lw $10, 20($1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back



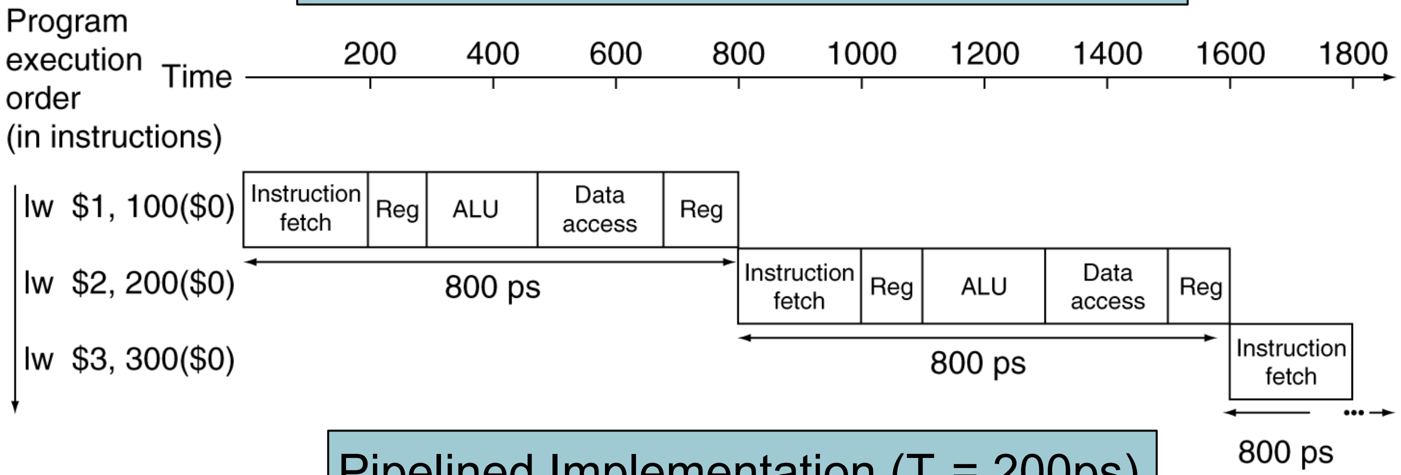
Cycle Time of Pipeline Processor

- Assume time for stages is
  - 100ps for register read or write
  - 200ps for memory, ALU
- Compare pipelined datapath with single-cycle datapath

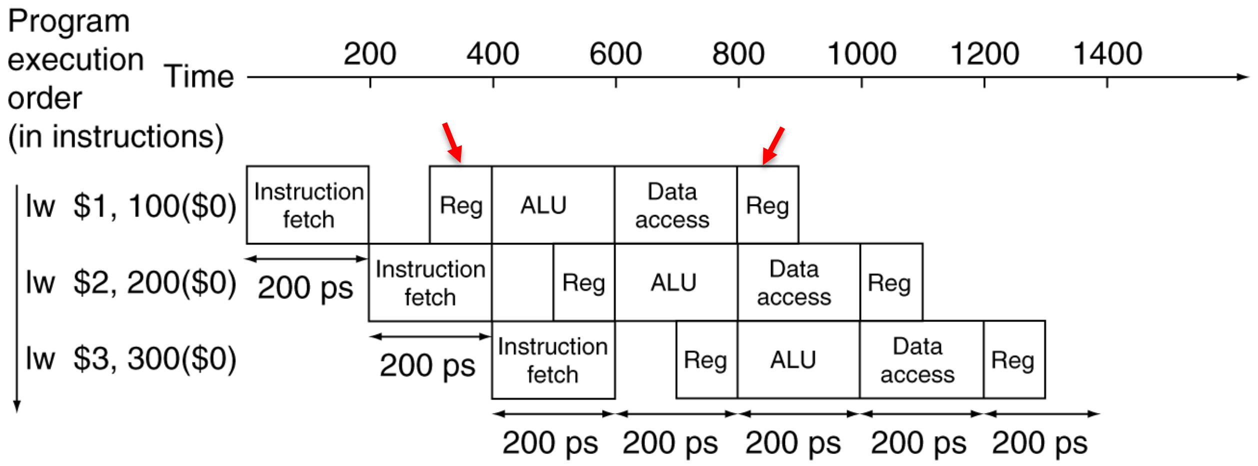
Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

Pipeline Performance

Single-cycle Implementation (T_c= 800ps)



Pipelined Implementation (T_c= 200ps)



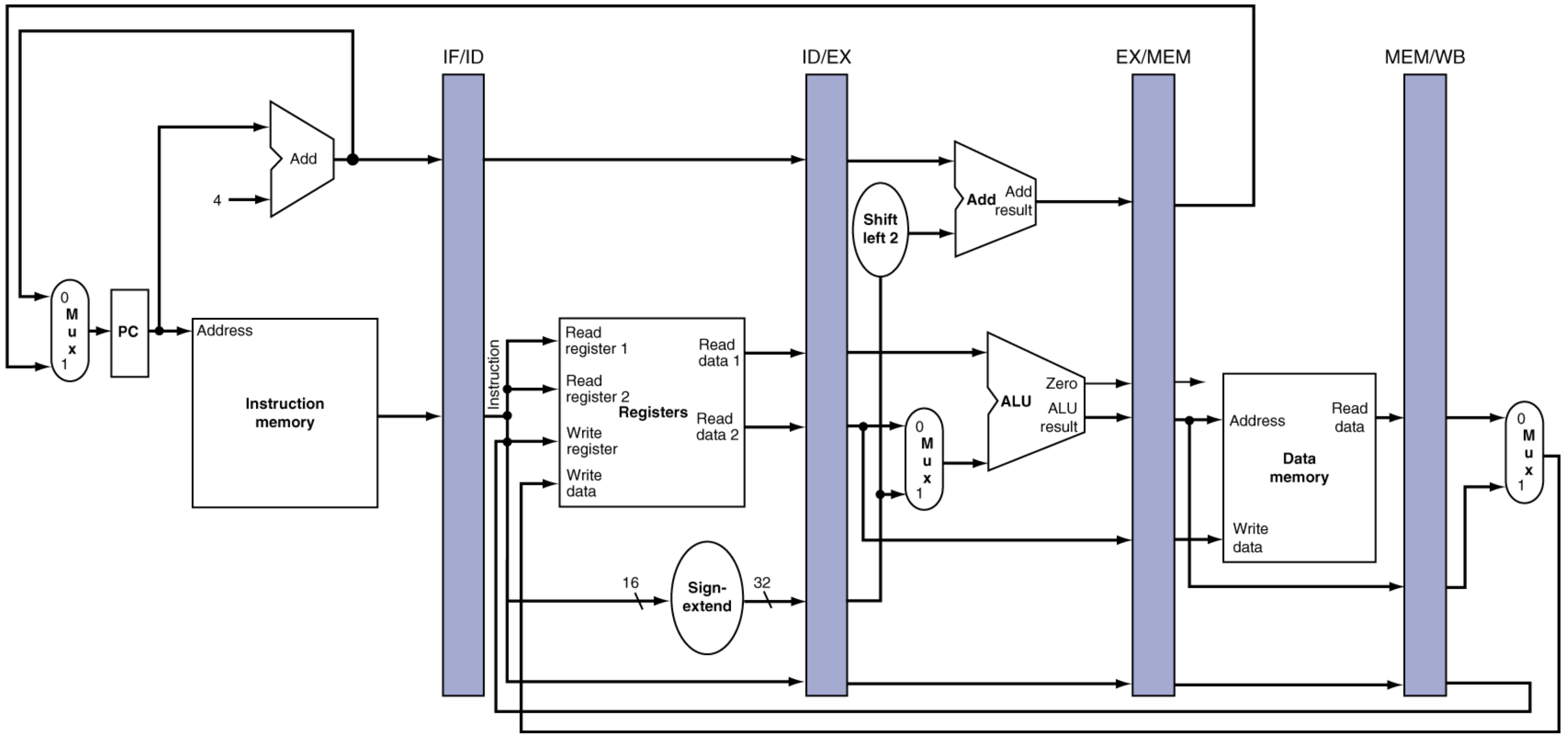
Depiction of Pipeline Operation

- Cycle-by-cycle flow of instructions through the pipelined datapath
  - "Single-clock-cycle" pipeline diagram
    - Shows pipeline usage in a single cycle
    - Highlight resources used
  - c.f. "multi-clock-cycle" diagram
    - Graph of operation over time
- We'll look at "single-clock-cycle" diagrams for load & store

Single-Cycle Pipeline Diagram

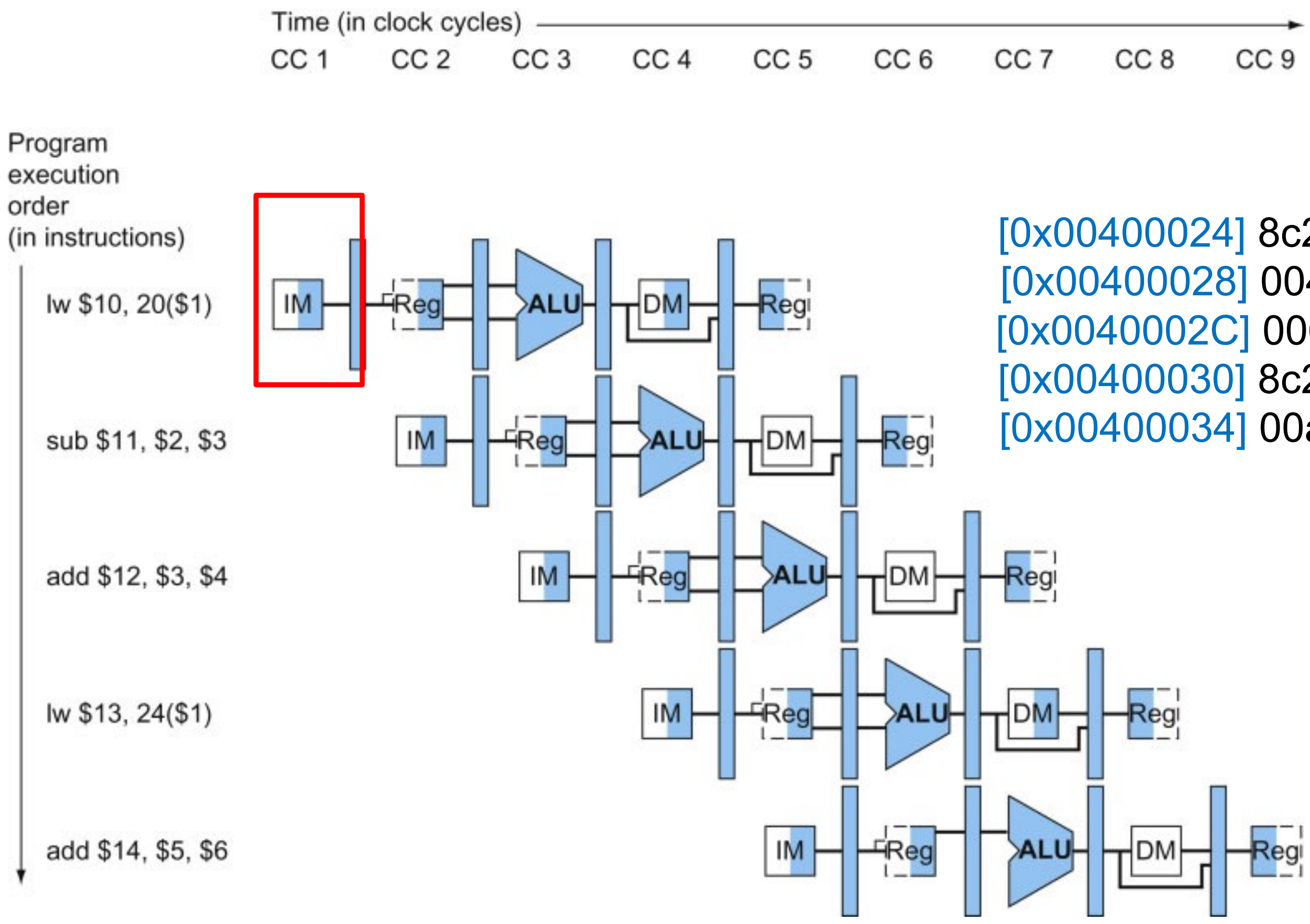
- State of pipeline in CC5

add \$14, \$5, \$6	lw \$13, 24 (\$1)	add \$12, \$3, \$4	sub \$11, \$2, \$3	lw \$10, 20(\$1)
Instruction fetch	Instruction decode	Execution	Memory	Write-back



Multi-Cycle Pipeline Diagram

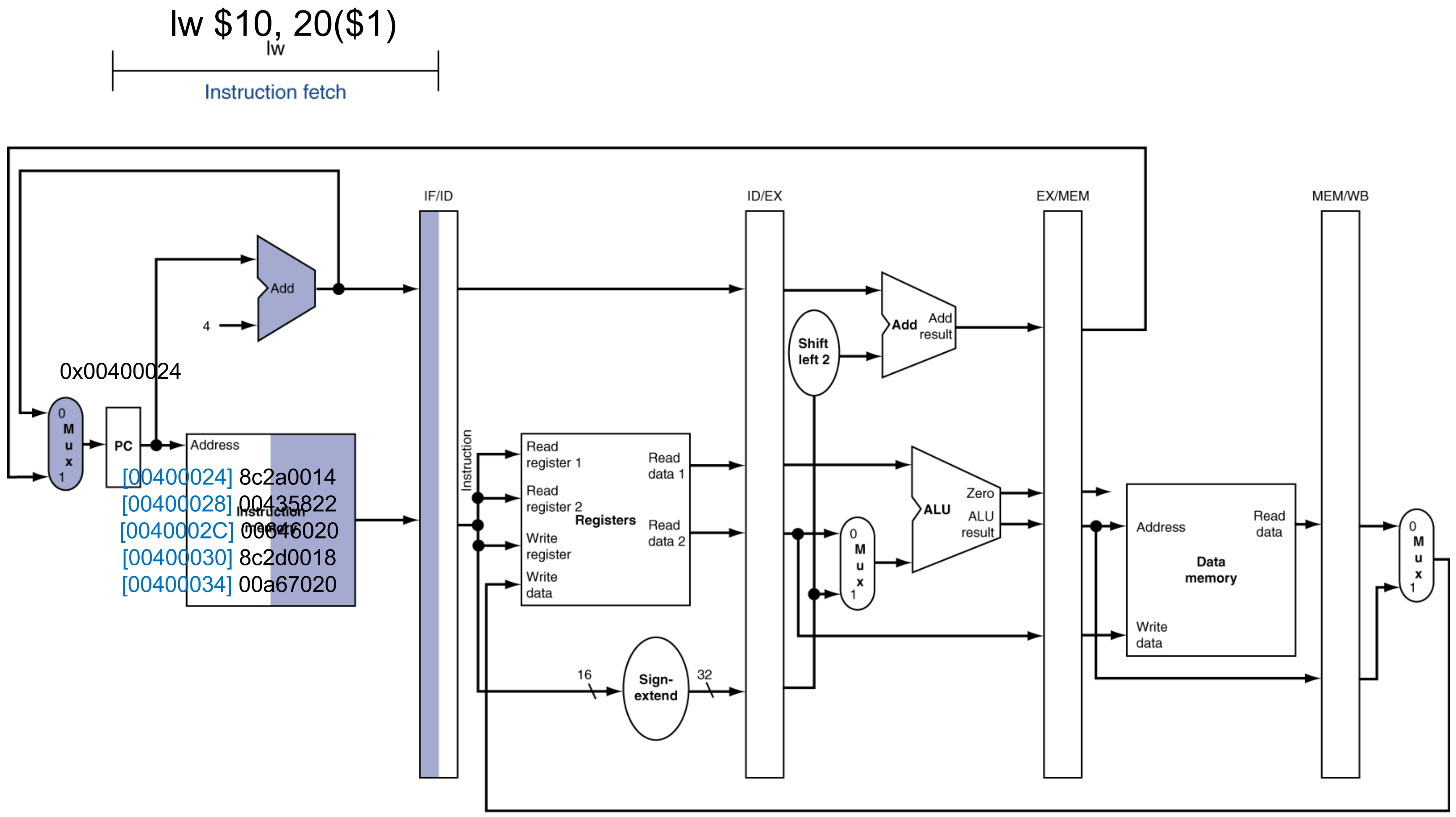
- showing resource usage

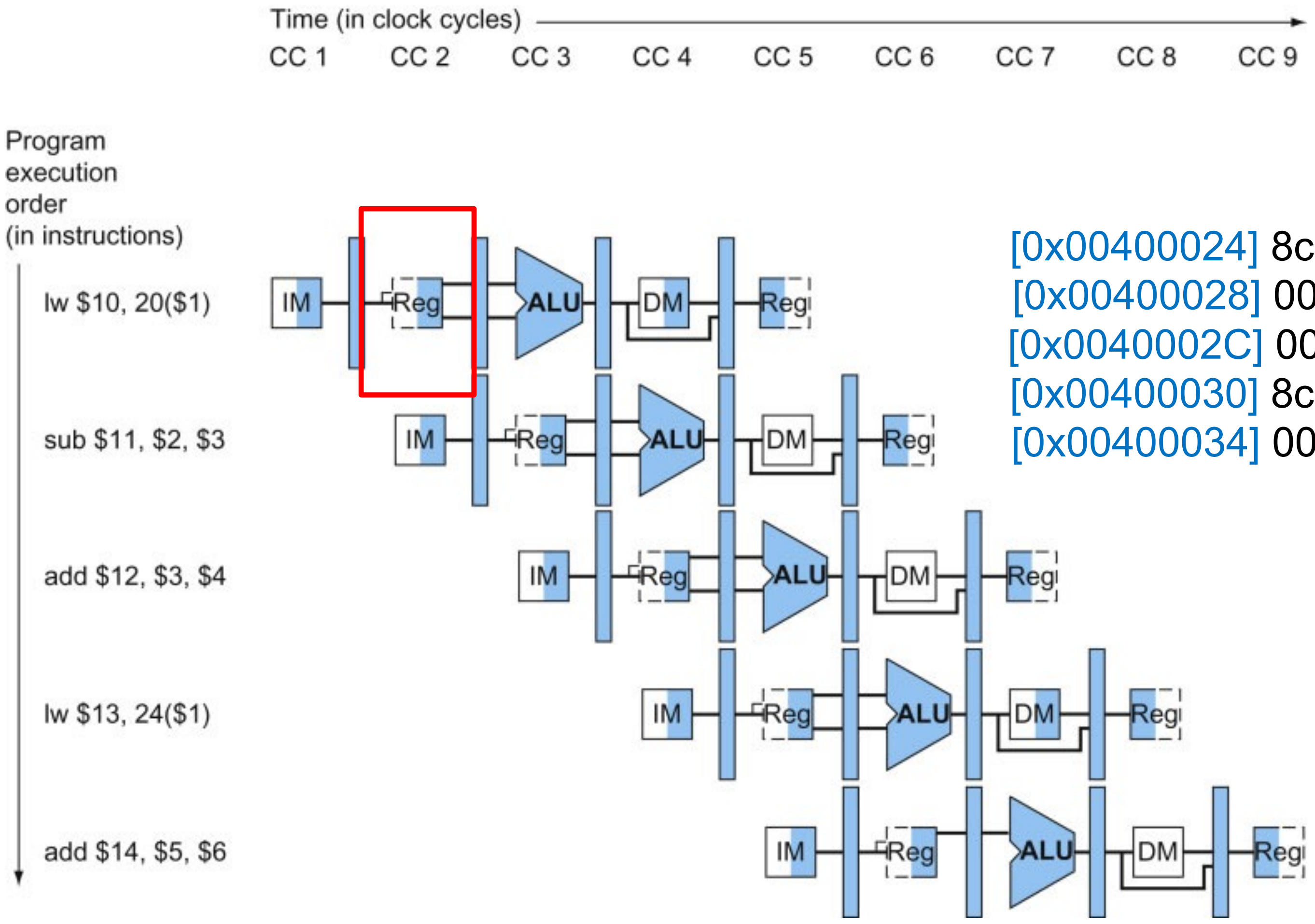


```
[0x00400024] 8c2a0014  lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018  lw
[0x00400034] 00a67020 add
```

IF for Load at CC1

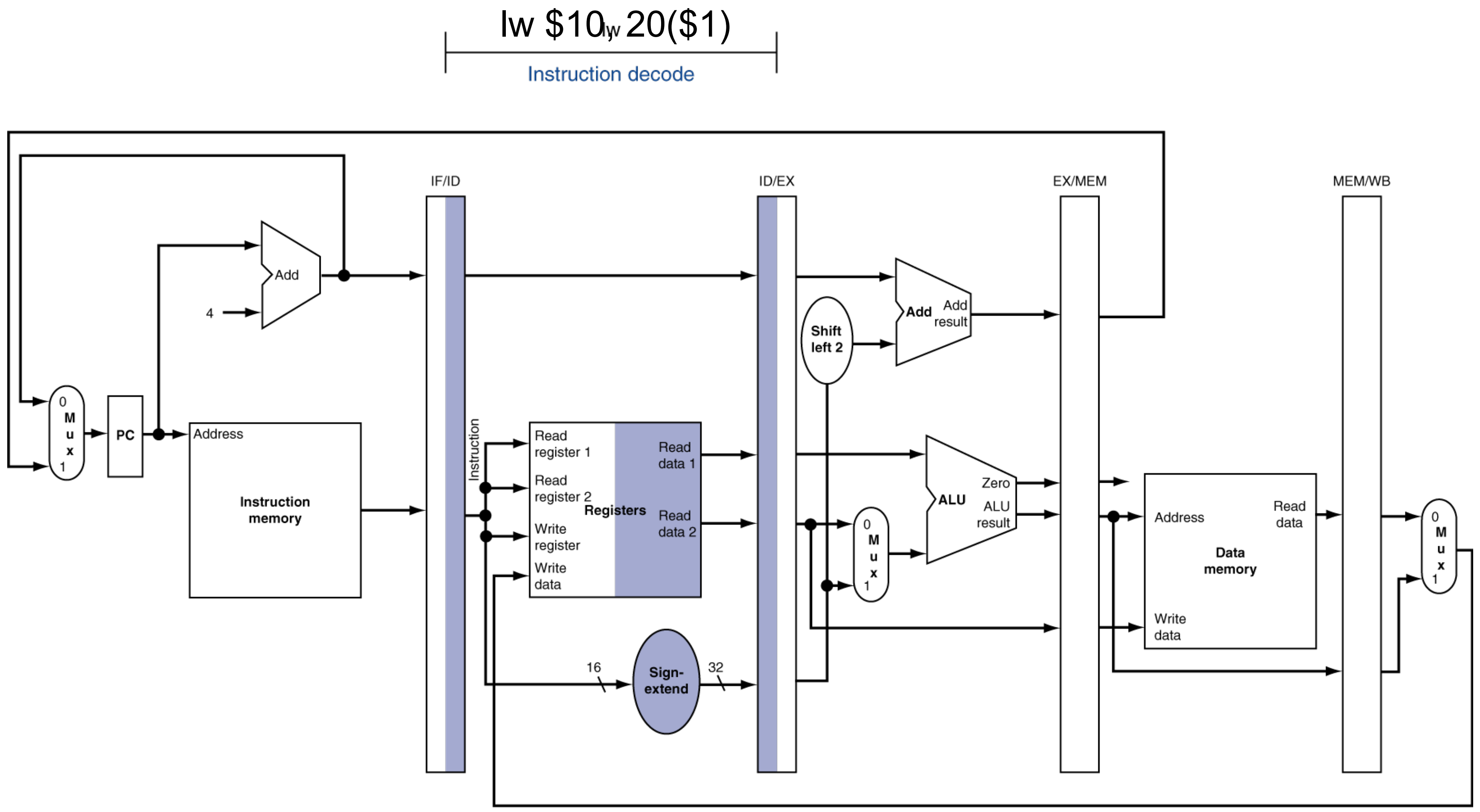
lw $10, 20($1)

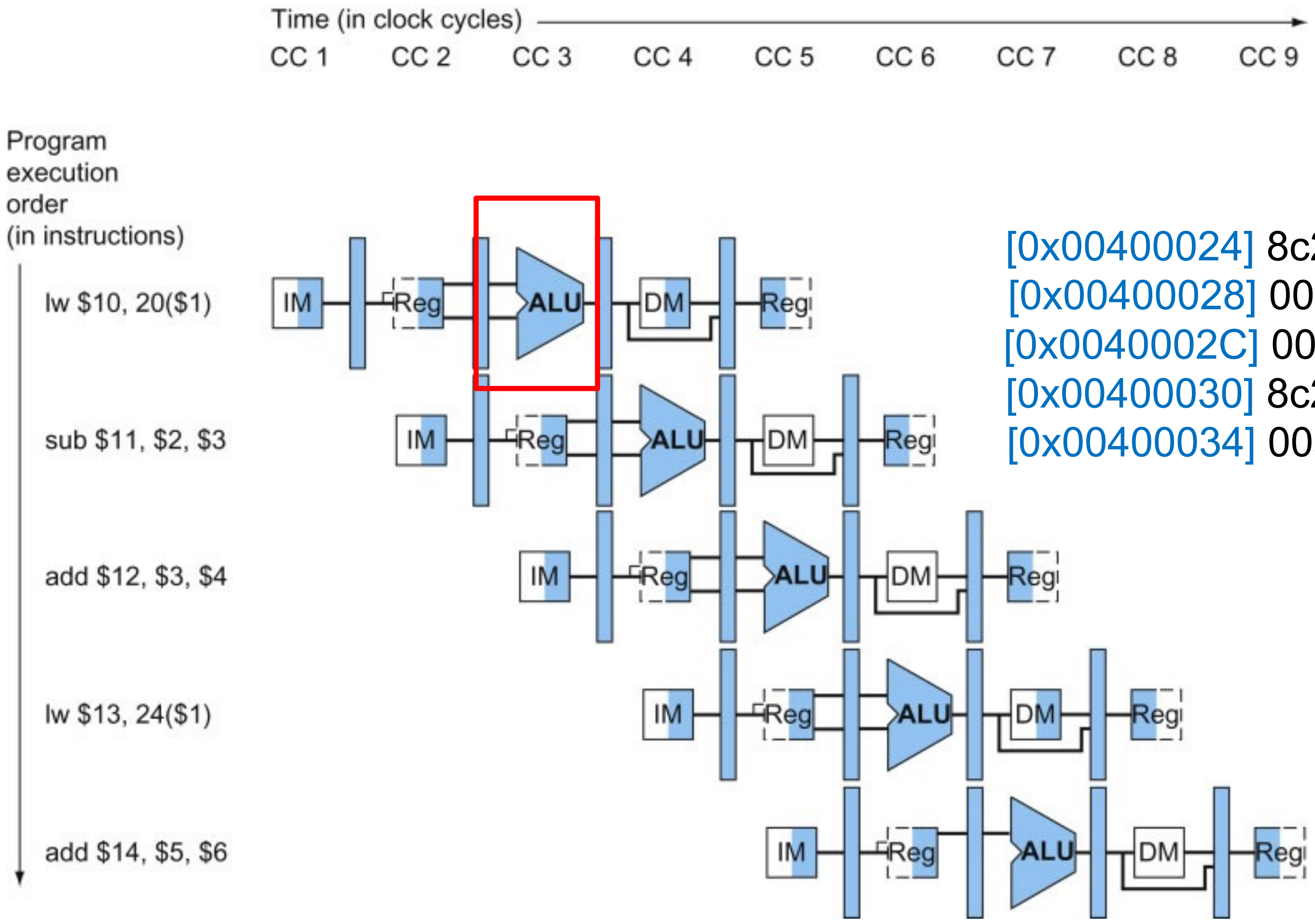




[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

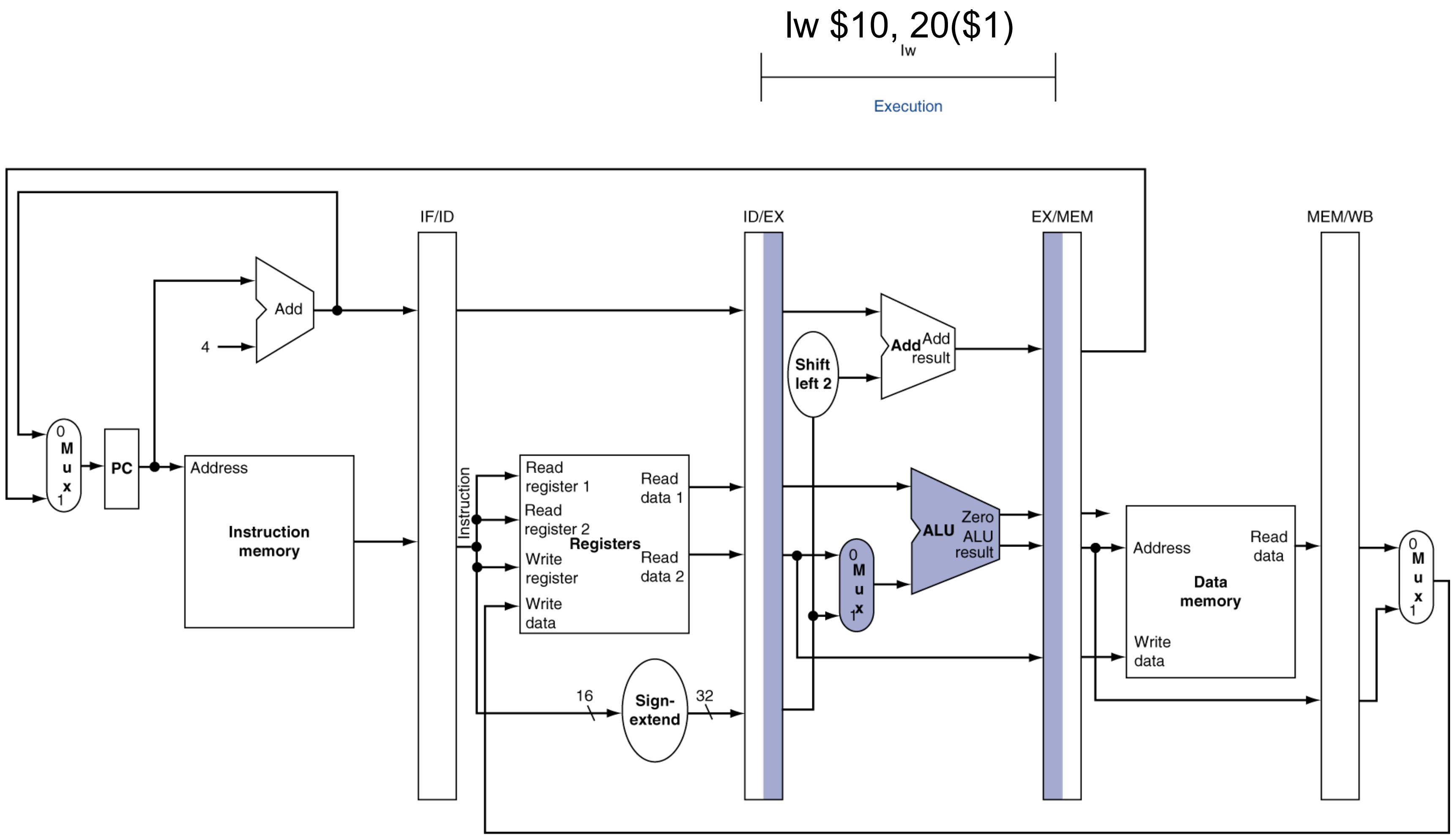
ID for Load at CC2

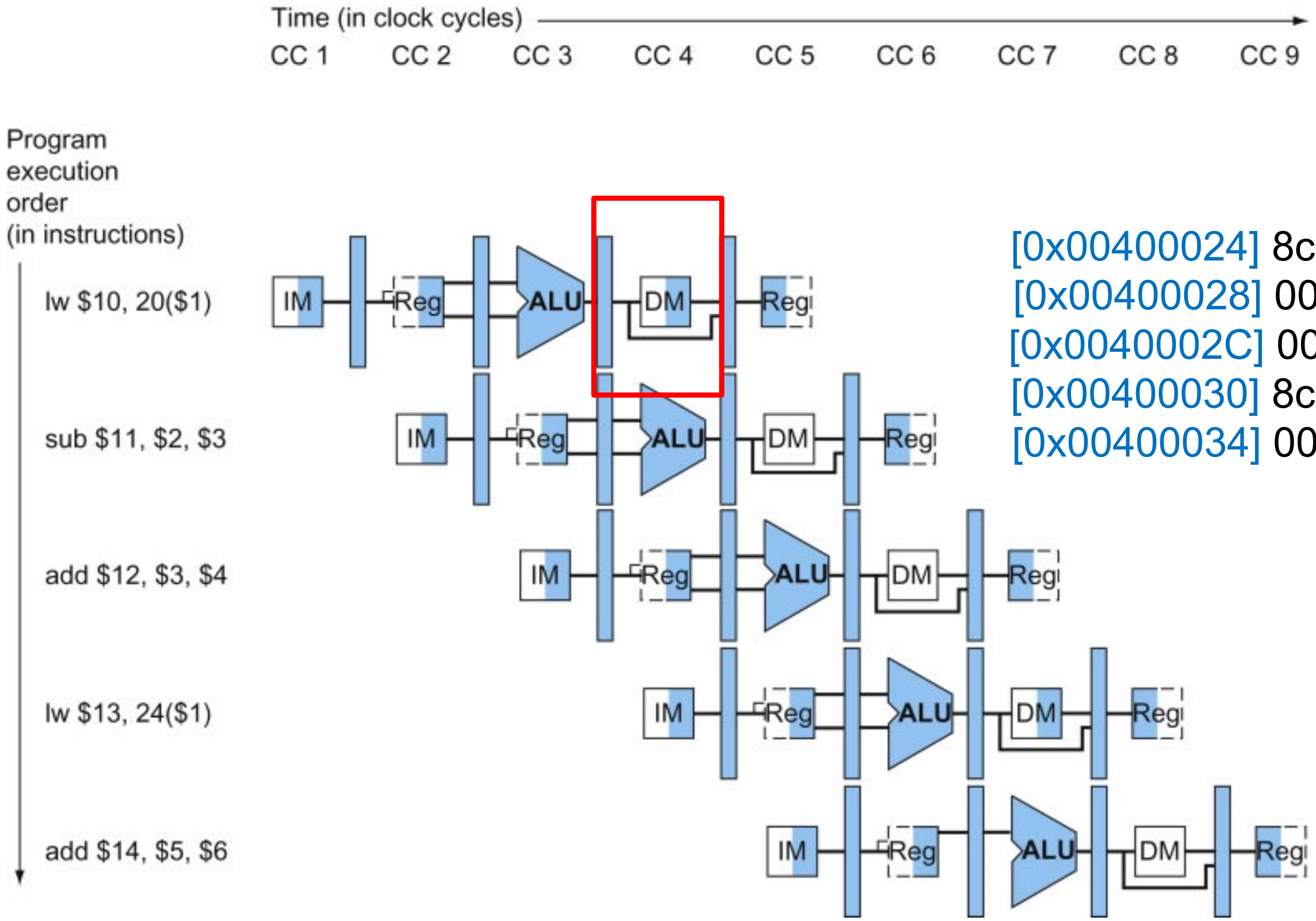




[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

EX for Load at CC3

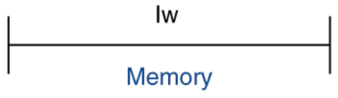


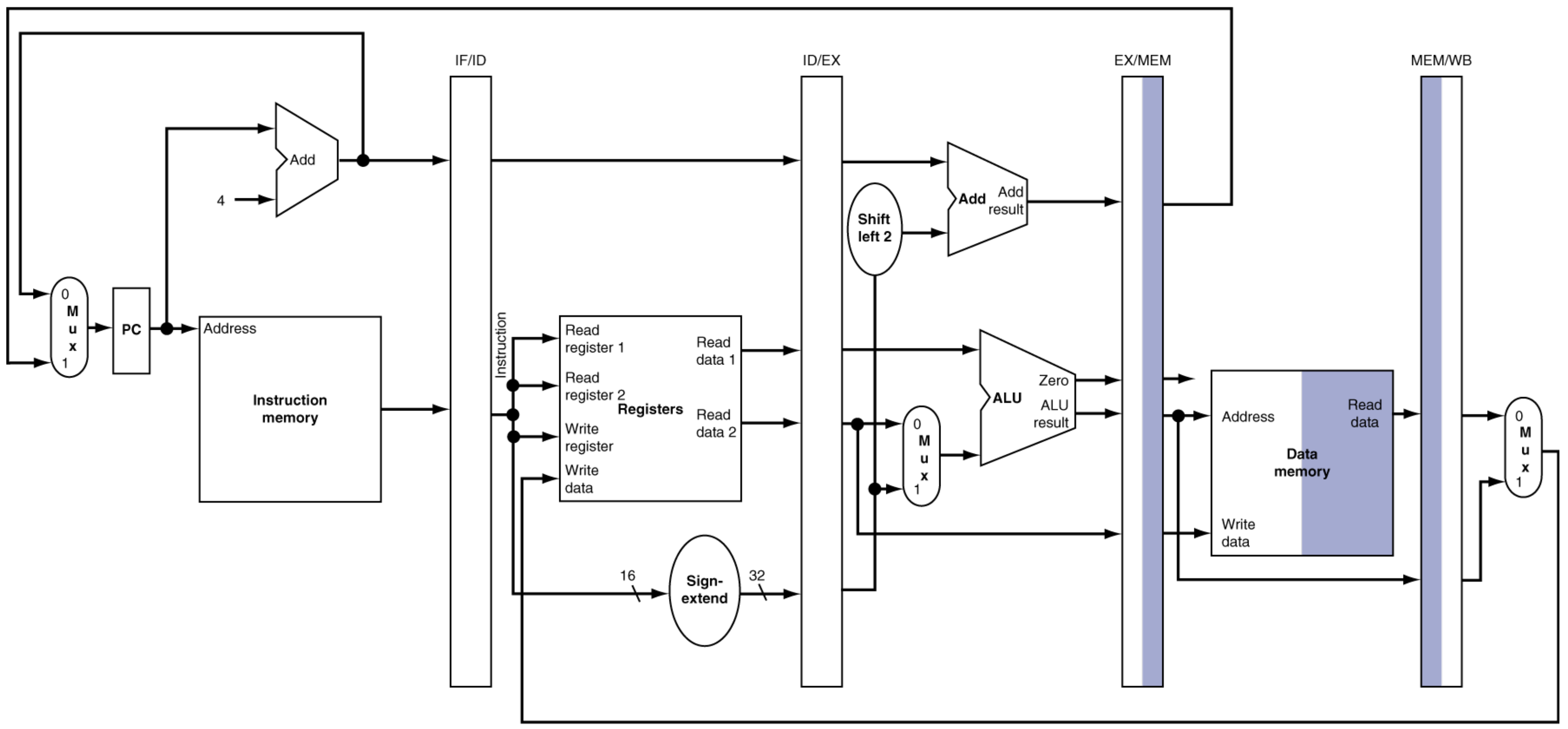


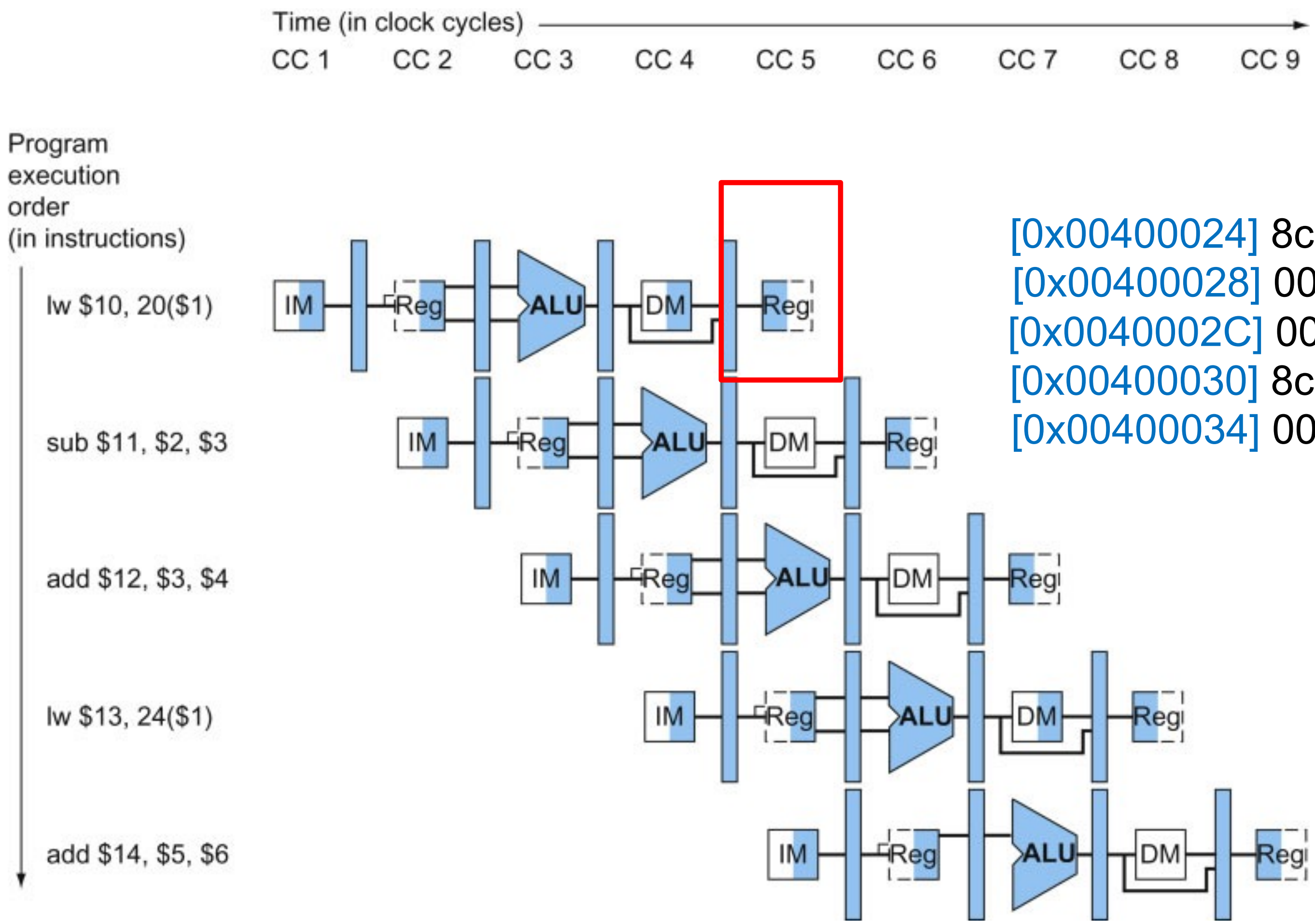
[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

MEM for Load at CC4

lw $10, 20($1)



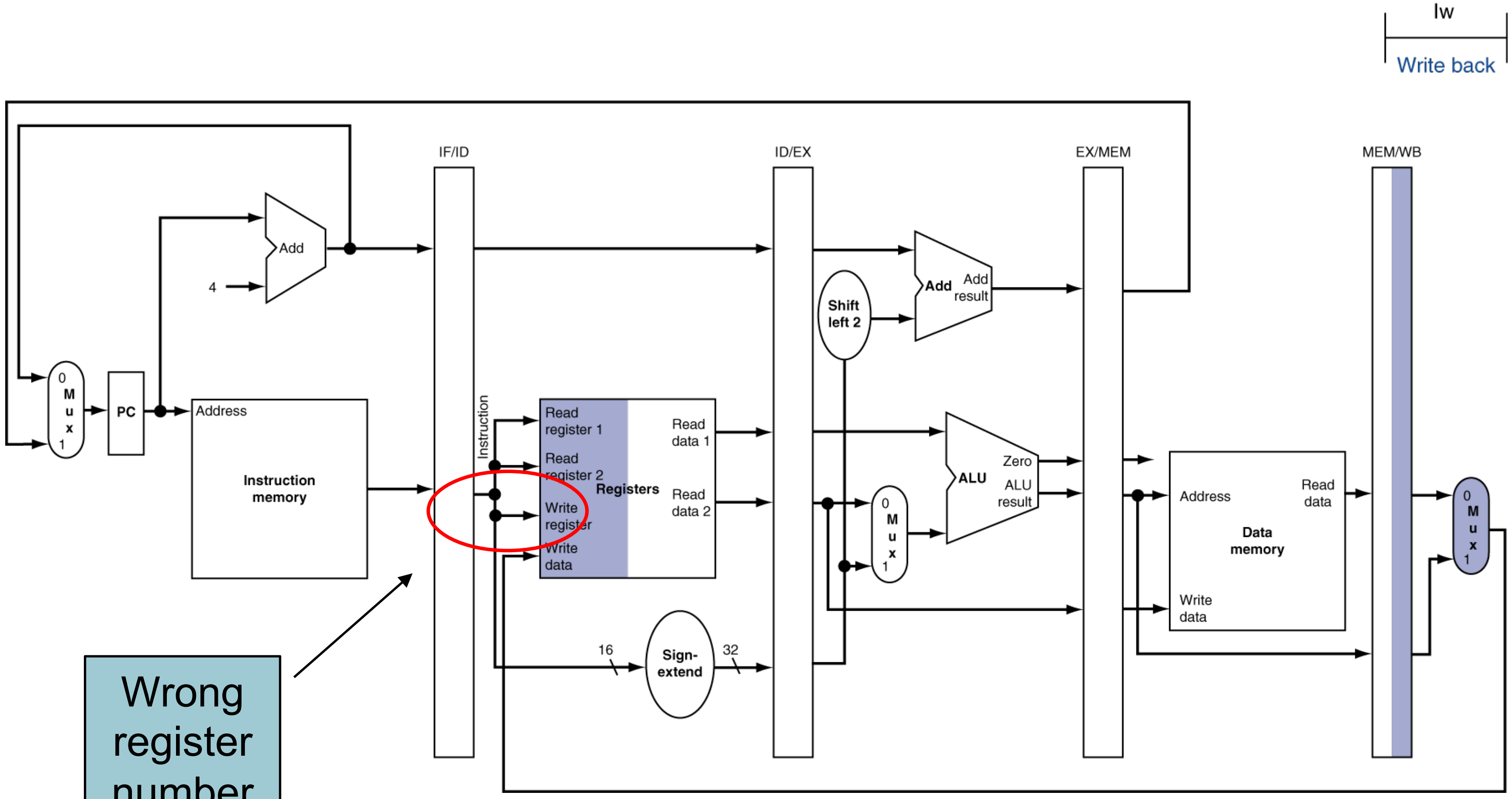




[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

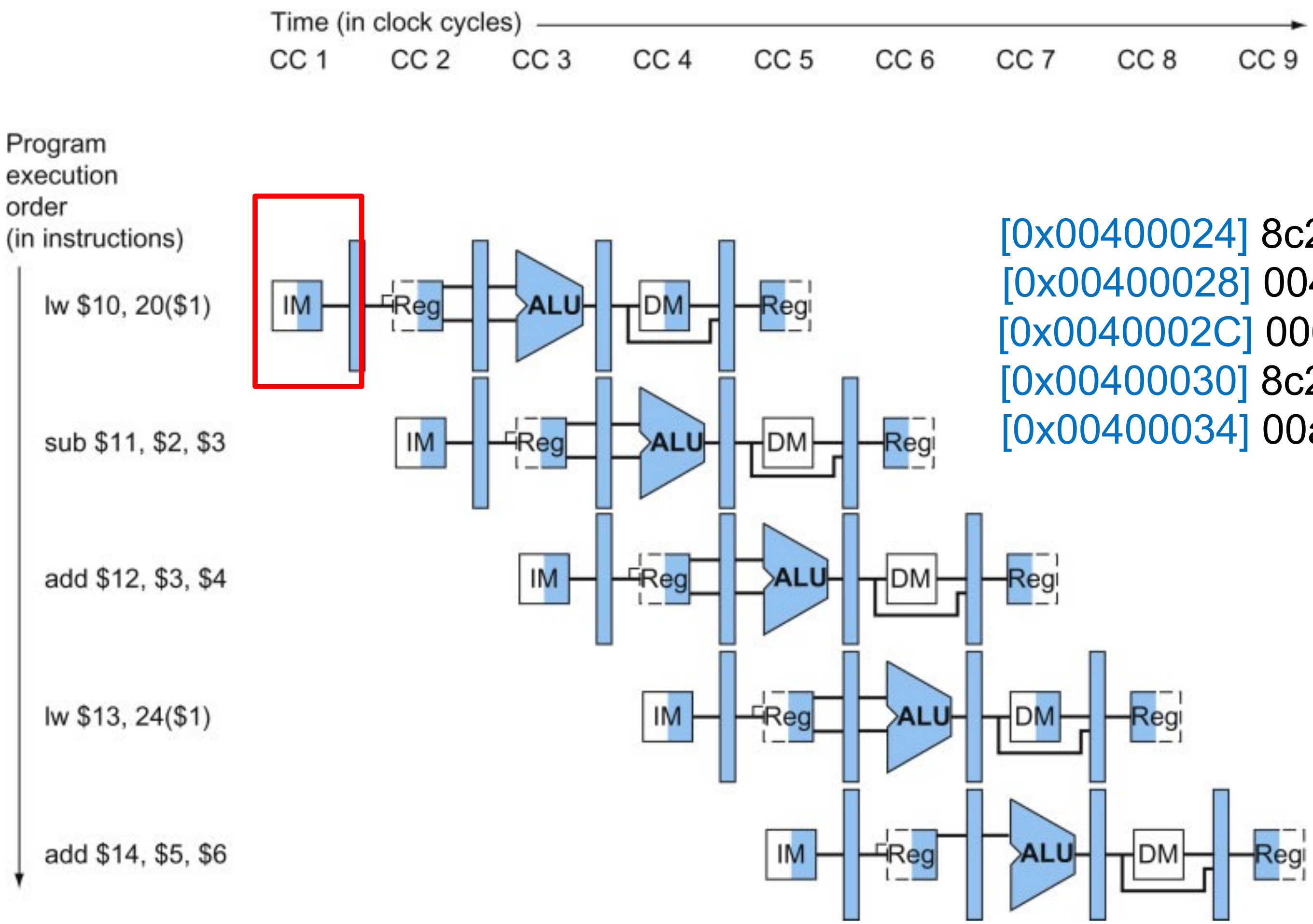
WB for Load at CC5

lw $10, 20($1)



Wrong register number

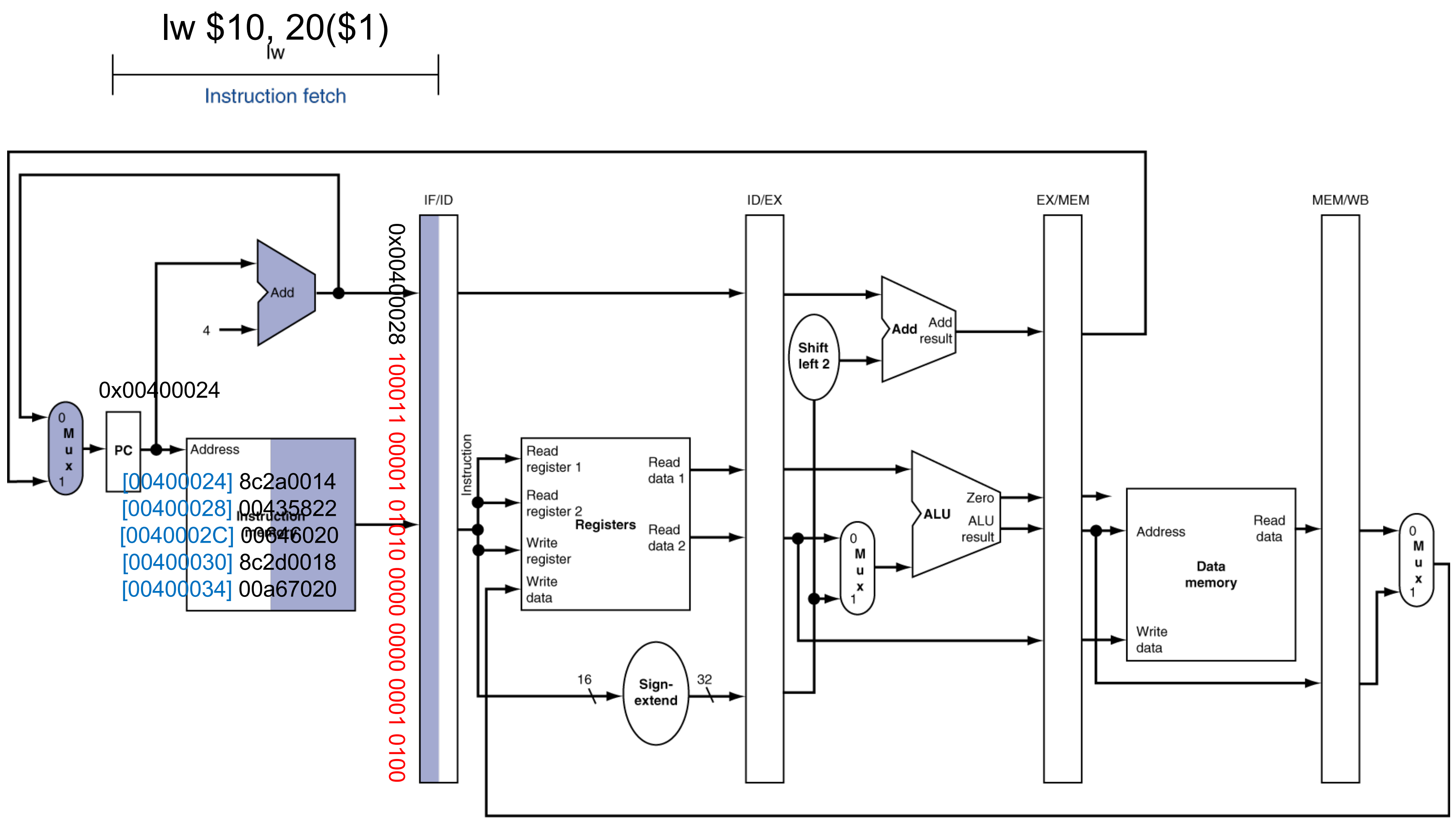
Detailed Diagram

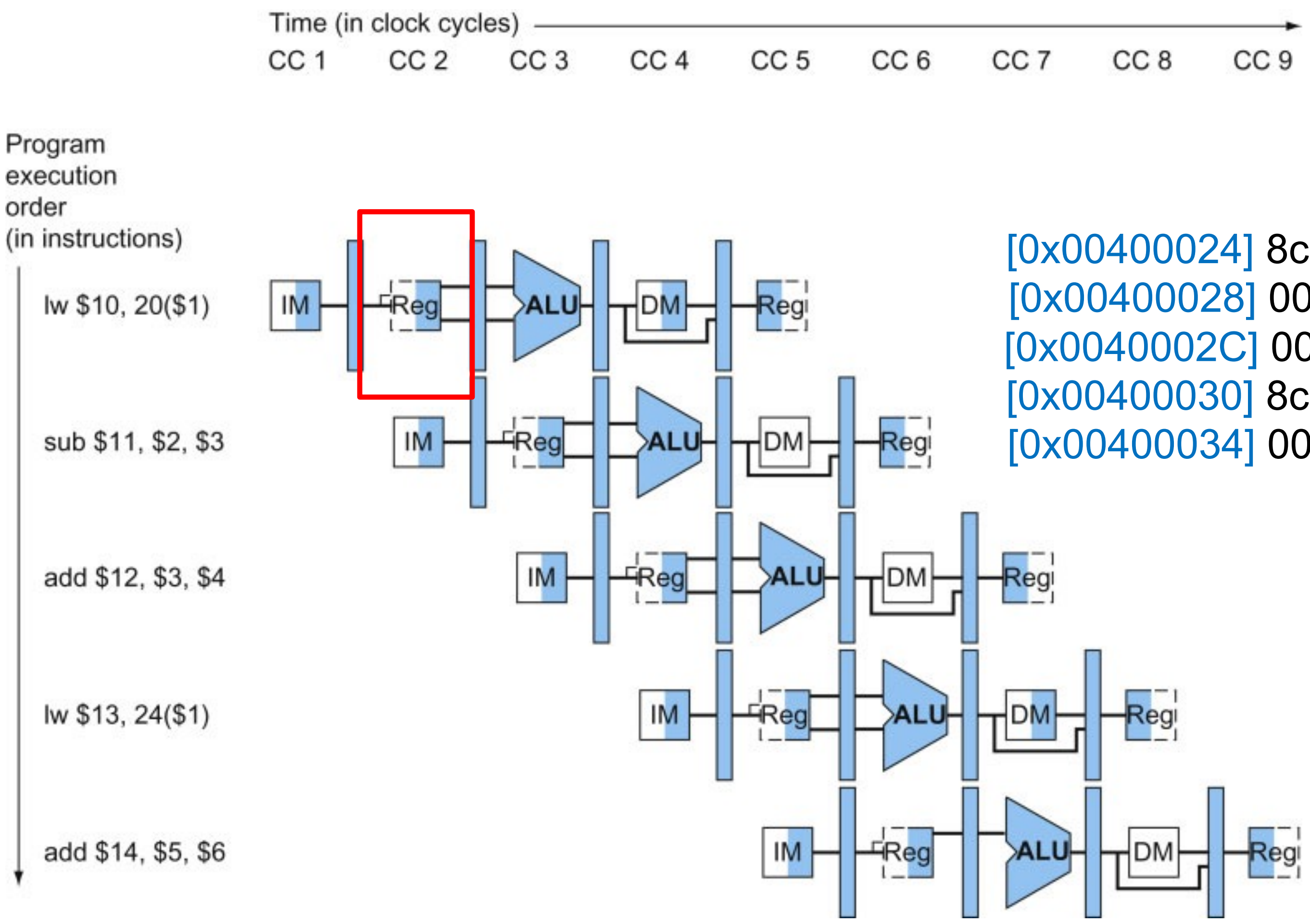


```
[0x00400024] 8c2a0014  lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018  lw
[0x00400034] 00a67020 add
```

IF for Load at CC1

lw $10, 20($1)

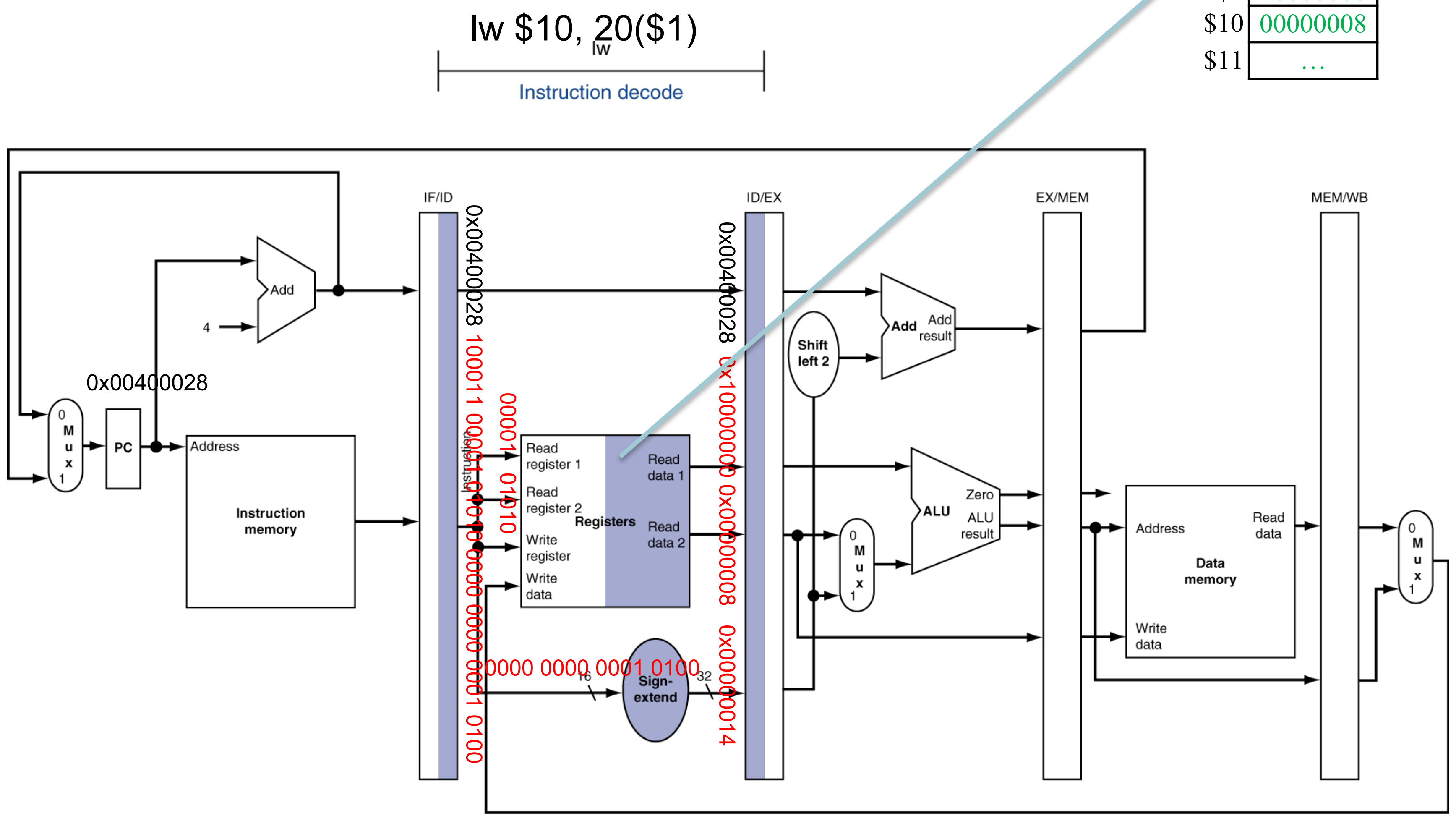


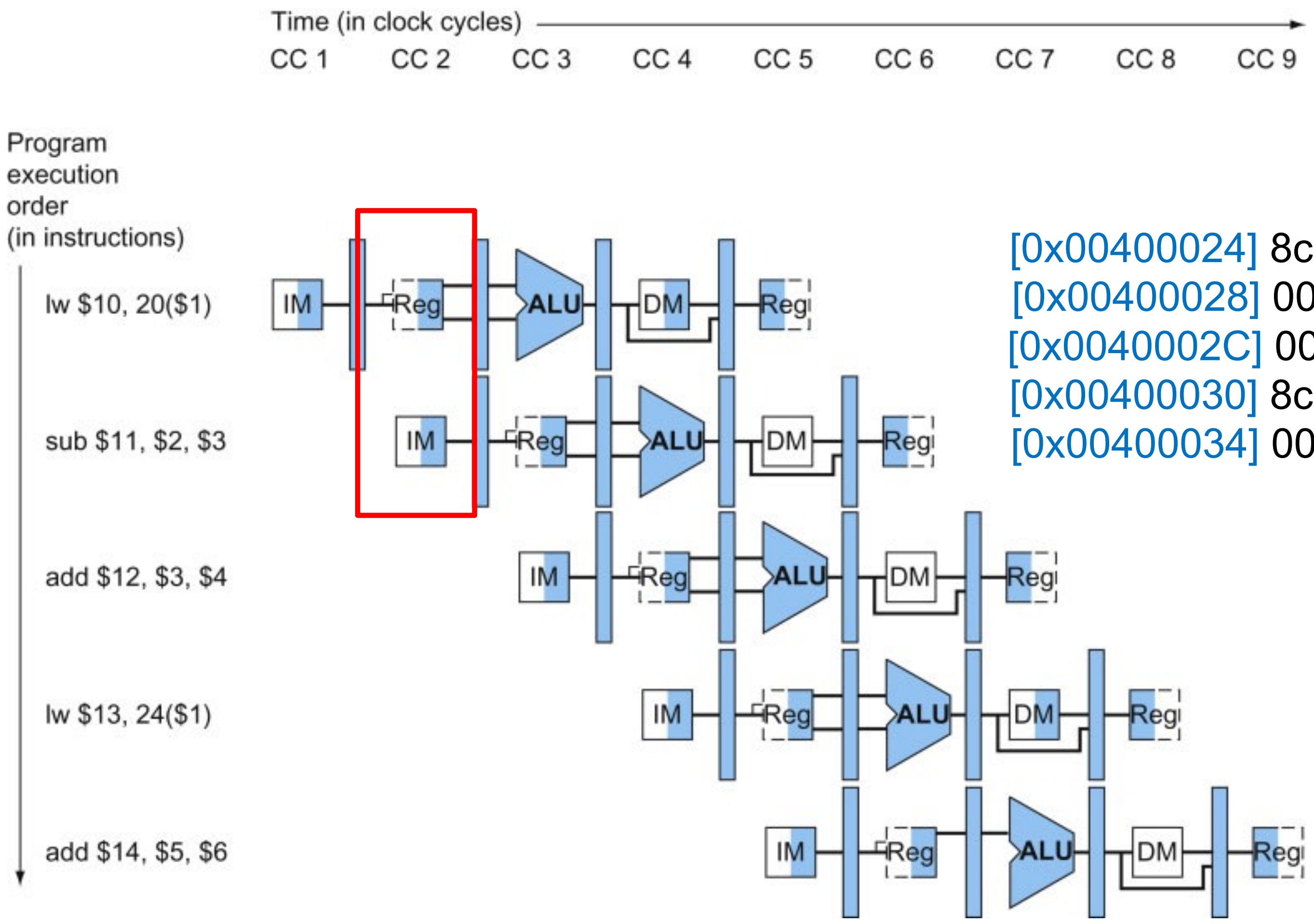


```
[0x00400024] 8c2a0014  lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018  lw
[0x00400034] 00a67020 add
```

ID for Load at CC2

$0	00000000
$1	10000000
$2	20000000
$3	30000000
$4	40000000
$10	00000008
$11	…



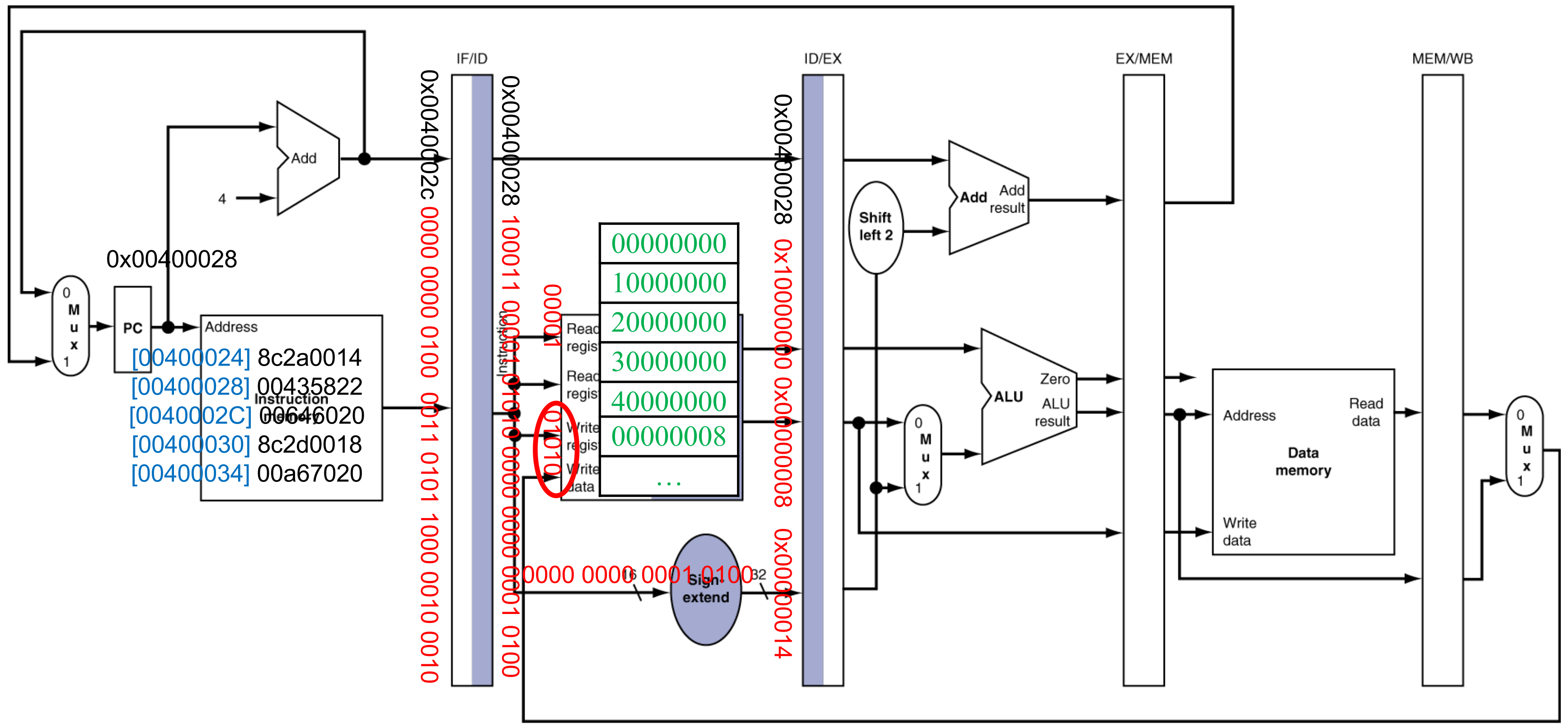


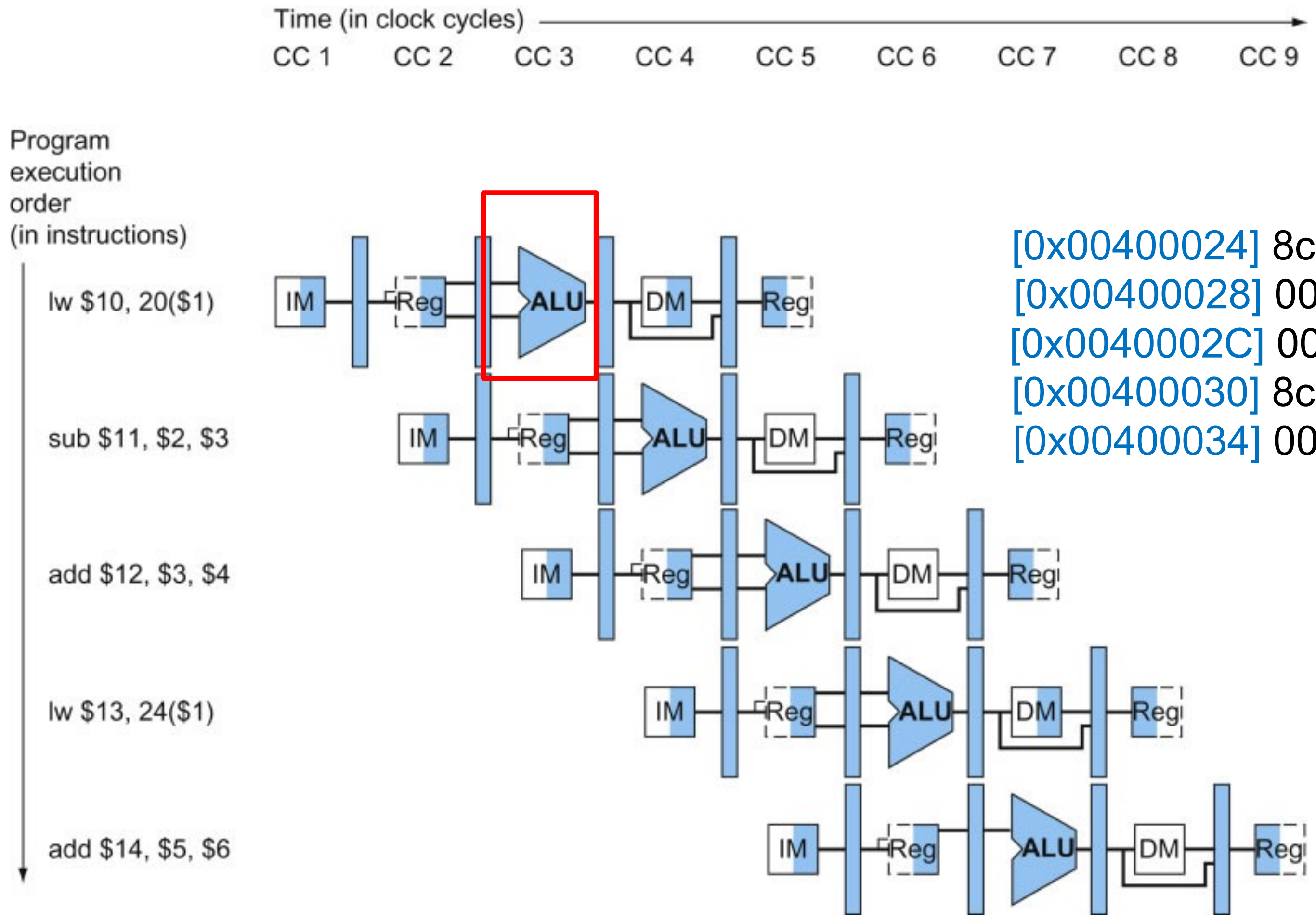
[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

ID for Load at CC2 (and more)

← sub $11,$2,$3 instruction fetch →

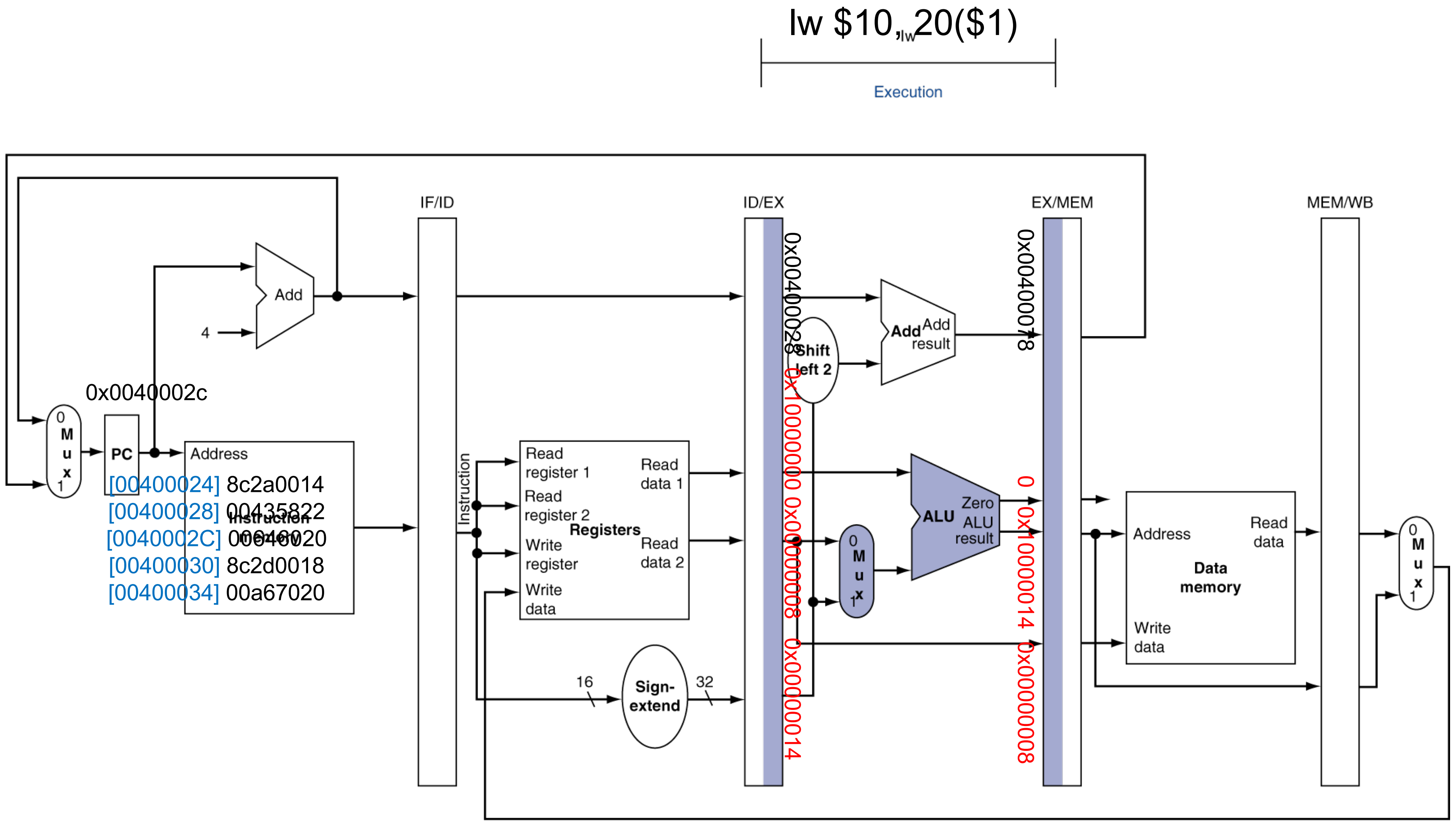
lw $10,20($1)

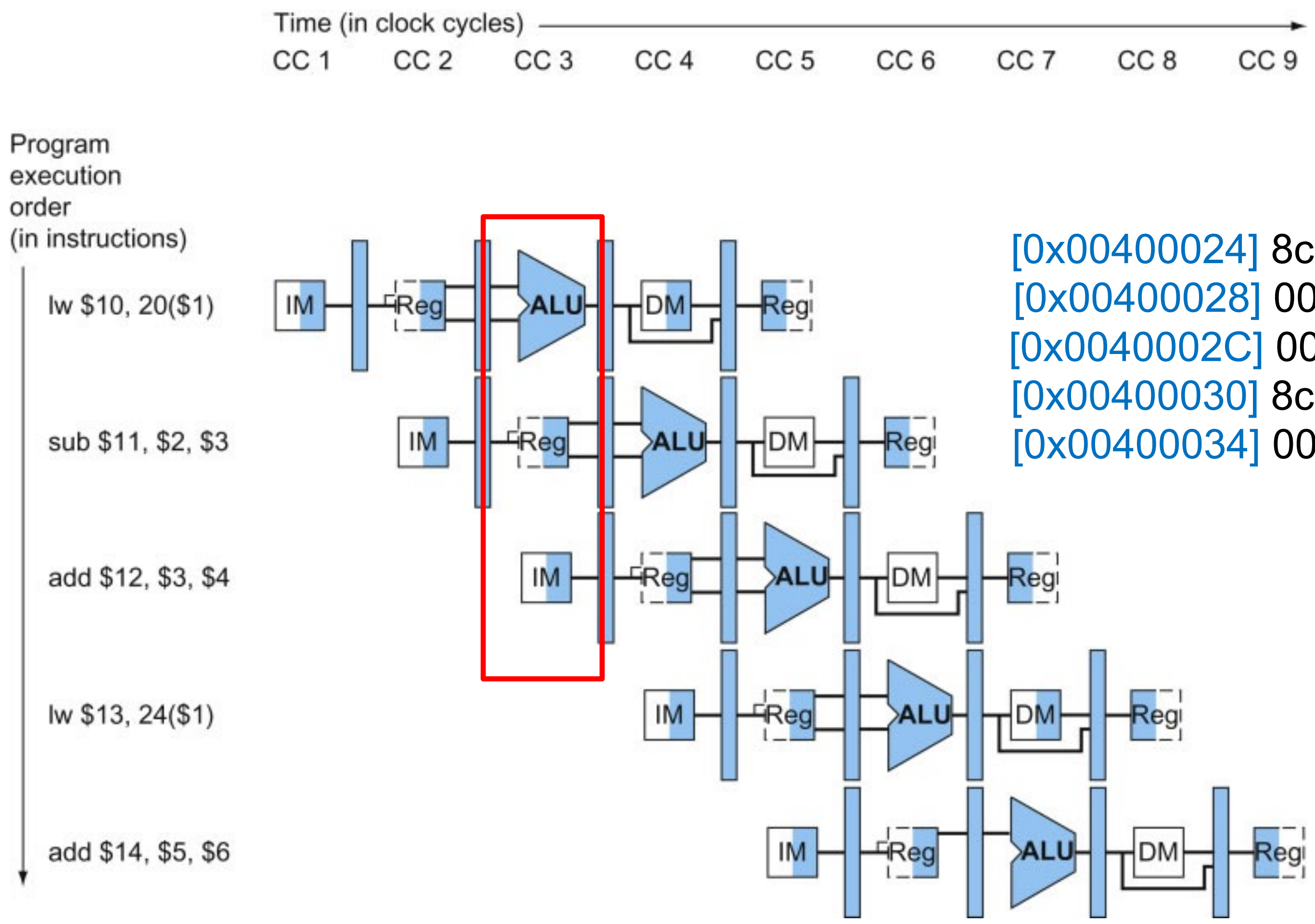




[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

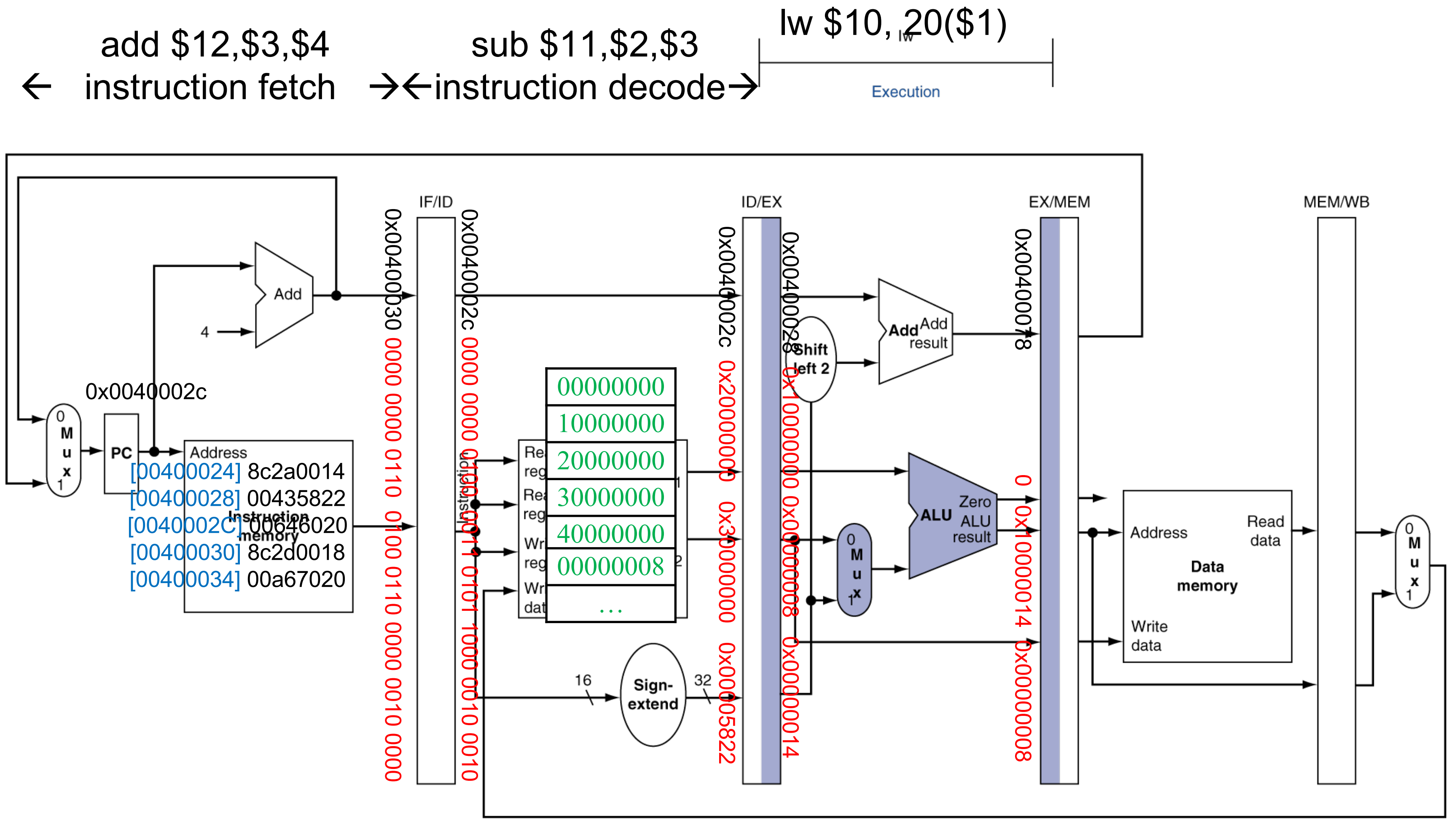
EX for Load at CC3

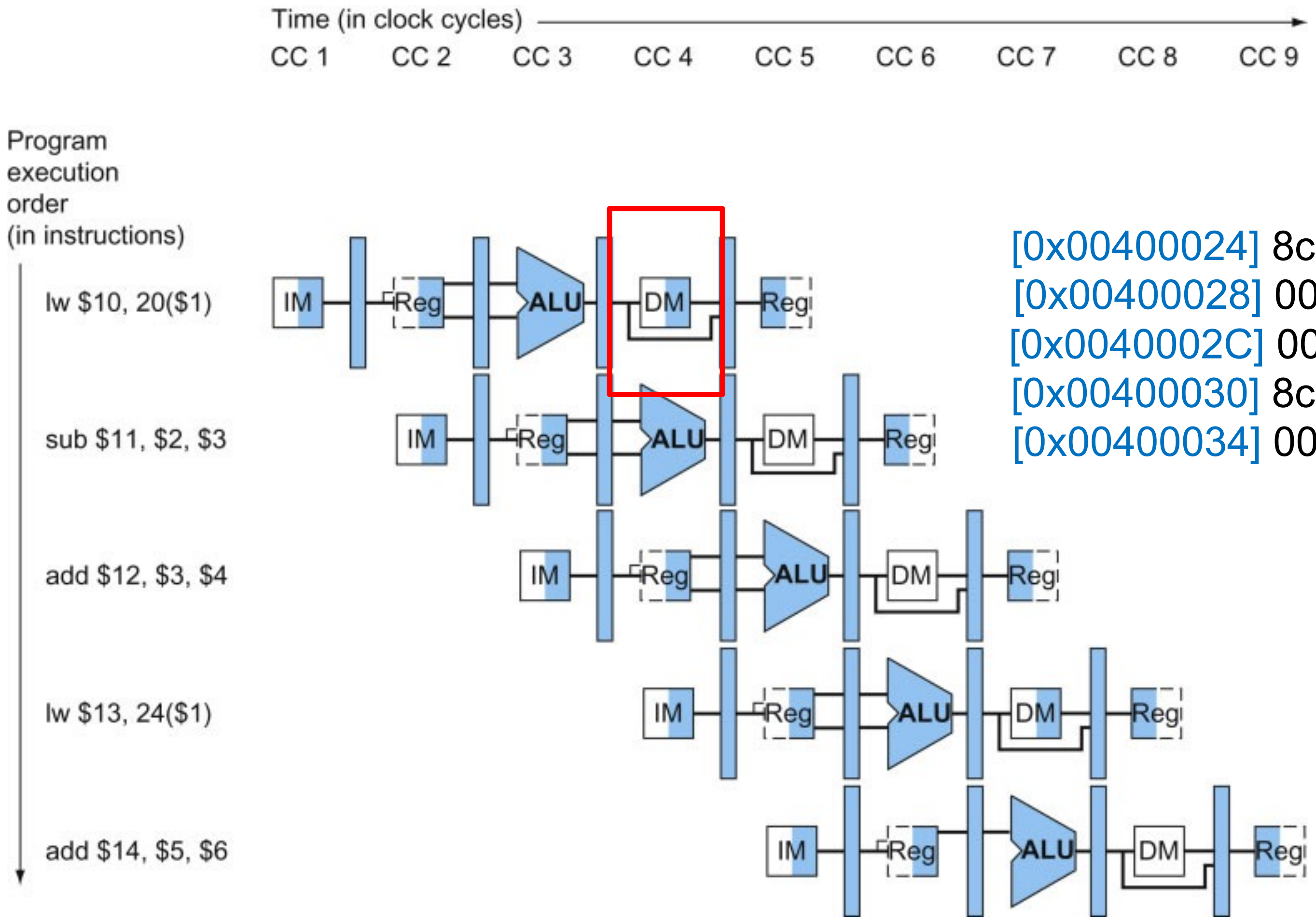




```
[0x00400024] 8c2a0014  lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018  lw
[0x00400034] 00a67020 add
```

EX for Load at CC3 (and more)

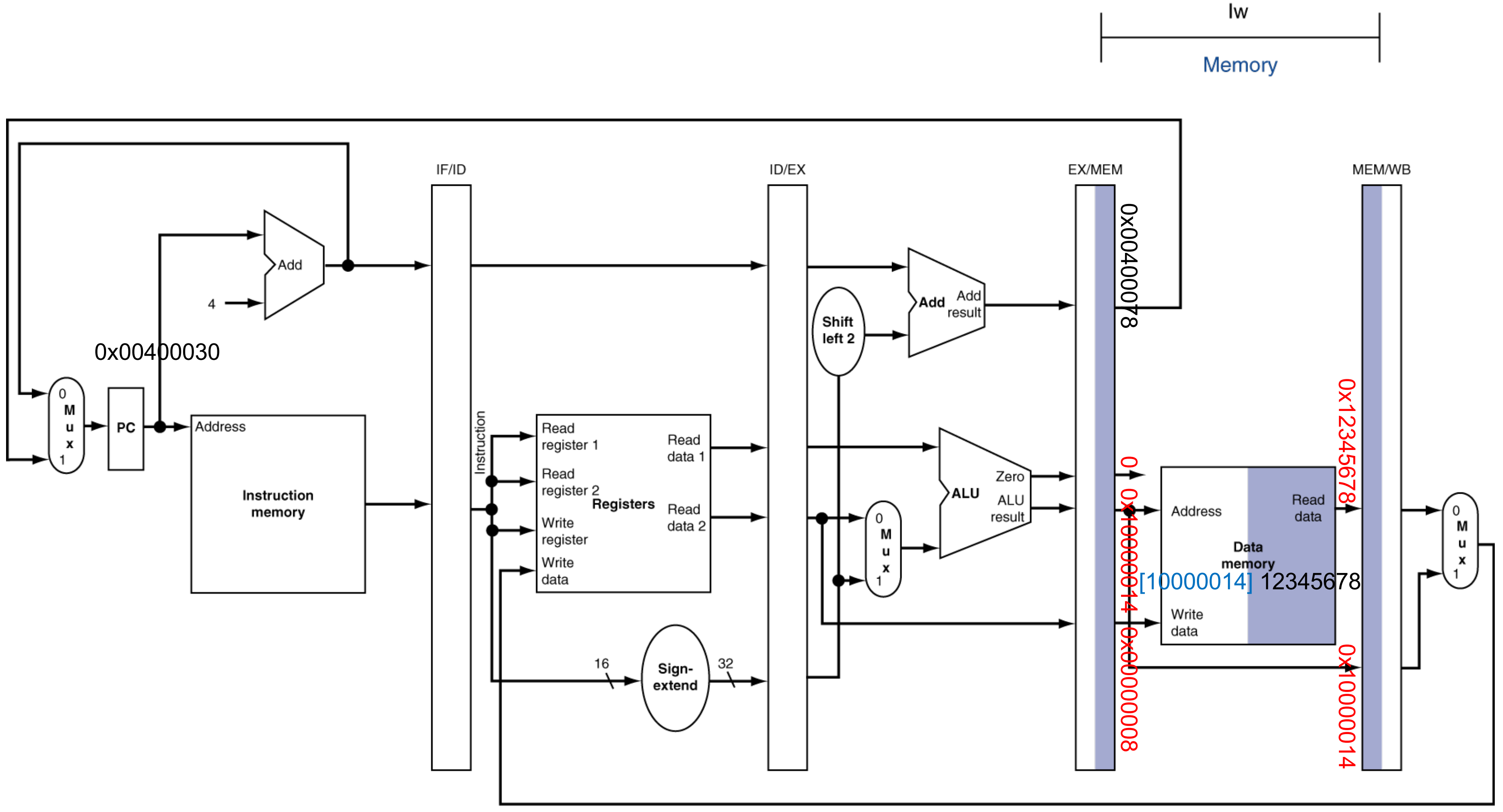


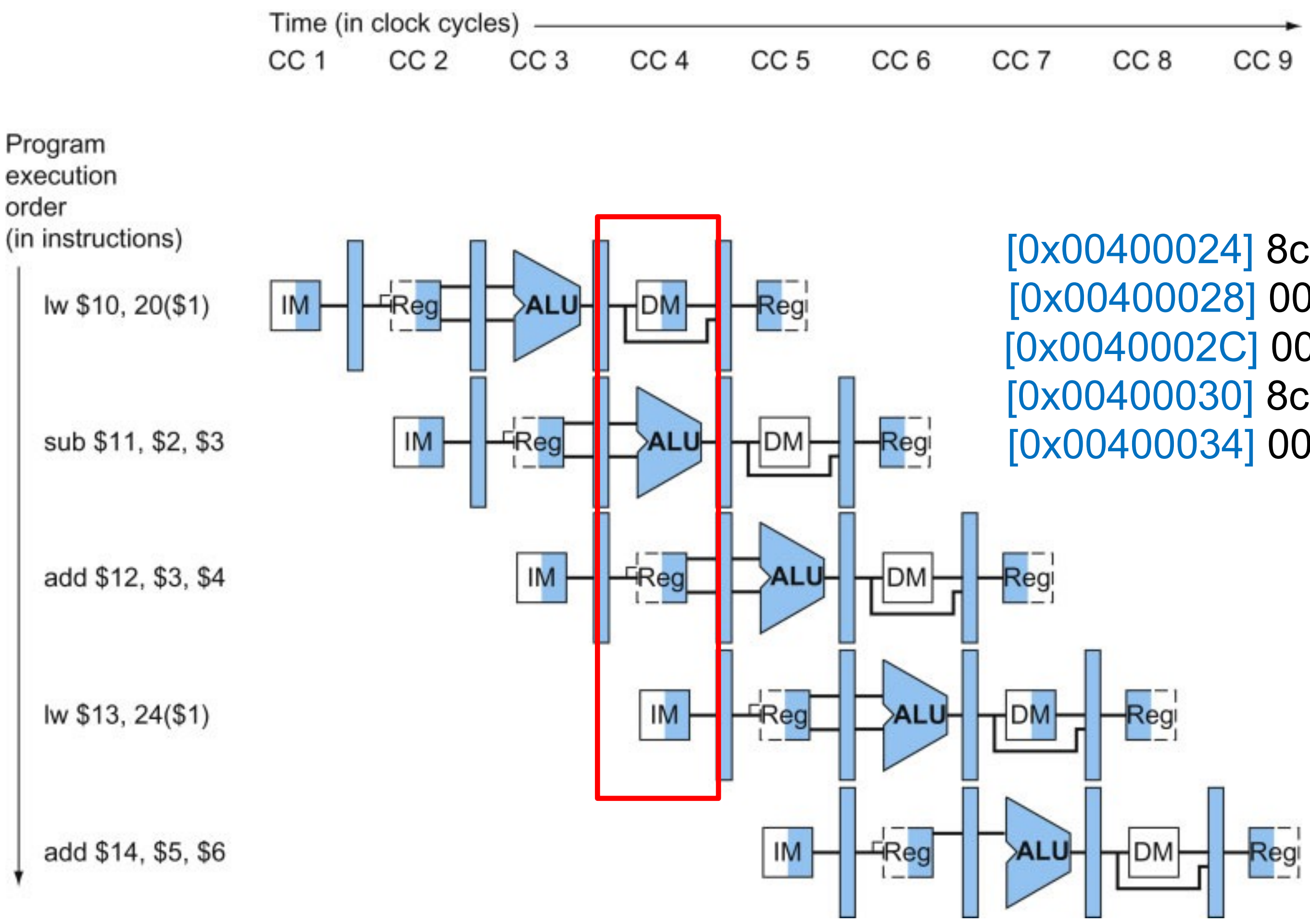


```
[0x00400024] 8c2a0014  lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018  lw
[0x00400034] 00a67020 add
```

MEM for Load at CC4

lw $10, 20($1)





[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

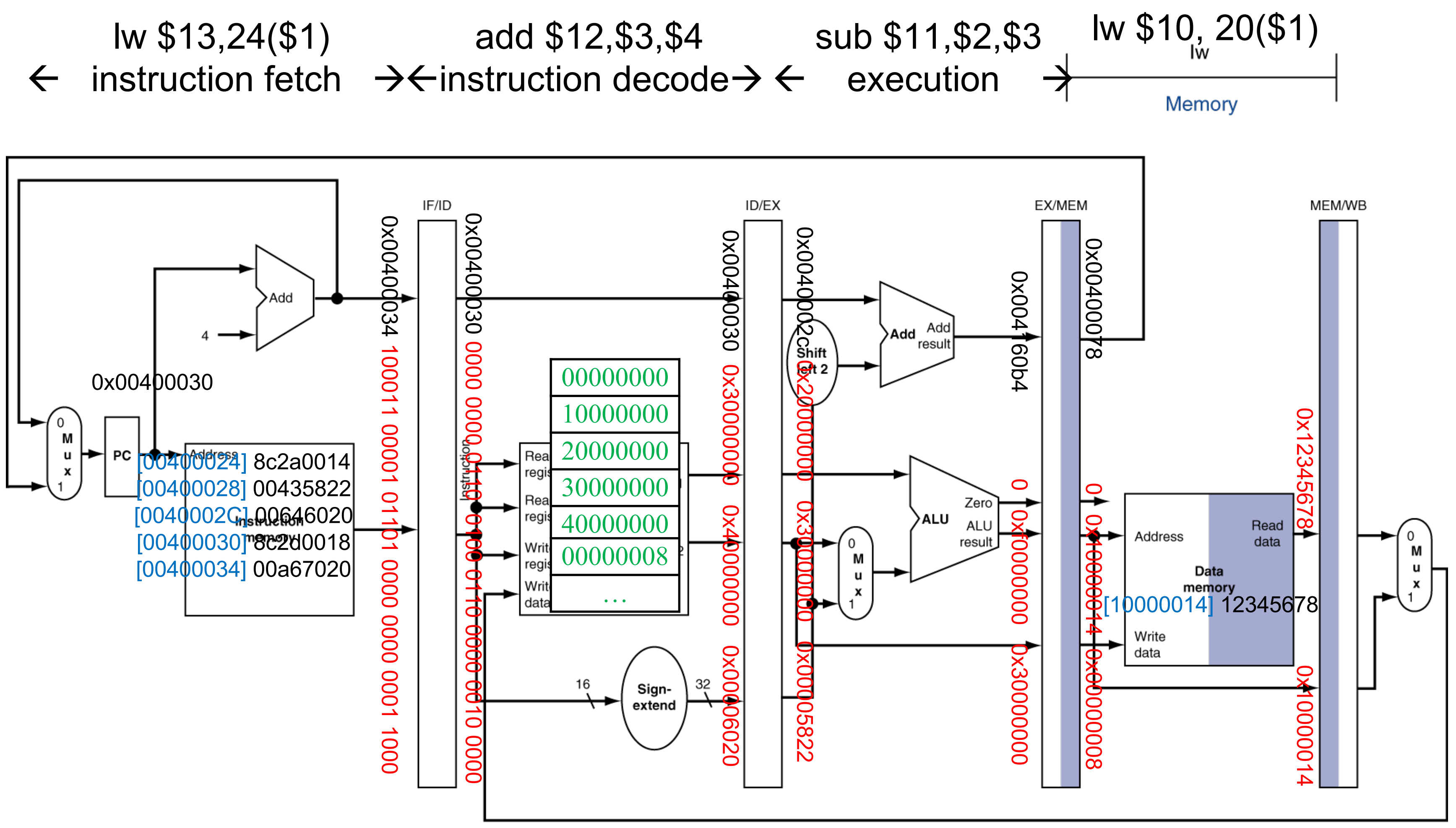
MEM for Load at CC4 (and more)

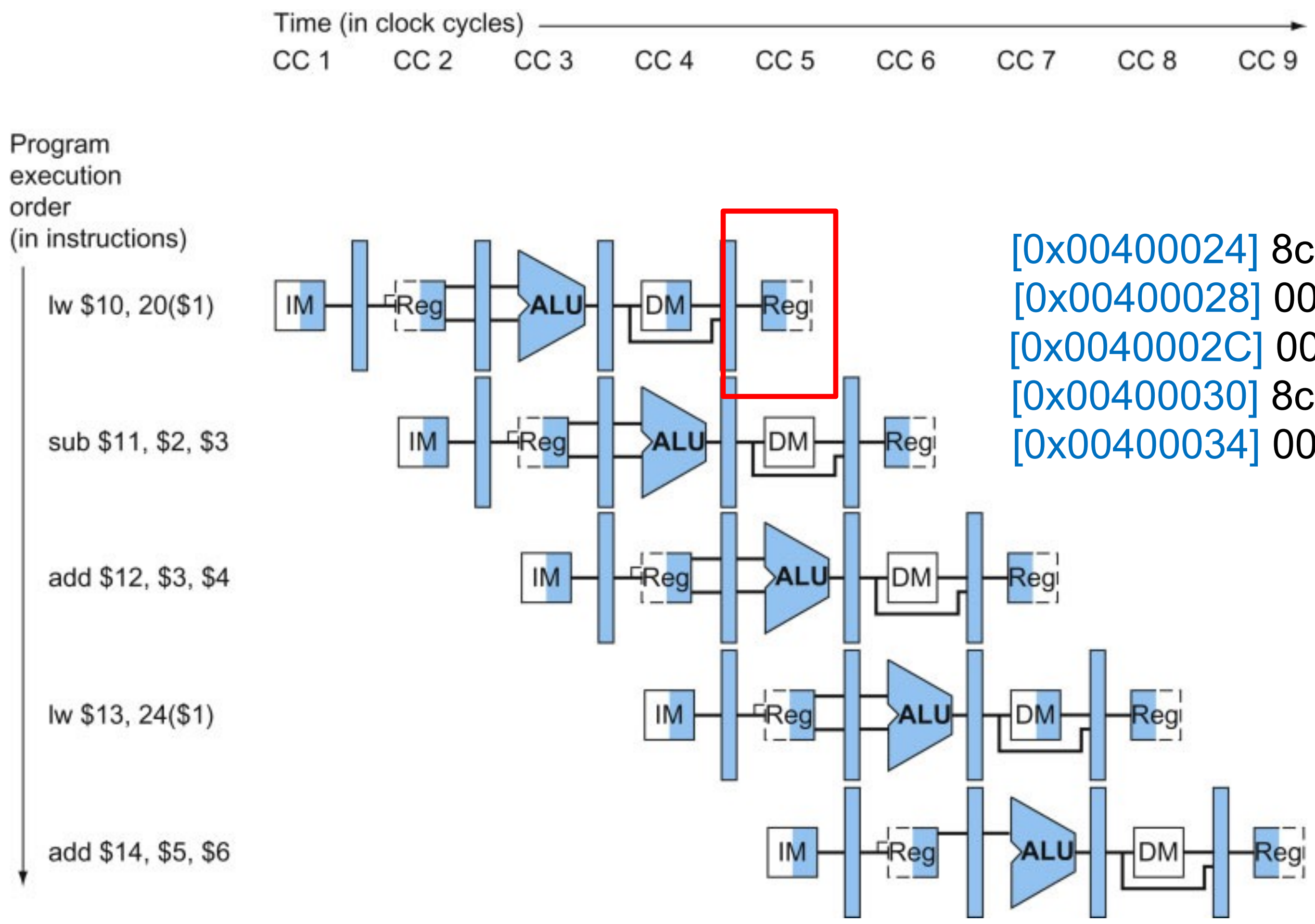
lw $13,24($1) ← instruction fetch →

add $12,$3,$4 ←instruction decode→

sub $11,$2,$3 ← execution →

lw $10, 20($1)





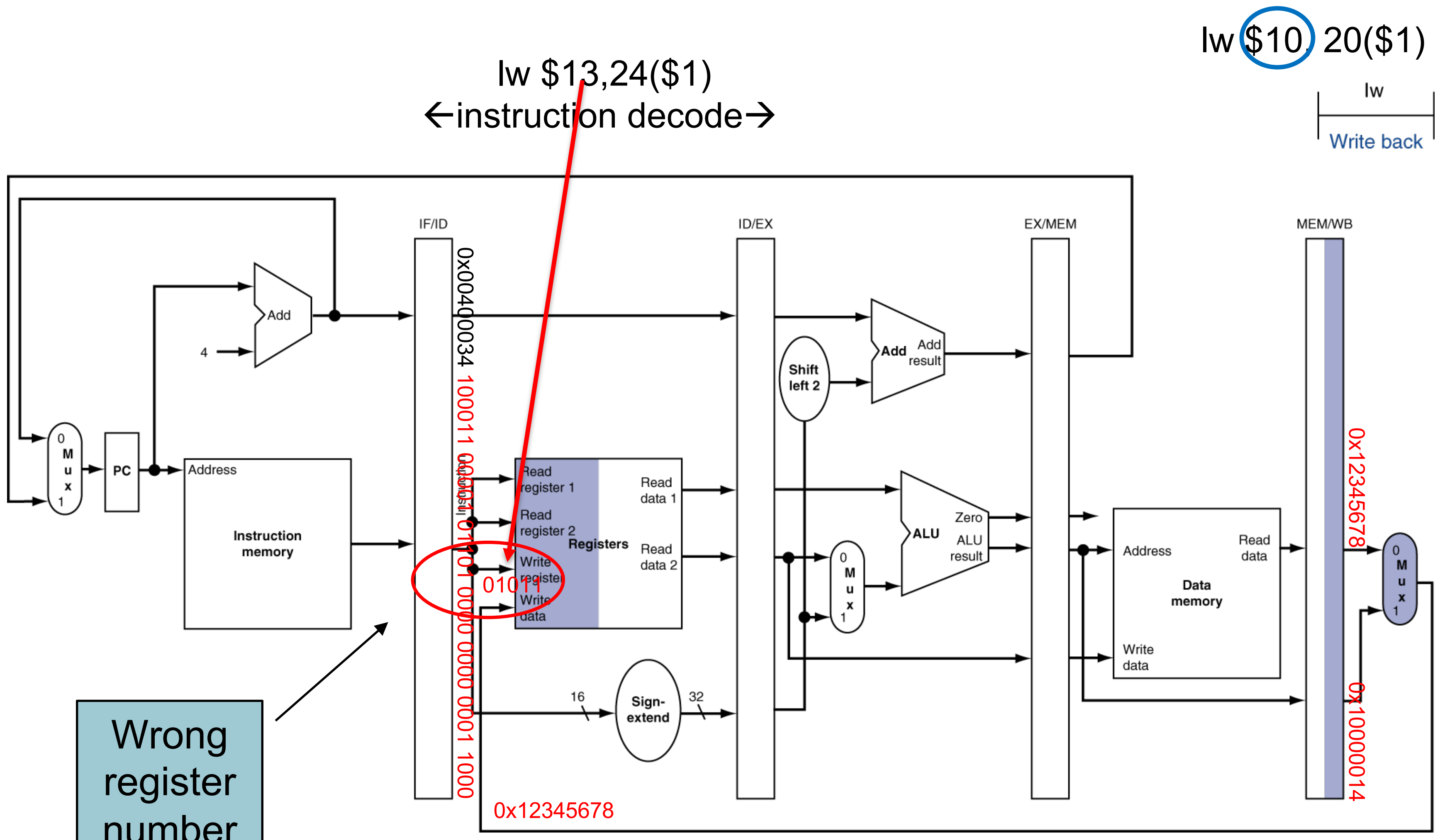
[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

WB for Load at CC5

lw $13,24($1)

←instruction decode→

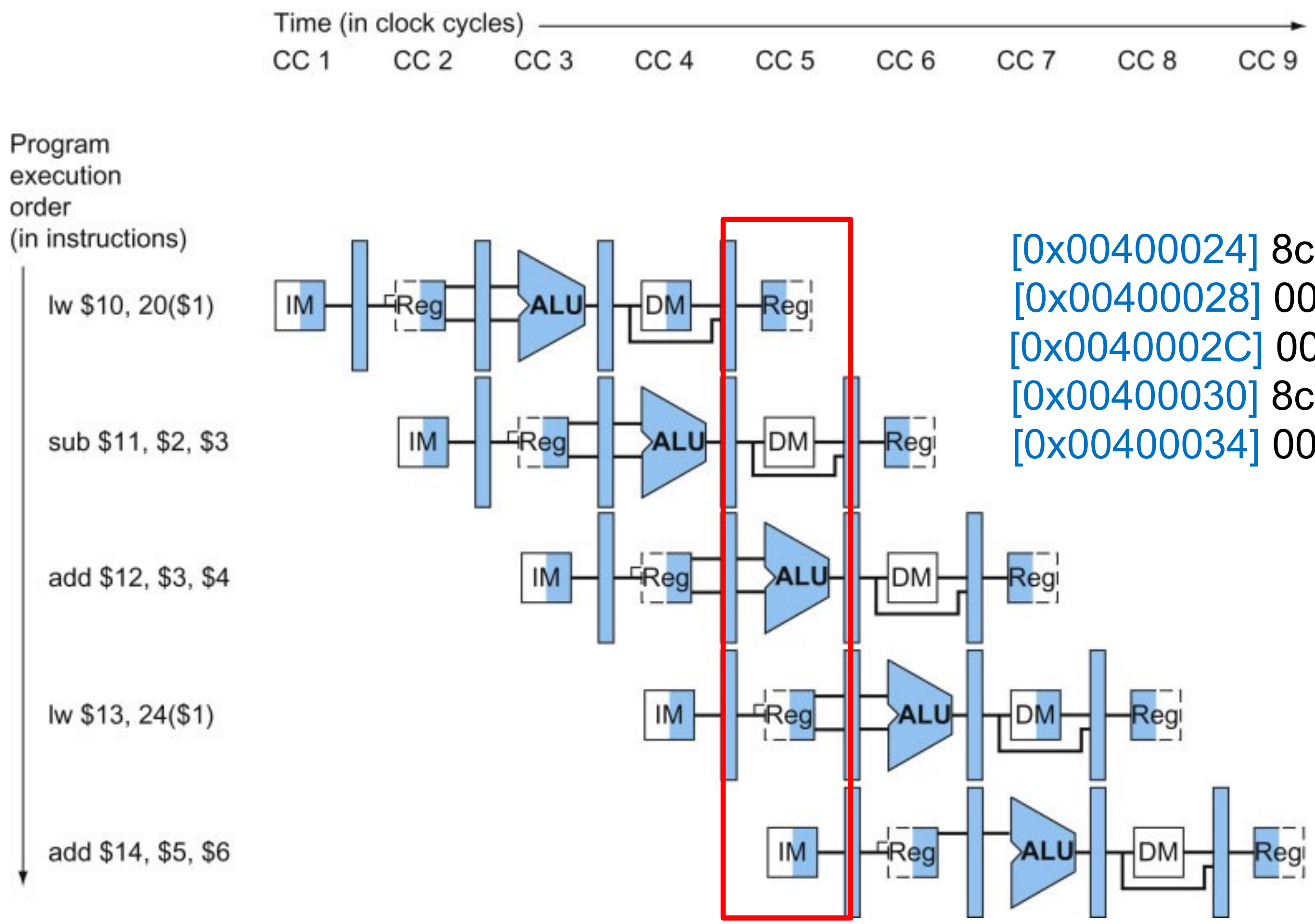
lw $10, 20($1)



Wrong register number

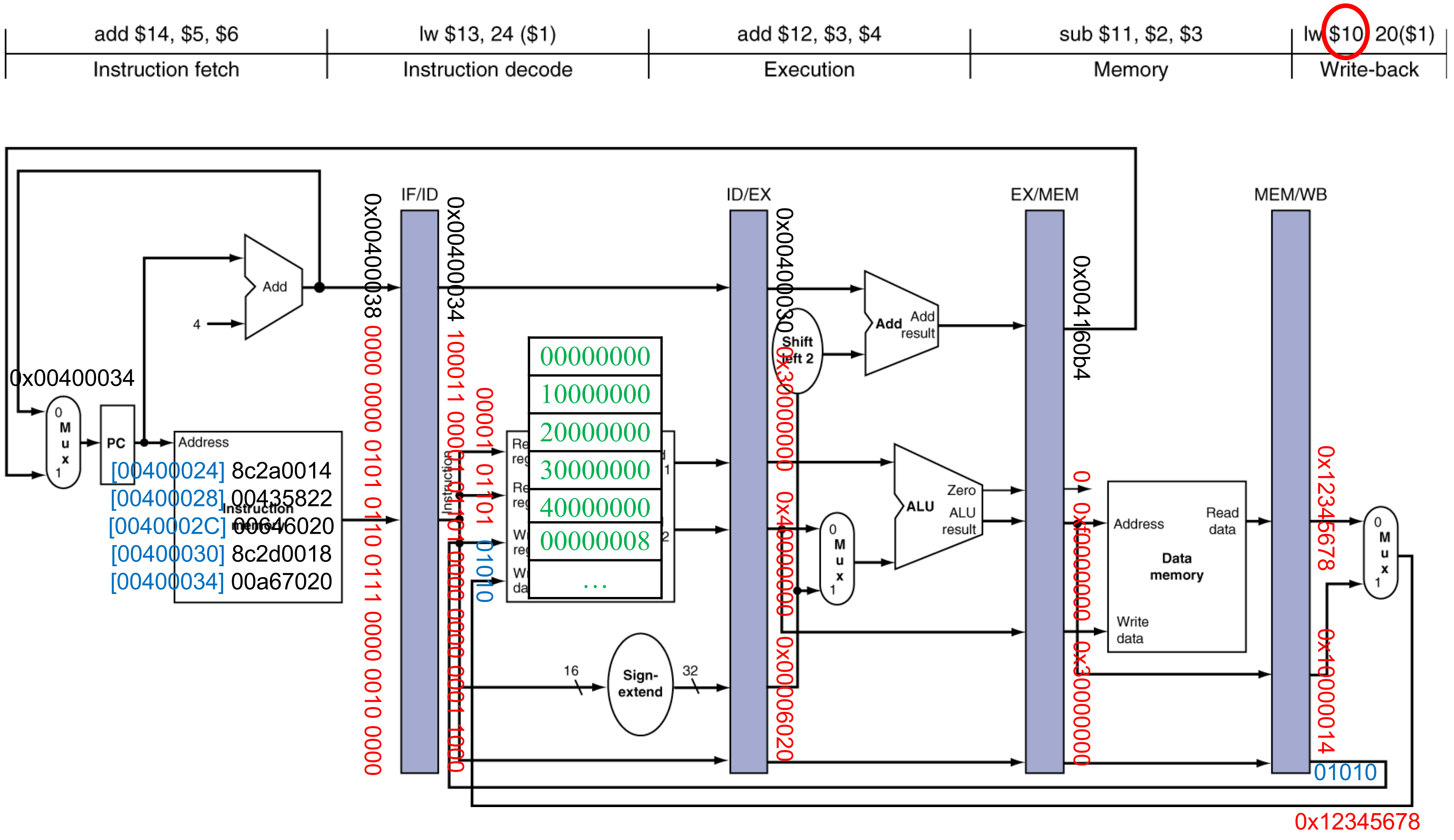
Multi-Cycle Pipeline Diagram

- resource usage



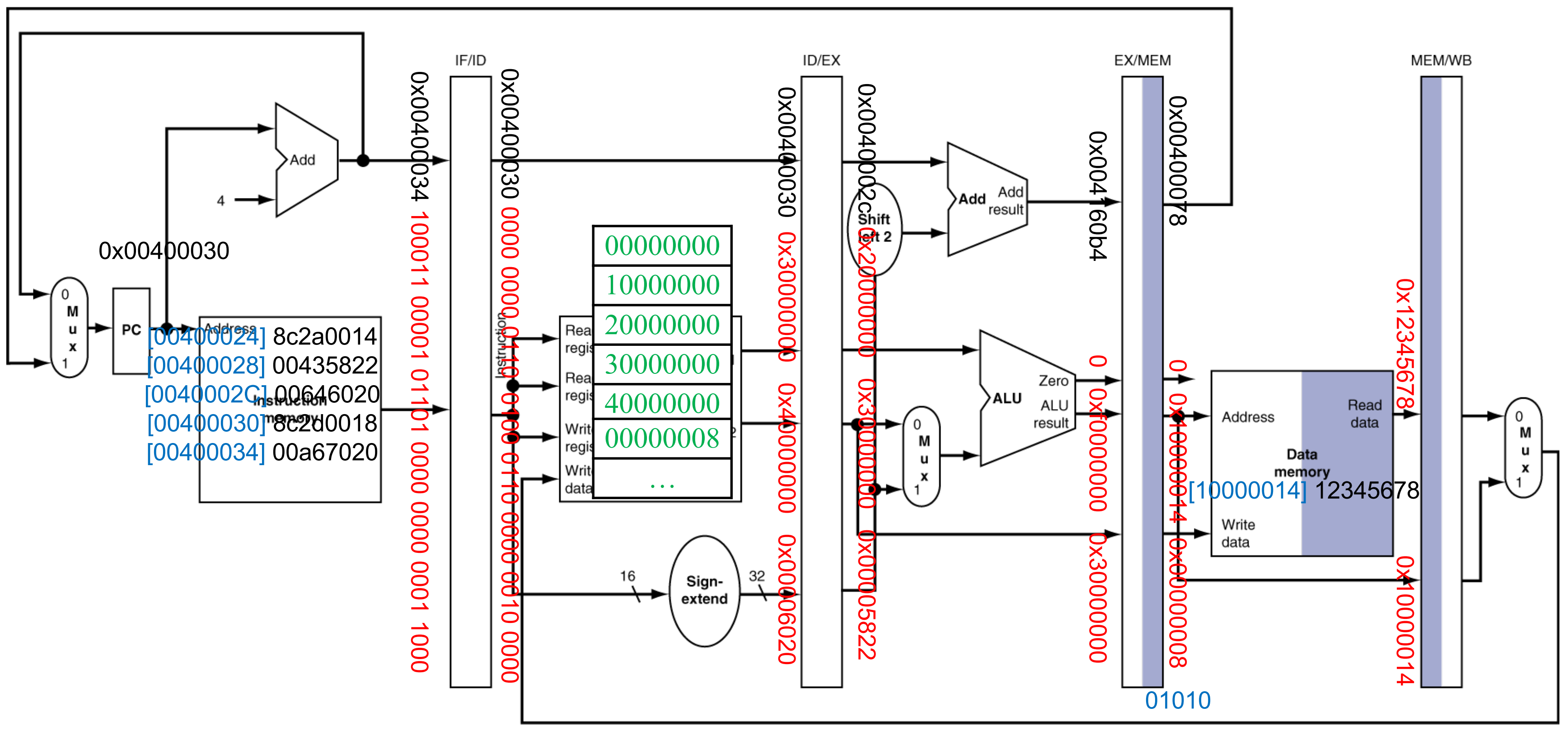
[0x00400024] 8c2a0014 lw
[0x00400028] 00435822 sub
[0x0040002C] 00646020 add
[0x00400030] 8c2d0018 lw
[0x00400034] 00a67020 add

at CC5

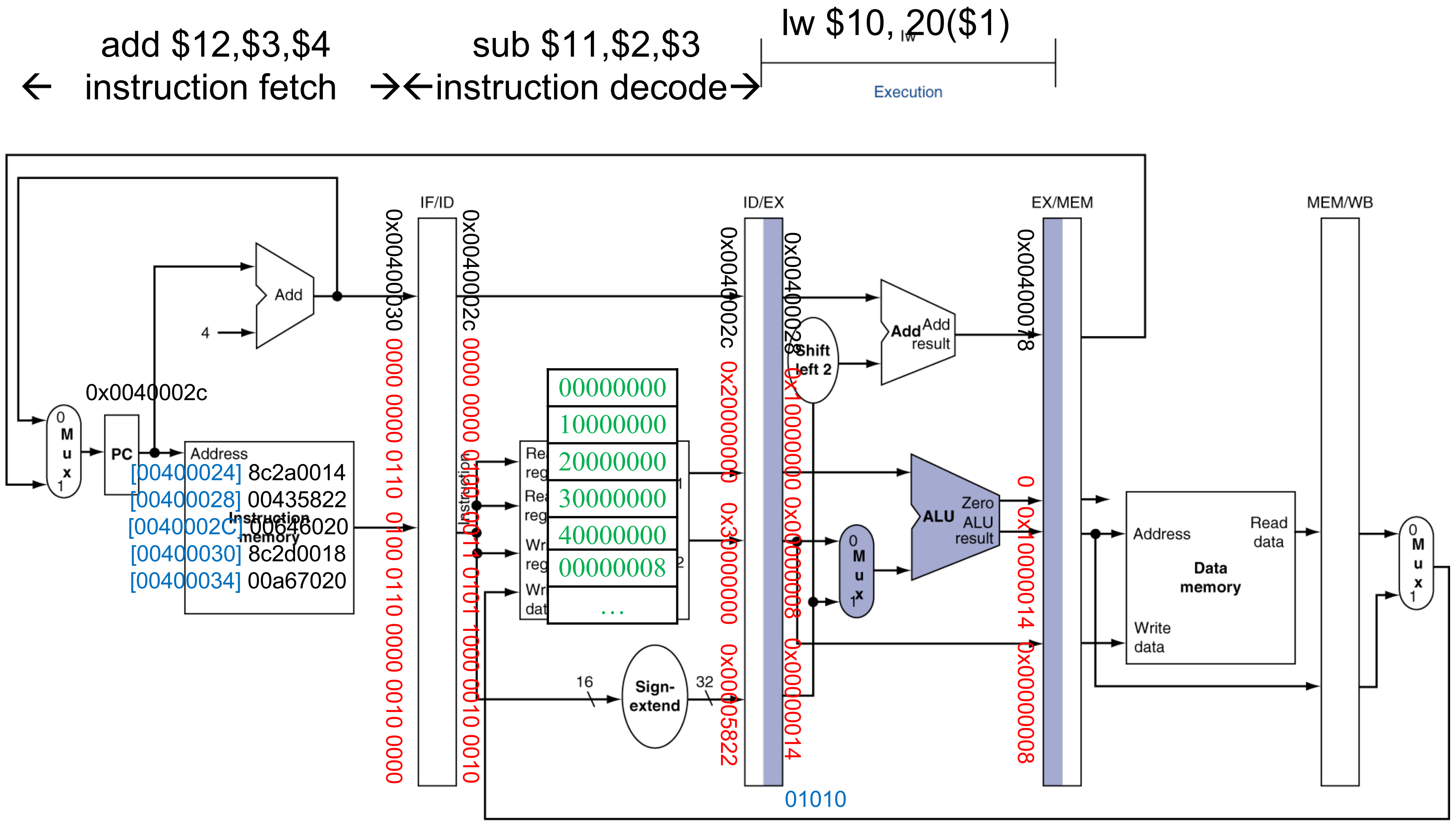

add $14, $5, $6
Instruction fetch
lw $13, 24 ($1)
Instruction decode
add $12, $3, $4
Execution
sub $11, $2, $3
Memory
lw $10 20($1)
Write-back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0x00400034
PC
Address
[00400024] 8c2a0014
[00400028] 00435822
[0040002C] 00646020
[00400030] 8c2d0018
[00400034] 00a67020
Instruction memory
0x00400038 0000 0000 0101 0110 0111 0000 0010 0000
0x00400034 100011 00001 01101 0000 0000 0001 1000
00001 01101 01010
00000000
10000000
20000000
30000000
40000000
00000008
...
Sign-extend
16
32
0x00400030 0x30000000 0x40000000 0x00006020
Shift left 2
Add Add result
ALU
Zero
ALU result
0x004160b4
0 0xf0000000 0x30000000
Address
Read data
Data memory
Write data
0x12345678
0x10000014
01010
0x12345678

at CC4

lw $13,24($1)	add $12,$3,$4	sub $11,$2,$3	lw $10, 20($1)
← instruction fetch →	← instruction decode →	← execution →	lw Memory



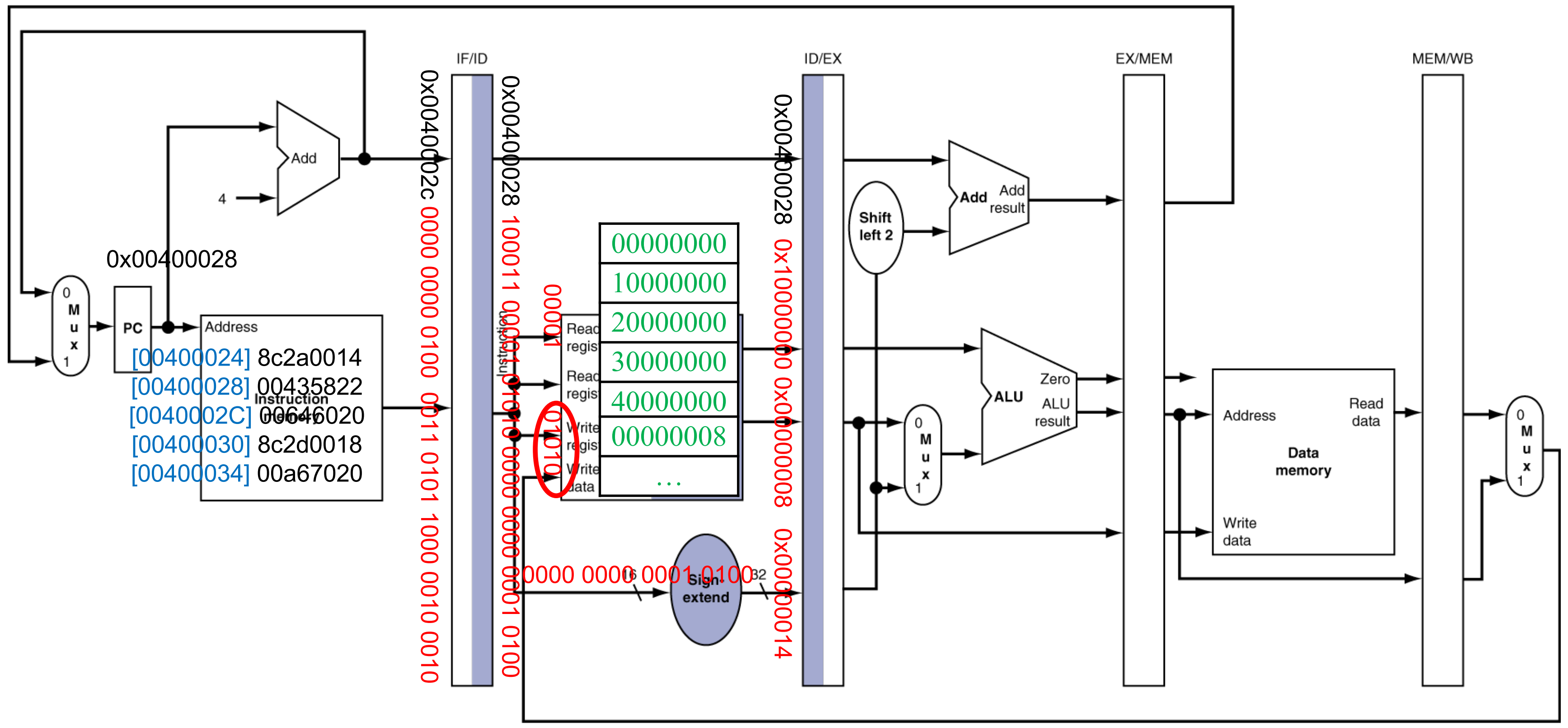
at CC3


add $12,$3,$4
← instruction fetch →
sub $11,$2,$3
← instruction decode →
lw $10, 20($1)
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0x0040002c
PC
Address
[00400024] 8c2a0014
[00400028] 00435822
[0040002C] 00646020
[00400030] 8c2d0018
[00400034] 00a67020
Instruction memory
0x00400030
0000 0000 0110 0100 0110 0000 0010 0000
0x0040002c
0000 0000 0100 0011 0101 1000 0010 0010
00000000
10000000
20000000
30000000
40000000
00000008
...
Sign-extend
16
32
0x0040002c
0x20000000
0x30000000
0x00005822
0x0040002c
0x10000000
0x00000008
0x00000014
01010
Shift left 2
Add
Add result
ALU
Zero
ALU result
0x00400078
0
0x10000014
0x00000008
Address
Read data
Data memory
Write data
Mux

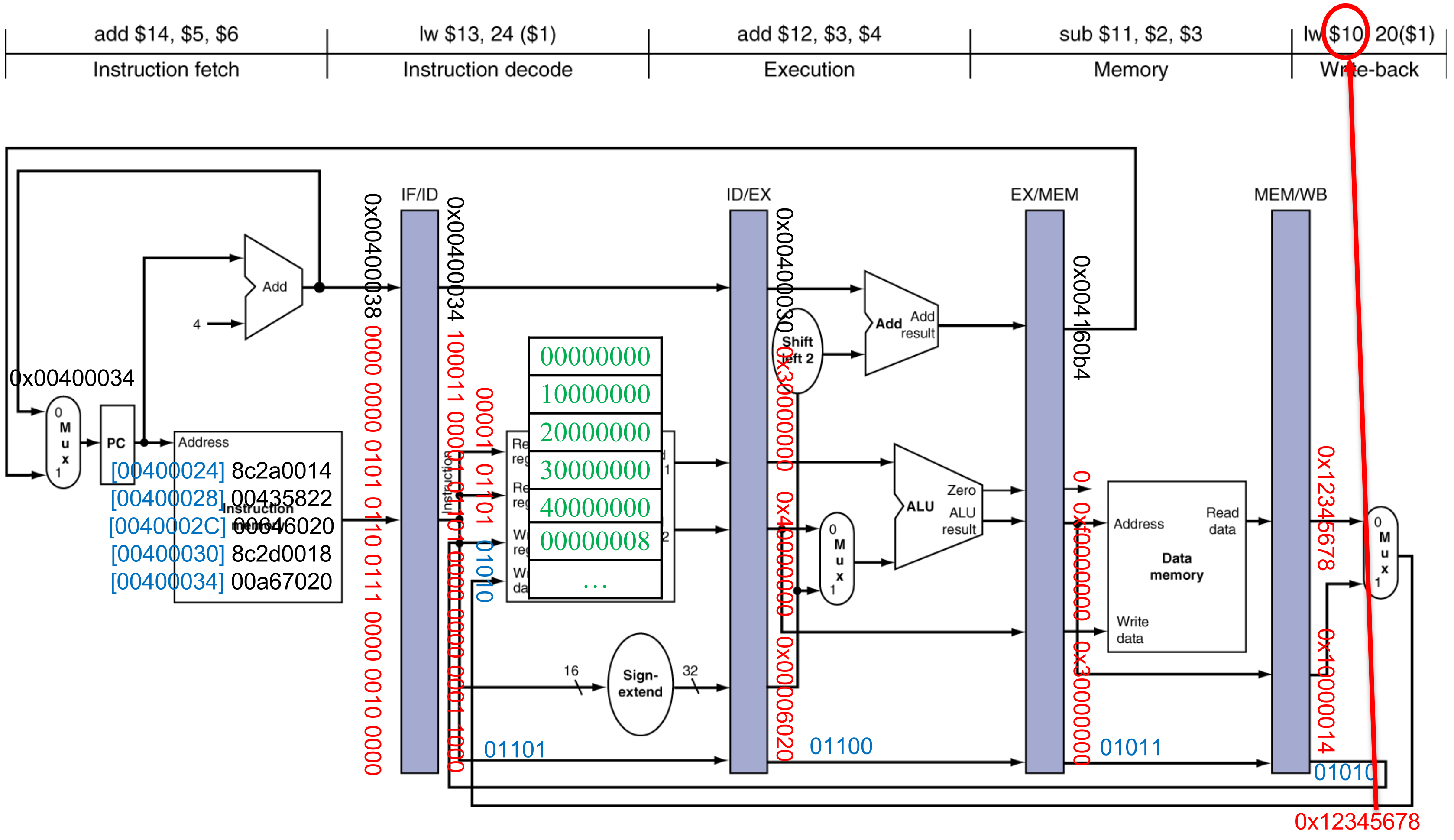
at CC2

sub $11,$2,$3
← instruction fetch →

lw $10,20($1)
Instruction decode

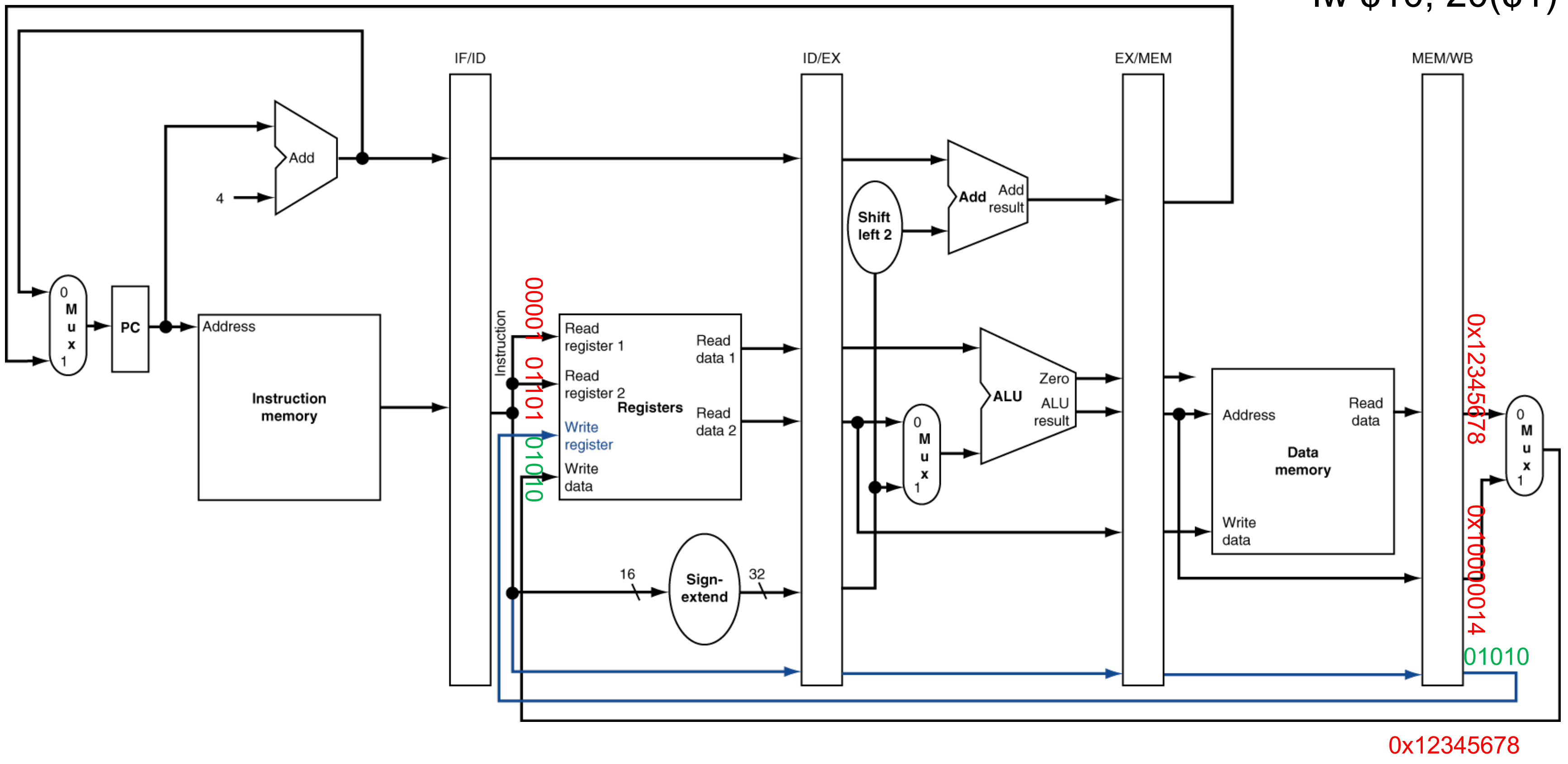


State of pipeline at CC5

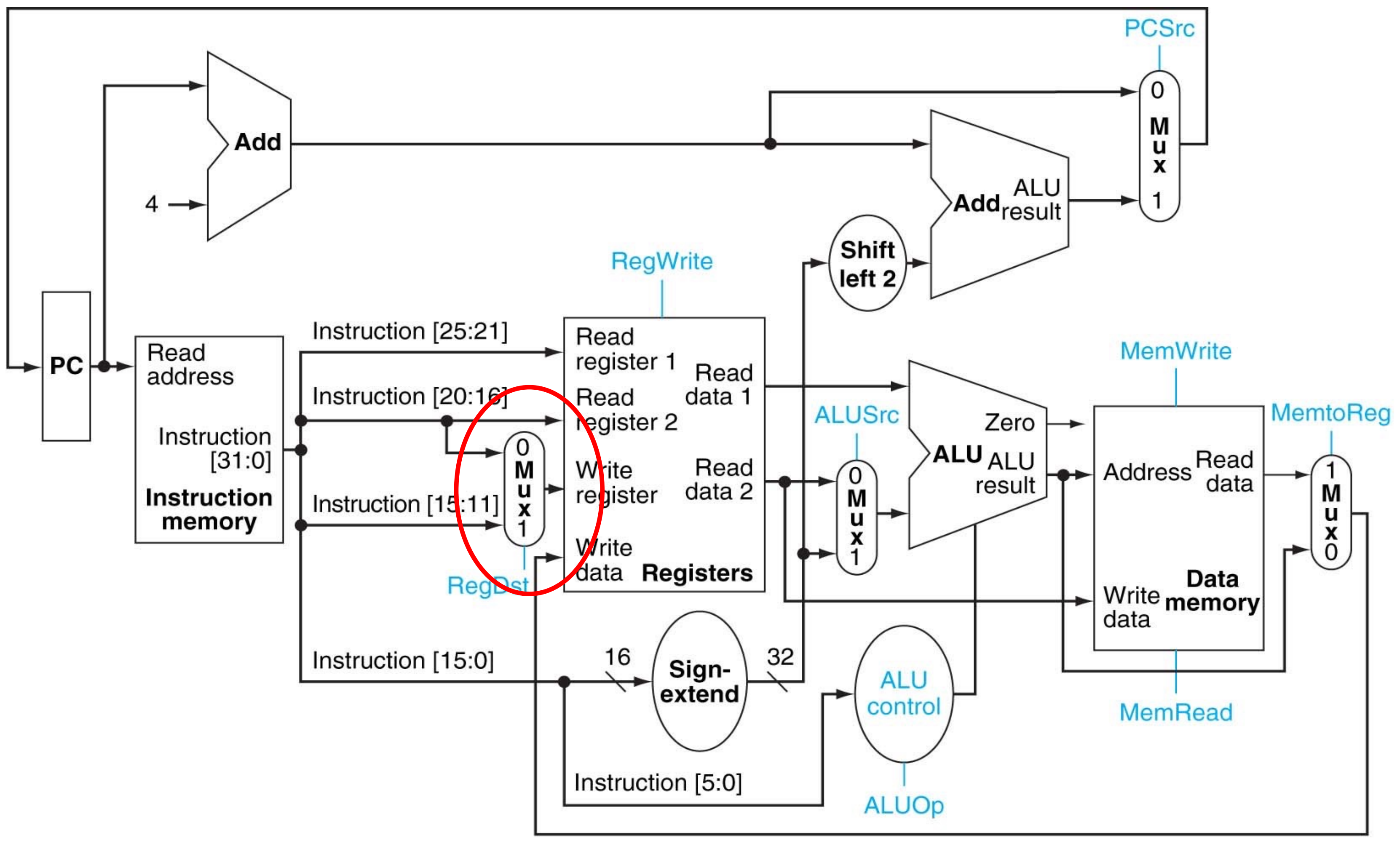


Corrected Datapath for Load

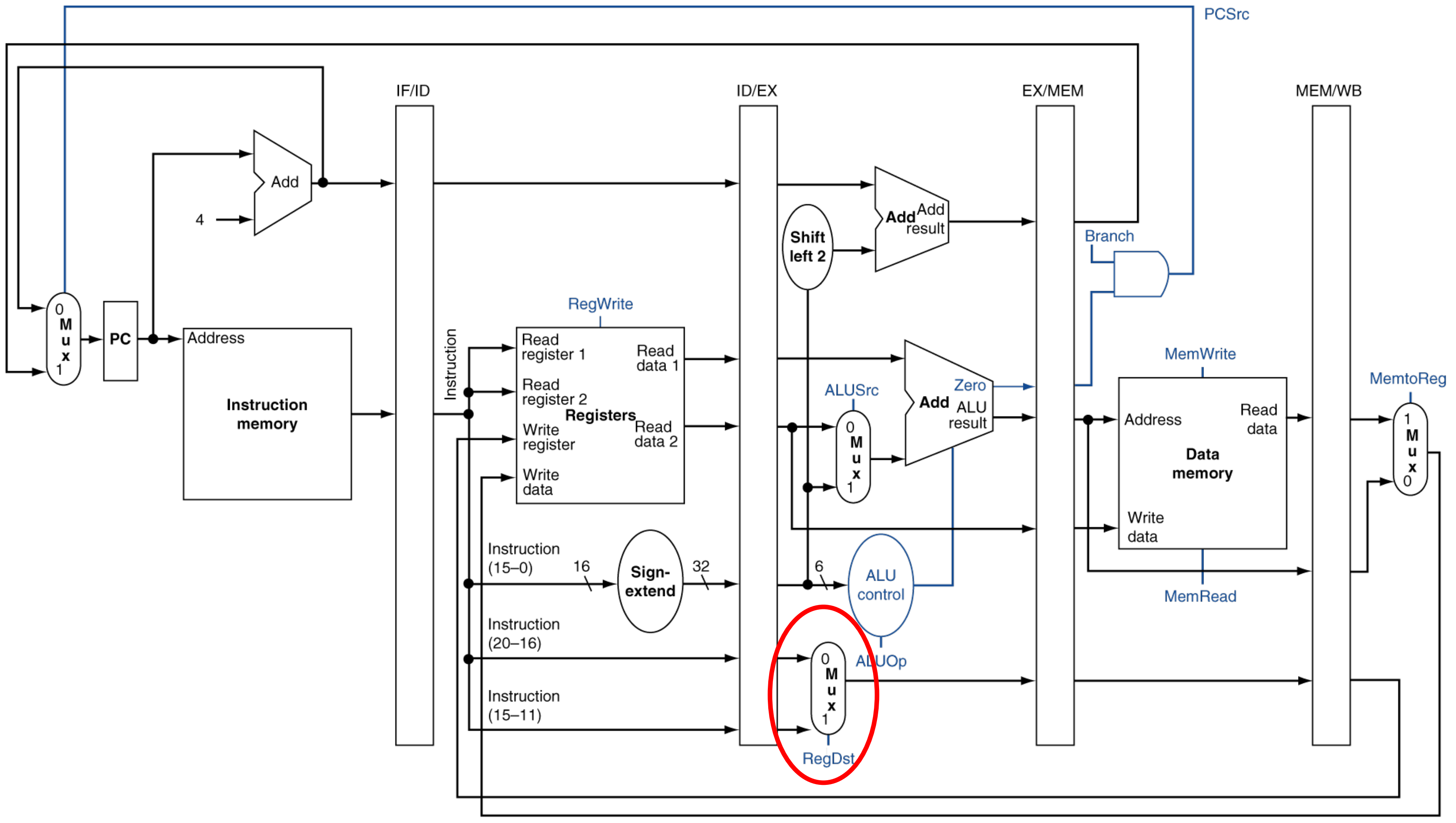
lw $10, 20($1)



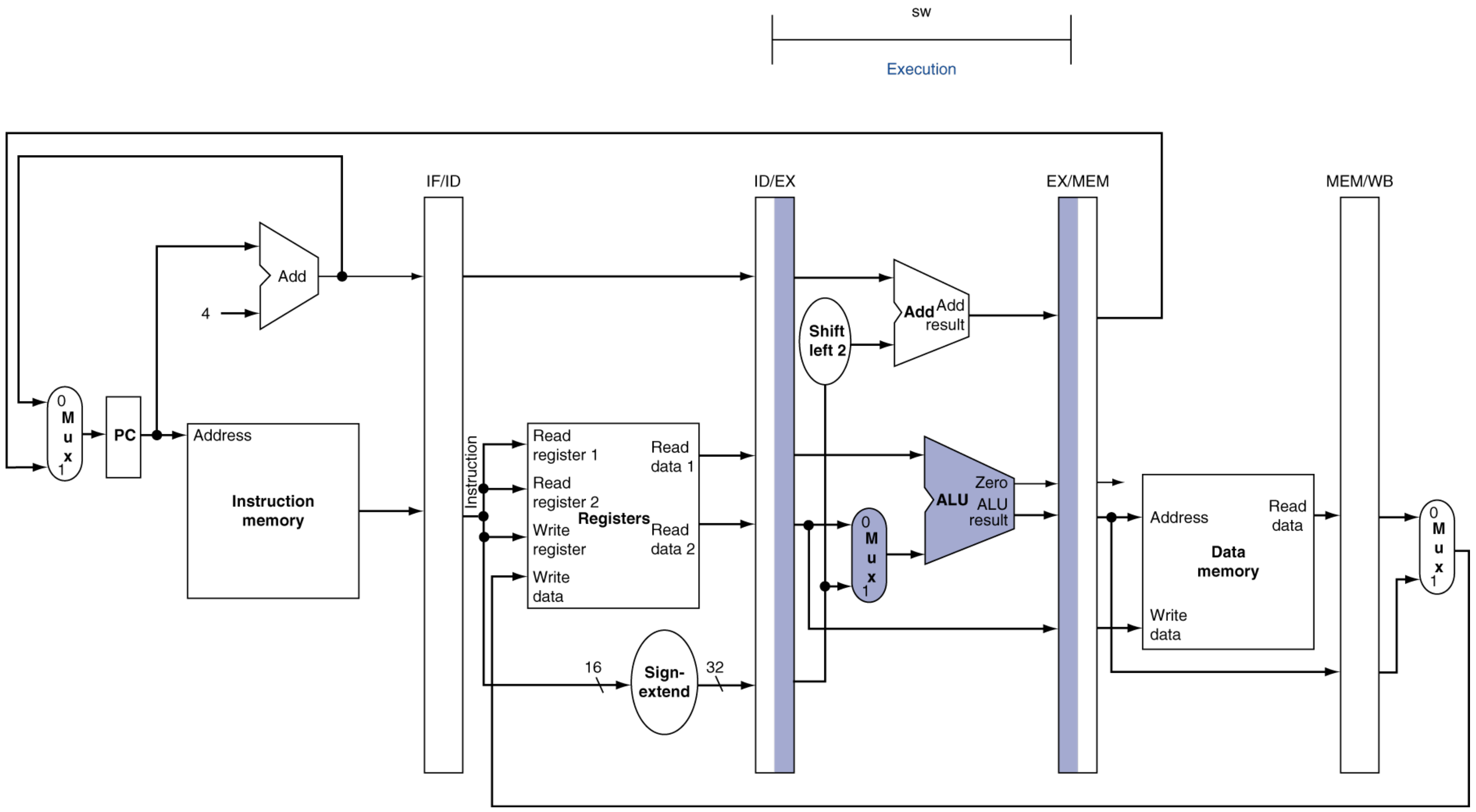
Single Cycle implementation



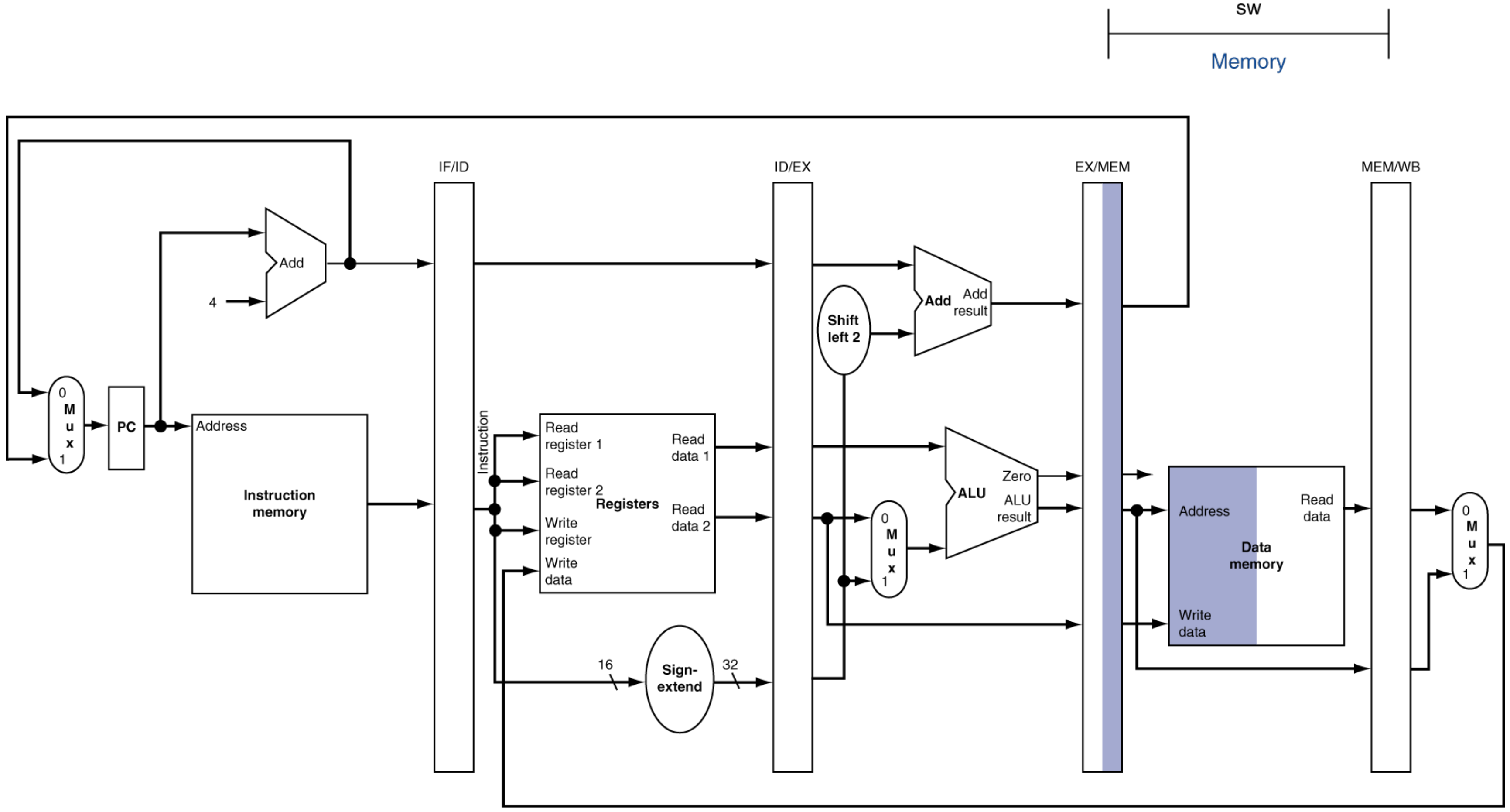
adding RegDst MUX



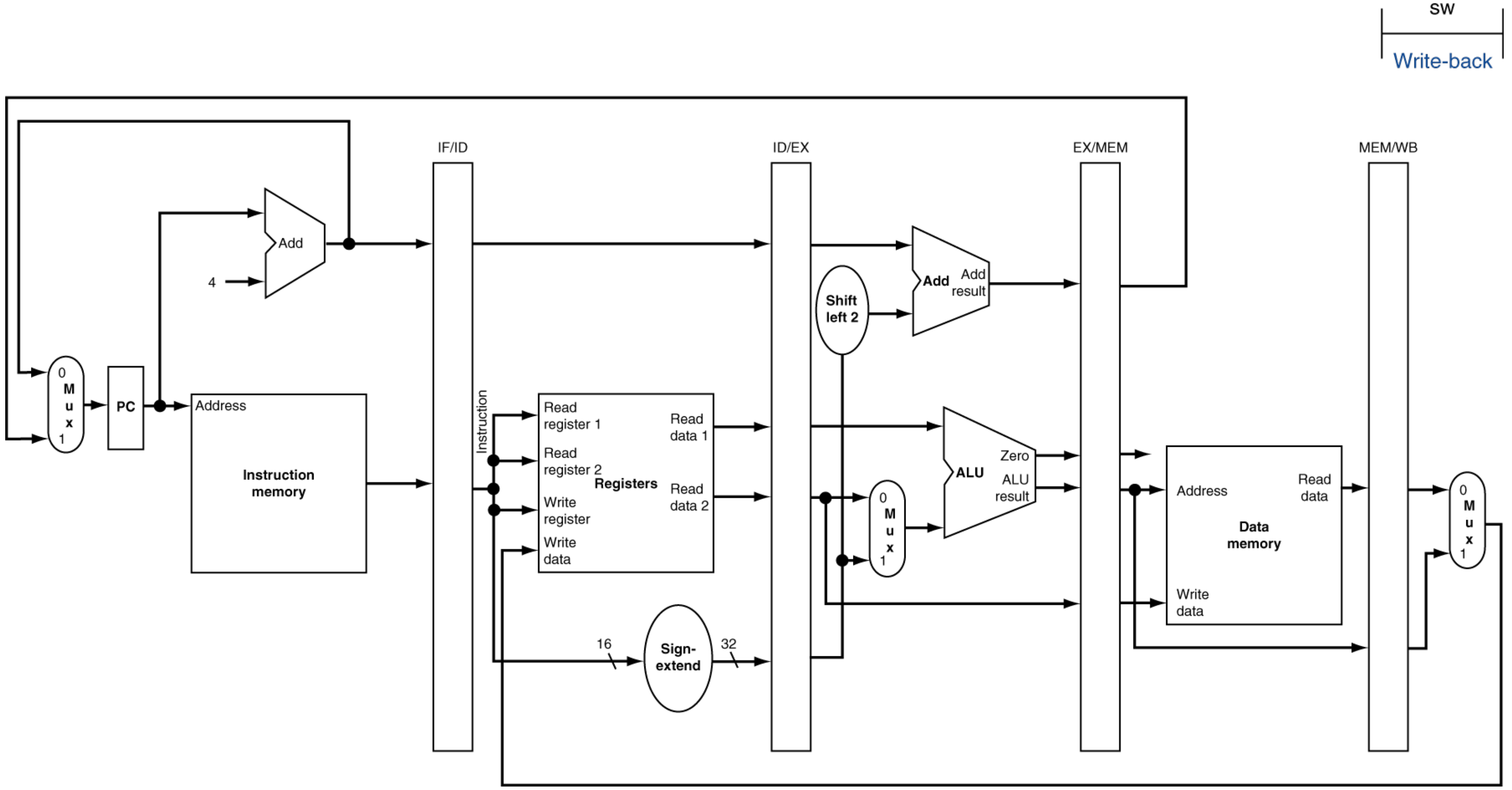
EX for Store


sw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

MEM for Store



WB for Store



SW
Write-back
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Data memory
Write data
Read data