ALU Design Fundamentals

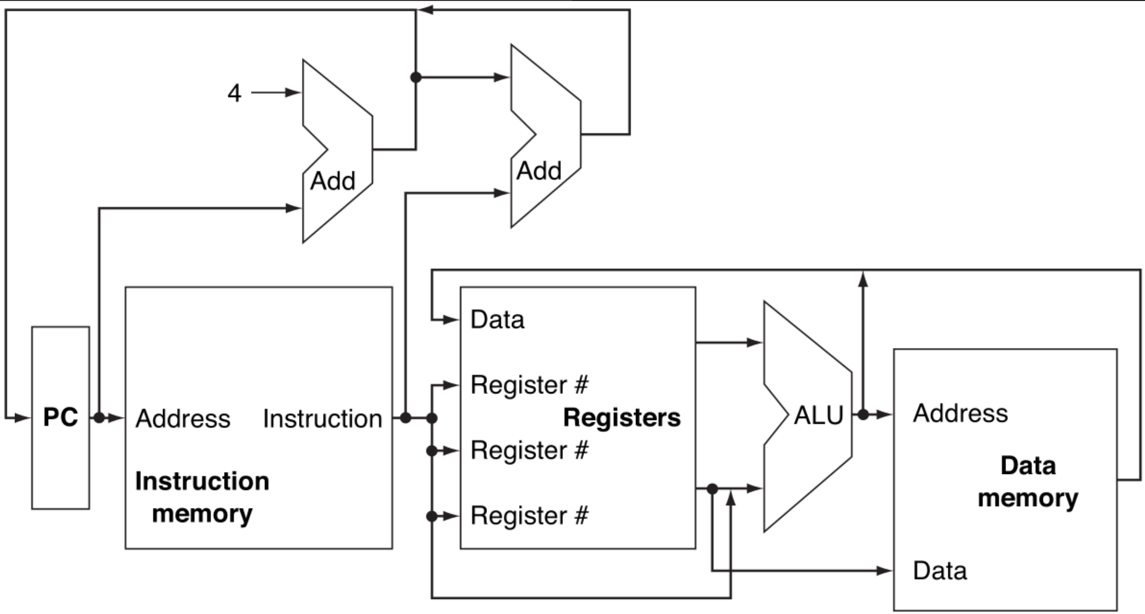
DSC 315: Computer Organization & Operating Systems

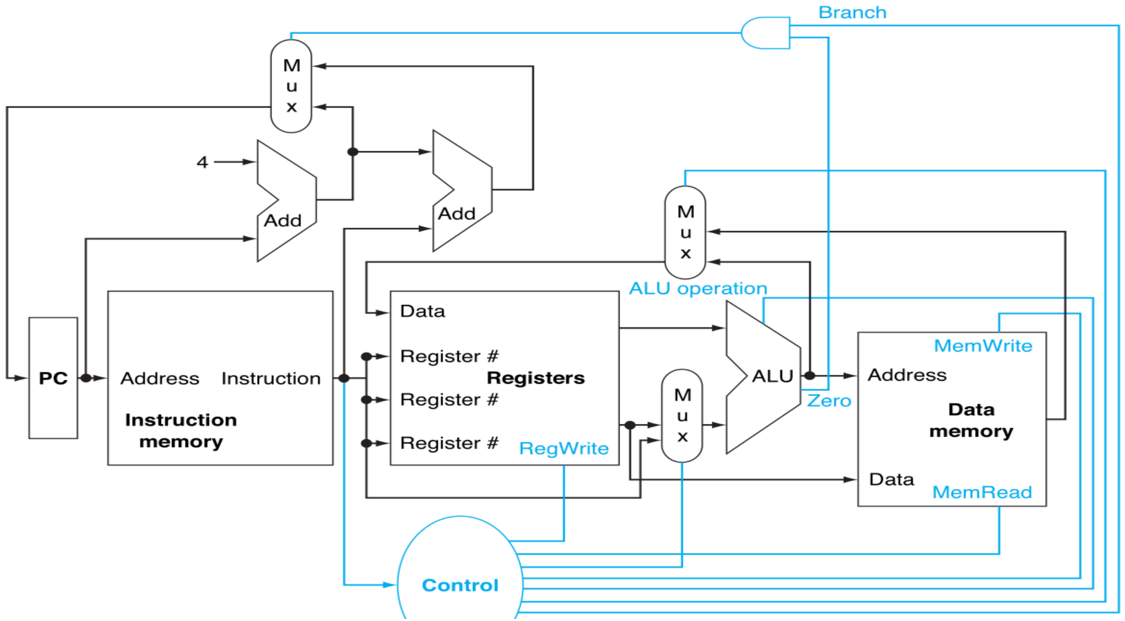
Dr. Laltu Sardar

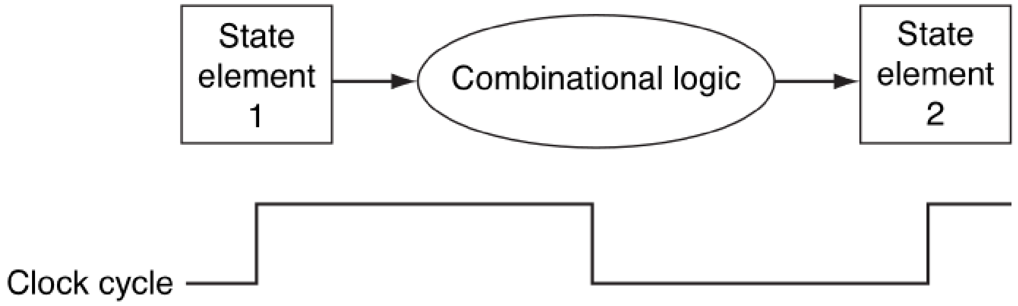
School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)

February 16, 2026

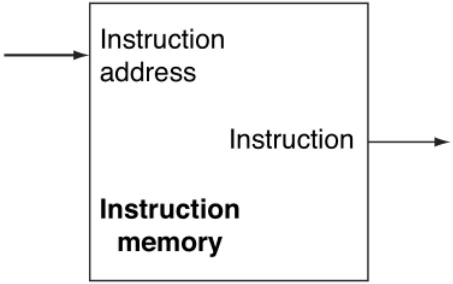
The processor: Building a Datapath


4
Add
Add
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
ALU
Address
Data memory
Data

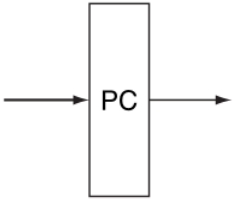

Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data memory
Data
MemRead
Control



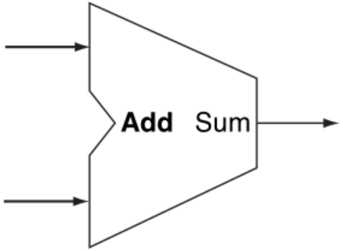
Combinational logic, state elements, and the clock are closely related.



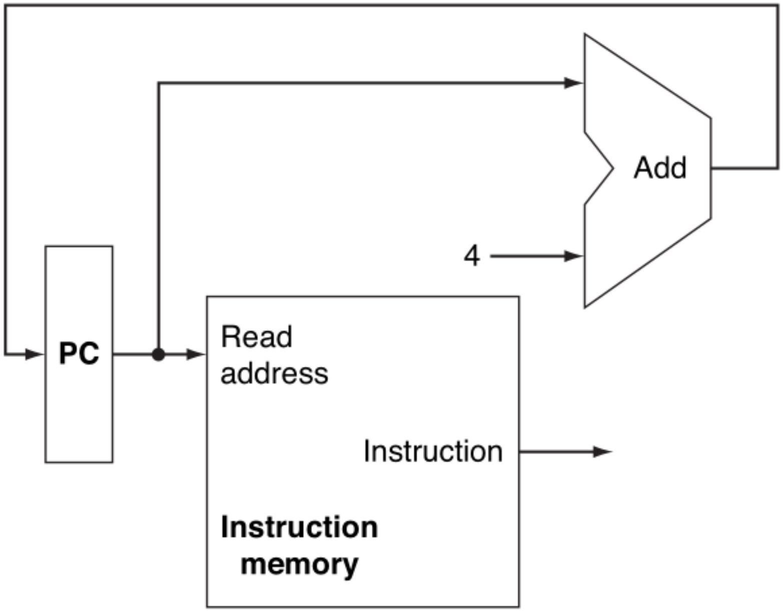
a. Instruction memory

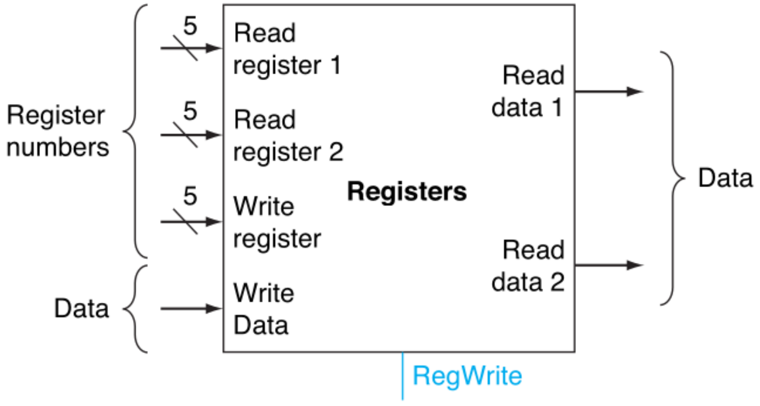


b. Program counter

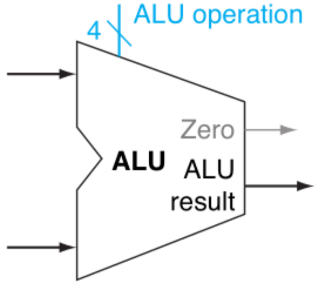


c. Adder

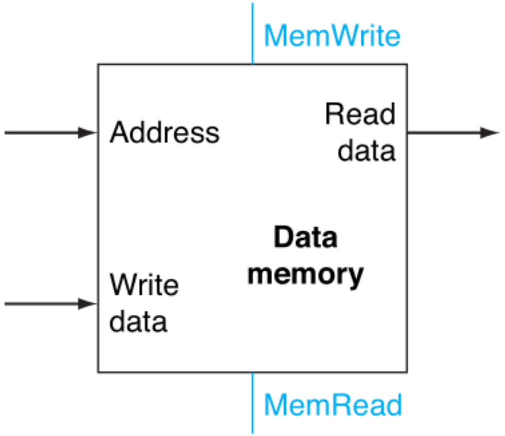

Add
4
PC
Read
address
Instruction
Instruction
memory



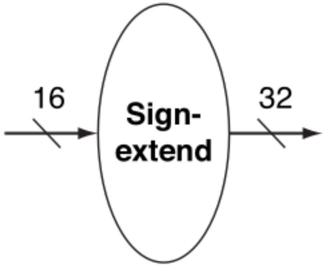
a. Registers



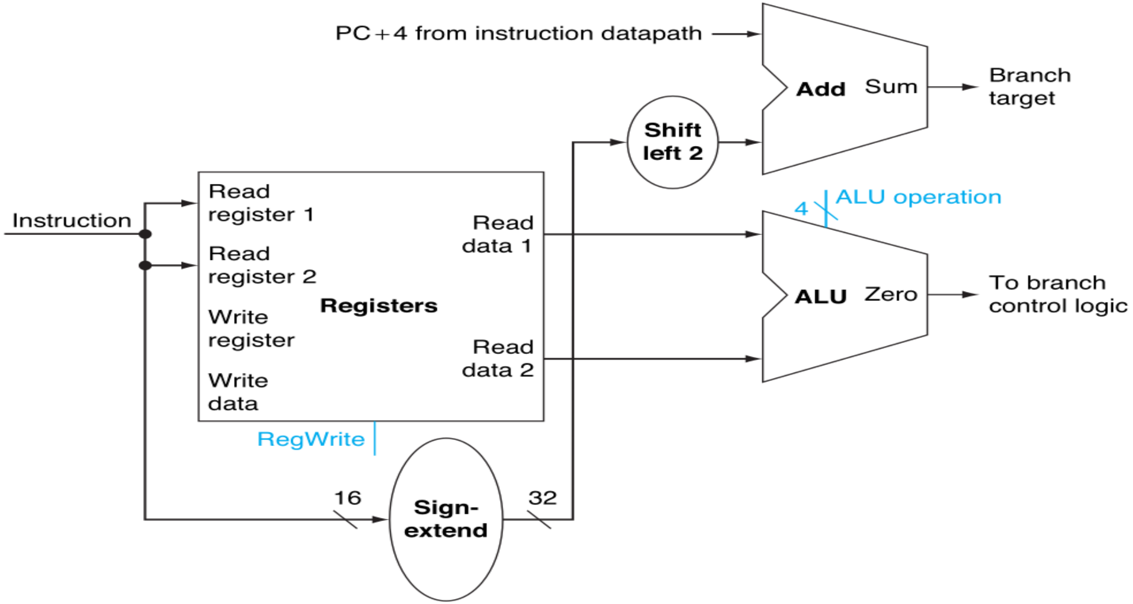
b. ALU

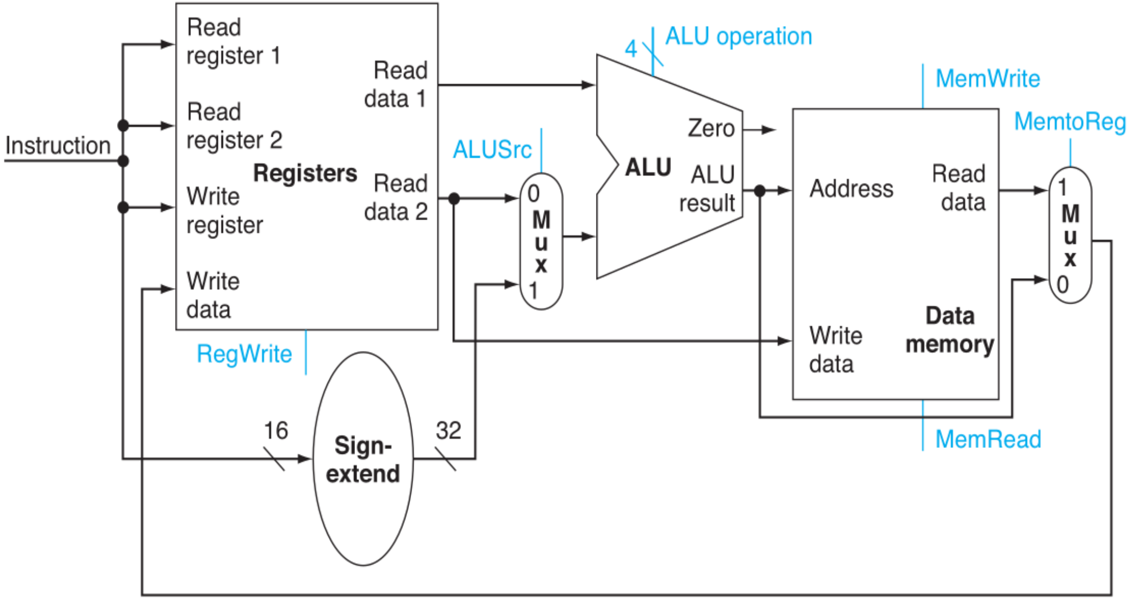


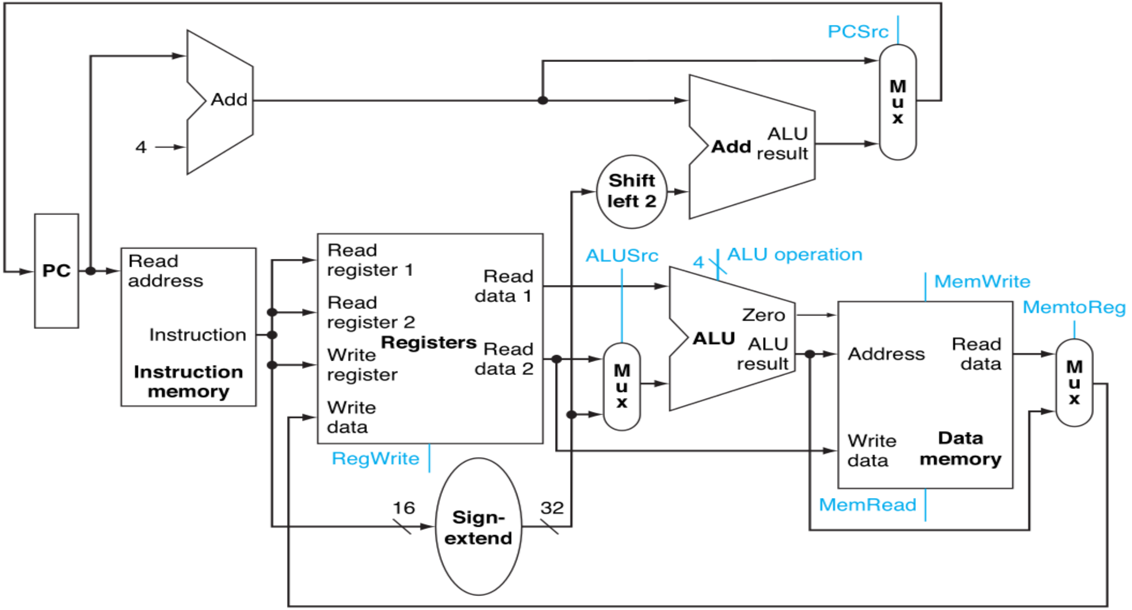
a. Data memory unit

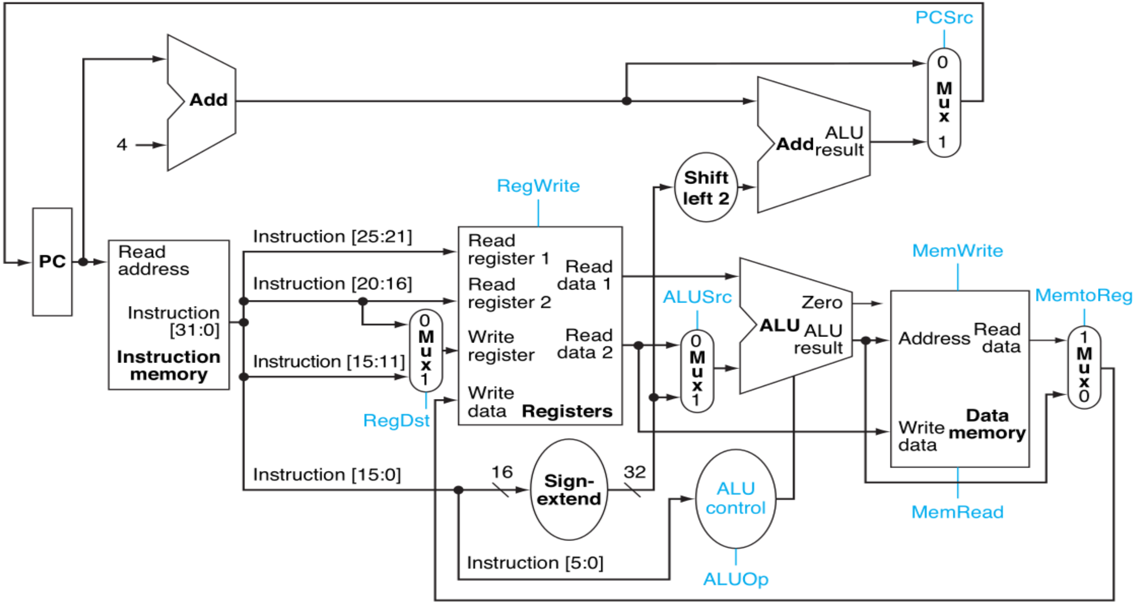


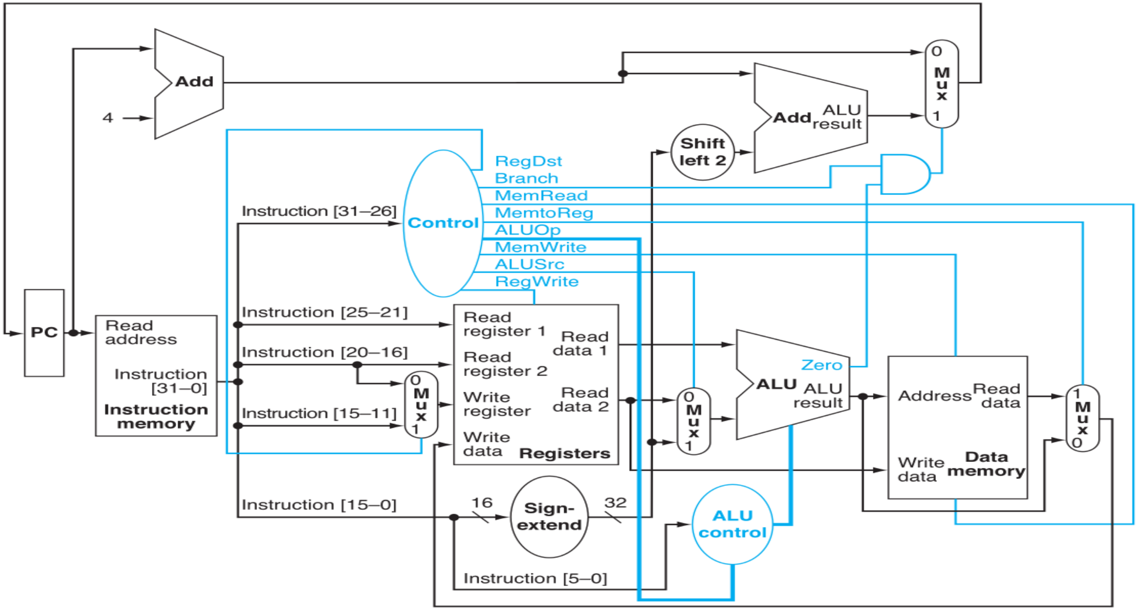
b. Sign extension unit

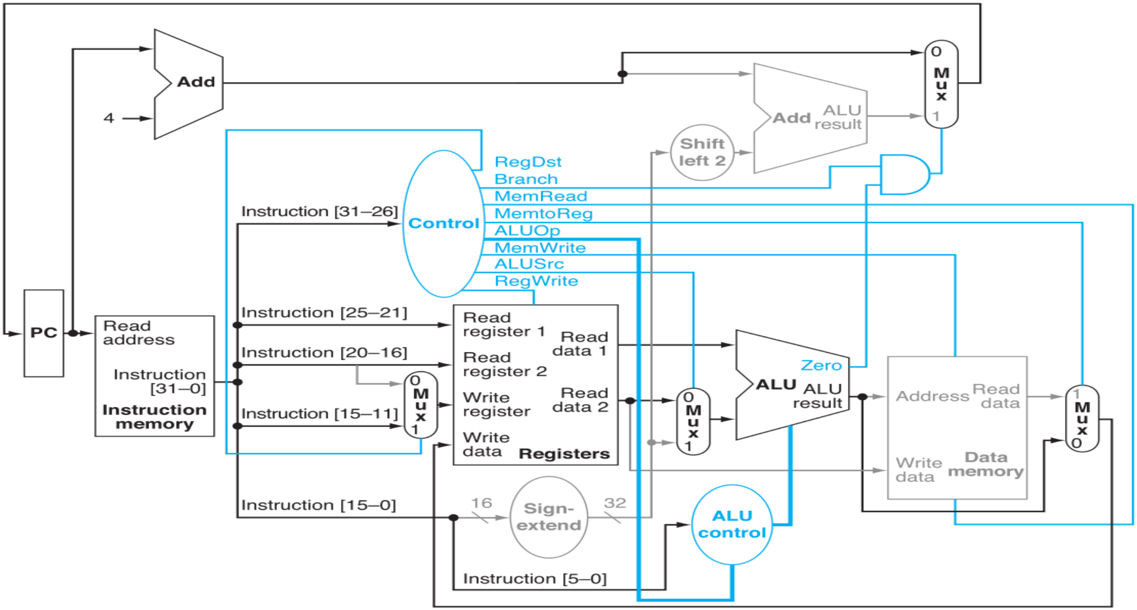

PC + 4 from instruction datapath
Add
Sum
Branch target
Shift left 2
4
ALU operation
Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
ALU
Zero
To branch control logic
RegWrite
16
Sign-extend
32

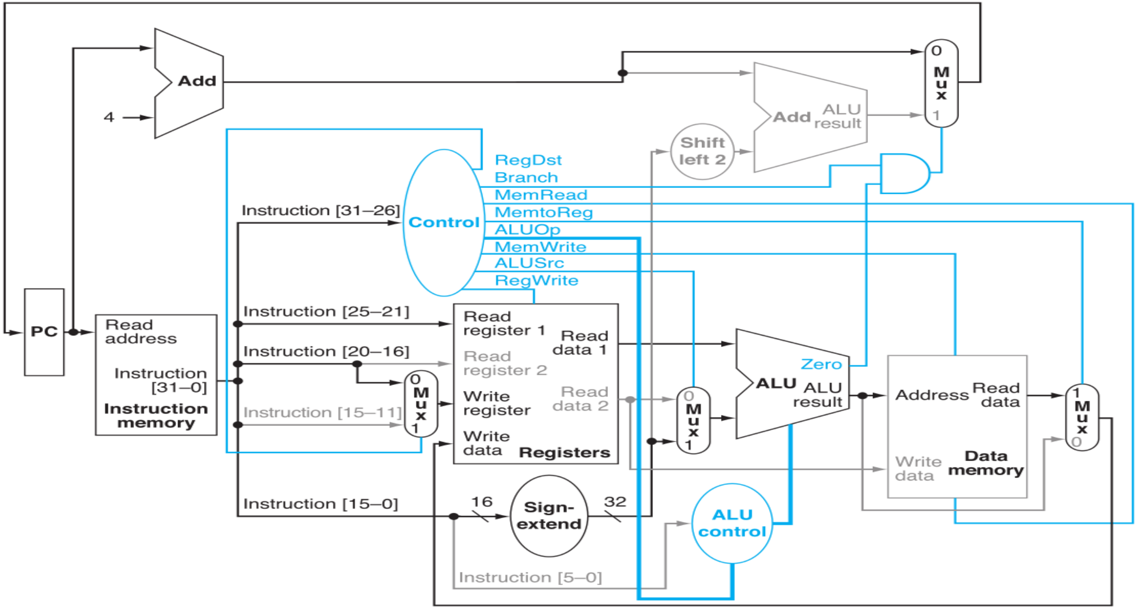

Instruction
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
RegWrite
16
Sign-extend
32
ALUSrc
0
M u x
1
4
ALU operation
ALU
Zero
ALU result
MemWrite
Address
Read data
Write data
Data memory
MemRead
MemtoReg
1
M u x
0

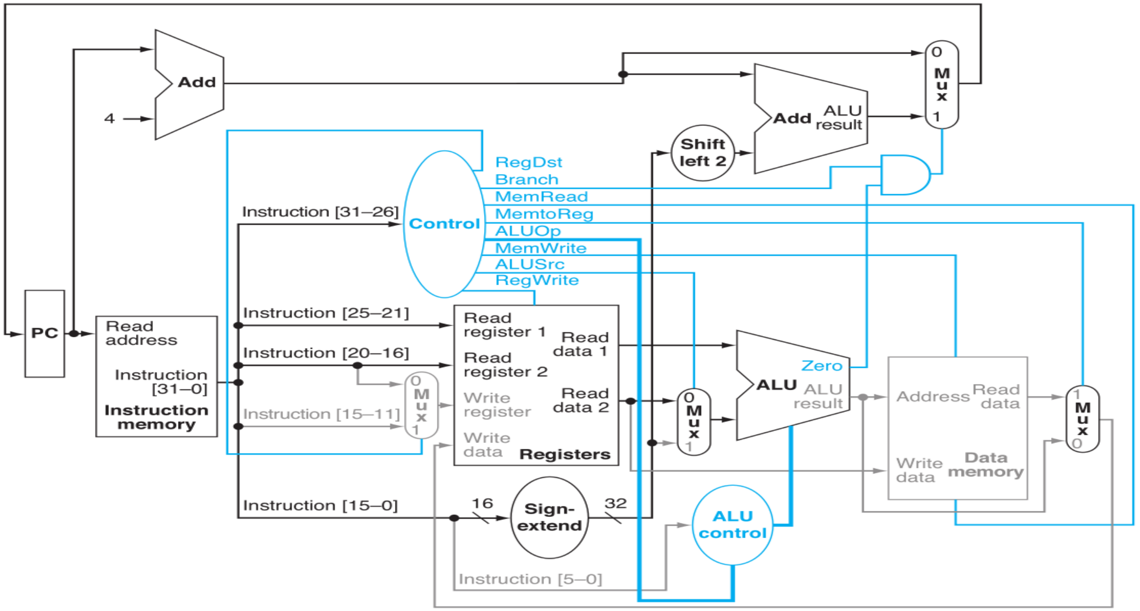

PCSrc
M u x
Add
4
Add
ALU result
Shift left 2
PC
Read address
Instruction
Instruction memory
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
RegWrite
16
Sign-extend
32
ALUSrc
M u x
4
ALU operation
Zero
ALU
ALU result
MemWrite
Address
Read data
Write data
Data memory
MemRead
MemtoReg
M u x

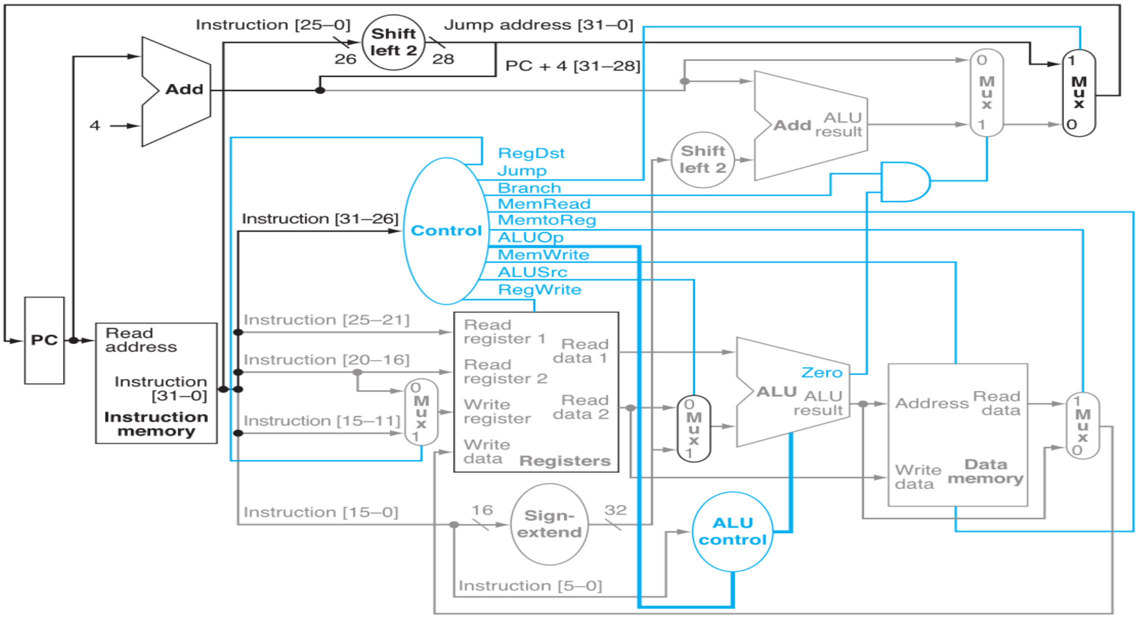

PCSrc
Add
4
0
Mux
1
Add
ALU result
Shift left 2
RegWrite
PC
Read address
Instruction [31:0]
Instruction memory
Instruction [25:21]
Instruction [20:16]
Instruction [15:11]
0
Mux
1
RegDst
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
ALUSrc
0
Mux
1
Zero
ALU
ALU result
MemWrite
Address
Read data
Write data
Data memory
MemRead
MemtoReg
1
Mux
0
Instruction [15:0]
16
Sign-extend
32
ALU control
Instruction [5:0]
ALUOp


Add
4
Add
ALU result
Shift left 2
M u x
0
1
PC
Read address
Instruction [31–0]
Instruction memory
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory

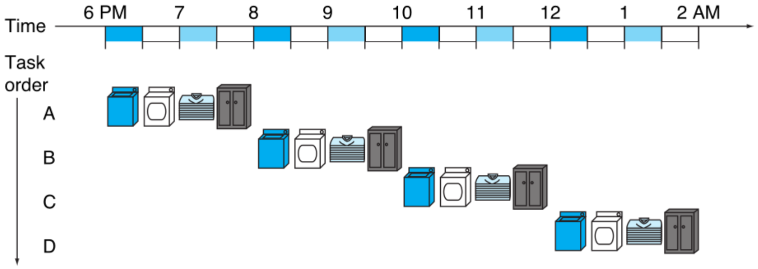

Add
4
Add
ALU result
Shift left 2
0 Mux 1
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
PC
Read address
Instruction [31–0]
Instruction memory
0 Mux 1
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
32
Sign-extend
0 Mux 1
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory
1 Mux 0

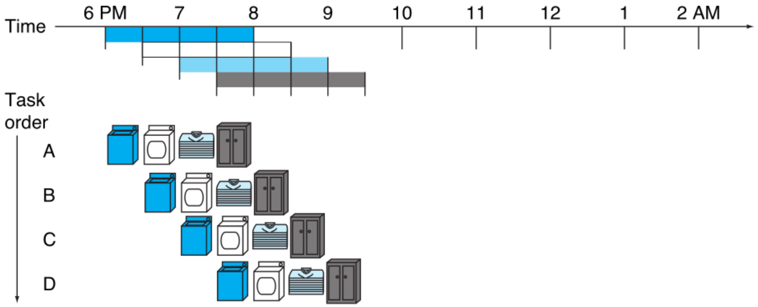

Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
32
Sign-extend
Shift left 2
Add
ALU result
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory
0 Mux 1
1 Mux 0


Add
4
Add
ALU result
Shift left 2
0
Mux
1
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
PC
Read address
Instruction [31–0]
Instruction memory
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
32
Sign-extend
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory


Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC + 4 [31–28]
Add
4
Add
ALU result
Shift left 2
0 M u x 1
1 M u x 0
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
0 M u x 1
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
0 M u x 1
ALU
Zero
ALU result
Address
Read data
Write data
Data memory
1 M u x 0
Instruction [15–0]
16
Sign-extend
32
ALU control
Instruction [5–0]

The Pipeline


Time
6 PM
7
8
9
10
11
12
1
2 AM
Task order
A
B
C
D


Time
6 PM
7
8
9
10
11
12
1
2 AM
Task order
A
B
C
D

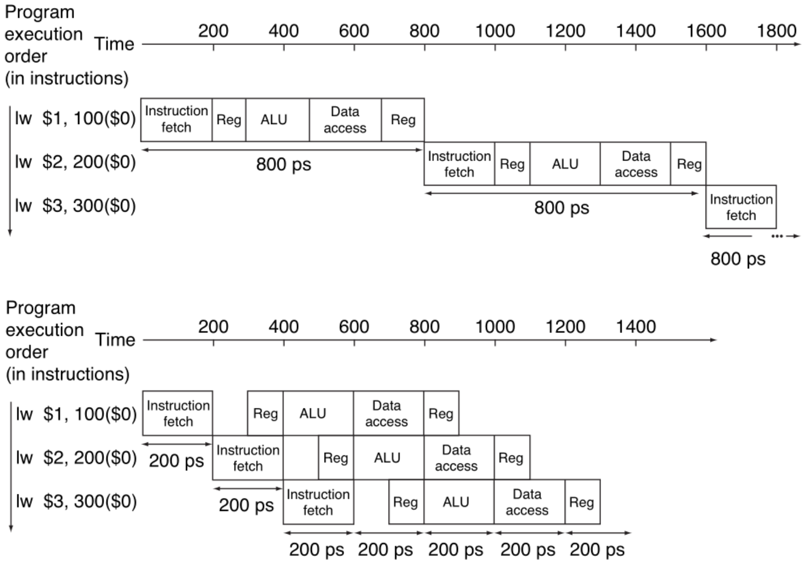


FIGURE 4.27 Single-cycle, nonpipelined execution in top versus pipelined execution in

Introduction to Pipeline Hazards

- Pipelining allows multiple instructions to overlap in execution.
- Hazards prevent the next instruction from executing in its designated clock cycle.
- Three main types:
    - Structural Hazards
    - Data Hazards
    - Control Hazards

Structural Hazard

Definition:

- Occurs when hardware resources are insufficient.
- Two or more instructions compete for the same resource.

Example:

- Single memory used for both instruction fetch and data access.

Handling Structural Hazards

- Stall the pipeline (insert bubble).
- Duplicate hardware resources.
- Improve pipeline design and scheduling.

Data Hazard

Definition:

- Occurs when an instruction depends on the result of a previous instruction.

Types:

- RAW (Read After Write)
- WAR (Write After Read)
- WAW (Write After Write)

Example of Data Hazard

Instruction Sequence

```
ADD R1, R2, R3
SUB R4, R1, R5
```

- SUB depends on result of ADD.
- This creates a RAW hazard.

Handling Data Hazards

- Pipeline stalling.
- Forwarding (bypassing).
- Register renaming.
- Compiler instruction scheduling.

Control Hazard

Definition:

- Occurs due to branch and jump instructions.
- Next instruction depends on branch outcome.

Example:

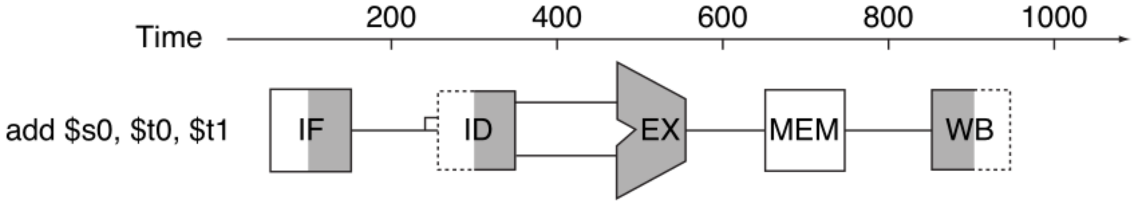
```
BEQ R1, R2, LABEL
```

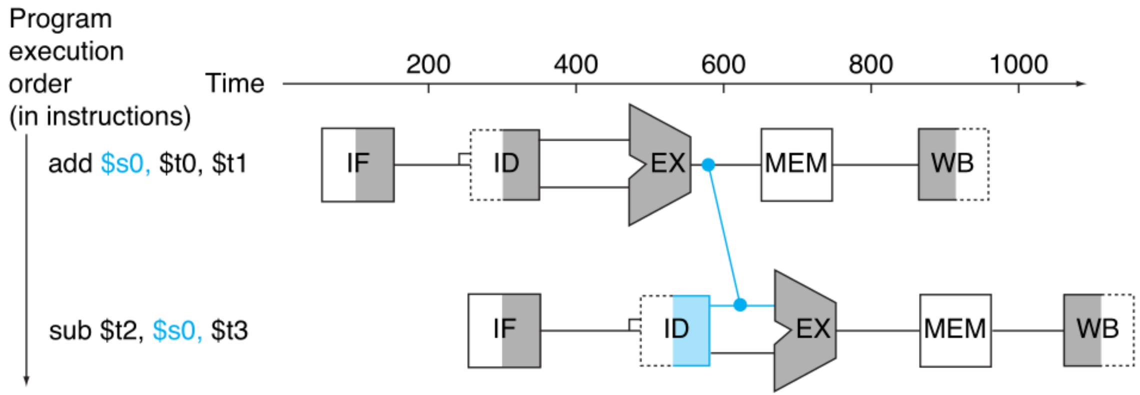
Handling Control Hazards

- Pipeline stalling.
- Static branch prediction.
- Dynamic branch prediction.
- Branch Target Buffer (BTB).
- Delayed branching.
- Speculative execution.

Summary

Hazard	Cause	Solutions
Structural	Resource conflict	Stall, duplicate hardware
Data	Data dependency	Forwarding, renaming
Control	Branch instructions	Prediction, speculation


Time
200
400
600
800
1000
add $s0, $t0, $t1
IF
ID
EX
MEM
WB


Program execution order (in instructions)
Time
200
400
600
800
1000
add $s0, $t0, $t1
IF
ID
EX
MEM
WB
sub $t2, $s0, $t3
IF
ID
EX
MEM
WB

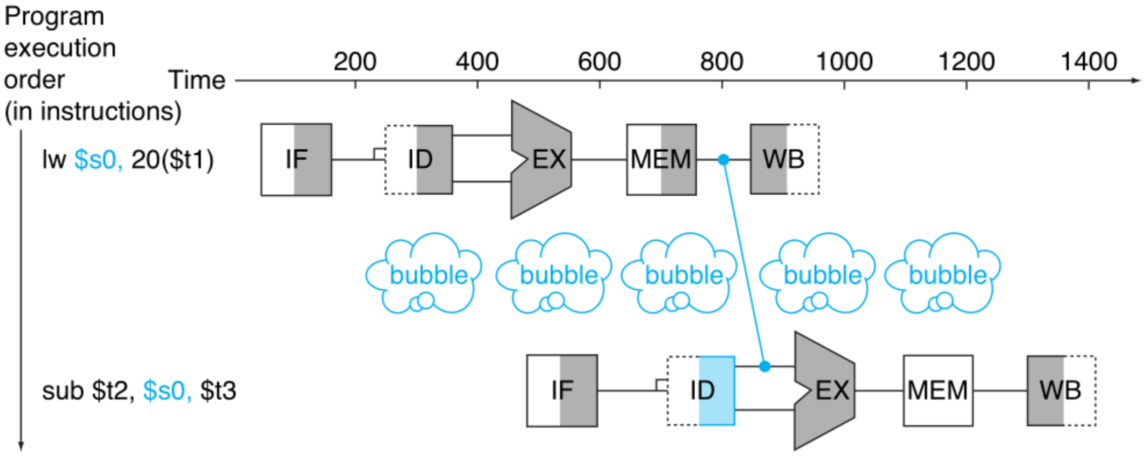


FIGURE 4.30 We need a stall even with forwarding when an R-format instruction following a load tries to use the data. Without the stall, the path from memory access stage output to execution

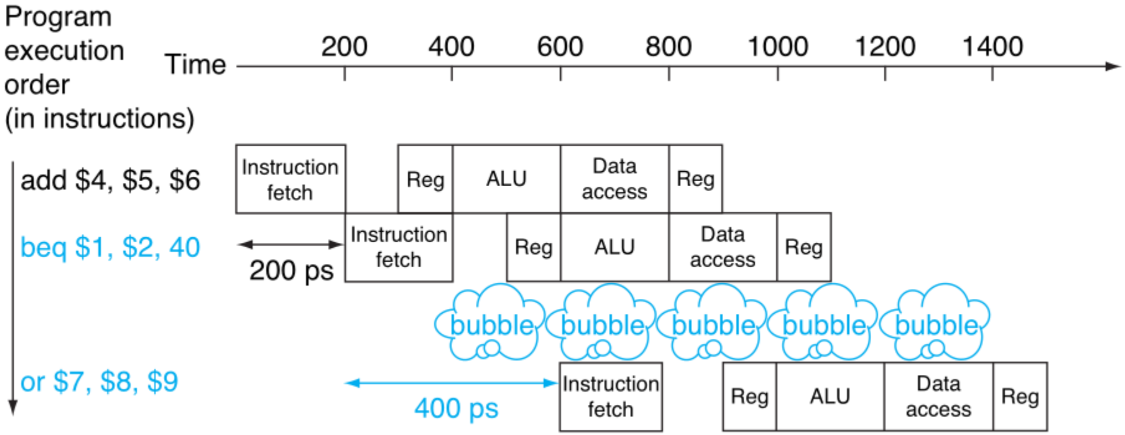
Consider the following code segment in C:

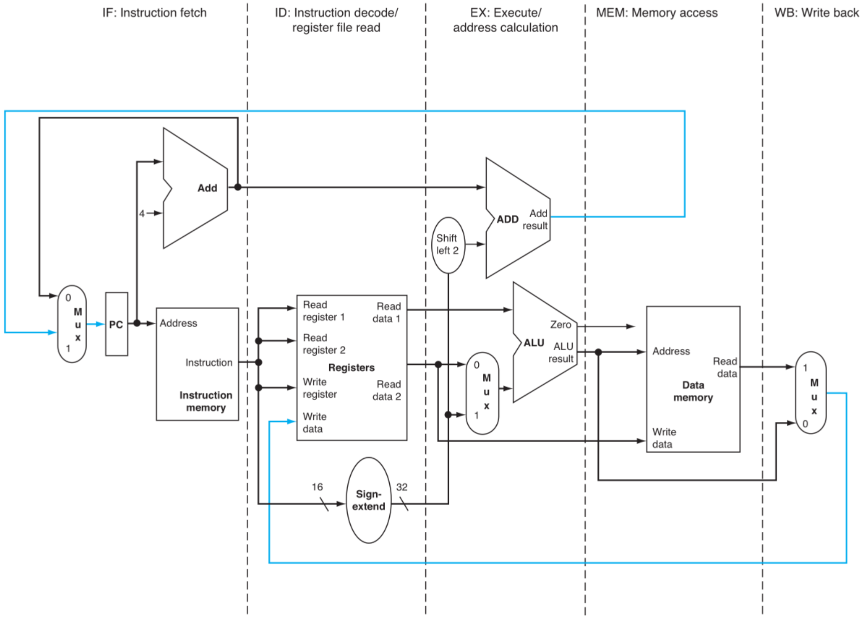
```
a = b + e;
c = b + f;
```

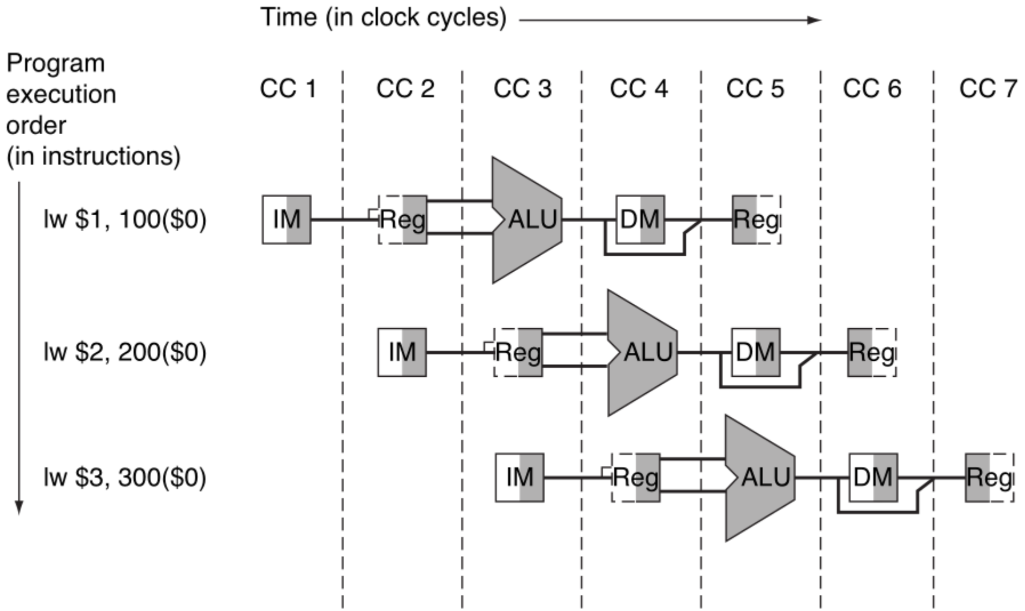
Here is the generated MIPS code for this segment, assuming all variables are in memory and are addressable as offsets from `$t0`:

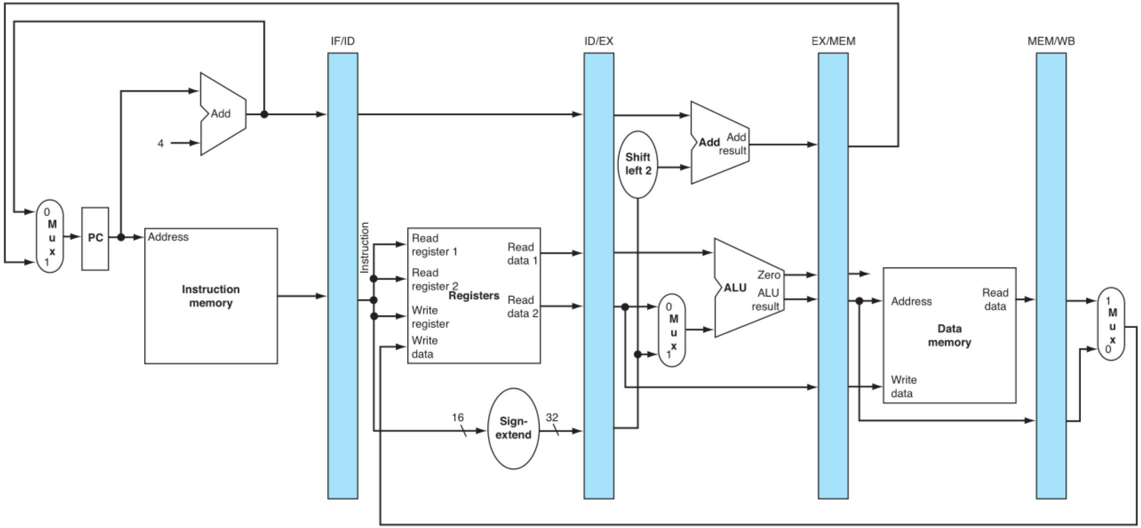
```
lw      $t1, 0($t0)
lw      $t2, 4($t0)
add     $t3, $t1,$t2
sw      $t3, 12($t0)
lw      $t4, 8($t0)
add     $t5, $t1,$t4
sw      $t5, 16($t0)
```

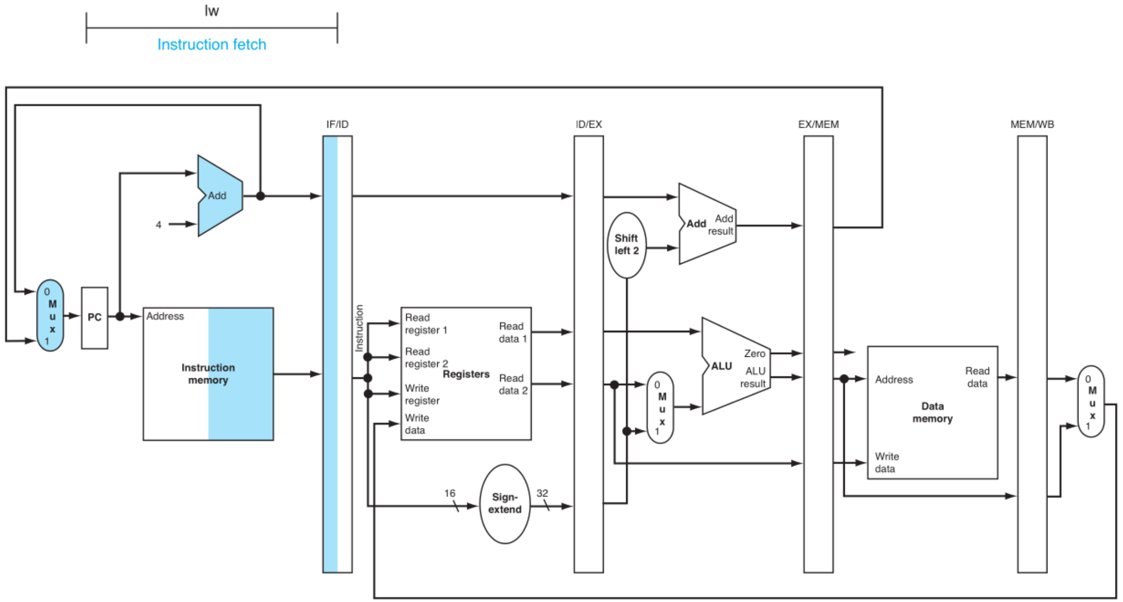
```
lw    $t1, 0($t0)
lw    $t2, 4($t0)
lw    $t4, 8($t0)
add   $t3, $t1,$t2
sw    $t3, 12($t0)
add   $t5, $t1,$t4
sw    $t5, 16($t0)
```

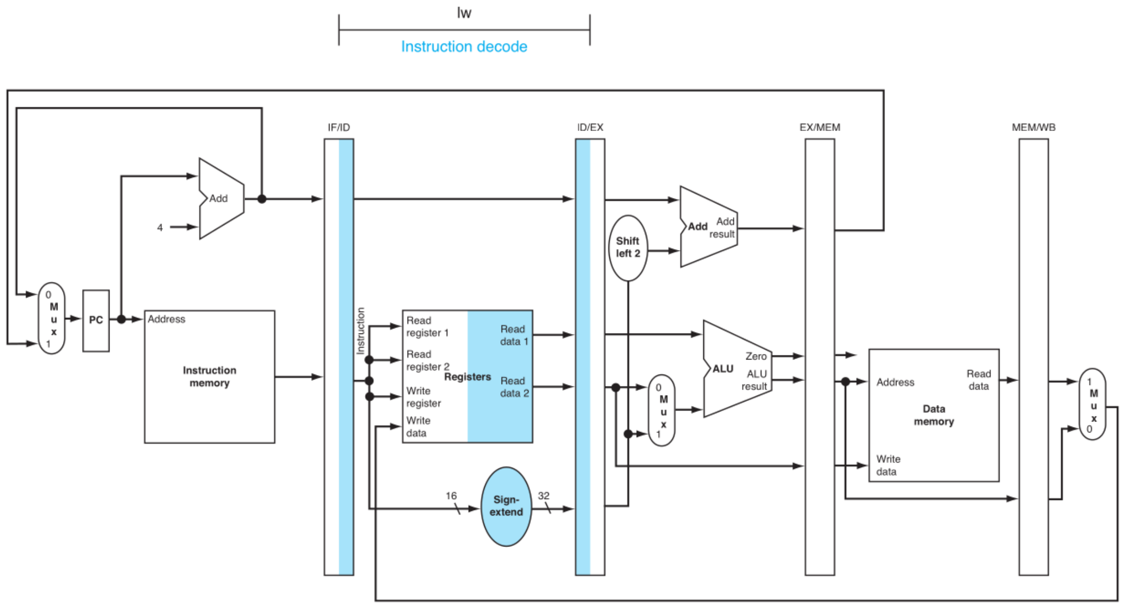

Program execution order (in instructions)
Time
200
400
600
800
1000
1200
1400
add $4, $5, $6
Instruction fetch
Reg
ALU
Data access
Reg
beq $1, $2, 40
200 ps
Instruction fetch
Reg
ALU
Data access
Reg
bubble
bubble
bubble
bubble
bubble
or $7, $8, $9
400 ps
Instruction fetch
Reg
ALU
Data access
Reg

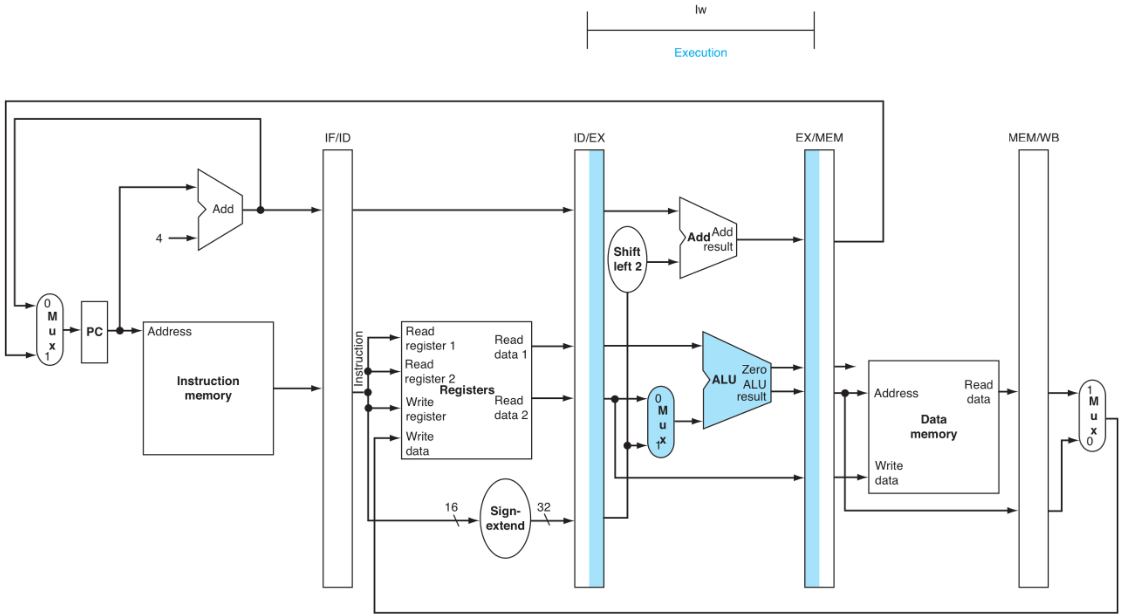

IF: Instruction fetch
ID: Instruction decode/ register file read
EX: Execute/ address calculation
MEM: Memory access
WB: Write back
Add
4
M u x
0
1
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Write register
Write data
Registers
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
ADD
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

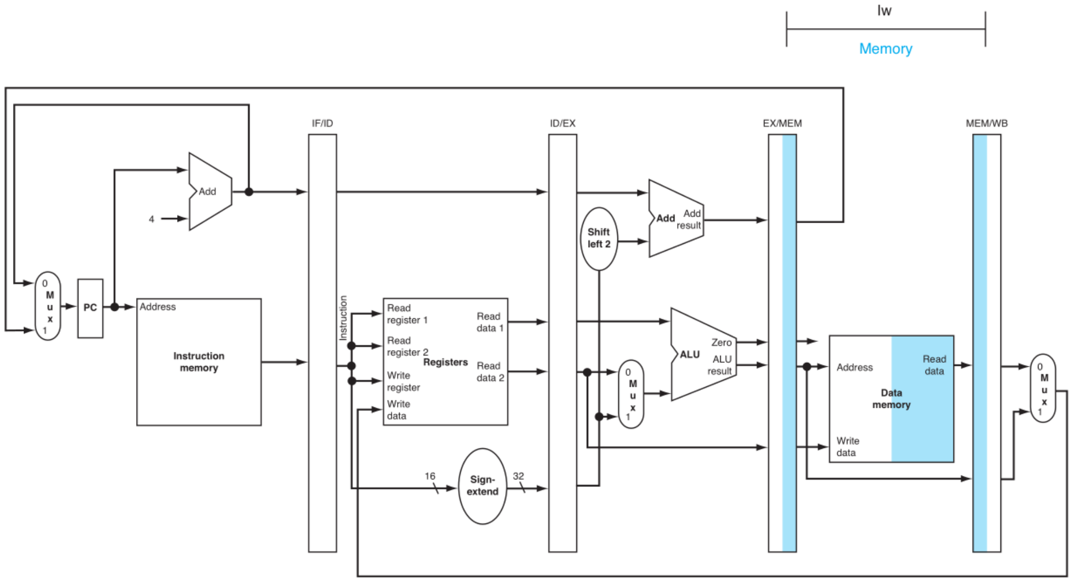

Time (in clock cycles)
Program execution order (in instructions)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
lw $1, 100($0)
lw $2, 200($0)
lw $3, 300($0)
IM
Reg
ALU
DM
Reg

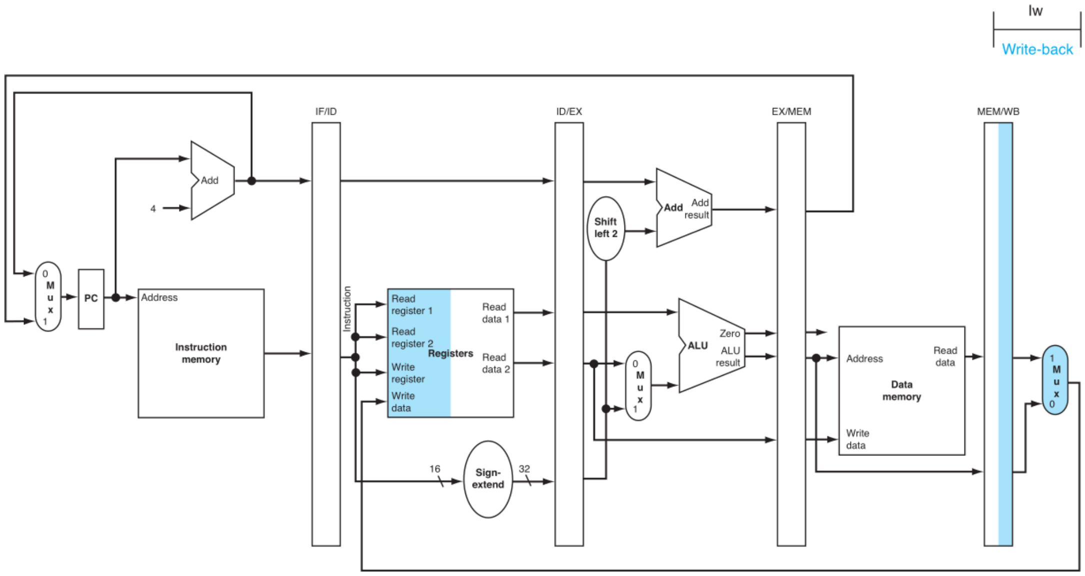

IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
0
M
u
x
1
ALU
Zero
ALU result
Address
Data memory
Write data
Read data
1
M
u
x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

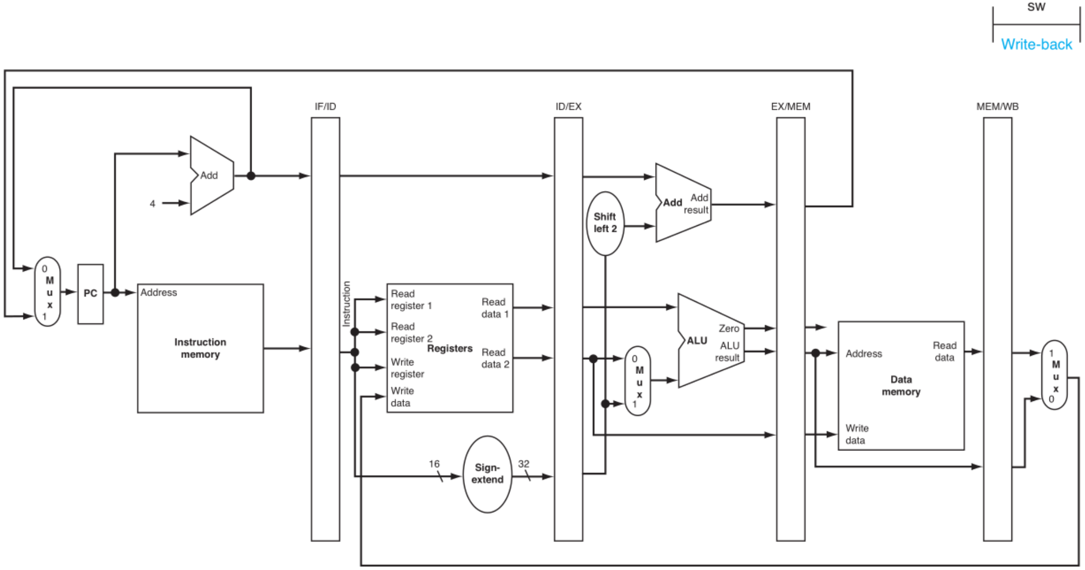

lw
Instruction fetch
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data

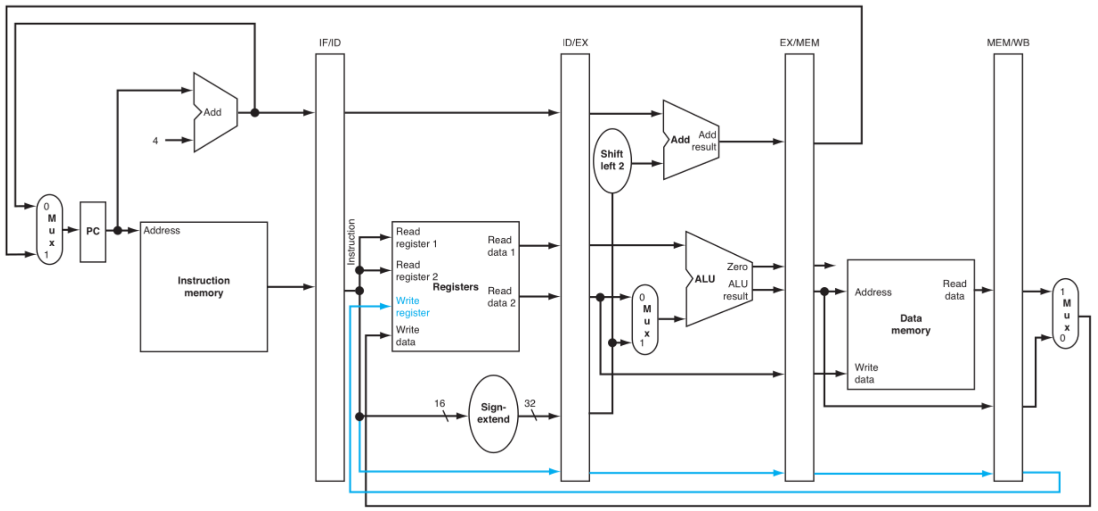

lw
Instruction decode
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign-extend
32
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Data memory
Read data
Write data


lw
Execution
IF/ID
ID/EX
EX/MEM
MEM/WB
Add
4
PC
Address
Instruction memory
Instruction
M u x
0
1
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
Data memory
Read data
Write data


lw
Memory
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


lw
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
0
M
u
x
1
Address
Read data
Data memory
Write data
1
M
u
x
0


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


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
0
M
u
x
1
Address
Read data
Data memory
Write data
1
M
u
x
0

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram
`laltu.sardar@iisertvm.ac.in`, `laltu.sardar.crypto@gmail.com`
Course webpage: `https://laltu-sardar.github.io/courses/corgos_2026.html`.

Carl Hamacher, Zvonko Vranesic, and Safwat Zaky.
Computer Organization.
McGraw-Hill, 6th edition, 2012.

John L. Hennessy and David A. Patterson.
Computer Architecture: A Quantitative Approach.
Morgan Kaufmann, 6th edition, 2017.

Vincent P. Heuring and Harry F. Jordan.
Computer System Design and Architecture.
Pearson, 2nd edition, 2007.

David A. Patterson and John L. Hennessy.
Computer Organization and Design: The Hardware/Software Interface.
Morgan Kaufmann, 4th edition, 2014.

William Stallings.
Computer Organization and Architecture: Designing for Performance.
Pearson, 10th edition, 2016.