Homework for Wednesday April 5, 2000

1. Convert the RISCEE 1 Architecture into a pipeline Architecture (like Figure 6.30) (showing the number data and control bits).

2. Build the control line table (like Figure 6.28) for the RISCEE3 pipeline architecture for RISCEE1 instructions: (addi, subi, load, store, beq, jmp, jal).

3. Homework 6.23, p534
4. Homework 6.24, p534

Lecture

Computer Architecture and Engineering

Pipeline Control,

Data Hazards

and Branch Hazards

Models

Single-cycle model (non-overlapping)

- Each instruction executes in a single cycle
- Every instruction and clock-cycle must be stretched to the slowest instruction (p.438)

Multi-cycle model (non-overlapping)

- Each instruction executes in variable number of cycles
- The clock-cycle must be stretched to the slowest step
- Ability to share functional units within the execution of a single instruction

Pipeline model (overlapping)

- Each instruction executes in several cycles
- The clock-cycle must be stretched to the slowest step
- Gains efficiency by overlapping the execution of multiple instructions, increasing hardware utilization. (p. 377)

Overhead

Single-cycle model

- 8 ns Clock (125 MHz), (non-overlapping)
- 1 ALU + 2 adders
- 0 Muxes
- 0 Datapath Register bits (Flip-Flops)

Multi-cycle model

- 2 ns Clock (500 MHz), (non-overlapping)
- 1 ALU + Controller
- 5 Muxes
- 160 Datapath Register bits (Flip-Flops)

Pipeline model

- 2 ns Clock (500 MHz), (overlapping)
- 2 ALU + Controller
- 4 Muxes
- 373 Datapath Register bits (Flip-Flops)

Pipeline Designing

- What makes it easy
  - all instructions are the same length
  - just a few instruction formats
  - memory operands appear only in loads and stores

- What makes it hard?
  - structural hazards: suppose we had only one memory
  - control hazards: need to worry about branch instructions
  - data hazards: an instruction depends on a previous instruction

Pipeline Hazards

Pipeline hazards

- Solution #1 always works: stall, delay & procrastinate!

Structural Hazards (i.e. fetching same memory bank)

- Solution #2: partition architecture

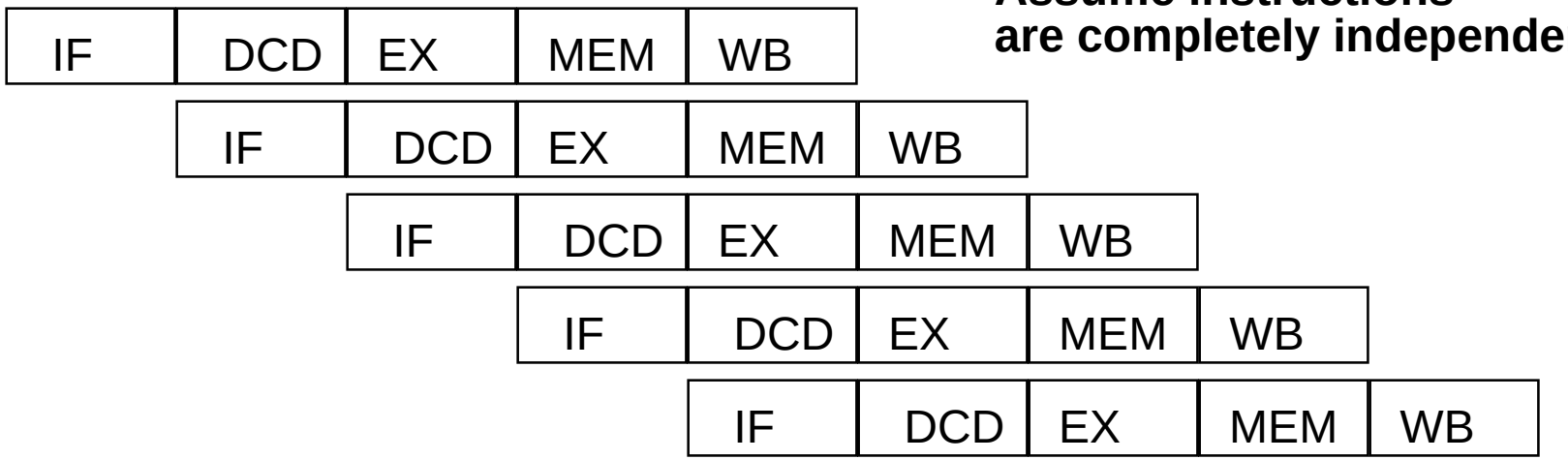
Control Hazards (i.e. branching)

- Solution #1: stall! but decreases throughput
- Solution #2: guess and back-track
- Solution #3: delayed decision: delay branch & fill slot

Data Hazards (i.e. register dependencies)

- Worst case situation
- Solution #2: re-order instructions
- Solution #3: forwarding or bypassing: delayed load

Ideal Pipelining

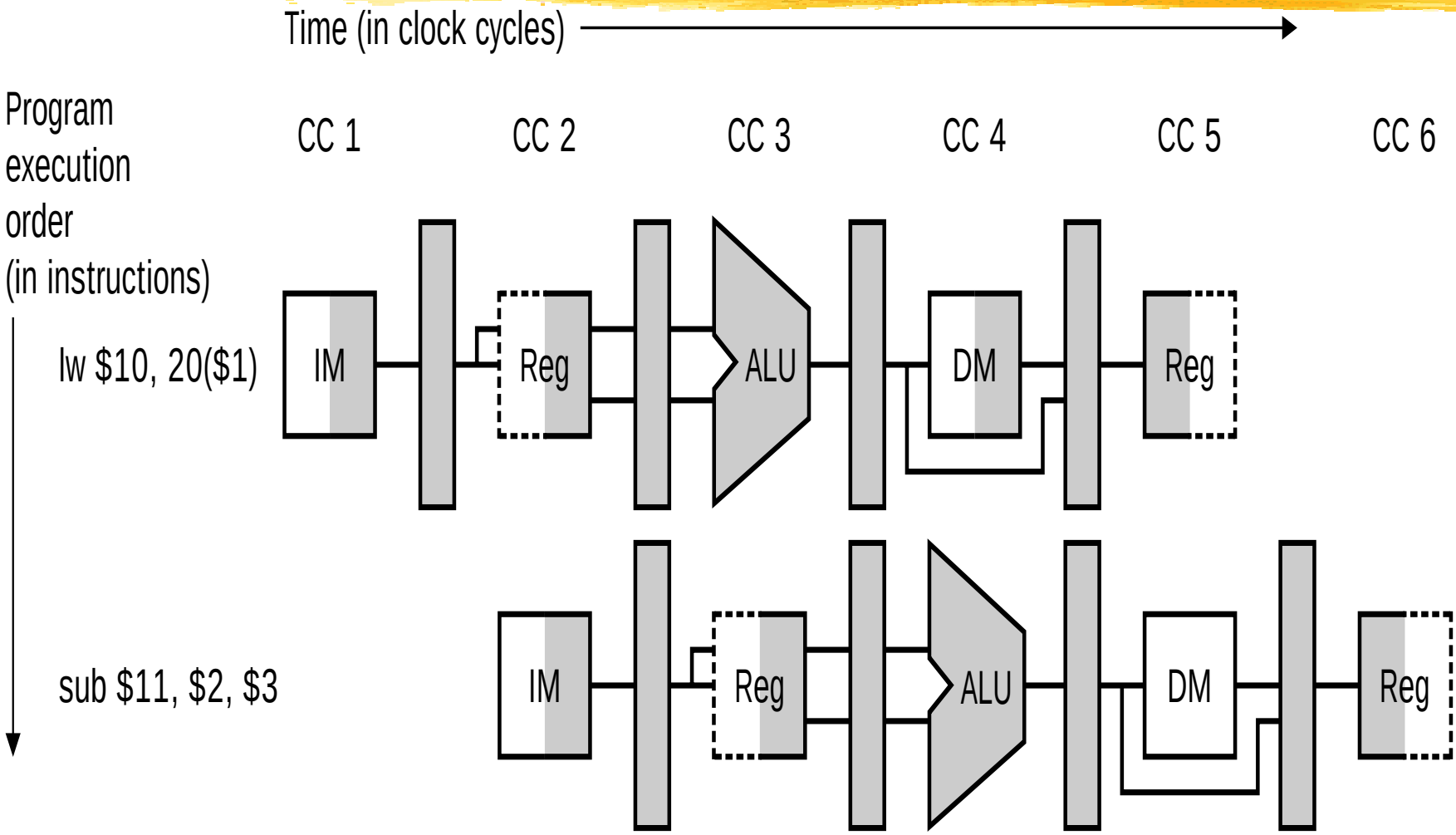


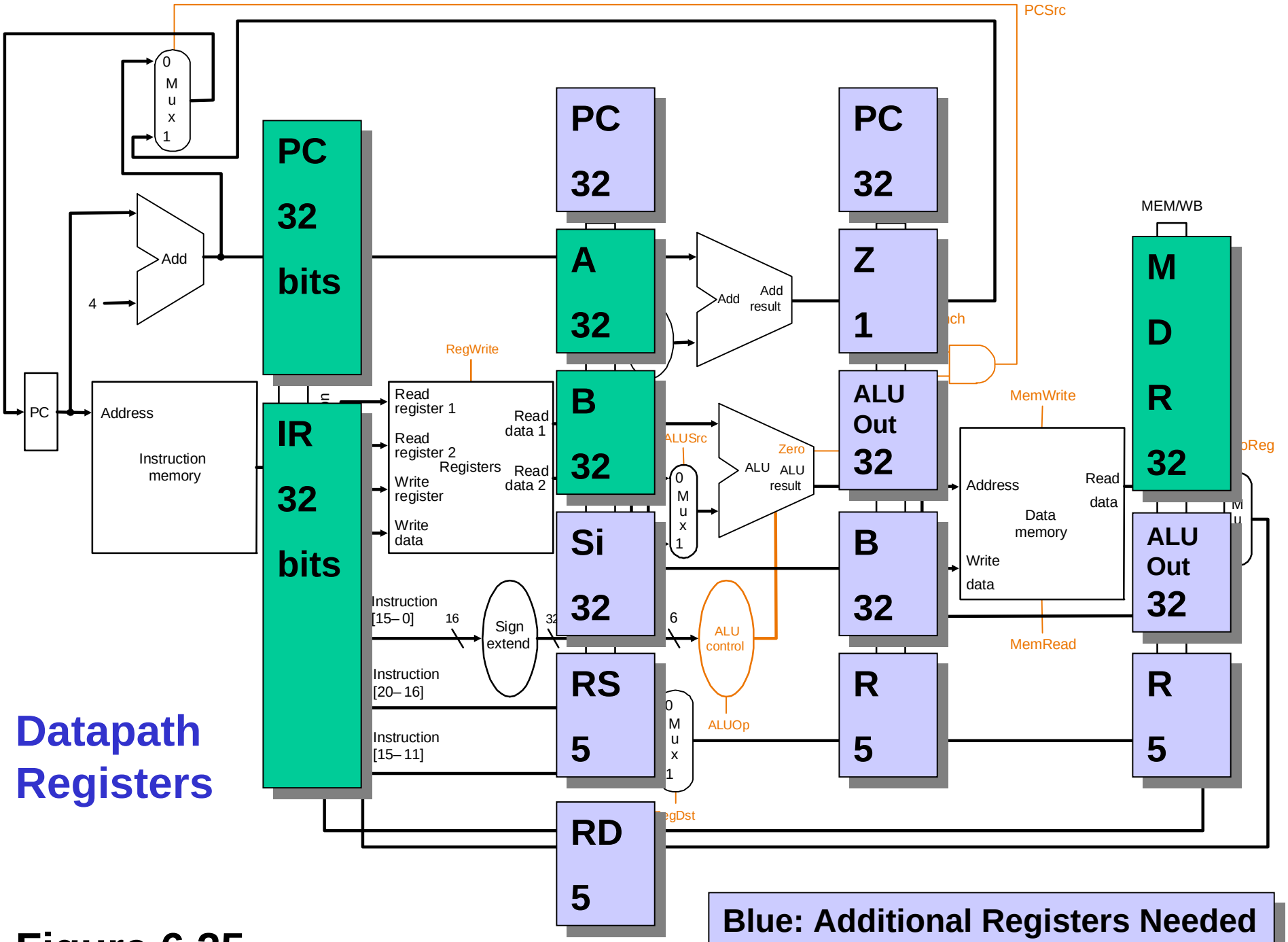
Assume instructions are completely independent!

Maximum Speedup ≤ Number of stages

$$\text{Speedup} \leq \frac{\text{Time for unpipelined operation}}{\text{Time for longest stage}}$$

Recap: Graphically Representing Pipelines





Datapath Registers

Blue: Additional Registers Needed

Figure 6.25

Pipeline Control: Controlpath Register bits

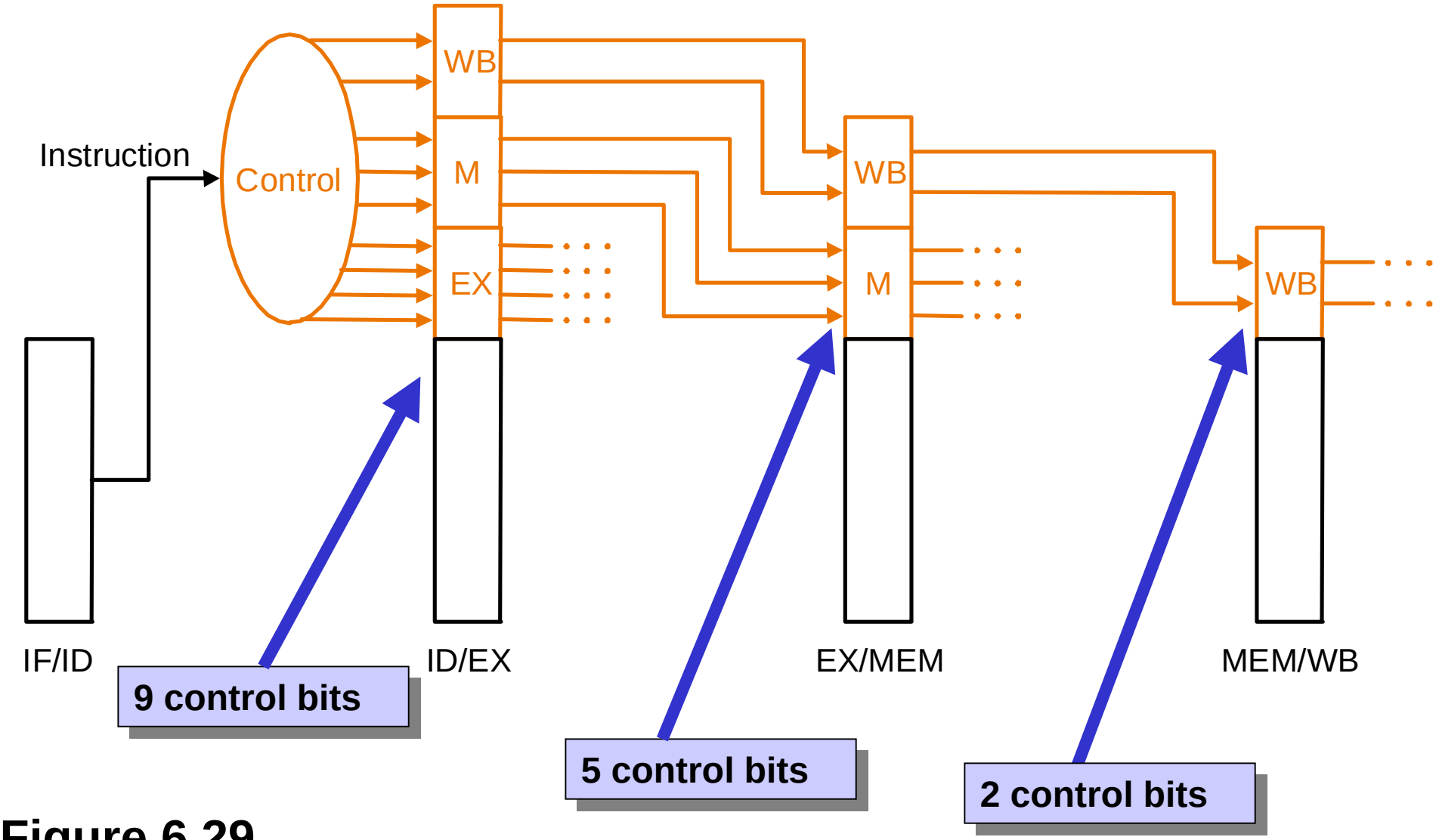


Figure 6.29

Pipeline Control: Controlpath Register bits

Figure 5.20, Single Cycle

Instruction	Reg Dst	ALU Src	Mem Reg	Reg Wrt	Mem Red	Mem Wrt	Bra-nch	ALU op1	ALU op0
R-format	1	0	0	1	0	0	0	1	0
lw	1	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Figure 6.28

	ID / EX control lines				EX / MEM control lines			MEM / WB cntrl lines	
Instruction	Reg Dst	ALU Op1	ALU Op0	ALU Src	Bra-nch	Mem Red	Mem Wrt	Reg Wrt	Mem Reg
R-format	1	1	0	0	0	0	0	1	0
lw	1	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Overhead

Single-cycle model

- 8 ns Clock (125 MHz), (non-overlapping)
- 1 ALU + 2 adders
- 0 Muxes
- 0 Datapath Register bits (Flip-Flops)

Multi-cycle model

- 2 ns Clock (500 MHz), (non-overlapping)
- 1 ALU + Controller
- 5 Muxes
- 160 Datapath Register bits (Flip-Flops)

Pipeline model

- 2 ns Clock (500 MHz), (overlapping)
- 2 ALU + Controller
- 4 Muxes
- 373 Datapath + 16 Controlpath Register bits (Flip-Flops)

Pipeline Datapath and Controlpath

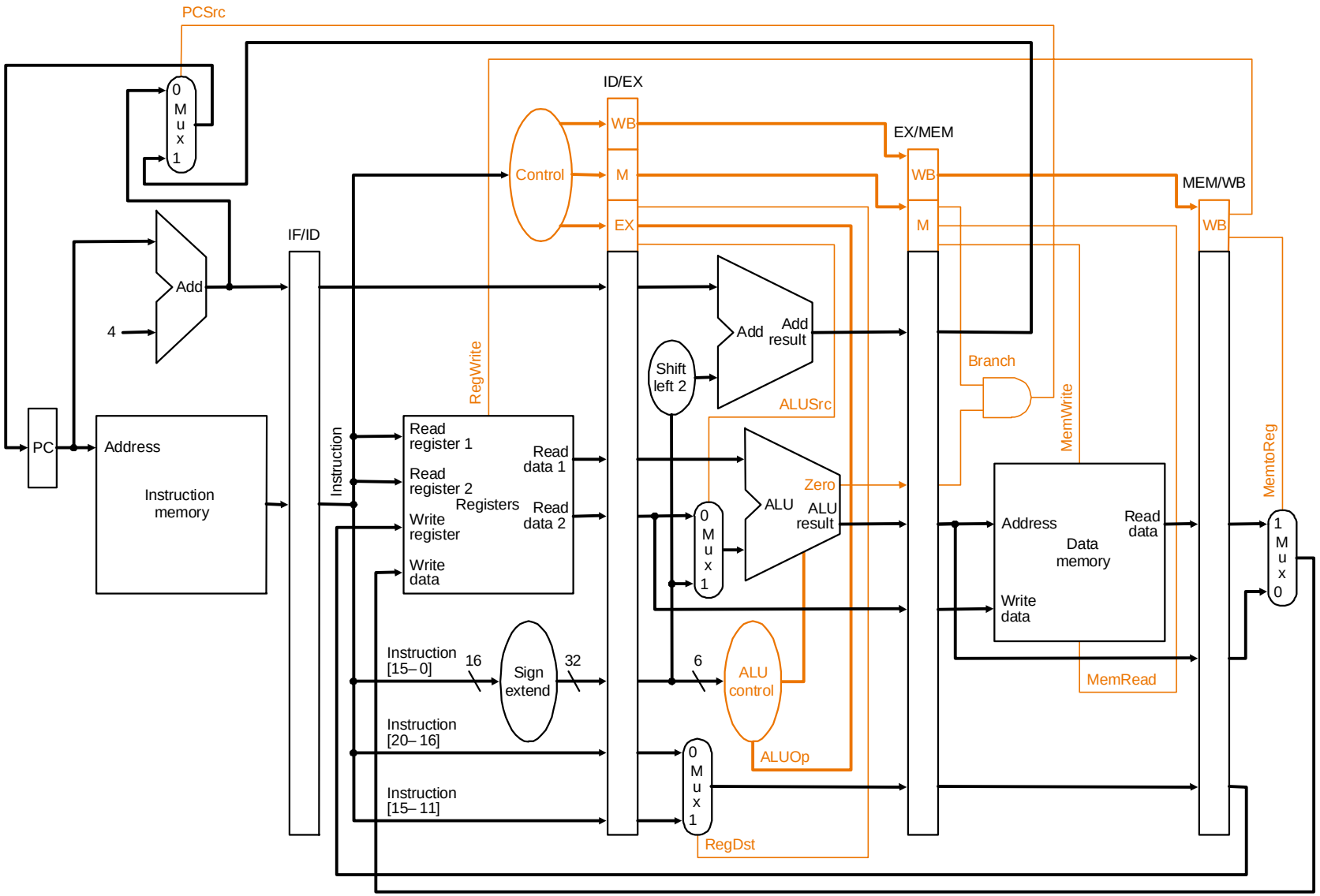
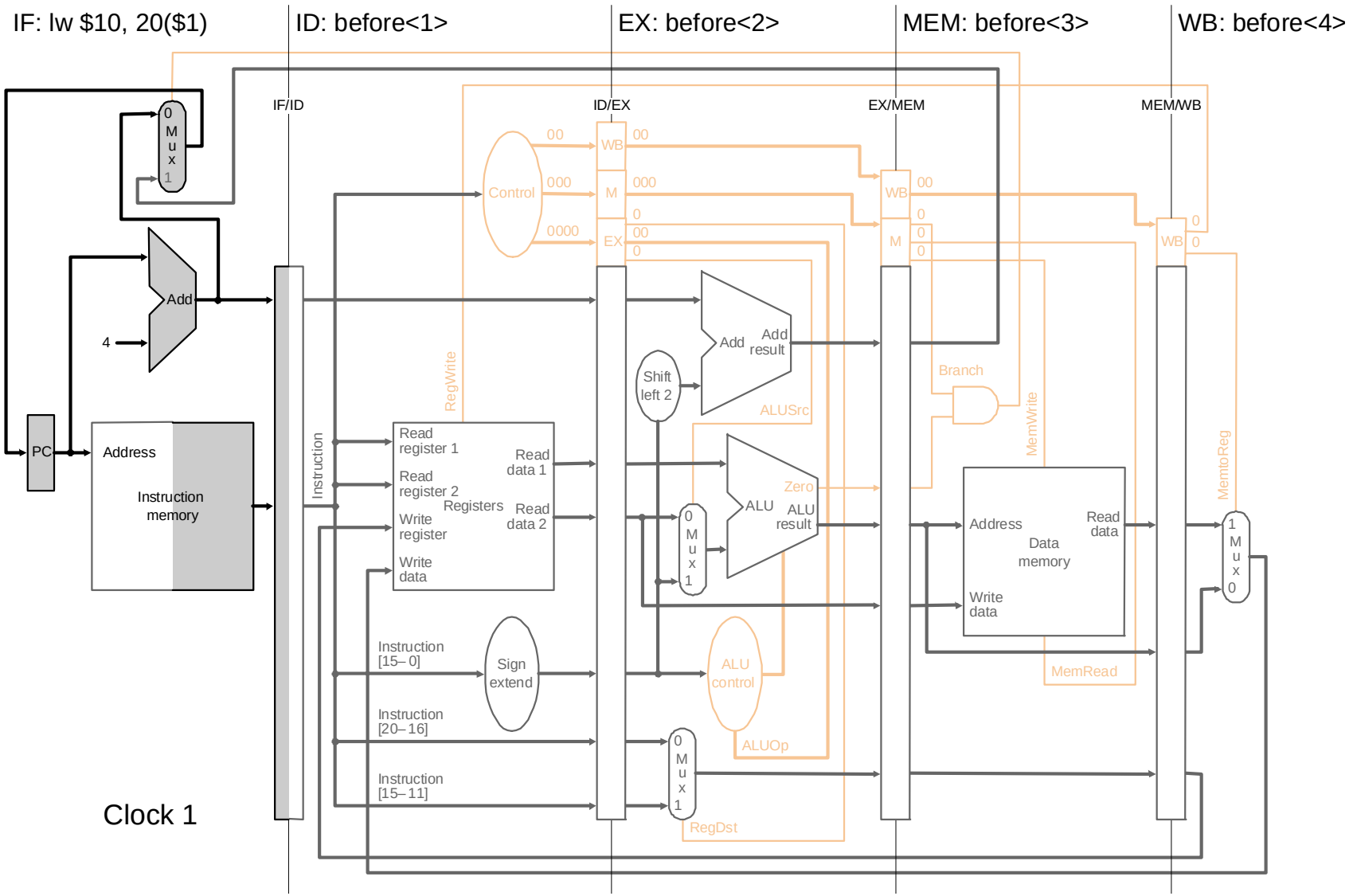


Figure 6.30

Figure 6.31a


IF: lw $10, 20($1)
ID: before<1>
EX: before<2>
MEM: before<3>
WB: before<4>
IF/ID
ID/EX
EX/MEM
MEM/WB
0
M
u
x
1
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Control
00
000
0000
WB
M
EX
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALUSrc
ALU
Zero
ALU result
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15–0]
Sign extend
Instruction [20–16]
Instruction [15–11]
ALU control
ALUOp
RegDst
Clock 1

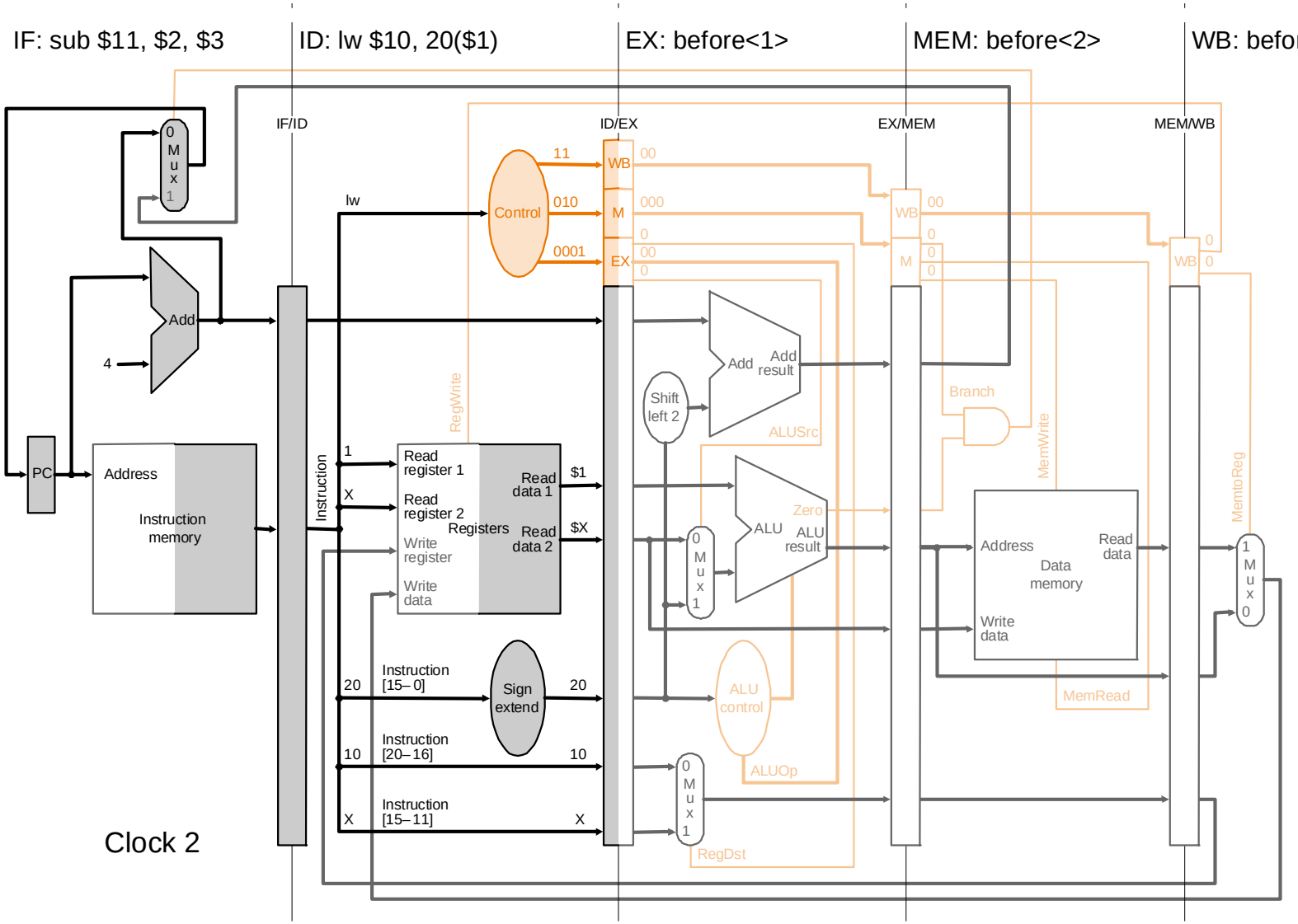
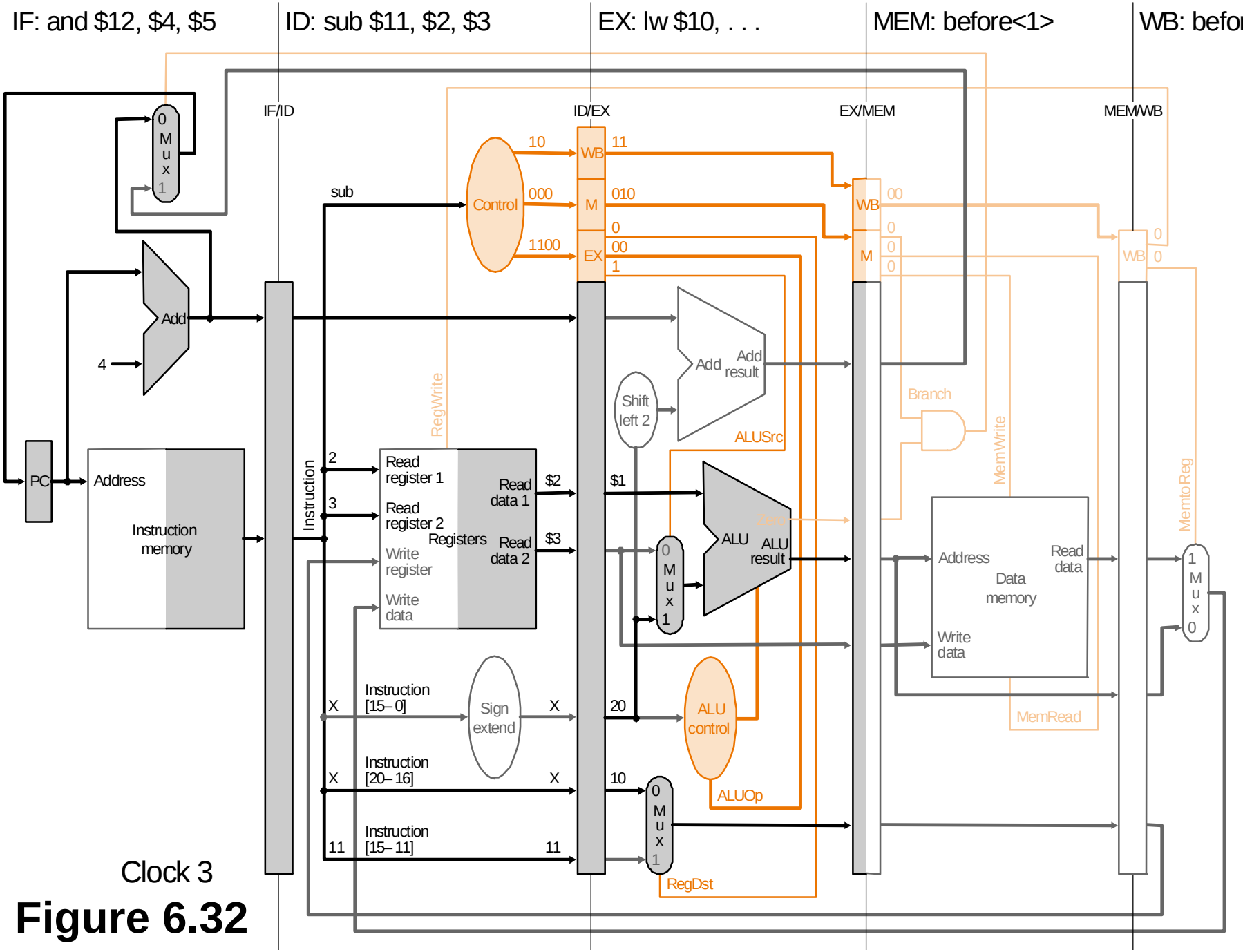

IF: sub $11, $2, $3
ID: lw $10, 20($1)
EX: before<1>
MEM: before<2>
WB: before<3>
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 2

Figure 6.31b



Clock 3

Figure 6.32

Figure 6.32b

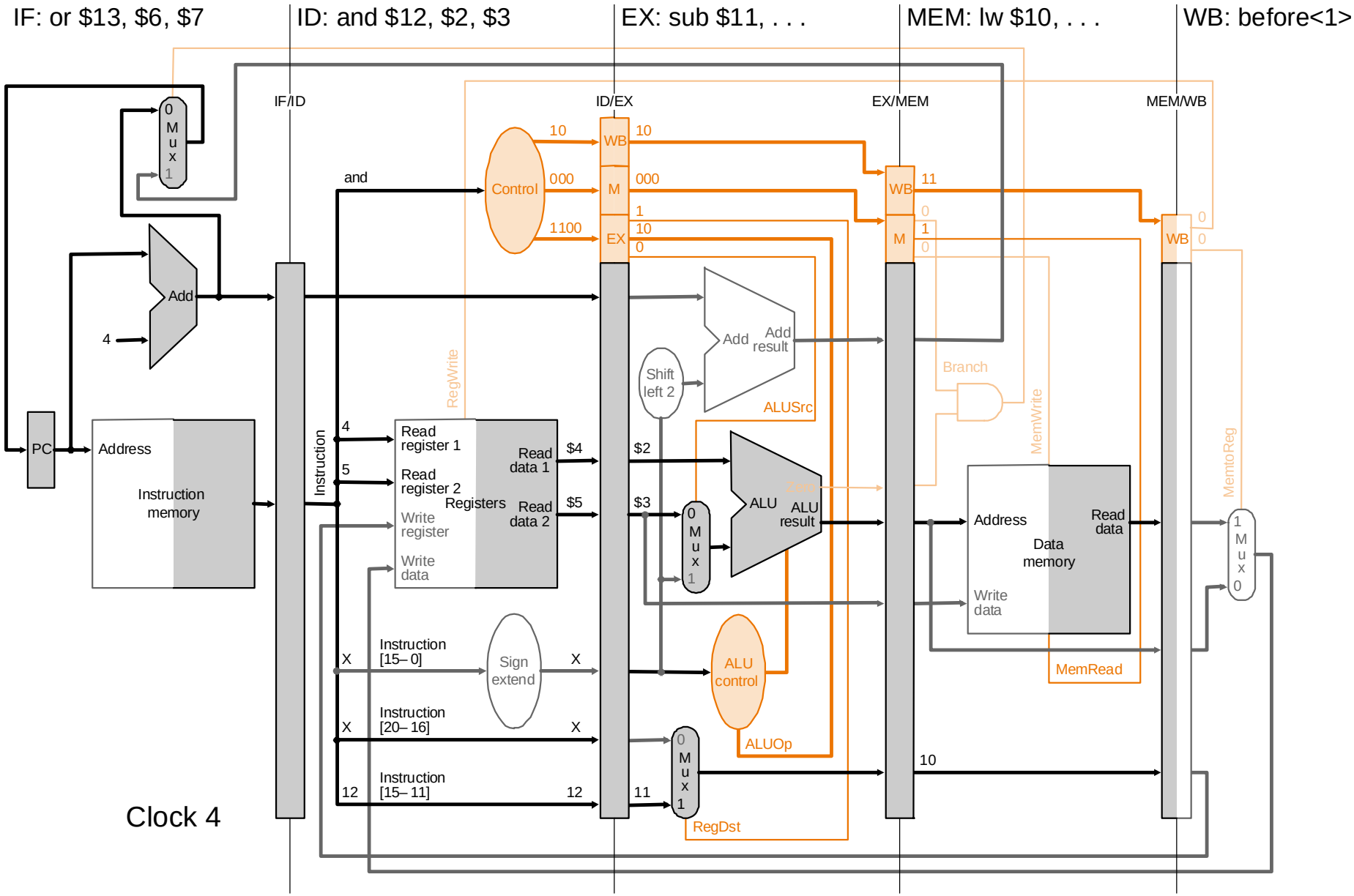

IF: or $13, $6, $7
ID: and $12, $2, $3
EX: sub $11, . . .
MEM: lw $10, . . .
WB: before<1>
IF/ID
ID/EX
EX/MEM
MEM/WB
0 M u x 1
Add
4
PC
Address
Instruction memory
Instruction
and
Control
10
000
1100
WB
M
EX
10
000
1
10
0
RegWrite
Add
Add result
Shift left 2
ALUSrc
4
5
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
$4
$5
$2
$3
ALU
ALU result
Zero
Branch
MemWrite
MemtoReg
Address
Data memory
Read data
Write data
MemRead
Instruction [15– 0]
Sign extend
X
Instruction [20– 16]
Instruction [15– 11]
12
11
ALU control
ALUOp
RegDst
11
0
1
10
Clock 4

Figure 6.33

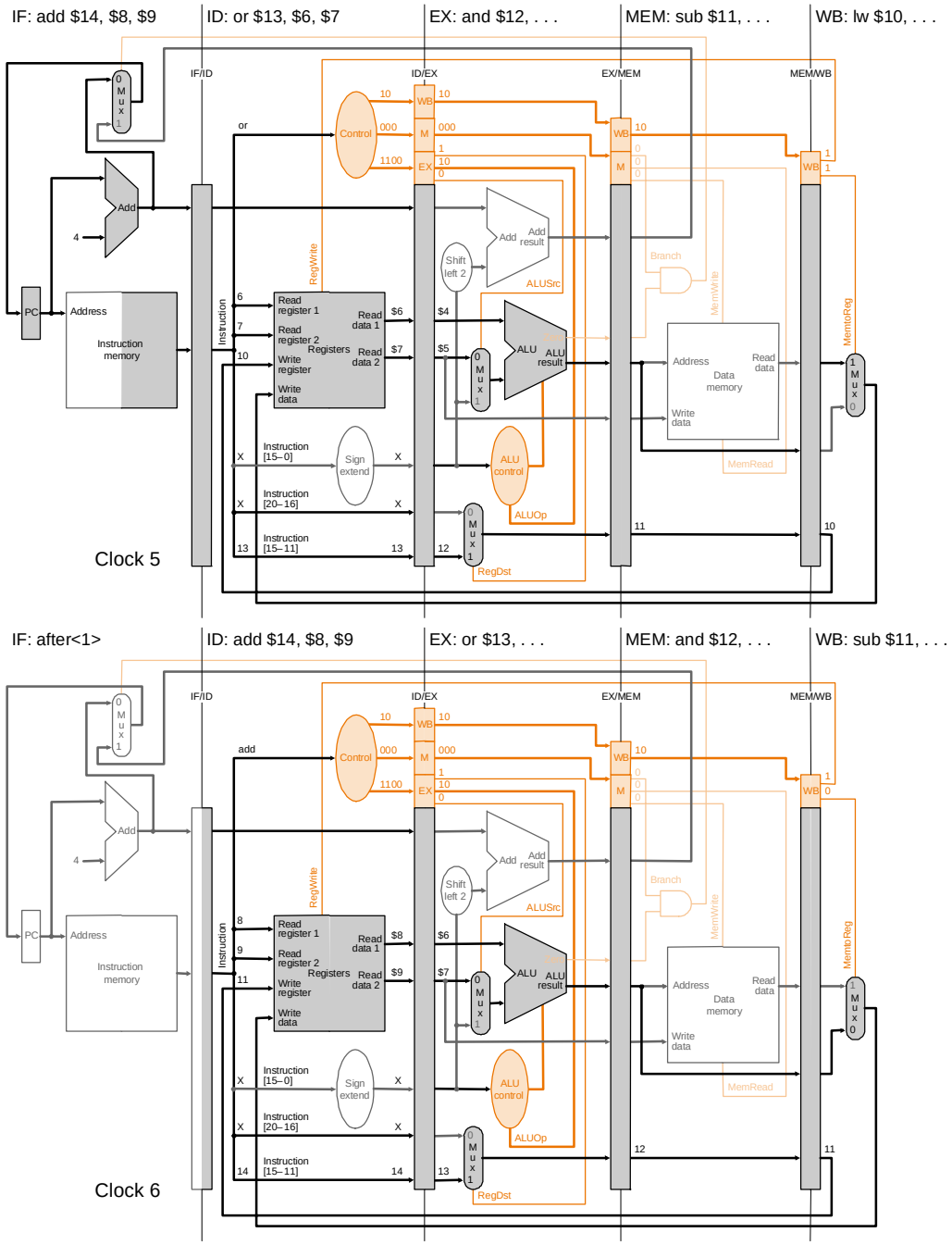

IF: add $14, $8, $9
ID: or $13, $6, $7
EX: and $12, . . .
MEM: sub $11, . . .
WB: lw $10, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 5
IF: after<1>
ID: add $14, $8, $9
EX: or $13, . . .
MEM: and $12, . . .
WB: sub $11, . . .
Clock 6
Add
Shift left 2
ALU
Registers
Instruction memory
Data memory
Sign extend
ALU control
Control
RegWrite
ALUSrc
ALUOp
RegDst
MemRead
MemWrite
Branch
MemtoReg

Figure 6.34

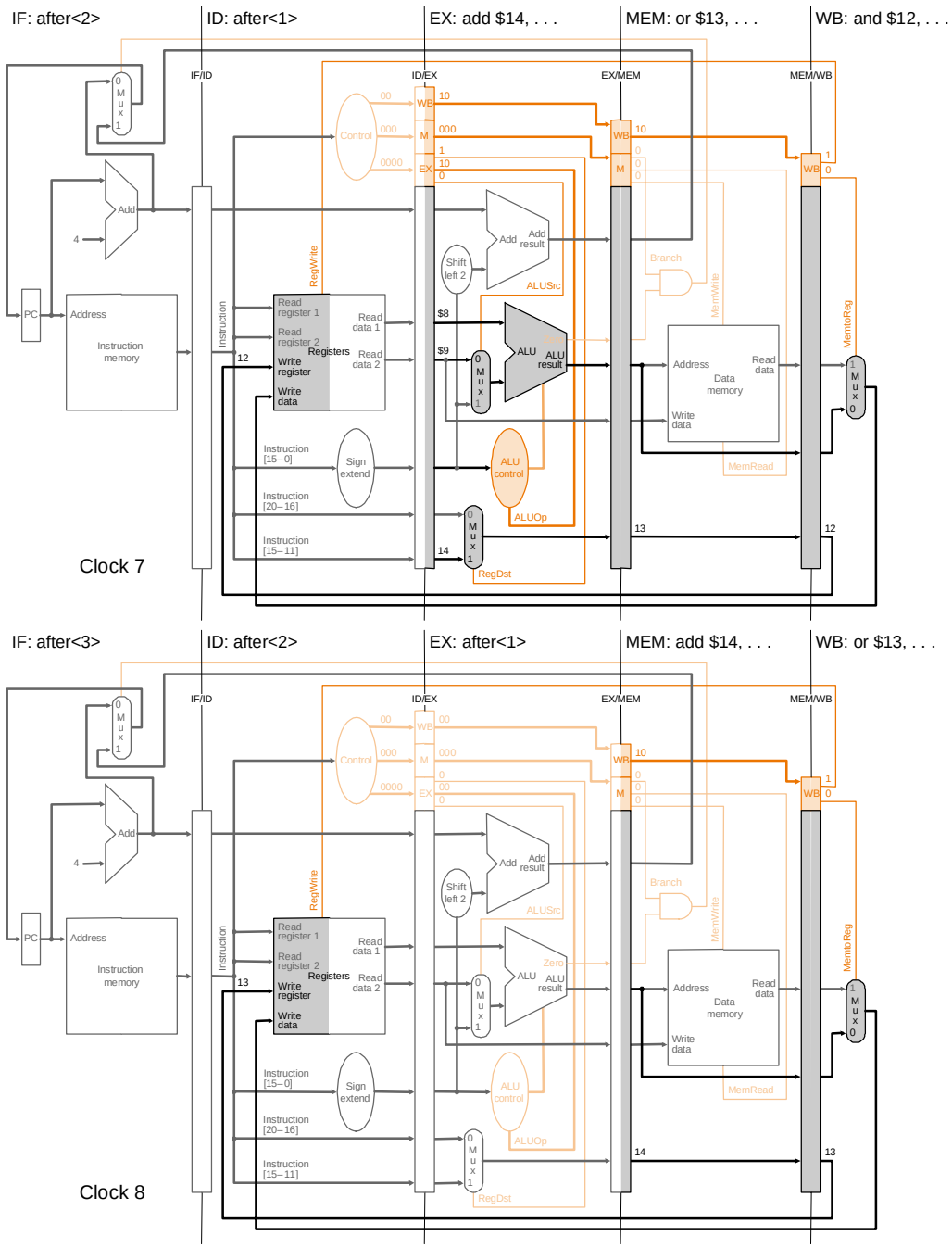
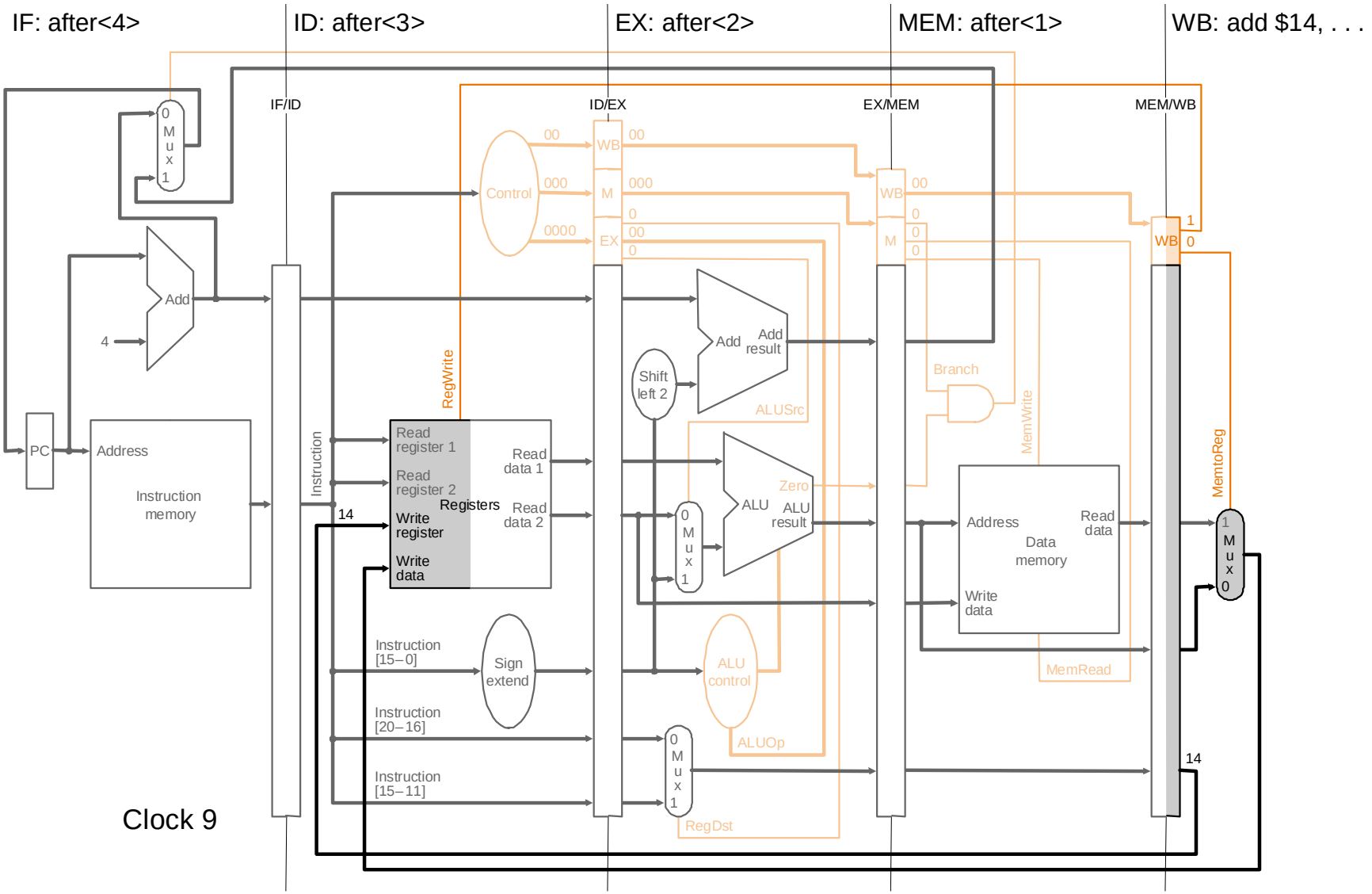

IF: after<2>
ID: after<1>
EX: add $14, . . .
MEM: or $13, . . .
WB: and $12, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
Clock 7
IF: after<3>
ID: after<2>
EX: after<1>
MEM: add $14, . . .
WB: or $13, . . .
Clock 8
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Control
Shift left 2
Add result
ALUSrc
ALU
Zero
ALU result
Branch
MemWrite
MemRead
MemtoReg
Data memory
Read data
Sign extend
ALU control
ALUOp
RegDst
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
$8
$9
12
13
14

Figure 6.35


IF: after<4>
ID: after<3>
EX: after<2>
MEM: after<1>
WB: add $14, . . .
IF/ID
ID/EX
EX/MEM
MEM/WB
0 M u x 1
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
14
Control
00
000
0000
WB
M
EX
Shift left 2
Add
Add result
ALUSrc
Zero
ALU
ALU result
Branch
MemWrite
Address
Data memory
Read data
Write data
MemRead
MemtoReg
1 M u x 0
Instruction [15–0]
Sign extend
Instruction [20–16]
Instruction [15–11]
ALU control
ALUOp
RegDst
Clock 9

Pipeline Datapath and Controlpath

Clock	<IF/ID>	<ID/EX>	<EX/MEM>	<MEM/WB>
	<PC, IR>	<PC, A, B, S, Rt, Rd>	<PC, Z, ALU, B, R>	<MDR, ALU, R>
0	<0,?>	<?,?,?,?,?,?>	<?,?,?,?,?>	<?,?,?>
1	<4,lw $10,20($1)>	<0,?,?,?,?,?>	<?,?,?,?,?>	<?,?,?>
2	<8,sub $11,$2,$3>	<4,C$1,C$10,20,$10,X>	<0,?,?,?,?>	<?,?,?>
3	<12,and $12,$4,$5>	<8,C$2,C$3,X,$3,$11>	<84,0,23,8,$10>	<?,?,?>
4	<16,or $13,$6,$7>	<12,C$4,C$5,X,$5,$12>	<X,1,0,4,$11>	<9,23,$10>
5	<20,add $14,$8,$9>	<16,C$6,C$7,X,$7,$13>	<X,0,1,7,$12>	<9,23,$10>

Contents of Register 1 = C$1 = 3; C$2=4; C$3=4; C$4=6; C$5=7; C$10=8; … Memory[23]=9;
Formats: add $rd,$rs=A,$rt=B; lw $rt=B,@($rs=A)