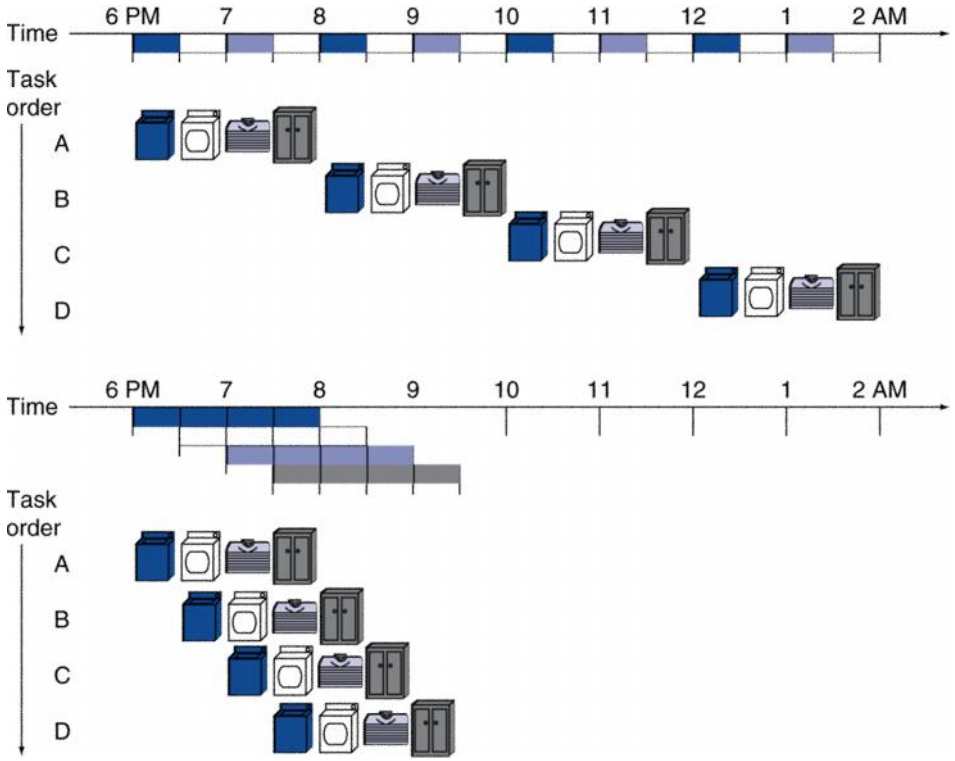
Pipelining -- Outline

- Basic idea
- Pipelined datapath
- Pipelined control
- Data hazards
- Three methods to handle data hazards

Pipelining Analogy

- Pipelined laundry: overlapping execution
  - Parallelism improves performance

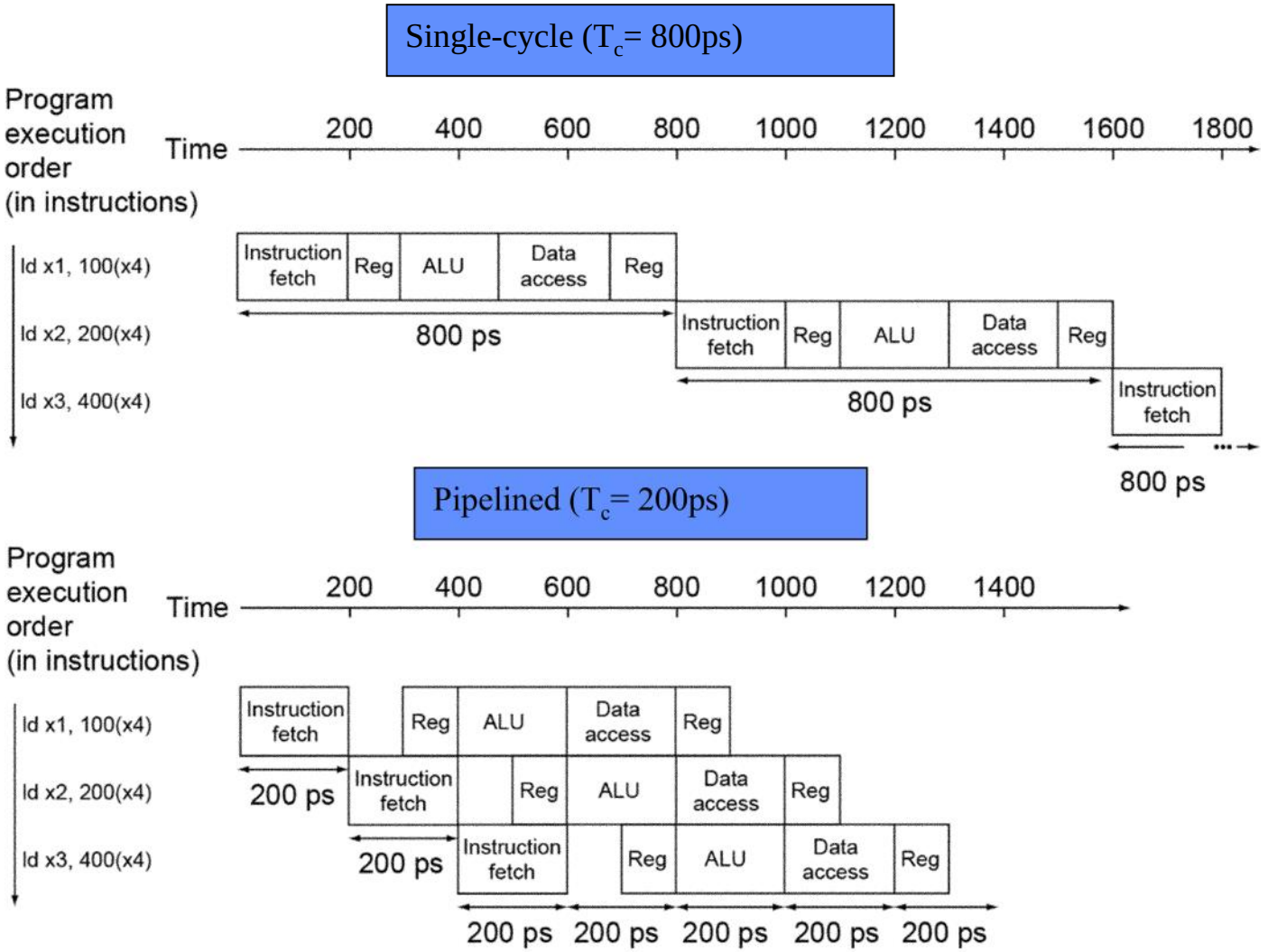


- Total time:
- Total time:
- Speedup
  = number of stages

Basic idea of pipeline

- For single cycle computer, roughly, 5 steps
  - IF, ID, EX, MEM, WB
- For the current step, hardware for other steps are idle.
- Use the idle hardware to work on other instructions
  - Overlap the instruction executions.
  - Add buffers to hold partial results of instruction execution

Pipeline Performance

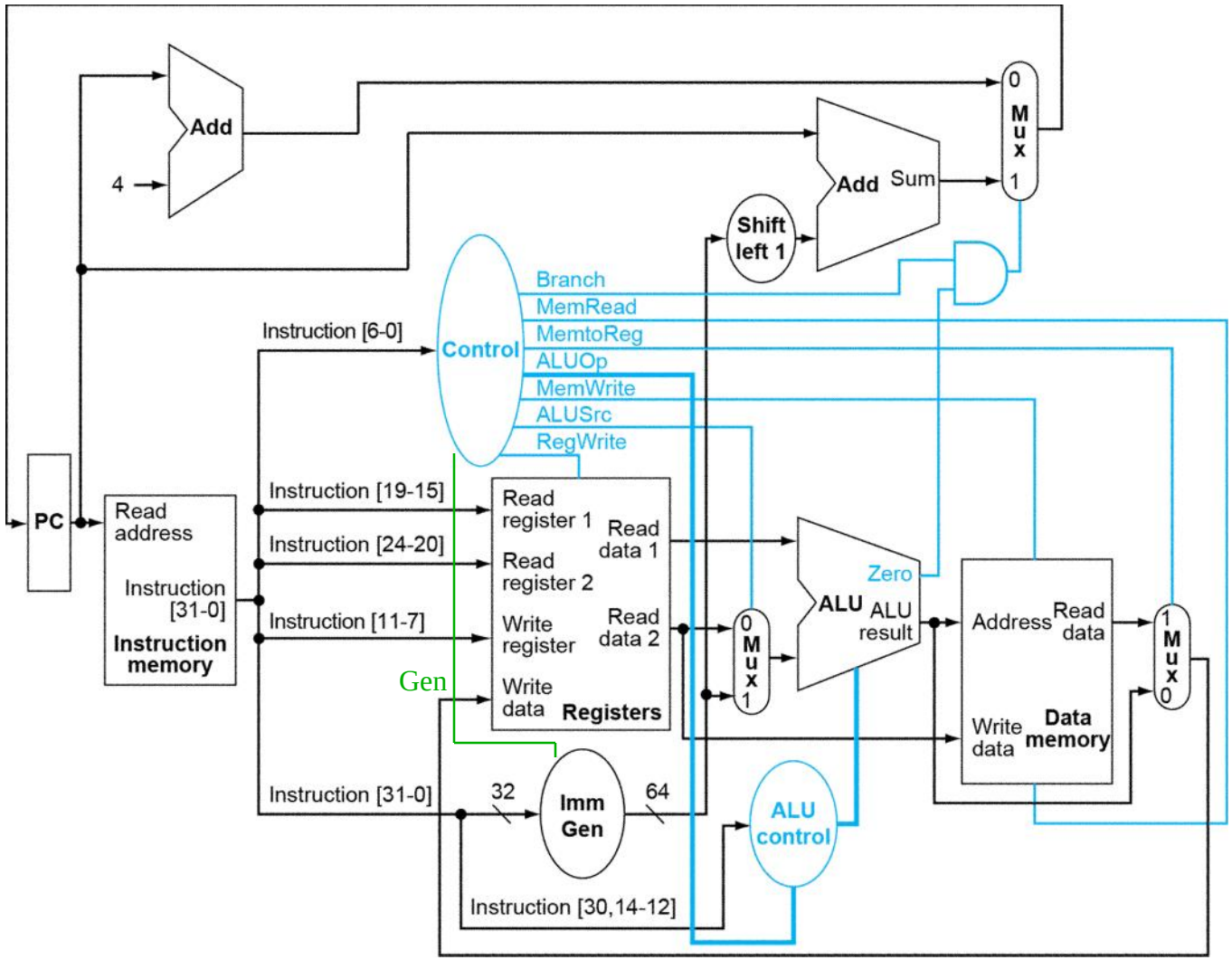


Pipelining: Basic Idea (con't)

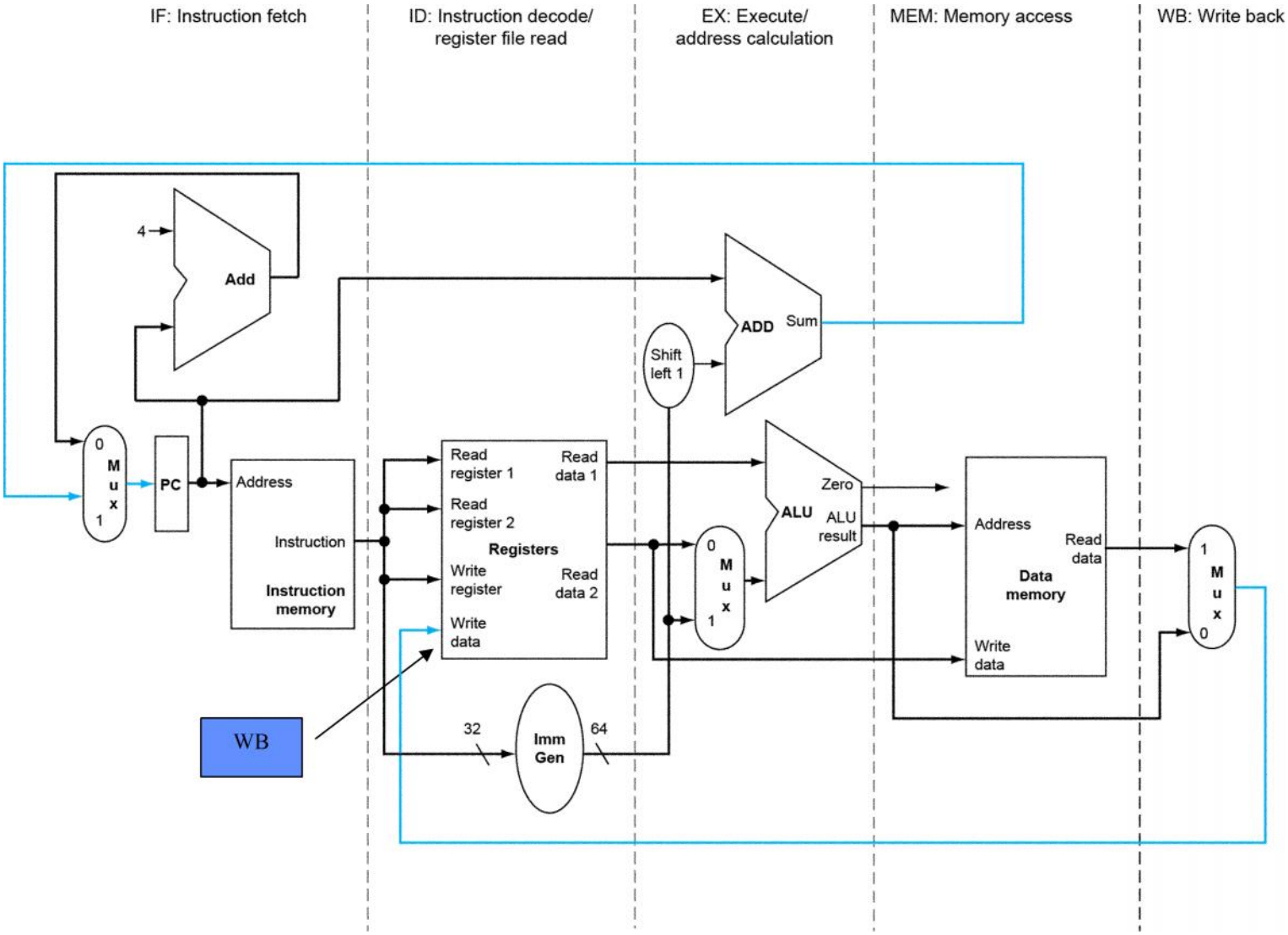
- Pipeline: implementation technique in which multiple inst. are overlapped in execution
  - Improve inst. throughput rather than inst. execution time
  - speedup = $\frac{Execution\ tim\ (nonoverlpped\ )}{Execution\ time\ (overlapped\ )} \approx \#\ stages$
  - pipe stage: balancing length of each stage with equal length, limited # pipe stages

MIPS Pipe: 5 stages

- IF: instruction fetch
- ID: inst. decode and register fetch
- EX: execution and effective address calculation
- MEM: memory access
- WB: write back to register

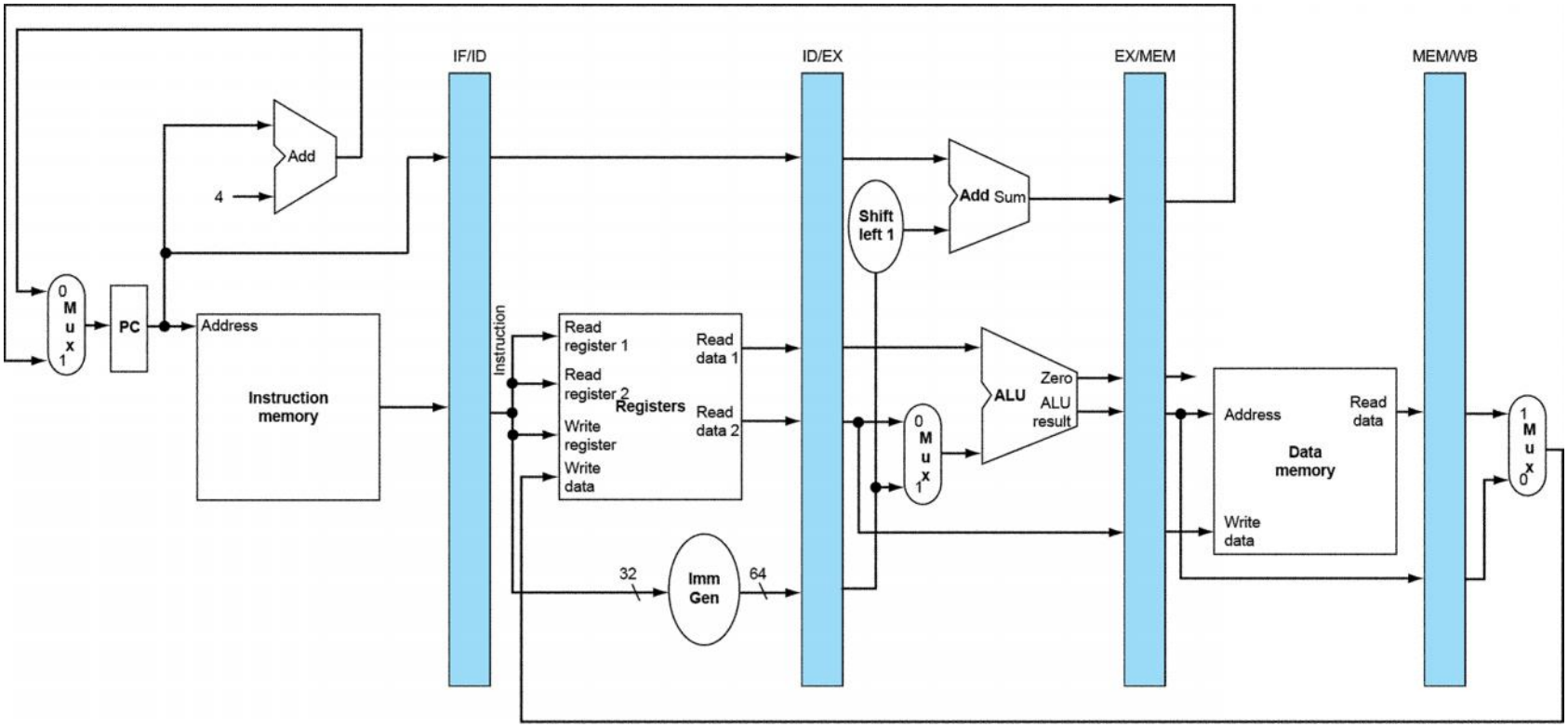

Add
4
Add
Sum
Shift left 1
0
M
u
x
1
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [6-0]
PC
Read address
Instruction [31-0]
Instruction memory
Instruction [19-15]
Instruction [24-20]
Instruction [11-7]
Instruction [31-0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Gen
32
Imm Gen
64
Instruction [30,14-12]
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory
1
M
u
x
0

RISC-V Pipelined Datapath

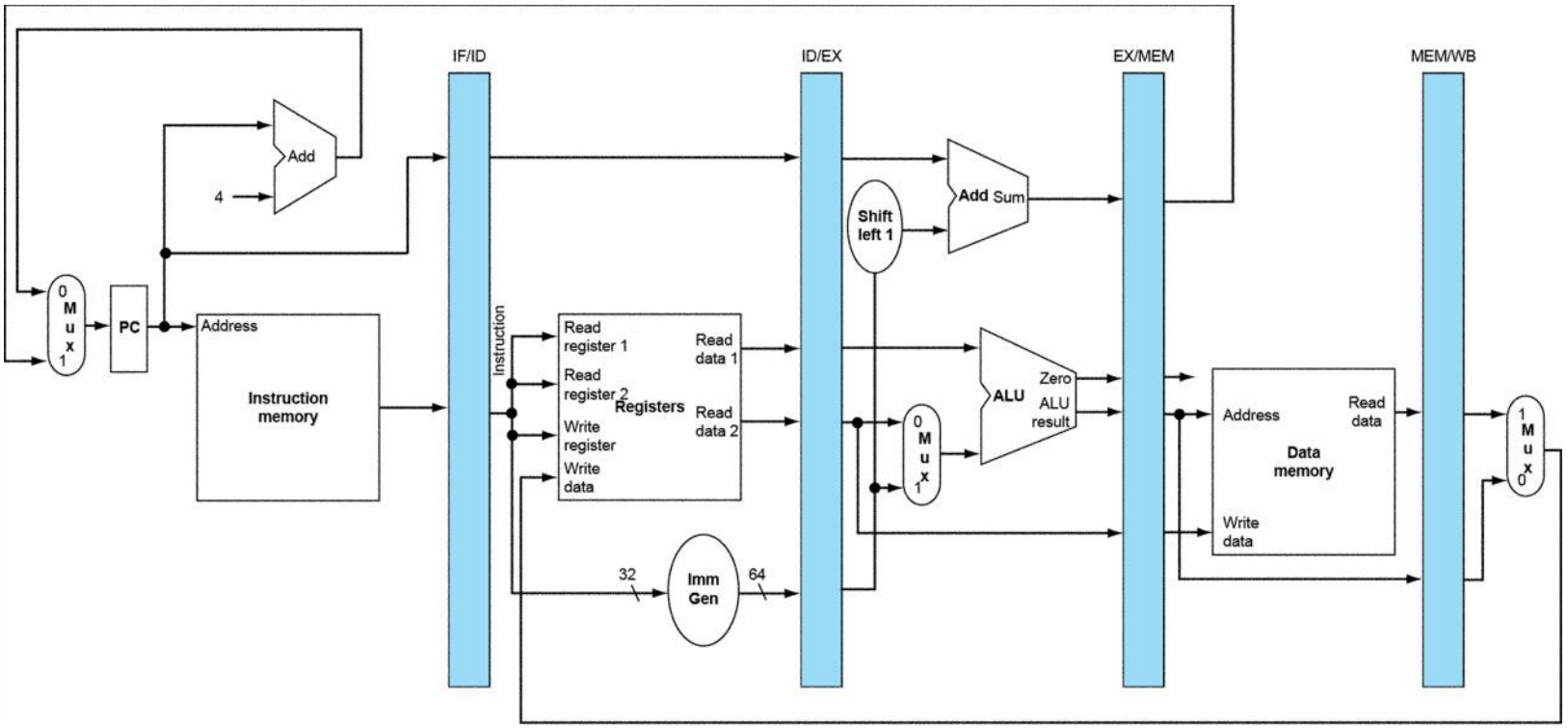


Pipeline registers

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



Pipelining -- Control?

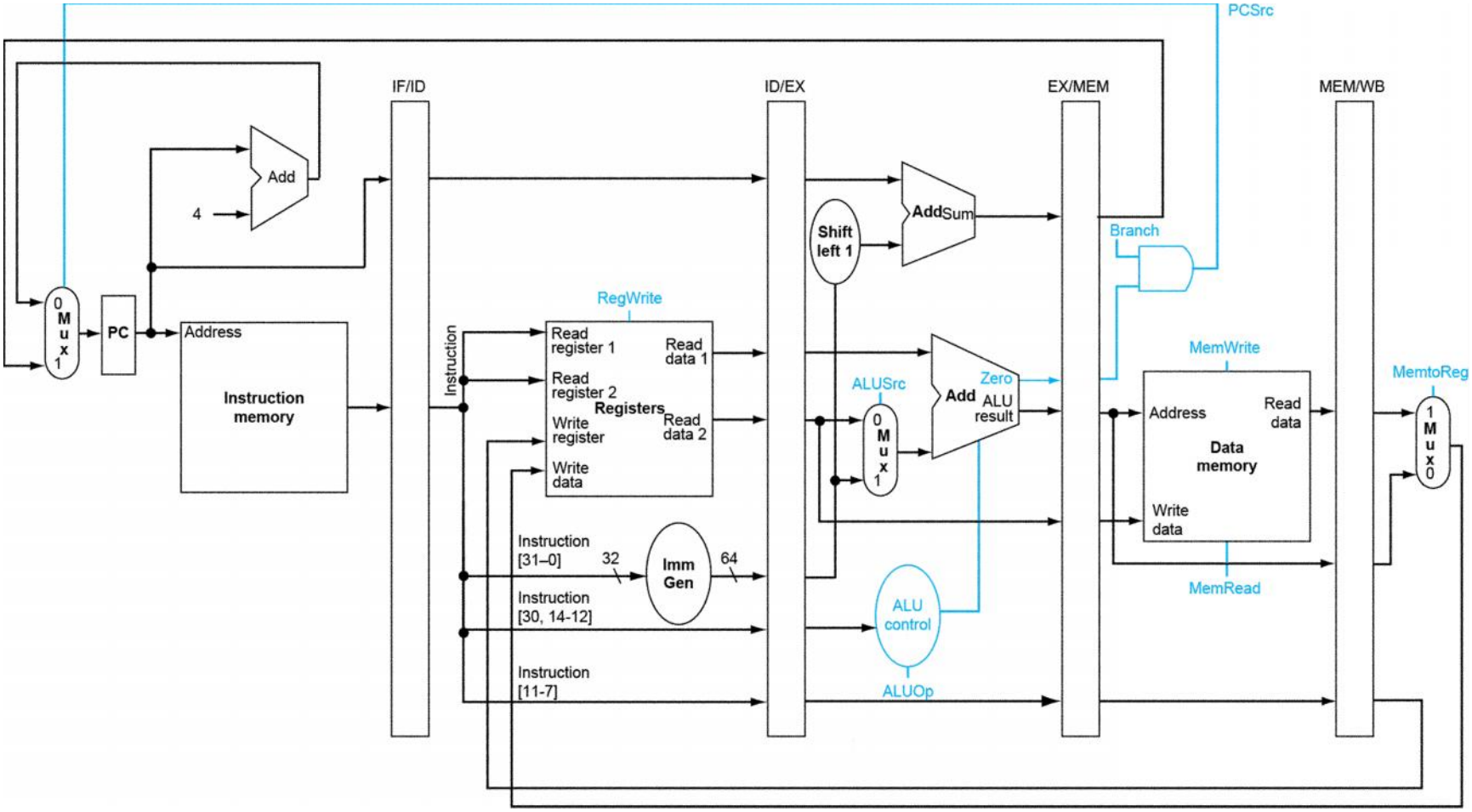


Pipeline control

- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC Increment
  - Instruction Decode / Register Fetch
  - Execution
  - Memory Stage
  - Write Back

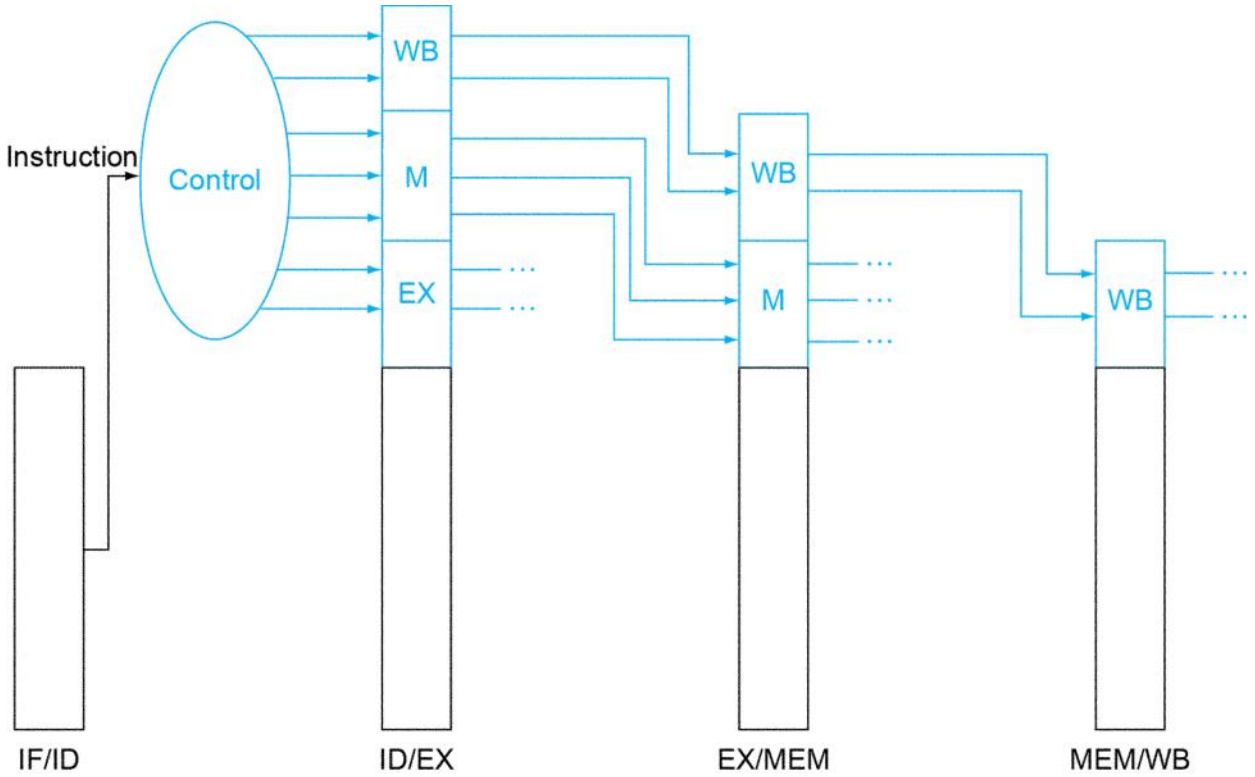
- Generate all control signals in the ID stage and pipeline them

Pipelined Control (Simplified)

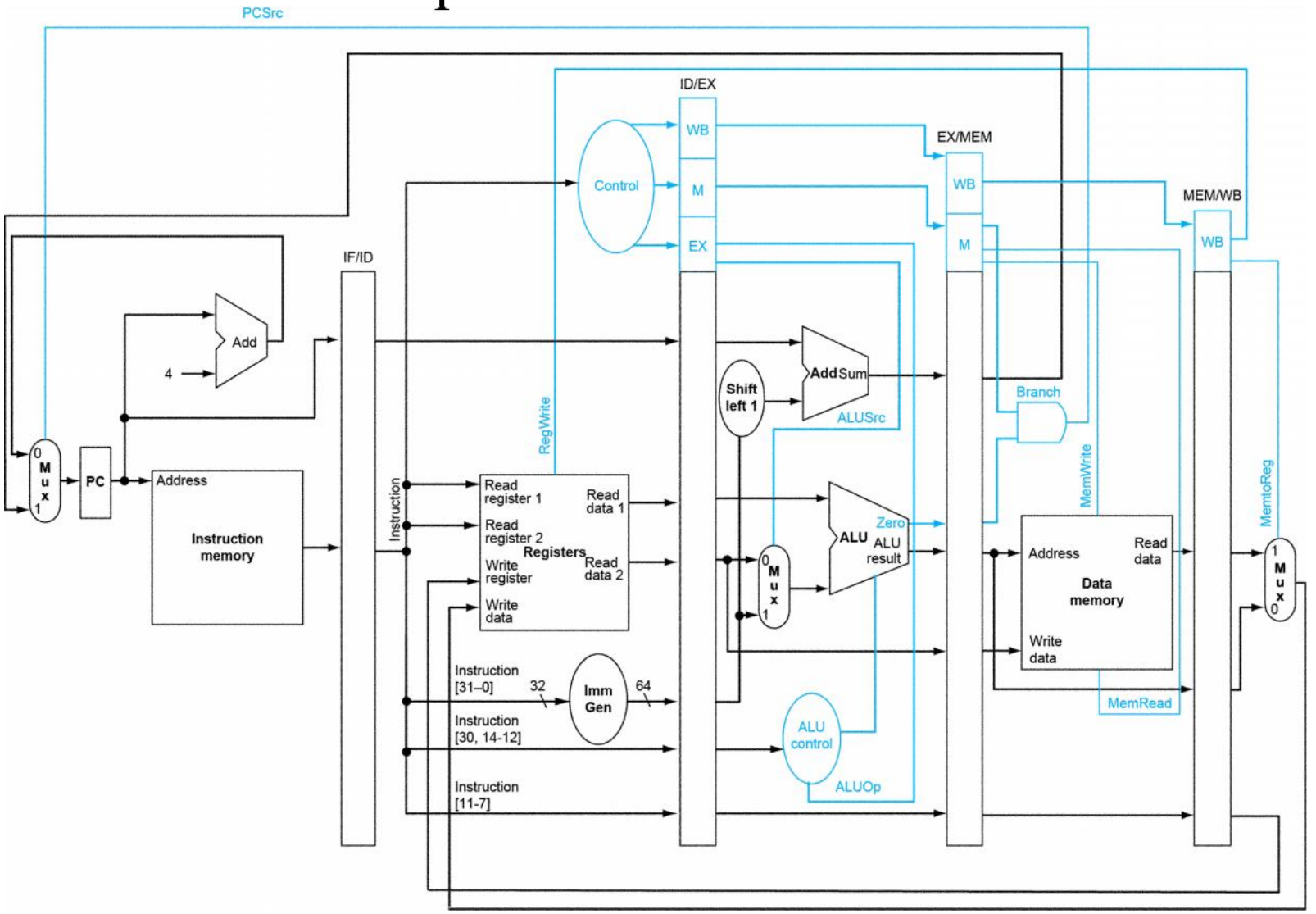


Pipelined Control

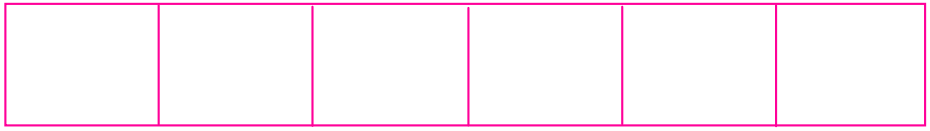
- Control signals derived from instruction
  - as in single-cycle implementation
  - passed along just like the data



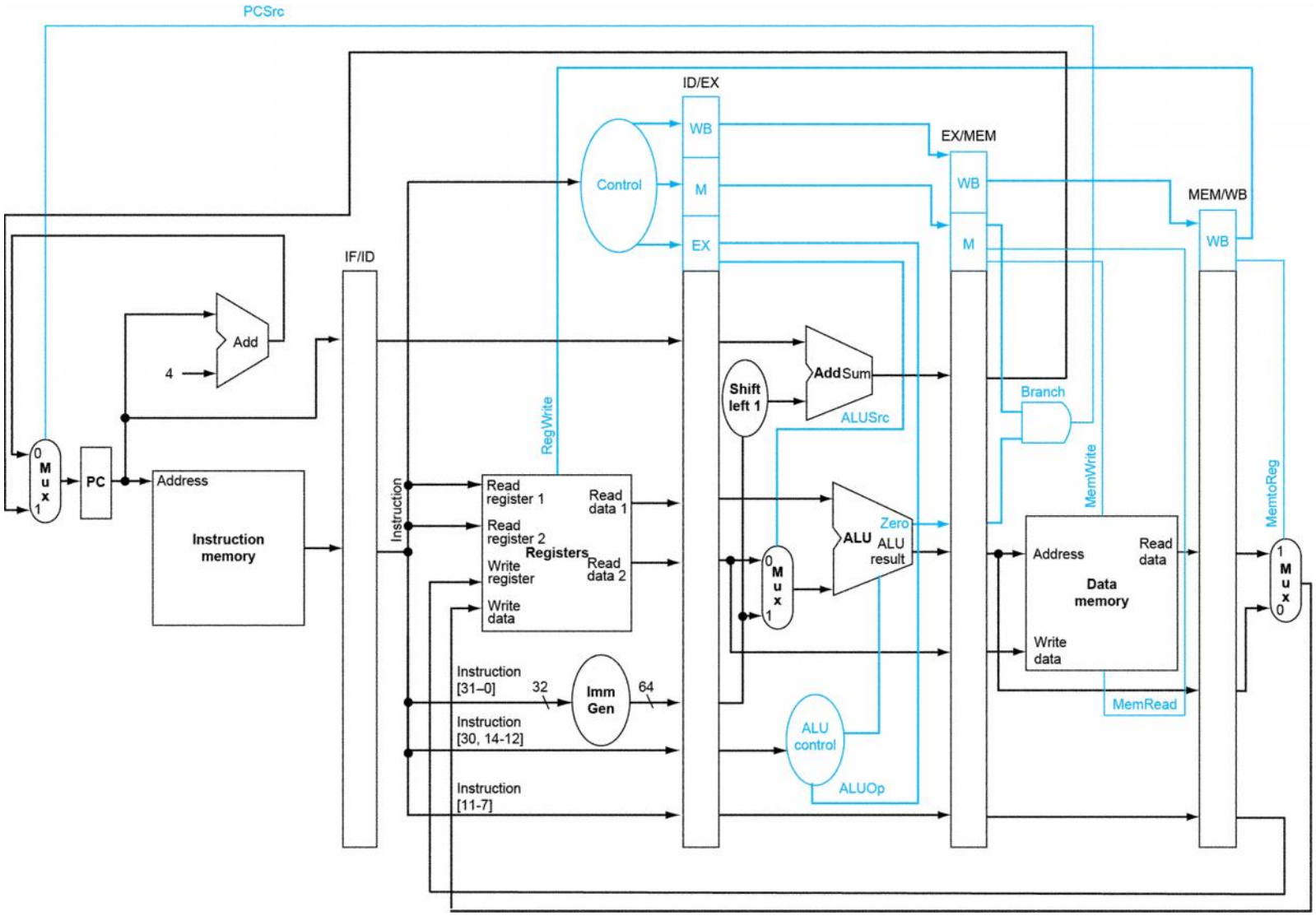
Pipelined Control

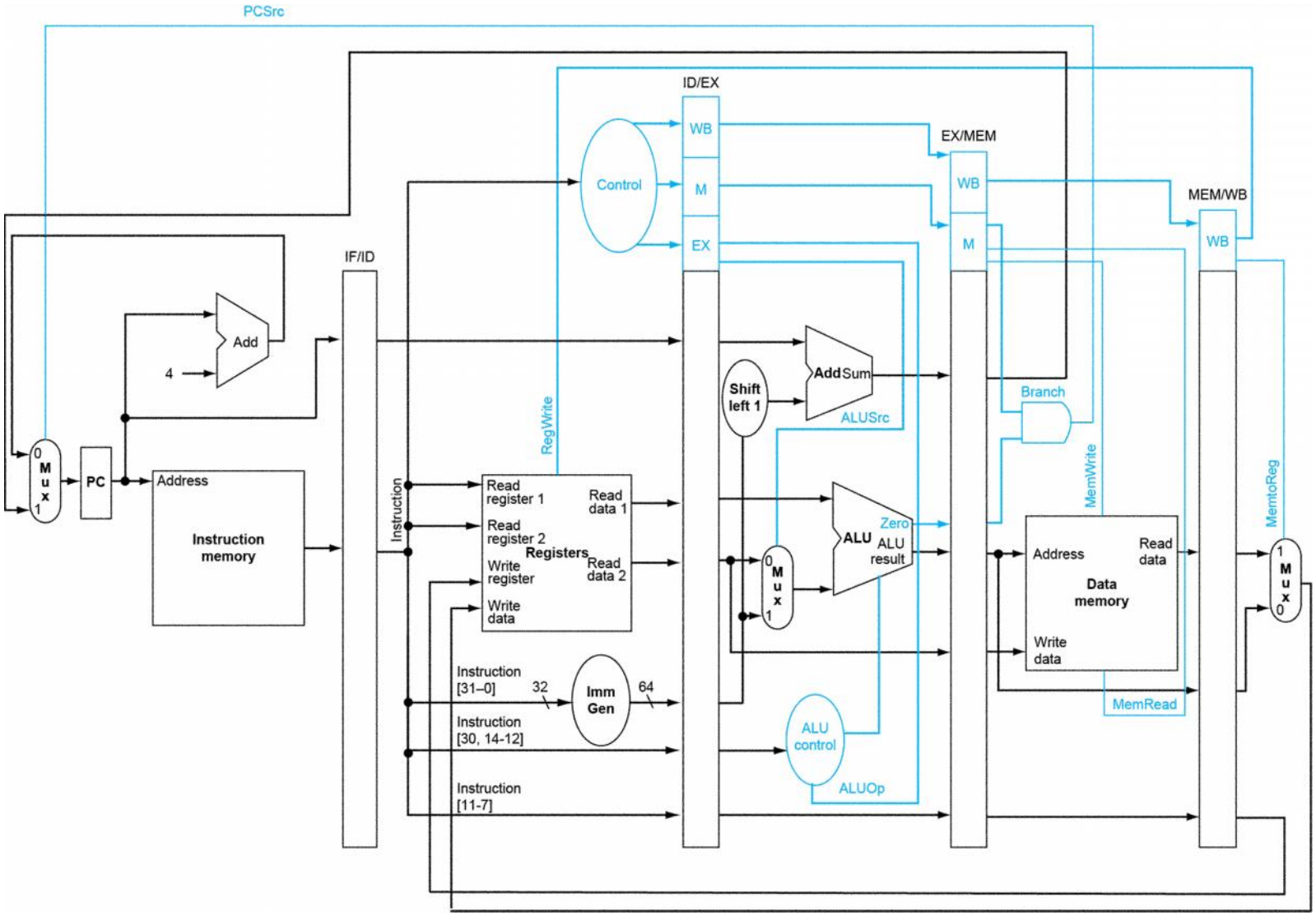


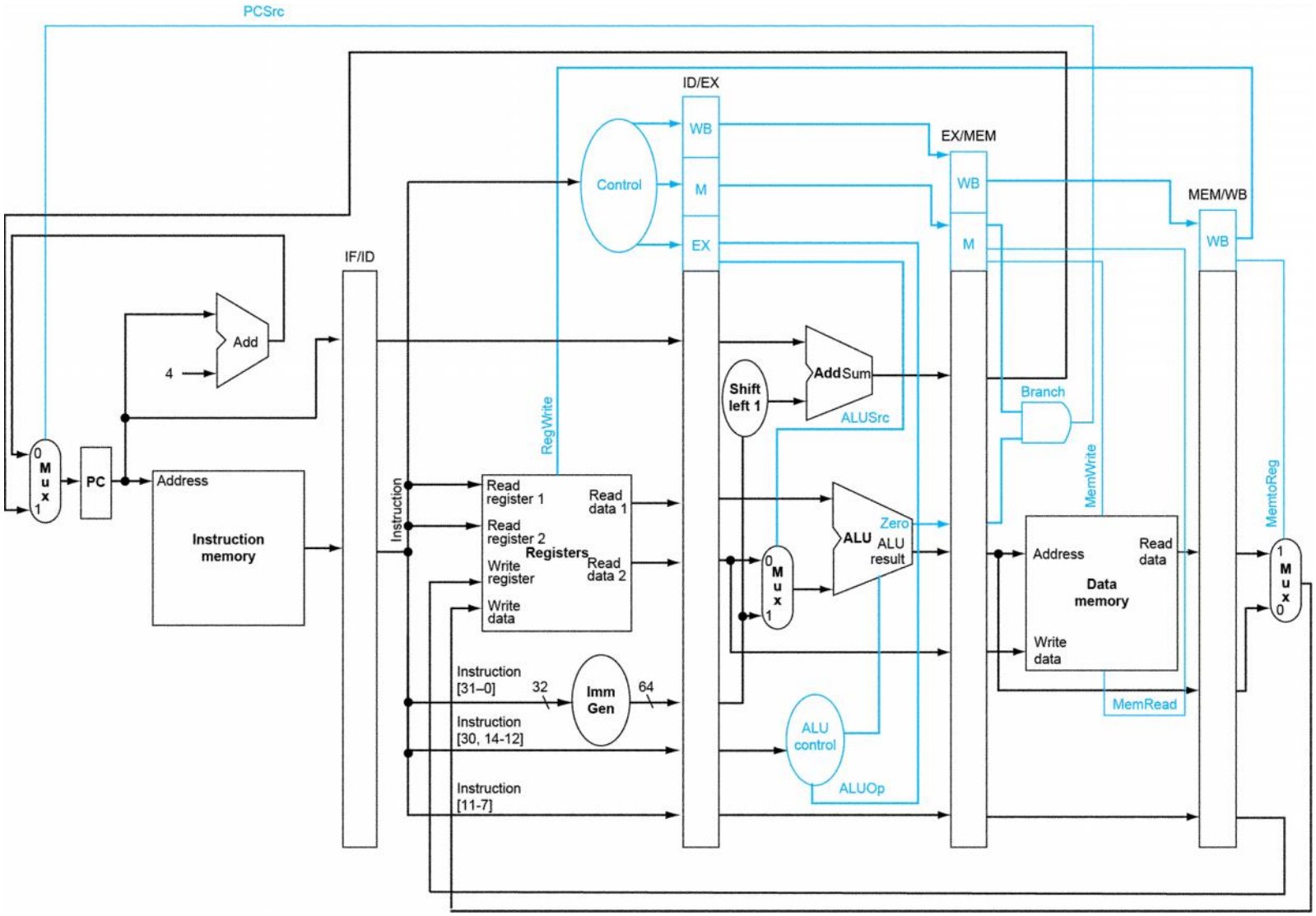
Example: How `add x4, x2, x3` pass the Pipe

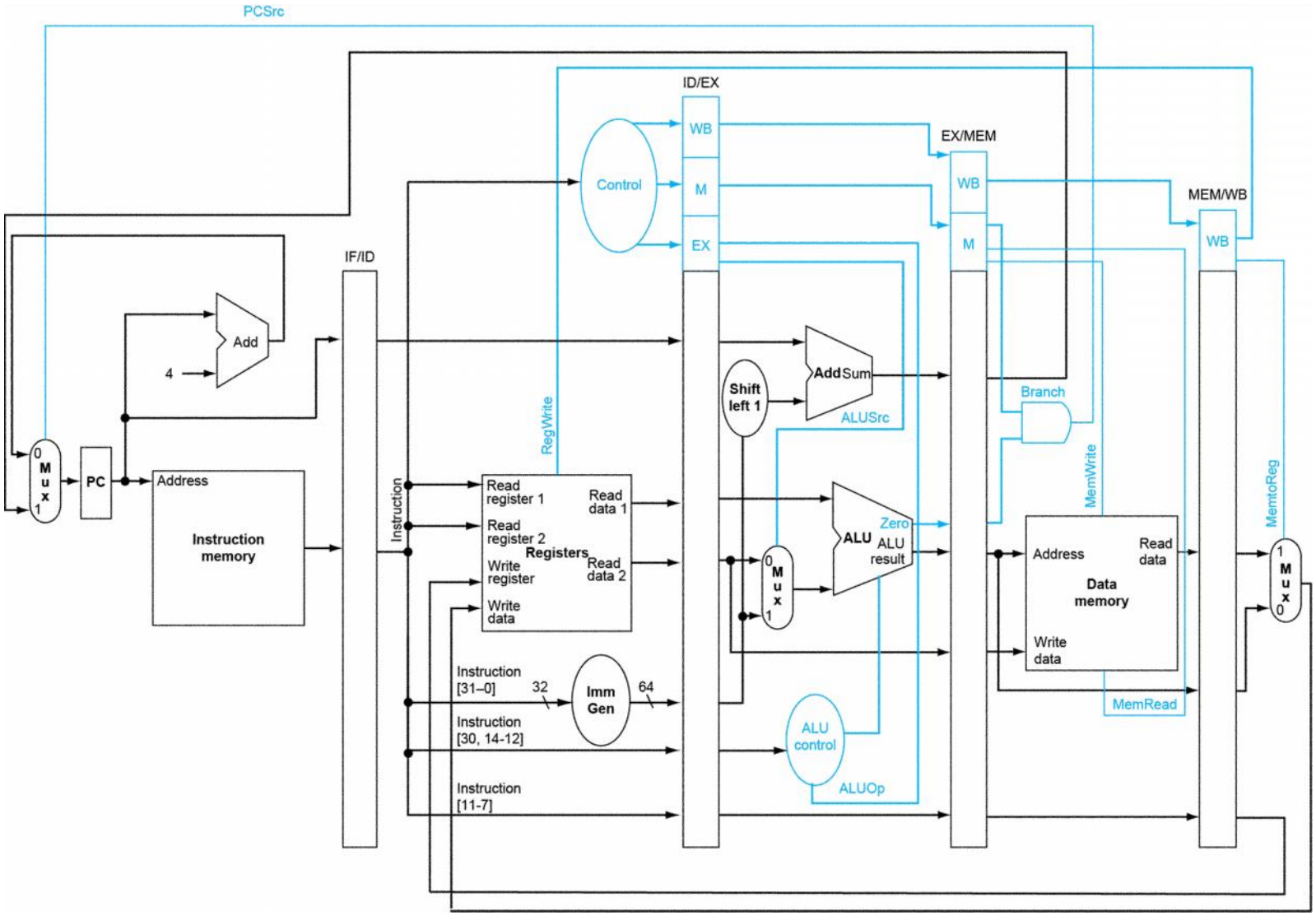


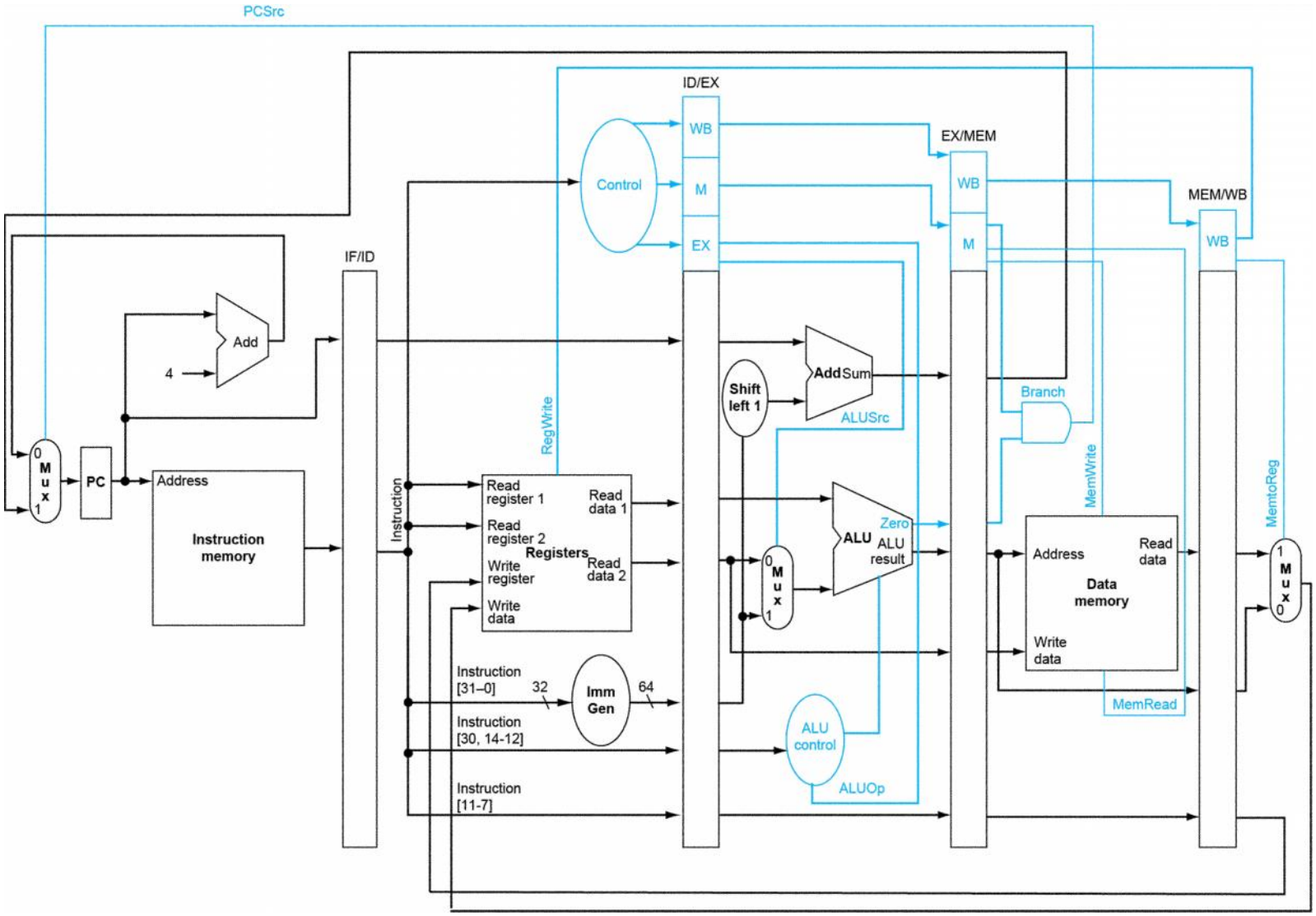
Cycle 1


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
Shift left 1
Add
Sum
ALUSrc
ALU
Zero
ALU result
ALU control
ALUOp
Branch
MemWrite
MemRead
MemtoReg
Address
Data memory
Read data
Write data


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Data memory
Read data
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp

Performance

- Execution time of execution *n* load word instructions
- Speedup
- CPI
- Size of each pipeline buffer:
  - IF/ID
  - ID/EX
  - EX/MEM
  - MEM/WB

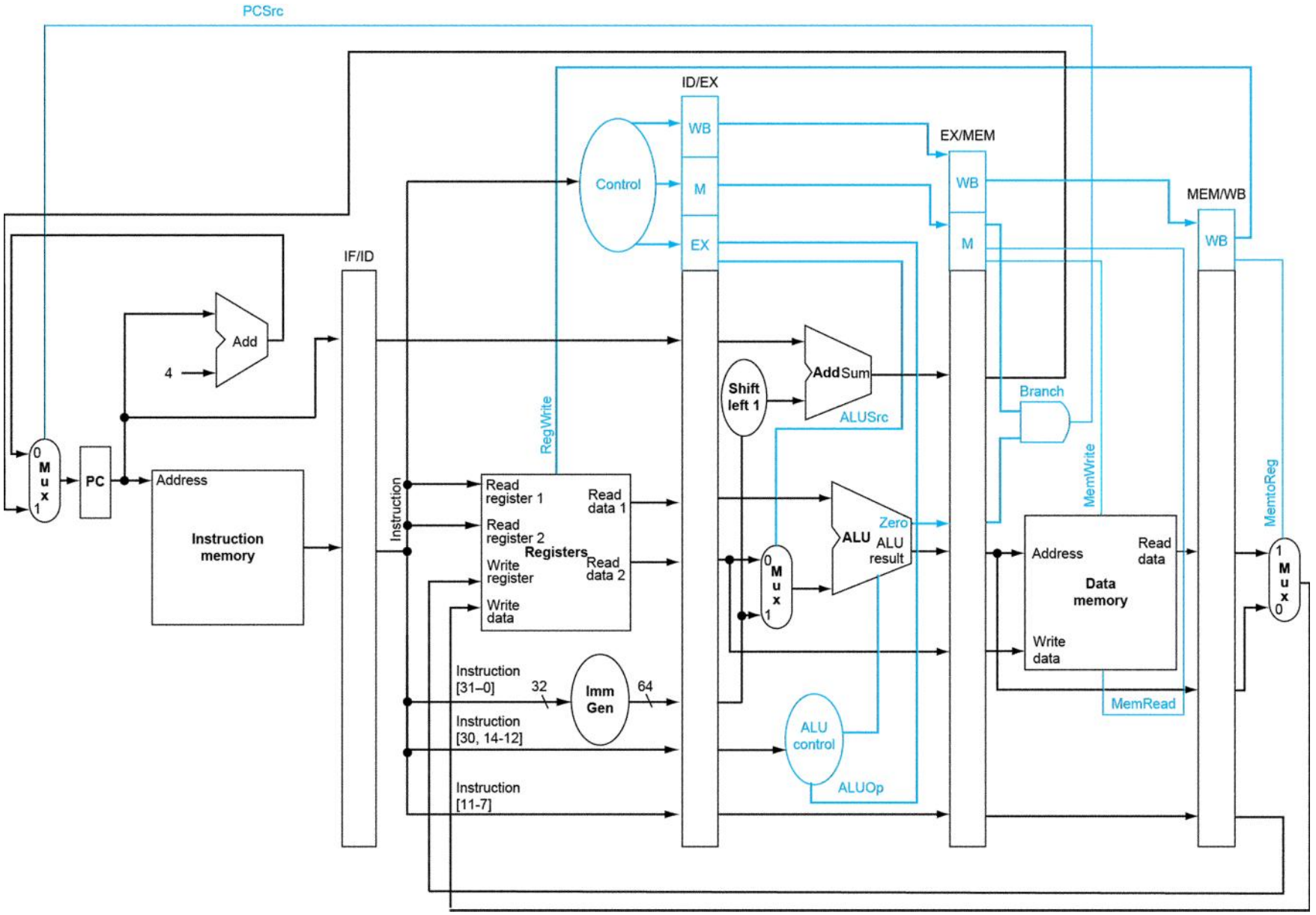
Example: How the five instructions go through the pipe, and what are the register values

- Determine the values of every field in 4 pipeline registers in cycle 5 for instructions

```
ld  x10, 24(x1)
sub x11, x2, x3
and x12, x4, x5
or  x13, x6, x7
add x14, x8, x9
```

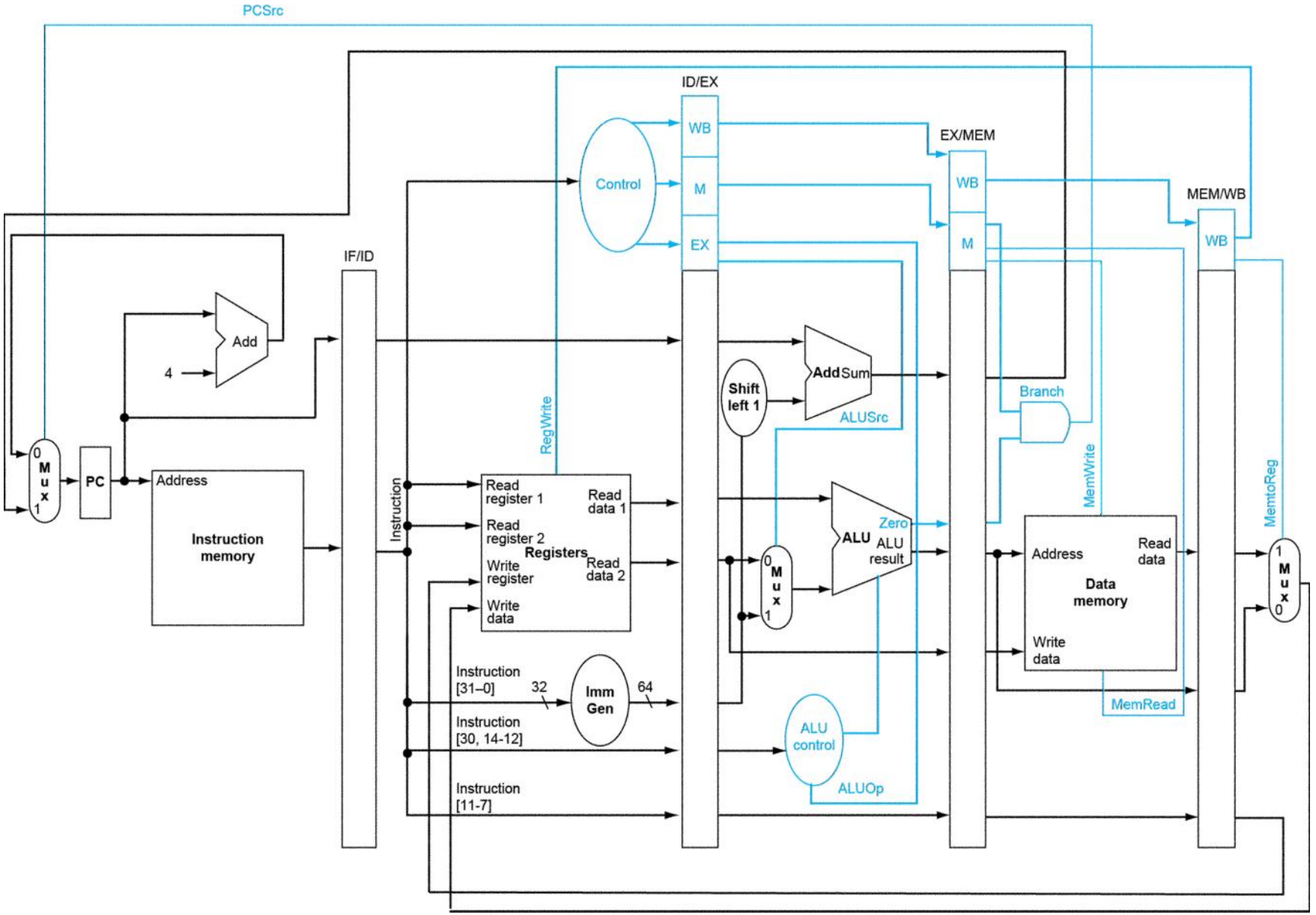
- Initial value of PC: 512
- Initial values of registers
  8+ register number*4
- Initial values of memory cells
  1000+memory address

- Assume register
  - MEM/WB contains info
    - "`ld x10, 24(x1)`"
  - EX/MEM: `sub x11, x2, x3`
  - ID/EX: `and x12, x4, x5`
  - IF/ID: `or x13, x6, x7`


PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
Control
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp

Construct the 5 instructions from a given state

- Stage 5: Regwrt MemtoReg = 11, WrtAddr=16
- Stage 4: WB=00, M=100, WrtAddr=31
- Stage 3: WB=00, M=001, EX=001,x5, x6, imm gen =16, WrtAddr=6
- Stage 2: WB=10, M=000 EX=100, rs1=2, rs2=4, inst[31:0]= $2^{22} + 2^{16} + 2^{10} + 51$


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 1
Add
Sum
ALUSrc
Branch
MemWrite
MemtoReg
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
MemRead
Instruction [31–0]
32
Imm Gen
64
Instruction [30, 14-12]
Instruction [11-7]
ALU control
ALUOp