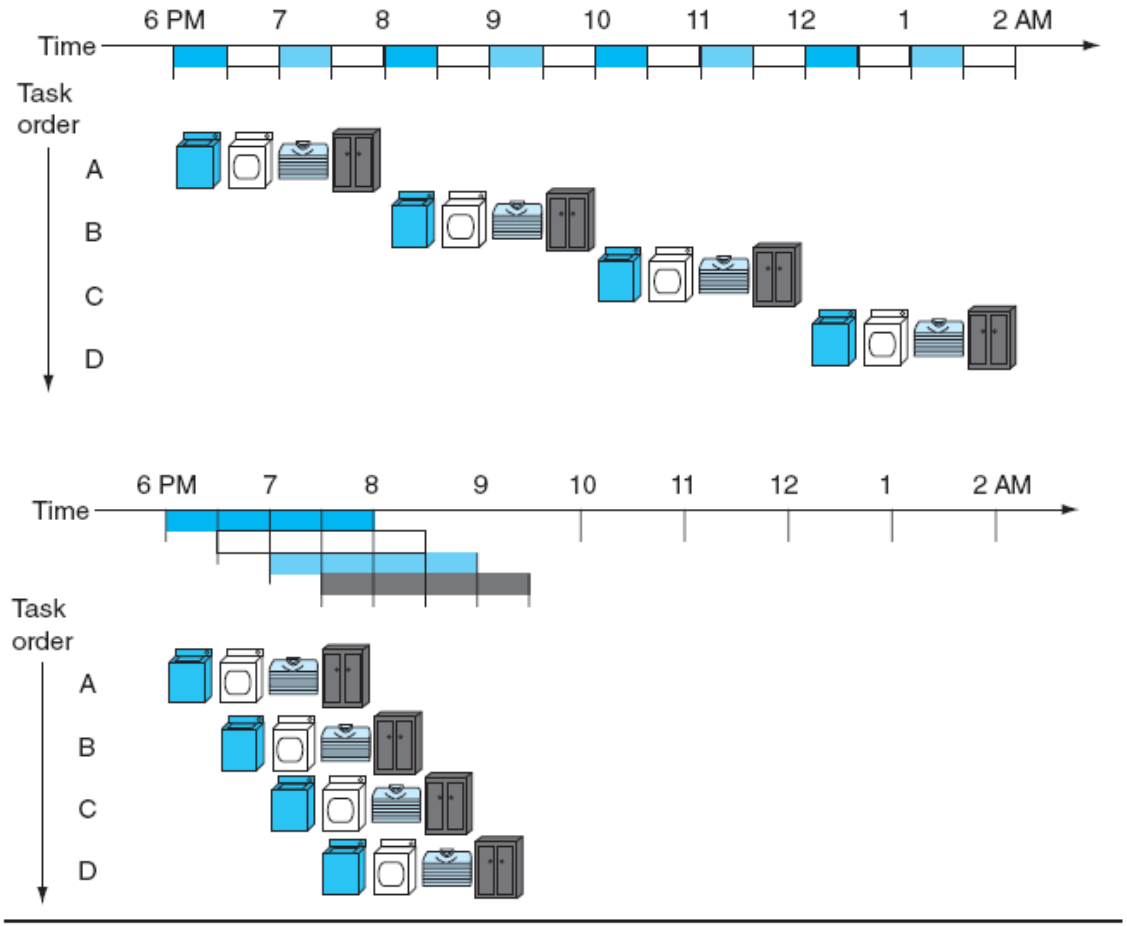
Pipelined Processor Design

EE/ECE 4305: Computer Architecture

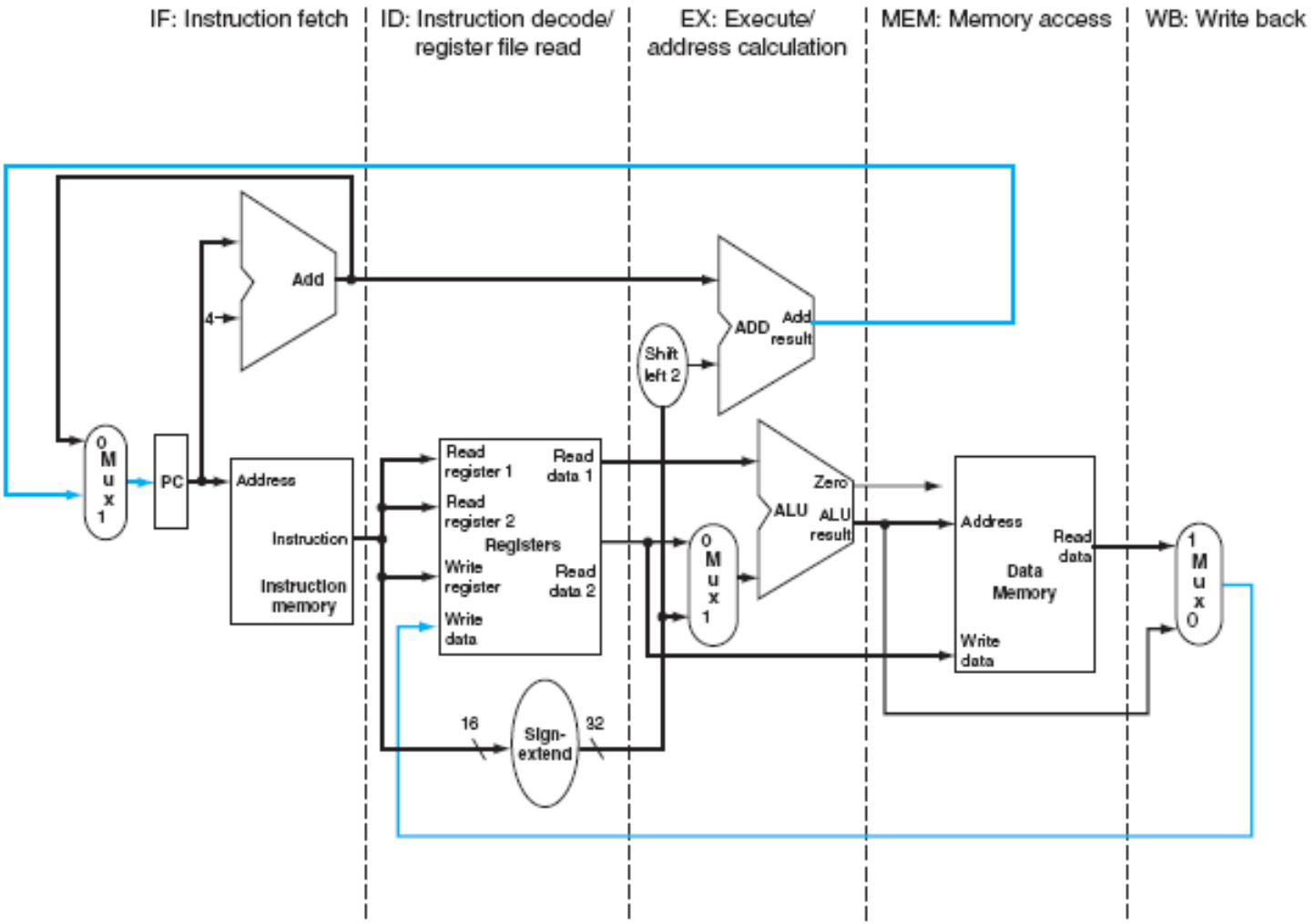
University of Minnesota Duluth

By Dr. Taek M. Kwon

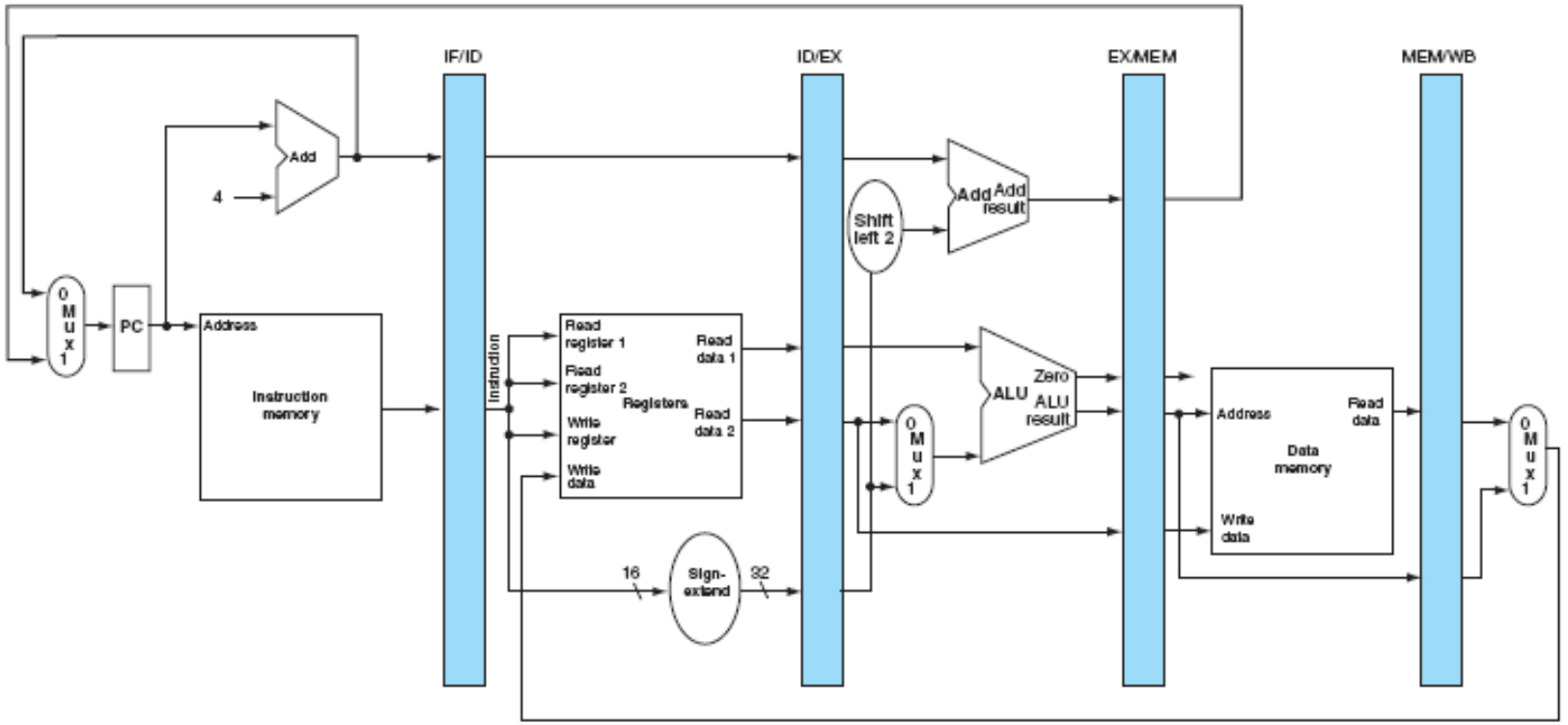
Concept


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

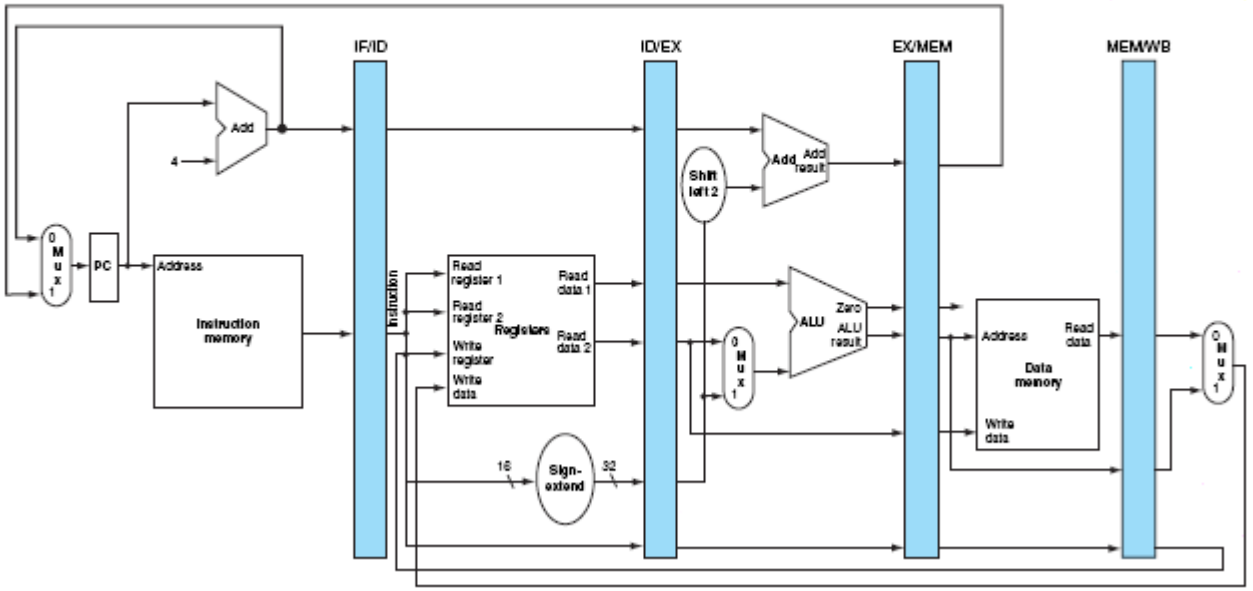
Identification of Pipeline Segments

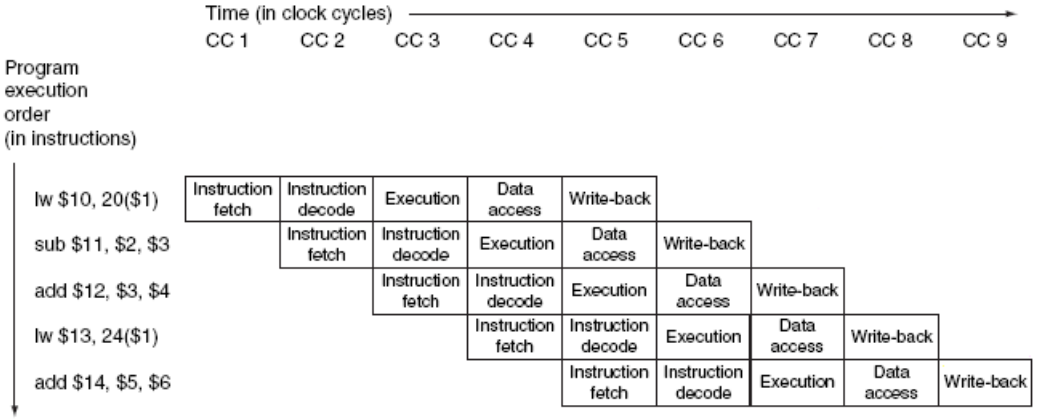


Add Pipeline Registers



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

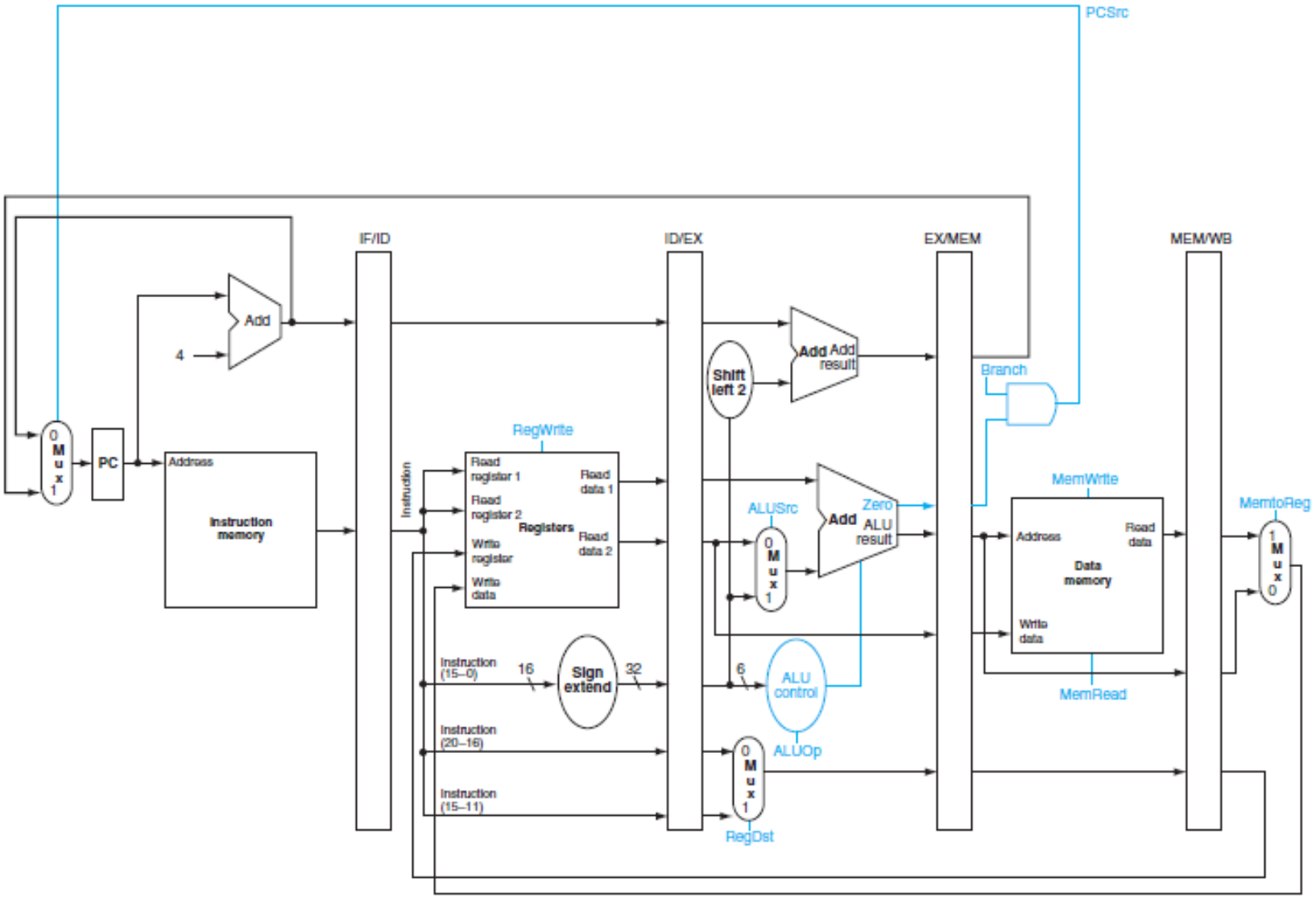




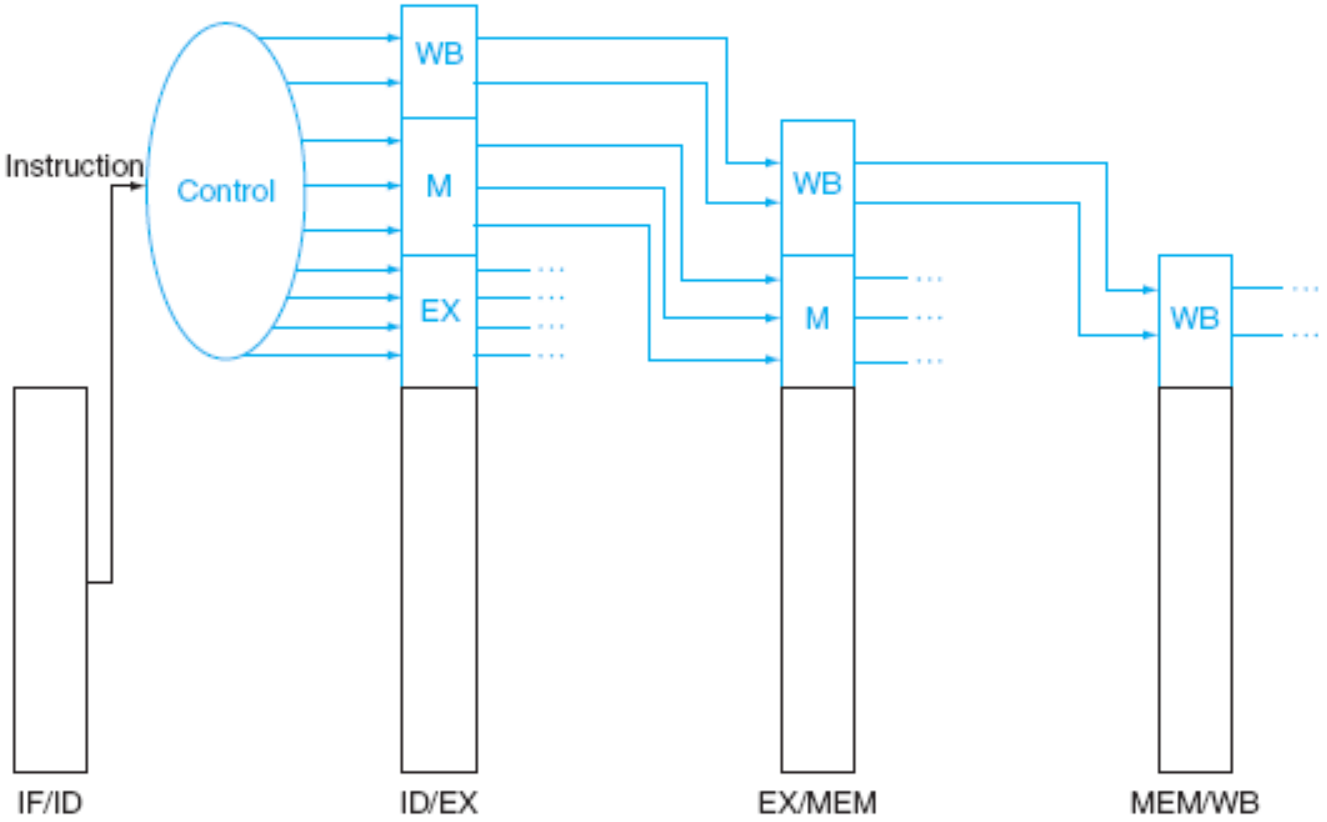
Pipeline Stage Control Signals

- **Fetch**: control signals to read IMem and PC write are always asserted. Nothing special here.
- **Decode/register-read**: no special control is needed
- **Execute**: RegDst, ALUOp, and ALUSrc
- **Mem Access**: Branch, MemRead, MemWrite
- **Write-Back**: MemtoReg and RegWrite

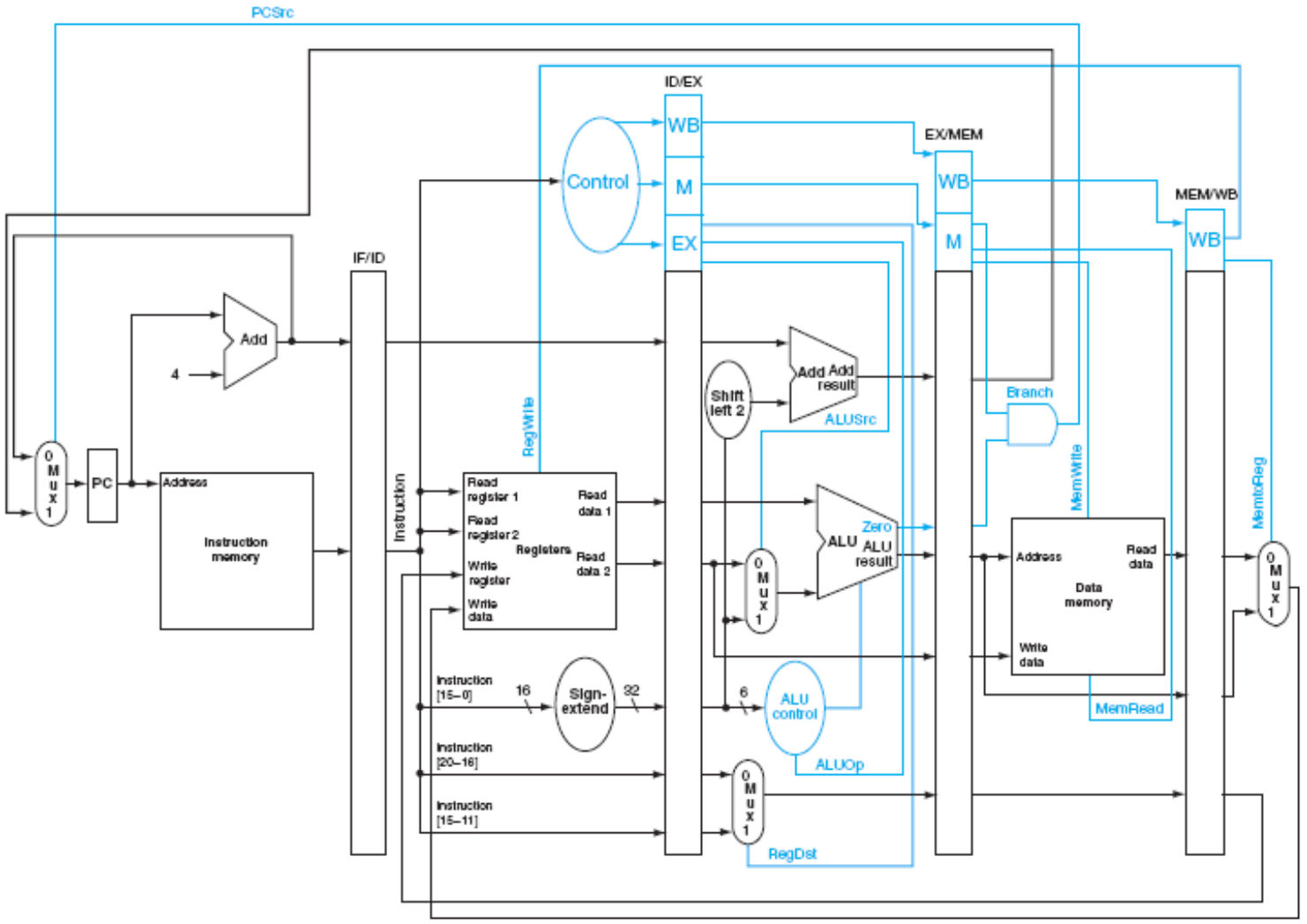
Control Signals Identified



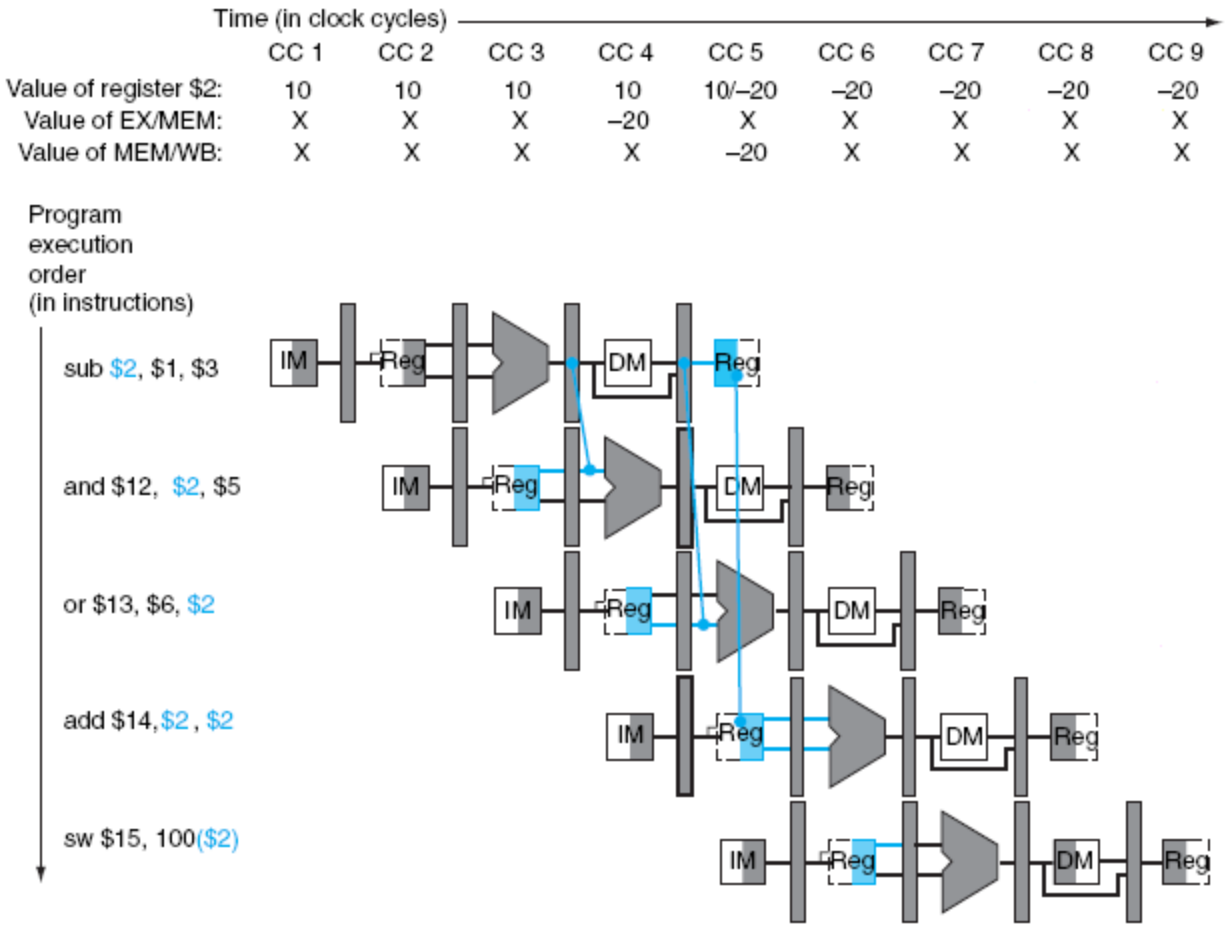
Control for the Pipeline Stages



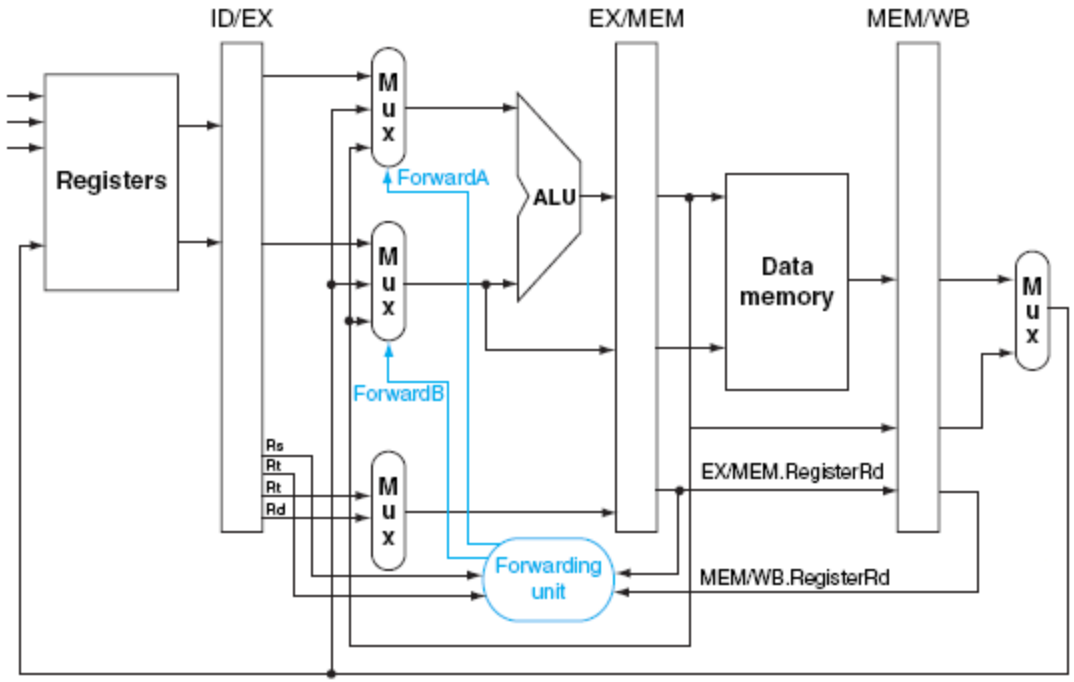
Control Portion of Pipeline Registers



Dependence and Data Forwarding

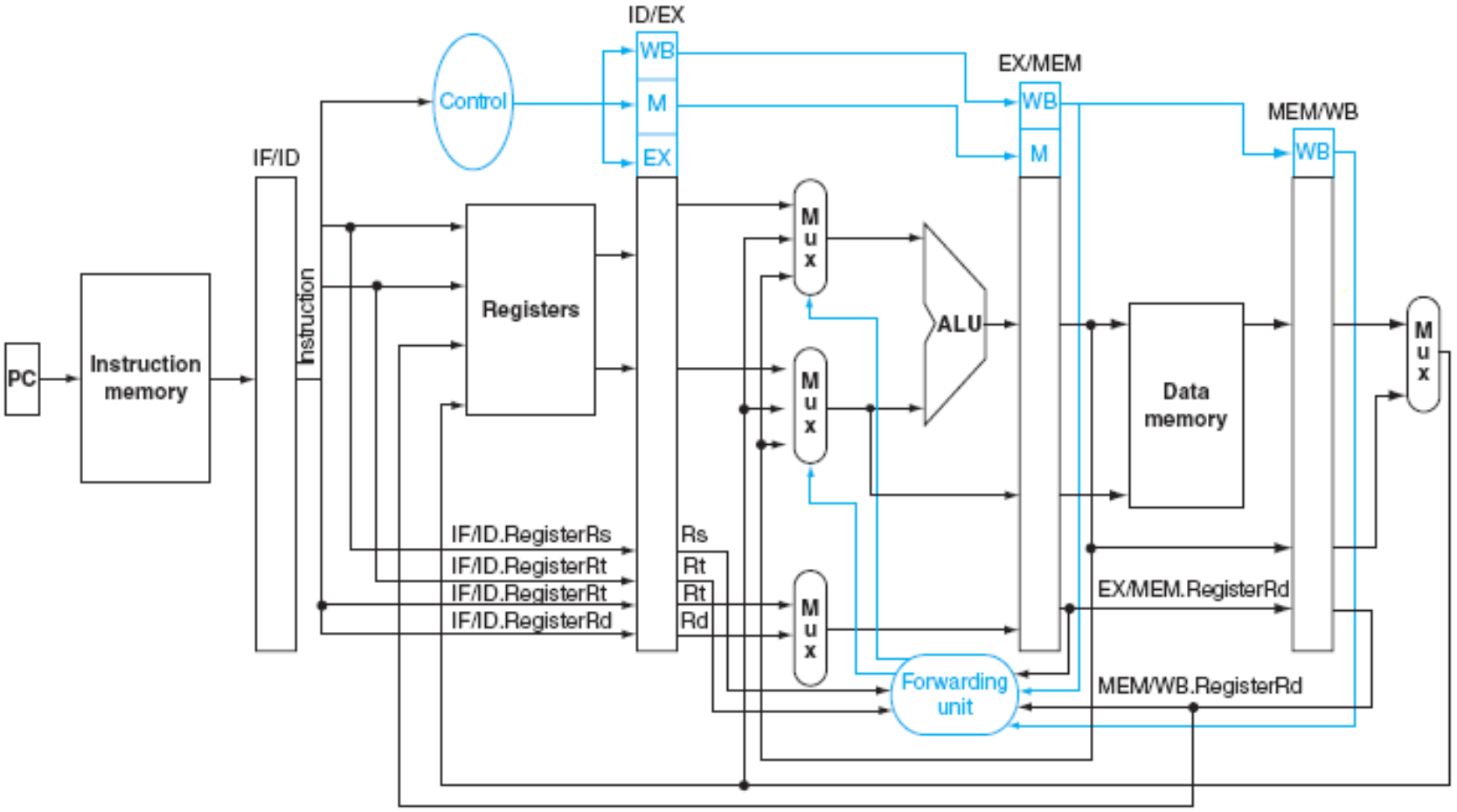


Adding Data Forwarding

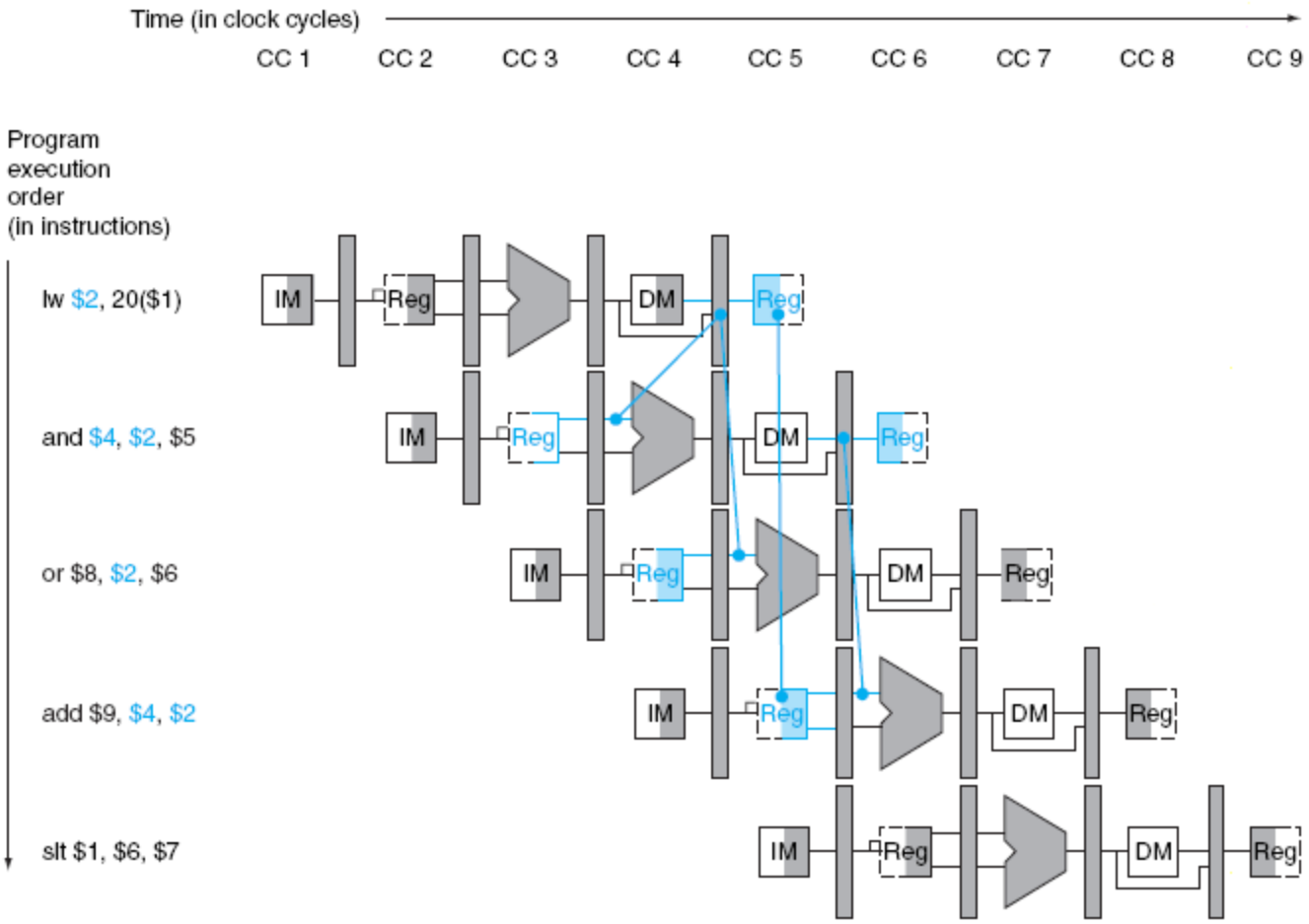


b. With forwarding

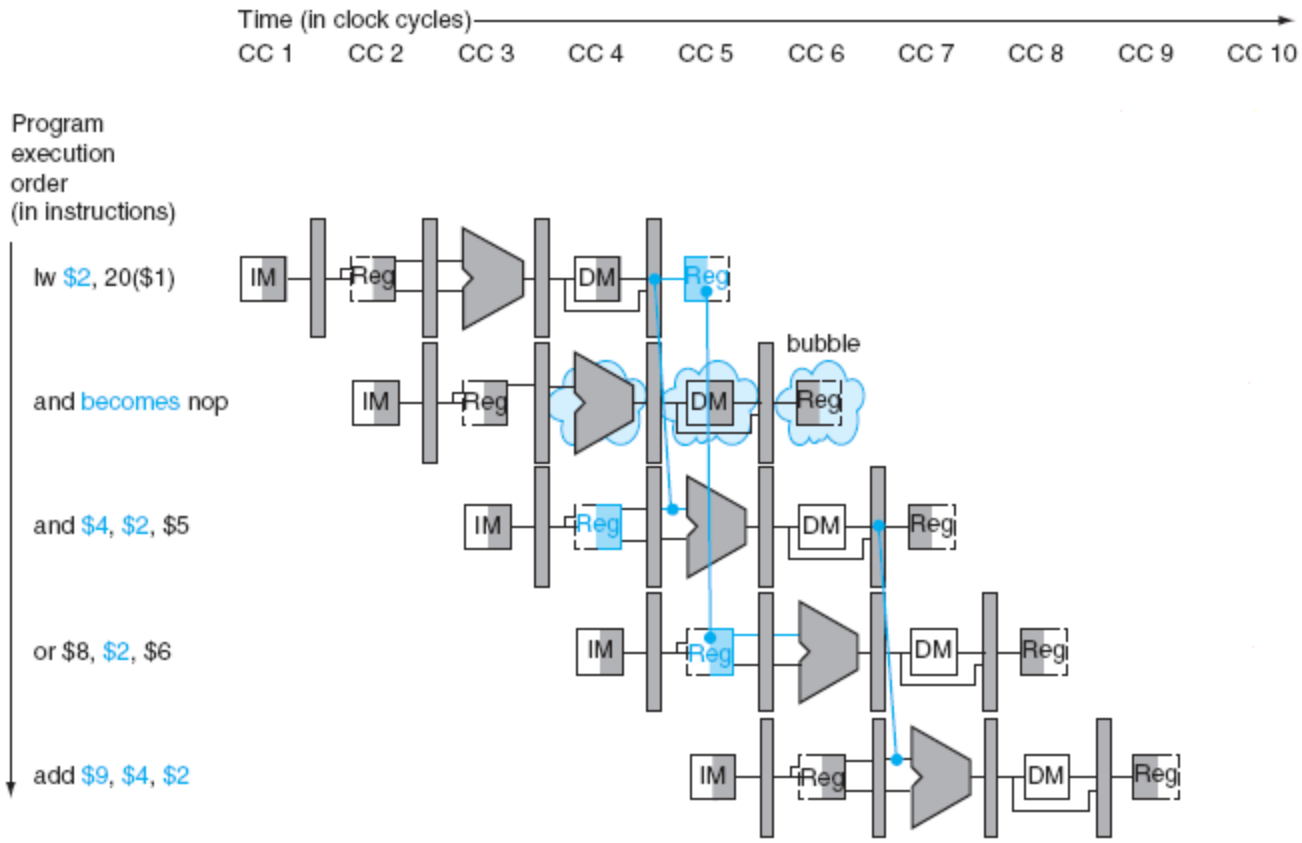
Data Forwarding Control



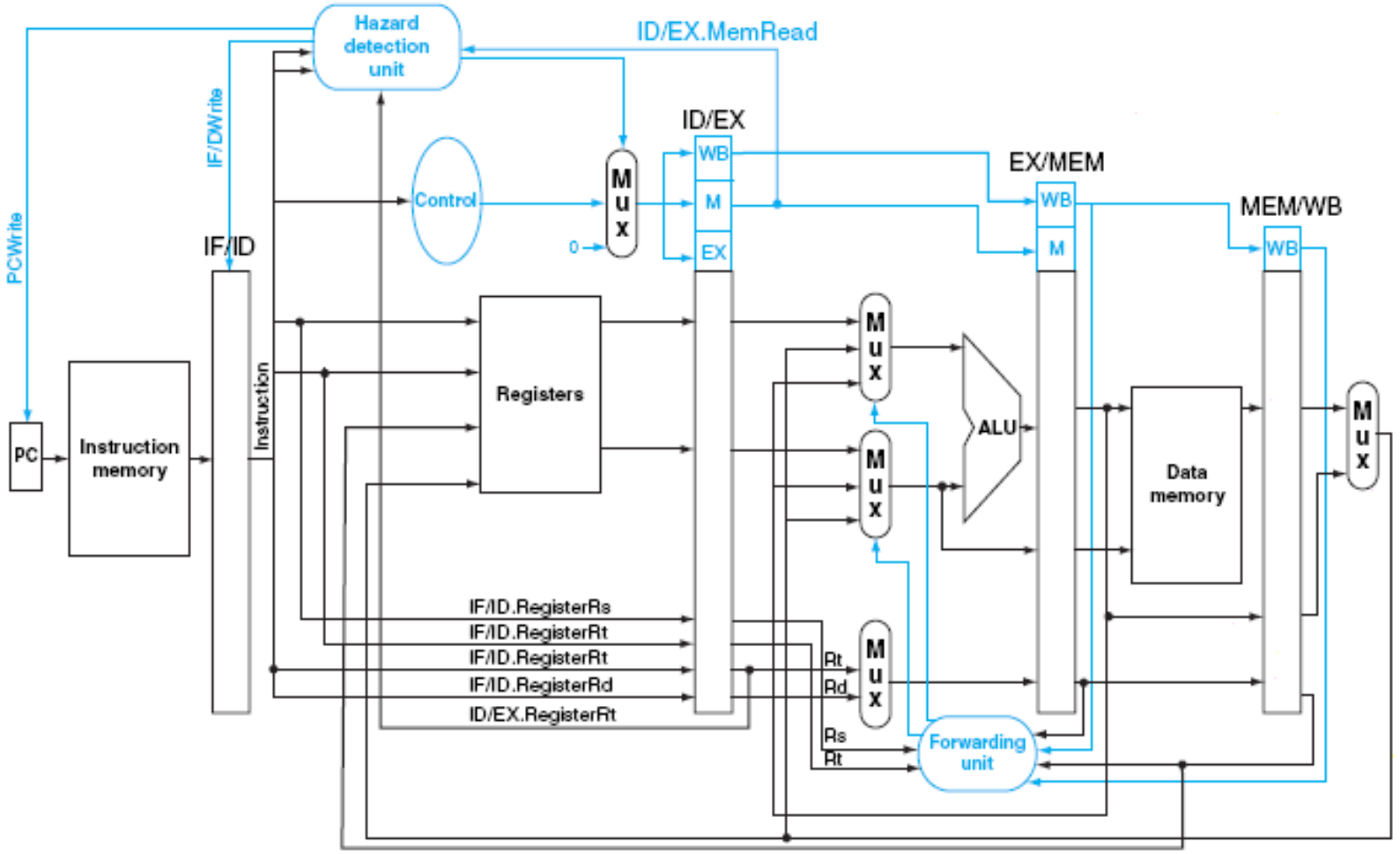
Data Hazards and Stall



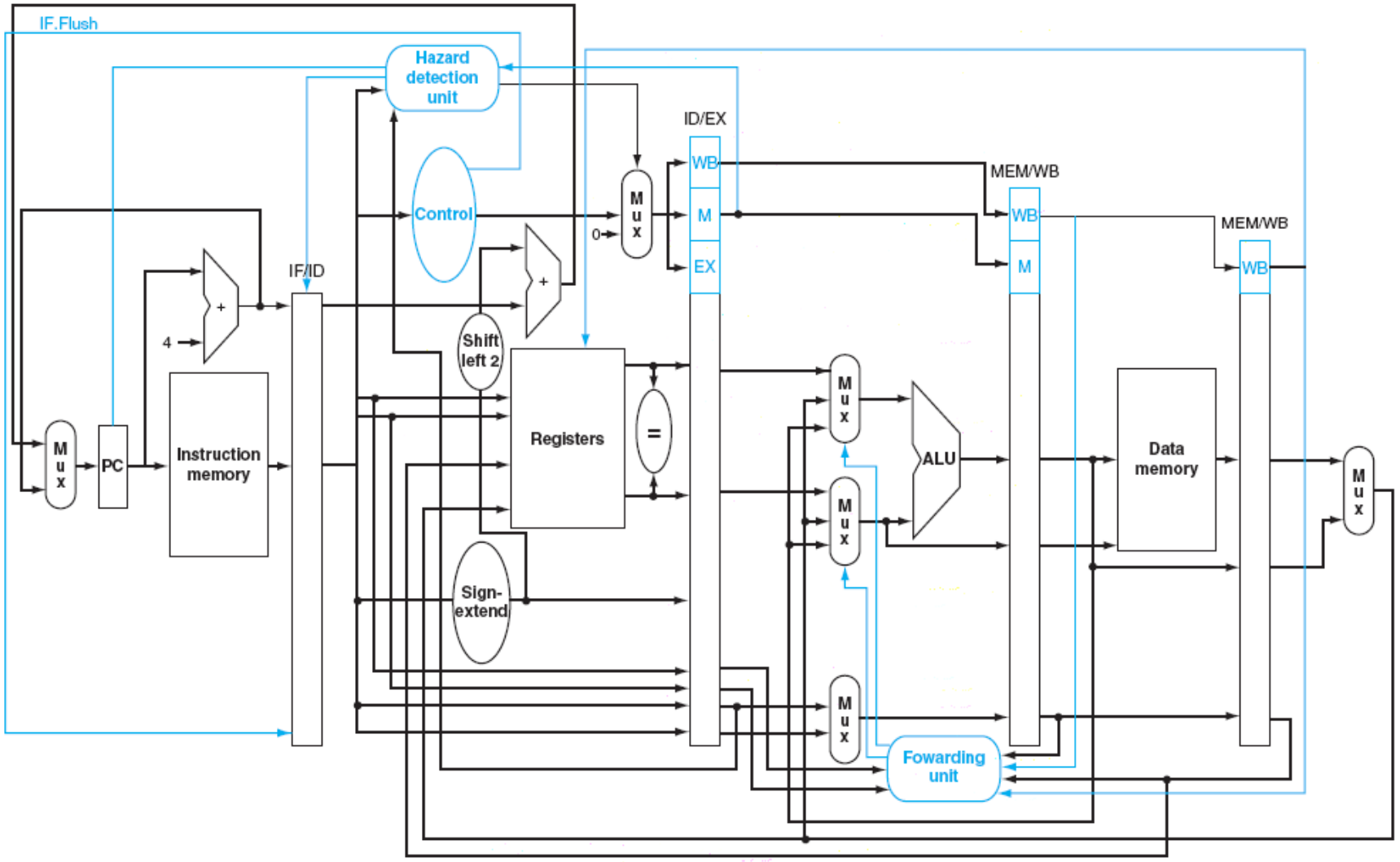
Stall: Insertion of nop



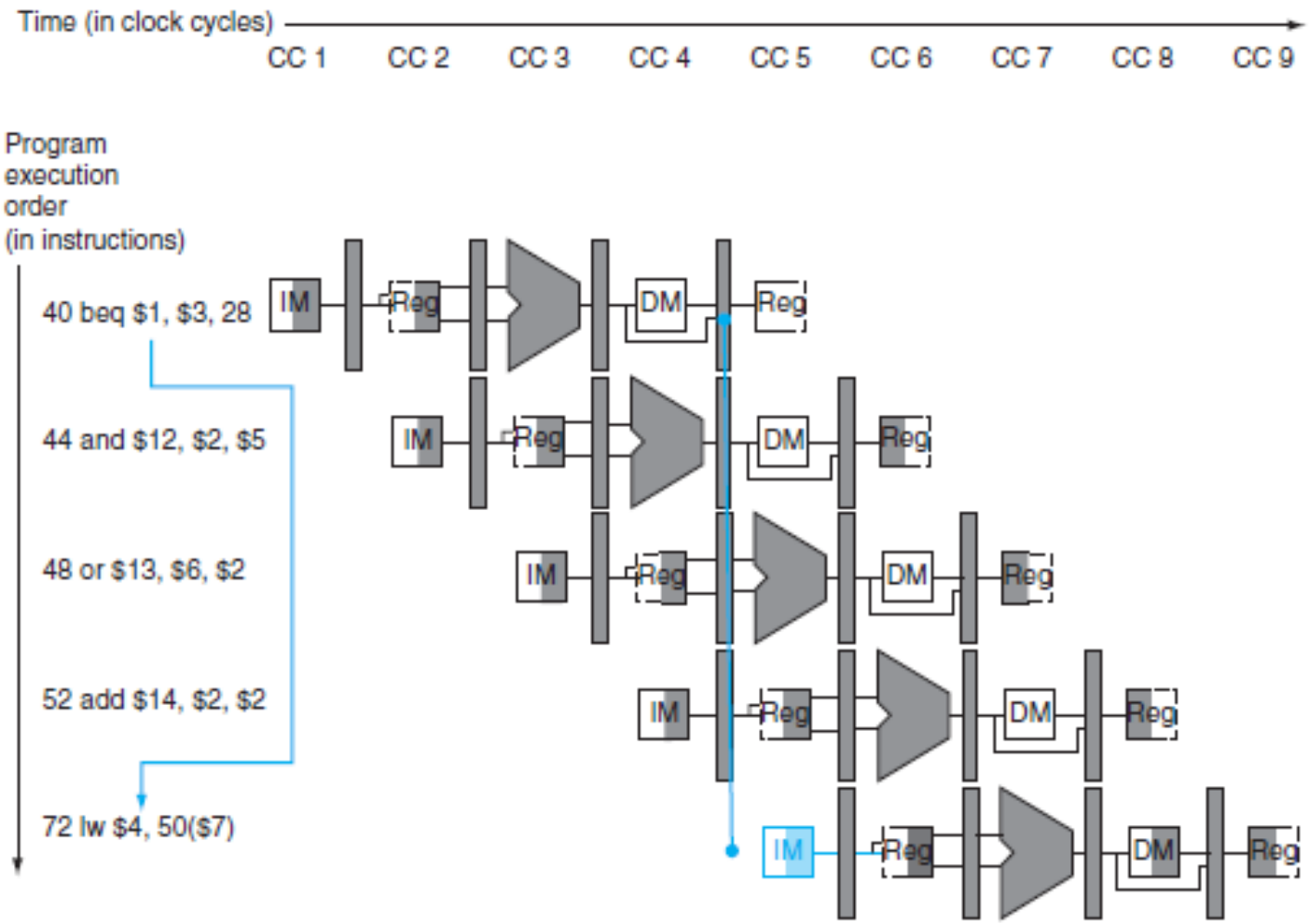
Controls for Forwarding and Hazard Detection



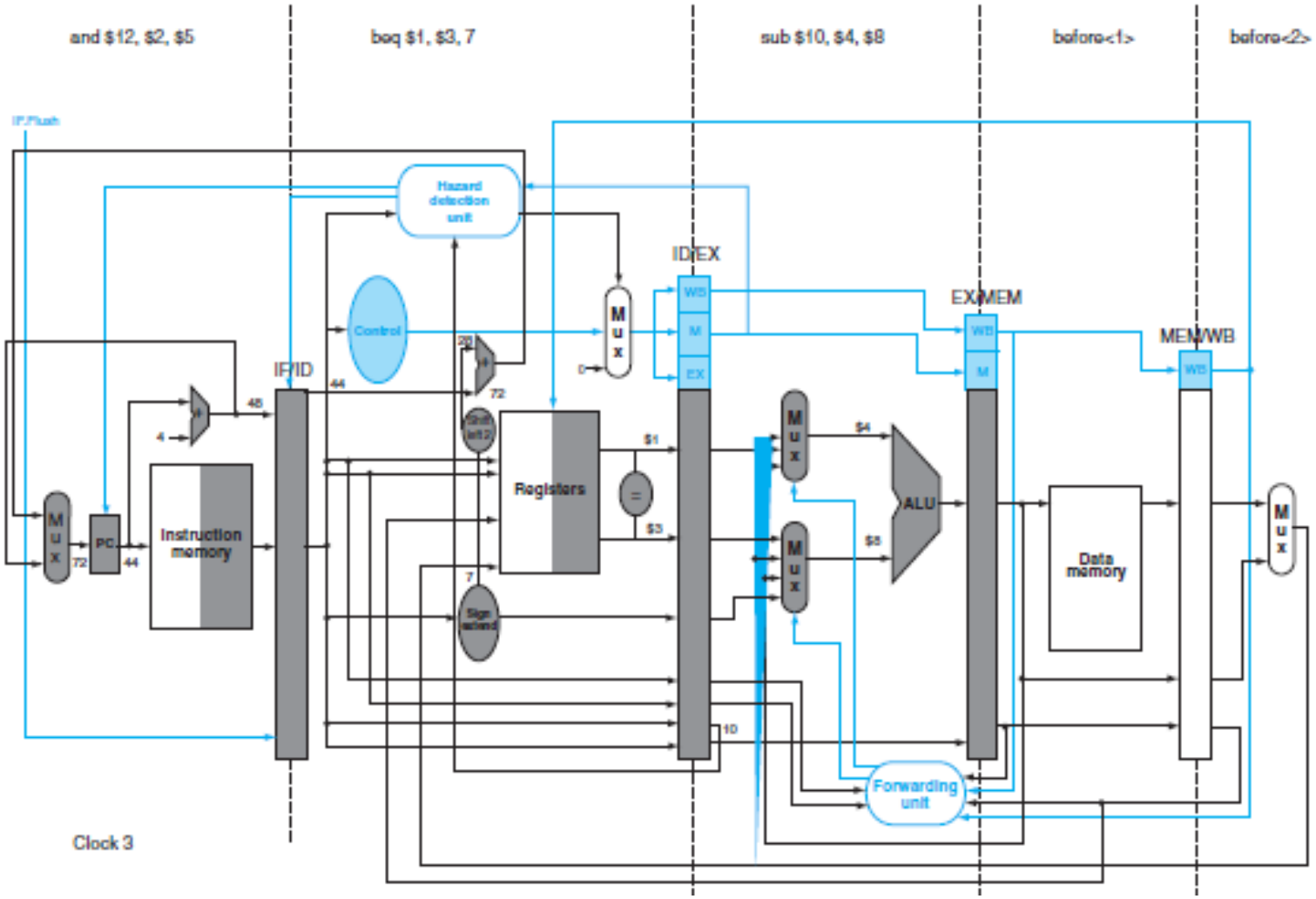
Control Hazard: IF.Flush for dealing with the delays caused by branch



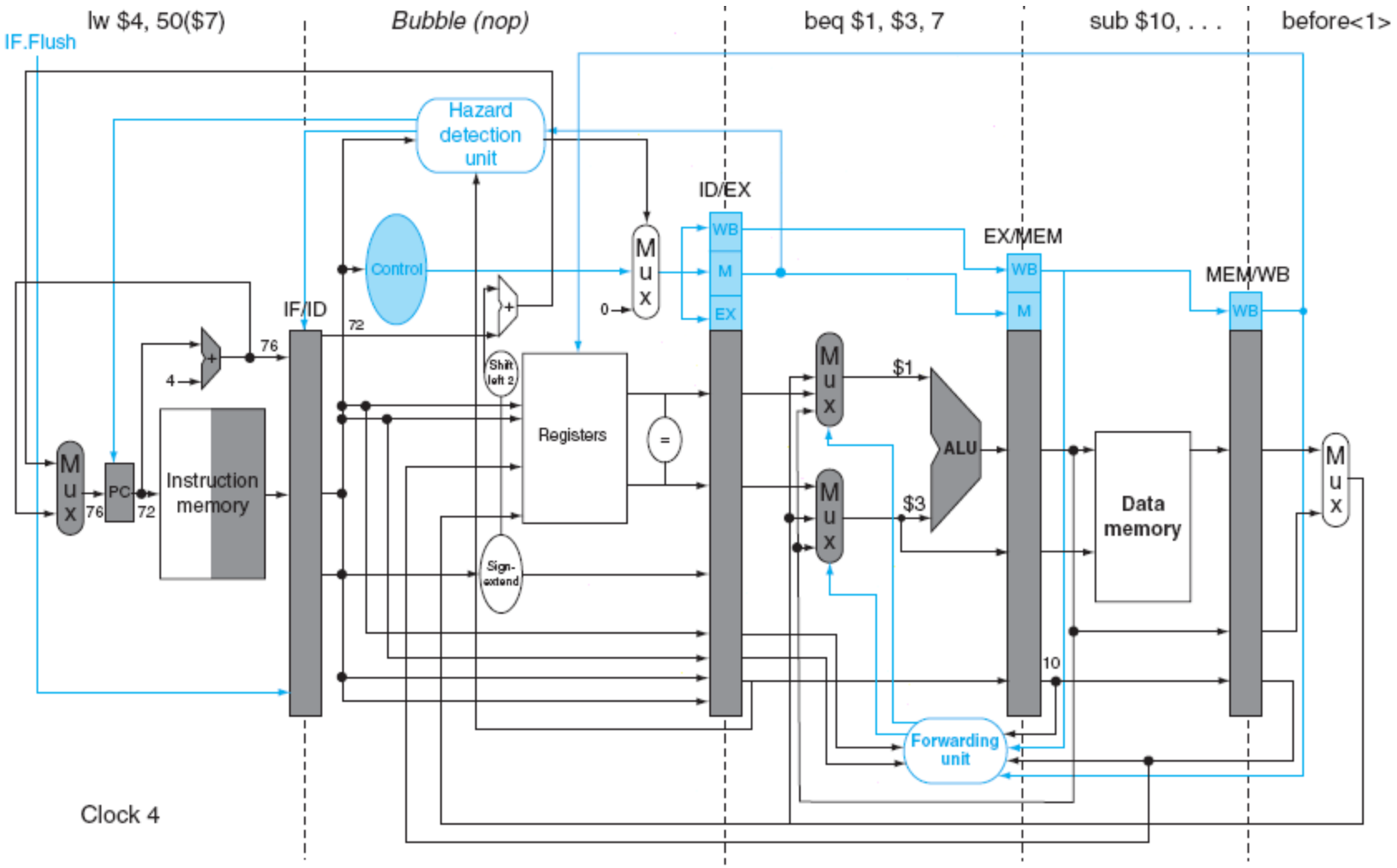
Impact of Branch Instruction



Branch decision on decode stage



Insertion of nop when branch taken



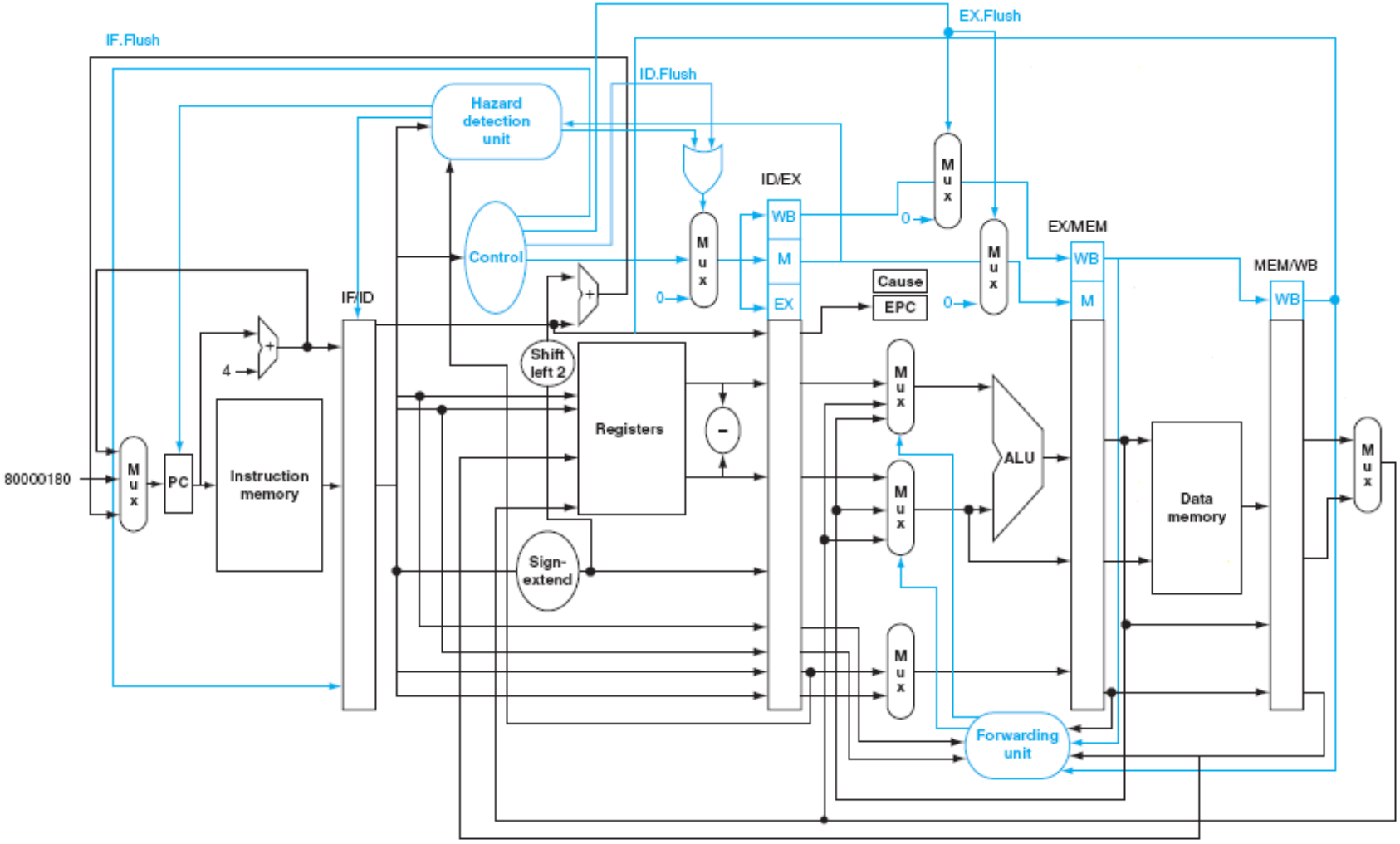
Exception in Pipeline

- The instruction that follows offending instruction must be flushed.
- Fetch must begin from the new address that implements the exception service routine.

Pipeline implementation of Exception

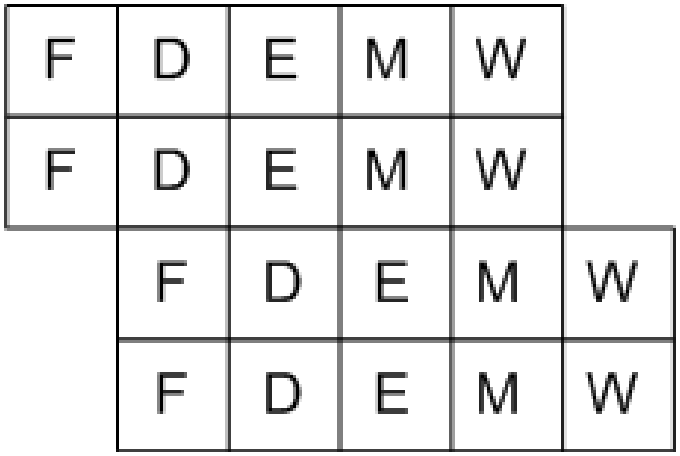
- Flushed instruction is replaced with nop register control signals
- A new control signal, called ID.flush, is ORed with the stall signal from the hazard detection unit
- To flush the execution stage, a new signal called EX.flush is used to zero the control lines in the pipeline buffer
- 0x80000180 is multiplexed to PC, which is the address for exception processing
- The address of the offending instruction is saved in the EPC and the cause in the Cause register

Controls that Include Exception



Extracting More Performance from Pipelining

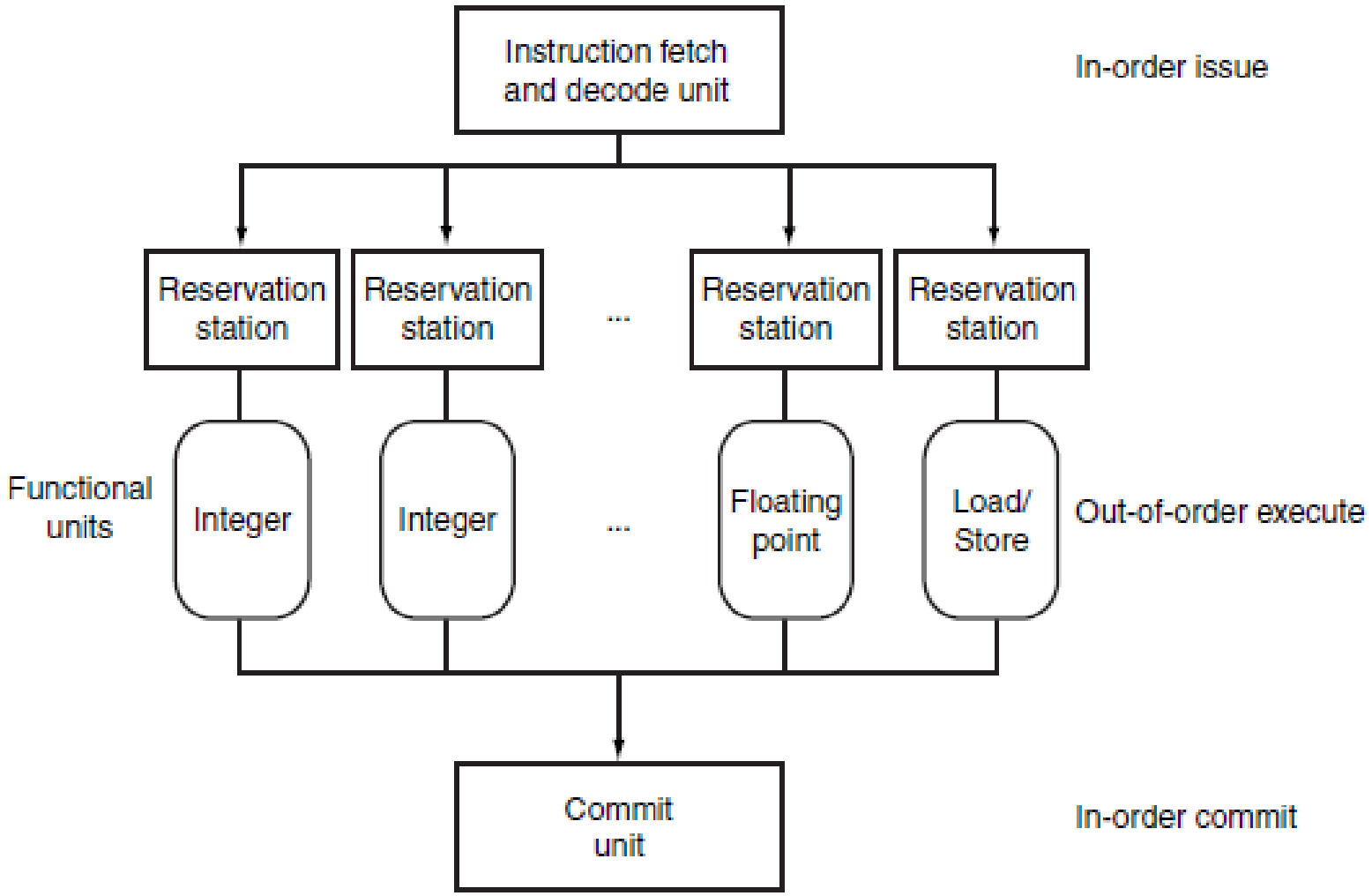
- Increase the depth of the pipeline
- Multiple issue pipelining (2-issue pipeline)



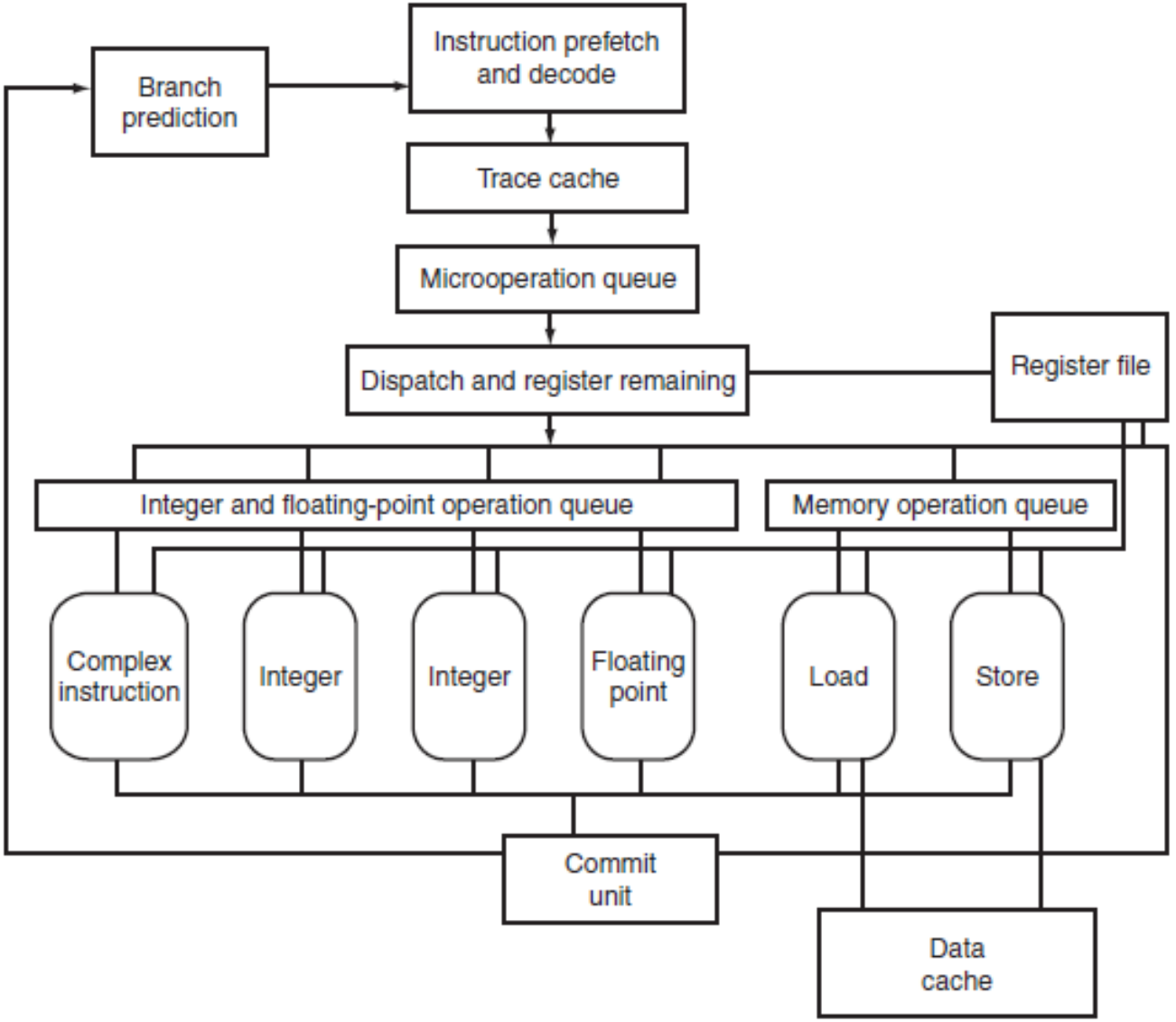
Dynamic Pipeline Scheduling

- Each Reservation Station (**RS**, buffer) holds the operand and the operation that corresponds the attached Functional Unit (**FU**)
- Instructions are fetched, decoded, and then distributed to RS.
- Each FU computes and produces the result as soon as the operands are ready.
- The results are either forwarded to the RS waiting for the result or to the Commit Unit (**CU**).
- The CU (reorder buffer) holds results until they are safe to put them to the register file or to store them into memory. It also directly supplies operands FUs.

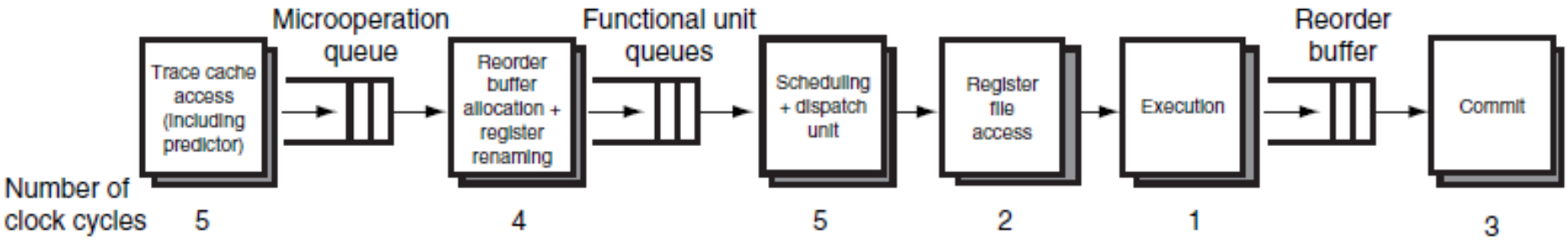
Dynamic Pipeline Scheduling



Microarchtecture of the Intel Pentium-4



Pentium-4 Pipeline



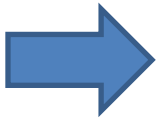
Register Renaming

- To effectively use the reorder buffer and the reservation stations, registers are often renamed.
- The instruction in a RS is buffered until all the operands are execution units are available. When the FU eventually produces the result, it is directly copied into the waiting RS bypassing the register. The feed registers are renamed and let them available for other instructions.

Register renaming illustration

Compiler Level

1. R1 <= Mem[10]
2. R1 <= R1 + 2
3. Mem[10] <= R1
4. R1 <= Mem[20]
5. R1 <= R1 + 4
6. Mem[20] <= R1



Renamed

1. R1 <= Mem[10]
2. R1 <= R1 + 2
3. Mem[10] <= R1

1. R2 <= Mem[20]
2. R2 <= R2 + 4
3. Mem[20] <= R2

Pentium 4 Pipeline

- IA-32 instructions are translated to micro-ops
- Micro-ops are executed by a dynamically scheduled speculative pipeline
- multiple functional units =7
- Deep pipeline (20 stages)
- Use of trace cache (holds predecoded sequence of micro-ops)
- 128 registers
- Upto 126 micro-ops can be queues at any point in time
- Extensive bypass network among functional units

IA-64 Architecture (1)

- RISC style register-to-register instruction set
- 128 integer and 128 floating point registers; 8 registers for branch
- Supports register windows (similar to SPARC)
- Instructions are encoded in bundles, which 128 bits wide
- Each bundle consists of 5-bit template field and three instructions (41 bits each)
- The template field specifies which of the five different execution units each instruction in the bundle requires

IA-64 Architecture (2)

- Five different types of execution units: (1) integer ALU, (2) non-integer ALU (shifters and multimedia operations), (3) memory unit, (4) FP unit, and (5) branch unit.
- The bundle formats can specify only a subset of all possible combinations of instruction types
- Supports predication (A technique that is used to eliminate branches by making the execution of an instruction dependent on a predicate, rather than dependent on a branch)

Reality Check

- Many dependencies cannot be alleviated
- Compilers or hardware often do not exactly know data dependencies; this leads to a conservative design
- Pointers are often assumed as potential dependencies
- Ability to predict branches is limited
- Memory systems are not able to keep the pipeline full
- Cache misses stall the pipeline

Fallacies and Pitfalls

1. Pipelining is easy (true or false?)
2. Pipelining ideas can be implemented independent of technology (true or false?)
3. Failure to consider instruction set design can adversely impact pipelining (true or false?)
4. Longer pipelines, multiple instruction issues and dynamically scheduled pipeline in 1990s helped sustain the single process performance increase (true or false?)