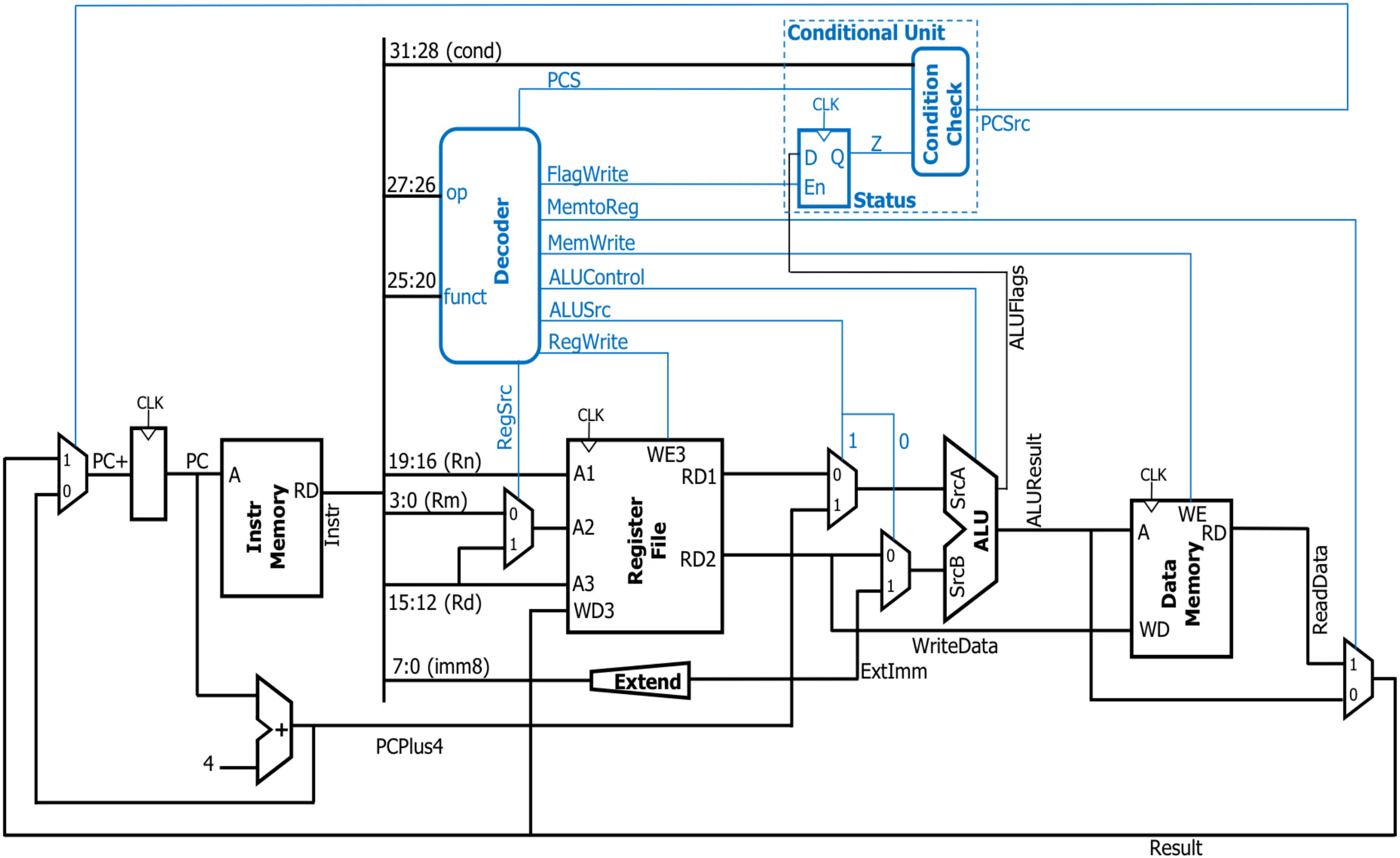
CG2028 COMPUTER ORGANIZATION

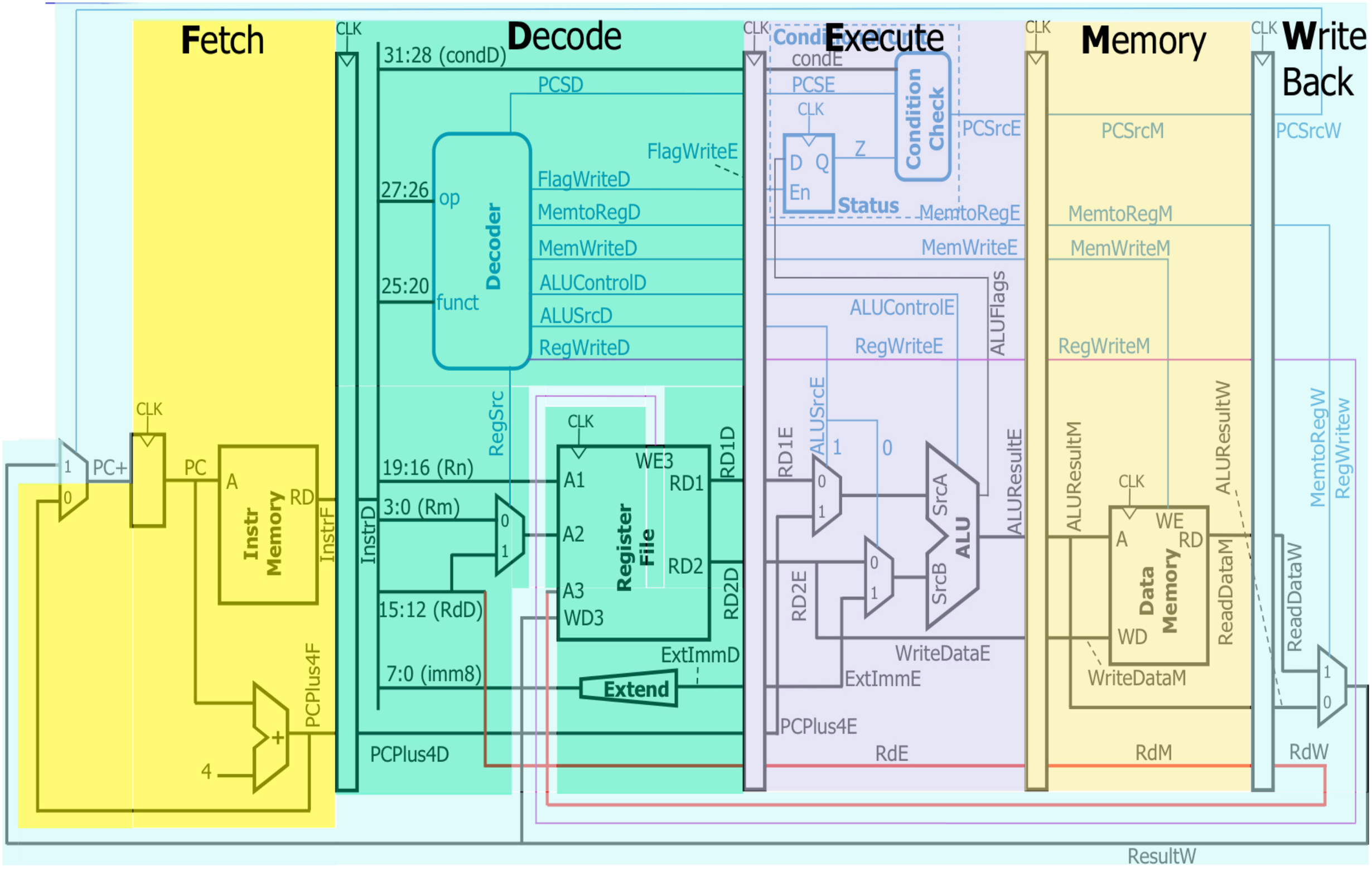
TUTORIAL 6: PIPELINING BASICS

GU JING

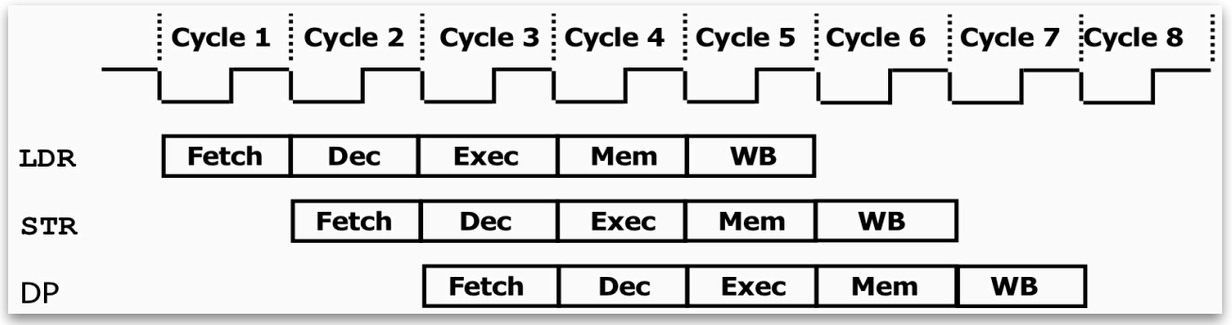
Single-Cycle Processor

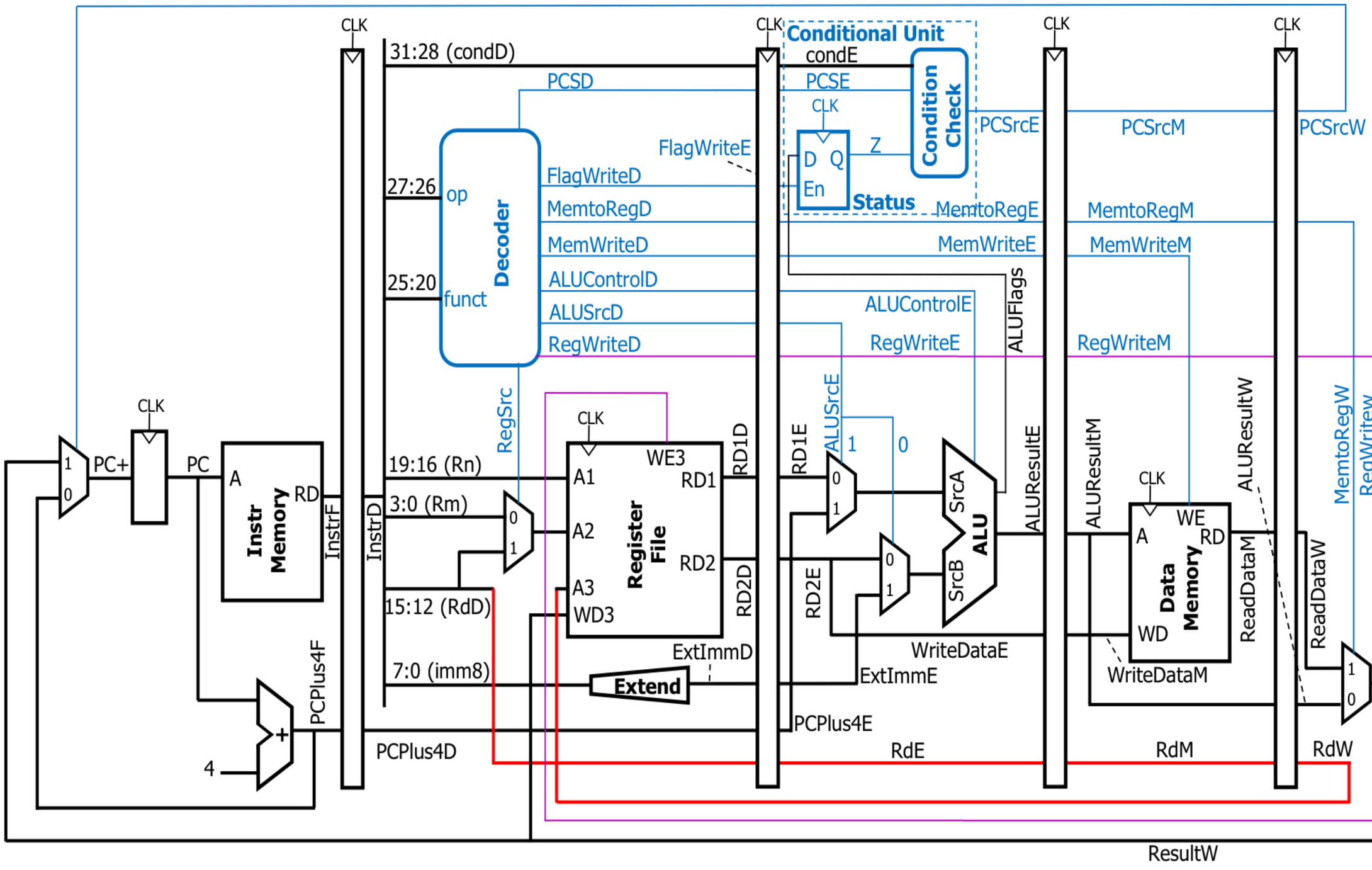


Pipeline Stages



Pipelined Processor





Pipeline Hazards

Structural Hazards: **attempt to use the same resource by two different instructions at the same time**

E.g. Read instruction and data from memory within the same cycle

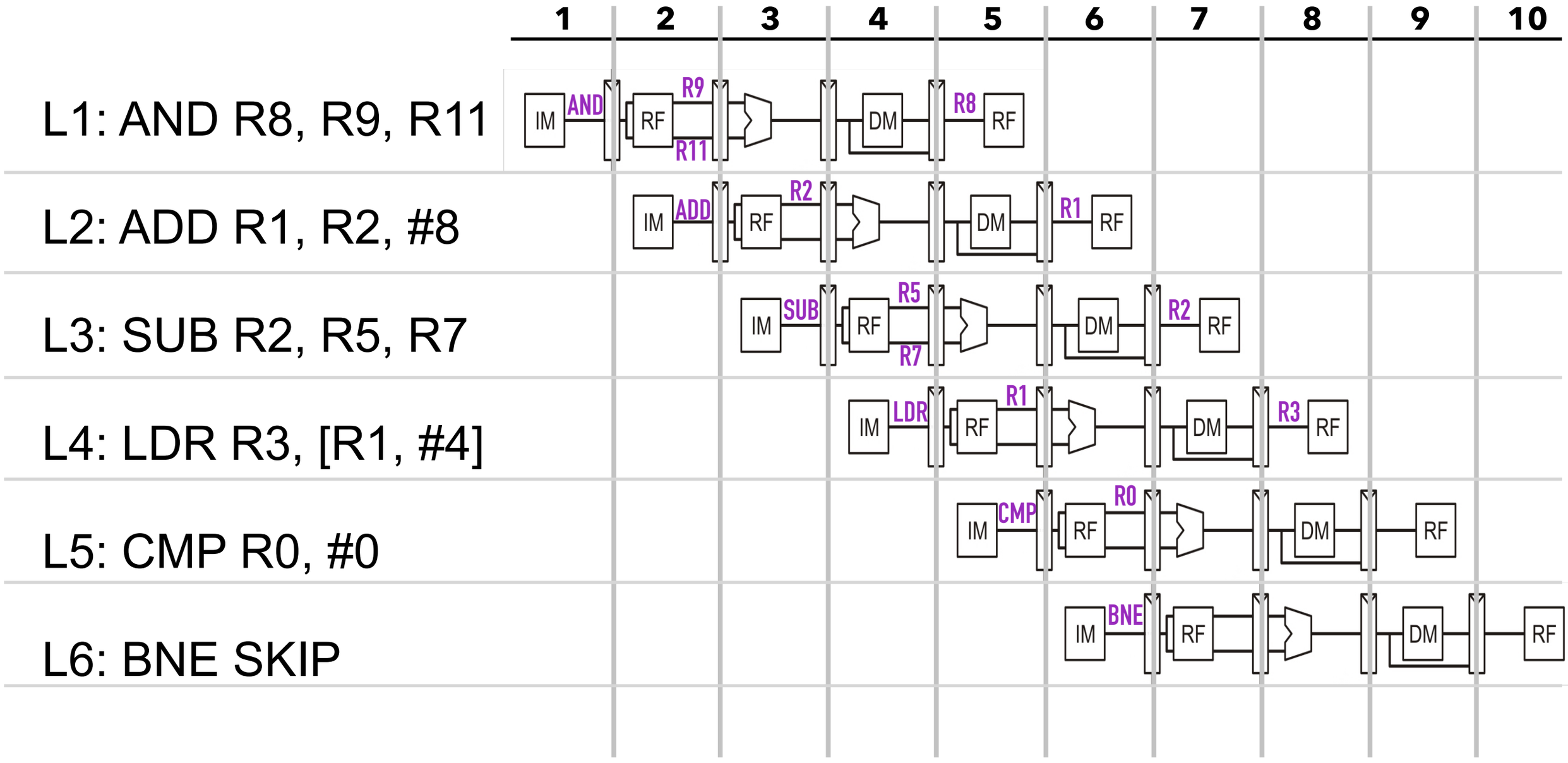
Data Hazards: **attempt to use data before it is ready.**

E.g. an instruction source operand(s) are produced by a prior instruction still in the pipeline

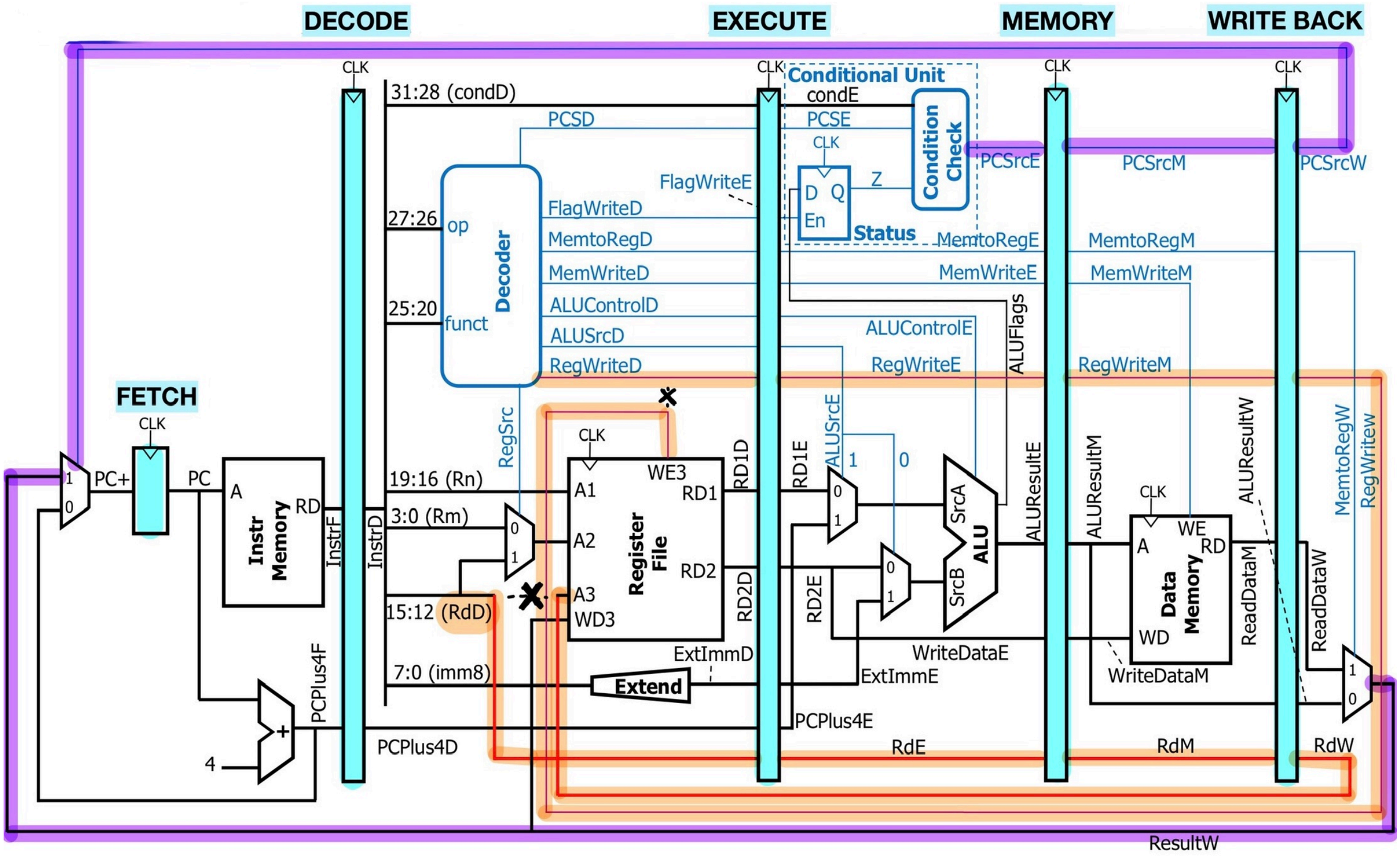
Control Hazards: **attempt to make a decision about program control flow before the condition has been evaluated and the new PC target address calculated.**

E.g. branch instructions, writes to R15, interrupts/exceptions

Q1. Insert NOPs in the code below to ensure correct operation in a 5-stage pipelined ARM processor.

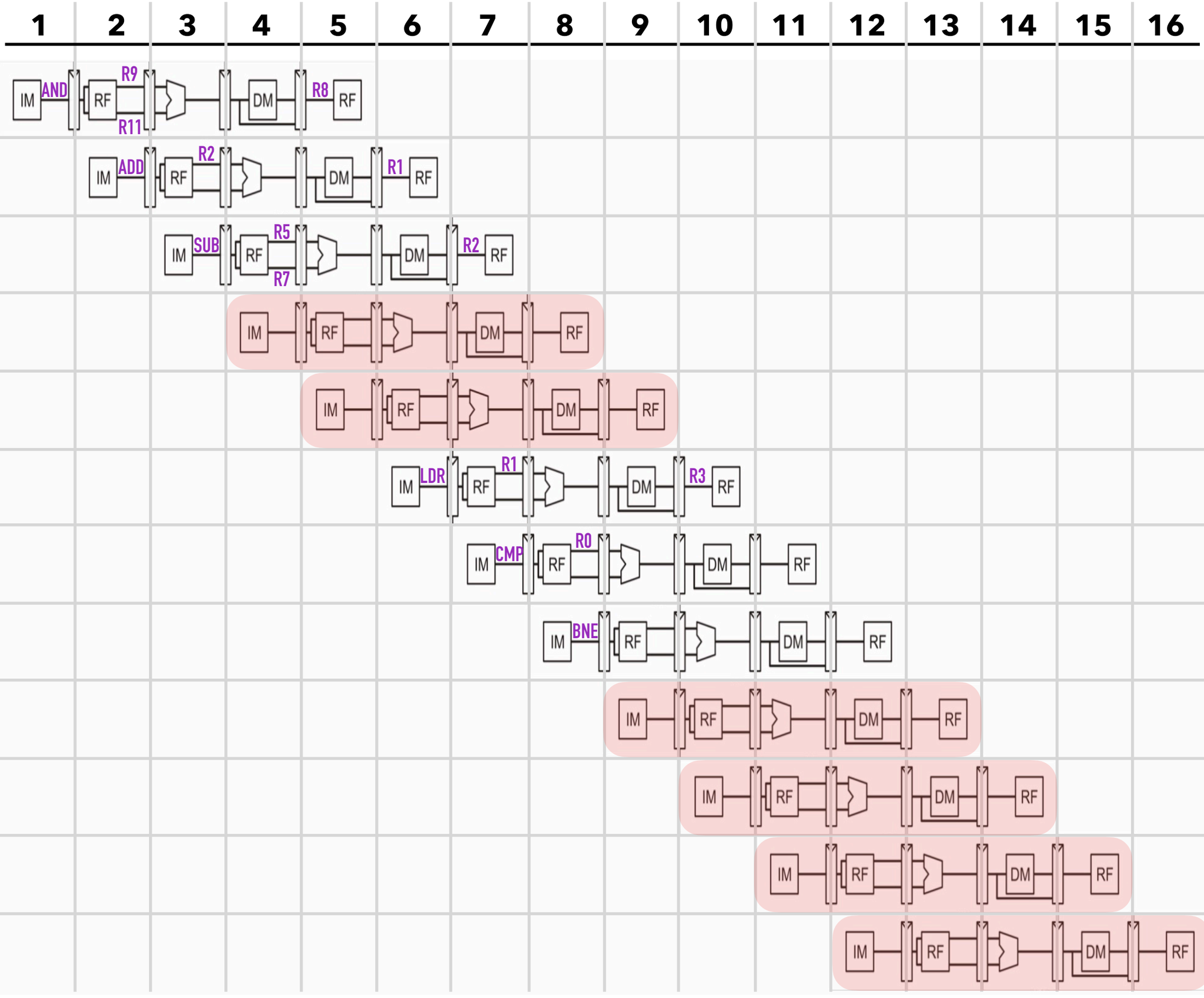


Pipelined Processor: Data Hazards & Control Hazards



CG2028 Tutorial 6: Pipelining Basics

Q1.



Q1. Insert NOPs in the code below to ensure correct operation in a 5-stage pipelined ARM processor.

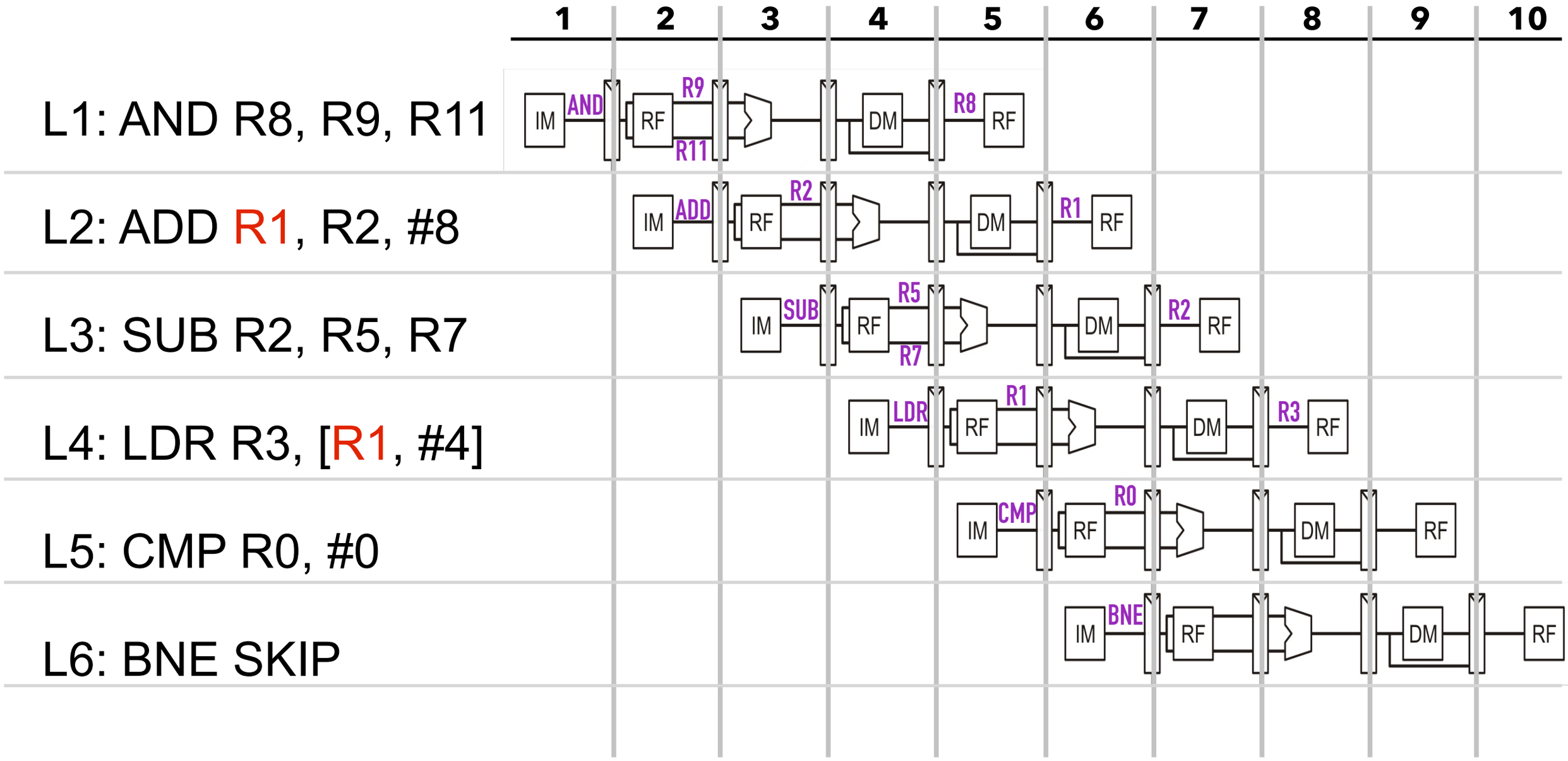
```
AND R8, R9, R11
ADD R1, R2, #8
SUB R2, R5, R7
NOP
NOP
LDR R3, [R1, #4]
CMP R0, #0
BNE SKIP
NOP
NOP
NOP
NOP
NOP
```

Ans:

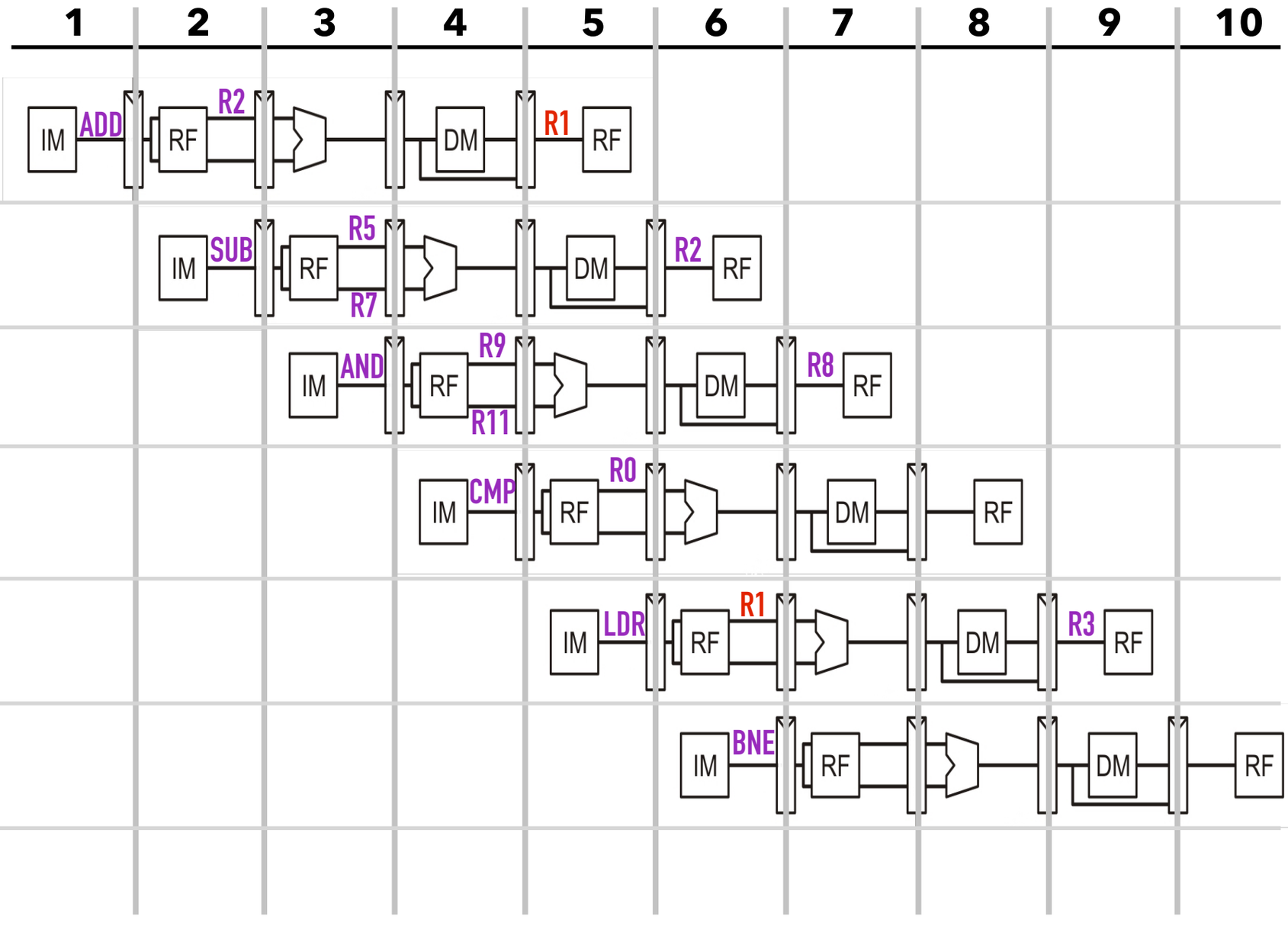
There is a data hazard when the source register for an instruction is written by (i.e., is the destination register of) an instruction still in the pipeline. If there is a data hazard, to ensure correct operation, we need to insert an appropriate number of NOPs such that the two instructions causing data hazard are spaced by at least 3 instructions.

Control hazards are caused by branch instructions or writes to R15. We need to insert 4 NOPs after each branch instruction to ensure program correctness.

Q2. Can the instructions in the code above be rearranged to reduce the number of NOPs inserted to eliminate data hazards?



Q2. Can the instructions in the code above be rearranged to reduce the number of NOPs inserted to eliminate data hazards?



L2: ADD R1, R2, #8

L3: SUB R2, R5, R7

L1: AND R8, R9, R11

L5: CMP R0, #0

L4: LDR R3, [R1, #4]

L6: BNE SKIP

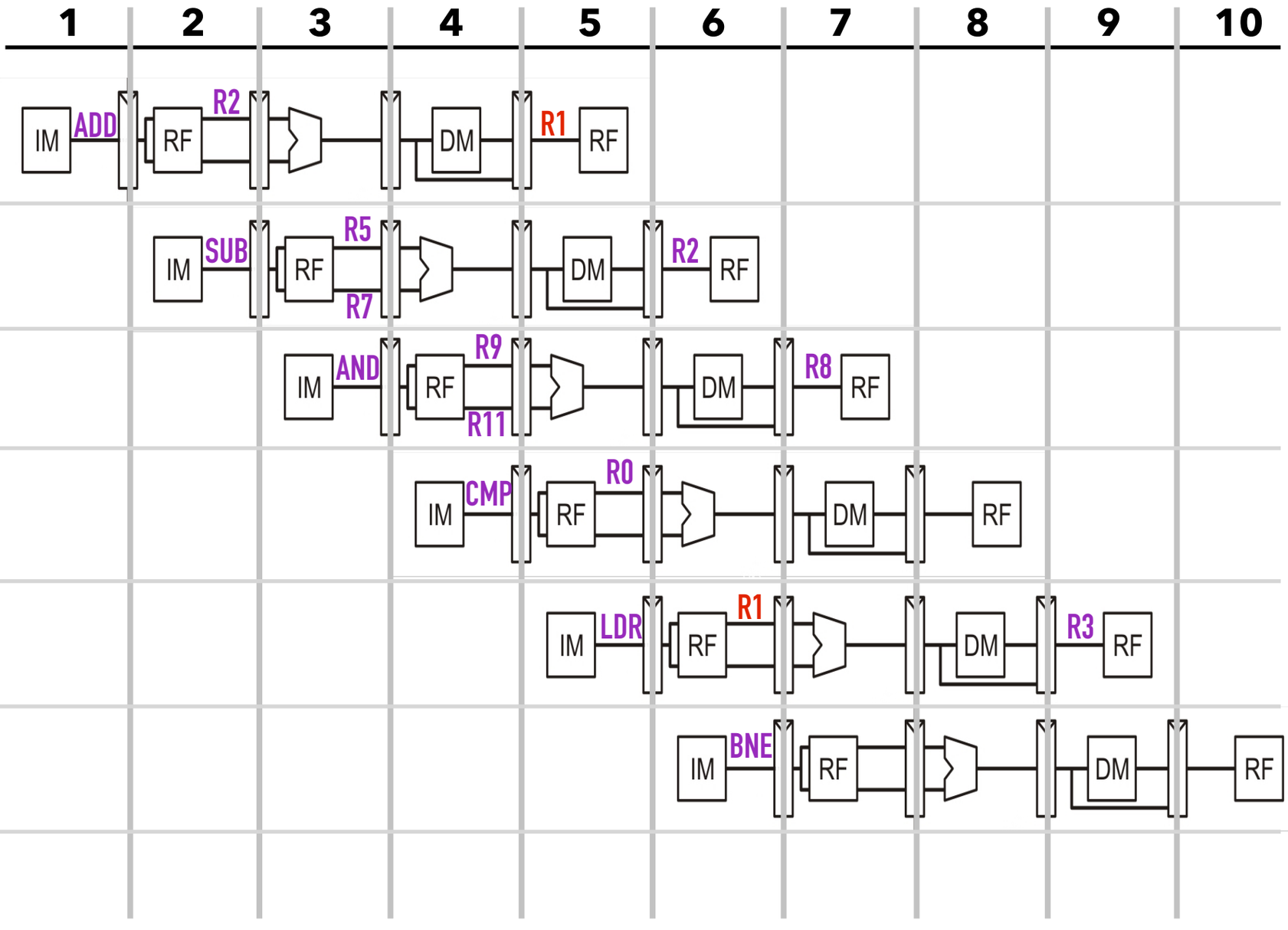
NOP

NOP

NOP

NOP

Q3. What is the total number of cycles required for executing the code you obtained in Q2 if the branch is not taken?



NOP

NOP

NOP

NOP

Q3. What is the total number of cycles required for executing the code you obtained in Q2 if the branch is not taken?

```
L2: ADD R1, R2, #8          L6: BNE SKIP
L3: SUB R2, R5, R7              NOP
L1: AND R8, R9, R11             NOP
L5: CMP R0, #0                  NOP
L4: LDR R3, [R1, #4]            NOP
```

Ans: 4 cycles are required to fill the pipeline. From cycle 5, one instruction completes every cycle. The total number of cycles = 4 + number of instructions = 4+10 = 14. The time taken is the same even if the branch is taken, as the 4 NOPs are inserted at compile-time and the processor executes them irrespective of whether the branch is taken.

```
L2: ADD R1, R2, #8          L6: BNE SKIP
L3: SUB R2, R5, R7              NOP
L1: AND R8, R9, R11             NOP
L5: CMP R0, #0                  NOP
L4: LDR R3, [R1, #4]            NOP
```

The number of NOPs can be reduced even further by moving some instructions from "before the branch" to "after the branch".

For example, L4 can be moved to follow L6. Apart from the fact that L4 has no data dependencies w.r.t L5 and L6*, we should note that L4 is an instruction which should be executed no matter whether the branch is taken or not. Since L4 (after being pushed below L6) enters the pipeline well before L6 changes the PC value (to BTA) if the branch is taken, L4 proceeds to completion normally and will write the correct result to R3.

Up to 4 independent instructions (L5 is not one of them) can be moved from before the branch to after the branch.

*The destination register of L4 is not one of the source registers of L5 or L6; the destination registers of L5 and L6 are not source registers for L4; L4 has a different destination register as compared to L5 and L6. There are some other possible complications associated with rearranging instructions, related to flags and memories. We don't have such issues here though.

THE END

For Consultation: eleguji@nus.edu.sg
Office: E4-03-10