CS-200
Computer Architecture
—
Part 4e. Instruction Level Parallelism
Scheduling Examples

Paolo Ienne
<paolo.ienne@epfl.ch>

RISC-V Program

```
1:      lw      x1, 0(x5)
2:      addi    x5, x1, 1
3:      lw      x1, 0(x6)
4:      add     x3, x8, x5
5:      subi    x2, x6, 1
6:      subi    x4, x3, 5
7:      add     x3, x2, x4
8:      lw      x2, 0(x7)
9:      or      x4, x2, x1
10:     subi    x7, x3, 9
```

Goal

- Determine the **execution schedule** for this code in five different architectures
- **Compare the number of cycles** required by the five architectures
- **Compute the CPI** (and IPC) of all architectures on this program

Architecture #1

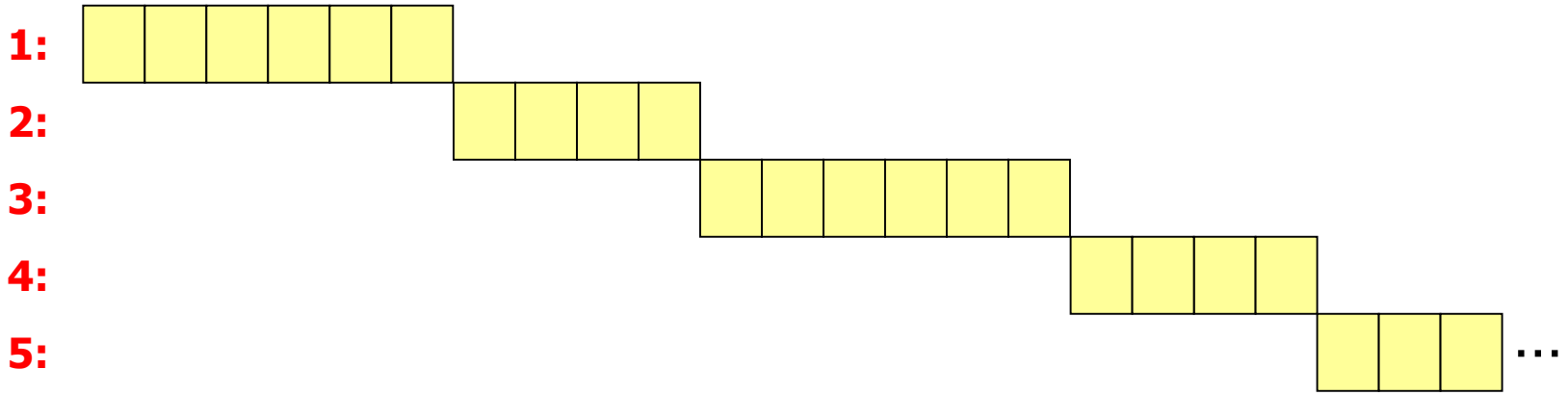
- **Multicycle** processor, not pipelined
- Execution latencies:
  - ALU Operations → 4 cycles
  - Memory Operations → 6 cycles

```
 1:  lw      x1, 0(x5)
 2:  addi    x5, x1, 1
 3:  lw      x1, 0(x6)
 4:  add     x3, x8, x5
 5:  subi    x2, x6, 1
 6:  subi    x4, x3, 5
 7:  add     x3, x2, x4
 8:  lw      x2, 0(x7)
 9:  or      x4, x2, x1
10:  subi    x7, x3, 9
```

ALU Operations → 4 cycles

Memory Operations → 6 cycles

Solution
Architecture #1



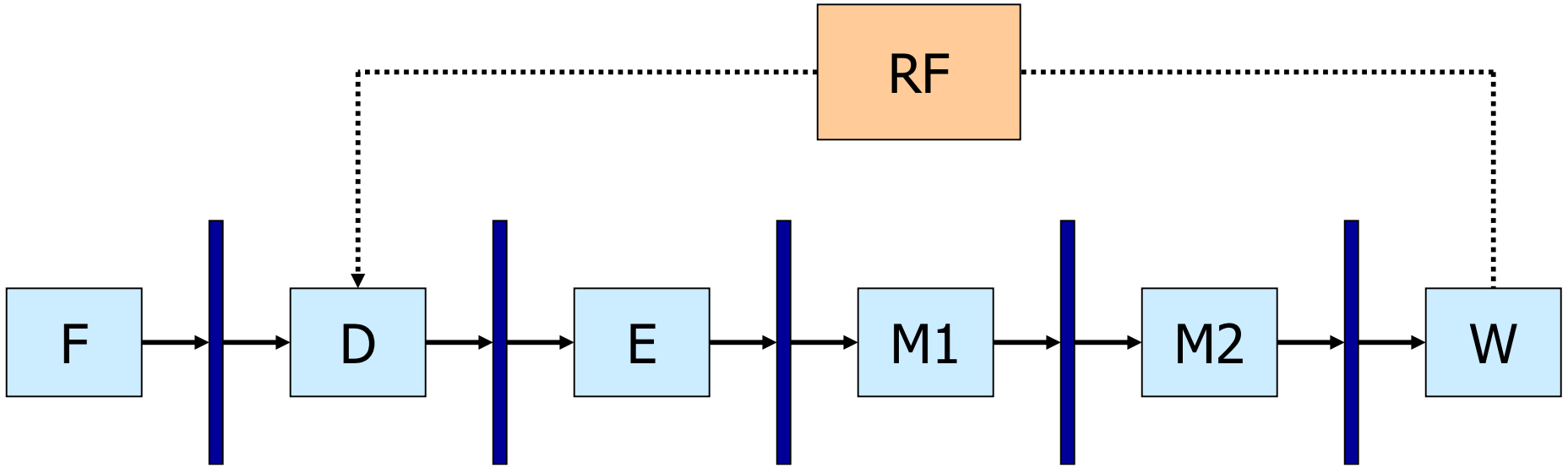

```
1:    lw     x1, 0(x5)    → 6 cycles
2:    addi   x5, x1, 1    → 4 cycles
3:    lw     x1, 0(x6)    → 6 cycles
4:    add    x3, x8, x5   → 4 cycles
5:    subi   x2, x6, 1    → 4 cycles
6:    subi   x4, x3, 5    → 4 cycles
7:    add    x3, x2, x4   → 4 cycles
8:    lw     x2, 0(x7)    → 6 cycles
9:    or     x4, x2, x1   → 4 cycles
10:   subi   x7, x3, 9    → 4 cycles
```

CPI = 4.6

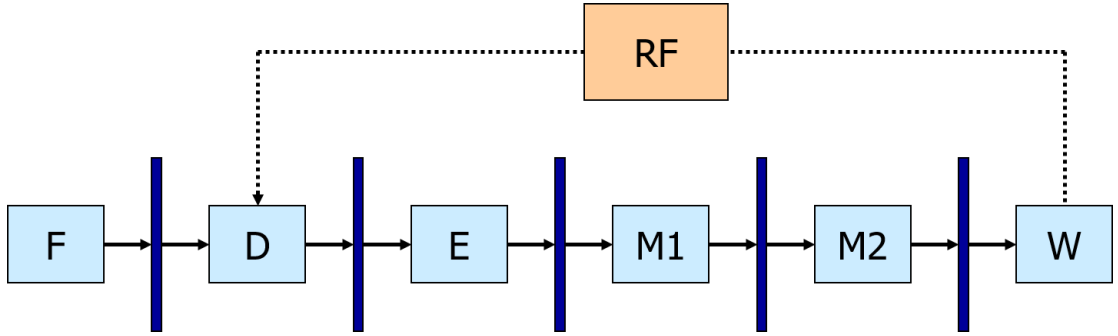
46 cycles = 10 instructions

Architecture #2

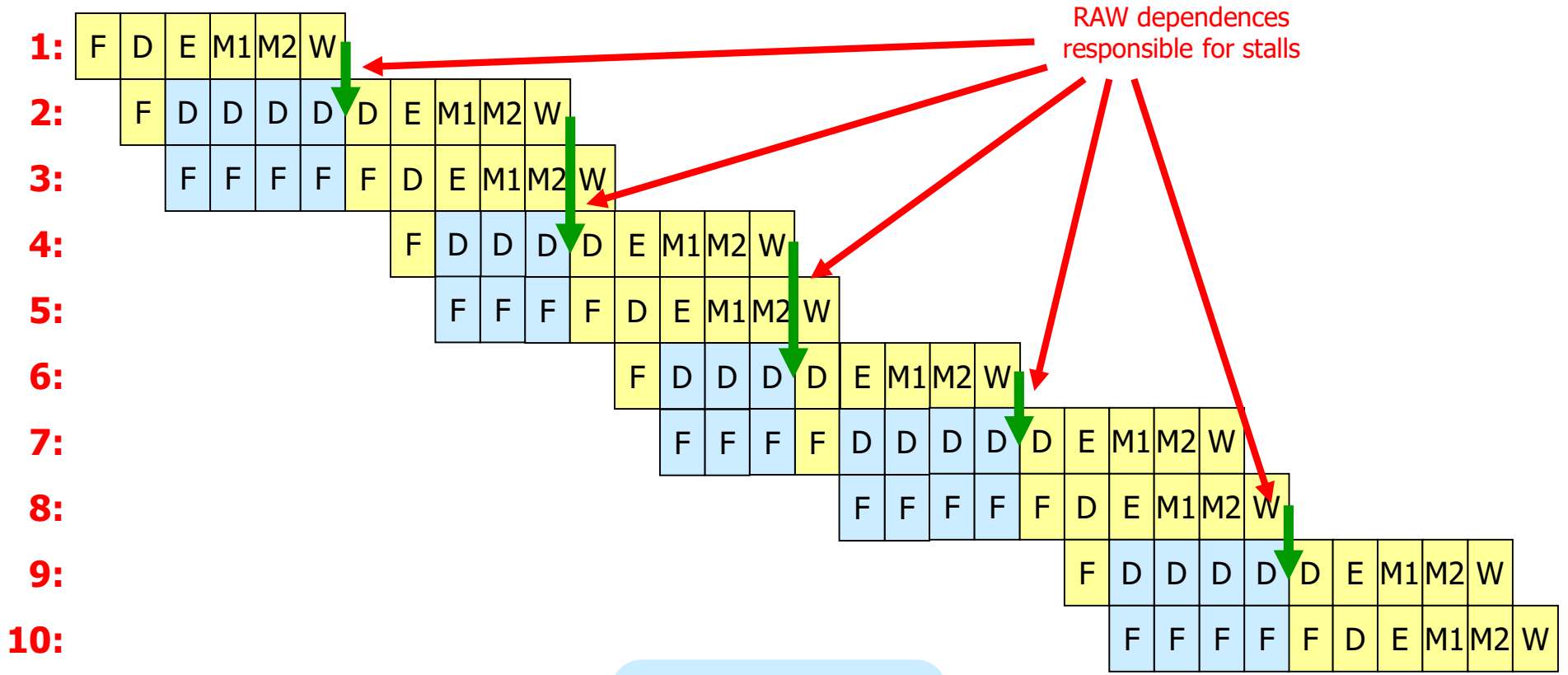
- 6-stage pipelined, no forwarding paths



```
 1:  lw      x1, 0(x5)
 2:  addi    x5, x1, 1
 3:  lw      x1, 0(x6)
 4:  add     x3, x8, x5
 5:  subi    x2, x6, 1
 6:  subi    x4, x3, 5
 7:  add     x3, x2, x4
 8:  lw      x2, 0(x7)
 9:  or      x4, x2, x1
10:  subi    x7, x3, 9
```



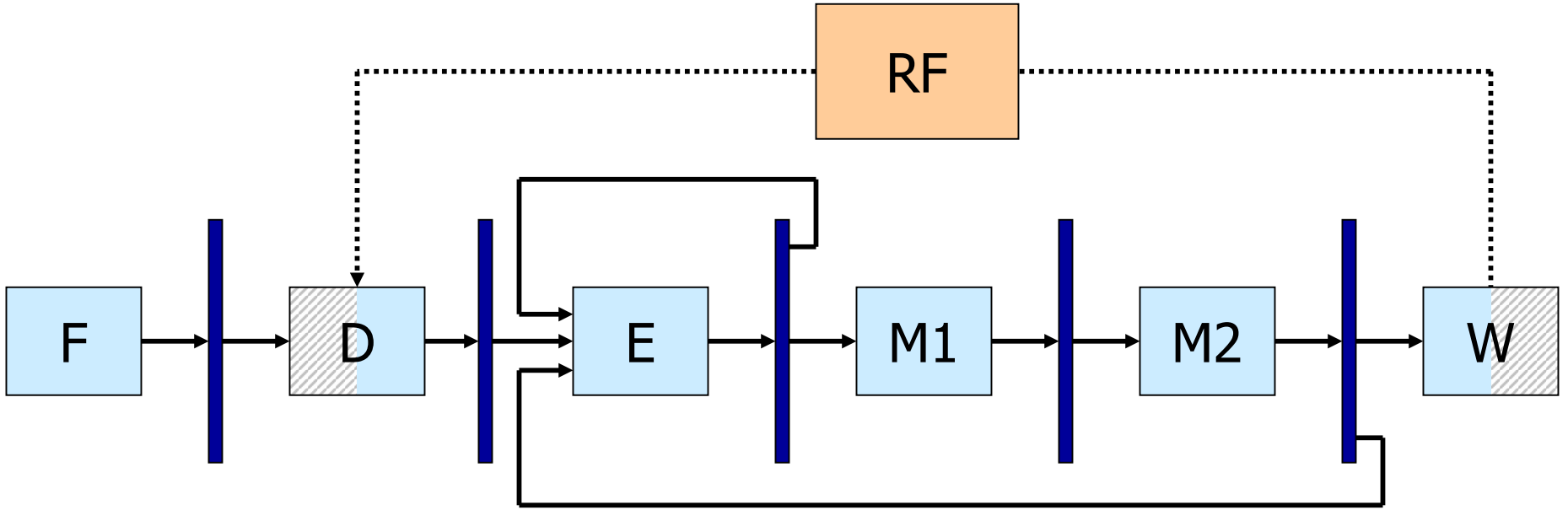
Solution
Architecture #2



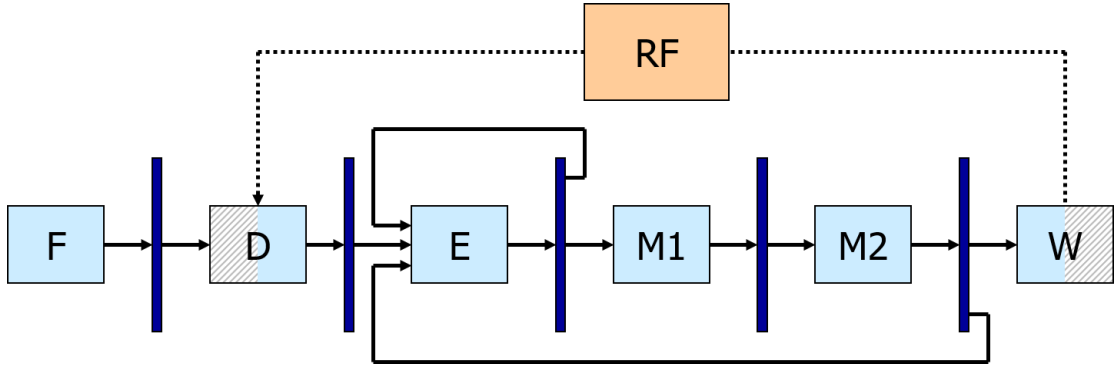
CPI = 3.3

Architecture #3

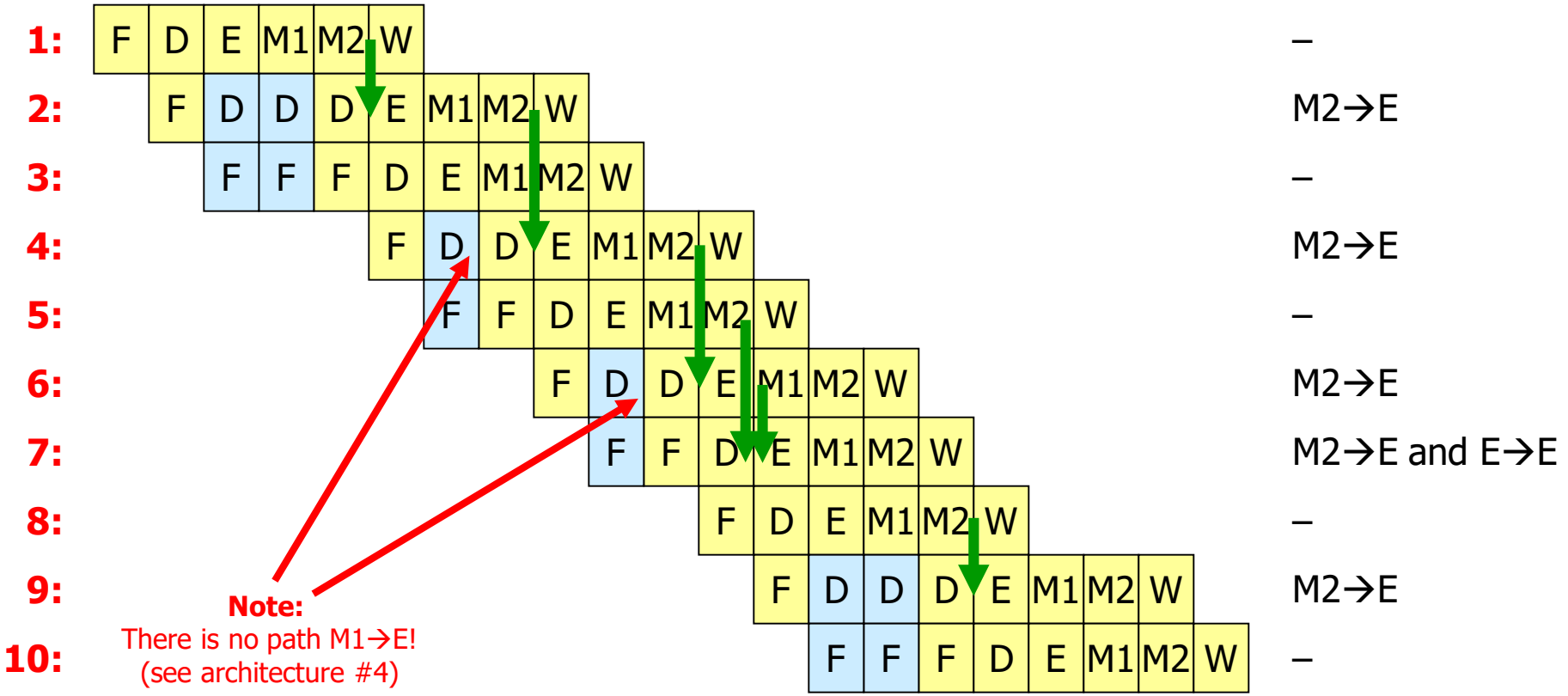
- 6-stage pipelined, some forwarding paths:
  - **E→E, M2→E**
  - **W→D**



```
 1:  lw      x1, 0(x5)
 2:  addi    x5, x1, 1
 3:  lw      x1, 0(x6)
 4:  add     x3, x8, x5
 5:  subi    x2, x6, 1
 6:  subi    x4, x3, 5
 7:  add     x3, x2, x4
 8:  lw      x2, 0(x7)
 9:  or      x4, x2, x1
10:  subi    x7, x3, 9
```



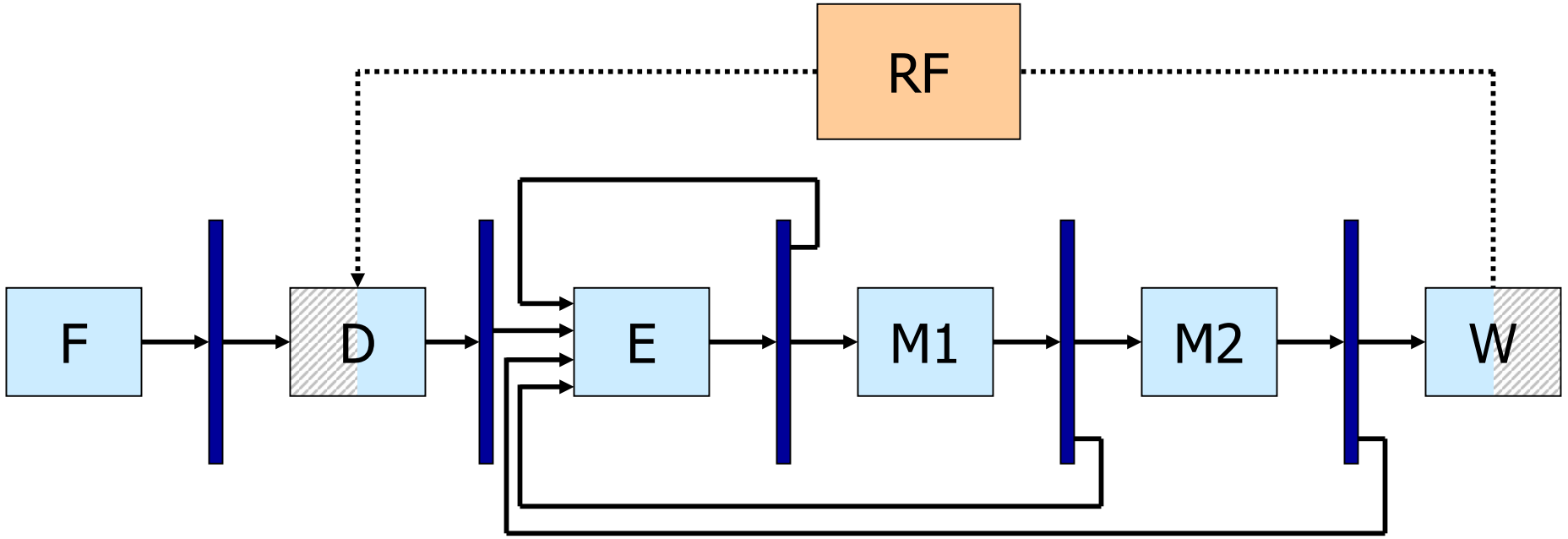
Solution
Architecture #3



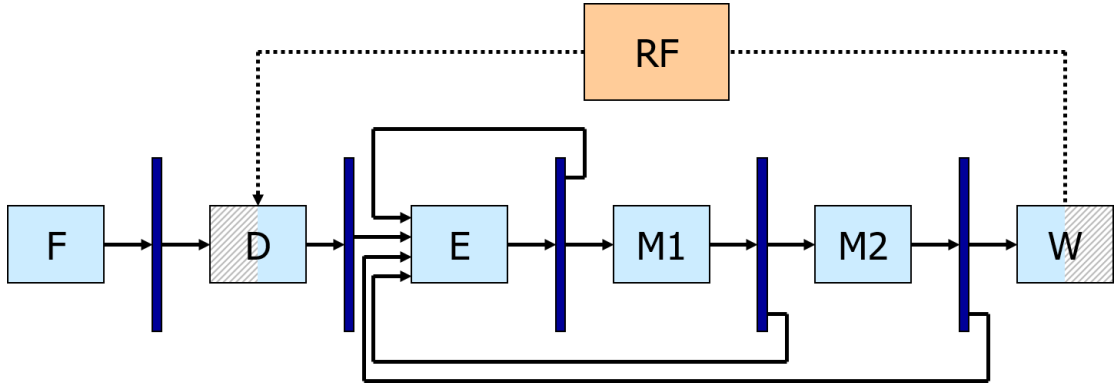
CPI = 2.1

Architecture #4

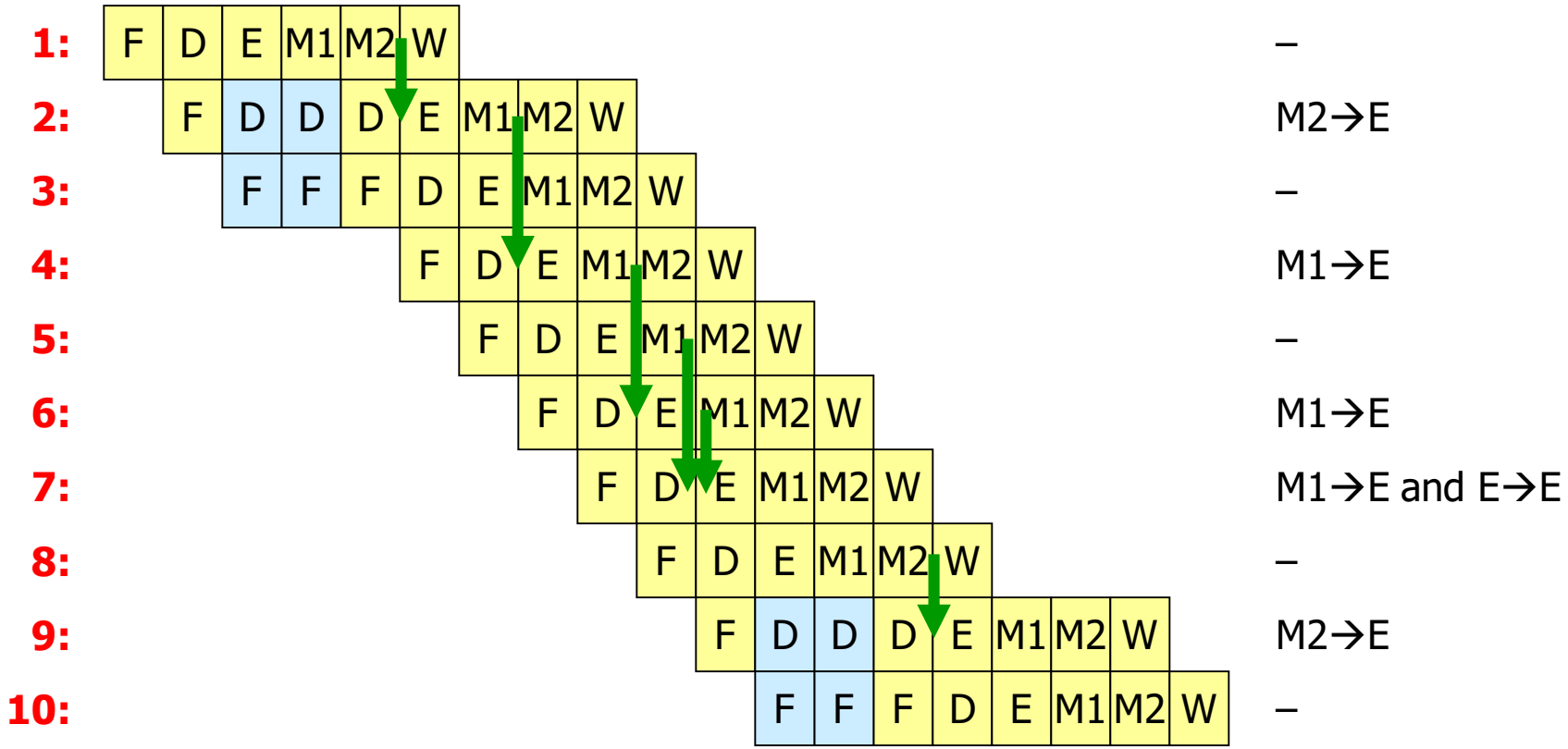
- 6-stage pipelined, all forwarding paths:
  - **E→E, M1→E, M2→E**
  - **W→D**



```
 1:  lw      x1, 0(x5)
 2:  addi    x5, x1, 1
 3:  lw      x1, 0(x6)
 4:  add     x3, x8, x5
 5:  subi    x2, x6, 1
 6:  subi    x4, x3, 5
 7:  add     x3, x2, x4
 8:  lw      x2, 0(x7)
 9:  or      x4, x2, x1
10:  subi    x7, x3, 9
```



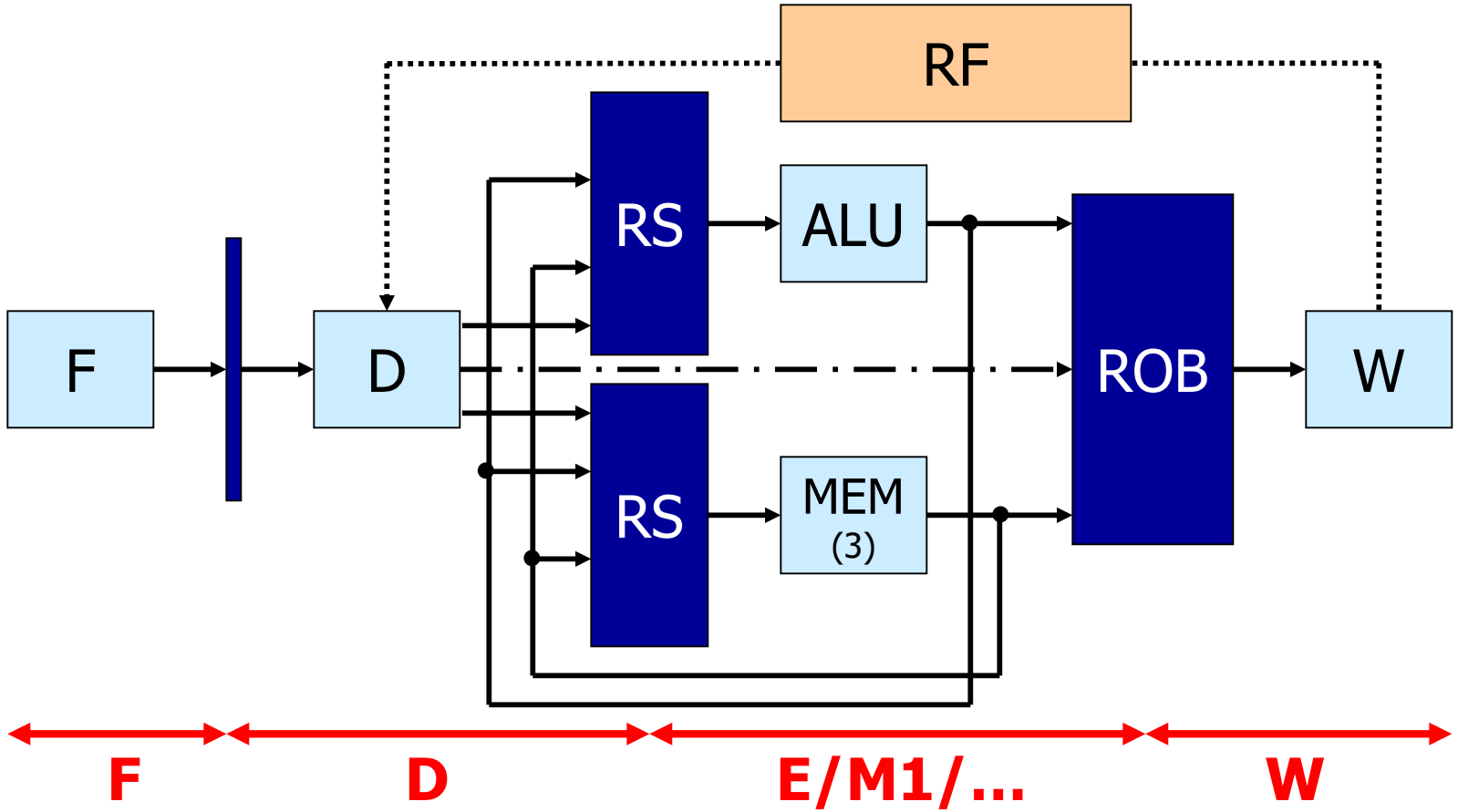
Solution
Architecture #4



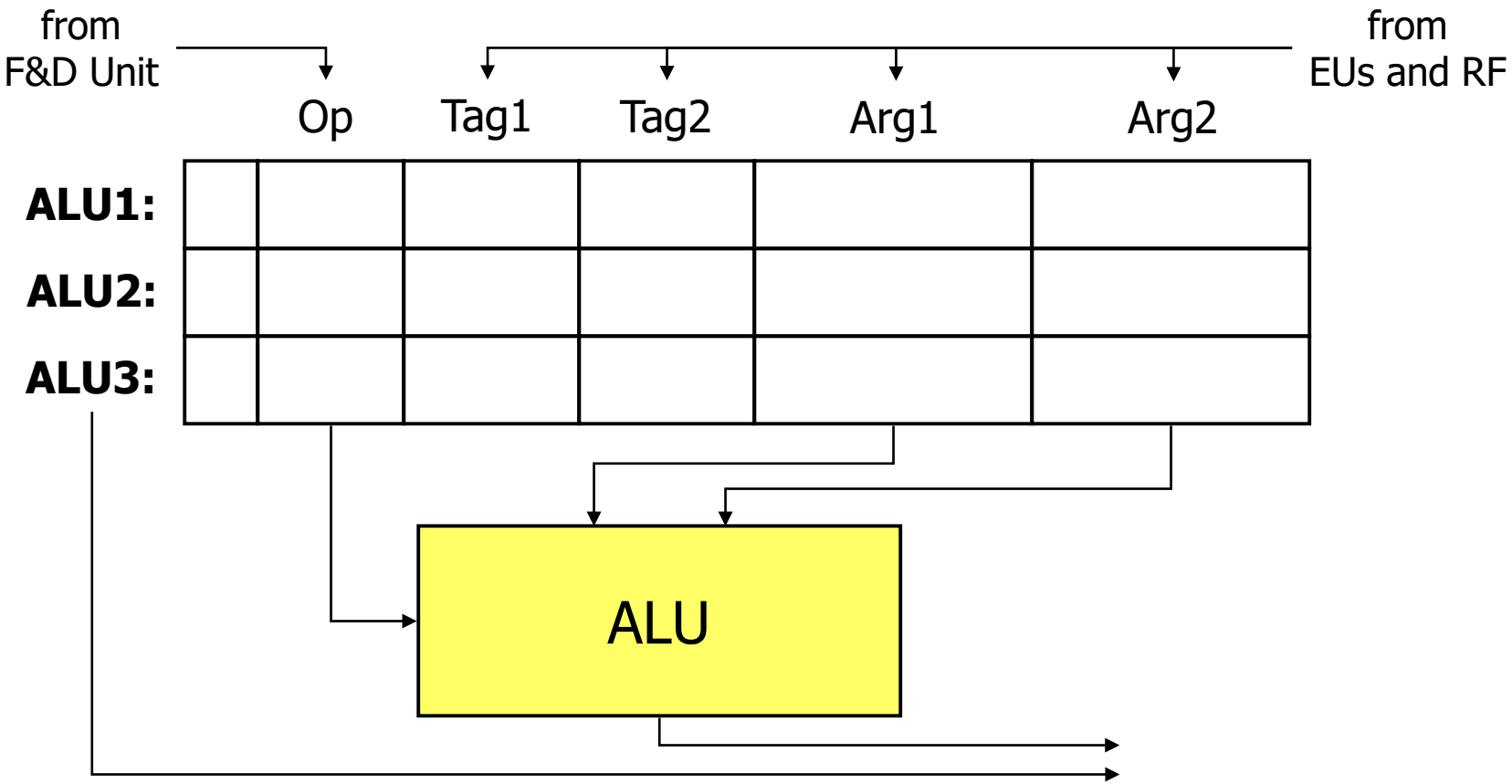
CPI = 1.9

Architecture #5

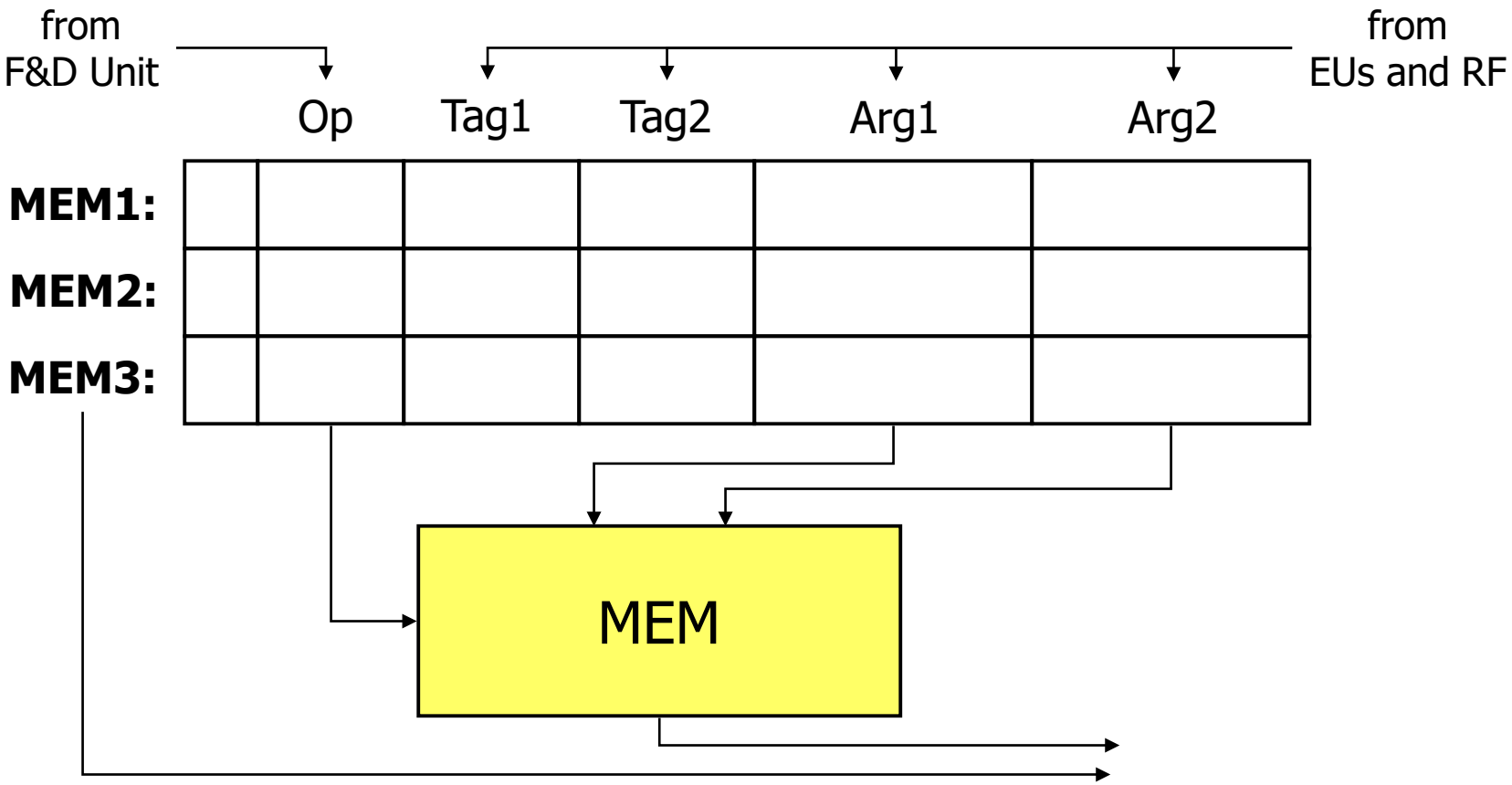
- Dynamically scheduled, out-of-order (OOO), unlimited RS and ROB size
- 1 ALU (latency 1) + 1 Memory Unit (latency 3)



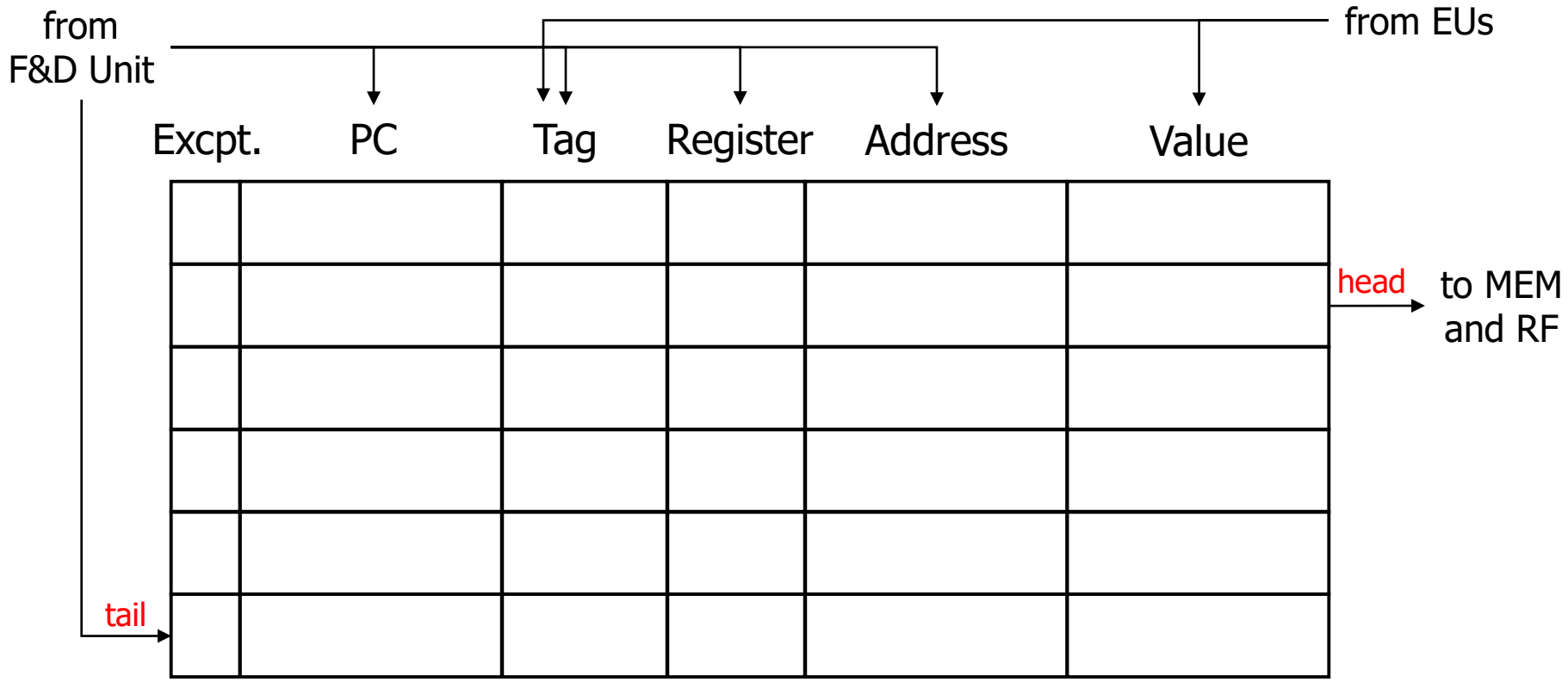
ALU Reservation Station (RS)



MEM Reservation Station (RS)

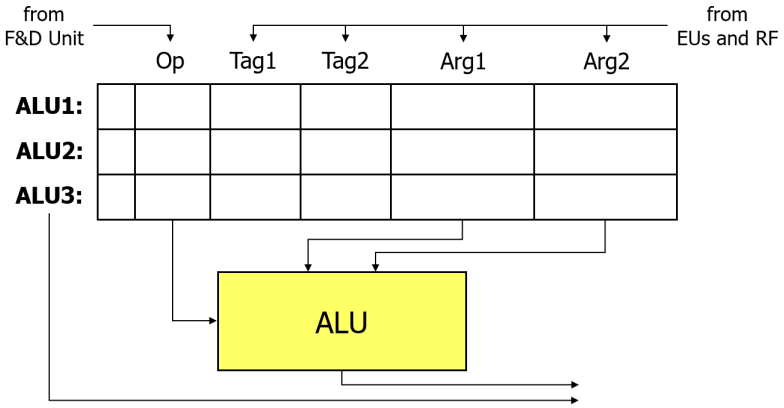


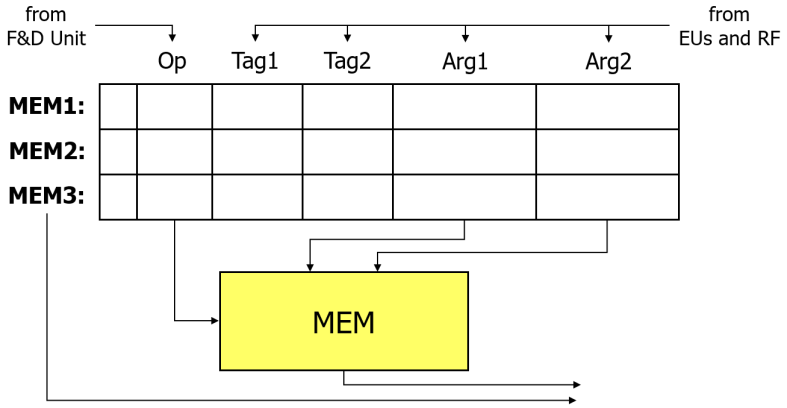
Reordering Buffer (ROB)

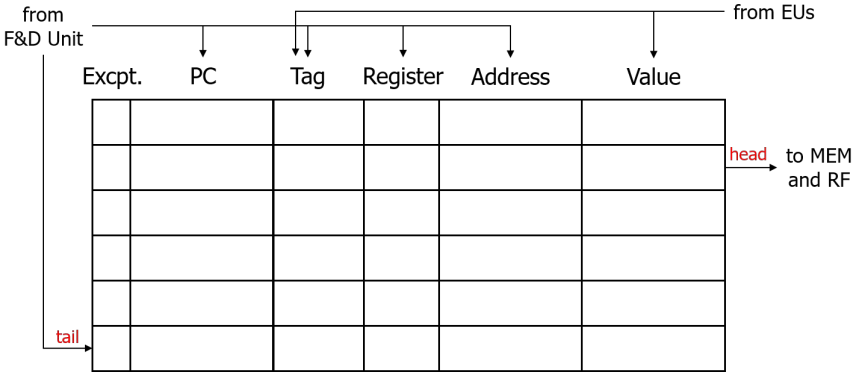


"To Do List" for Simulating Dynamically Scheduled Processors

- At the beginning of each cycle
  - **E phase**—Issue ready instructions:
    - From all RSs, issue as many ready instructions as there are FUs available
  - **W phase**—Writeback results in order:
    - Remove top entry from ROB if completed
- At the end of each cycle
  - **D phase**—Load result of decoding stage:
    - To the relevant RS, including ready register values
    - To the ROB, to prepare the placeholder for the result
  - **E phase**—Broadcast results from all FUs:
    - To all RSs (incl. deallocation of the entry)
    - To the ROB

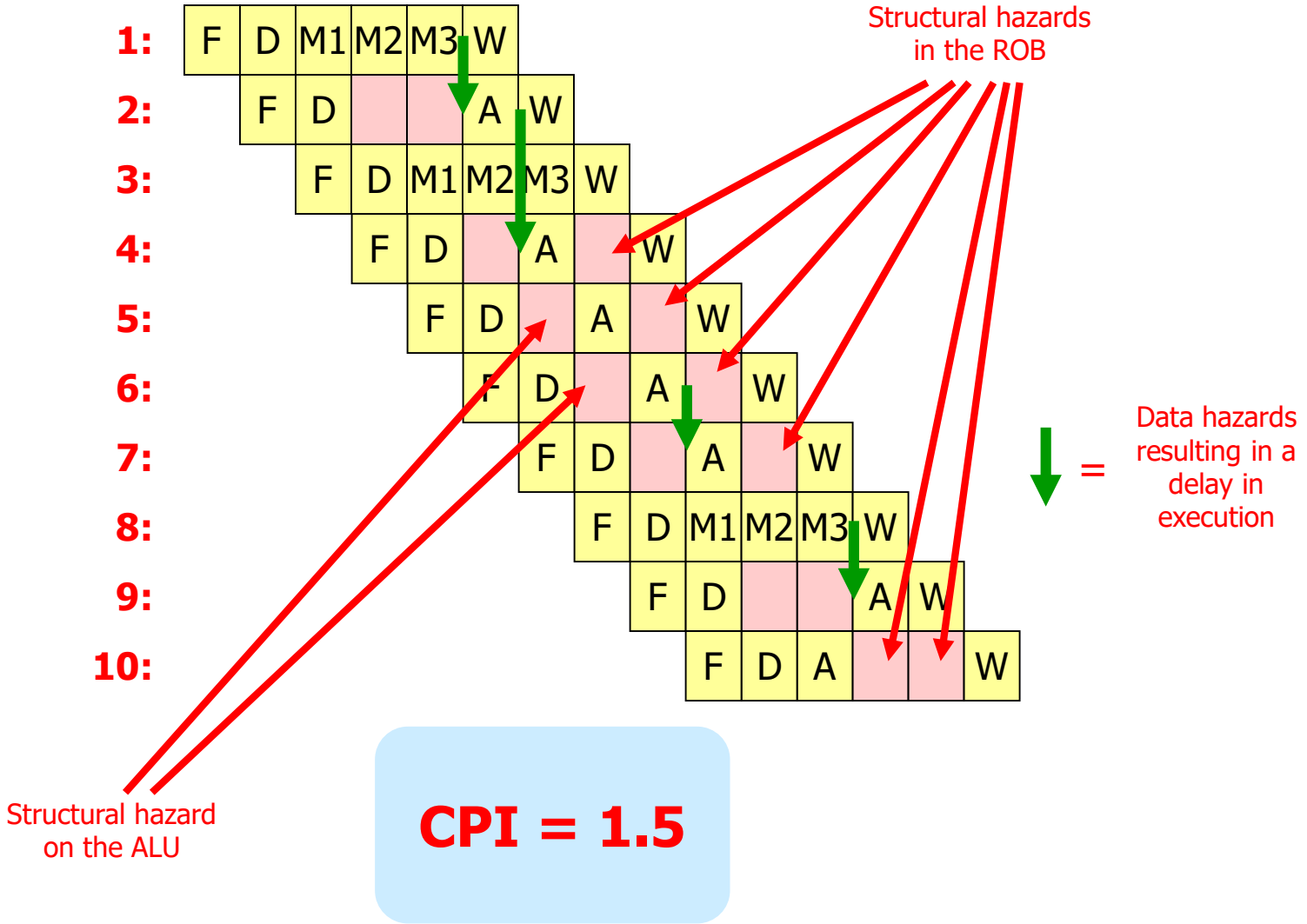






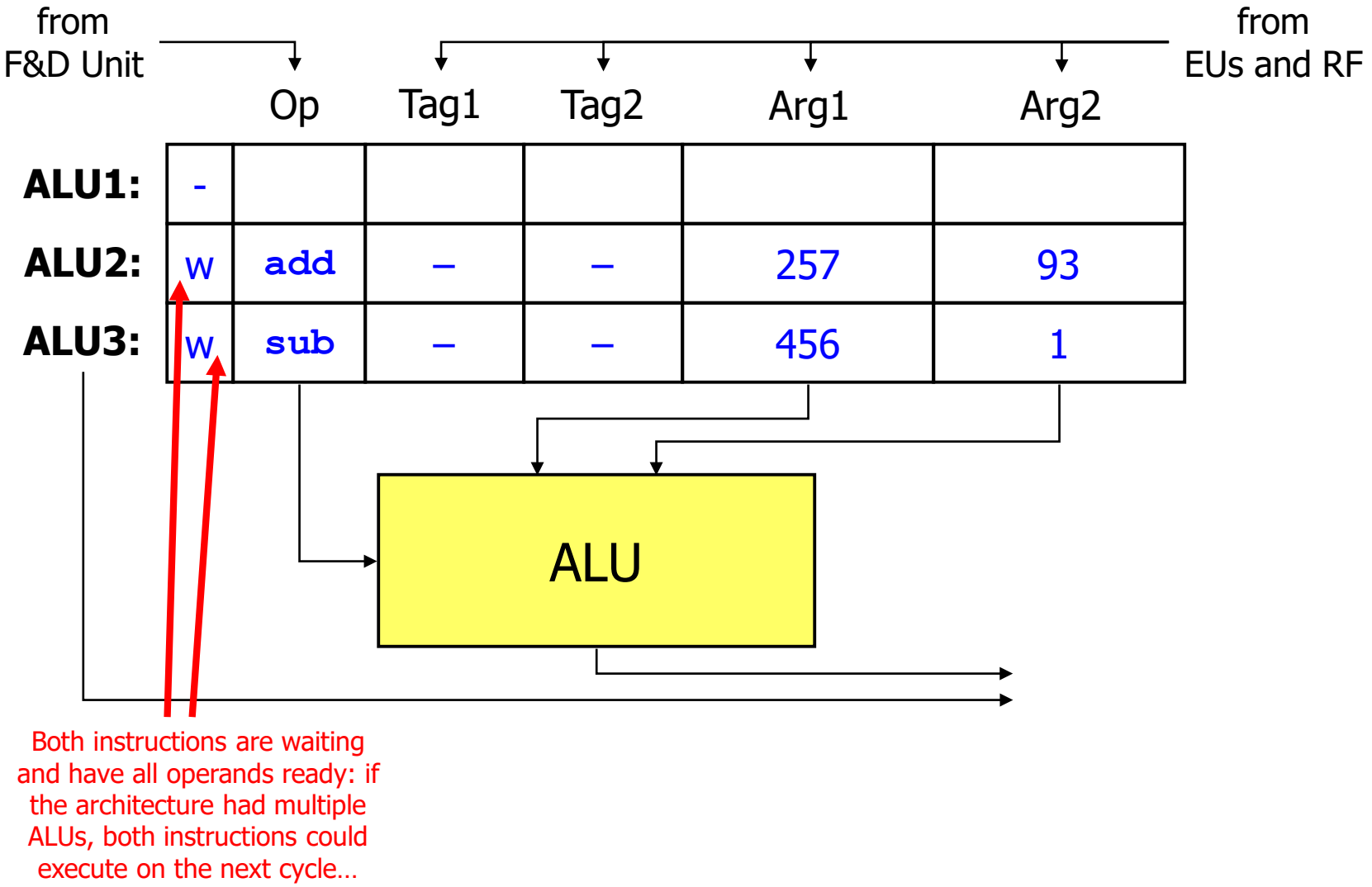

```
 1:  lw      x1, 0(x5)
 2:  addi    x5, x1, 1
 3:  lw      x1, 0(x6)
 4:  add     x3, x8, x5
 5:  subi    x2, x6, 1
 6:  subi    x4, x3, 5
 7:  add     x3, x2, x4
 8:  lw      x2, 0(x7)
 9:  or      x4, x2, x1
10:  subi    x7, x3, 9
```

Solution
Architecture #5



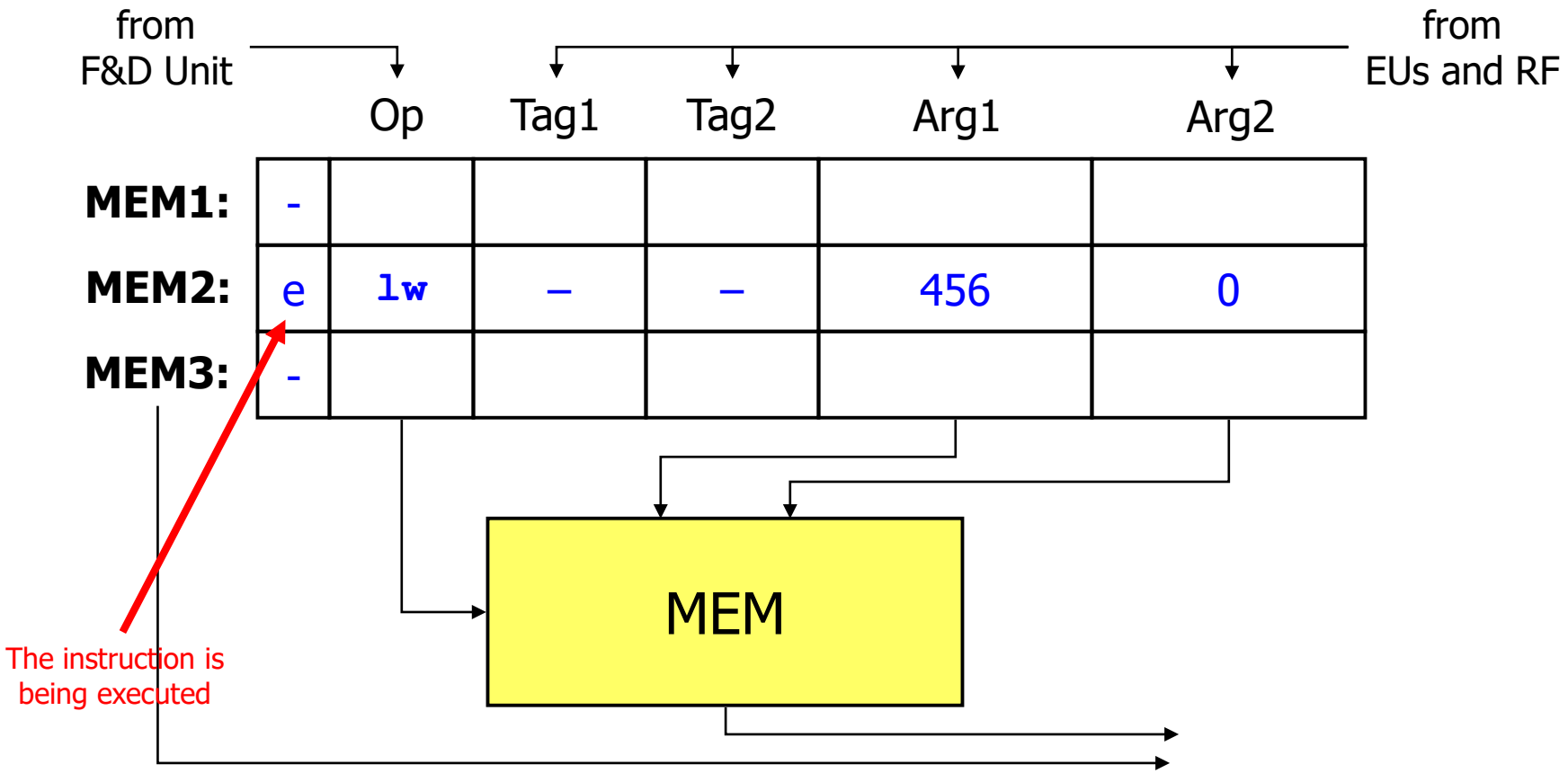
Solution

ALU RS at the end of the 6th cycle



Solution

MEM RS at the end of the 6th cycle



Solution
ROB at the end of the 6th cycle