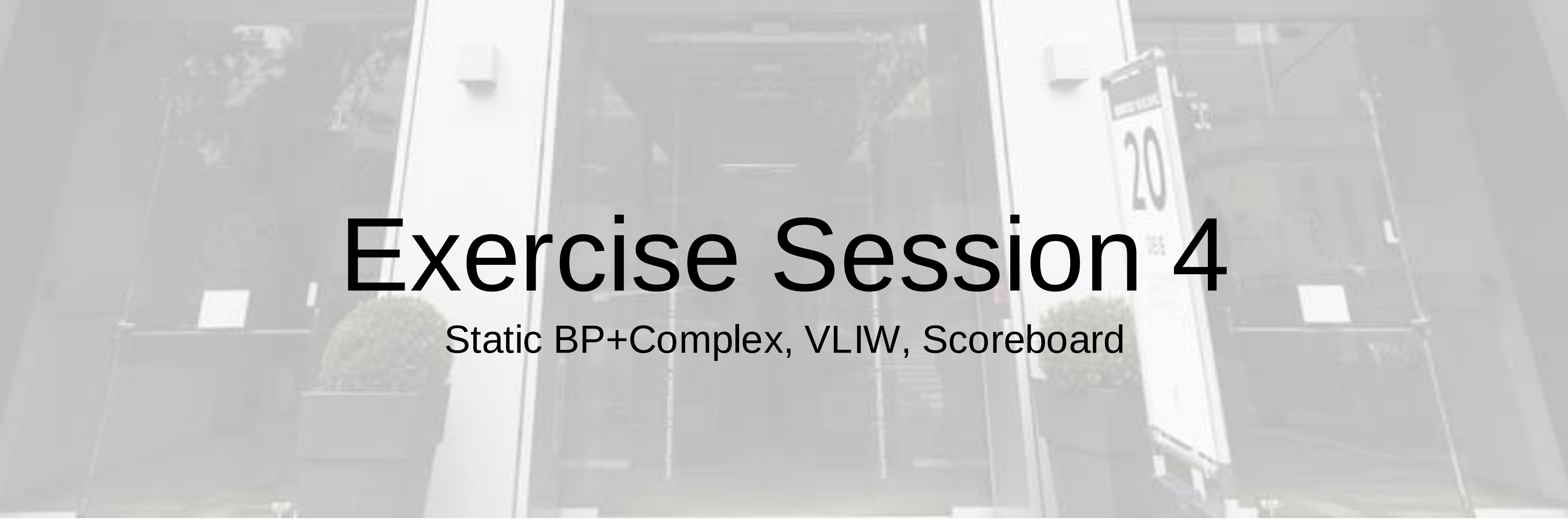


# Exercise Session 4

Static BP+Complex, VLIW, Scoreboard

Advanced Computer Architectures

Politecnico di Milano

April 2nd, 2025

Alessandro Verosimile <alessandro.verosimile@polimi.it>

## Recall: Static Branch Prediction Techniques

**Branch Always Not Taken (Predicted-Not-Taken)**

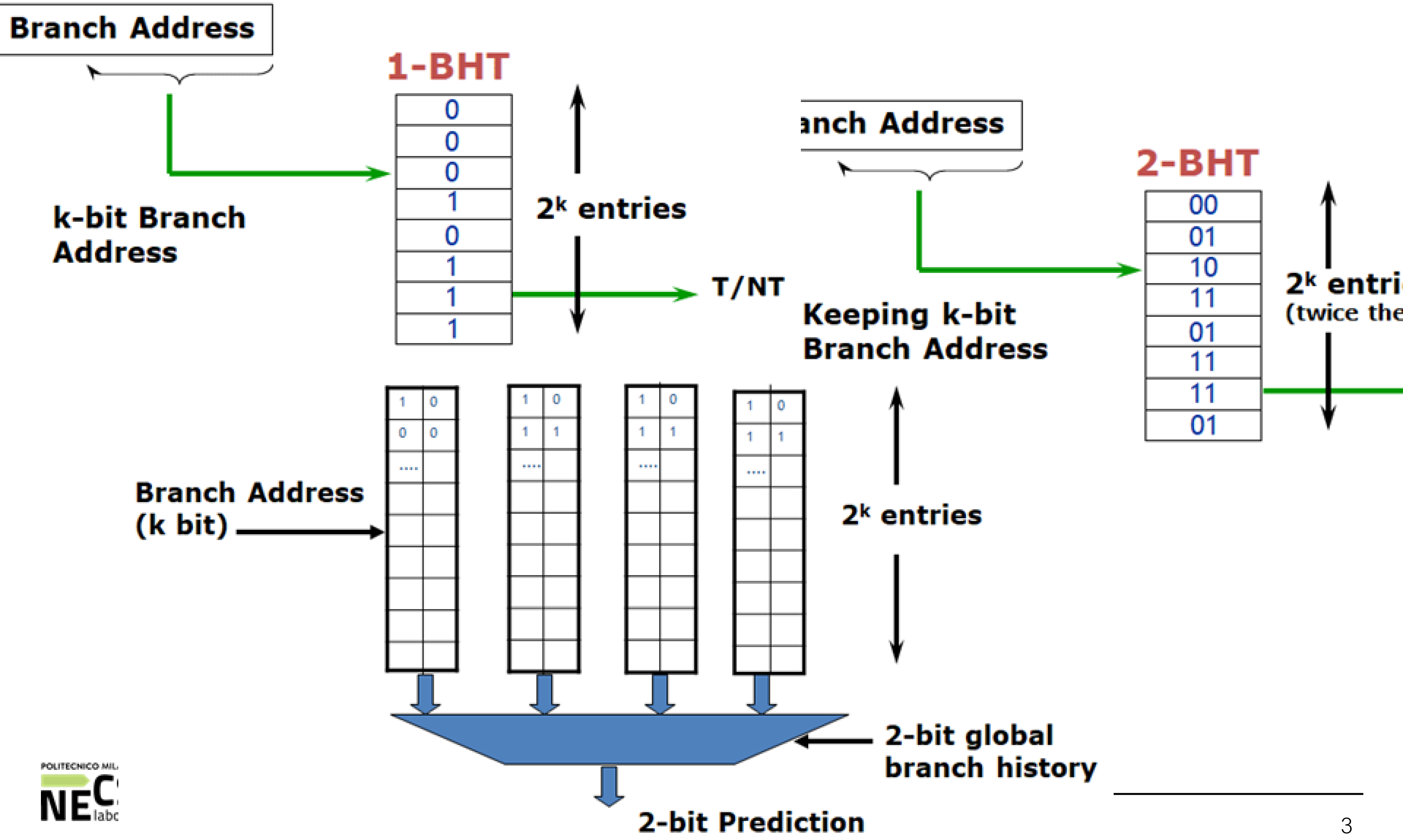
**Branch Always Taken (Predicted-Taken)**

**Backward Taken Forward Not Taken (BTFNT)**

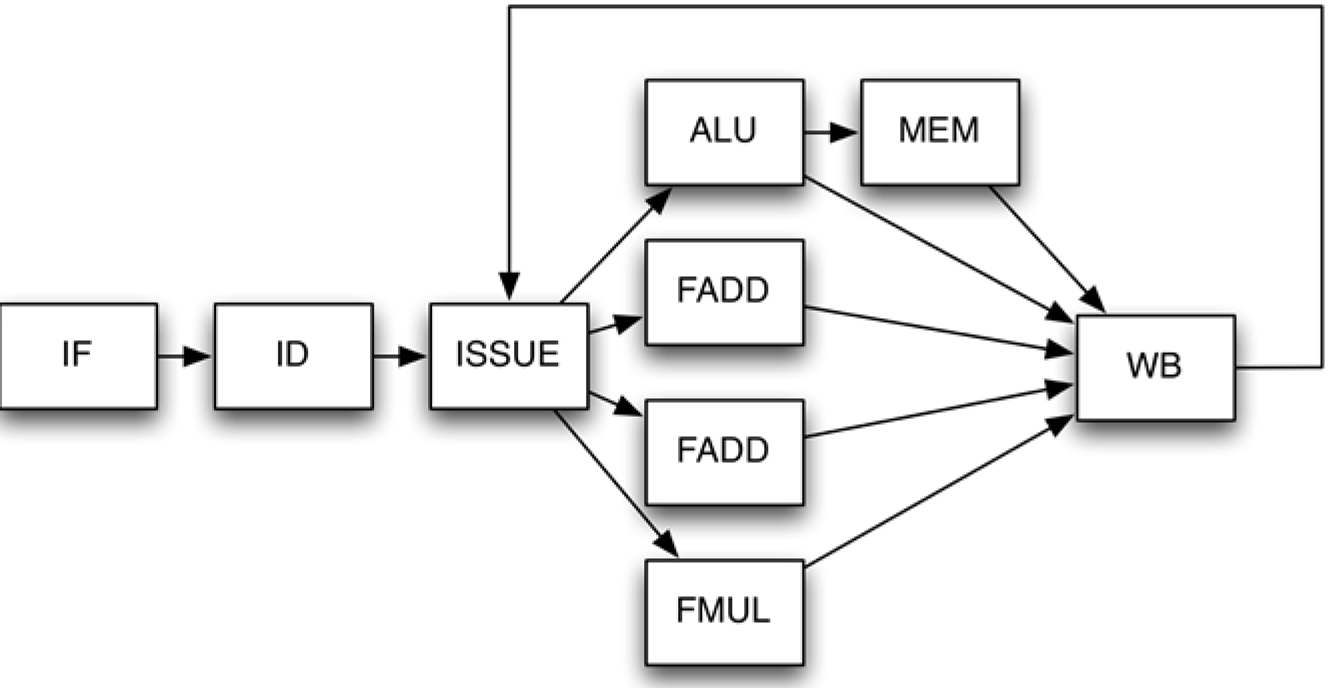
**Profile-Driven Prediction**

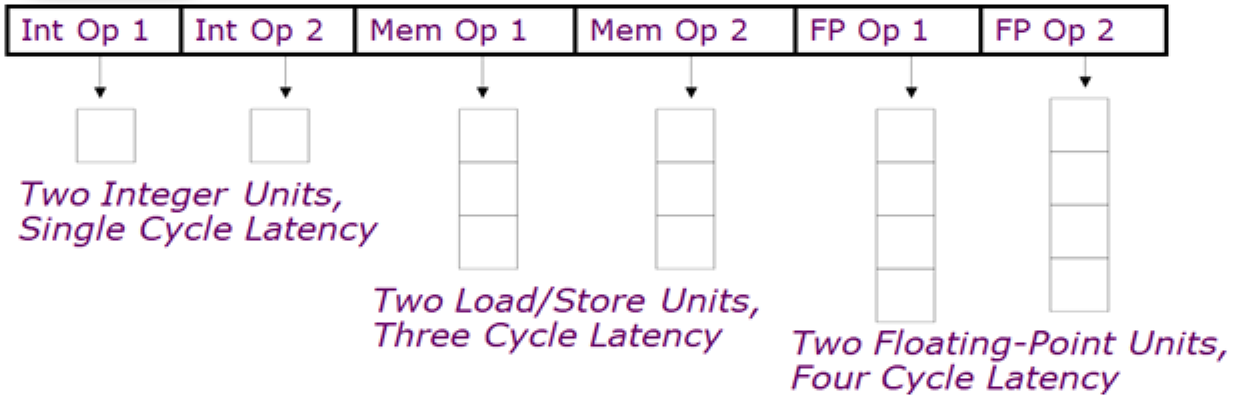
**Delayed Branch**

# Recall: Dynamic Branch Prediction



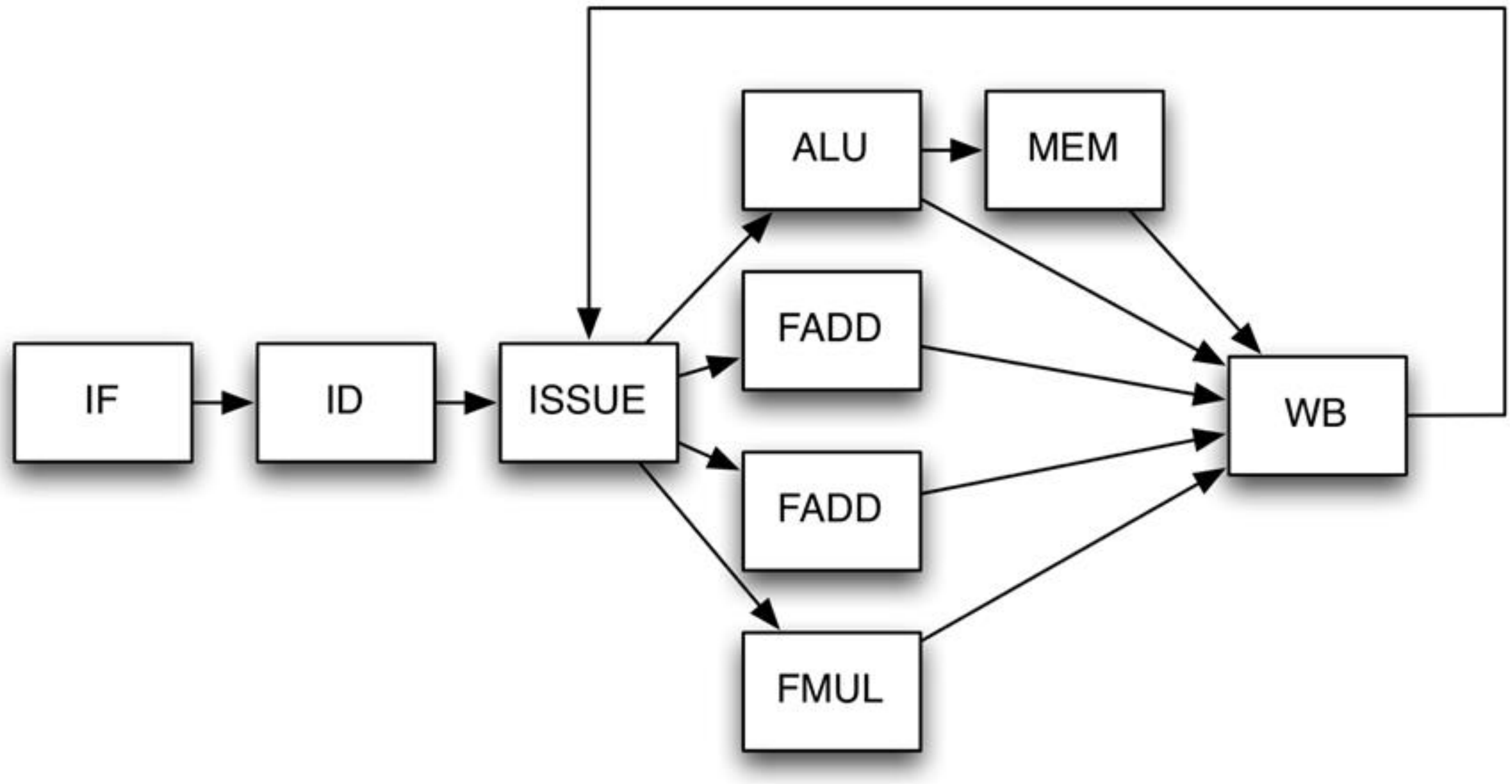
# Recall: OoO and VLIW





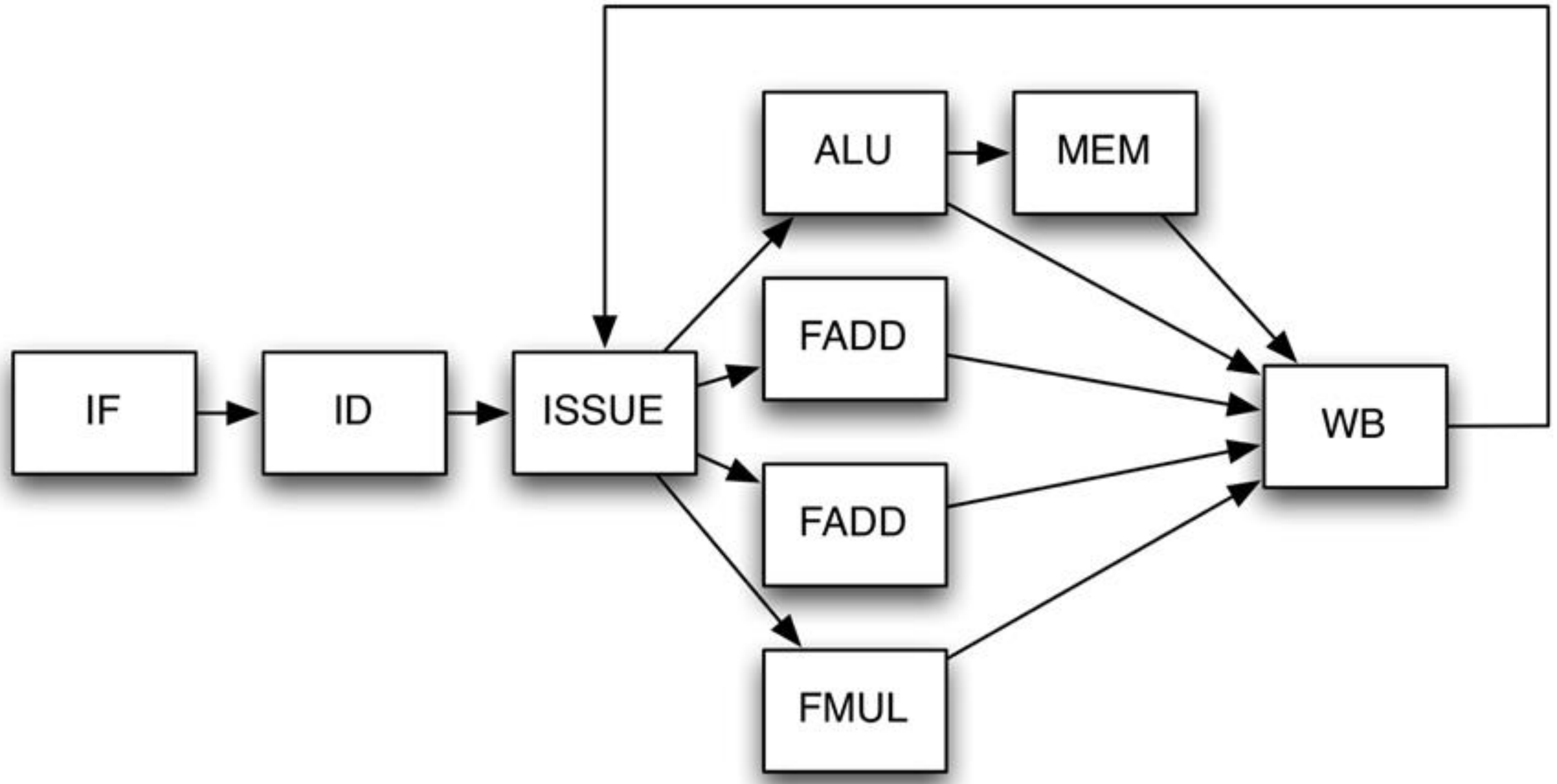
Int Op 1
Int Op 2
Mem Op 1
Mem Op 2
FP Op 1
FP Op 2
Two Integer Units, Single Cycle Latency
Two Load/Store Units, Three Cycle Latency
Two Floating-Point Units, Four Cycle Latency

## Exe: Complex Pipeline

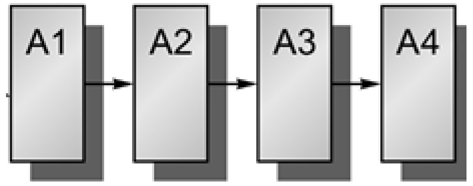


# Exe: Complex Pipeline

In this problem we will examine the execution of a code segment on the following single-issue out-of-order processor:

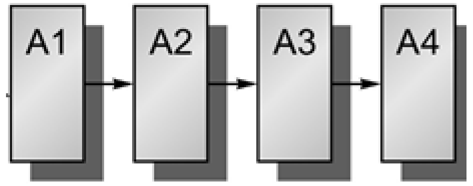


## You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage
- The target address for a branch is available in the FETCH stage

# You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage
- The target address for a branch is available in the FETCH stage

## Code

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

## Code and Architecture

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

|  |  |
|---|---|
| ALU OP: | 1 cycle |
| MEM OP: | 3 cycles |
| FP ADD: | 3 cycles |
| FP MULT: | 5 cycles |

# Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

## Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3,$f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1-I2**
**RAW F3 I2-I3**
**RAW F3 I4-I5**
**RAW F3 I4-I6**
**RAW R7 I8-I9**

# Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3,$f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1-I2**
**RAW F3 I2-I3**
**RAW F3 I4-I5**
**RAW F3 I4-I6**
**RAW R7 I8-I9**
**WAR R7 I8-I5**
**WAR R7 I8-I3**

## Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3,$f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1–I2**
**RAW F3 I2–I3**
**RAW F3 I4–I5**
**RAW F3 I4–I6**
**RAW R7 I8–I9**
**WAR R7 I8–I5**
**WAR R7 I8–I3**
**WAR R6 I7–I1**
**WAR R6 I7–I4**

# Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1–I2**
**RAW F3 I2–I3**
**RAW F3 I4–I5**
**RAW F3 I4–I6**
**RAW R7 I8–I9**
**WAR R7 I8–I5**
**WAR R7 I8–I3**
**WAR R6 I7–I1**
**WAR R6 I7–I4**
**WAW F3 I2–I4**

# Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1–I2**
**RAW F3 I2–I3**
**RAW F3 I4–I5**
**RAW F3 I4–I6**
**RAW R7 I8–I9**
**WAR R7 I8–I5**
**WAR R7 I8–I3**
**WAR R6 I7–I1**
**WAR R6 I7–I4**
**WAW F3 I2–I4**
**WAR F3 I3–I4**

# Conflicts

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3,$f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1-I2**
**RAW F3 I2-I3**
**RAW F3 I4-I5**
**RAW F3 I4-I6**
**RAW R7 I8-I9**
**WAR R7 I8-I5**
**WAR R7 I8-I3**
**WAR R6 I7-I1**
**WAR R6 I7-I4**
**WAW F3 I2-I4**
**WAR F3 I3-I4**
**CNTRL**

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I1 - I2 F2 |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I4 F3<br>WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6<br>WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7,<br>WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I4 F3<br>WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6<br>WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7,<br>WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | ISs | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I4 F3<br>WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6<br>WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7,<br>WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I4 F3 WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6 WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7, WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6 WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7, WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | ISs | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | ISs | ISs | ISs | ISs | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | Ds | Ds | Ds | Ds | Ds | Ds | Ds | Ds | Ds | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | Fs | Fs | Fs | Fs | Fs | Fs | Fs | Fs | Fs | F | D | ISs | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6 WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7, WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I1-I7 r6 WAR I4-I7 r6, |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7, WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | F | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | ~~WAR I1-I7 r6~~ ~~WAR I4-I7 r6,~~ Struct WB |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WAR I3-I8 r7, WAR I5-I8 r7, |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | F | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | ~~WAR I1-I7 r6~~ ~~WAR I4-I7 r6,~~ ~~Struct WB~~ |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | F s | F | D s | D s | D s | D | IS s | IS | E1 | W | | | ~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ Struct ALU |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | RAW R7 I8-I9 |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | F | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | ~~WAR I1-I7 r6~~ ~~WAR I4-I7 r6,~~ ~~Struct WB~~ |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | F s | F | D s | D s | D s | D | IS s | IS | E1 | W | | | ~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ ~~Struct ALU~~ |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | ~~RAW R7 I8-I9~~ |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | CNTRL |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | F | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | ~~WAR I1-I7 r6~~ ~~WAR I4-I7 r6,~~ ~~Struct WB~~ |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | F s | F | D s | D s | D s | D | IS s | IS | E1 | W | | | ~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ ~~Struct ALU~~ |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | ~~RAW R7 I8-I9~~ |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | F s | F s | F s | F s | F | D | ~~CNTRL~~ |

# Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | F | D | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | | | | | ~~RAW I1 - I2 F2~~ |
| 3 | st $f3, VA($r7) | | | F | D | IS s | IS s | IS s | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | | | | | | | | | ~~RAW I2 - I3 F3~~ |
| 4 | ld $f3, VC($r6) | | | | F | D s | D s | D s | D s | D s | D s | D s | D s | D s | D | IS | E1 | E2 | E3 | W | | | | | | | | | | ~~WAR I3-I4 F3~~ ~~WAW I2-I4 F3~~ |
| 5 | st $f3, VC($r7) | | | | | F s | F s | F s | F s | F s | F s | F s | F s | F s | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | ~~RAW I4 - I5 F3~~ |
| 6 | fadd $f4,$f4,$f3 | | | | | | | | | | | | | | | F | D | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | ~~RAW I4 - I6 F3~~ |
| 7 | addi $r6, $r6, 4 | | | | | | | | | | | | | | | | F | D s | D | IS s | IS s | IS | E1 | E1 | E1 s | W | | | | ~~WAR I1-I7 r6~~ ~~WAR I4-I7 r6,~~ ~~Struct [illegible]~~ |
| 8 | addi $r7, $r7, 4 | | | | | | | | | | | | | | | | | F s | F | D s | D s | D s | D | IS s | IS | E1 | W | | | ~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ ~~Struct ALU~~ |
| 9 | blt $r7, $r8, FOR | | | | | | | | | | | | | | | | | | | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | ~~RAW R7 I8-I9~~ |
| 10 | (New Loop Iteration) | | | | | | | | | | | | | | | | | | | | | | | F s | F s | F s | F s | F | D | ~~CNTRL~~ |

**What if a Static Branch Prediction?**

## Recall: Static Branch Prediction Techniques

**Branch Always Not Taken (Predicted-Not-Taken)**

**Branch Always Taken (Predicted-Taken)**

**Backward Taken Forward Not Taken (BTFNT)**

**Profile-Driven Prediction**

**Delayed Branch**

# No Branch Prediction

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | | | | | | | | | | | | RAW I1 - I2 F2 |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | E2 | E3 | W | | | | | | | | | | WAR I3-I4 F3 WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | WAR I1-I7 r6 WAR I4-I7 r6, Struct WB |
| 8 | addi $r7, $r7, 4 | F | D s | D s | D s | D s | D | IS s | IS | E1 | W | | | WAR I3-I8 r7, WAR I5-I8 r7, Struct WB |
| 9 | blt $r7, $r8, FOR | | F s | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | RAW R7 I8-I9 |
| 10 | (Following Instruction) | | | | | | | F s | F s | F s | F s | F | D | CTRL |

## Recall: Static Branch Prediction Techniques

**Branch Always Not Taken (Predicted-Not-Taken)**

**Branch Always Taken (Predicted-Taken)**

**Backward Taken Forward Not Taken (BTFNT)**

**Profile-Driven Prediction**

**Delayed Branch**

# Static Branch Prediction: NT

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | | | | | | | | | | | | RAW I1 - I2 F2 |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | E2 | E3 | W | | | | | | | | | | WAR I3-I4 F3 WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | WAR I1-I7 r6 WAR I4-I7 r6, Struct WB |
| 8 | addi $r7, $r7, 4 | F | D s | D s | D s | D s | D | IS s | IS | E1 | W | | | WAR I3-I8 r7, WAR I5-I8 r7, Struct WB |
| 9 | blt $r7, $r8, FOR | | F s | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | RAW R7 I8-I9 |
| 10 | **(Following Instruction)** | | | | | | | F | D | IS s | IS s | IS flush? | E1 flush ? | CNTRL |

## Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

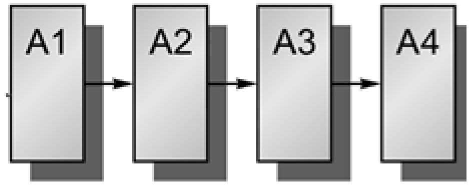
**Branch Always Taken (Predicted-Taken)**

Backward Taken Forward Not Taken (BTFNT)

Profile-Driven Prediction

Delayed Branch

## Recall: You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage
- The target address for a branch is available in the FETCH stage

POLITECNICO MILANO NECST laboratory

POLITECNICO MILANO 1863

# Static Branch Prediction: T

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | FOR: ld $f2,VB($r6) | | | | | | | | | | | | | |
| 2 | fadd $f3, $f2, $f6 | | | | | | | | | | | | | RAW I1 - I2 F2 |
| 3 | st $f3, VA($r7) | | | | | | | | | | | | | RAW I2 - I3 F3 |
| 4 | ld $f3, VC($r6) | E2 | E3 | W | | | | | | | | | | WAR I3-I4 F3<br>WAW I2-I4 F3 |
| 5 | st $f3, VC($r7) | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | | RAW I4 - I5 F3 |
| 6 | fadd $f4,$f4,$f3 | ISs | ISs | ISs | IS | E1 | E2 | E3 | W | | | | | RAW I4 - I6 F3 |
| 7 | addi $r6, $r6, 4 | Ds | D | ISs | ISs | IS | E1 | E1s | E1s | W | | | | WAR I1-I7 r6 WAR I4-I7 r6, Struct WB |
| 8 | addi $r7, $r7, 4 | F | Ds | Ds | Ds | Ds | D | ISs | IS | E1 | W | | | WAR I3-I8 r7, WAR I5-I8 r7, Struct WB |
| 9 | blt $r7, $r8, FOR | | Fs | Fs | Fs | Fs | F | D | ISs | ISs | IS | E1 | W | RAW R7 I8-I9 |
| 10 | **( I1 rexec)** | | | | | | | F | D | ISs | ISs | IS flush? | E1 flush? | CNTRL,<br>RAW I7 - I10 R6 |

## Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

Branch Always Taken (Predicted-Taken)

**Backward Taken Forward Not Taken (BTFNT)**

Profile-Driven Prediction

Delayed Branch

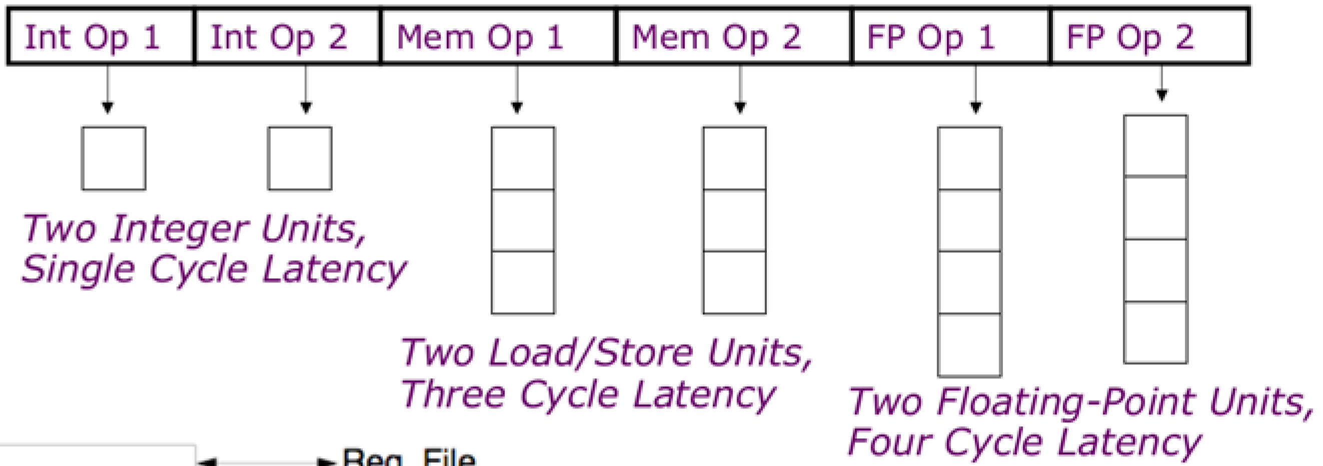
# Static Branch Prediction: BTFNT

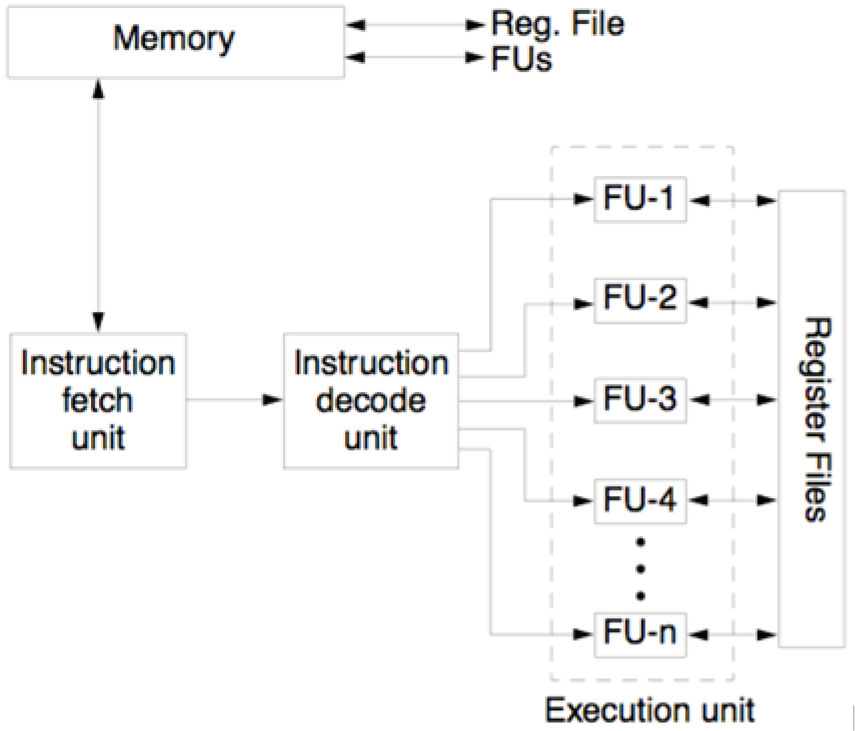
ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

CC 28

| | Instruction | C17 | C18 | C19 | C20 | C21 | C22 | C23 | C24 | C25 | C26 | C27 | C28 | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | FOR: ld $f2,VB($r6) | | | | | | | | | | | | | |
| **2** | fadd $f3, $f2, $f6 | | | | | | | | | | | | | RAW I1 - I2 F2 |
| **3** | st $f3, VA($r7) | | | | | | | | | | | | | RAW I2 - I3 F3 |
| **4** | ld $f3, VC($r6) | E2 | E3 | W | | | | | | | | | | WAR I3-I4 F3 WAW I2-I4 F3 |
| **5** | st $f3, VC($r7) | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | | RAW I4 - I5 F3 |
| **6** | fadd $f4,$f4,$f3 | IS s | IS s | IS s | IS | E1 | E2 | E3 | W | | | | | RAW I4 - I6 F3 |
| **7** | addi $r6, $r6, 4 | D s | D | IS s | IS s | IS | E1 | E1 s | E1 s | W | | | | WAR I1-I7 r6 WAR I4-I7 r6, Struct WB |
| **8** | addi $r7, $r7, 4 | F | D s | D s | D s | D s | D | IS s | IS | E1 | W | | | WAR I3-I8 r7, WAR I5-I8 r7, Struct WB |
| **9** | blt $r7, $r8, FOR | | F s | F s | F s | F s | F | D | IS s | IS s | IS | E1 | W | RAW R7 I8-I9 |
| **10** | **( I1 rexec)** | | | | | | | F | D | IS s | IS s | IS flush? | E1 flush ? | CNTRL, RAW I7 - I10 R6 |

# Recall VLIW and Static Scheduling


Int Op 1
Int Op 2
Mem Op 1
Mem Op 2
FP Op 1
FP Op 2
Two Integer Units, Single Cycle Latency
Two Load/Store Units, Three Cycle Latency
Two Floating-Point Units, Four Cycle Latency



**Static Scheduling**: Rely on software for identifying potential parallelism (example of List Based Scheduling with ASAP)

VLIW (Very Long Instruction Word) processors expect dependency-free code.

# Exe 2 VLIW: schedule

# Exe 2 VLIW: Architecture

- Consider the program be executed on a **3-issue** VLIW MIPS (Very Long Instruction Word) architecture with **3 fully pipelined functional** units

- **Integer ALU** with **1** cycle latency to **next Integer/FP** and **2** cycle latency to **next Branch**

- **Memory** Unit with **3** cycle latency

- **Floating** Point Unit with **3** cycle latency (each **FPU** can complete one add or one multiply per clock cycle)

- Branch completed with **1 cycle delay slot** (branch solved in ID stage)

## Recall Delayed Branch
### (Static Branch Prediction Techniques )

- The job of the compiler is to make the instruction placed in the branch delay slot valid and useful.
- There are three ways in which the branch delay slot can be scheduled:

1. From before
2. From target
3. From fall-through

## Exe 2 VLIW: schedule

- Considering **one iteration** of the loop
- schedule the assembly code for the 3-issue VLIW machine in the following table by using the **list-based scheduling**
- **Do not** use neither software pipelining nor loop unrolling nor modifying loop indexes.
- Please do not need to write in NOPs (can leave blank).

# Exe 2 VLIW: the code

**Assembly Code:**

```
FOR:  ld     $f2, VB($r6)
      fadd   $f3, $f2, $f6
      st     $f3, VA($r7)
      ld     $f3, VC($r6)
      st     $f3, VC($r7)
      fadd   $f4,$f4,$f3
      addi   $r6, $r6, 4
      addi   $r7, $r7, 4
      blt    $r7, $r8, FOR
```

## Recall: Conflicts

**Assembly Code:**

```
I1: FOR: ld $f2, VB($r6)
I2: fadd $f3, $f2, $f6
I3: st $f3, VA($r7)
I4: ld $f3, VC($r6)
I5: st $f3, VC($r7)
I6: fadd $f4,$f4,$f3
I7: addi $r6, $r6, 4
I8: addi $r7, $r7, 4
I9: blt $r7, $r8, FOR
```

**RAW F2 I1–I2**
**RAW F3 I2–I3**
**RAW F3 I4–I5**
**RAW F3 I4–I6**
**RAW R7 I8–I9**
**WAR R7 I8–I5**
**WAR R7 I8–I3**
**WAR R6 I7–I1**
**WAR R6 I7–I4**
**WAW F3 I2–I4**
**WAR F3 I3–I4**
**CNTRL**

## Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

**ALU** 1 cc Integer, 2 cc Branch

**MU** 3 cc

**FPU** 3 cc

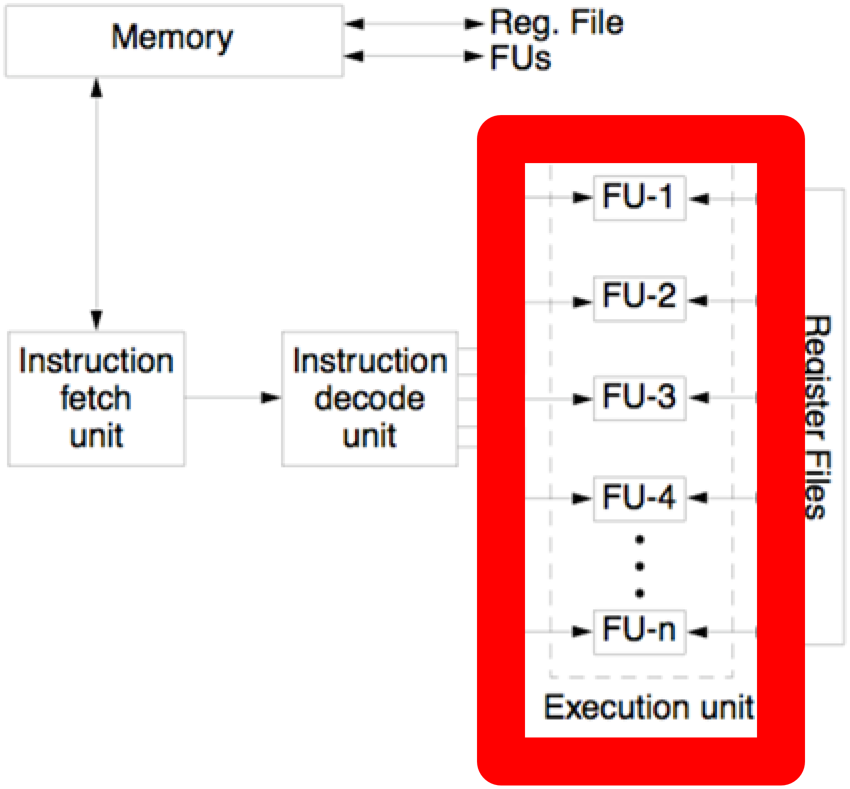
# Exe 2 VLIW: schedule

```
FOR: ld     $f2, VB($r6)
     fadd   $f3, $f2, $f6
     st     $f3, VA($r7)
     ld     $f3, VC($r6)
     st     $f3, VC($r7)
     fadd   $f4,$f4,$f3
     addi   $r6, $r6, 4
     addi   $r7, $r7, 4
     blt    $r7, $r8, FOR
```

**ALU** 1 cc Integer, 2 cc Branch

**MU** 3 cc

**FPU** 3 cc



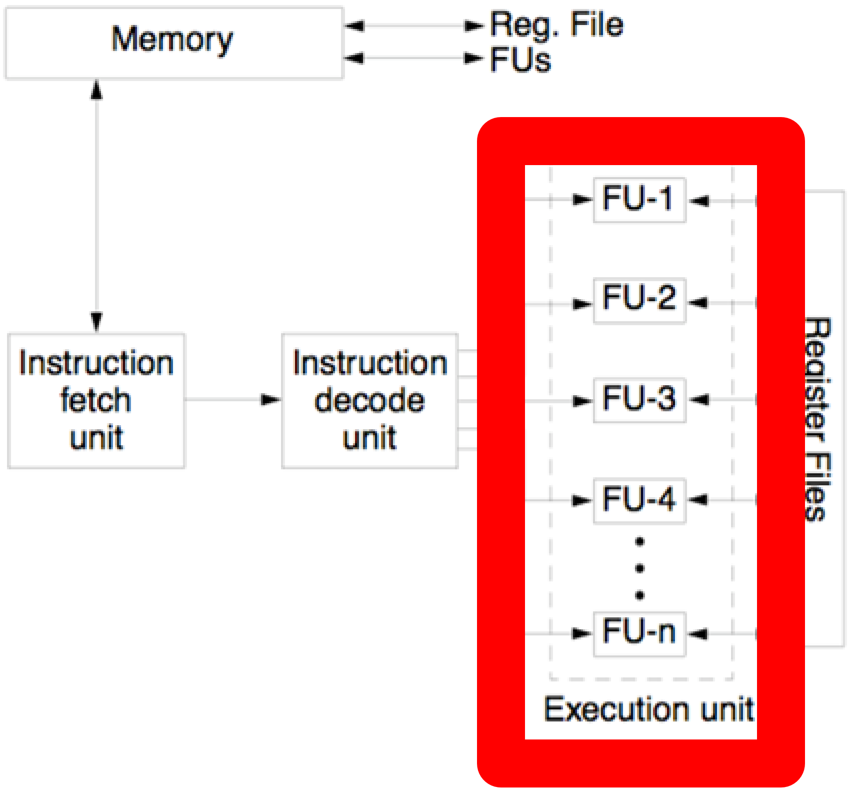
# Exe 2 VLIW: schedule

```
FOR: ld     $f2, VB($r6)
     fadd   $f3, $f2, $f6
     st     $f3, VA($r7)
     ld     $f3, VC($r6)
     st     $f3, VC($r7)
     fadd   $f4,$f4,$f3
     addi   $r6, $r6, 4
     addi   $r7, $r7, 4
     blt    $r7, $r8, FOR
```

**ALU** 1 cc Integer, 2 cc Branch

**MU** 3 cc

**FPU** 3 cc



- Schedule the instructions
- Calculate the FP ops / cycle

## Exe 2 VLIW: schedule

```
FOR:  ld    $f2, VB($r6)
      fadd  $f3, $f2, $f6
      st    $f3, VA($r7)
      ld    $f3, VC($r6)
      st    $f3, VC($r7)
      fadd  $f4,$f4,$f3
      addi  $r6, $r6, 4
      addi  $r7, $r7, 4
      blt   $r7, $r8, FOR
```

|  | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 |  |  |  |
| C2 |  |  |  |
| C3 |  |  |  |
| C4 |  |  |  |
| C5 |  |  |  |
| C6 |  |  |  |
| C7 |  |  |  |
| C8 |  |  |  |
| C9 |  |  |  |
| C10 |  |  |  |
| C11 |  |  |  |
| C12 |  |  |  |
| C13 |  |  |  |
| C14 |  |  |  |
| C15 |  |  |  |

# Exe 2 VLIW: schedule

```
ld    $f2, VB($r6)
fadd  $f3, $f2, $f6
st    $f3, VA($r7)
ld    $f3, VC($r6)
st    $f3, VC($r7)
fadd  $f4,$f4,$f3
addi  $r6, $r6, 4
addi  $r7, $r7, 4
blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | |
| C5 | | | |
| C6 | | | |
| C7 | | | |
| C8 | | | |
| C9 | | | |
| C10 | | | |
| C11 | | | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) 1cc | |
| C2 | | 2cc | |
| C3 | | 3cc | |
| C4 | | | |
| C5 | | | |
| C6 | | | |
| C7 | | | |
| C8 | | | |
| C9 | | | |
| C10 | | | |
| C11 | | | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) 1cc | |
| C2 | | 2cc | |
| C3 | | 3cc | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | | |
| C8 | | | |
| C9 | | | |
| C10 | | | |
| C11 | | | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld   $f2, VB($r6)
     fadd $f3, $f2, $f6
     st   $f3, VA($r7)
     ld   $f3, VC($r6)
     st   $f3, VC($r7)
     fadd $f4,$f4,$f3
     addi $r6, $r6, 4
     addi $r7, $r7, 4
     blt  $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) | |
|---|---|---|---|---|
| C1 | | ld $f2, VB($r6) | | |
| C2 | | | | |
| C3 | | | | |
| C4 | | | fadd $f3, $f2, $f6 | 1cc |
| C5 | | | | 2cc |
| C6 | | | | 3cc |
| C7 | | | | |
| C8 | | | | |
| C9 | | | | |
| C10 | | | | |
| C11 | | | | |
| C12 | | | | |
| C13 | | | | |
| C14 | | | | |
| C15 | | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) | |
|---|---|---|---|---|
| C1 | | ld $f2, VB($r6) | | |
| C2 | | | | |
| C3 | | | | |
| C4 | | | fadd $f3, $f2, $f6 | 1cc |
| C5 | | | | 2cc |
| C6 | | | | 3cc |
| C7 | | st $f3, VA($r7) | | |
| C8 | | | | |
| C9 | | | | |
| C10 | | | | |
| C11 | | | | |
| C12 | | | | |
| C13 | | | | |
| C14 | | | | |
| C15 | | | | |

POLITECNICO MILANO 1863 NECST laboratory

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | | | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) 1cc | |
| C9 | | 2cc | |
| C10 | | 3cc | |
| C11 | | | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) 1cc | |
| C9 | | 2cc | |
| C10 | | 3cc | |
| C11 | | st $f3, VA($r7) | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld   $f2, VB($r6)
     fadd $f3, $f2, $f6
     st   $f3, VA($r7)
     ld   $f3, VC($r6)
     st   $f3, VC($r7)
     fadd $f4,$f4,$f3
     addi $r6, $r6, 4
     addi $r7, $r7, 4
     blt  $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | | st $f3, VA($r7) | |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld     $f2, VB($r6)
     fadd   $f3, $f2, $f6
     st     $f3, VA($r7)
     ld     $f3, VC($r6)
     st     $f3, VC($r7)
     fadd   $f4,$f4,$f3
     addi   $r6, $r6, 4
     addi   $r7, $r7, 4
     blt    $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld     $f2, VB($r6)
     fadd   $f3, $f2, $f6
     st     $f3, VA($r7)
     ld     $f3, VC($r6)
     st     $f3, VC($r7)
     fadd   $f4,$f4,$f3
     addi   $r6, $r6, 4
     addi   $r7, $r7, 4
     blt    $r7, $r8, FOR
```

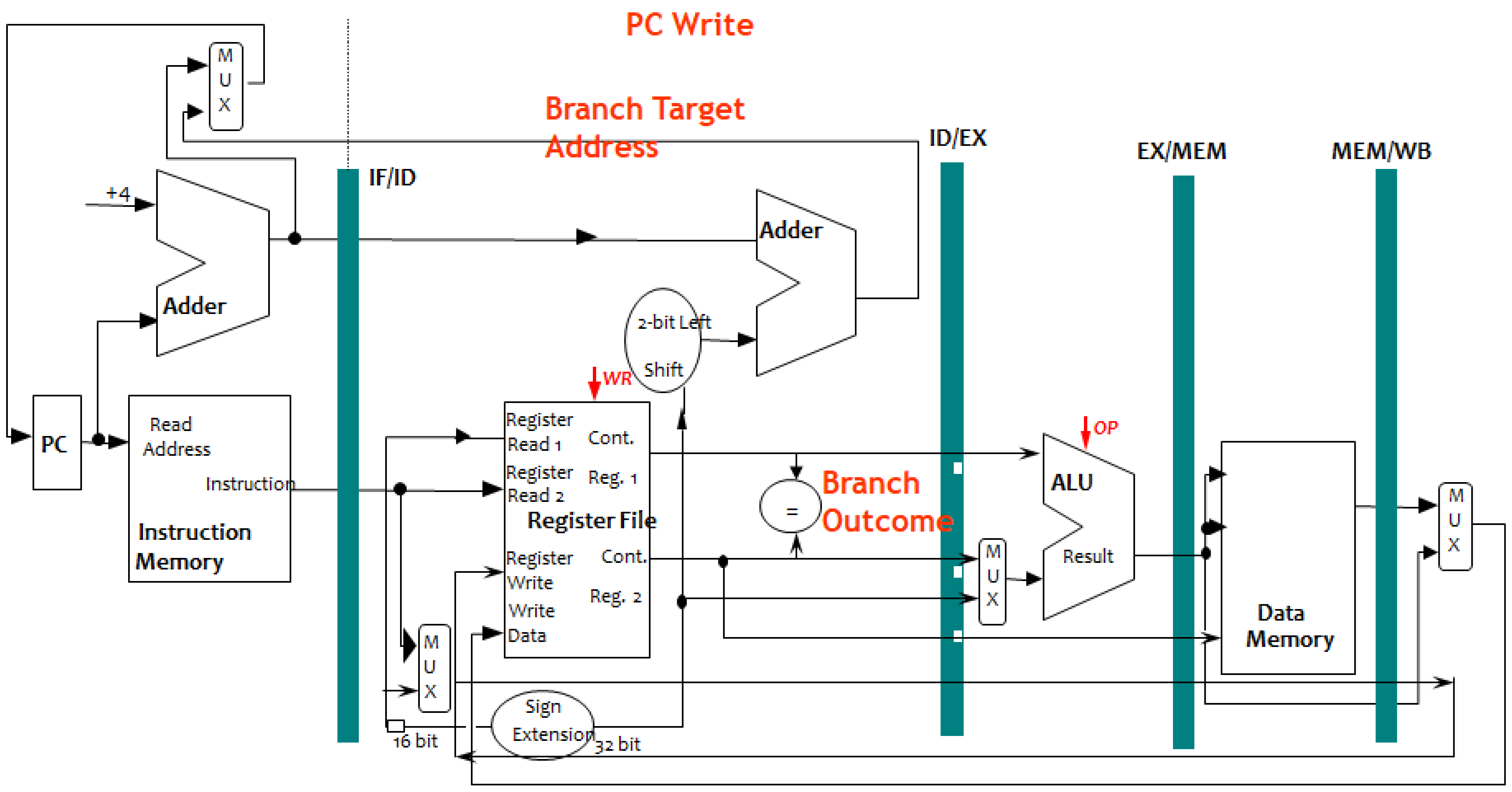
| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | | | |
| C14 | | | |
| C15 | | | |

# Recall: Early-evaluation PC



# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | blt $r7, $r8, FOR | | |
| C14 | | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld    $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st    $f3, VA($r7)
     ld    $f3, VC($r6)
     st    $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | blt $r7, $r8, FOR | | |
| C14 | (br delay slot) | | |
| C15 | | | |

# Exe 2 VLIW: The Resulting Code

```
FOR:  ld    $f2, VB($r6)
      fadd  $f3, $f2, $f6
      st    $f3, VA($r7)
      ld    $f3, VC($r6)
      st    $f3, VC($r7)
      fadd  $f4,$f4,$f3
      addi  $r6, $r6, 4
      addi  $r7, $r7, 4
      blt   $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| I1 | NOP | ld $f2, VB($r6) | NOP |
| I2 | NOP | NOP | NOP |
| I3 | NOP | NOP | NOP |
| I4 | NOP | NOP | fadd $f3, $f2, $f6 |
| I5 | NOP | NOP | NOP |
| I6 | NOP | NOP | NOP |
| I7 | NOP | st $f3, VA($r7) | NOP |
| I8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | NOP |
| I9 | NOP | NOP | NOP |
| I10 | NOP | NOP | NOP |
| I11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| I12 | NOP | NOP | NOP |
| I13 | blt $r7, $r8, FOR | NOP | NOP |
| I14 | (br delay slot) | NOP | NOP |

# Exe 2 VLIW: schedule

```
FOR: ld   $f2, VB($r6)
     fadd $f3, $f2, $f6
     st   $f3, VA($r7)
     ld   $f3, VC($r6)
     st   $f3, VC($r7)
     fadd $f4,$f4,$f3
     addi $r6, $r6, 4
     addi $r7, $r7, 4
     blt  $r7, $r8, FOR
```

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | blt $r7, $r8, FOR | | |
| C14 | (br delay slot) | | |
| C15 | | | |

# Exe 2 VLIW: schedule

```
FOR: ld     $f2, VB($r6)
     fadd  $f3, $f2, $f6
     st     $f3, VA($r7)
     ld     $f3, VC($r6)
     st     $f3, VC($r7)
     fadd  $f4,$f4,$f3
     addi  $r6, $r6, 4
     addi  $r7, $r7, 4
     blt    $r7, $r8, FOR
```

FPops/cycle=

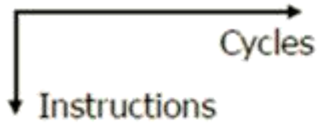
=2 fadds / 14 cycles =

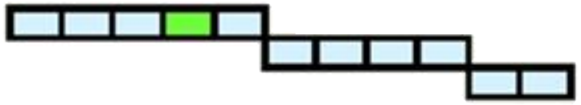
= 0.143

| | Integer ALU(1/2 b) | Memory Unit(3cc) | FPU(3cc) |
|---|---|---|---|
| C1 | | ld $f2, VB($r6) | |
| C2 | | | |
| C3 | | | |
| C4 | | | fadd $f3, $f2, $f6 |
| C5 | | | |
| C6 | | | |
| C7 | | st $f3, VA($r7) | |
| C8 | addi $r6, $r6, 4 | ld $f3, VC($r6) | |
| C9 | | | |
| C10 | | | |
| C11 | addi $r7, $r7, 4 | st $f3, VA($r7) | fadd $f4, $f4 , $f3 |
| C12 | | | |
| C13 | blt $r7, $r8, FOR | | |
| C14 | (br delay slot) | | |
| C15 | | | |

# Recall: The ILP Architecture Journey

Steps towards exploiting more ILP

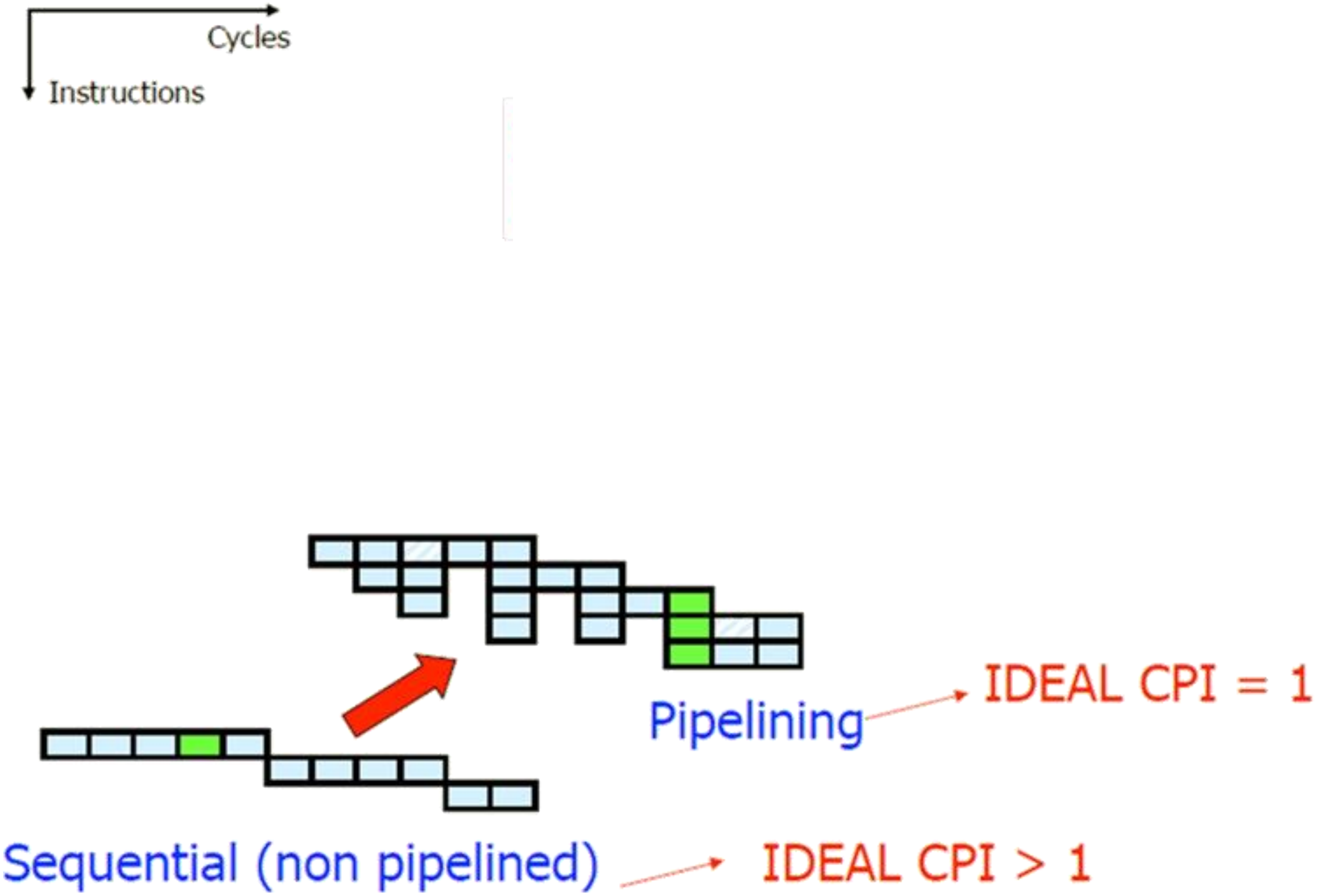




Sequential (non pipelined) $\longrightarrow$ IDEAL CPI > 1

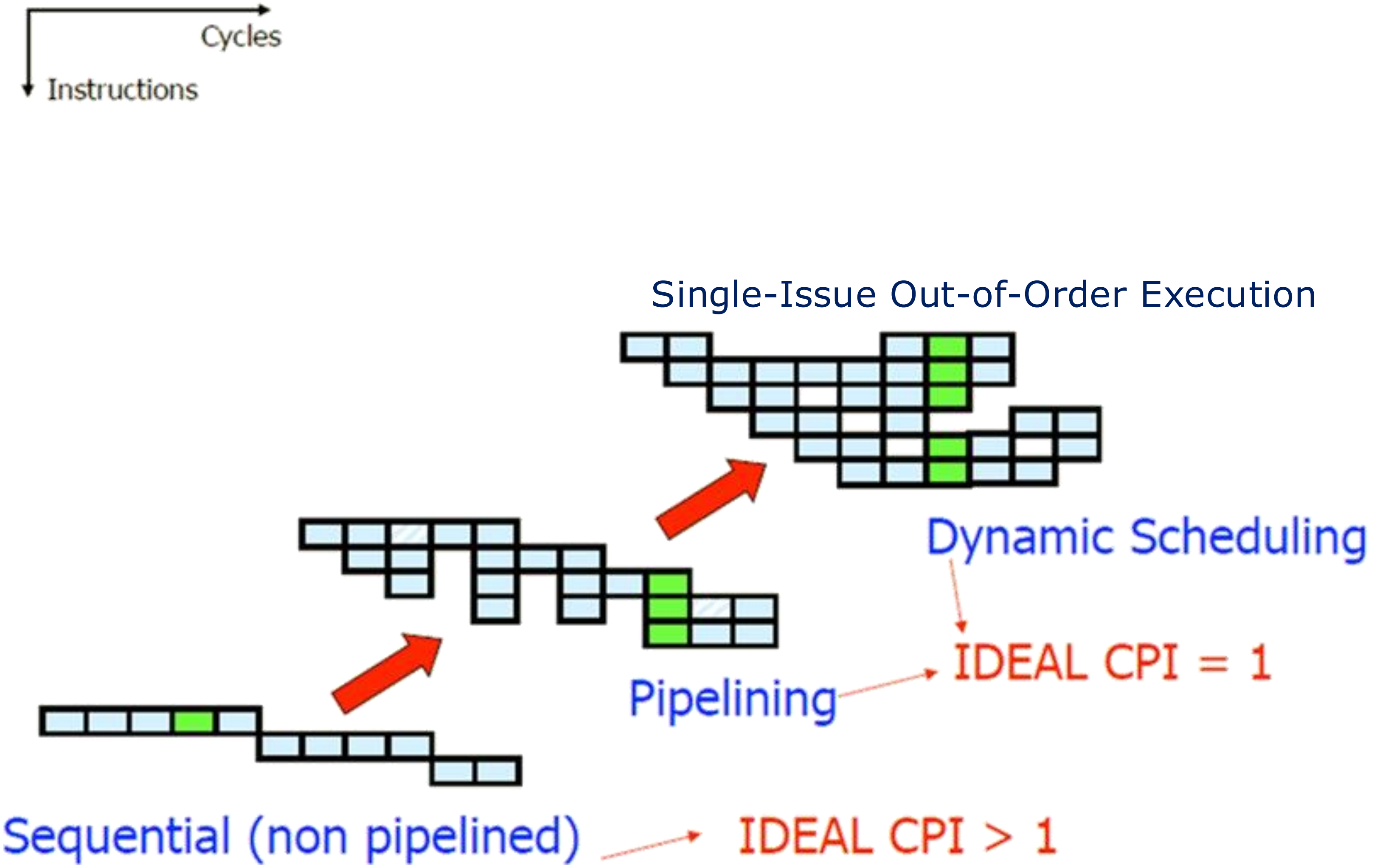
# Recall: The ILP Architecture Journey

## Steps towards exploiting more ILP



# Recall: The ILP Architecture Journey

Steps towards exploiting more ILP



## Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

## Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

- HW rearranges the instruction execution to reduce stalls

## Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

- HW rearranges the instruction execution to reduce stalls

Enables out-of-order execution and completion (commit)

- Out-of order execution introduces possibility of WAR, WAW data hazards.

## Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

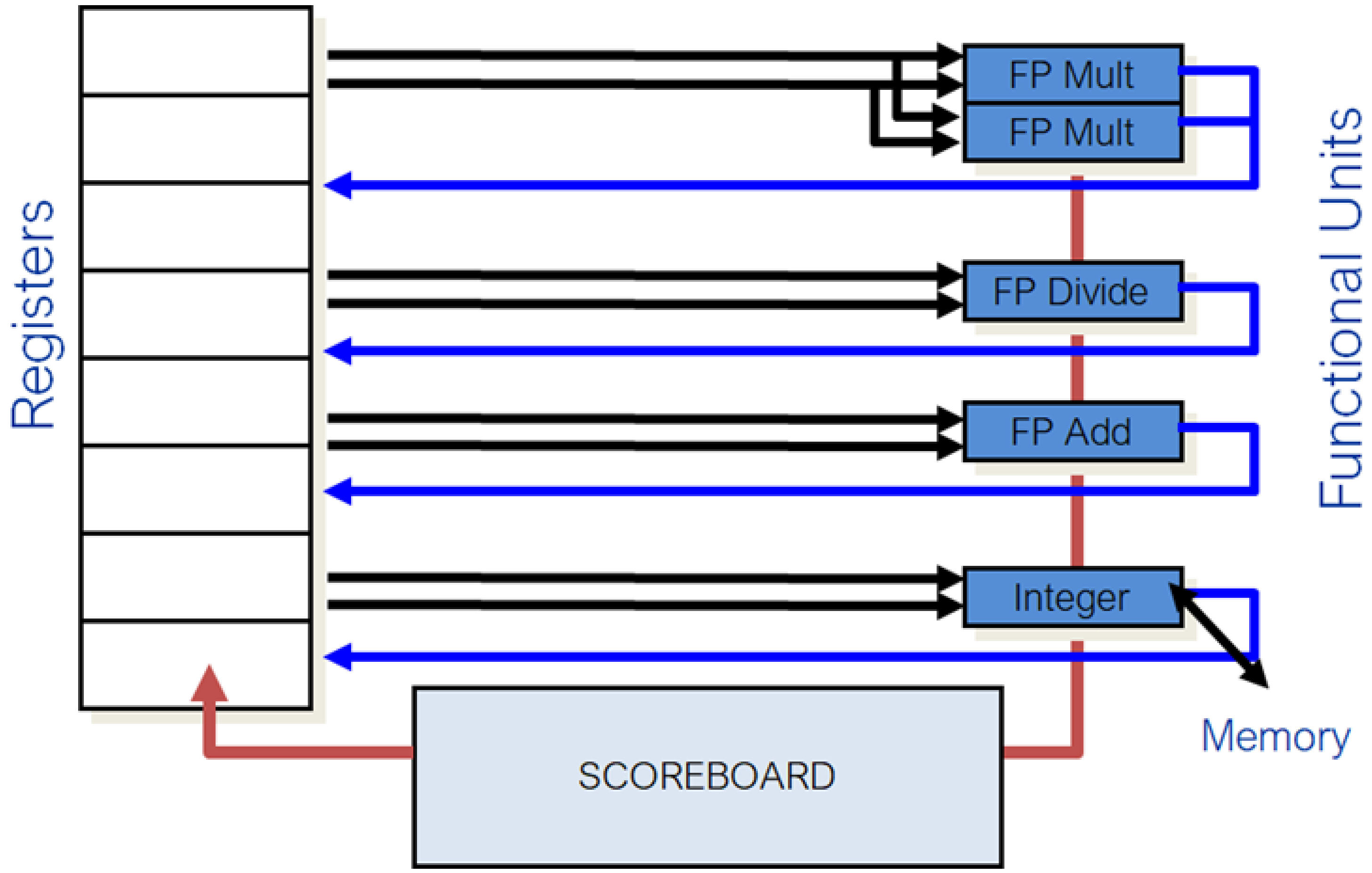
- HW rearranges the instruction execution to reduce stalls

Enables out-of-order execution and completion (commit)

- Out-of order execution introduces possibility of WAR, WAW data hazards.

First implemented in CDC6600 (1963)

# Exe 1 Scoreboard



Parallel operation in the control data 6600

## Recall: the Scoreboard pipeline

| ISSUE | READ OPERAND | EXE COMPLETE | WB |
|---|---|---|---|
| Decode instruction; | Read operands; | Operate on operands; | Finish exec; |
| Structural FUs check; WAW checks | RAW check; WAR if need to read | Notify Scoreboard on completion; | WAR &Struct check (FUs will hold results); Can overlap issue/read&write 4 Structural Hazard; |

# Exe 1 Scoreboard: the Code

```
I1:  LD F6 32+ R2
I2:  ADDD F2 F6 F4
I3:  MULTD F0 F4 F2
I4:  SUBD F12 F2 F6
I5:  ADDD F0 F12 F2
```

# Exe 1.1 Scoreboard: Conflicts

```
I1:  LD F6 32+ R2
I2:  ADDD F2 F6 F4
I3:  MULTD F0 F4 F2
I4:  SUBD F12 F2 F6
I5:  ADDD F0 F12 F2
```

# Exe 1.1 Scoreboard: Conflicts

**RAW F6 I1-I2**

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

**RAW F2 I2-I3**

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

**RAW F2 I2-I3**

**RAW F2 I2-I4**

POLITECNICO MILANO 1863

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

**RAW F2 I2-I3**

**RAW F2 I2-I4**

**RAW F2 I2-I5**

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

**RAW F2 I2-I3**

**RAW F2 I2-I4**

**RAW F2 I2-I5**

**RAW F12 I4-I5**

# Exe 1.1 Scoreboard: Conflicts

```
I1:   LD F6 32+ R2
I2:   ADDD F2 F6 F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

**RAW F6 I1-I2**

**RAW F6 I1-I4**

**RAW F2 I2-I3**

**RAW F2 I2-I4**

**RAW F2 I2-I5**

**RAW F12 I4-I5**

**WAW F0 I3-I5**

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| **I1: LD F6 32+ R2** | 1 | 2 | 7 | 8 |
| **I2: ADDD F2 F6 F4** | 2 | 9 | 11 | 12 |
| **I3: MULTD F0 F4 F2** | 4 | 13 | 43 | 44 |
| **I4: SUBD F12 F2 F6** | 3 | 9 | 11 | 12 |
| **I5: ADDD F0 F12 F2** | 13 | 17 | 19 | 20 |

- Is there a "configuration" that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a "configuration" that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a "configuration" that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a "configuration" that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.2 Scoreboard: ∃ a configuration?

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| I1: LD F6 32+ R2 | 1 | 2 | 7 | 8 |
| I2: ADDD F2 F6 F4 | 2 | 9 | 11 | 12 |
| I3: MULTD F0 F4 F2 | 4 | 13 | 43 | 44 |
| I4: SUBD F12 F2 F6 | 3 | 9 | 11 | 12 |
| I5: ADDD F0 F12 F2 | 13 | 17 | 19 | 20 |

- Is there a "configuration" that can respect the shown execution?
- How many units? Which kind? What latency?

# Exe 1.3 Scoreboard: if not correct, write right one

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | | | | | | |
| I2 | ADDD F2 F6 F4 | | | | | | |
| I3 | MULTD F0 F4 F2 | | | | | | |
| I4 | SUBD F12 F2 F6 | | | | | | |
| I5 | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

# Exe 1.3 Scoreboard: if not correct, write right one

CC 0

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | | | | | | |
| I2 | ADDD F2 F6 F4 | | | | | | |
| I3 | MULTD F0 F4 F2 | | | | | | |
| I4 | SUBD F12 F2 F6 | | | | | | |
| I5 | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one

CC 1

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| **I1** | LD F6 32+ R2 | **1** | | | | | **MU** |
| **I2** | ADDD F2 F6 F4 | | | | | | |
| **I3** | MULTD F0 F4 F2 | | | | | | |
| **I4** | SUBD F12 F2 F6 | | | | | | |
| **I5** | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one CC 2

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| **I1** | LD F6 32+ R2 | **1** | **2** | | | | **MU** |
| **I2** | ADDD F2 F6 F4 | **2** | | | | | **FPU1** |
| **I3** | MULTD F0 F4 F2 | | | | | | |
| **I4** | SUBD F12 F2 F6 | | | | | | |
| **I5** | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, <u>single write</u> port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one

CC 3

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | | | | MU |
| I2 | ADDD F2 F6 F4 | 2 | | | | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | | FPU2 |
| I4 | SUBD F12 F2 F6 | | | | | | |
| I5 | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, <u>single write</u> port for the pool**
**1 MEM 2 cc latency**

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

POLITECNICO MILANO 1863

# Exe 1.3 Scoreboard: if not correct, write right one
CC 4

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| **I1** | LD F6 32+ R2 | **1** | **2** | **4** | | | **MU** |
| **I2** | ADDD F2 F6 F4 | **2** | | | | **RAW F6** | **FPU1** |
| **I3** | MULTD F0 F4 F2 | **3** | | | | **RAW F2** | **FPU2** |
| **I4** | SUBD F12 F2 F6 | **4** | | | | | **FPU3** |
| **I5** | ADDD F0 F12 F2 | | | | | | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one
## CC 5

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | | | | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | | | | | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, <u>single write</u> port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one CC 5

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | | | | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | | | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

**RAW F6 I1-I2**
**RAW F6 I1-I4**
**RAW F2 I2-I3**
**RAW F2 I2-I4**
**RAW F2 I2-I5**
**RAW F12 I4-I5**
**WAW F0 I3-I5**

# Exe 1.3 Scoreboard: if not correct, write right one CC 6

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | | | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | | | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

# Exe 1.3 Scoreboard: if not correct, write right one CC 9

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | | | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1–I2~~
~~RAW F6 I1–I4~~
RAW F2 I2–I3
RAW F2 I2–I4
RAW F2 I2–I5
RAW F12 I4–I5
WAW F0 I3–I5

# Exe 1.3 Scoreboard: if not correct, write right one CC 10

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | | | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | | | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

# Exe 1.3 Scoreboard: if not correct, write right one CC 11

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| **I1** | LD F6 32+ R2 | **1** | **2** | **4** | **5** | | **MU** |
| **I2** | ADDD F2 F6 F4 | **2** | **6** | **9** | **10** | **RAW F6** | **FPU1** |
| **I3** | MULTD F0 F4 F2 | **3** | **11** | | | **RAW F2** | **FPU2** |
| **I4** | SUBD F12 F2 F6 | **4** | **11** | | | **RAW F2** | **FPU3** |
| **I5** | ADDD F0 F12 F2 | | | | | **WAW F0** | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

# Exe 1.3 Scoreboard: if not correct, write right one
CC 14

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

POLITECNICO MILANO 1863

# Exe 1.3 Scoreboard: if not correct, write right one
CC 14

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | | RAW F2 | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

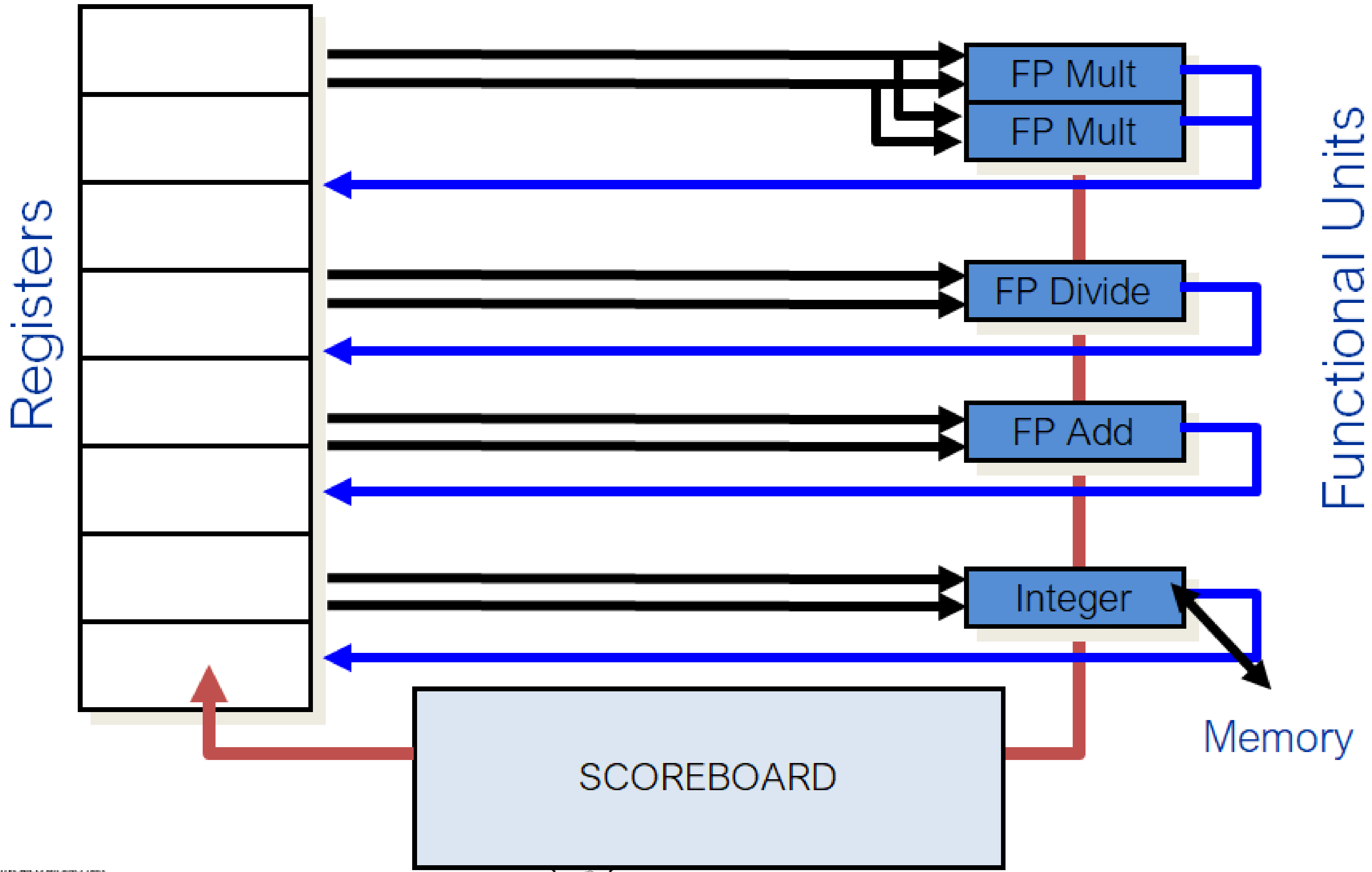
SINGLE WRITE PORT?!?!?!!?!?

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

# Exe 1.3 Scoreboard: if not correct, write right one



# Exe 1.3 Scoreboard: if not correct, write right one CC 15

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | 15 | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | | RAW F2 + Struct RF | FPU3 |
| I5 | ADDD F0 F12 F2 | | | | | WAW F0 | |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, <u>single write </u>port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

# Exe 1.3 Scoreboard: if not correct, write right one CC 16

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | 15 | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | 16 | RAW F2 + Struct RF | FPU3 |
| I5 | ADDD F0 F12 F2 | 16 | | | | WAW F0 | FPU4 |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1–I2~~
~~RAW F6 I1–I4~~
~~RAW F2 I2–I3~~
~~RAW F2 I2–I4~~
~~RAW F2 I2–I5~~
~~RAW F12 I4–I5~~
~~WAW F0 I3–I5~~

# Exe 1.3 Scoreboard: if not correct, write right one

CC 17

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | 15 | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | 16 | RAW F2 + Struct RF | FPU3 |
| I5 | ADDD F0 F12 F2 | 16 | 17 | | | WAW F0 | FPU4 |

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1–I2~~
~~RAW F6 I1–I4~~
~~RAW F2 I2–I3~~
~~RAW F2 I2–I4~~
~~RAW F2 I2–I5~~
~~RAW F12 I4–I5~~
~~WAW F0 I3–I5~~

# Exe 1.3 Scoreboard: if not correct, write right one
CC 21

| | Instruction | ISSUE | READ OPERAND | EXE COMPLETE | WB | Hazards | Unit |
|---|---|---|---|---|---|---|---|
| I1 | LD F6 32+ R2 | 1 | 2 | 4 | 5 | | MU |
| I2 | ADDD F2 F6 F4 | 2 | 6 | 9 | 10 | RAW F6 | FPU1 |
| I3 | MULTD F0 F4 F2 | 3 | 11 | 14 | 15 | RAW F2 | FPU2 |
| I4 | SUBD F12 F2 F6 | 4 | 11 | 14 | 16 | RAW F2 + Struct RF | FPU3 |
| I5 | ADDD F0 F12 F2 | 16 | 17 | 20 | 21 | WAW F0 | FPU4 |

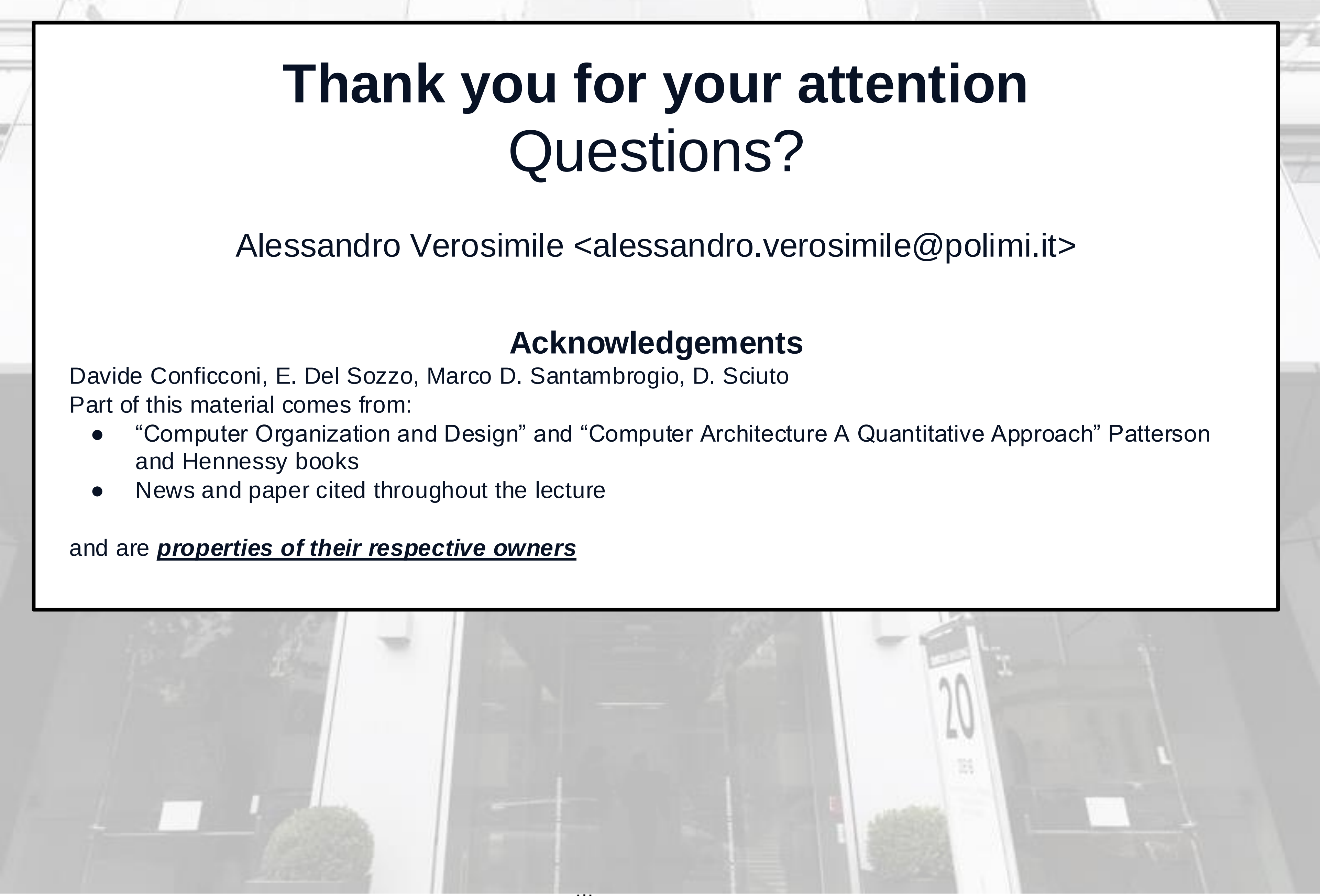
If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

**4 FPALU 3 cc latency, single write port for the pool**
**1 MEM 2 cc latency**

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
~~RAW F12 I4-I5~~
~~WAW F0 I3-I5~~

POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863



# Thank you for your attention
## Questions?

Alessandro Verosimile <alessandro.verosimile@polimi.it>

**Acknowledgements**

Davide Conficconi, E. Del Sozzo, Marco D. Santambrogio, D. Sciuto
Part of this material comes from:

- “Computer Organization and Design” and “Computer Architecture A Quantitative Approach” Patterson and Hennessy books
- News and paper cited throughout the lecture

and are ***<u>properties of their respective owners</u>***