

PIPELINING: BRANCH AND MULTICYCLE INSTRUCTIONS

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

Overview

- Announcement

 - ▣ Homework 1 submission deadline: Jan. 30th

- This lecture

 - ▣ Control hazards in the five-stage pipeline

 - ▣ Multicycle instructions

 - Pipelined

 - Unpipelined

 - ▣ Reorder buffer

Control Hazards

- Example C/C++ code

```
for (i=100; i > 0; i--) {  
    sum = sum + i;  
}  
total = total + sum;
```

How many branches are in this code?

Control Hazards

□ Example C/C++ code

```
for (i=100; i > 0; i--) {  
    sum = sum + i;  
}  
total = total + sum;
```

add r1, r0, #100

for:

beq r0, r1, next

add r2, r2, r1

sub r1, r1, #1

J for

next:

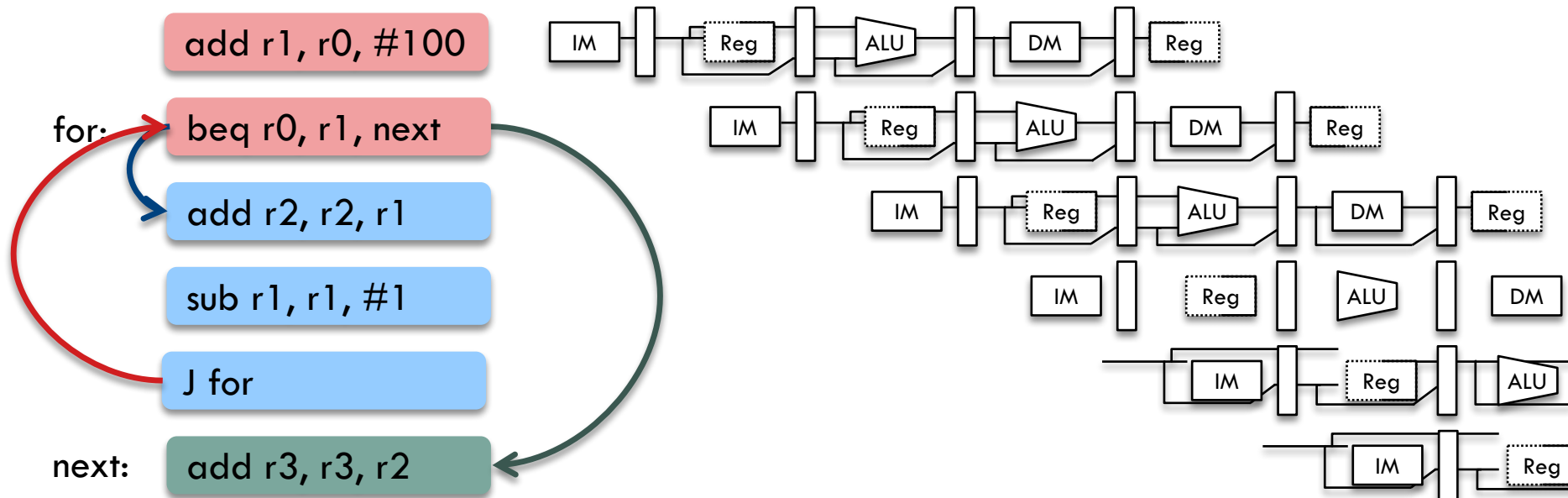
add r3, r3, r2

What are possible target instructions?

Control Hazards

□ Example C/C++ code

```
for (i=100; i > 0; i--) {  
    sum = sum + i;  
}  
total = total + sum;
```



What happens inside the pipeline?

Handling Control Hazards

- 1. introducing stall cycles and delay slots
- How many cycles/slots?
- One branch per every six instructions on average!!

for:

add r1, r0, #100

beq r0, r1, next

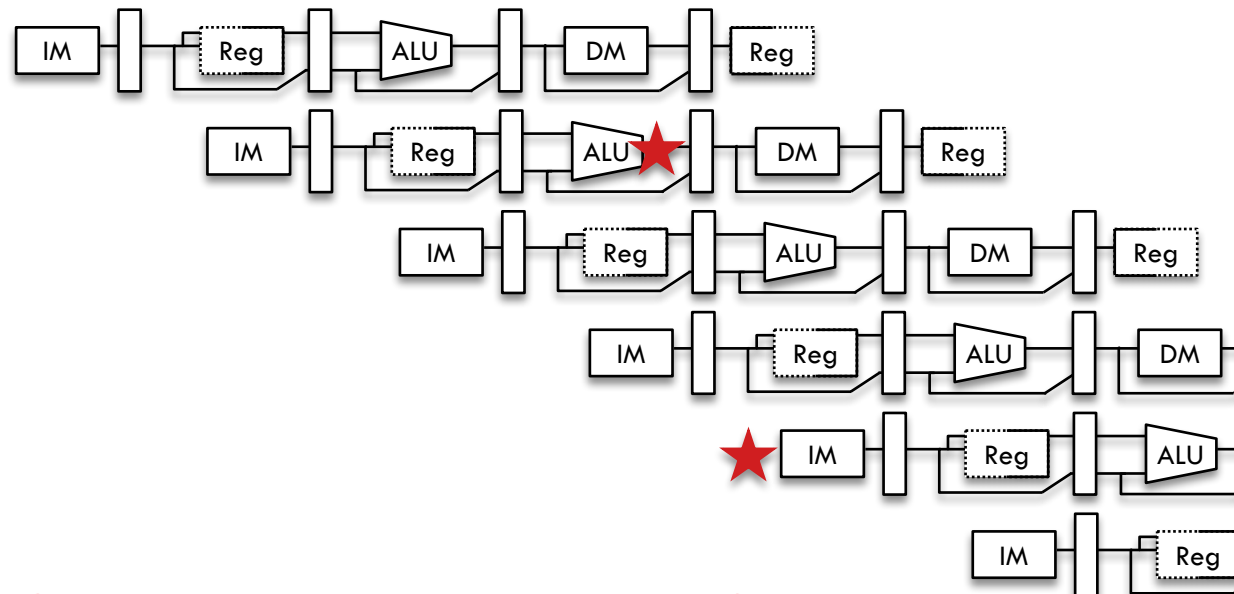
nothing

nothing

add r2, r2, r1

sub r1, r1, #1

J for



2 additional delay slots per 6 cycles!

Handling Control Hazards

- 1. introducing stall cycles and delay slots
 - How many cycles/slots?
 - One branch per every six instructions on average!!

for:

add r1, r0, #100

beq r0, r1, next

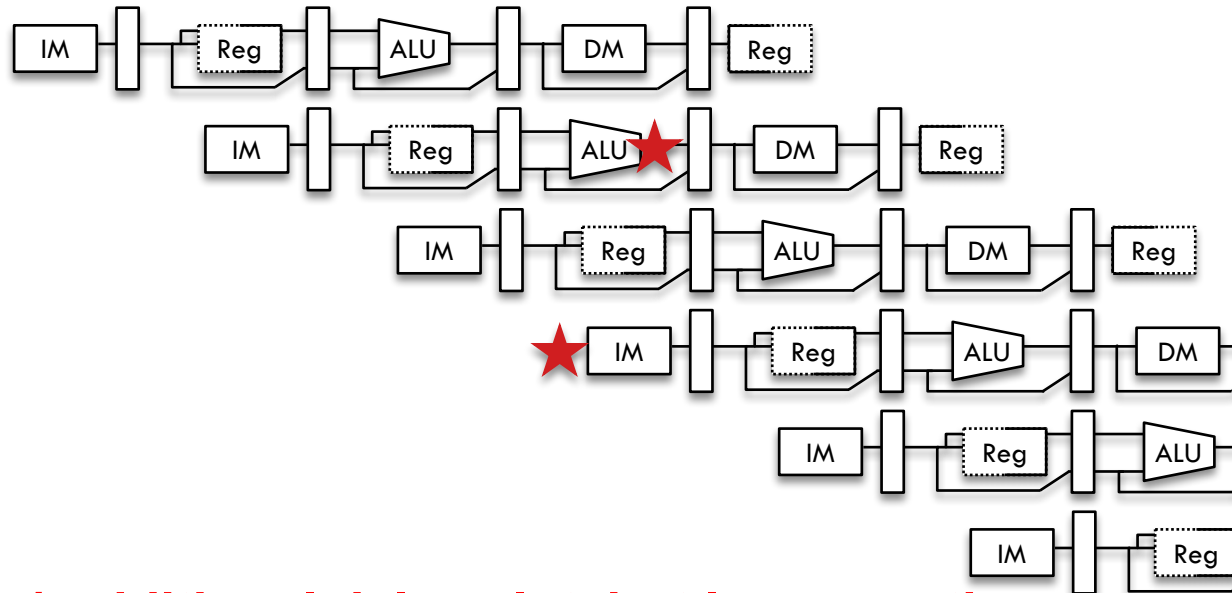
nothing

add r2, r2, r1

sub r1, r1, #1

J for

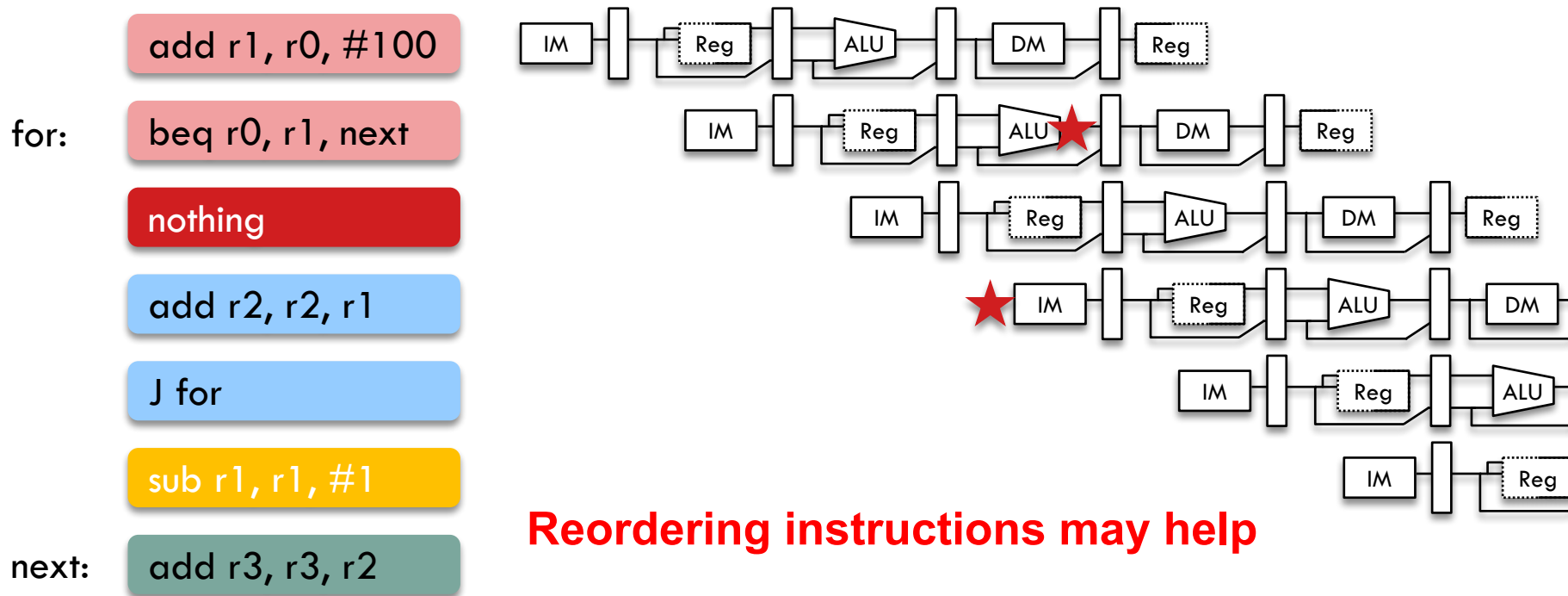
nothing



1 additional delay slot, but longer path

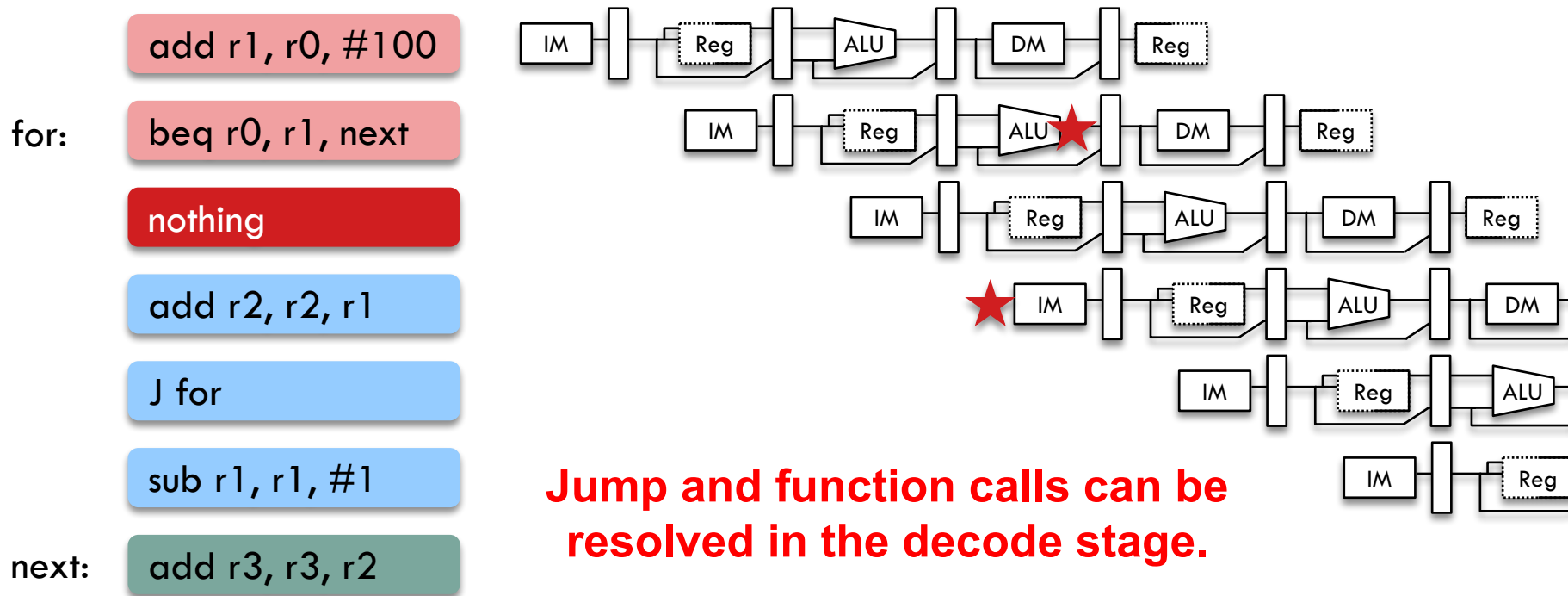
Handling Control Hazards

- 1. introducing stall cycles and delay slots
 - How many cycles/slots?
 - One branch per every six instructions on average!!



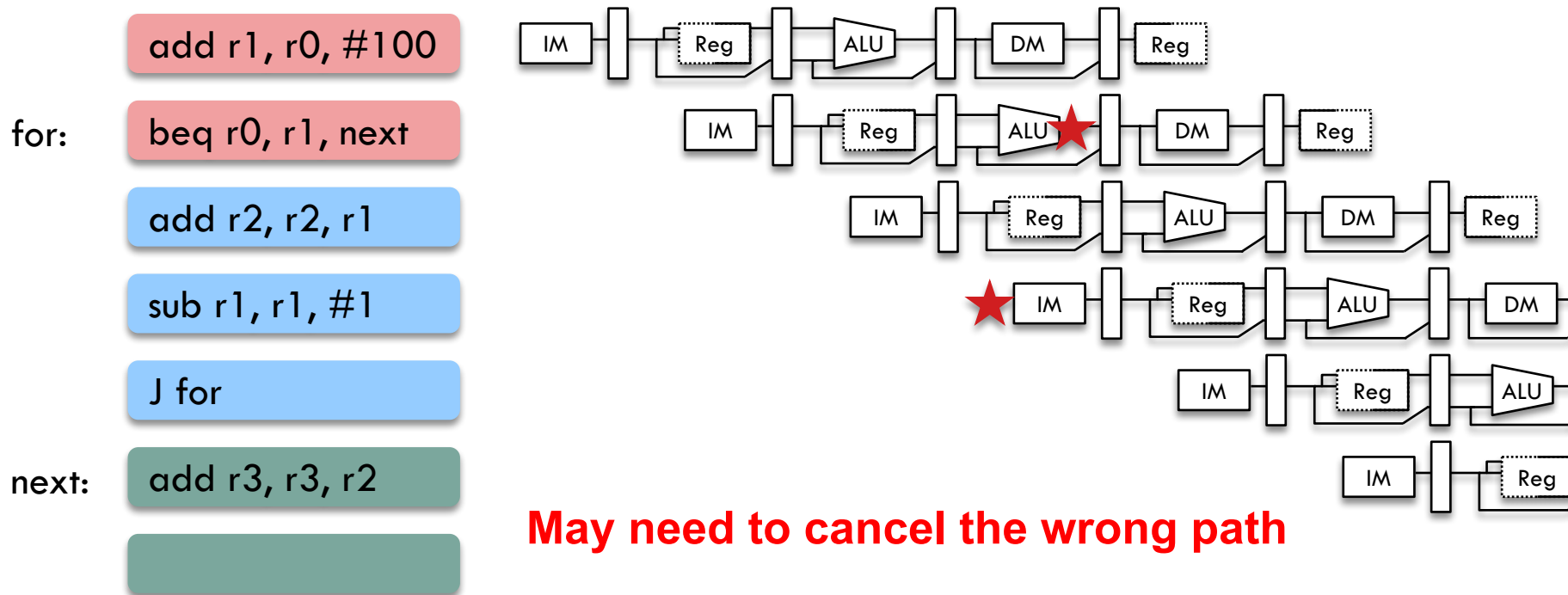
Handling Control Hazards

- 1. introducing stall cycles and delay slots
 - How many cycles/slots?
 - One branch per every six instructions on average!!



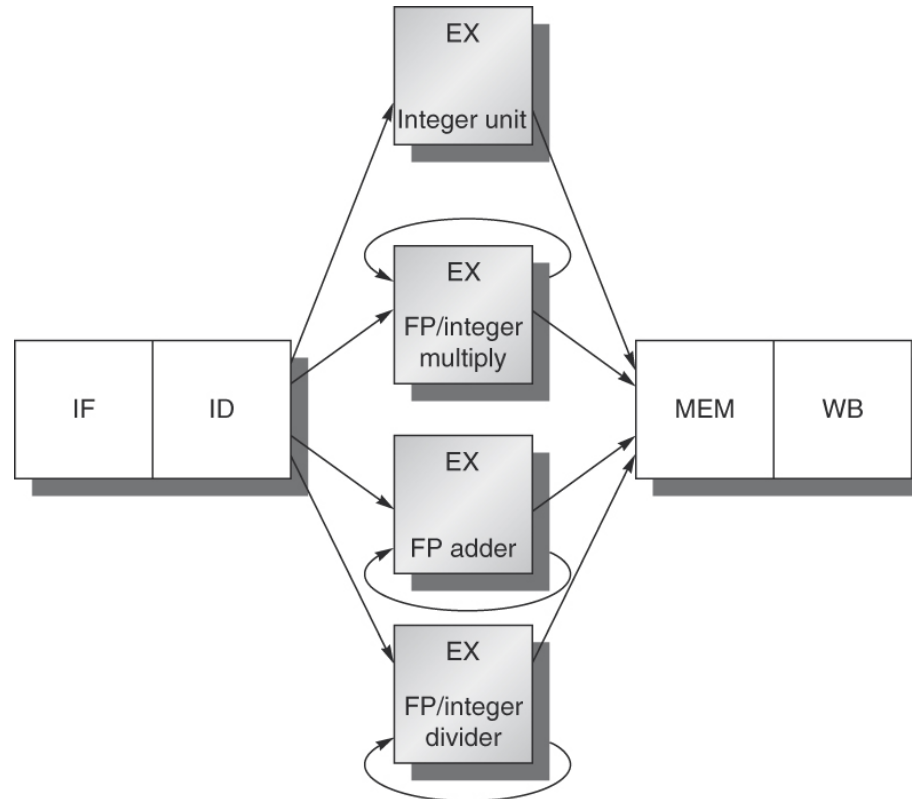
Handling Control Hazards

- 1. introducing stall cycles and delay slots
- 2. predict the branch outcome
 - simply assume the branch is taken or not taken
 - predict the next PC



Multicycle Instructions

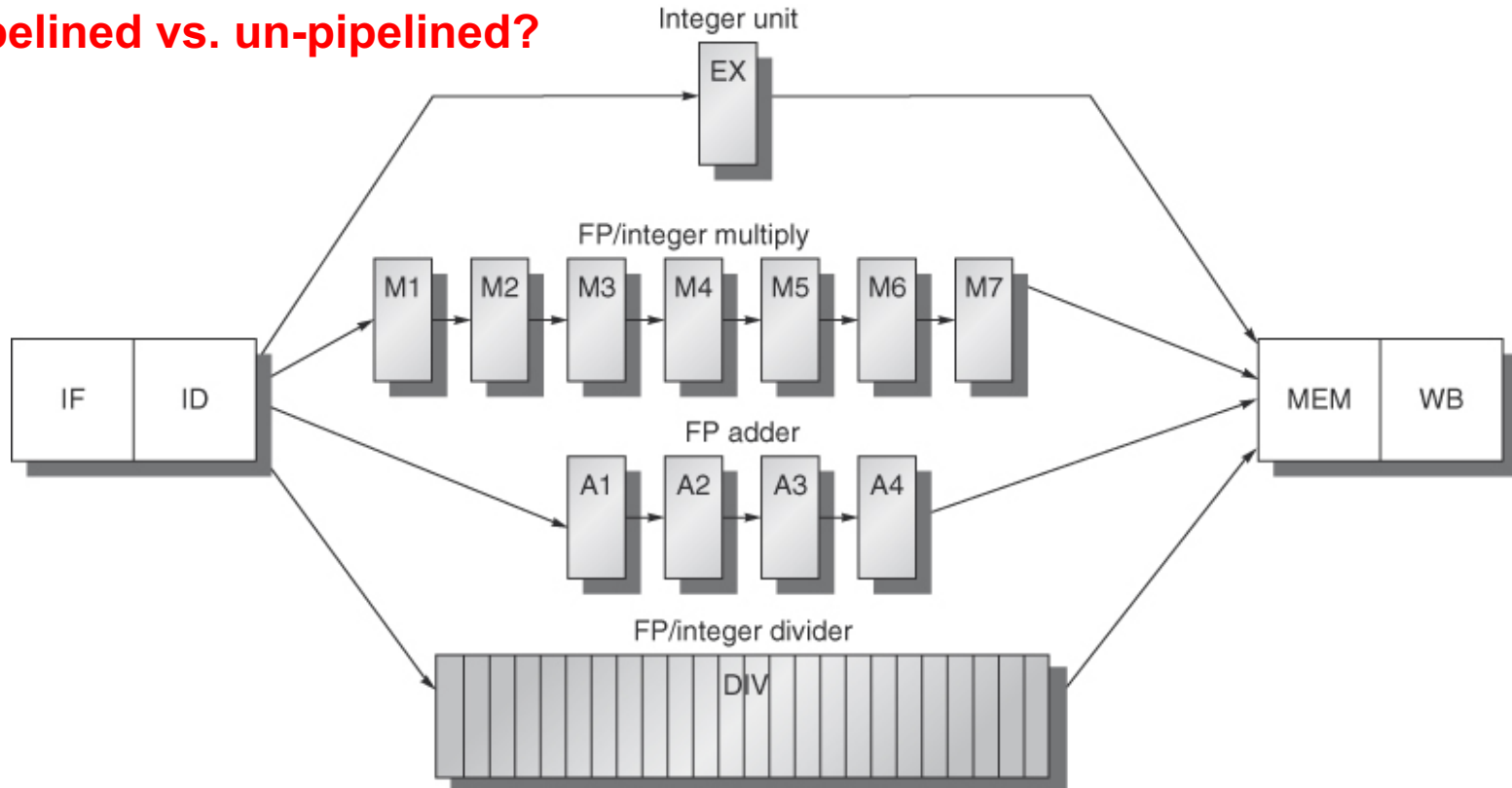
- Not all of the ALU operations complete in one cycle
 - ▣ Typically, FP operations need more time



Multicycle Instructions

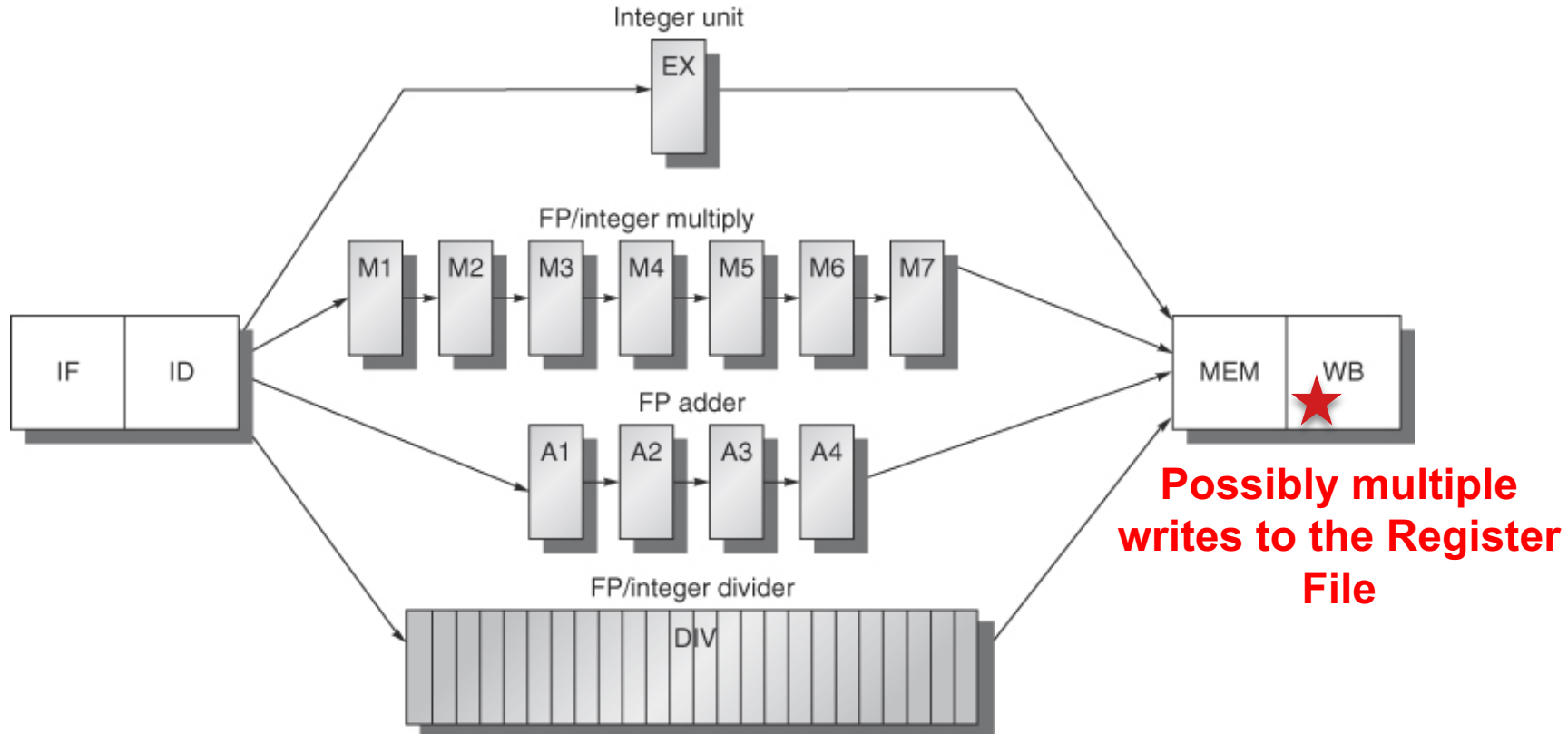
- Not all of the ALU operations complete in one cycle
 - ▣ pipelined and un-pipelined multicycle functional units

Pipelined vs. un-pipelined?



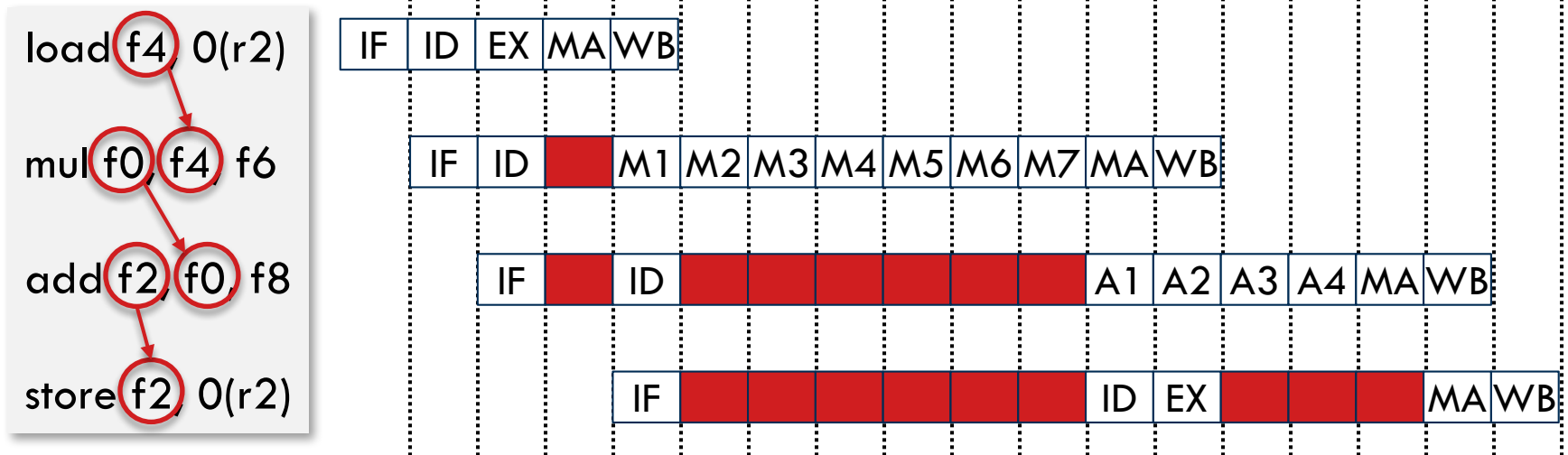
Multicycle Instructions

- Structural hazards
 - ▣ potentially multiple RF writes



Multicycle Instructions

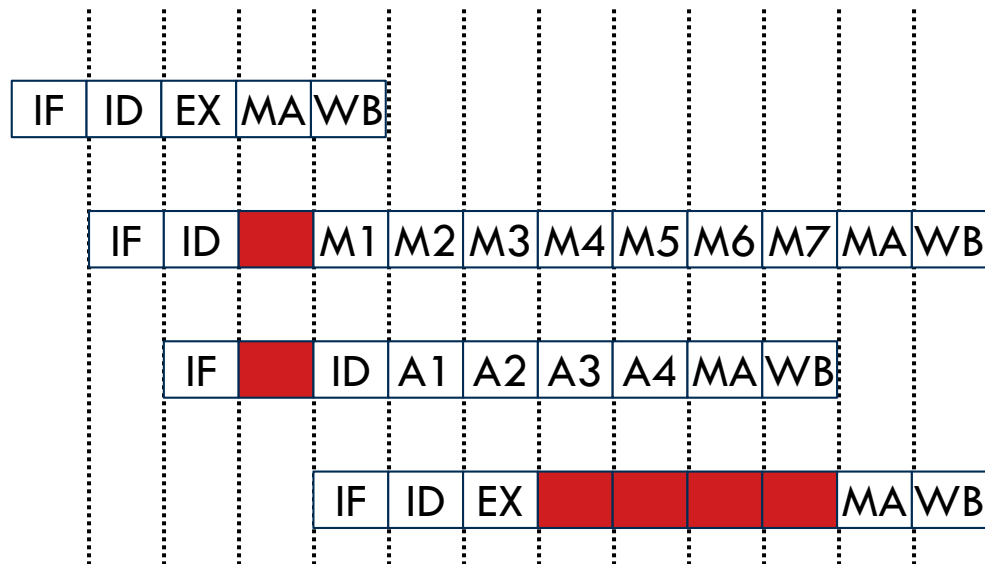
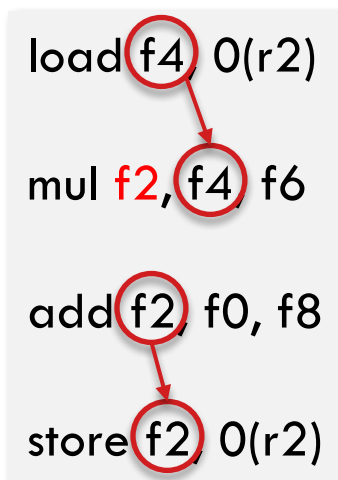
- Data hazards
 - ▣ more read-after-write hazards



Multicycle Instructions

- Data hazards
 - ▣ potential write-after-write hazards

load f4, 0(r2)
mul f2, f4, f6
add f2, f0, f8
store f2, 0(r2)



**Out of Order
Write-back!!**

Multicycle Instructions

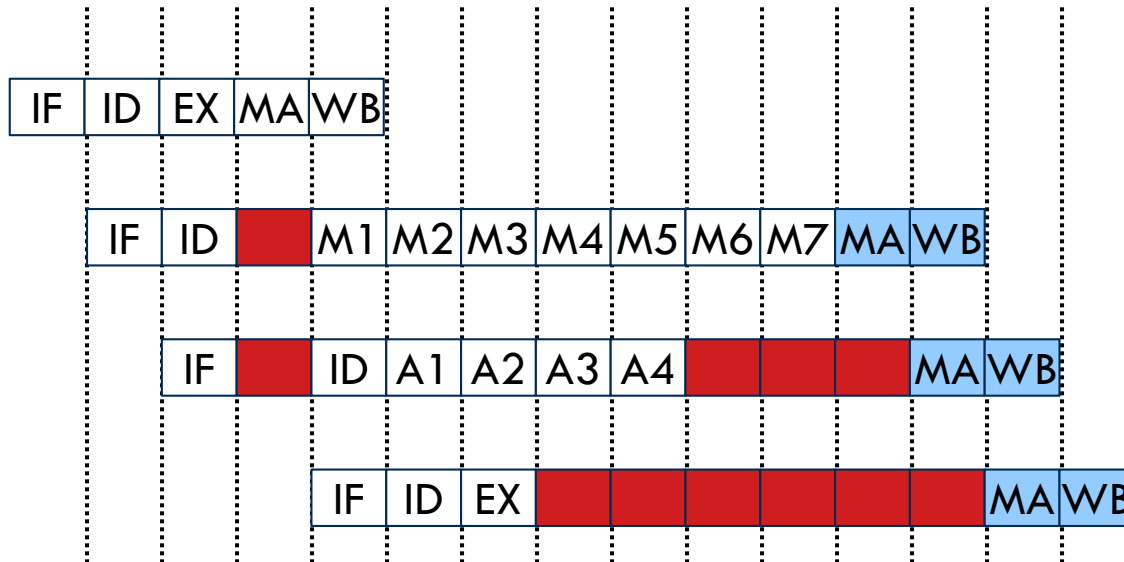
- Data hazards
 - ▣ potential write-after-write hazards

load f4, 0(r2)

mul f2, f4, f6

add f2, f0, f8

store f2, 0(r2)



**In-Order
Writes**

Multicycle Instructions

- Imprecise exception

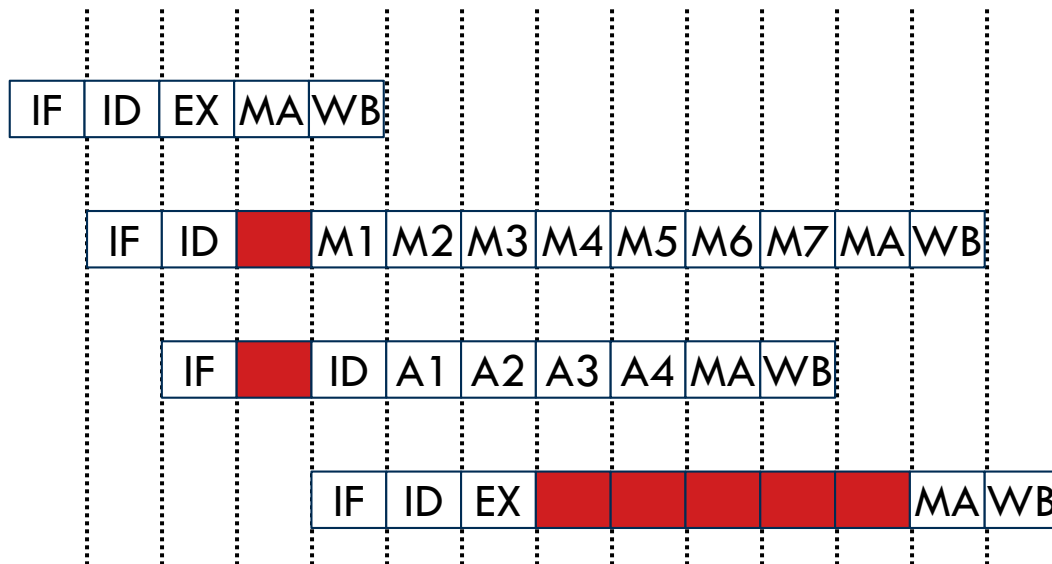
- ▣ instructions do not necessarily complete in program order

load f4, 0(r2)

mul f2, f4, f6

add f3, f0, f8

store f2, 0(r2)

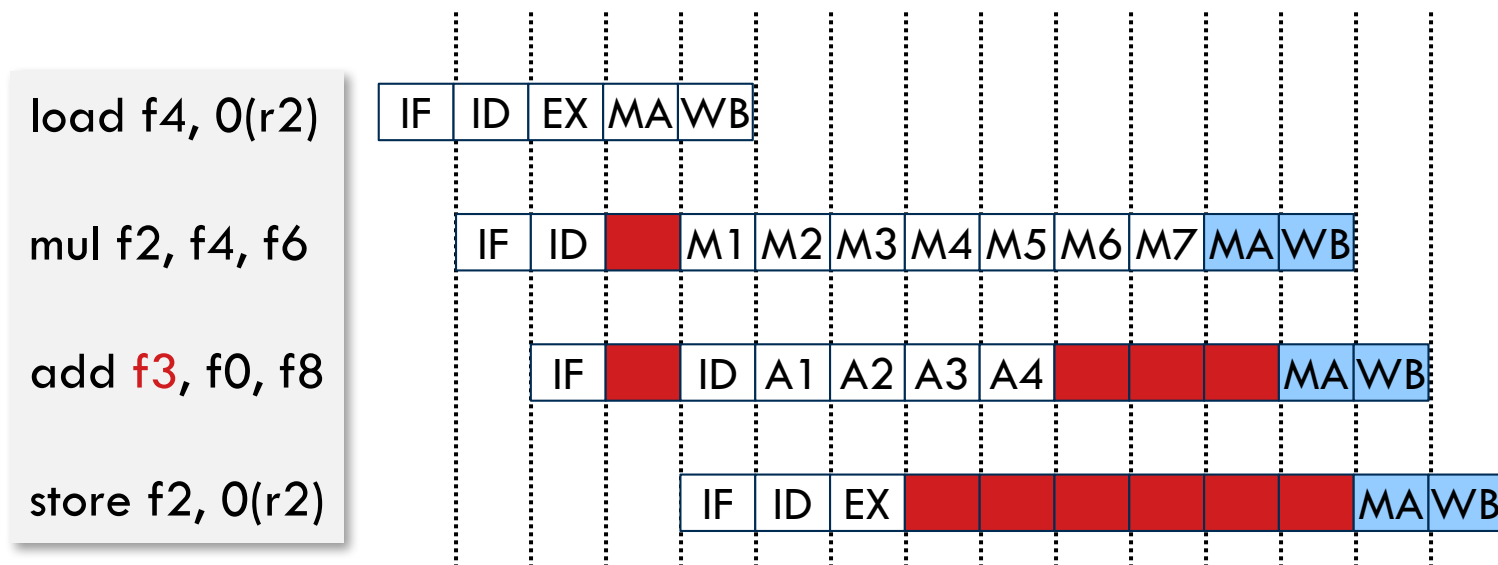


Overflow!!

Multicycle Instructions

- Imprecise exception

- ▣ state of the processor must be kept updated with respect to the program order

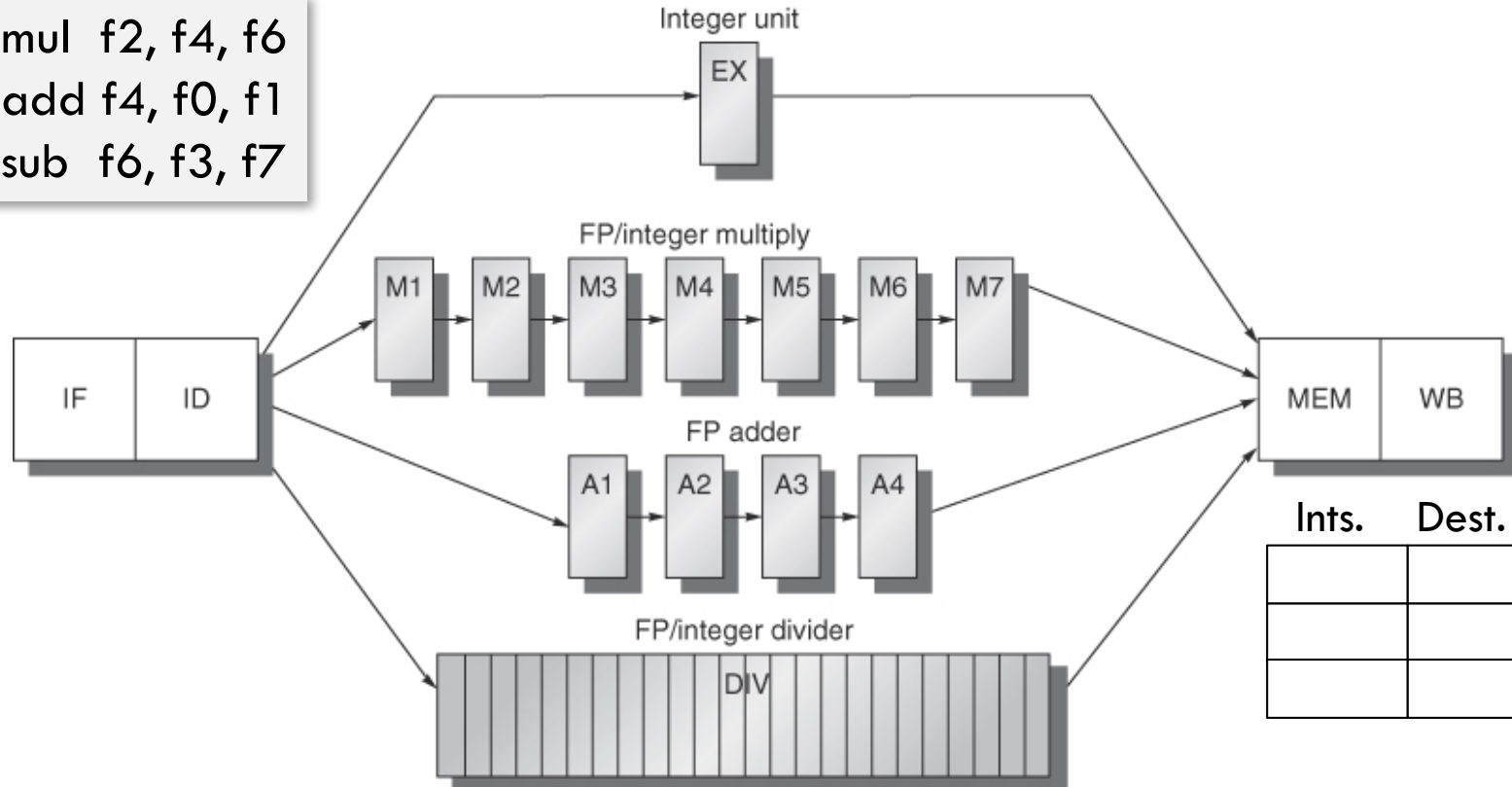


In-order register file updates

Reorder Buffer

□ Multicycle Instructions

```
mul f2, f4, f6  
add f4, f0, f1  
sub f6, f3, f7
```



Reorder Buffer

□ Multicycle Instructions

```
mul f2, f4, f6  
add f4, f0, f1  
sub f6, f3, f7
```

