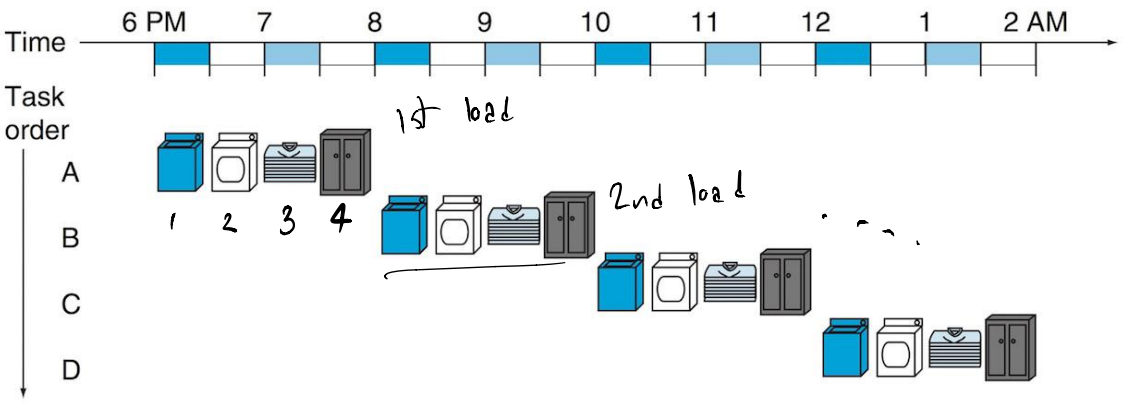
Pipelining

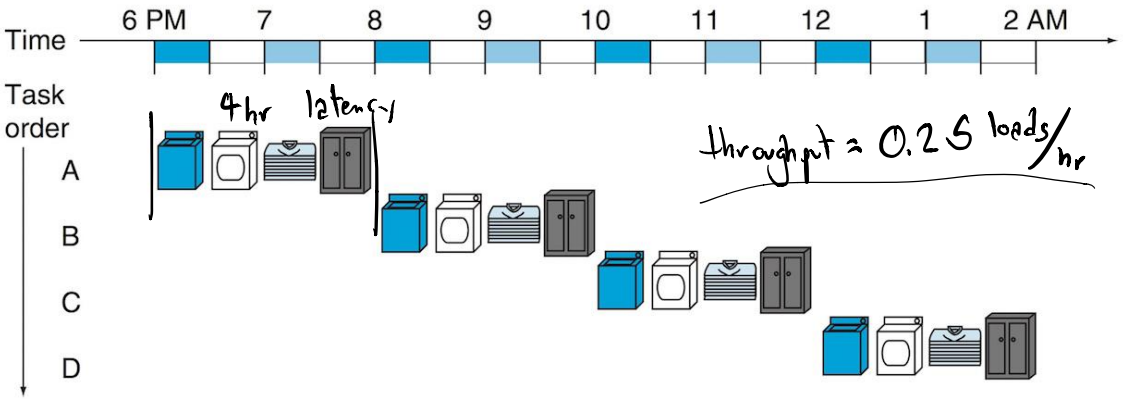
CSC 211 – December 1, 2020

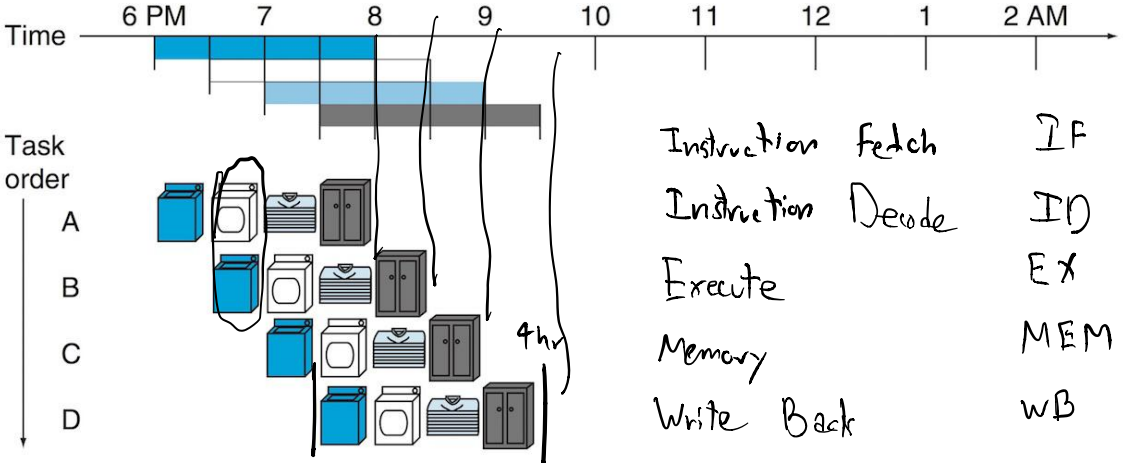
Datapath Lab Q&A

No questions!

What is Pipelining?


Time
6 PM
7
8
9
10
11
12
1
2 AM
Task order
A
B
C
D
1st load
2nd load
1
2
3
4





throughput = 1 load/hr

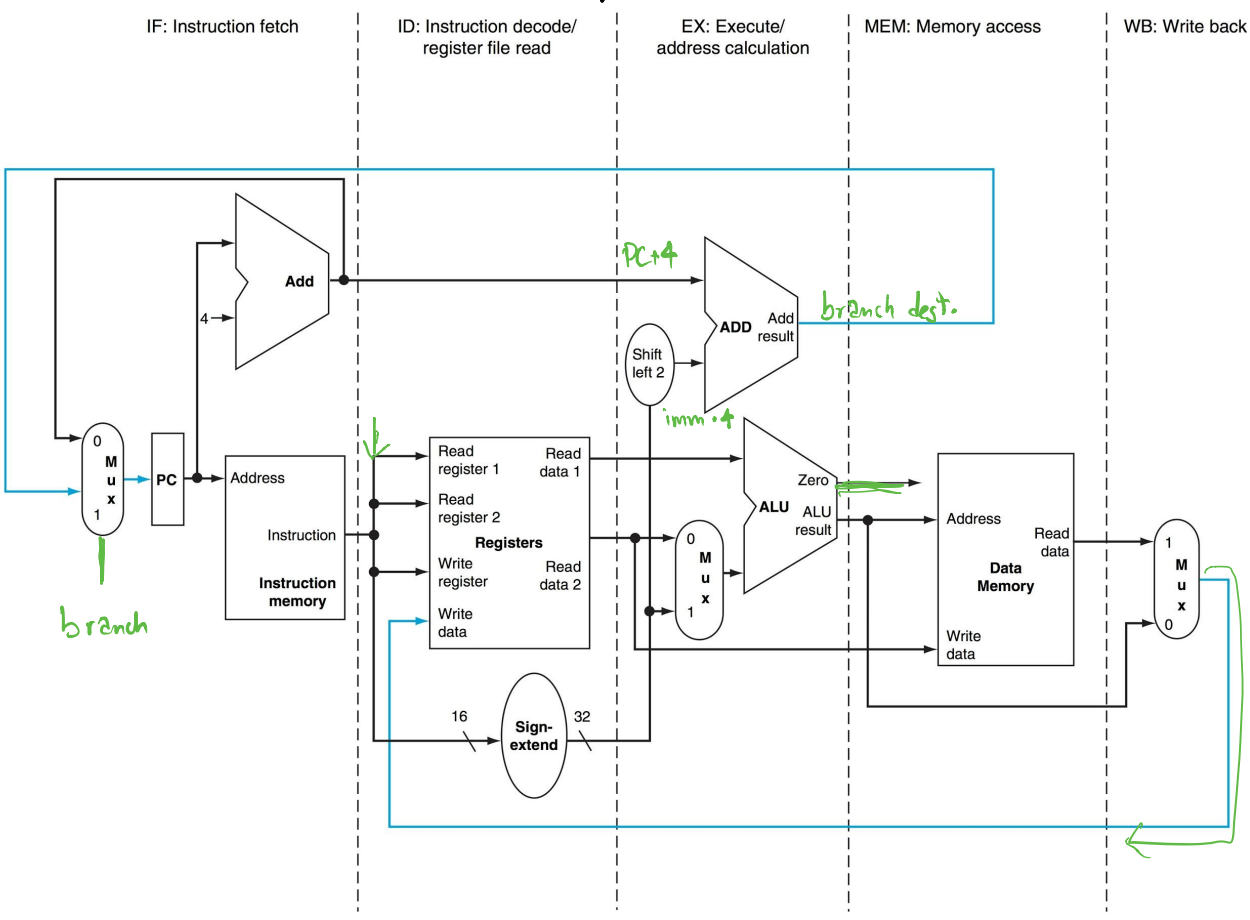
Instruction Fetch IF

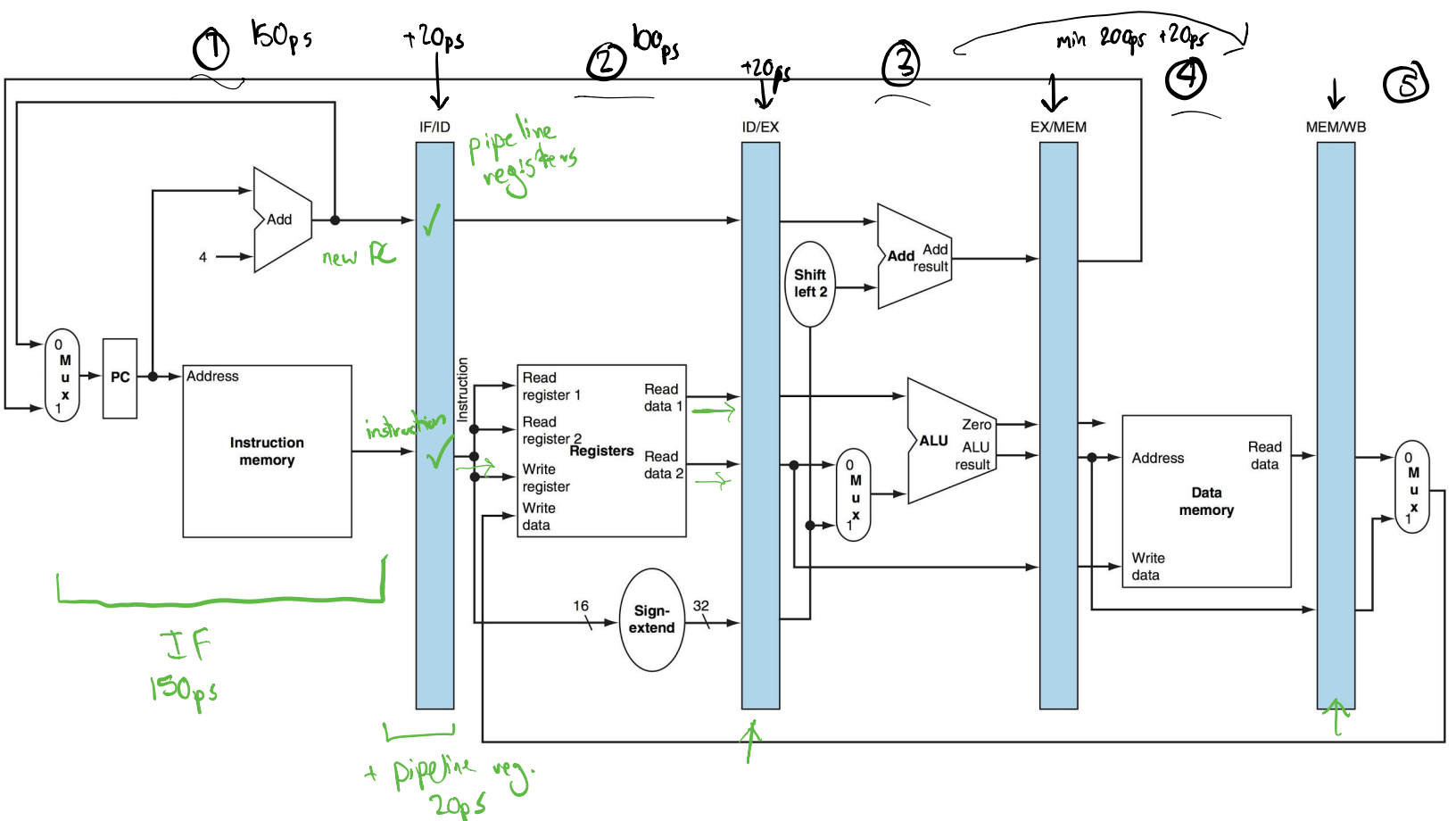
Instruction Decode ID

Execute EX

Memory MEM

Write Back WB


IF: Instruction fetch
ID: Instruction decode/ register file read
EX: Execute/ address calculation
MEM: Memory access
WB: Write back
Add
4
PC+4
ADD
Add result
branch dest.
Shift left 2
imm*4
0
M
u
x
1
branch
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
ALU
Zero
ALU result
Address
Read data
Data Memory
Write data
16
Sign-extend
32


1
150ps
+20ps
2
100ps
+20ps
3
min 200ps +20ps
4
5
IF/ID
ID/EX
EX/MEM
MEM/WB
pipeline registers
Add
4
new PC
M u x
0
1
PC
Address
Instruction memory
instruction
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
Add
Add result
ALU
Zero
ALU result
Address
Read data
Data memory
Write data
16
Sign-extend
32
IF
150ps
+ pipeline reg.
20ps

Comparing Performance

total time from start to end of the instruction's execution.

Compute maximum instruction latency and for single-cycle and pipelined datapaths using these parameters:

IF = 150ps

ID = 100ps

EX = 100ps

MEM = 200ps

WB = 100ps

Pipeline registers = 20ps

pipelining only

Latency

Throughput

Single-Cycle

650_{ps}

$\frac{1000_{ps}}{650_{ps/inst.}} \approx 1.5\ inst./ns$

$= 1.5$ billion $\frac{inst.}{sec.}$

Pipelined

$5 \cdot 220_{ps} = 1100_{ps}$

$\frac{1000_{ps}}{220_{ps/inst.}} \approx 4.5\ inst./ns$

$= 4.5$ billion $inst./sec$

Little's Law

$$L = \lambda \cdot W$$

L = length of queue
(or # of tasks running concurrently)

λ = completion rate
(inst. throughput)

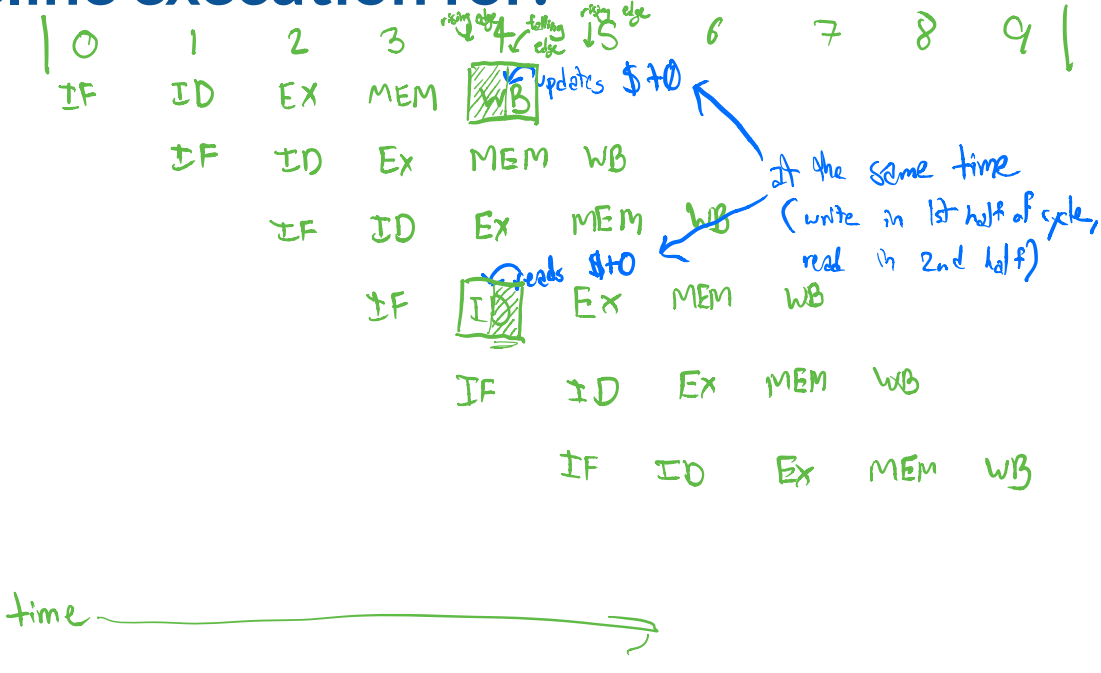
W = wait time
(inst. latency)

$$\lambda = \frac{L}{W} \Rightarrow \quad \lambda = \frac{5 \text{ inst.}}{1100_{ps.}} \rightarrow \frac{1 \text{ inst.}}{220_{ps}}$$

Pipeline Practice

Show the pipeline execution for:

```
addi $t0, $zero, 10
addi $t1, $zero, 11
addi $t2, $zero, 12
addi $t0, $t0, 1
addi $t1, $t1, 2
addi $t2, $t2, 3
```



add $t0, $t0, $zero ⇒ nop

Show the pipeline execution for:

Assume $t0 is initially 10

```
addi $t0, $zero, 0
addi $t0, $t0, 1
addi $t0, $t0, 1
addi $t0, $t0, 1
addi $t0, $t0, 1
addi $t0, $t0, 1
```

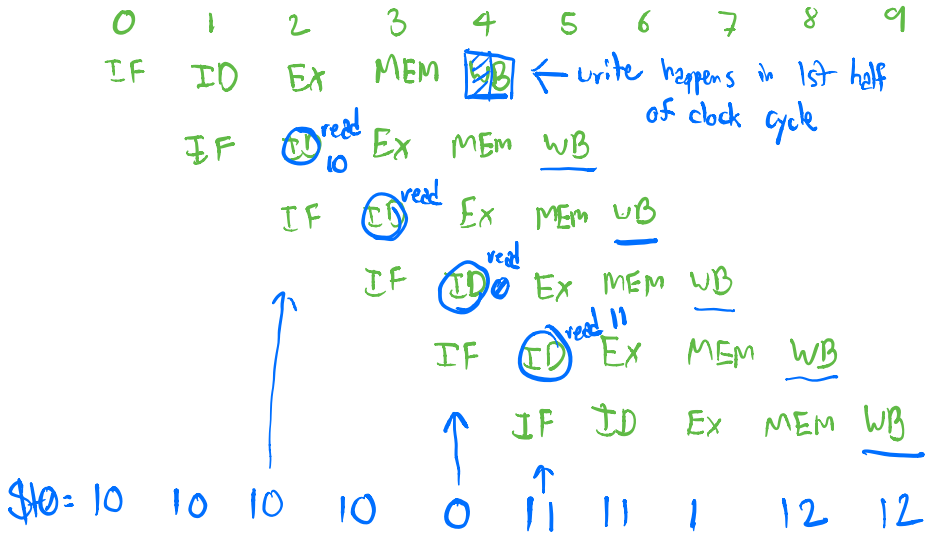
2x nop

2x nop

2x nop

2x nop

2x nop

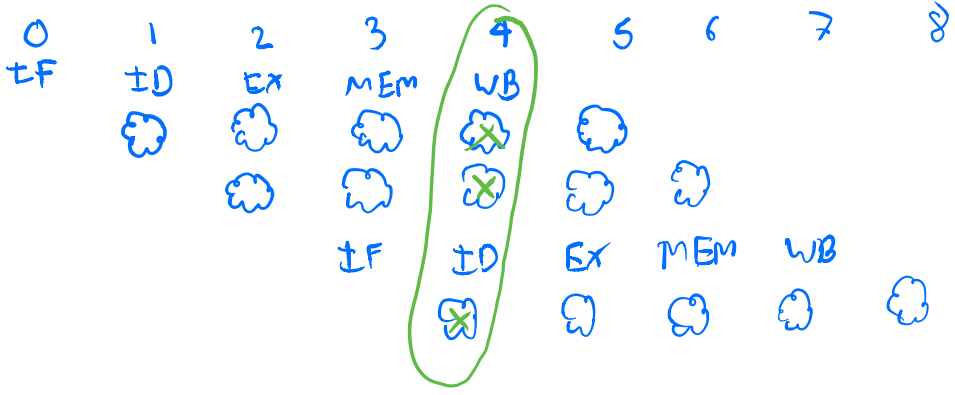


What is the value of $t0 at the end? $t0 = 12

How can we fix this instruction sequence?

Here's the fixed version

```
addi $t0, $zero, 0
nop
nop
addi $t0, $t0, 1
nop
nop
addi $t0, $t0, 1
nop
nop
addi $t0, $t0, 1
```



Static Scheduling Continued: reorder instructions

Show the pipeline execution for:

```
lw $s0, 0($t0)
addi $t0, $t0, 4
lw $s1, 0($t1)
addi $t1, $t1, 4
add $s2, $s0, $s1
```

	0	1	2	3	4	5	6	7	8
lw $s0, 0($t0)	IF	ID	EX	MEM	WB				
addi $t0, $t0, 4		IF	ID	EX	MEM	WB			
lw $s1, 0($t1)			IF	ID	EX	MEM	WB		
addi $t1, $t1, 4				IF	ID	EX	MEM	WB	
add $s2, $s0, $s1					IF	ID	EX	MEM	WB

Where will this sequence go wrong?
(where does it differ from a single-cycle datapath?)

The last instruction uses $s1 before it has been written

How can we fix this inst. sequence w/ static scheduling?

Swap the order of the 2nd and 3rd insts

Show the pipeline execution for:

```
A addi $t0, $zero, 0
B add  $t0, $t0, $s0
C add  $t0, $t0, $s1
D addi $t1, $zero, 1
E add  $t1, $t1, $s2
F add  $t1, $t1, $s3
```

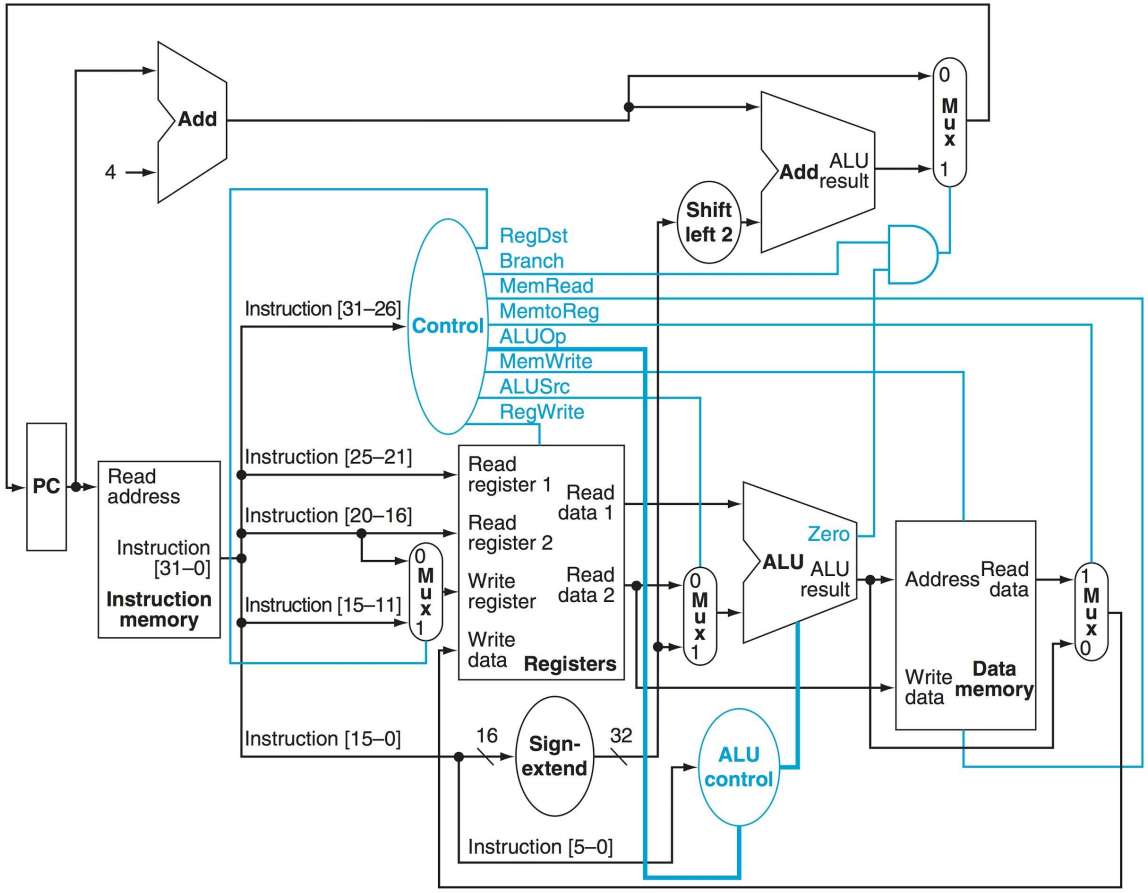
A
D
nop
B
E
nop
C
F

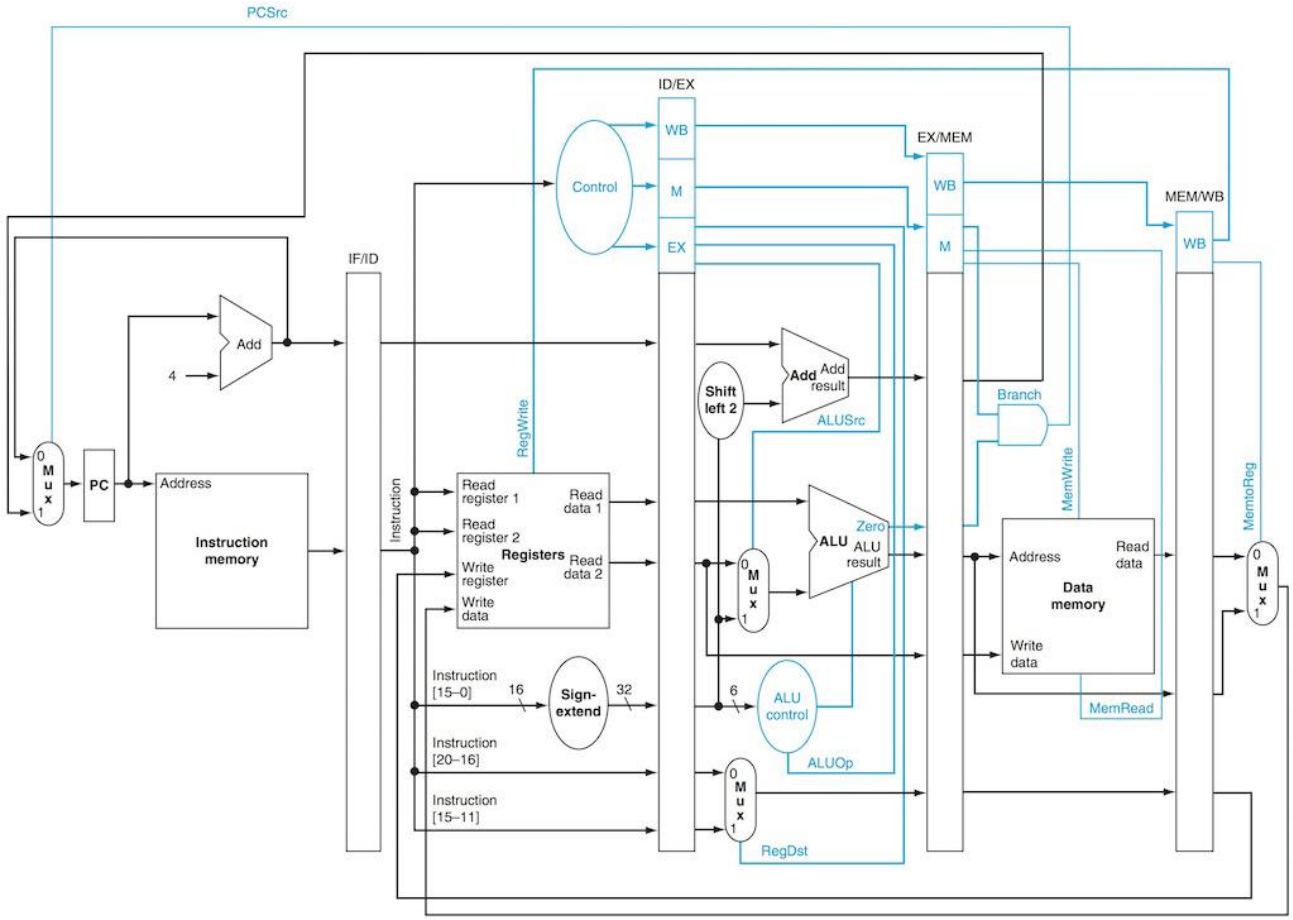
1. Where does this inst. sequence go wrong on a pipelined datapath?
   Instructions B, C, E and F read values before they are ready.
2. How can we fix it w/ static scheduling?
   See schedule above

Show the pipeline execution for:

```
  addi $s0, $zero, 0
  j somewhere
  addi $s0, $s0, 1
  addi $s1, $s1, 1
  ...
somewhere:
  addi $s1, $zero, 1
```

Controlling a Pipelined Datapath


Add
4
Add
ALU result
0
M
u
x
1
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
1
M
u
x
0


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
0
M
u
x
1
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [15–0]
16
Sign-extend
32
Instruction [20–16]
Instruction [15–11]
Shift left 2
Add
Add result
ALUSrc
ALU
Zero
ALU result
6
ALU control
ALUOp
RegDst
Branch
MemWrite
MemRead
MemtoReg
Address
Read data
Data memory
Write data