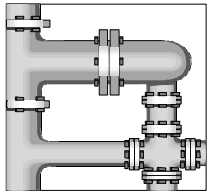
How Computers Work
Lecture 12

Introduction to Pipelining



A Common Chore of College Life

Propagation Times

Tpd_{wash} = _____ Tpd_{dry} = _____

Doing 1 Load

Step 1:

Step 2:

Total Time = _____

= _____

Doing 2 Loads

Combinational (Harvard) Method

Step 1:

Step 2:

Step 3:

Step 4:

Total Time

= _____

= _____

Doing 2 Loads
Pipelined (MIT) Method

Step 1:

Step 2:

Step 3:

Total Time

= _____

= _____

Doing N Loads

- Harvard Method:_____
- MIT Method:_____

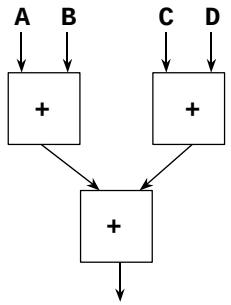
A Few Definitions

Latency: Time for 1 object to pass through entire system.
(= _____ for Harvard laundry)
(= _____ for MIT laundry)

Throughput: Rate of objects going through.
(= _____ for Harvard laundry)
(= _____ for MIT laundry)

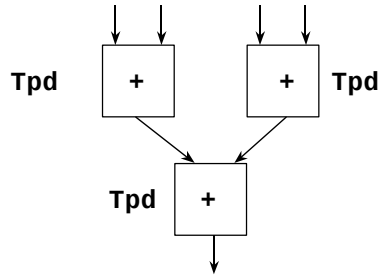
A Computational Problem

Add 4 Numbers:



A + B + C + D

As a Combinational Circuit



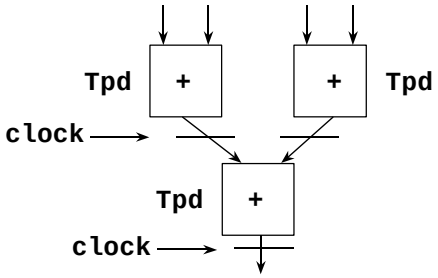
Throughput

1 / 2 Tpd

Latency

2 Tpd

As a Pipelined Circuit



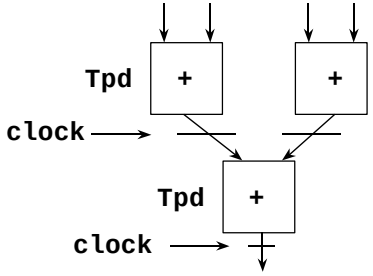
Throughput

1 / Tpd

Latency

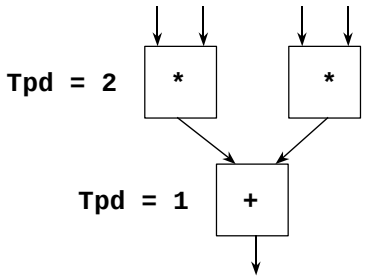
2 Tpd

Simplifying Assumptions



1. **Synchronous inputs**
2. **Ts = Th = 0**
   Tpd c-q = 0
   Tcd c-q = 0

An Inhomogeneous Case (Combinational)



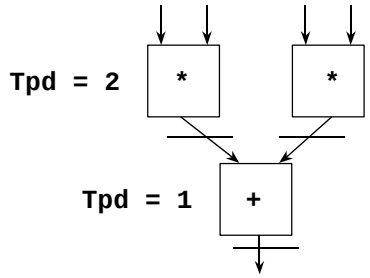
Throughput

1 / 3

Latency

3

An Inhomogeneous Case (Pipelined)



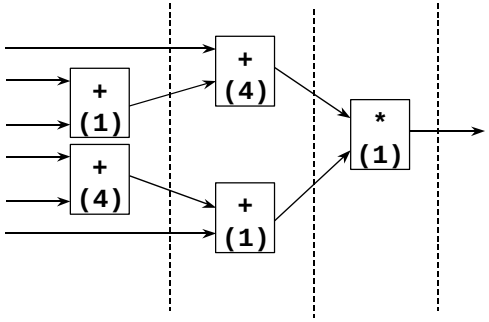
Throughput

1 / 2

Latency

4

How about this one?



Comb. Latency

6

Comb. Throughput

1/6

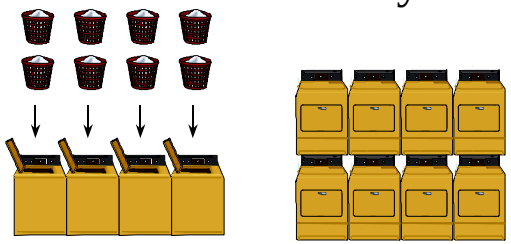
Pipe. Latency

12

Pipe. Throughput

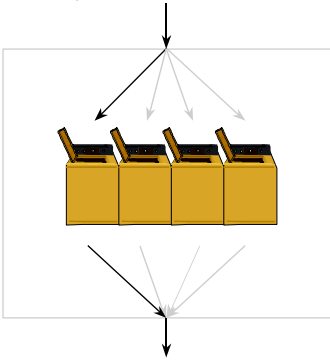
1/4

How MIT Students *REALLY* do Laundry



Steady State Throughput = _____
Steady State Latency = _____

Interleaving
(an alternative to Pipelining)



For N Units of delay Tpd, steady state

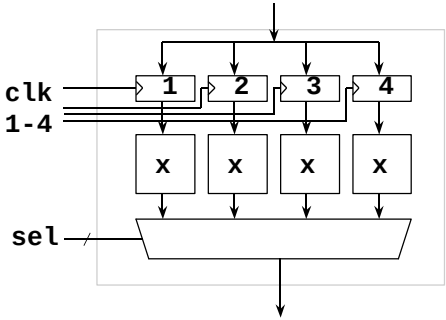
Throughput

N / Tpd

Latency

Tpd

Interleaving Parallel Circuits

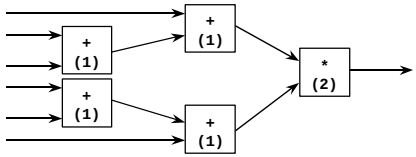


Definition of a Well-Formed Pipeline

- Same number of registers along path from any input to every computational unit
  - Insures that every computational unit sees inputs IN PHASE
- Is true (non-obvious) whenever the # of registered between all inputs and all outputs is the same.

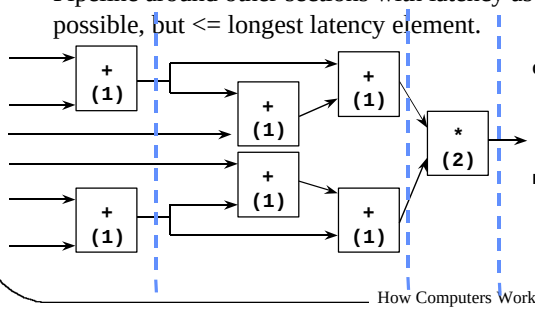
Method for Forming Well-Formed Pipelines

- Add registers to system output at will
- Propagate registers from intermediate outputs to intermediate inputs, cloning registers as necessary.



Method for Maximizing Throughput

- Pipeline around longest latency element
- Pipeline around other sections with latency as large as possible, but <= longest latency element.



Comb. Latency

5

Comb. Throughput

1/5

Pipe. Latency

6

Pipe. Throughput

1/2

A Few Questions

- Assuming a circuit is pipelined for optimum throughput with 0 delay registers, is the pipelined throughput always greater than or equal to the combinational throughput?
  - A: Yes
- Is the pipelined latency ever less than combinational latency?
  - A: No
- When is the pipelined latency equal to combinational latency?
  - A: If contents of all pipeline stages have equal combinational latency

CPU Performance

MIPS = Millions of Instructions Per Second

Freq = Clock Frequency, MHz

CPI = Clocks per Instruction

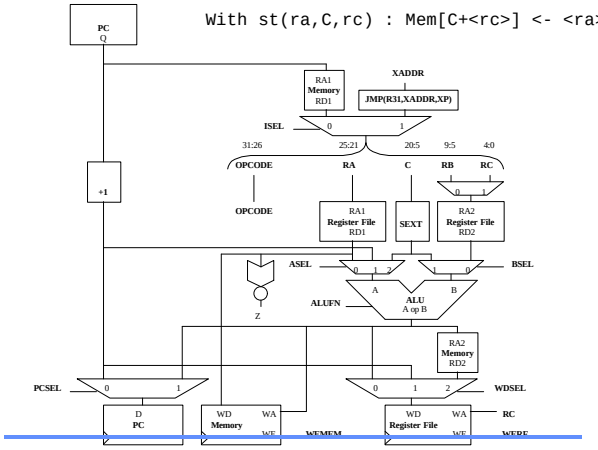
$$\text{MIPS} = \frac{\text{Freq}}{\text{CPI}}$$

To Increase MIPS:

1. **DECREASE CPI.**
   - **RISC reduces CPI to 1.0.**
   - **CPI < 0? Tough... we'll see multiple instruction issue machines at end of term.**
2. **INCREASE Freq.**
   - **Freq limited by delay along longest combinational path; hence**
   - ***PIPELINING* is the key to improved performance through fast clocks.**

Review: A Top-Down View of the Beta Architecture

```
With st(ra,C,rc) : Mem[C+<rc>] <- <ra>
```



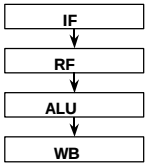
Pipeline Stages

GOAL: Maintain (nearly) 1.0 CPI, but increase clock speed.

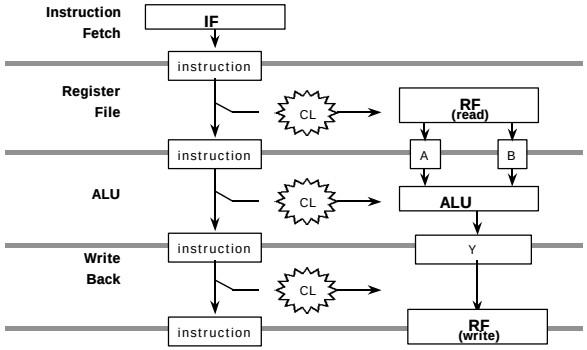
APPROACH: structure processor as 4-stage pipeline:

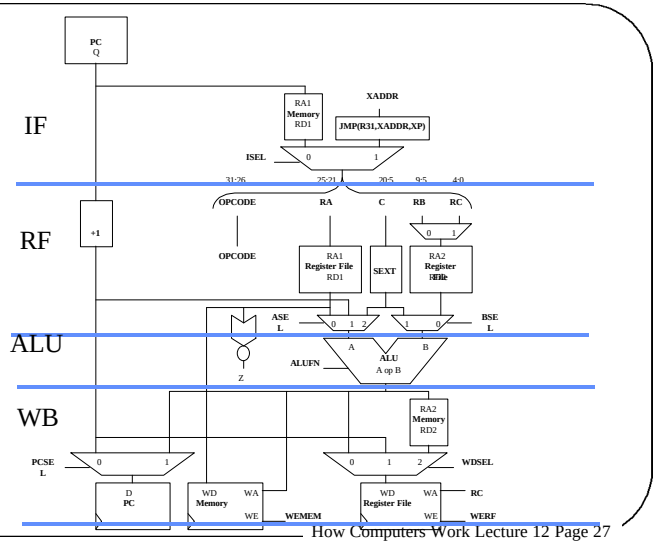
- **Instruction Fetch stage:** Maintains PC, fetches one instruction per cycle and passes it to
- **Register File stage:** Reads source operands from register file, passes them to
- **ALU stage:** Performs indicated operation, passes result to
- **Write-Back stage:** writes result back into register file.

WHAT OTHER information do we have to pass down the pipeline?



Sketch of 4-Stage Pipeline



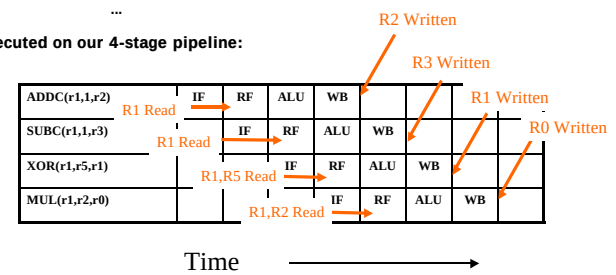


4-Pipeline Parallelism

Consider a sequence of instructions:

...
ADDC(r1, 1, r2)
SUBC(r1, 1, r3)
XOR(r1, r5, r1)
MUL(r1, r2, r0)
...

Executed on our 4-stage pipeline:



Pipeline Problems

BUT, consider instead:

```
LOOP:   ADD(r1, r2, r3)
        CMPLEC(r3, 100, r0)
        BT(r0, LOOP)
        XOR(r31, r31, r3)
        MUL(r1, r2, r2)
        ...
```

ADD(r1,r2,r3)	IF	RF	ALU	WB				
CMPLEC(r3,100,r0)		IF	RF	ALU	WB			
BT(r0.LOOP)			IF	RF	ALU	WB		
XOR(r31,r31,r3)				IF	RF	ALU	WB	
MUL(r1,r2,r2)					IF	RF	ALU	WB

Time →

Pipeline Hazards

PROBLEM:

Contents of a register WRITTEN by instruction k is READ by instruction k+1... before its stored in RF! EG:

```
ADD(r1, r2, r3)
CMPLEC(r3, 100, r0)
MULC(r1, 100, r4)
SUB(r1, r2, r5)
```

fails since CMPLEC sees "stale" <r3>.

R3 Written

R3 Read

ADD(r1,r2,r3)	IF	RF	ALU	WB				
CMPLEC(r3,100,r0)		IF	RF	ALU	WB			
BT(r0.LOOP)			IF	RF	ALU	WB		
XOR(r31,r31,r3)				IF	RF	ALU	WB	
MUL(r1,r2,r2)					IF	RF	ALU	WB

Time →

R3 Written

ADD(r1,r2,r3)	IF	RF	ALU	WB				
CMPLEC(r3,100,r0)		IF	RF	ALU	WB			
			IF	RF	ALU	WB		
CMPLEC(r3,100,r0)				IF	RF	ALU	WB	
					IF	RF	ALU	WB

R3 Read

SOLUTIONS:

1. "Program around it".
   ... document weirdo semantics, declare it a software problem.
   - Breaks sequential semantics!
   - Costs code efficiency.

EXAMPLE: Rewrite

ADD(r1, r2, r3)		ADD(r1, r2, r3)
CMPLEC(r3, 100, r0)		MULC(r1, 100, r4)
MULC(r1, 100, r4)	**as**	SUB(r1, r2, r5)
SUB(r1, r2, r5)		CMPLEC(r3, 100, r0)

HOW OFTEN can we do this?

SOLUTIONS:

2. Stall the pipeline.
   Freeze IF, RF stages for 2 cycles,
   inserting NOPs into ALU IR...

R3 Written

ADD(r1,r2,r3)	IF	RF	ALU	WB							
NOP		IF	RF	ALU	WB						
NOP			IF	RF	ALU	WB					
CMPLEC(r3,100,r0)				IF	RF	ALU	WB				
BT(r0.LOOP)					IF	RF	ALU	WB			
XOR(r31,r31,r3)						IF	RF	ALU	WB		
MUL(r1,r2,r2)							IF	RF	ALU	WB	

R3 Read

DRAWBACK: SLOW

SOLUTIONS:

3. Bypass Paths.

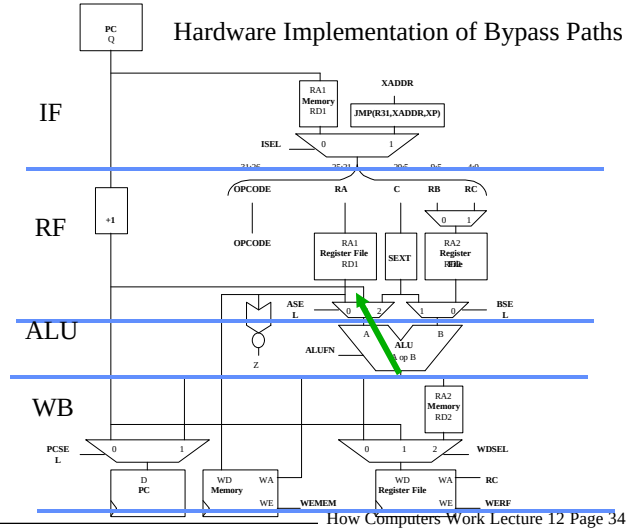
   Add extra data paths & control logic to re-route data in problem cases.

<R1>+<R2> Produced

<R1>+<R2> Used

ADD(r1,r2,r3)	**IF**	**RF**	**ALU**	**WB**				
CMPLEC(r3,100,r0)		**IF**	**RF**	**ALU**	**WB**			
BT(r0.LOOP)			**IF**	**RF**	**ALU**	**WB**		
XOR(r31,r31,r3)				**IF**	**RF**	**ALU**	**WB**	
MUL(r1,r2,r2)					**IF**	**RF**	**ALU**	**WB**

Hardware Implementation of Bypass Paths



Next Time:

- Detailed Design of
  - Bypass Paths + Control Logic
- What to do when Bypass Paths Don't Work
  - Branch Delays / Tradeoffs
  - Load/Store Delays / Tradeoffs
  - Multi-Stage Memory Pipeline