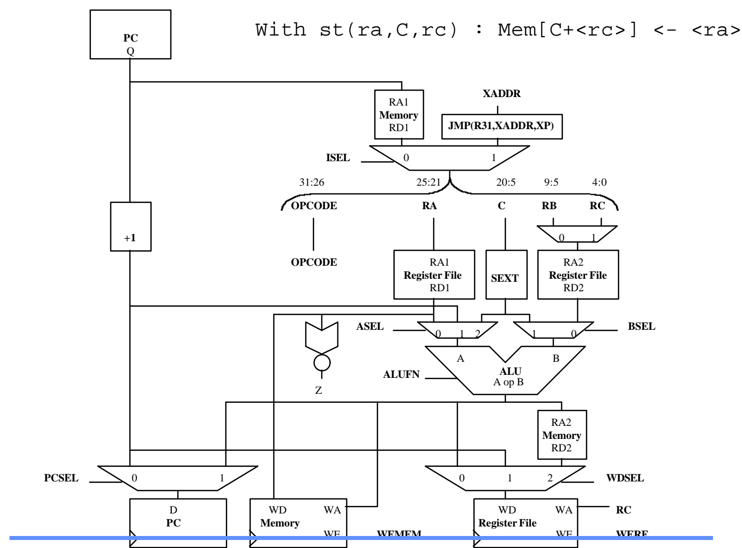
How Computers Work
Lecture 13
Details of the Pipelined Beta

Review: 1-Stage Beta (Top-Down View)



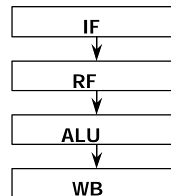
Review: Pipeline Stages

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

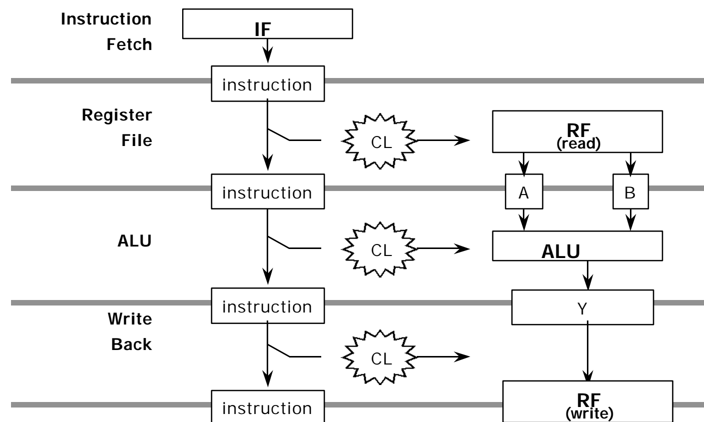
APPROACH: structure processor as 4-stage pipeline:

- **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?



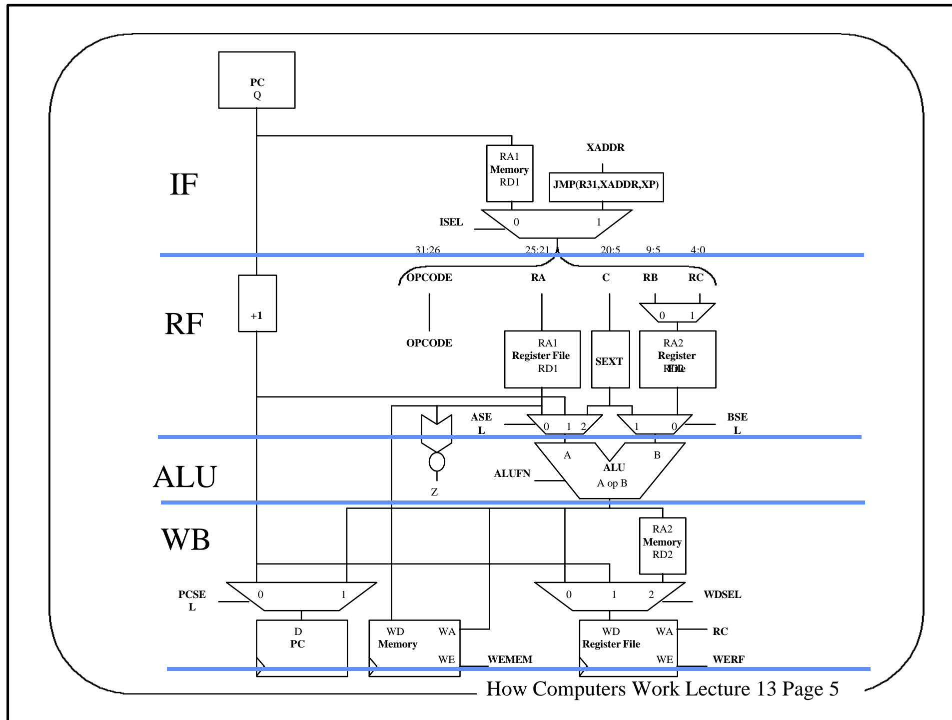
Sketch of 4-Stage Pipeline



NEED ALSO:

<PC> - for Branch/JMP instrs!

Lets make it ***real....***



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)

fails since CMPLEC sees "stale" <r3>.

R3 Written

ADD(r1,r2,r3)	IF	RF	ALU	WB				
CMPLEC(r3,100,r0)	R3 Read	IF	RF	ALU	WB			

Time →

How Computers Work Lecture 13 Page 6

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	

R3 Written

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

as

```
ADD(r1, r2, r3)
MULC(r1, 100, r4)
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				

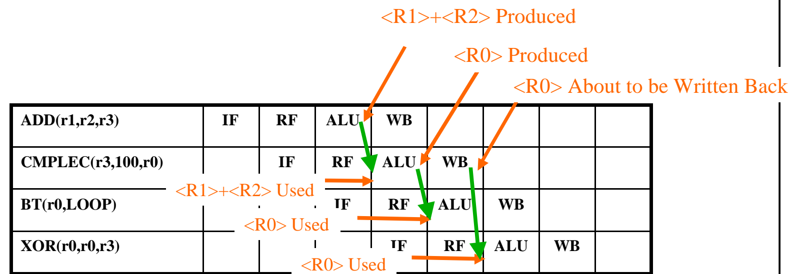
R3 Read

DRAWBACK: SLOW

SOLUTIONS:

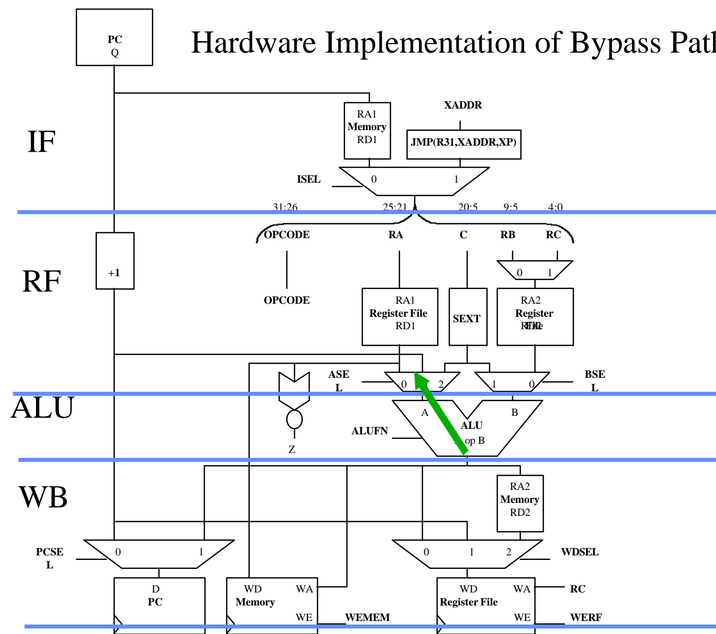
3. Bypass Paths.

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

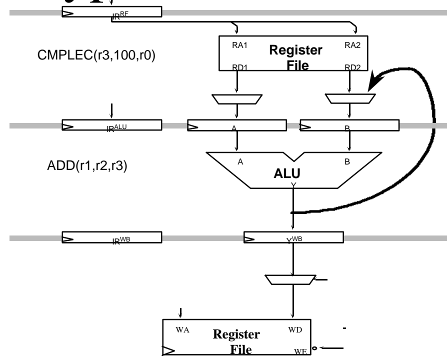


ADD(r1,r2,r3)	IF	RF	ALU	WB				
CMPLEC(r3,100,r0)		IF	RF	ALU	WB			
BT(r0,LOOP)			IF	RF	ALU	WB		
XOR(r0,r0,r3)				IF	RF	ALU	WB	

Hardware Implementation of Bypass Paths

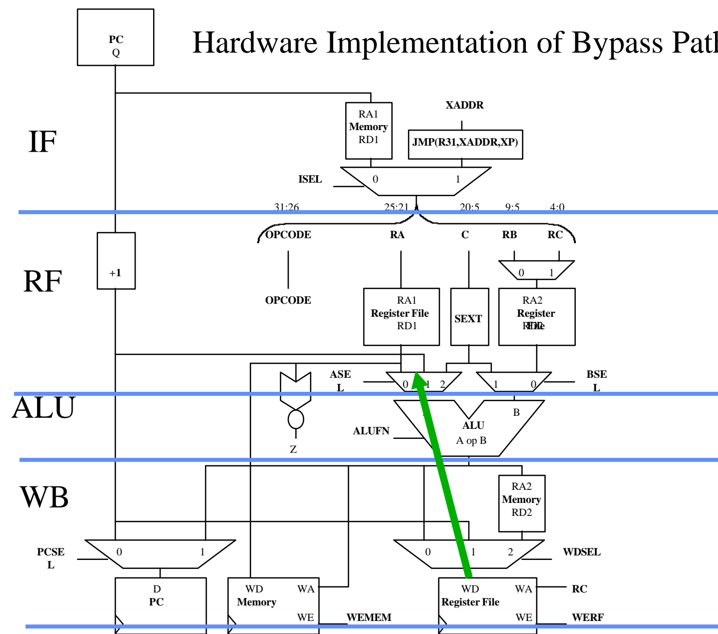


Bypass Paths - I



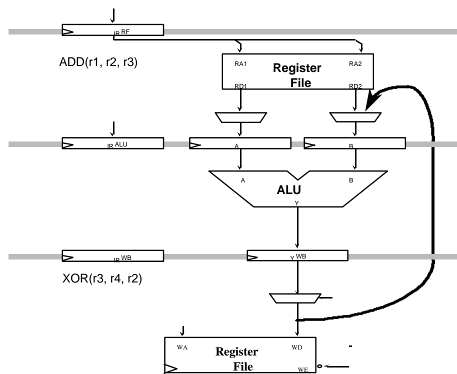
How Computers Work Lecture 13 Page 11

Hardware Implementation of Bypass Paths



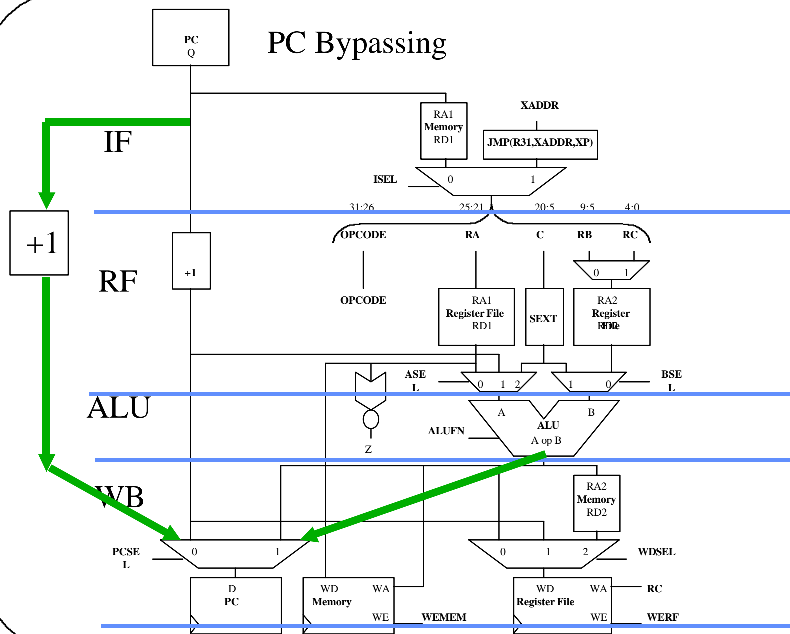
How Computers Work Lecture 13 Page 12

Bypass Paths - II



How Computers Work Lecture 13 Page 13

PC Bypassing



How Computers Work Lecture 13 Page 14

BRANCH DELAY SLOTS

PROBLEM: One (or more) following instructions have been pre-fetched by the time a branch is taken.

POSSIBLE SOLUTIONS:

1. "Program around it". Either

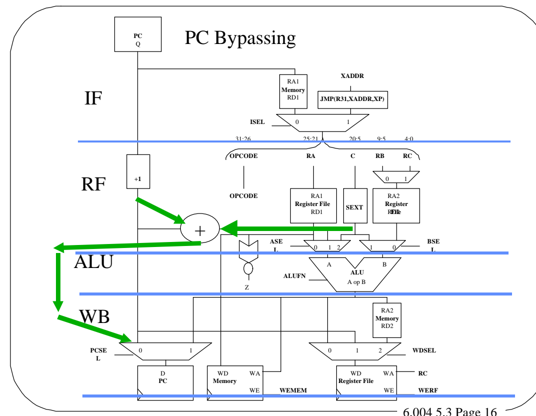
   1a. Follow each BR with 2 NOP instructions; or

   1b. Make your compiler clever enough to move USEFUL instructions following branches.

2. Make pipeline "annul" instructions following branches which are taken, eg by disabling

   WERF and WEMEM and PCSEL.

Can we shorten the number of delay slots? A: Yes (by 1)



Load Delays

Consider LOADS:

Can we fix all these problems using our previous bypass paths?

LD(r1, 0, r4)

ADD(r1, r4, r5)

XOR(r3, r4, r6)

ANSWER: No – only 1 (XOR) !

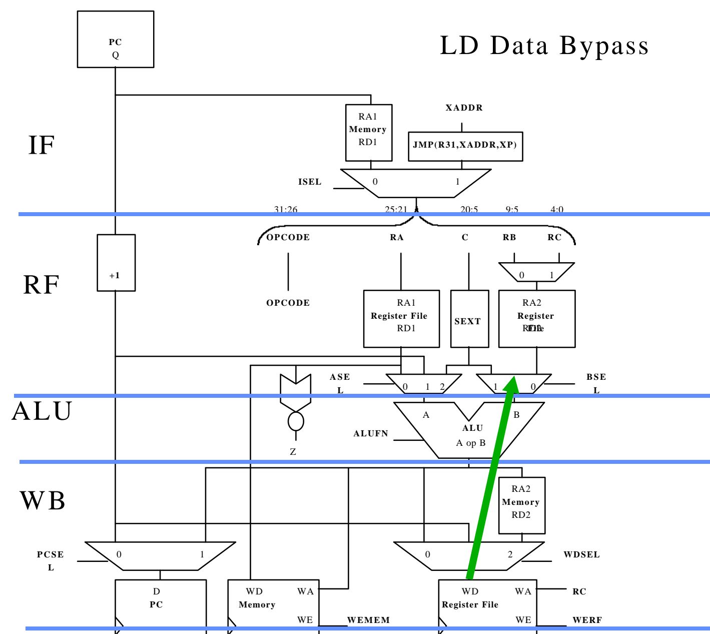
For a LD instruction fetched during clock i, data isn't returned from memory until late into cycle i + 3 !

LD Data Available

LD(r1, 0, r4)	IF	RF	ALU	WB				
ADD(r1, r4, r5)		IF	RF	ALU	WB			
XOR(r3, r4, r6)			IF	RF	ALU	WB		

R4 Needed

R4 Needed



Load Delays - II

Load Timing Problems:

```
LD(r1, 0, r4)
ADD(r1, r4, r5)
XOR(r3, r4, r6)
```

Problem 1 (LD destination r4 → ADD source r4)

Problem 2 (LD destination r4 → XOR source r4)

Can relegate both problems to Compiler.

Alternatively, fix Problem 2 using

Bypass Paths

and fix Problem 1 using

NOPs / Stalls

Load Problems - III

But, but, what about **FASTER** processors?

FACT: Processors will become fast relative to memories!

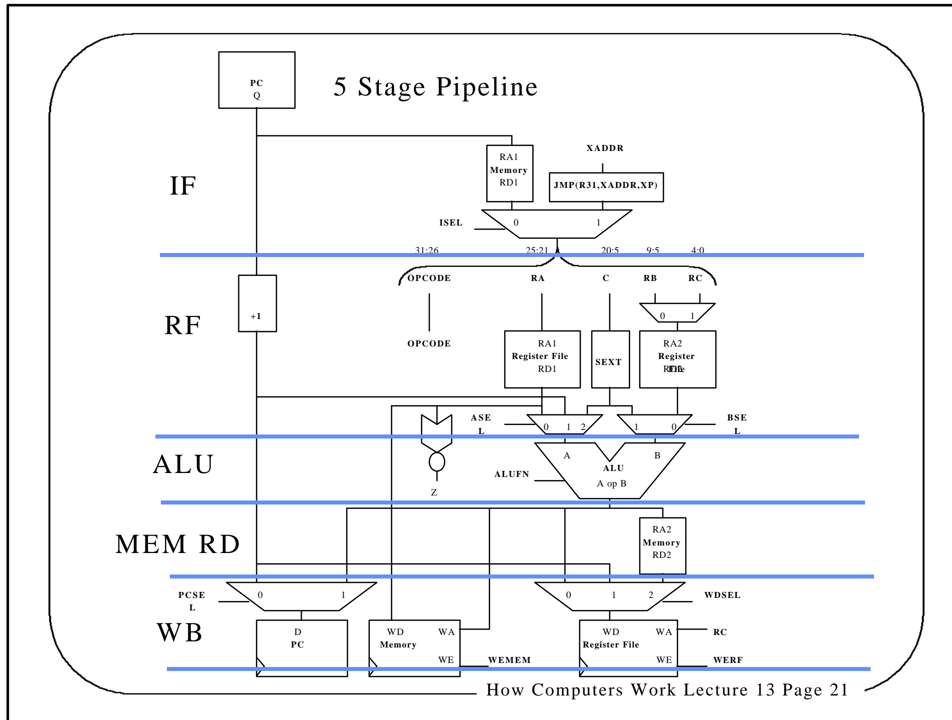
Do we just lengthen the cycle time?

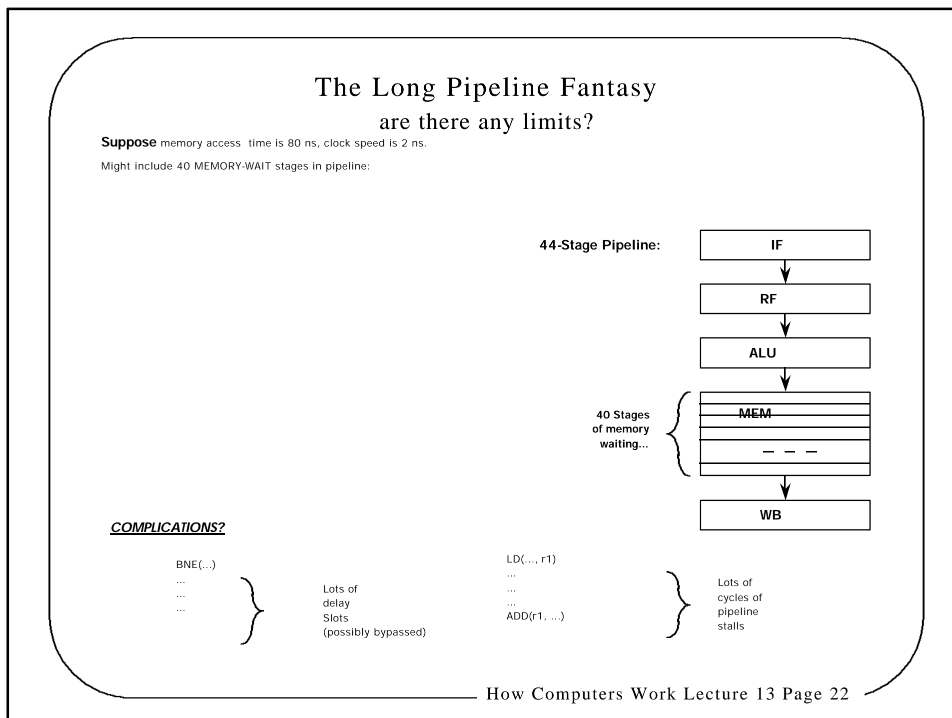
ALTERNATIVE: Longer pipelines.

1. Add "MEMORY WAIT" stages between START of read operation & return of data.
2. Build pipelined memories, so that multiple (say, N) memory transactions can be in progress at once.
3. (Optional). Stall pipeline when the N limit is exceeded.

4-Stage pipeline requires 1 instruction's delay.

5-Stage pipeline requires 2 instruction's delay.


5 Stage Pipeline
PC
Q
IF
RF
ALU
MEM RD
WB
XADDR
JMP(R31,XADDR,XP)
RA1
Memory
RD1
ISEL
OPCODE
RA
C
RB
RC
Register File
SEXT
RA2
ASEL
BSEL
ALUFN
ALU
A op B
Z
Memory
RD2
PCSEL
WDSEL
PC
WD
WA
WE
WEMEM
WERF
How Computers Work Lecture 13 Page 21


The Long Pipeline Fantasy
are there any limits?
Suppose memory access time is 80 ns, clock speed is 2 ns.
Might include 40 MEMORY-WAIT stages in pipeline:
44-Stage Pipeline:
IF
RF
ALU
MEM
40 Stages of memory waiting...
WB
COMPLICATIONS?
BNE(...)
...
...
...
Lots of delay Slots (possibly bypassed)
LD(..., r1)
...
...
...
ADD(r1, ...)
Lots of cycles of pipeline stalls
How Computers Work Lecture 13 Page 22

What Have We Learned Today?

- Pipelining improves throughput by lowering clock period
- Pipelining cannot improve latency
- Data Hazards can be fixed with
  - Re-Programming, NOPs, Bypass Paths
- Branch Hazards can be fixed with
  - Re-Programming, NOPs, Annulment
- Memory Hazards can be fixed with
  - Re-Programming, NOPs, (1) Bypass Path
- As in Karate, balance is important … too much pipelining is BAD

Next Time:

- Implicit Multiple Issue
- Automatic Out-of-Order Execution