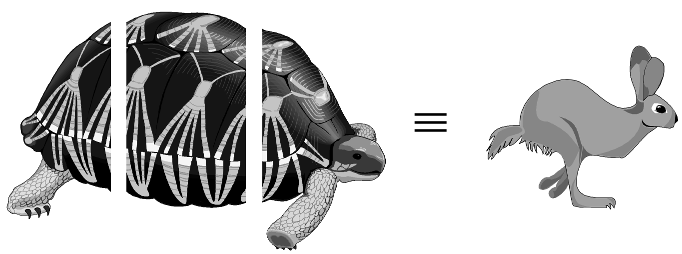
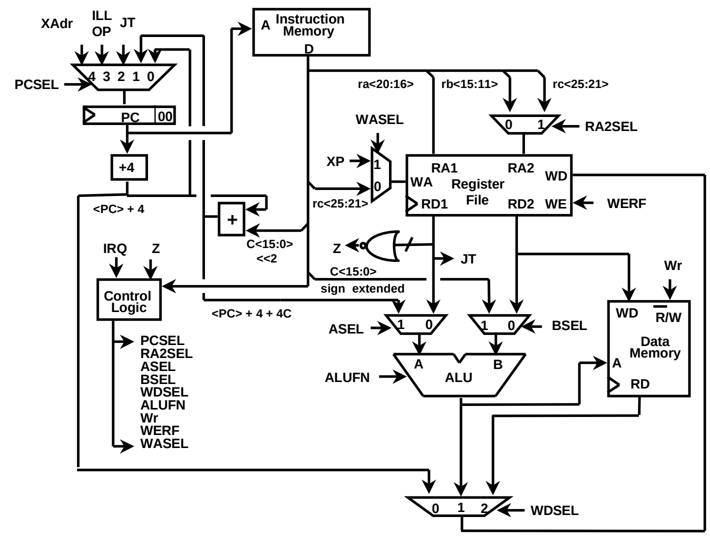
6004 Spring 1998: L30

Pipelined β Processor



1

Recap: Unpipelined β



What can we do to make it run FASTER?

2

CPU Performance

$$\text{MIPS} = \frac{\text{Frequency in MHz}}{\text{Clocks Per Instruction (CPI)}}$$

To increase MIPS:

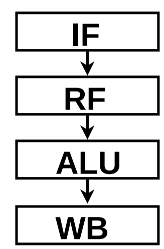
- Decrease CPI
  - » Insruction set simplicity reduces CPI to 1.0
  - » To reduce CPI below 1.0 need multiple instruction issue machines (stay tuned!)
- Increase Frequency
  - » Frequency limited by longest combinational path from register outputs to register inputs
  - » Can therefore increase frequency by pipelining

Pipeline Stages

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

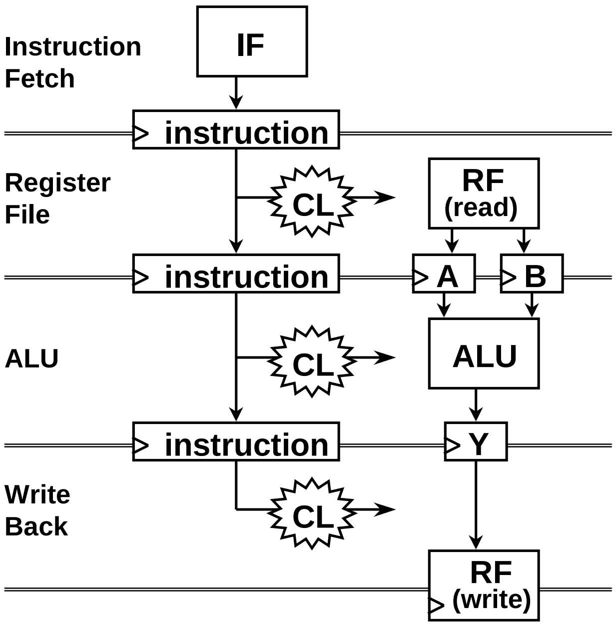
Approach: Structure processor as 4-stage pipeline

- Instruction Fetch: Maintains PC. Fetches one instruction per cycle.
- Register File: Reads source operands from register file.
- ALU: Performs indicated operation.
- Write-Back: Writes result back into register file.



6004 Spring 1998: L30

Sketch of 4-Stage Pipeline

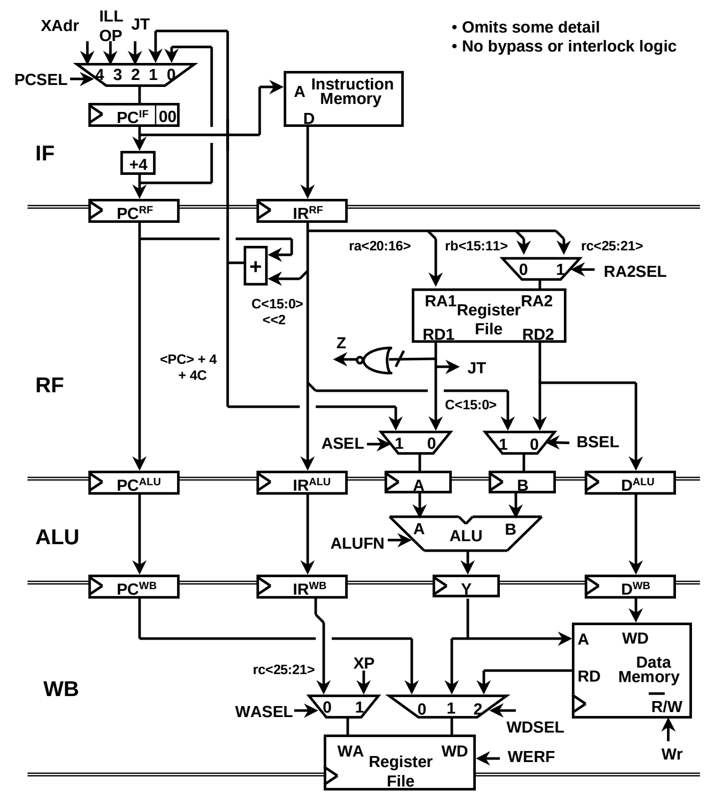


Need to pass ____ through the stages for branch and jump instructions

5

4-Stage β Pipeline

• Omits some detail
• No bypass or interlock logic



6

4-Pipeline Execution

Consider a sequence of instructions

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

executed on the 4-stage pipeline:

TIME (cycles) ⟶

Pipeline ↓	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*ADDC*	*SUBC*	*XOR*	*MUL*	...		
RF		*ADDC*	*SUBC*	*XOR*	*MUL*	...	
ALU			*ADDC*	*SUBC*	*XOR*	*MUL*	...
WB				*ADDC*	*SUBC*	*XOR*	*MUL*

Branch Execution

Consider a different sequence:

LOOP: CMPLEC(r3, 100, r0)
ADD(r1, r2, r3)
SUB(r1, r2, r4)
BNE(r0, LOOP)
XOR(r31, r31, r3)
...

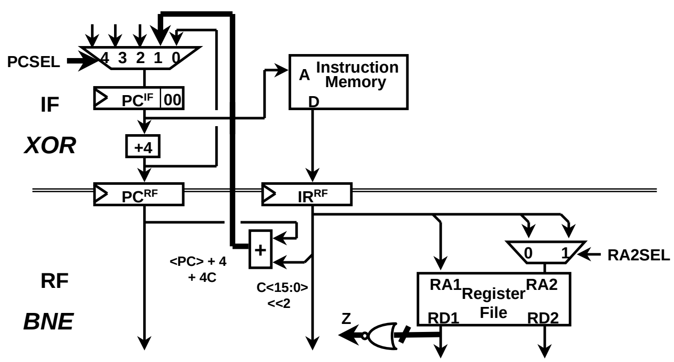
	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*CMP*	*ADD*	*SUB*	*BNE*	*?*		
RF		*CMP*	*ADD*	*SUB*	*BNE*		
ALU			*CMP*	*ADD*	*SUB*	*BNE*	
WB				*CMP*	*ADD*	*SUB*	*BNE*

What instruction should we fetch after *BNE* ?
ANSWER: ____________

6004 Spring 1998: L30

Branch Execution - II

	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*CMP*	*ADD*	*SUB*	*BNE*	*?*		
RF		*CMP*	*ADD*	*SUB*	*BNE*		
ALU			*CMP*	*ADD*	*SUB*	*BNE*	
WB				*CMP*	*ADD*	*SUB*	*BNE*



PC^{IF} will have the correct address in the ________ clock cycle

9

Branch Delay Slots

Problem: One (or more) following instructions have been prefetched by the time a branch is taken.

Possible solutions are

- **"Program around it"**
  - » **Follow each branch instruction with a *NOP = ADD(r31, r31, r31)* instruction.**
  - » **Make the compiler clever enough to move useful instructions after branches. These instructions will be executed regardless of whether the branch is taken or not.**
- **Make pipeline "annul" instruction following branch which is taken, e.g., by <u>disabling</u> RF write and Memory write.**

10

Data Hazards

Consider the sequence:

ADD(r1, r2, r3)
CMPLEC(r3, 5, r0)
MULC(r1, r2, r4)
SUB(r1, r2, r5)
...

	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*ADD*	*CMP*	*MUL*	*SUB*			
RF		*ADD*	*CMP*	*MUL*	*SUB*		
ALU			*ADD*	*CMP*	*MUL*	*SUB*	
WB				*ADD*	*CMP*	*MUL*	*SUB*

ADD **writes new value in** ***r3*** **during i + 3 cycle, which is available beginning of i + 4 cycle. Value of** ***r3*** **read by** ***CMP*** **during i + 2 cycle.** ***CMP*** **reads old value of** ***r3*****.**

11

Data Hazards: Solution - I

Solution I (software)

- **“Program around it”**
  - » **Compiler inserts NOPs**
  - » **Compiler rewrites instruction sequence**

ADD(r1, r2, r3)
CMPLEC(r3, 5, r0)

Insert how many NOPs here ?

Rewrite

ADD(r1, r2, r3)		*ADD(r1, r2, r3)*
CMPLEC(r3, 5, r0)		*MUL(r1, r2, r4)*
MUL(r1, r2, r4)	as	*SUB(r1, r2, r5)*
SUB(r1, r2, r5)		*CMPLEC(r3, 5, r0)*

12

Data Hazards: Solution - II

Solution II (hardware)

- **Detect problem and stall the pipeline**
  - **Freeze IF, RF stages for 2 cycles and insert NOPs into IR^{ALU} for 2 cycles**

ADD(r1, r2, r3)

CMPLEC(r3, 5, r0)

***r3* read**

	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*ADD*	*CMP*	*MUL*	*MUL*	*MUL*	*SUB*	
RF		*ADD*	*CMP*	*CMP*	*CMP*	*MUL*	*SUB*
ALU			*ADD*	NOP_1	NOP_2	*CMP*	*MUL*
WB				*ADD*	NOP_1	NOP_2	*CMP*

***r3* written**

Data Hazards: Solution - III

Solution III (hardware)

- **Bypass paths.**
  - **Extra data paths and control logic which re-route data in problem cases.**

ADD(r1, r2, r3)

CMPLEC(r3, 5, r0)

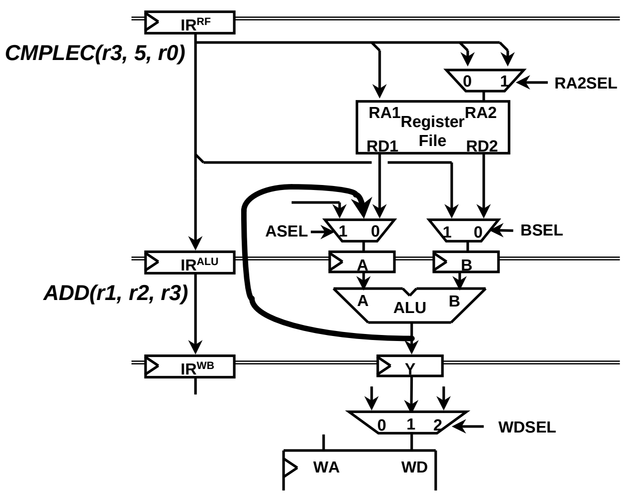
r3* read** ***r3* compared to *5

	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*ADD*	*CMP*	*MUL*	*SUB*			
RF		*ADD*	*CMP*	*MUL*	*SUB*		
ALU			*ADD*	*CMP*	*MUL*	*SUB*	
WB				*ADD*	*CMP*	*MUL*	*SUB*

***r1 + r2* computed** ***r3* written**

6004 Spring 1998: L30

Bypass Paths - I



Select bypass if

$OPCODE^{RF}$ = OP, OPC, ... AND

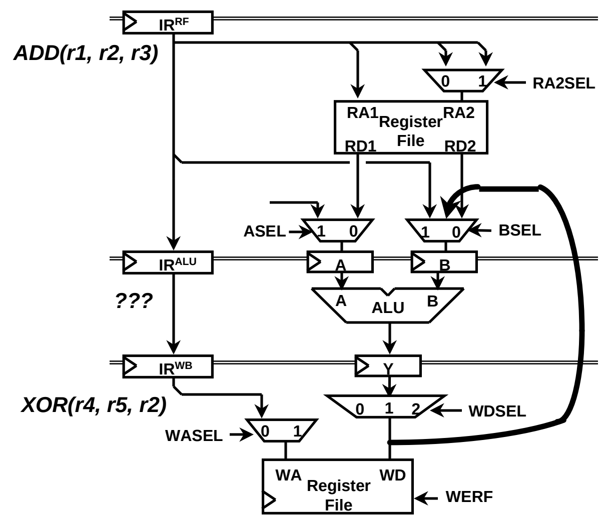
$OPCODE^{ALU}$ = OP, OPC, ... AND

ra^{RF} = rc^{ALU}

(Similar path for other ALU input)

15

Bypass Paths - II



Select bypass if

$OPCODE^{RF}$ = OP AND

WERF = 1 AND

rb^{RF} = WA

(Similar path for other ALU input)

16

Loads

Consider the sequence:

LD(r1, 0, r4) → Problem 1
ADD(r1, r4, r5) → Problem 2
XOR(r3, r4, r6)

r4 read 1 *r4* read 2

	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*LD*	*ADD*	*XOR*				
RF		*LD*	*ADD*	*XOR*			
ALU			*LD*	*ADD*	*XOR*		
WB				*LD*	*ADD*	*XOR*	

r4 written

Can relegate both problems to compiler.

Can bypass paths fix both *r4* read problems?

17