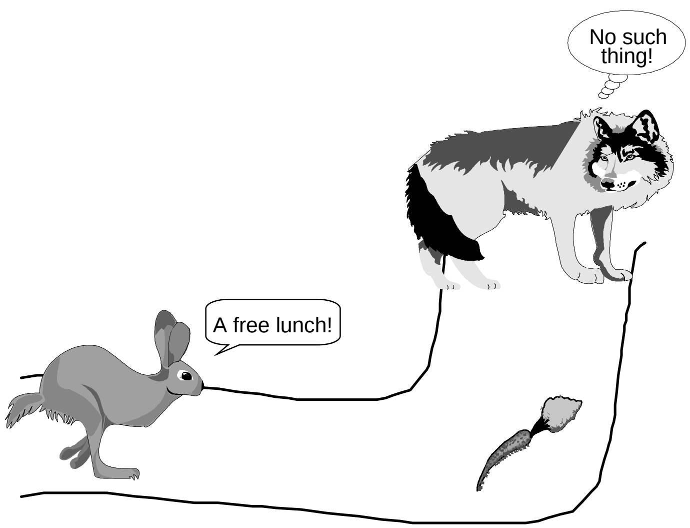
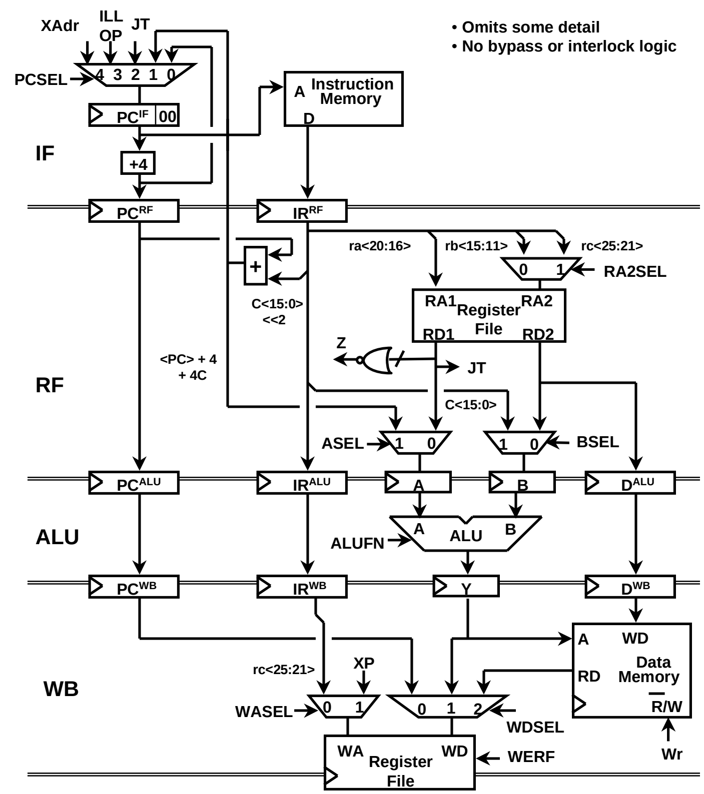
Pipelined β Processor Subtleties



1

Recap: 4-Stage β Pipeline

- Omits some detail
- No bypass or interlock logic

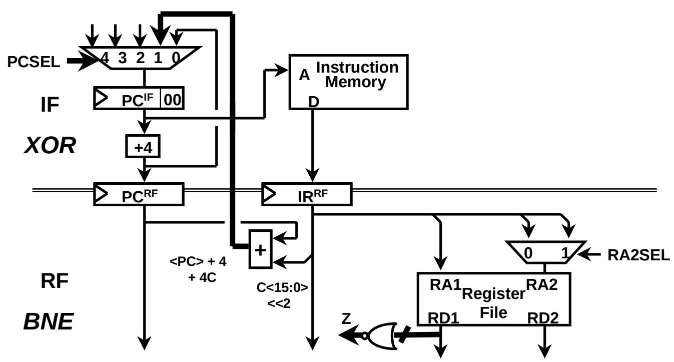


2

6004 Spring 1998: L32

Recap: Branch Hazards

	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 only in the i + 5 clock cycle, not i + 4

3

Recap: 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*****.**

4

6004 Spring 1998: L32

Loads

Consider the sequence:

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

***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*	

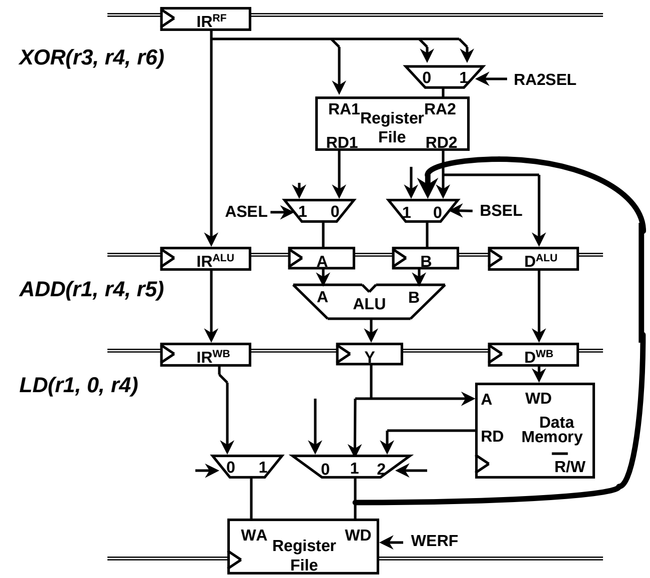
***r4* written**

Can relegate both problems to compiler.

Can bypass paths fix <u>both</u> *r4* read problems?

5

Bypass Paths



Above bypass will work for Problem 2
will not work for Problem 1

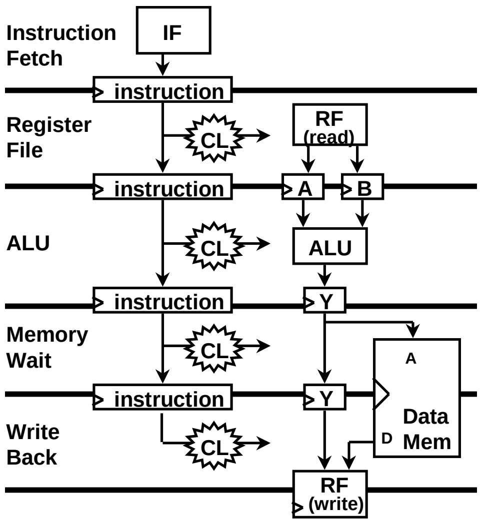
6

Load Timing

- Fact: Processors will become faster relative to memories!
  - Do we just lengthen the cycle time?
  - Alternative: Longer pipelines
- Longer pipelines by:
  - Add “Memory Wait” stages between start of read operation and return of data.
  - Build pipelined memories, so that multiple memory transactions can be in progress at once.
- Read access time for:
  - 4-Stage pipeline: 1 clock
  - 5-Stage pipeline: 2 clocks

7

5-Stage Pipeline

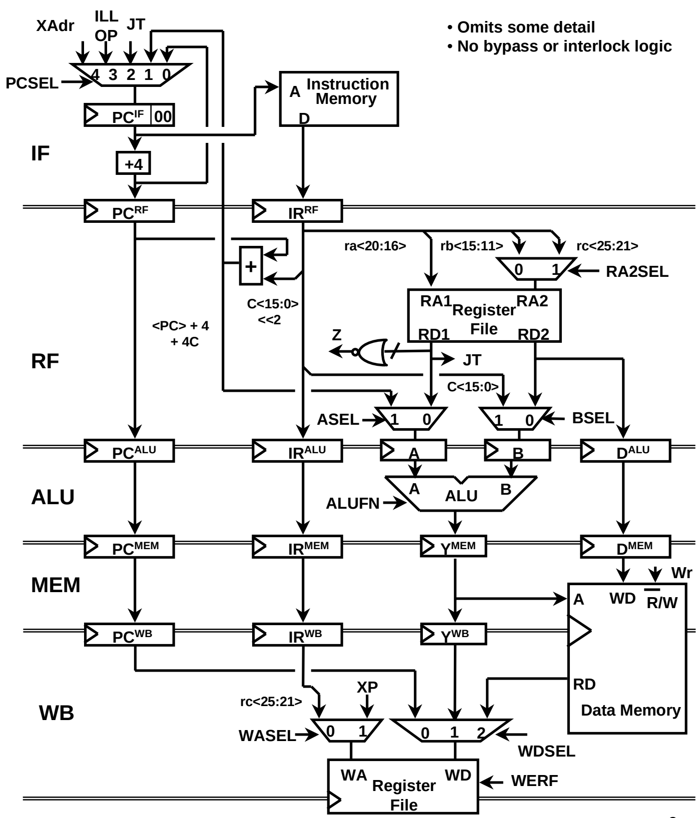


Memory access time is dominant, so break it across two stages

8

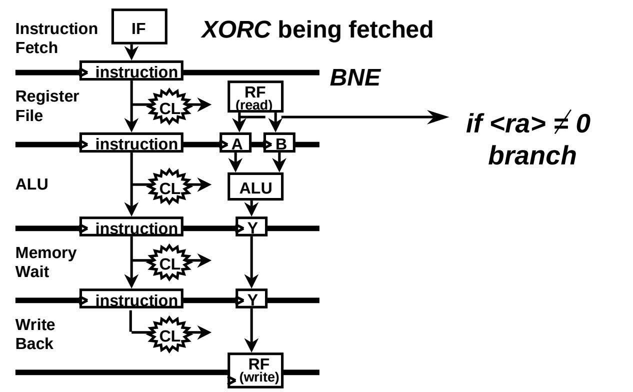
5-Stage β Pipeline

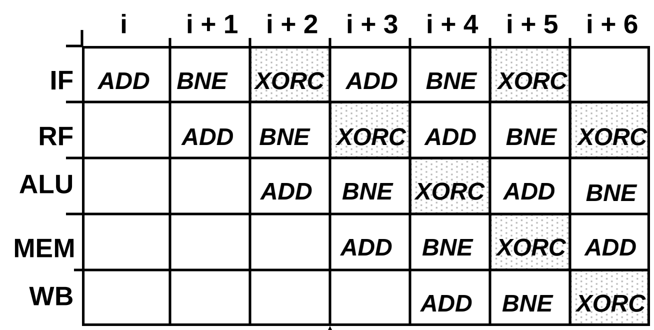
- Omits some detail
- No bypass or interlock logic



9

Branch Delay Slots



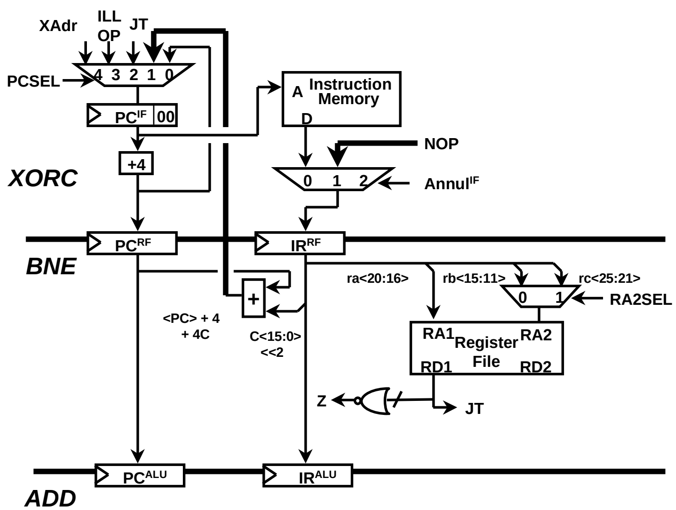


	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*ADD*	*BNE*	*XORC*	*ADD*	*BNE*	*XORC*	
RF		*ADD*	*BNE*	*XORC*	*ADD*	*BNE*	*XORC*
ALU			*ADD*	*BNE*	*XORC*	*ADD*	*BNE*
MEM				*ADD*	*BNE*	*XORC*	*ADD*
WB					*ADD*	*BNE*	*XORC*

Branch Decision Time

10

Annulling Prefetched Instructions



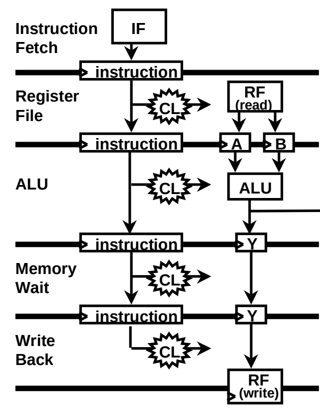
$Annul^{IF}$ = 1 if branch is taken, i.e., PCSEL = 1

Similarly for *JMP* instruction

Branch Decision Timing

Architectural issue:

- **Simple branch condition logic, e.g., test if *<ra> = 0*. Decision in RF stage.**
- **More powerful Compare-and-Branch instructions, e.g., test if *<ra> > <rb>*. Decision in ALU stage.**



Will have ____ branch delay slots and instructions to annul.

6004 Spring 1998: L32

Linkage Register Write Timing

Assume *r28 = LP* is initially *100*.

OR(r1, r2, r3)
BEQ(r31, f, r28)
NOP
...
f: *ADDC(r28, 0, r0)*
XORC(r28, 0, r1)
AND(r4, r5, r6)

***r28* read *r28* read**

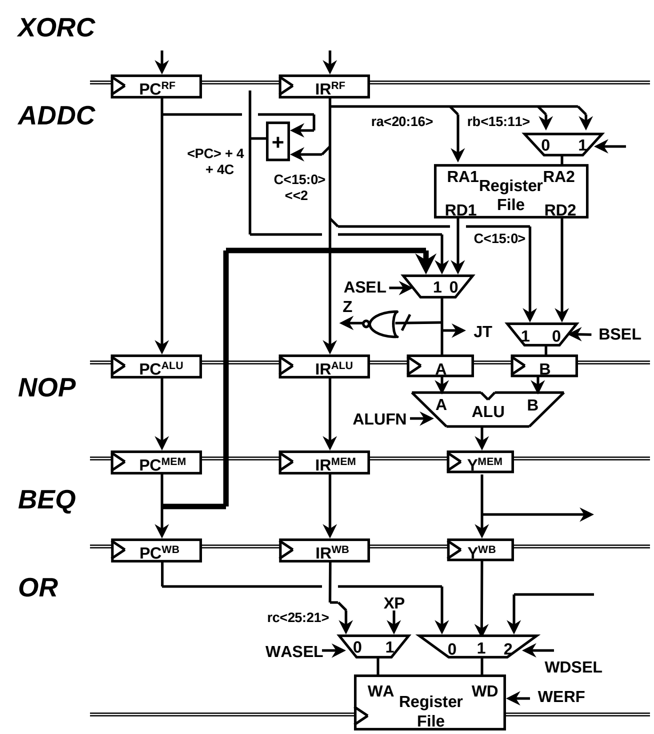
	i	i + 1	i + 2	i + 3	i + 4	i + 5	i + 6
IF	*OR*	*BEQ*	*NOP*	*ADDC*	*XORC*	*AND*	
RF		*OR*	*BEQ*	*NOP*	*ADDC*	*XORC*	*AND*
ALU			*OR*	*BEQ*	*NOP*	*ADDC*	*XORC*
MEM				*OR*	*BEQ*	*NOP*	*ADDC*
WB					*OR*	*BEQ*	*NOP*

***BEQ* decision**

r28* gets *<PC> + 4

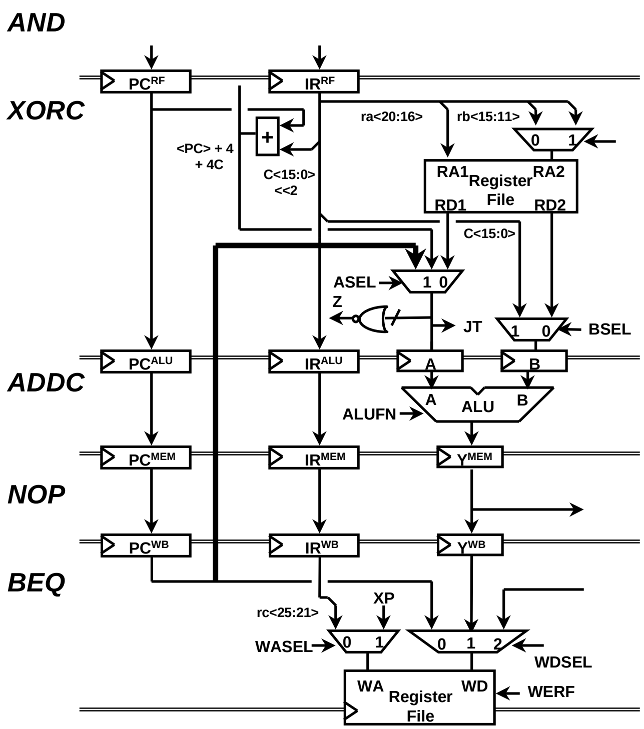
13

Branch Destination Bypass - I



14

Branch Destination Bypass - II



Exceptions

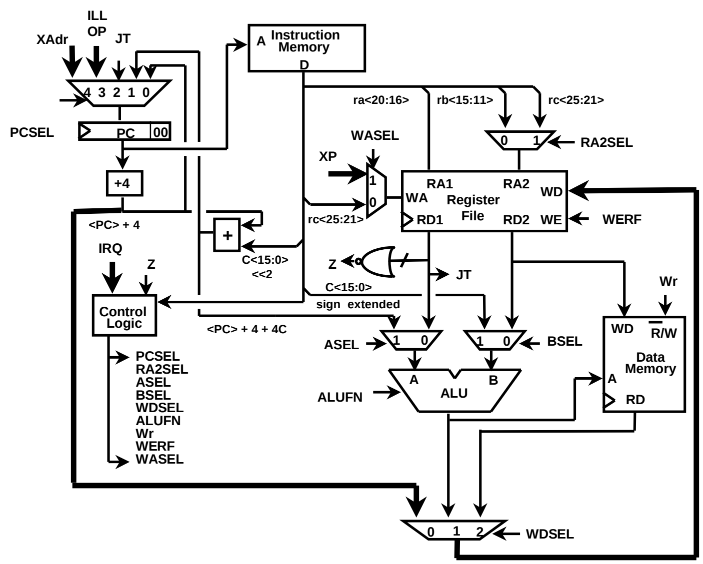
Traps: Calls to Operating System (OS)
- Write a character on console display
- Illegal opcode for instruction

Faults: Handlers for errors
- Illegal memory address

Asynchronous Interrupts: Handlers for I/O events
- Interrupt request line raised by keyboard

Effect of exceptions are to branch to appropriate handlers, service exception, and then return to program code

Exceptions



Upon detecting exception

XP **← *<PC> + 4***

PC **← ILL OP** ***or*** **XAdr**

Recall *XP* is *r30*

Illegal Opcode Traps

Trap illegal instructions to an OS handler which can

- **Interpret them in software or**
- **Print a humane error report.**

On bad opcode in RF stage

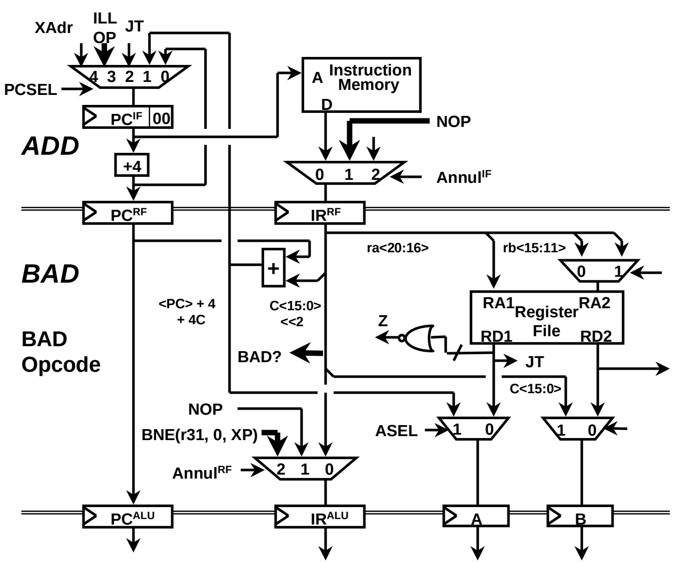
- **Select ILL OP address as next *<PC>***
- **Annul instruction in IF stage**
- **Substitute *BNE(r31, 0, XP)* for bad instruction**
- **Need bypass path to make *XP* usable by code at ILL OP address.**

User Program	**OS handler**
...	ILL OP: *ST(r0, savereg, r31)*
BAD(...)	*LD(XP, -4, r0)*
ADD(...)	*...*
...	*LD(r31, savereg, r0)*
	JMP(XP, r31)

Illegal Opcode Traps in β Pipeline

LD(XP, -4, r0)

ST(r0, savereg, r31)



19