Outline

- Pipelining & Performance
- Instruction pipeline
- Speed up
- Pipelined datapath & control
- hazards / resolution

Performance

machine organization - pipelining

$$\text{Seconds/pgm} = \text{inst/pgm} \times \text{CPI} \times \text{seconds/cycle}$$

"instruction count" ↑ Software optimization

cycle time ↑ levels of logic between registers

Examples of pipelining

- Laundry
- mfg line / assembly line
- buffet / cafeteria

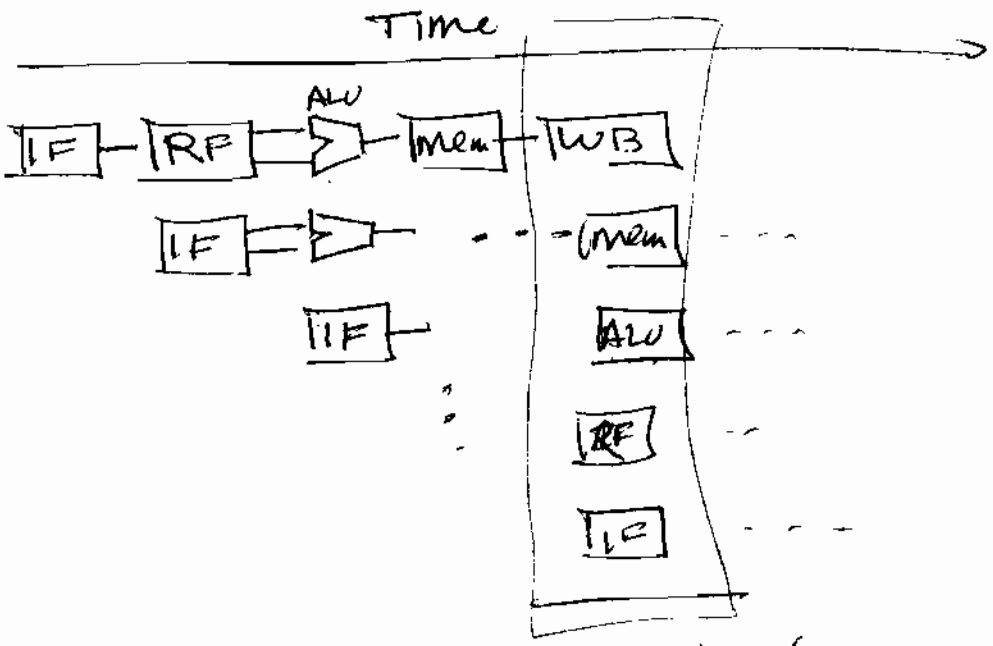
} - resources to complete the steps in a task
- multiple tasks at different stages
- use every stage at once

messed & Understood by me	Date	Invented by	Date
		Recorded by	

From Page No.

Instruction Processing Pipeline

- appears as if instruction are excated one after another, but
- multiple instructions execute at one



instruction every cycle.

- 5 instructions at once

Witnessed & Understood by me,	Date	Invented by	Date
		Recorded by	

Page No.

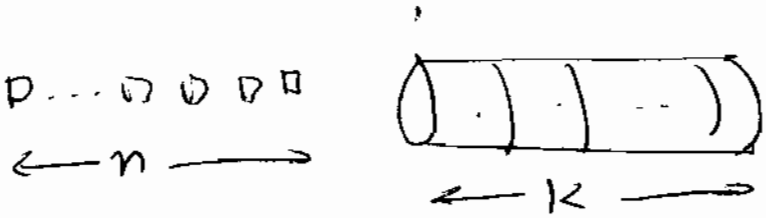
Speedup - due to enhancement X

$$= \frac{\text{Time (w/o X)}}{\text{Time (w/ X)}}$$

Ex: 1 m cycles w/o pipeline,
300 k " w/ 5 stage pipeline

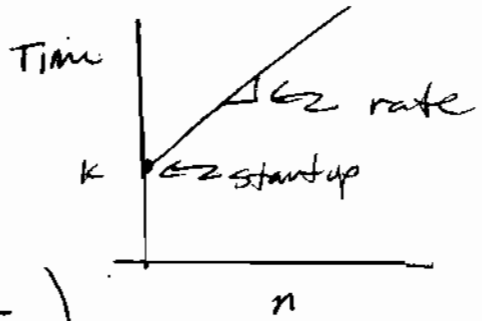
$$SU = \frac{1\,m}{.300\,m} = 3.33$$

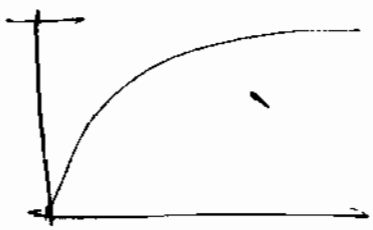
— o —



$$\text{Time}(n,k) = n+k$$

$$SU(n,k) = \frac{k \cdot n}{n+k} = k\left(\frac{n}{k+n}\right)$$



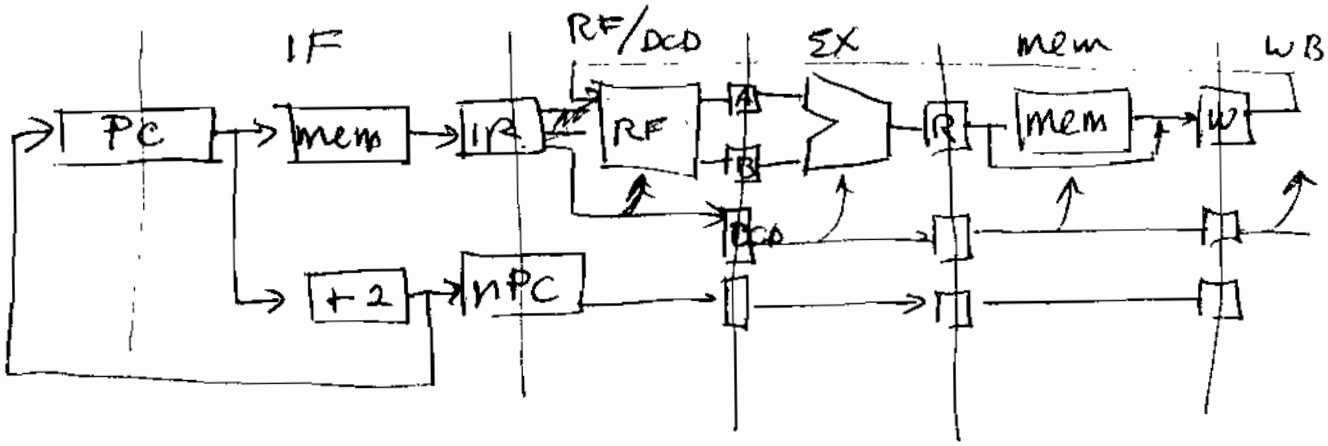


To Page No.

Witnessed & Understood by me	Date	Invented by	Date
		Recorded by	

From Page No. __

Pipelined Inst Processing in detail




```
4040   ADD   R1, R2, R3
  44   OR    R4, R5, R2
  48   LW    R5, R4(R2)
  4C   SUB   R6, R5, R2
  50   ADDI  R7, R2, 16
  54   J     foo
  58   ADDI  R8, R8, 4
```

BEQ R1, R2, foo

Control Hazard

- don't know if instruction is a jmp or branch until decode
- have already fetched the next one.

Classic: Kill the "prefetched" inst following the Jmp / branch
- Nop

Delay Slot: Execute the instruction following the jmp / br then the target

- compiler tries to put something useful there, else NOP

To Page

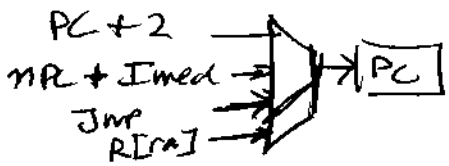
Witnessed & Understood by me.	Date	Invented by	Date
		Recorded by	

MIPS: delayed branch
- nPC relative

When is branch resolved?

- simple conditionals: zero, ~zero, eq, neq
- LT / GT take two inst

⇒ Branch condition resolved in DCD stage
- single delay slot
- compute all possible nPC and select at the last PS



Data Hazard

- Cannot use a value before it is produced
  - Real-after-write
  - data dependence

Ex
```
ADD  R1, R2, R3
OR   R4, R1, R5
```

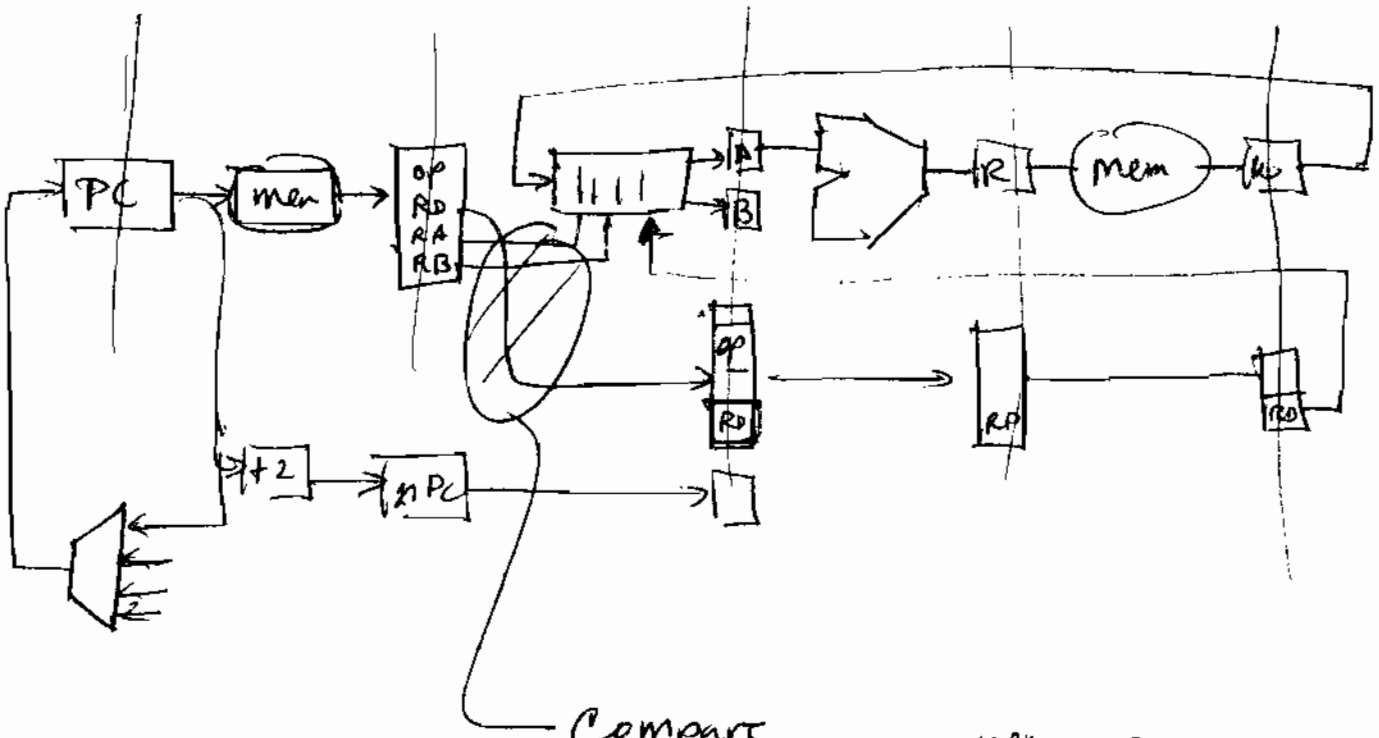
Solution: Wait

- detect the hazard
- stall dependent inst. till hazard goes away

(or force compiler to avoid it
- then organization is made visible

Witnessed & Understood by me.	Date	Invented by	Date
		Recorded by	

From Page No.



Compare

$R_A == R_D^{ex}, R_D^{mem}, R_D^{wb}$

$R_B == \quad " \quad , \quad " \quad "$

if equal

- keep inst in IR
- no PC update
- send NOP down the pipeline (Bubble)

Other data hazards

WAR

```
Add  R1, R1, R2
     R2, R3, R4
```

- read in RF, write in WB

WAW

```
ADD  R1, R1, R2
B    R1, foo
OR   R1, R3, R4
```

- instructions issue in order
  complete in order

To Page No.

Witnessed & Understood by me,	Date	Invented by	Date
		Recorded by	

Optimization: avoid pipeline stalls

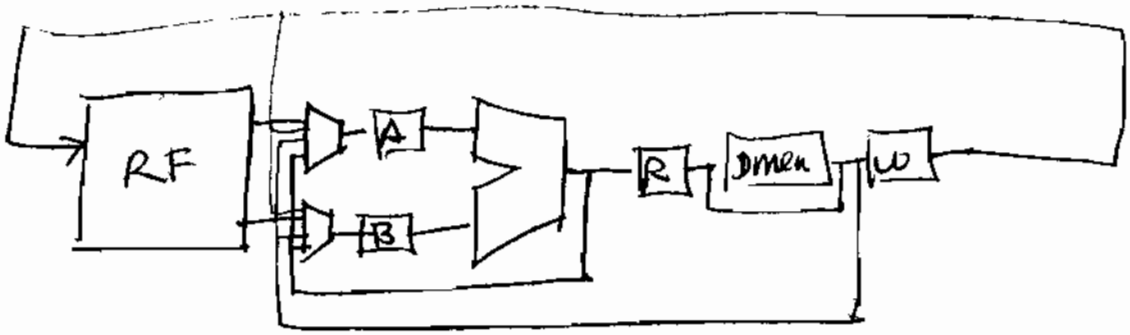
- arrange code to separate dependent instructions

```
Add  R1, R2, R3
Addi R3, R1, 16
SUB  R2, R2, 1
```

⋮

Bypass -

- Add mux to grab values in the datapath that are valid but have not been written to the register file
- Arith/Log : R or W
  Load : only W ⇐ Load delay slot



MIPS – WB in 1st half of cycle, RF in second
- additional bypass
- more MUX to other side of A/B

Witnessed & Understood by me.	Date	Invented by	Date
		Recorded by	