Pipelining

CS 281 Lecture Notes
12/5/2016

Pipelining

① Start with Examples

Toy Making
- Cut
- Sand
- Add Wheels/Decorations
- Paint

Laundry
- Wash
- Dry
- Fold

Start to Finish Time -- latency

Did it improve with pipelining?

No!

What did improve?

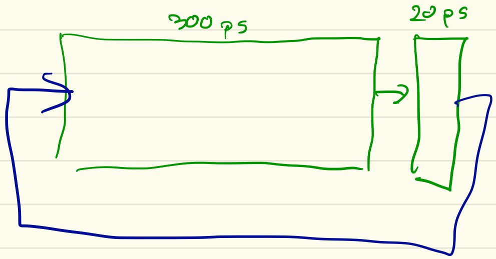
Suppose 50 people to do laundry: Total Time?

Non-pipelined $50 \cdot (1 + 3/4 + \frac{1}{2})$

Pipelined ~52

1	Jill	W D F
2	Joe	W D F
3	Barb	W D F
48	Sue	W D F
49	⋮	W D F
50	prime	W D F

Digital Logic / Sequential Systems



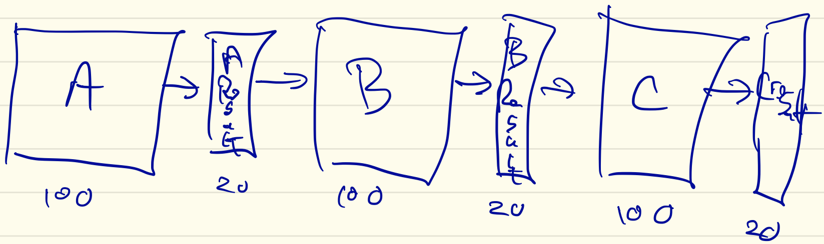
320 ps shortest time for 1 cycle

1 unit work/320 ps, so Throughput rate is $\frac{1}{320 \cdot 10^{-12}} \approx 3.125 \cdot 10^9 \frac{\text{units work}}{\text{sec}}$

Suppose, can divide into 3 stages of 100 ps

Need ability to "hold" info that is input to 1 stage ⇒ memory / Registers

Note: <u>different</u> regs



Now latency : 360 ps

1 unit work / 120 ps $= \frac{1}{120 \cdot 10^{-12}} = 8.33 \cdot 10^9$

Pipeline Diagram

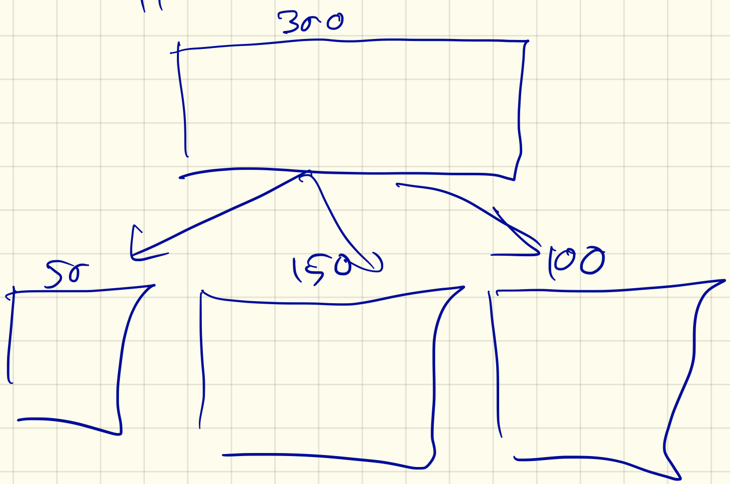
OP1	A	B	C		
OP2		A	B	C	
OP3			A	B	C

time →

Vert slice:
Concurrency at instant in time

Horiz. slice
view by operation

What happens when we cannot evenly divide?



Clock determined by longest leg

$150 + 20 \Rightarrow$ shortest cycle

Latency:

\# stages $\times$ cycle $= 170 \cdot 3 = 510$

Throughput:

1 instruction/unit / 170ps

$$= \frac{1}{170 \cdot 10^{-12}} = 5.88 \cdot 10^{9} \text{ ops/sec}$$

Challenge: Balance

What is limit? How deeply can we pipeline?

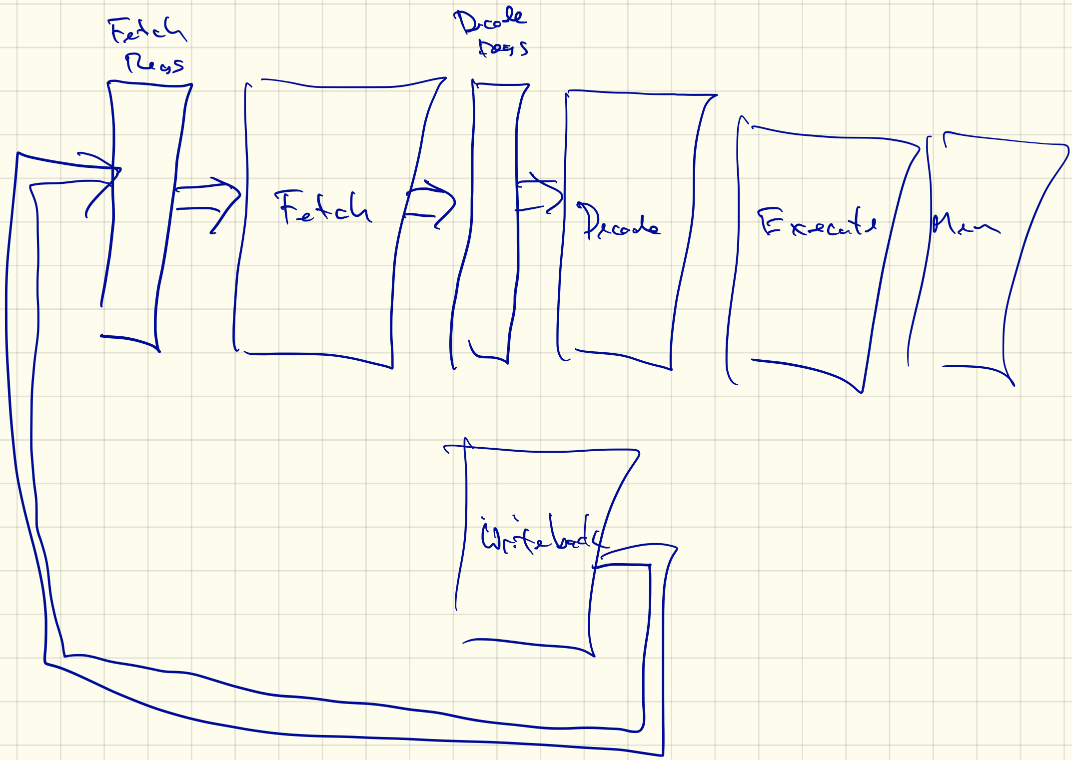
20 ps loading regs

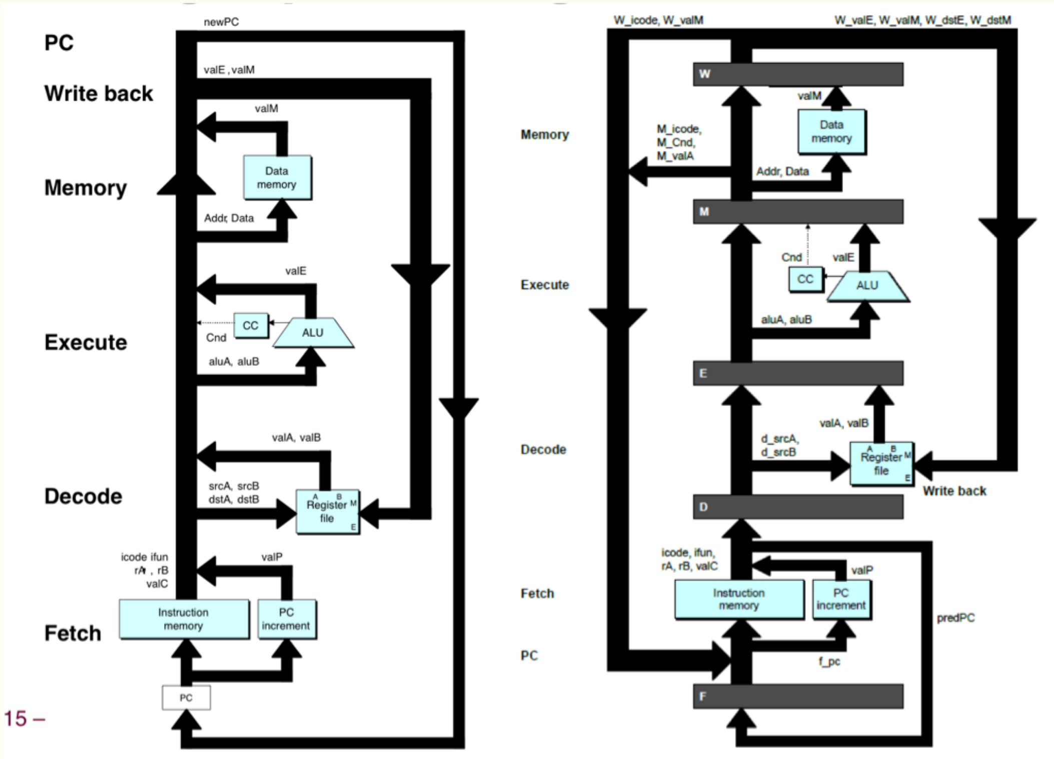
			Latency	Tput
Say we do	6 stages	50 + 20	420	$\frac{1}{70 \cdot 10^{-12}}$
	10 stages	30 + 20	500	$\frac{1}{50 \cdot 10^{-12}}$

Holy Grail? No Problem in dependencies

→ Data Hazard

Goal CPU Pipeline





Fetch

- Select current PC
- Read instruction
- Compute incremented PC

Decode

- Read program registers

Execute

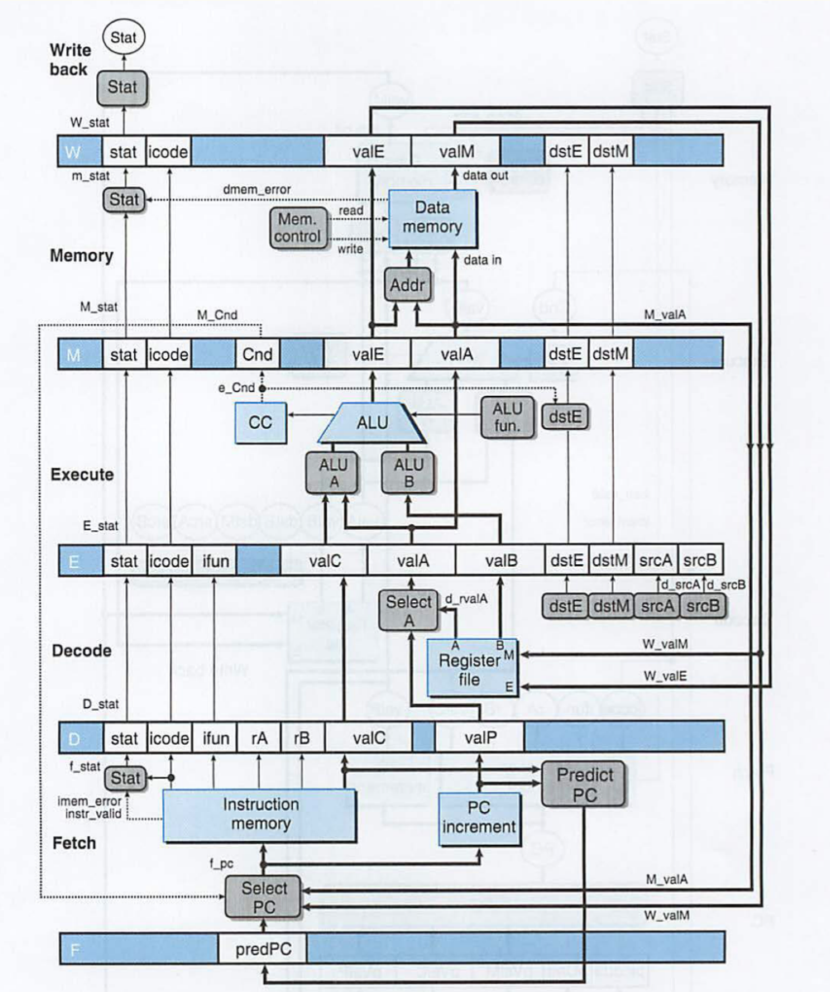
- Operate ALU

Memory

- Read or write data memory

Write Back

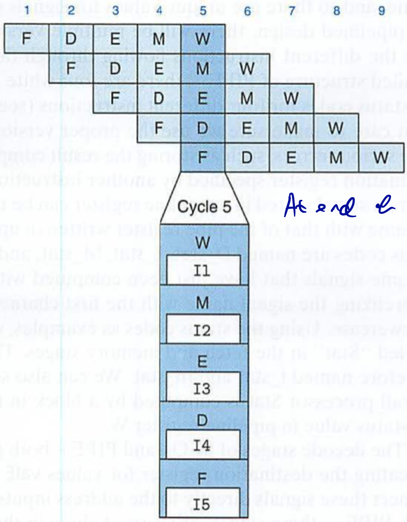
- Update register file



Write back
Stat
Stat
W_stat
W
stat
icode
valE
valM
dstE
dstM
m_stat
data out
Stat
dmem_error
Data memory
Mem. control
read
write
Memory
Addr
data in
M_stat
M_Cnd
M_valA
M
stat
icode
Cnd
valE
valA
dstE
dstM
e_Cnd
CC
ALU
ALU fun.
dstE
ALU A
ALU B
Execute
E_stat
E
stat
icode
ifun
valC
valA
valB
dstE
dstM
srcA
srcB
d_srcA
d_srcB
Select A
d_rvalA
dstE
dstM
srcA
srcB
Decode
A
B
M
Register file
E
W_valM
W_valE
D_stat
D
stat
icode
ifun
rA
rB
valC
valP
f_stat
Stat
Predict PC
imem_error
instr_valid
Instruction memory
PC increment
Fetch
f_pc
Select PC
M_valA
W_valM
F
predPC

time

```
irmovq    $1,%rax    #I1
irmovq    $2,%rbx    #I2
irmovq    $3,%rcx    #I3
irmovq    $4,%rdx    #I4
halt                 #I5
```



At end of cycle