CS 61C:
Great Ideas in Computer Architecture

Lecture 12: *Control & Operating Speed*

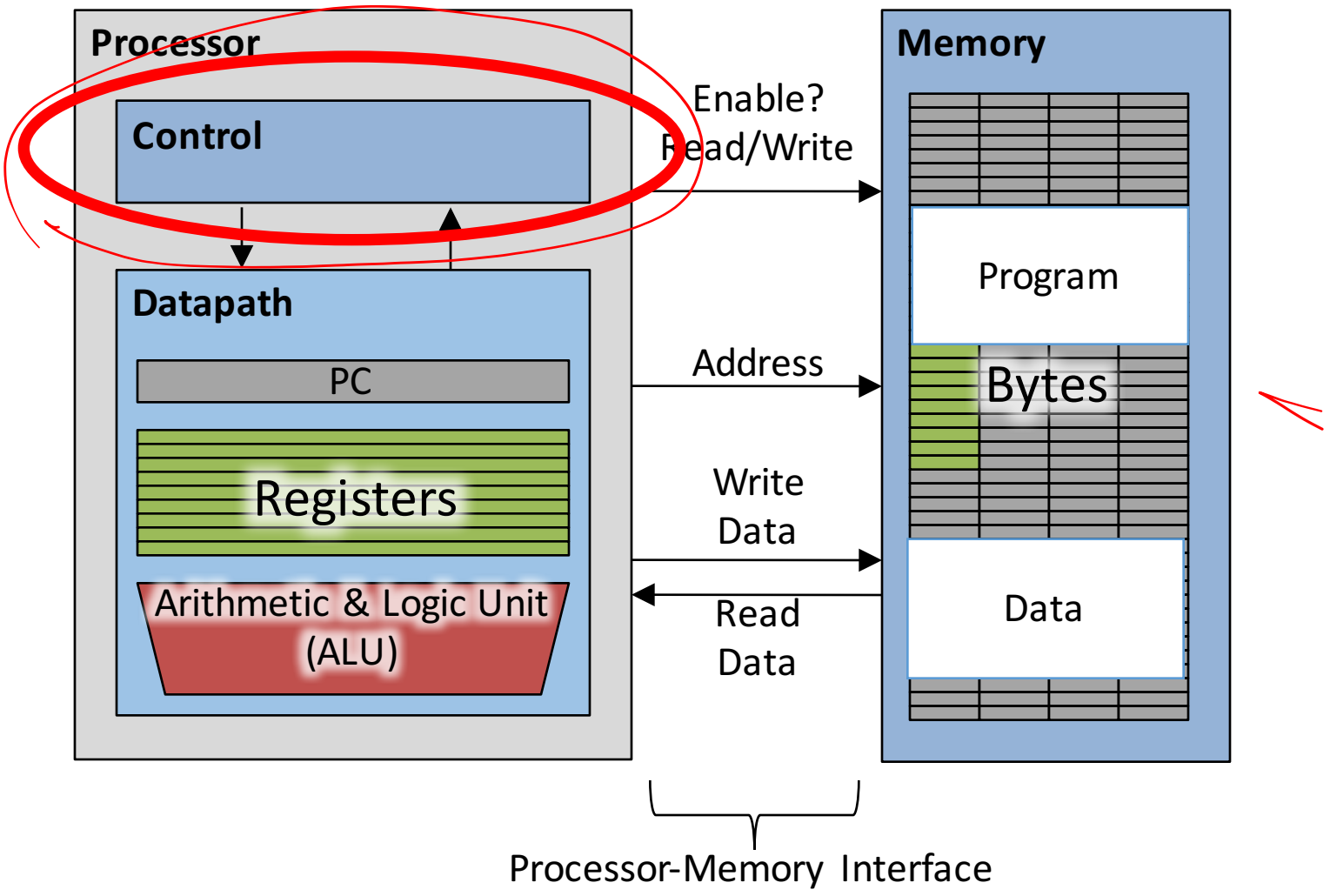
Bernhard Boser & Randy Katz

http://inst.eecs.berkeley.edu/~cs61c

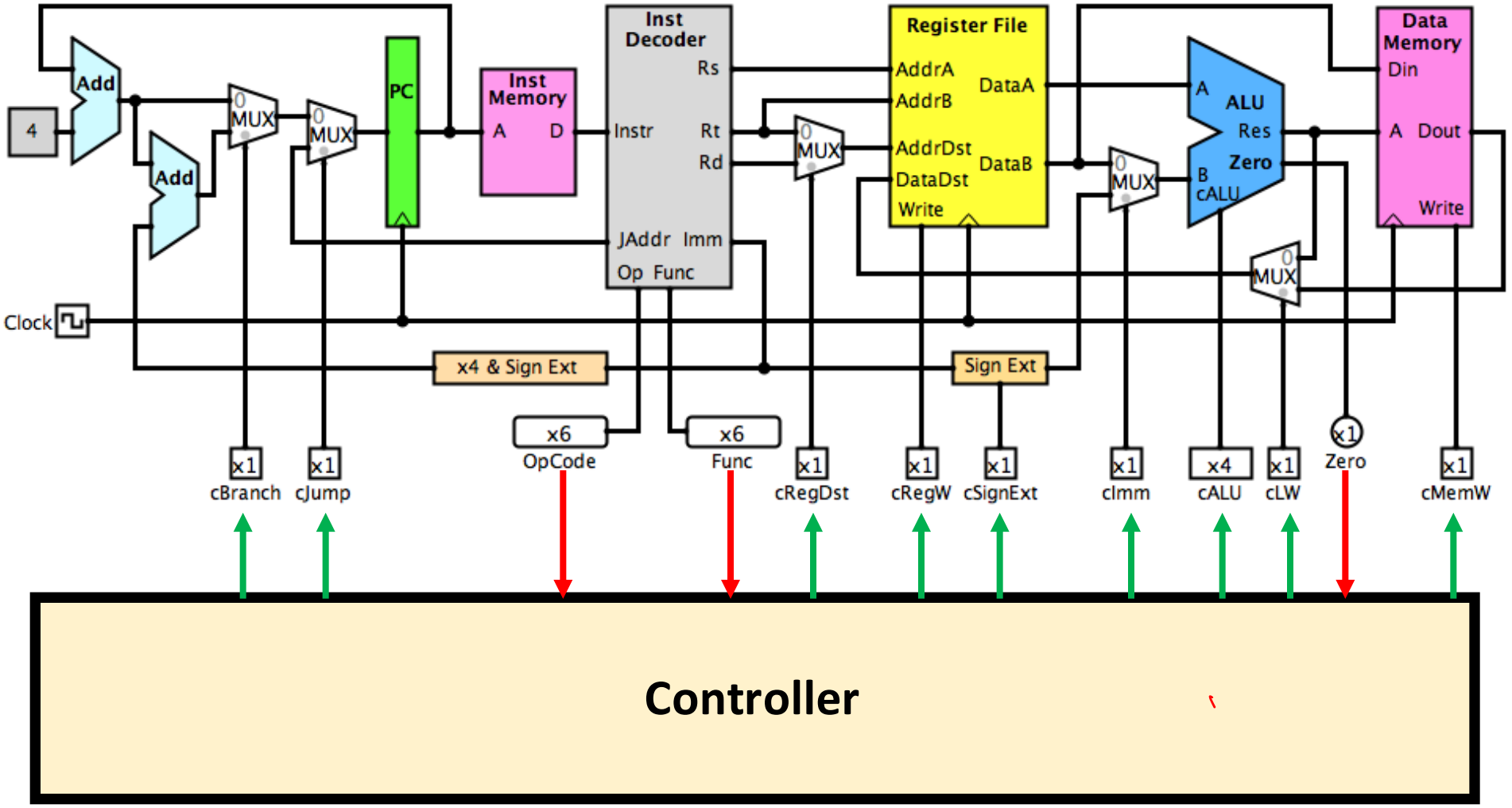
Agenda

- **Controller**
- Instruction Timing
- Performance Measures
- Introduction to Pipelining
- Pipelined MIPS Datapath
- And in Conclusion, ...

Processor


Processor
Control
Datapath
PC
Registers
Arithmetic & Logic Unit (ALU)
Memory
Program
Bytes
Data
Enable?
Read/Write
Address
Write Data
Read Data
Processor-Memory Interface

Controller


Add
4
Add
0
MUX
0
MUX
PC
Inst
Memory
A
D
Inst
Decoder
Rs
Instr
Rt
Rd
JAddr
Imm
Op
Func
0
MUX
Register File
AddrA
AddrB
DataA
AddrDst
DataB
DataDst
Write
0
MUX
A
ALU
Res
Zero
B
cALU
Data
Memory
Din
A
Dout
Write
0
MUX
Clock
x4 & Sign Ext
Sign Ext
x6
OpCode
x6
Func
x1
cBranch
x1
cJump
x1
cRegDst
x1
cRegW
x1
cSignExt
x1
cImm
x4
cALU
x1
cLW
x1
Zero
x1
cMemW
Controller

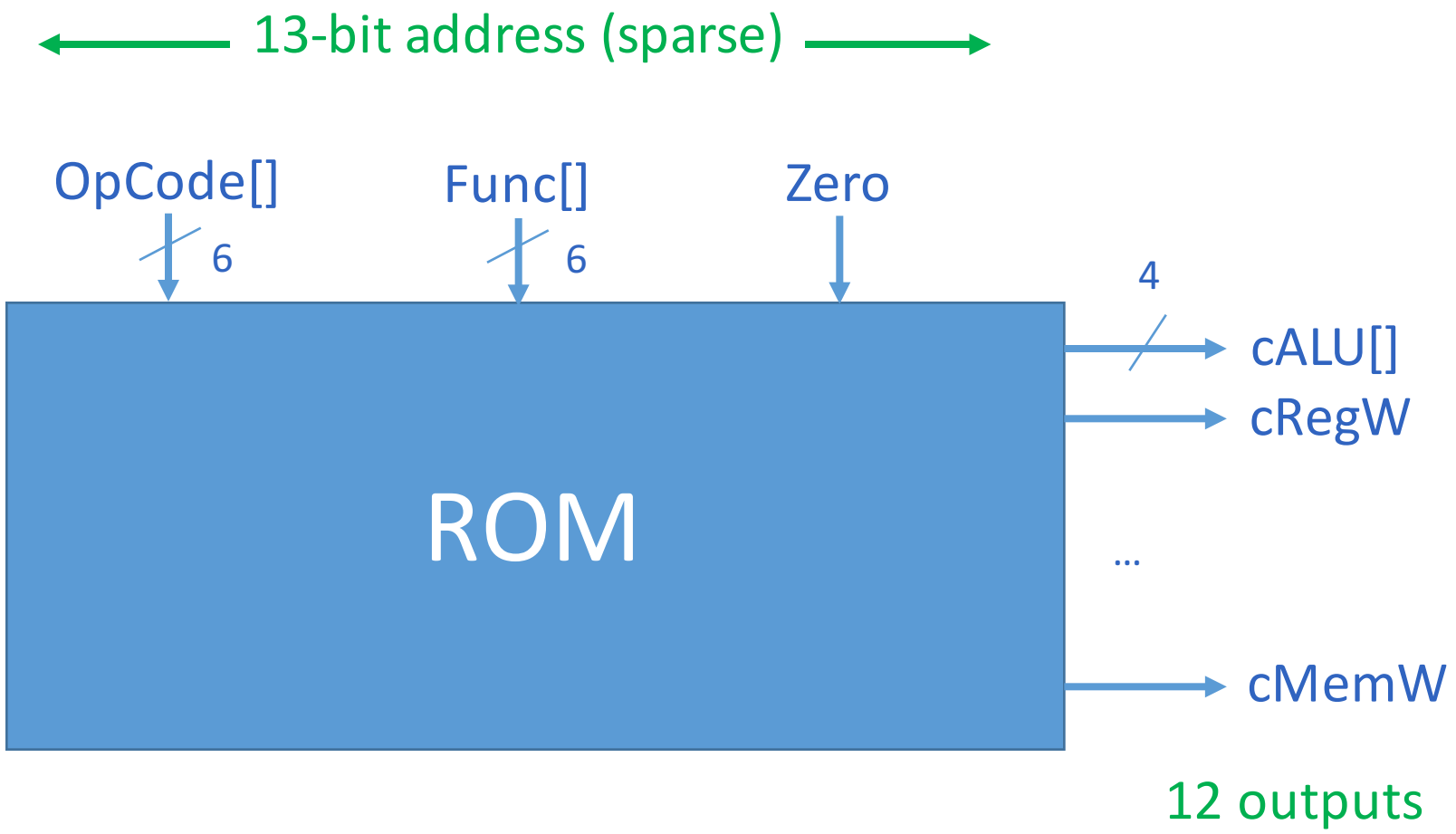
Control: Truth Table

Instr	Op	Func	Zero	cALU	cRegW	cRegDst	cSignExt	cImm	cBr	cJ	cLW	cMemW
addu	0x0	0x20	X	add	1	0	0	0	0	0	0	0
sub	0x0	0x22	X	sub	1	0	0	0	0	0	0	0
or	0x0	0x25	X	sub	1	0	0	0	0	0	0	0
slt	0x0	0x2a	X	slt	1	0	0	0	0	0	0	0
addiu	0x8	X	X	add	1	1	1	1	0	0	0	0
ori	0xd	X	X	or	1	1	1	1	0	0	0	0
beq	0x4	X	1	sub	0	X	X	0	1	0	X	0
beq	0x4	X	0	sub	0	X	X	0	0	0	X	0
bne	0x5	X	0	sub	0	X	X	0	1	0	X	0
bne	0x5	X	1	sub	0	X	X	0	0	0	X	0
j	0x2	X	X	X	0	X	X	X	X	1	X	0
lw	0x23	X	X	add	1	1	1	1	0	0	1	0
sw	0x2b	X	X	add	0	X	1	1	0	0	X	1
dowhile		X	1	add	1	1	X	0	1	0	0	0
dowhile		X	0	add	1	1	X	0	0	0	0	0

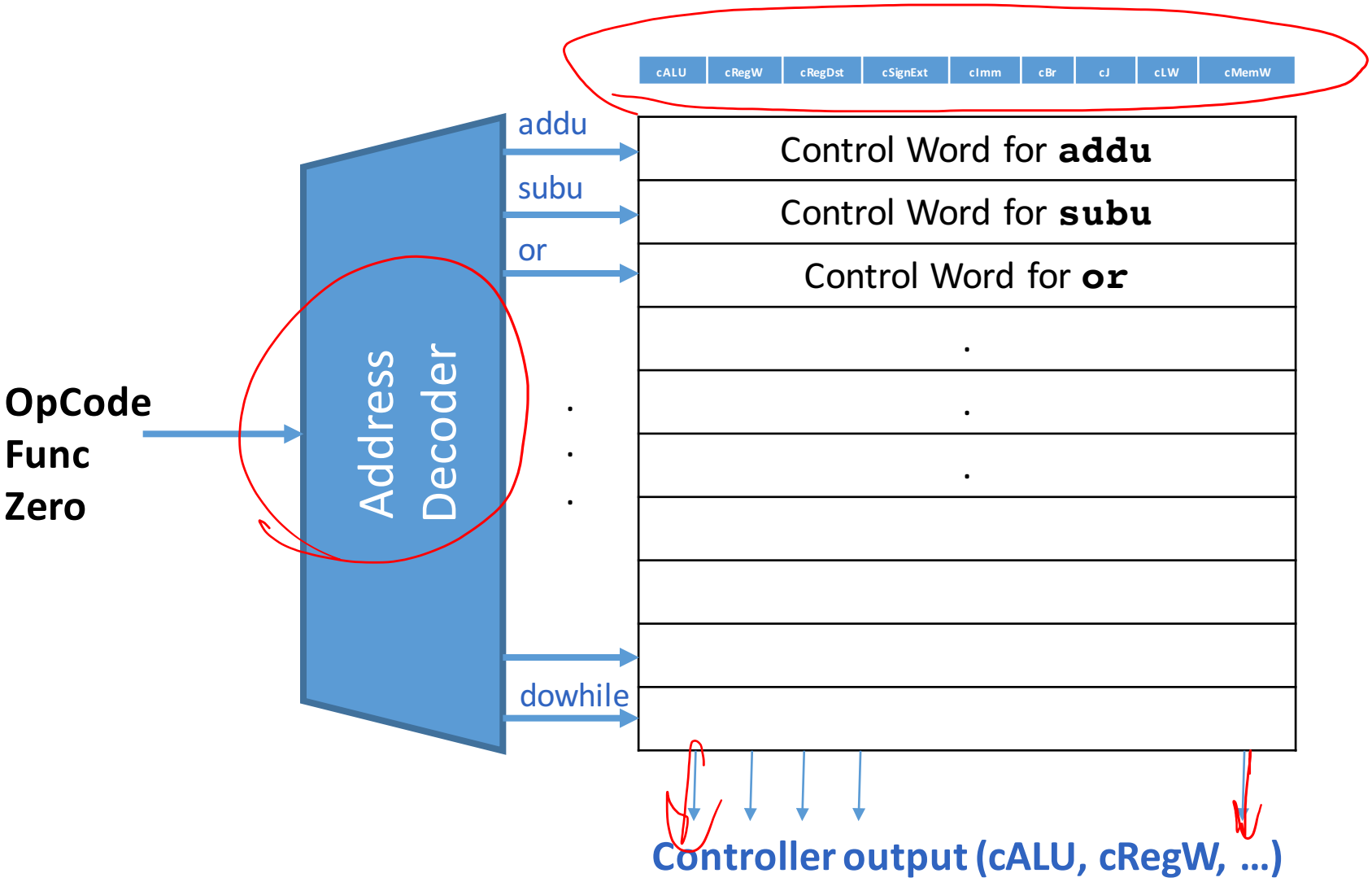
Control Realization Options

- Combinatorial Logic
- ROM
  - "Read Only Memory"
  - Regular structure
  - Can be easily reprogrammed
    - fix errors
    - add instructions

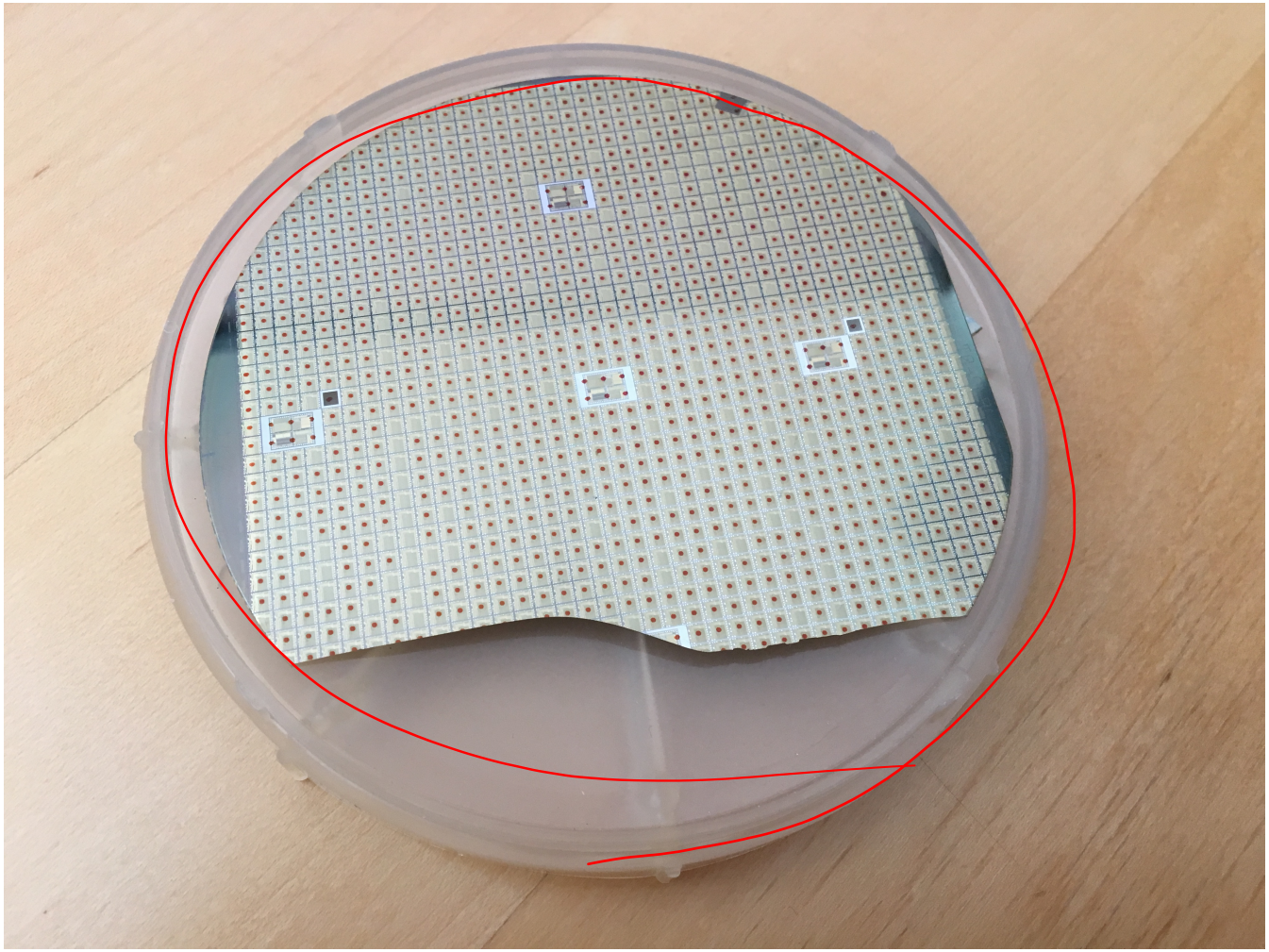
ROM-based Control



ROM Controller Implementation



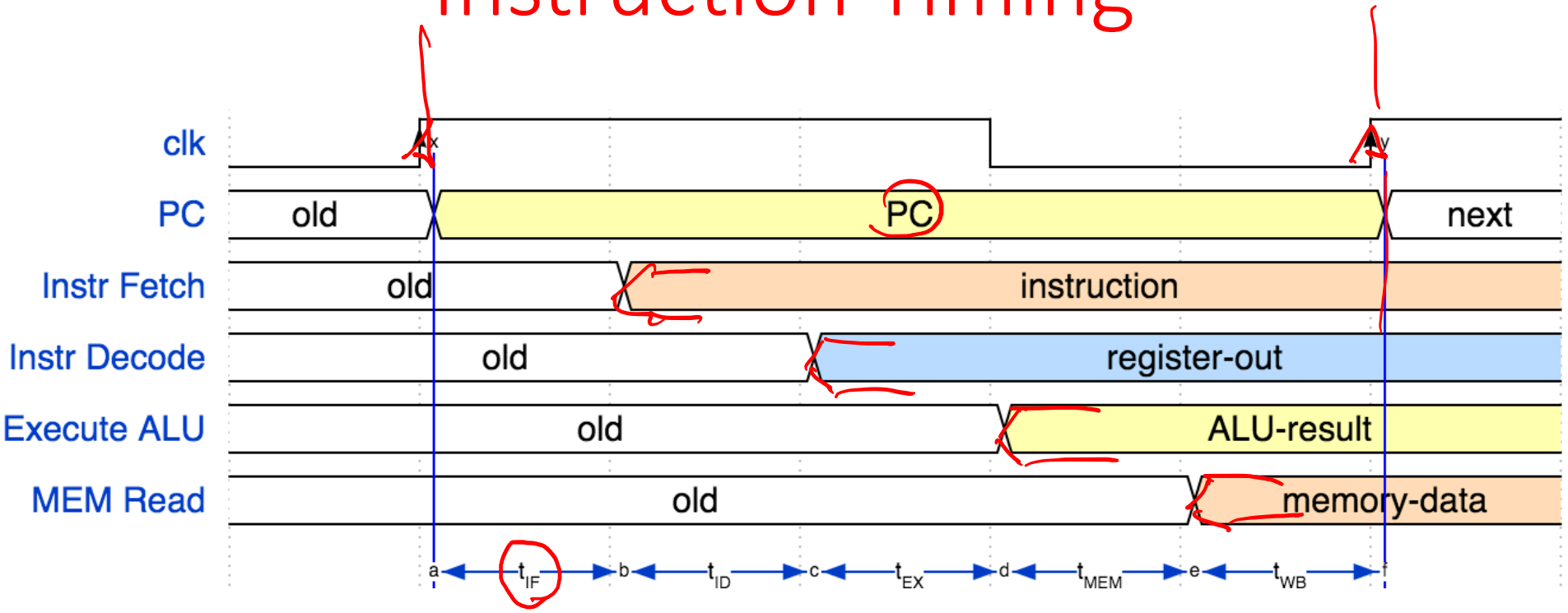
And finally, a chip!



Agenda

- Controller
- **Instruction Timing**
- Performance Measures
- Introduction to Pipelining
- Pipelined MIPS Datapath
- And in Conclusion, ...

Instruction Timing



IF	ID	EX	MEM	WB	Total
I-MEM	Reg Read	ALU	D-MEM	Reg W	
200 ps	100 ps	200 ps	200 ps	100 ps	800 ps

Instruction Timing

Instr	IF = 200ps	ID = 100ps	ALU = 200ps	MEM=200ps	WB = 100ps	Total
add	X	X	X		X	600ps
beq	X	X	X			500ps
j	X	X				300ps
lw	X	X	X	X	X	800ps
sw	X	X	X	X		700ps

- Maximum clock frequency
  - f_{max} = 1/800ps = 1.25 GHz
- Most blocks idle most of the time
  - E.g. $f_{max,ALU}$ = 1/200ps = 5 GHz!
  - How can we keep ALU busy all the time
  - 5 Mio adds/sec, rather than just 1.25 Mio?
  - Idea: Factories use 3 shift work – equipment is always busy!

Agenda

- Controller
- Instruction Timing
- **Performance Measures**
- Introduction to Pipelining
- Pipelined MIPS Datapath
- And in Conclusion, ...

Performance Measures

- ”Our” MIPS executes instructions at 1.25 GHz
  - 1 instruction every 800 ps

- Can we improve its performance?
  - What do we mean with this statement?
  - Not so obvious:
    - Quicker response time?
    - More jobs per time (e.g. web requests)?
    - Longer battery life?

Transportation Analogy

	Sports Car	Bus
Passenger Capacity	2	50
Travel Speed	200 mph	50 mph
Gas Mileage	5 mpg	2 mpg

50 Mile trip:

	Sports Car	Bus
Travel Time	15 min	60 min
Time for 100 passengers	750 min	120 min
Gallons per passenger	5 gallons	0.5 gallons

Computer Analogy

Transportation	Computer
Trip Time	Program execution time: e.g. time to update display
Time for 100 passengers	Throughput: e.g. number of server requests handled per hour
Gallons per passenger	Energy per task*: e.g. how many movies you can watch per battery charge or energy bill for datacenter

* Note: power is not a good measure since CPU power dissipation varies considerably depending on CPU state: clock speed, # of active cores, coprocessors (e.g. graphics unit), …

Program Execution Time (P&H 1.6)

$$t_{exec} = \frac{\text{Seconds}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \text{CPI} \times t_{cycle} = \frac{1}{f_s}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{CPI}}{f_s}$$

Key: fast execution requires product to be short, not just individual factors!

Instructions per Program

$$t_{exec} = \frac{\textbf{Instructions}}{\textbf{Program}} \times \frac{\text{CPI}}{f_S}$$

Determined by

- Task
- Algorithm, e.g. *O(N²)* vs *O(N)*
- Programming language
- Compiler
- Processor

(Average) Clock cycles per Instruction

$$t_{exec} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\textbf{CPI}}{f_S}$$

Determined by

- Processor
- E.g. for "our" single cycle MIPS design, CPI = 1
- Complex instructions (e.g. **`strcpy`**), CPI >> 1
- Superscalar processors, CPI < 1 (next lecture)

Clock Rate

$$t_{exec} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{CPI}}{\mathbf{f_S}}$$

Determined by

- Processor architecture, e.g. pipelining
- Instruction set
  - complex instructions are difficult to make fast
- Technology (e.g. 14nm versus 28nm)
- Power budget

Speed Tradeoff Example

- For some task (e.g. image compression) …

	Processor A	Processor B
# Instructions	1 Million	1.5 Million
Average CPI	2.5	1
Clock rate f_s	2.5 GHz	2 GHz
Execution time	1 ms	0.75 ms

Processor B is faster for this task, despite executing more instructions and having a lower clock rate!

Energy per Task

$$E_{prog} = \frac{\text{Energy}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Energy}}{\text{Instruction}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{C}{2} \times V_{sup}^{2}$$

"Capacitance", technology, processor features e.g. # of cores

Supply voltage, e.g. 1V

For low energy (power), want low supply V_{sup} and low C

Energy Tradeoff Example

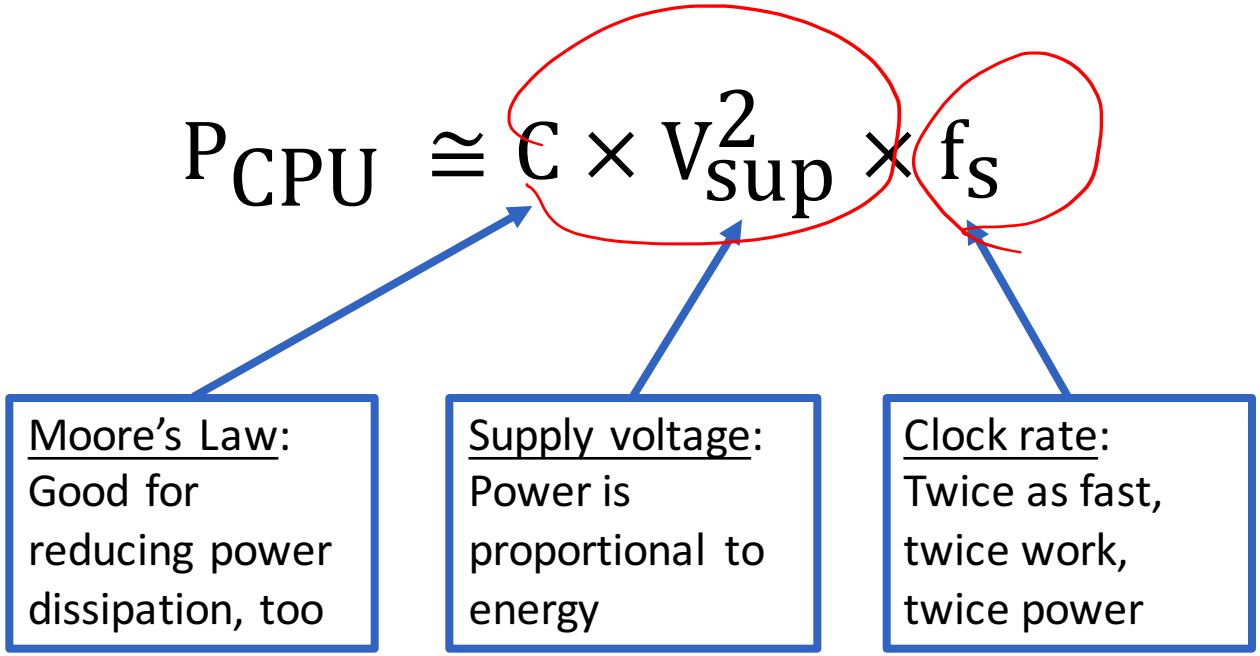
- "Next generation" processor
  - C (Moore's Law): -15 %
  - Supply voltage, V_{sup}: -15 %
  - Energy consumption: $1 - (1-0.85)^3 = -39\ \%$

- Significantly improved energy efficiency thanks to
  - Moore's Law AND
  - Reduced supply voltage

Power Dissipation

- CPU Power dissipation:

$$P_{CPU} \cong C \times V_{sup}^2 \times f_s$$

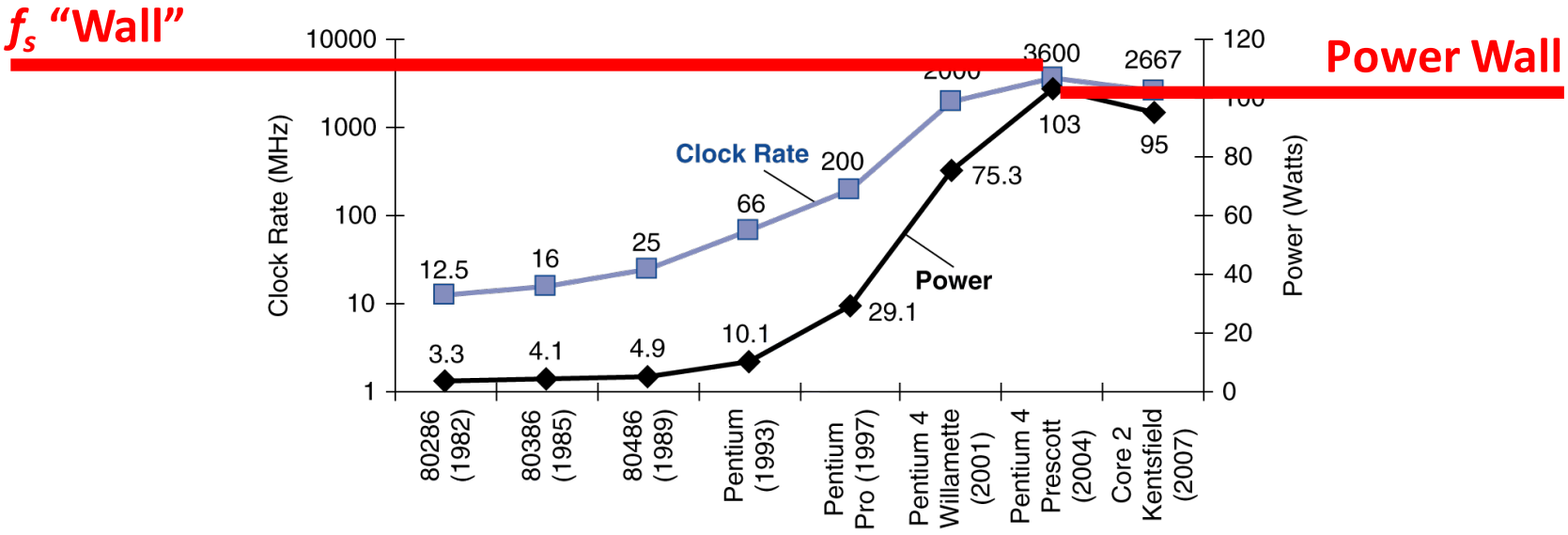


Power Wall (P&H 1.7)

$$P_{CPU} \cong C \times V_{sup}^2 \times f_s$$

Practical & Economical Maximum: ~ 100 W per chip	Moore's Law: Good for reducing power dissipation, too	Supply voltage: Power is proportional to energy	Clock rate: Twice as fast, twice work, twice power
Changes between generations:			
Before ~ 2004	⇓	⇓	⇑
After 2004	⇓	**Constant ~ 1V**	**approx. constant**

CPU Performance Trends



- Processor clock rates fs no longer increasing
  - Power dissipation would be prohibitive (chip would "melt")
- Program speed no longer increasing automatically with each new processor generation
  - Big improvements are still possible
  - Doing so requires insight into how the processor works
  - CS 61C gives you this insight & shows you how to apply it!

Your Turn

Choosing a processor for a datacenter:
Which one results in lowest total power dissipation?
(energy bill)

	A	B	C	D	E
Clock rate	2 GHz	4 GHz	3 GHz	2.5 GHz	3 GHz
CPI	0.5	2	1	1	0.5
Power	40 W	100 W	60 W	50 W	80 W

Your Turn

Choosing a processor for a datacenter:
Which one results in lowest total power dissipation?

Metric: instructions per Watt = f_s / CPI / Power

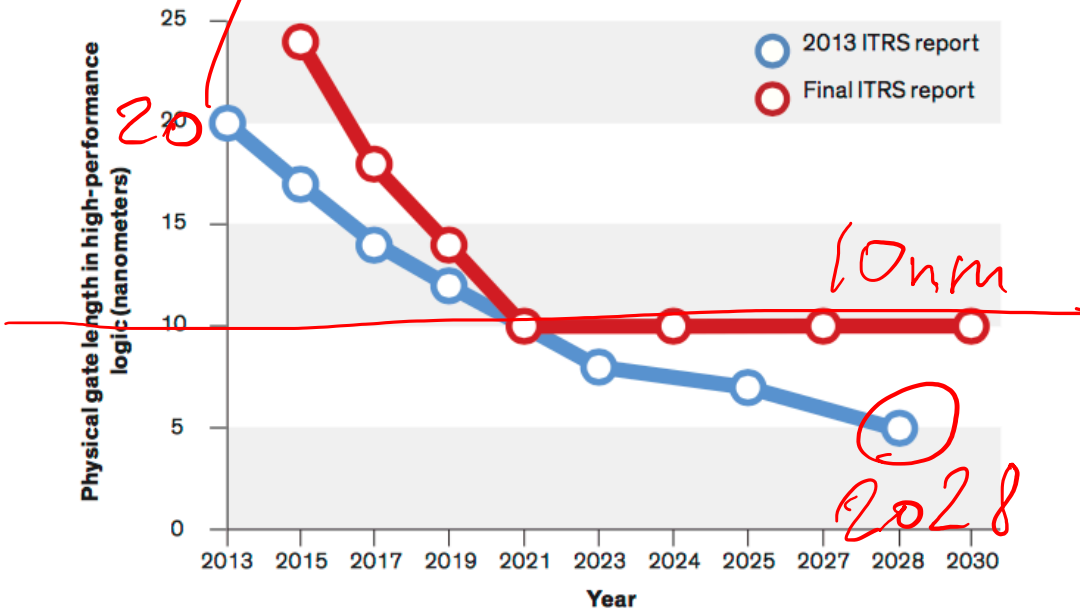
	A	B	C	D	E
Clock rate	2 GHz	4 GHz	3 GHz	2.5 GHz	3 GHz
CPI	0.5	2	1	1	0.5
Power	40 W	100 W	60 W	50 W	80 W
Instr / W	100 M/W	20 M/W	50 M/W	50 M/W	75 M/W

In the News

TRANSISTORS COULD STOP SHRINKING IN 2021

A key industry report forecasts an end to traditional scaling of transistors

IEEE Spectrum Magazine, September 2016, pp. 9



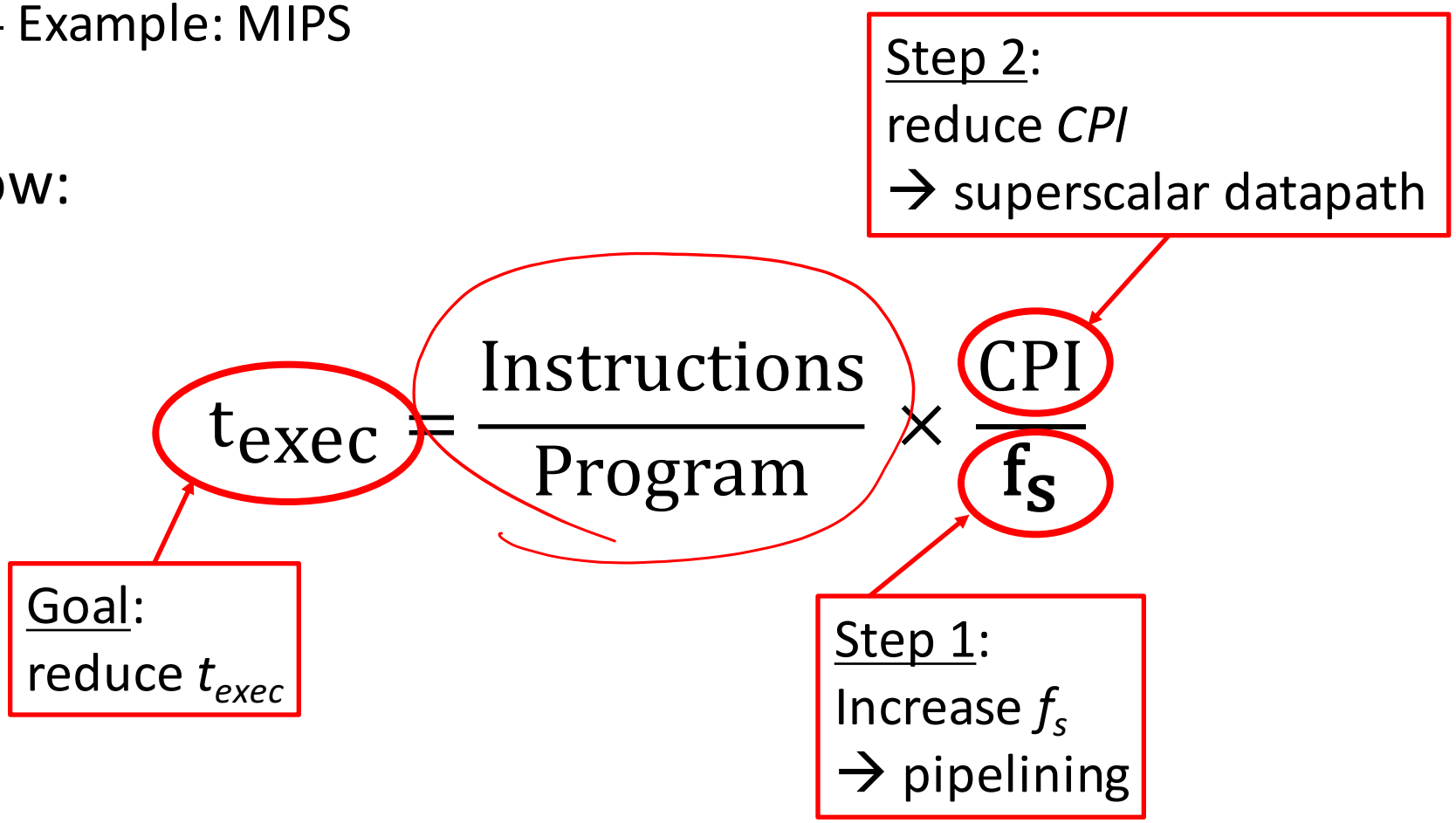
International Technology Roadmap for Semiconductors

Agenda

- Controller
- Instruction Timing
- Performance Measures
- **Introduction to Pipelining**
- Pipelined MIPS Datapath
- And in Conclusion, ...

The Plan

- Fast computer
  - Example: MIPS
- How:



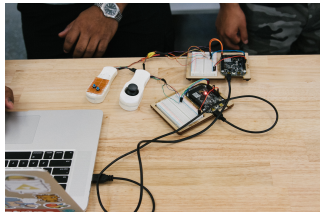
Pipelining

- A familiar example:
  - Getting a university degree


Year 1


Year 2

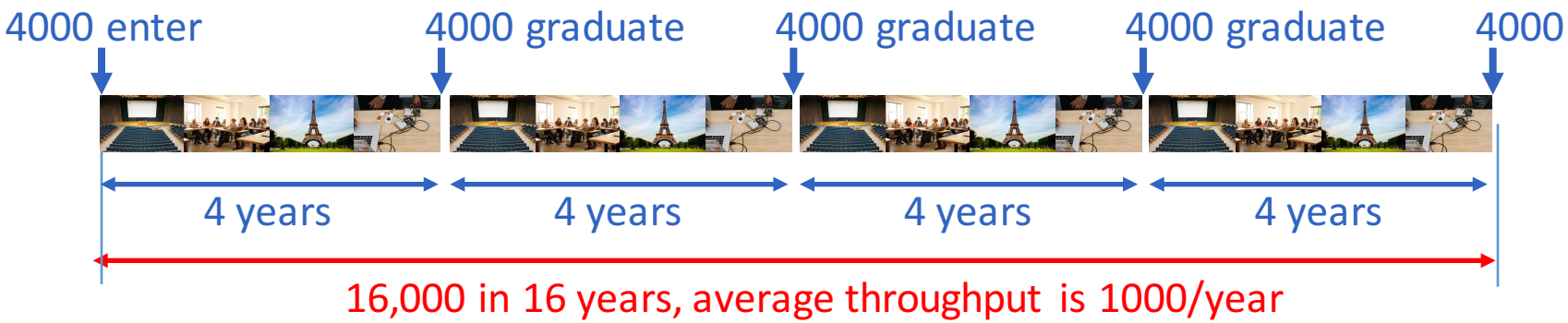

Year 3


Year 4

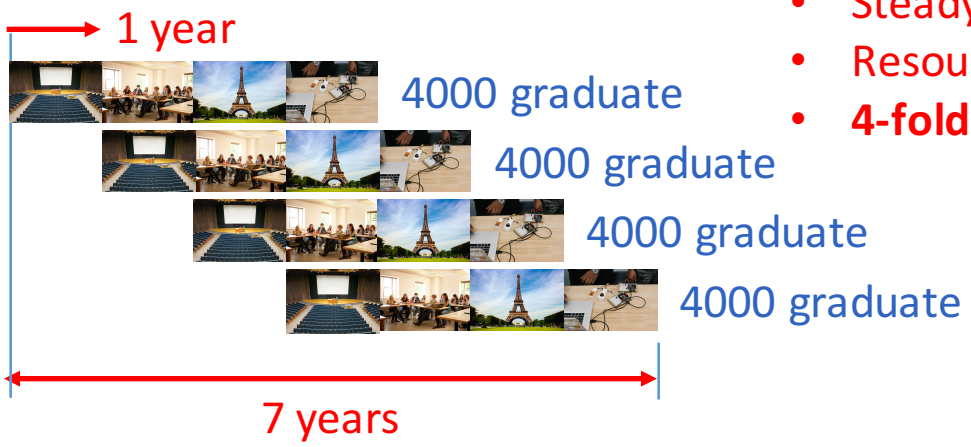
- Shortage of Computer scientists (your startup is growing):
  - How long does it take to educate 16,000?

Computer Scientist Education

- Option 1: ***serial***



- Option 2: ***pipelining***



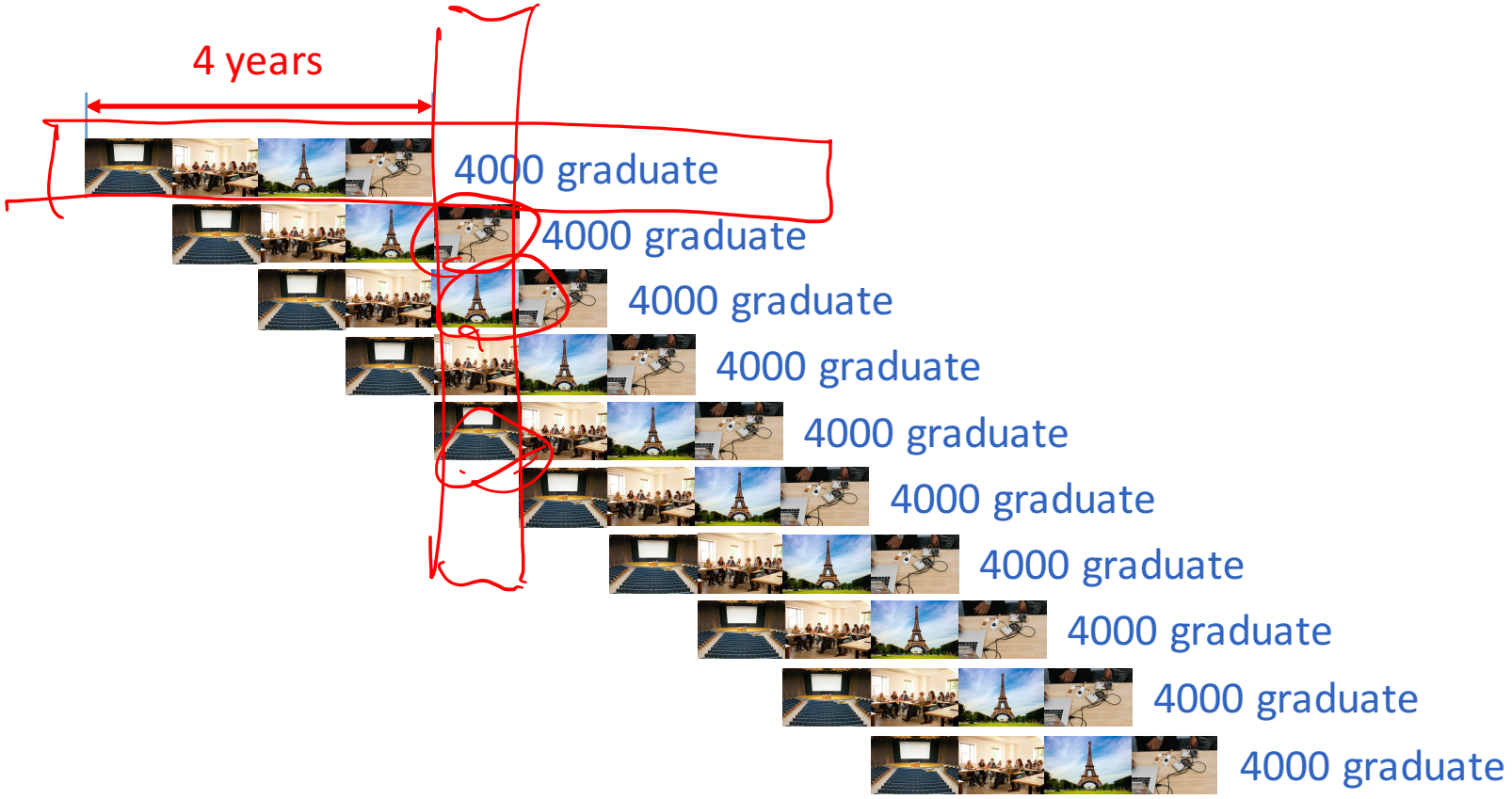
- 16,000 in 7 years
- Steady state throughput is 4000/year
- Resources used efficiently
- **4-fold improvement over serial education**

Latency versus Throughput

- Latency
  - Time from entering college to graduation
  - Serial 4 years
  - Pipelining 4 years
- Throughput
  - Average number of students graduating each year
  - Serial 1000
  - Pipelining 4000
- Pipelining
  - Increases throughput (4x in this example)
  - But does nothing to latency
    - sometimes worse (additional overhead e.g. for shift transition)

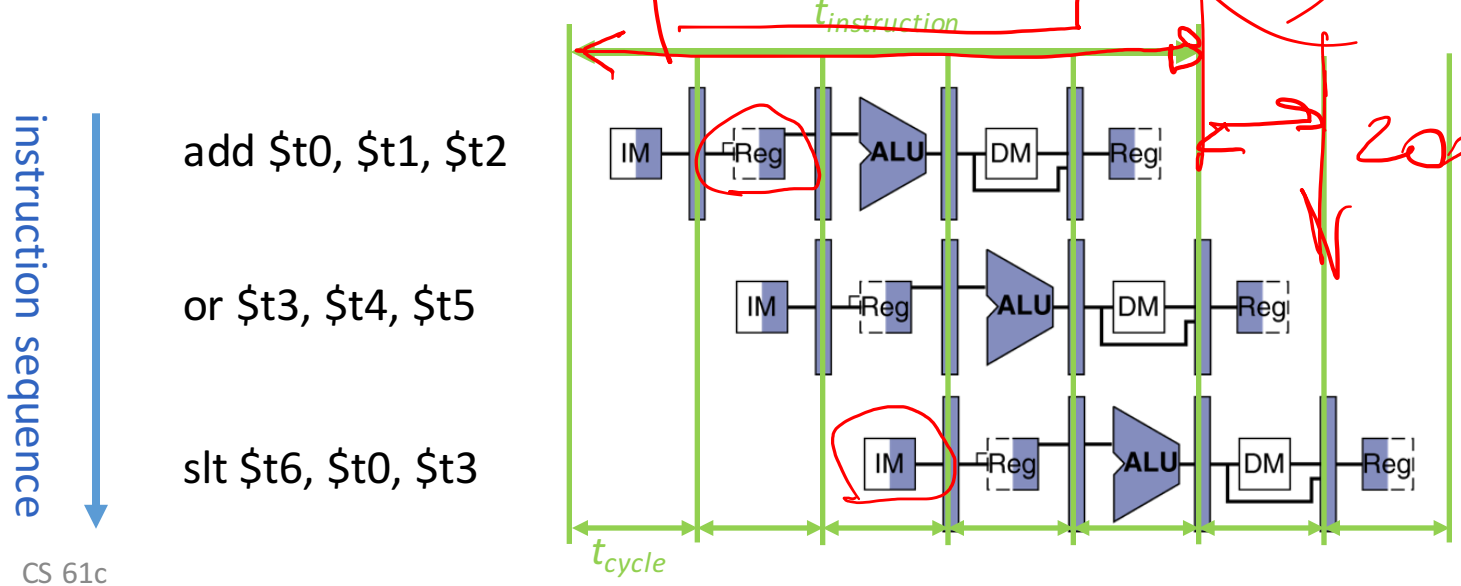
Simultaneous versus Sequential

- What happens *sequentially*?
- What happens *simultaneously*?

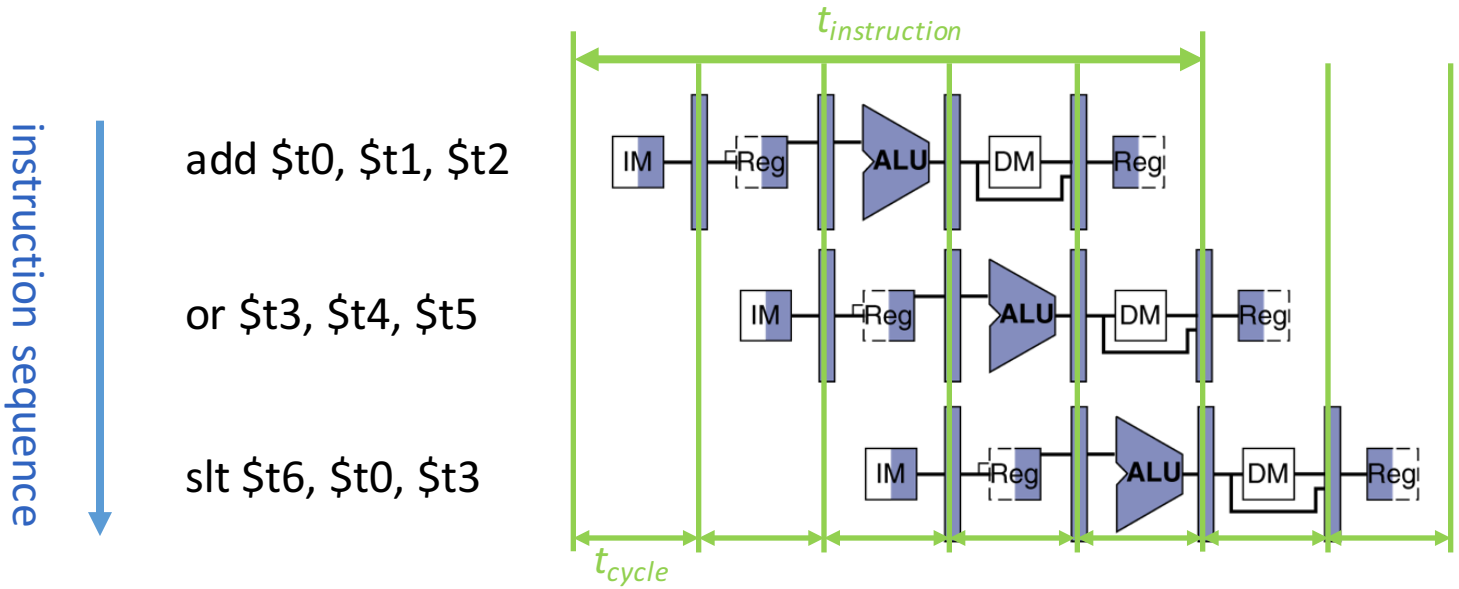


Pipelining with MIPS

Phase	Pictogram	t_{step} Serial	t_{cycle} Pipelined
Instruction Fetch	IM	200 ps	200 ps
Reg Read	Reg	100 ps	200 ps
ALU	ALU	200 ps	200 ps
Memory	DM	200 ps	200 ps
Register Write	Reg	100 ps	200 ps
$t_{instruction}$	IM Reg ALU DM Reg	**800 ps**	**1000 ps**



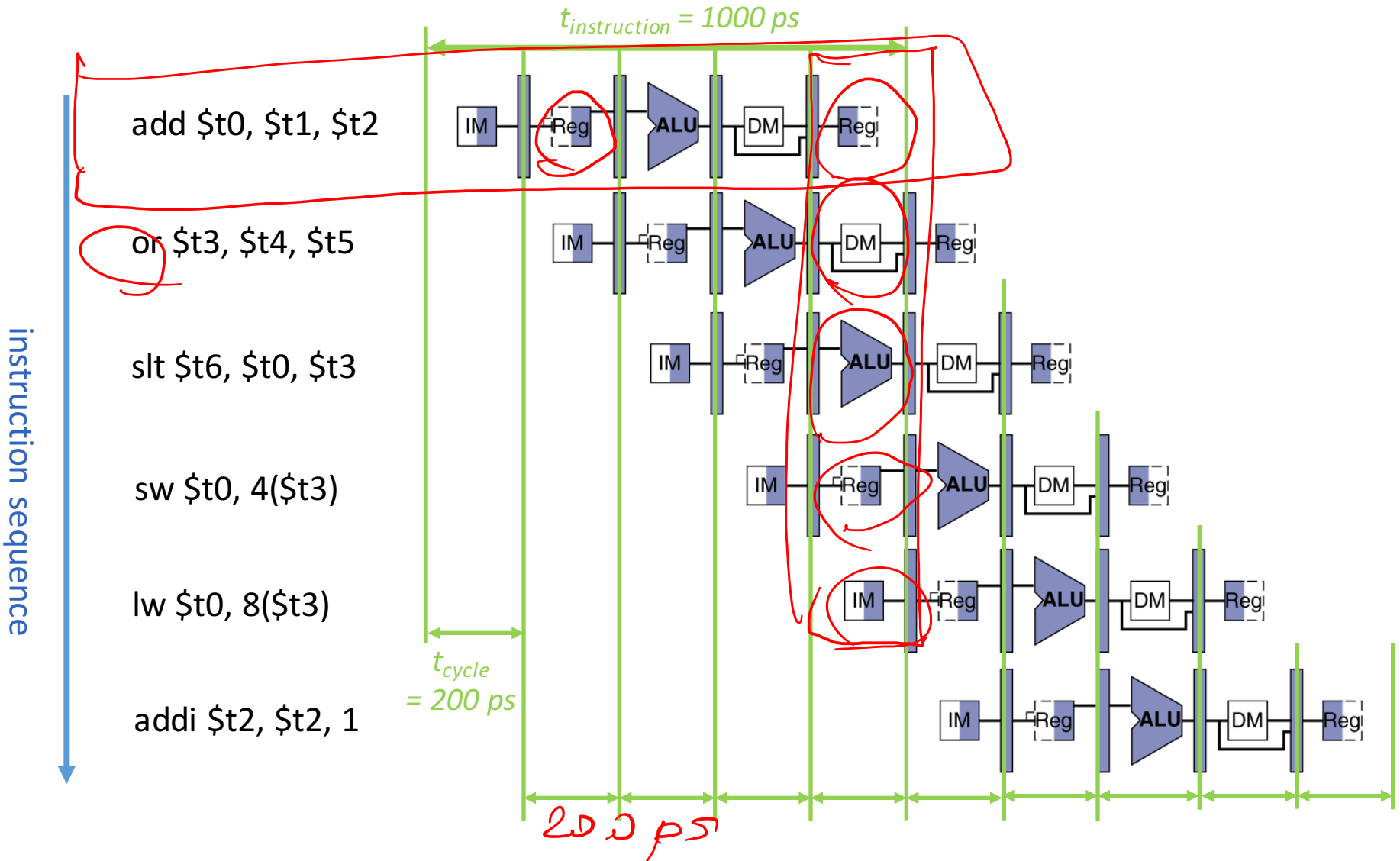
Pipelining with MIPS



	Single Cycle	Pipelining
Timing	t_{step} = 100 … 200 ps	t_{cycle} = 200 ps
	Register access only 100 ps	All cycles same length
Instruction time, $t_{instruction}$	= t_{cycle} = 800 ps	1000 ps
Clock rate, f_s	1/800 ps = 1.25 GHz	1/200 ps = 5 GHz
Relative speed	1 x	4 x

Sequential vs Simultaneous

What happens sequentially, what happens simultaneously?



Your Turn

- Pipeline with following stages:
  1. Instruction Fetch 150 ps
  2. Memory Read 200 ps
  3. ALU 100 … 400 ps (depending on opcode)
  4. Memory Write 250 ps
- What is the maximum pipeline clock rate f_s:

Answer	f_s
A	10 GHz
B	2.5 GHz
C	1 GHz
D	4 GHz
E	1 / 700 ps

Your Turn

- Pipeline with following stages:
  1. Instruction Fetch 150 ps
  2. Memory Read 200 ps
  3. ALU 100 … 400 ps (depending on opcode)
  4. Memory Write 250 ps
- What is the maximum pipeline clock rate f_c:

Answer	f_c
A	10 GHz
B	2.5 GHz
C	1 GHz
D	4 GHz
E	1 / 700 ps

Agenda

- Controller
- Instruction Timing
- Performance Measures
- Introduction to Pipelining
- Pipelined MIPS Datapath
- **And in Conclusion, …**

And in Conclusion, …

- Controller
  - Tells universal datapath how to execute each instruction
- Instruction timing
  - Set by instruction complexity, architecture, technology
  - Pipelining increases clock frequency, "instructions per second"
    - But does not reduce time to complete instruction
- Performance measures
  - Different measures depending on objective
    - Response time
    - Jobs / second
    - Energy per task