CMP 338: *Third Class*

HW 2 solution

Conversion between bases

The TINY processor

Abstraction and separation of concerns

Circuit design big picture

Moore's law and chip fabrication cost

Performance

- What does to measure?
- Processor performance and execution time
- The CPUtime equation

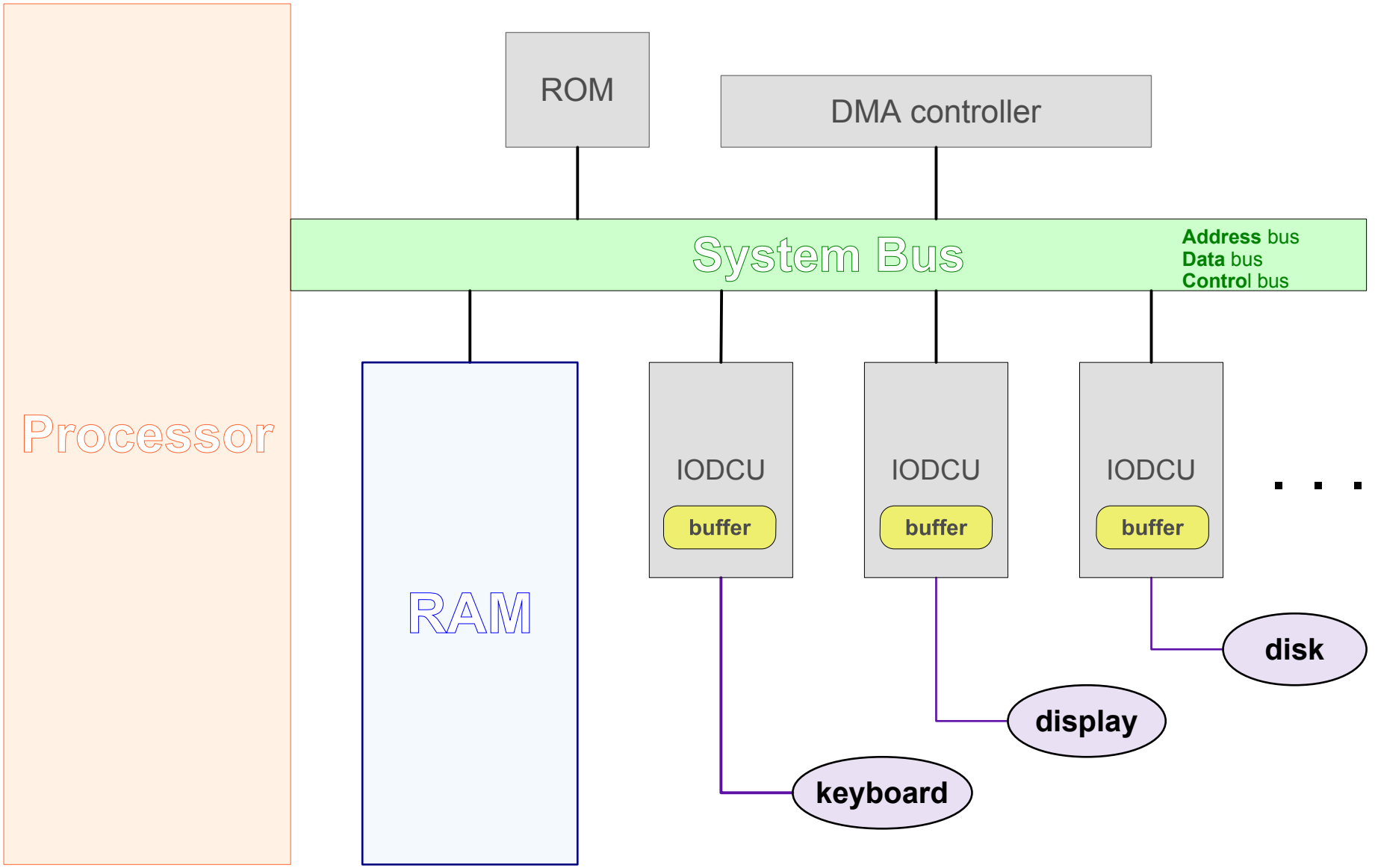
For next class: HW 3; *review* 1.6; *read* 1.10, 2.1-2.3

HW 2: Basic Processor Model

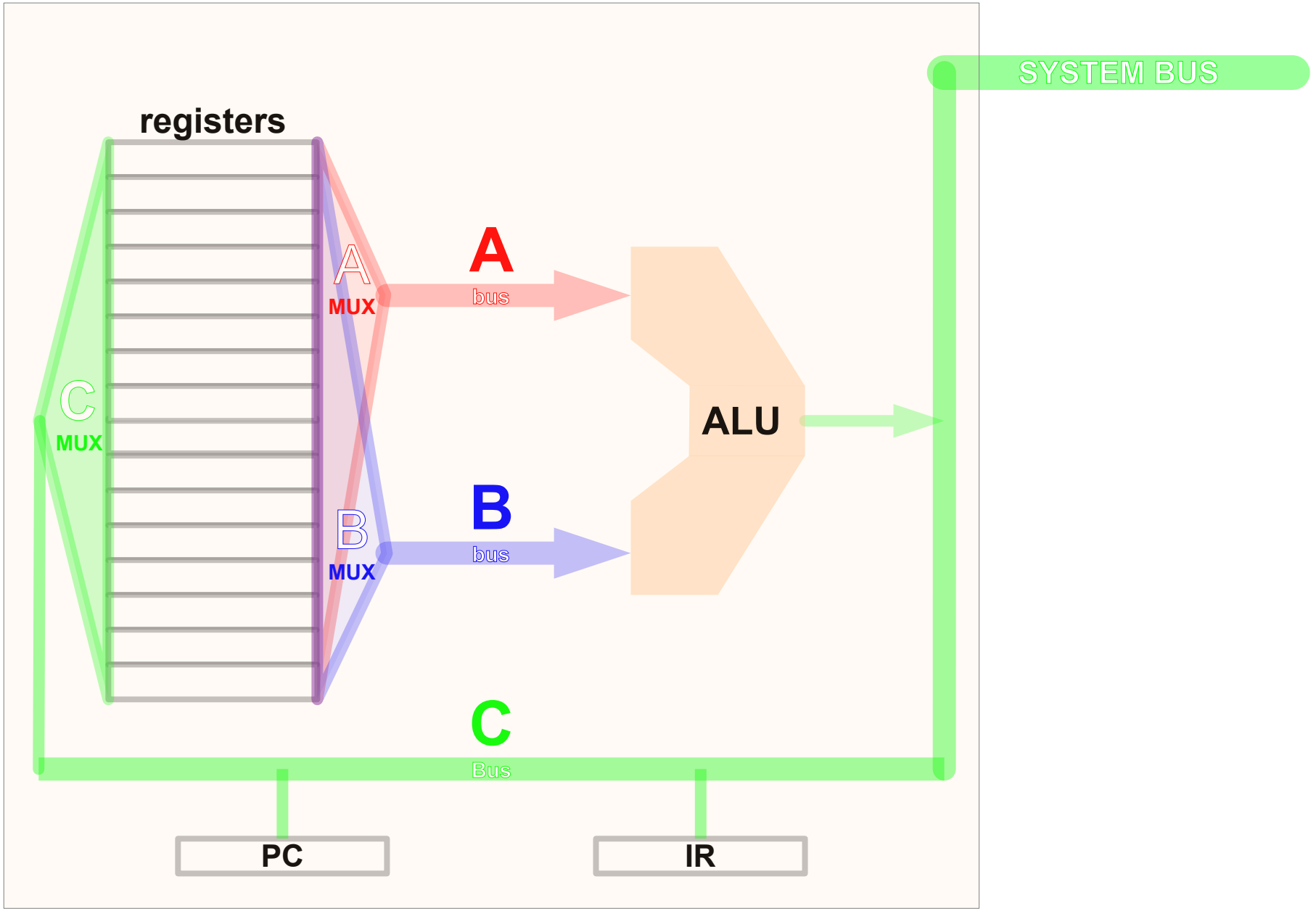
In two or three sentences of your own words define, describe, or discuss the following components of the basic processor model:

1. Register
2. Register File
3. A (B) multiplexer (MUX)
4. A (B) Bus
5. ALU — *Arithmetic / Logical Unit*
6. C Bus
7. Program Counter (PC)
8. Instruction Register (IR)

Basic Computer Model



Basic Processor Model



HW 2 : Processor Components

Register

Memory containing one word (32 bits) or one double word (64 bits) of data.

Register File

A collection of (usually 32) consecutively numbered (0 .. 31) registers.

A (B) multiplexer (MUX)

Selects a specified register to feed to the A (B) bus.

HW 2 : Processor Components

A (B) Bus

Takes the value (or its complement) from the register determined by its MUX and copies it to one of the inputs of the ALU.

ALU — *Arithmetic / Logical Unit*

Performs an arithmetic (**+**, **–**, etc.) operation or a logical (bitwise **and**, **or**, etc.) operation of the values given it on the A and B buses and puts the result on the C bus.

HW 2 : Processor Components

C Bus

Can take a value from the ALU and place it in a specified register. Also, an extension of the System bus onto the processor. Used to *fetch* instructions into the IR, *load* a specified register with the value at a specified address in main memory, and *store* the value in a specified register into a specified address in main memory.

HW 2 : Processor Components

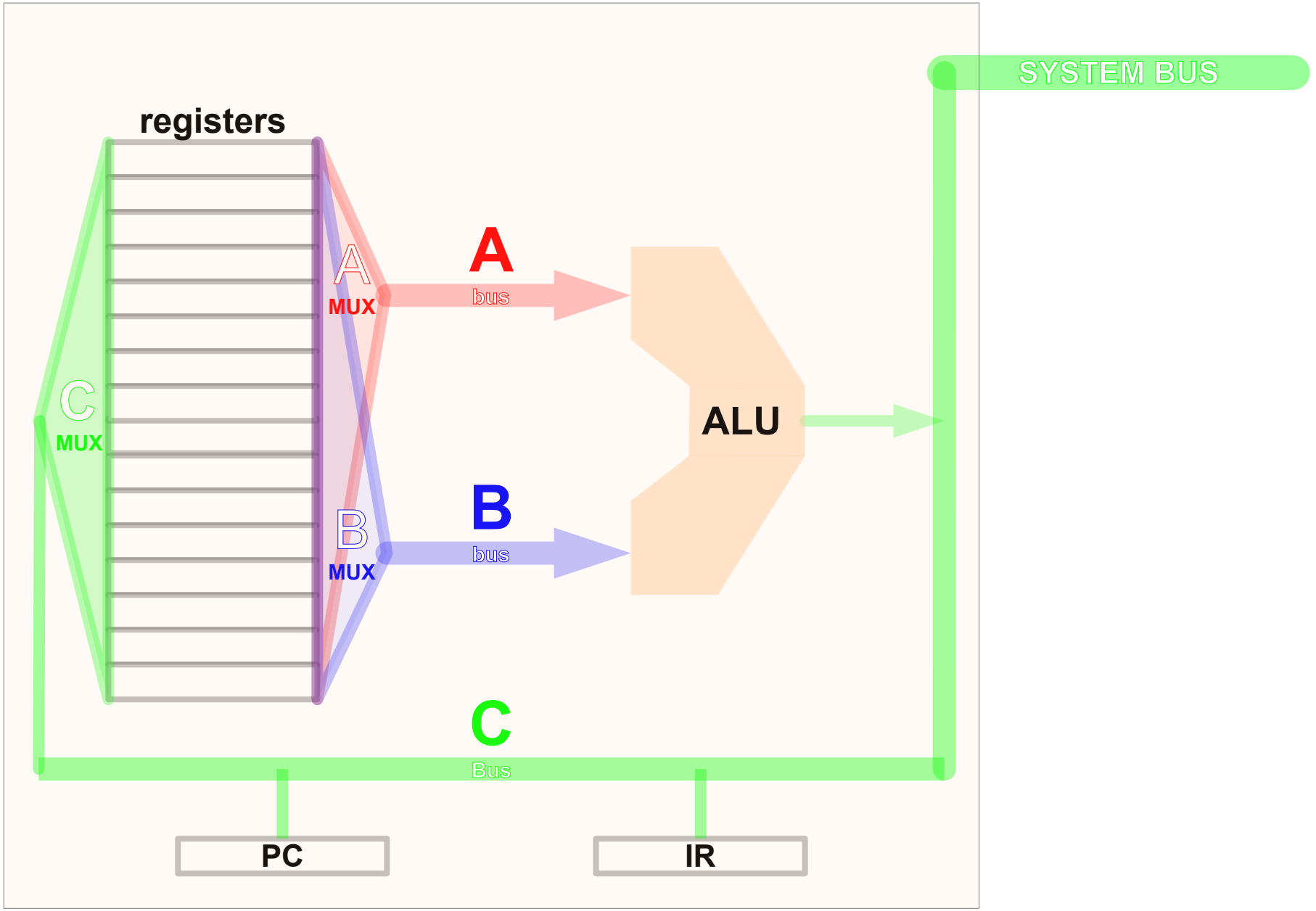
Program Counter (PC)

Instruction Address Register contains the address in main memory of the next instruction to be executed.

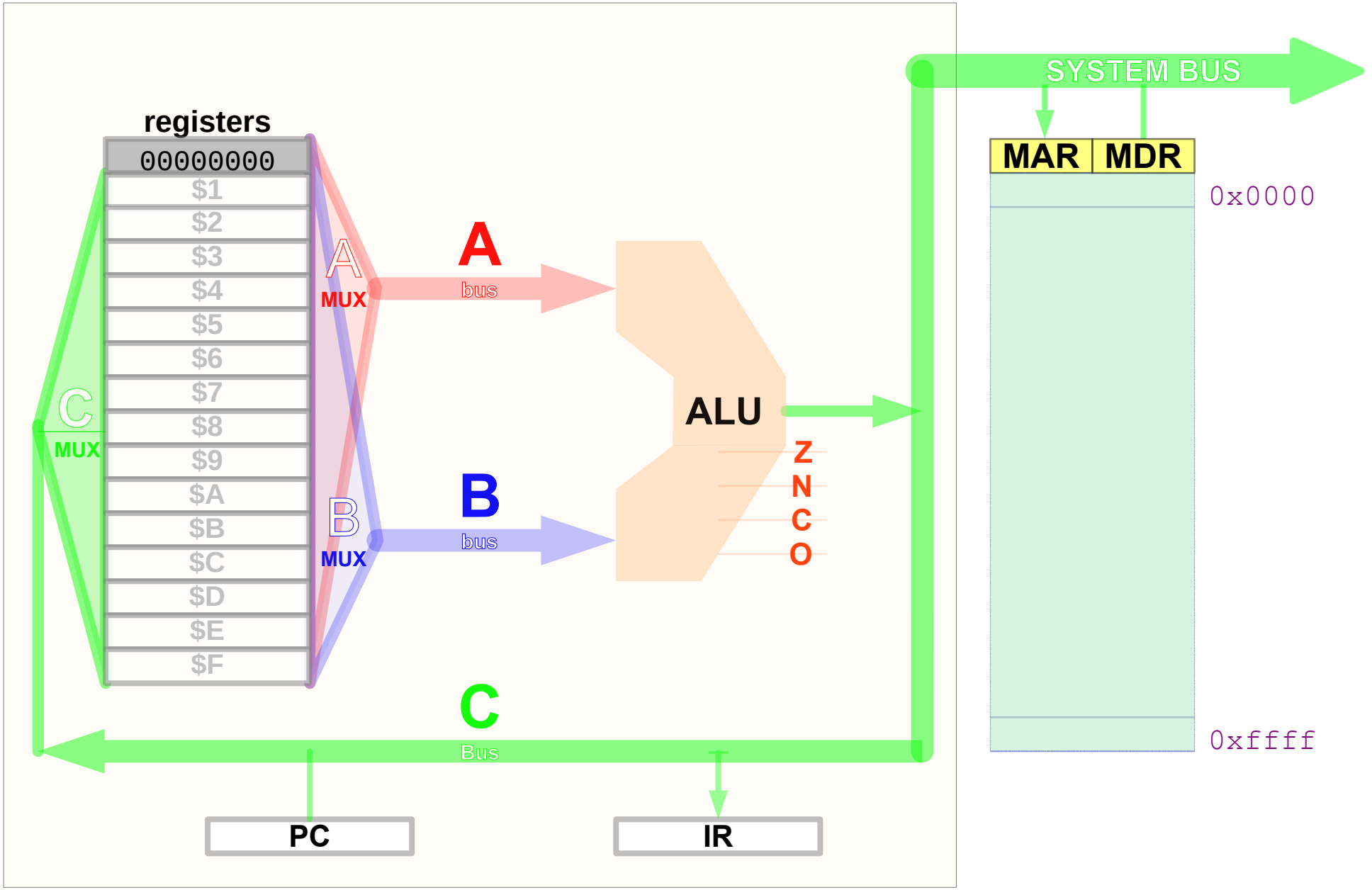
Instruction Register (IR)

Contains the instruction currently being executed. Determines what operations are performed by other components of the processor during the execution.

Basic Processor Model



The TINY Computer



Integers in Different Bases

Base 10 (***decimal*** — ten fingers)

$$4129_{10} = 4*10^3 + 1*10^2 + 2*10^1 + 9*10^0$$

Base 2 (***binary*** — two fingers)

$$1011_{2} = 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$$

Base 16 (***hexadecimal*** — sixteen fingers)

$$A3F8_{16} = 10*16^3 + 3*16^2 + 15*16^1 + 8*16^0$$

Conversion between base 2 and base 16 is easy!

0x A3F8 = 0b 1010 0011 1111 0100

Four Hundred and Thirty Seven

$\mathbf{437_{10}}$	$\mathbf{110110101_2}$	$\mathbf{1B5_{16}}$
$4 \cdot 10^2 = 100$	$1 \cdot 2^8 = 256$	$1 \cdot 16^2 = 256$
$+ 3 \cdot 10^1 = 30$	$+ 1 \cdot 2^7 = 128$	$+ 11 \cdot 16^1 = 128$
$+ 7 \cdot 10^0 = 7$	$+ 0 \cdot 2^6 = 0$	$+ 5 \cdot 16^0 = 5$
	$+ 1 \cdot 2^5 = 32$	
	$+ 1 \cdot 2^4 = 16$	
	$+ 0 \cdot 2^3 = 0$	
	$+ 1 \cdot 2^2 = 4$	
	$+ 0 \cdot 2^1 = 0$	
	$+ 1 \cdot 2^0 = 1$	

Base 2 ↔ Base 10

From base 2 to base 10

Add the power of 2 corresponding to each 1

Example: $01100100_2 = 2^6 + 2^5 + 2^2 = 64 + 32 + 4 = 100_{10}$

From base 10 to base 2

Express number as sum of distinct powers of 2

$209_{10} = 128 + 64 + 16 + 1 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^0$

Add zero times the missing powers of 2

$209_{10} = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

Write coefficients from highest to lowest power of 2

$209_{10} = 11010001_2$

Powers of 2

2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32968

Base 16 ↔ Base 2

From base 16 to base 2

Replace each hex digit with its 4-bit binary equivalent

$6E30AC58_{16} = 01101110001100001010110001011000_2$

From base 2 to base 16

Pad left with 0 until length is multiple of 4

$1100100100111011_2 = 0001100100100111011_2$

Replace consecutive sequences of 4 bits with hex digit

$0001100100100111011_2 = 1927B_{16}$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

First 16 Non Negative Integers

Decimal		Binary		Hexadecimal	
0	8	0000	1000	0	8
1	9	0001	1001	1	9
2	10	0010	1010	2	A
3	11	0011	1011	3	B
4	12	0100	1100	4	C
5	13	0101	1101	5	D
6	14	0110	1110	6	E
7	15	0111	1111	7	F

8 Great Computer Architecture Ideas

Design for ***Moore's Law***

Use ***abstraction*** to simplify design

Make the ***common case fast***

Performance *via* ***parallelism***

Performance *via* ***pipelining***

Performance *via* ***prediction***

Hierarchy of memories

Dependability *via* redundancy

Separation of Concerns

Interface

Boundary – between objects or systems
Protocol – rules for interaction between parties
Contract – formalized expectations

Distribution of Labor

User (consumer) ignores *implementation*
Provider (producer) ignores *application*

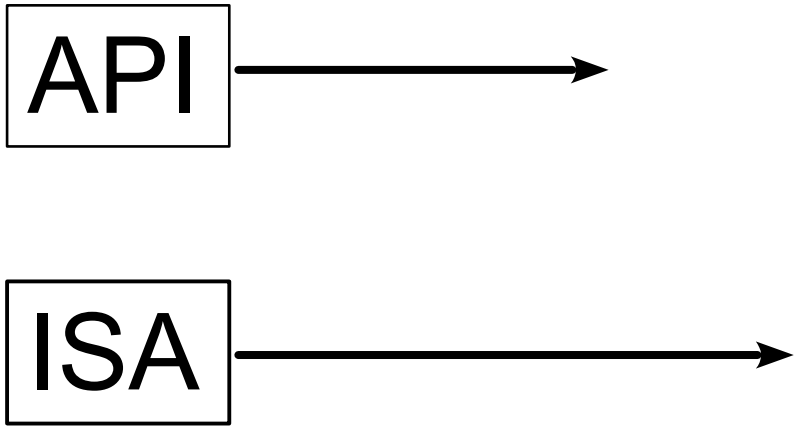
Instruction Set Architecture **ISA**

Between hardware and software

Application Program Interface **API**

Between application program and operating system

Interface Map



What Happens to Your Program

Computer Design – The Big Picture

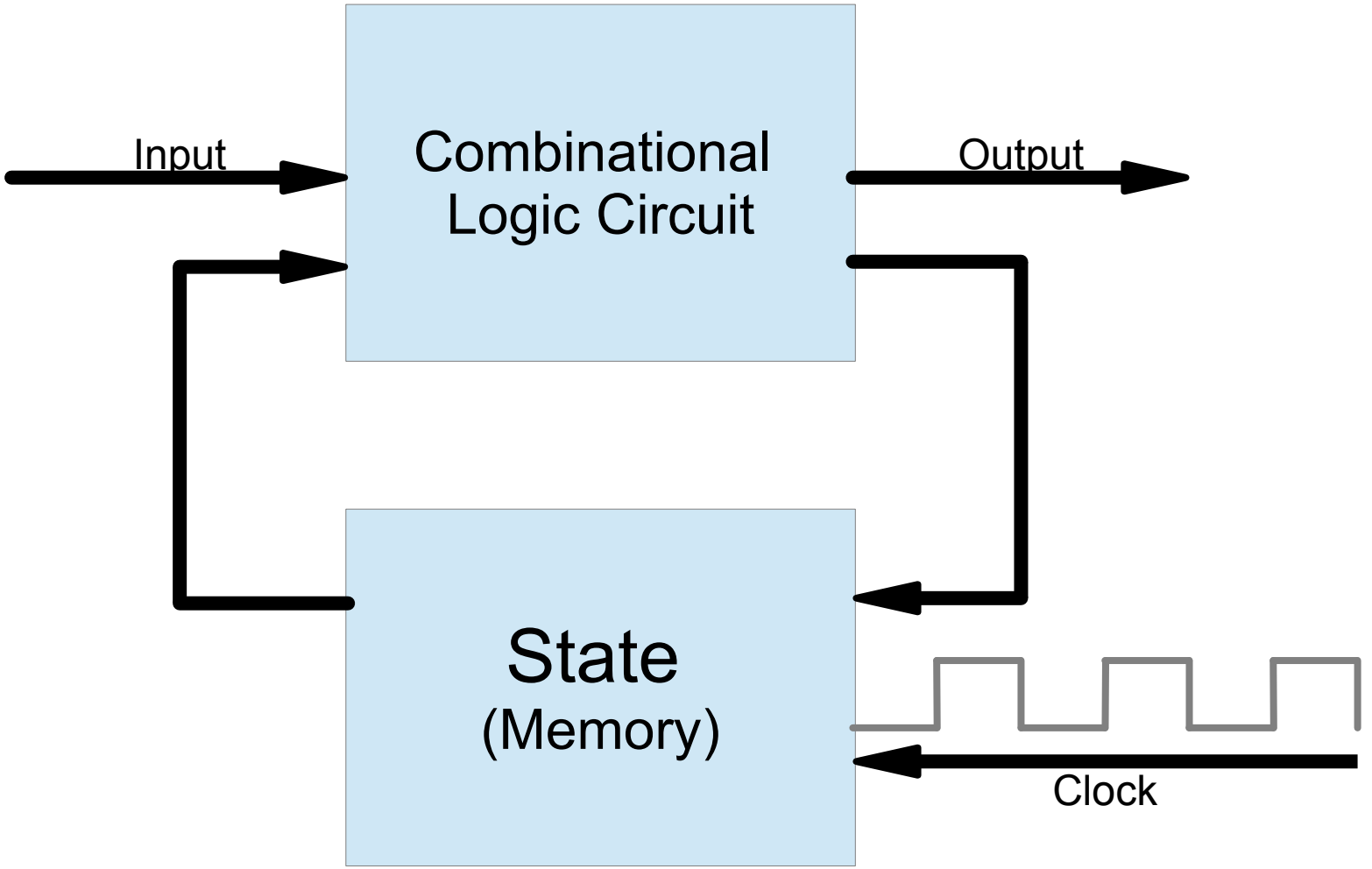
A computer is one big ***sequential*** circuit

- **Abstract** into discrete sequential components
  - ***Combinational*** circuits + memory + clock

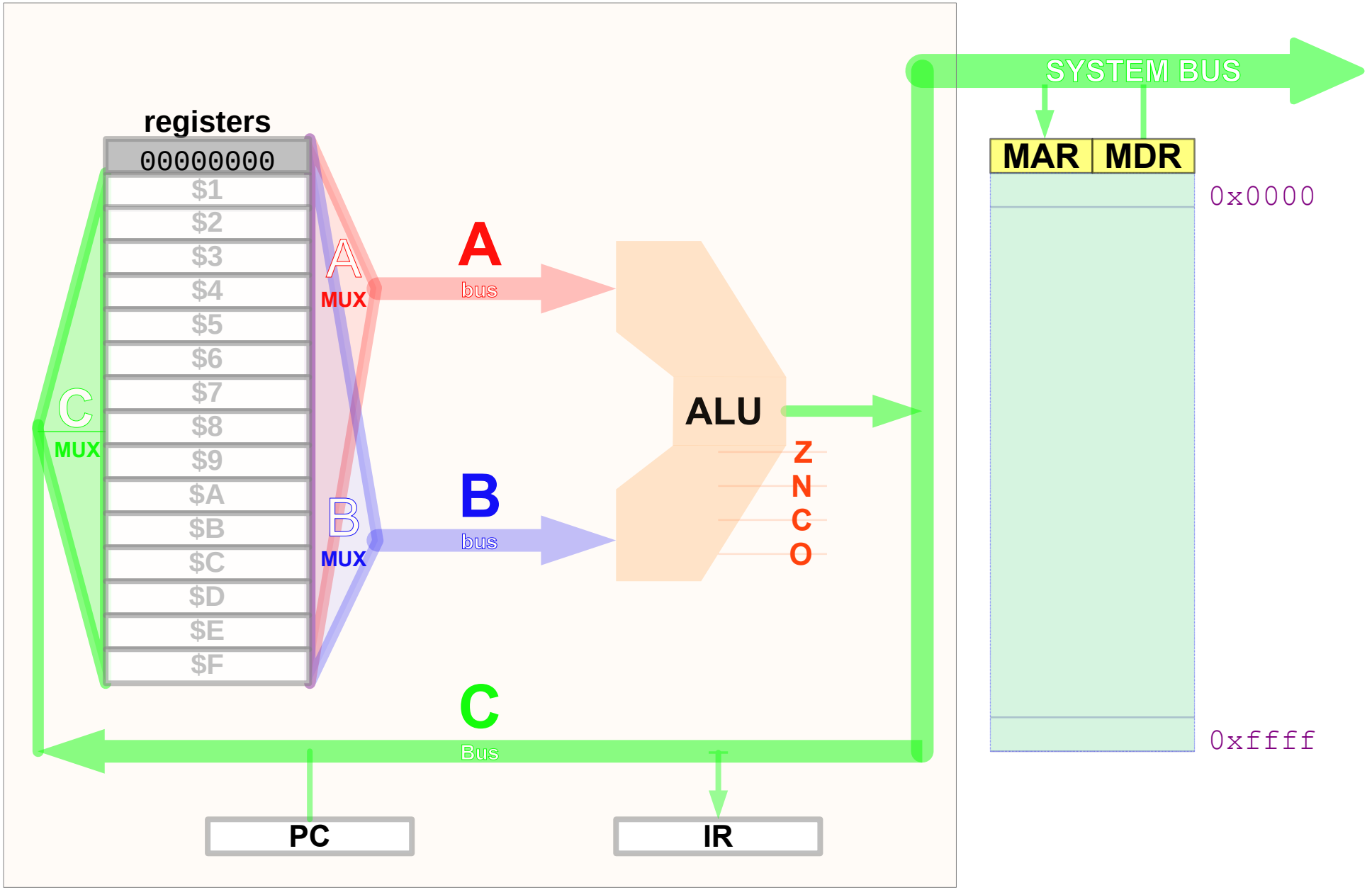
Combinational circuit design

1. Specify semantics
   - Black Box input and output
   - *Truth Table* (Input determines output)
2. Truth table → *Boolean formula*
3. Minimize boolean formula (Karnaugh Maps)
4. Boolean formula → combinational circuit

Synchronous Sequential Circuit



The TINY Computer



Black Box Logic Design

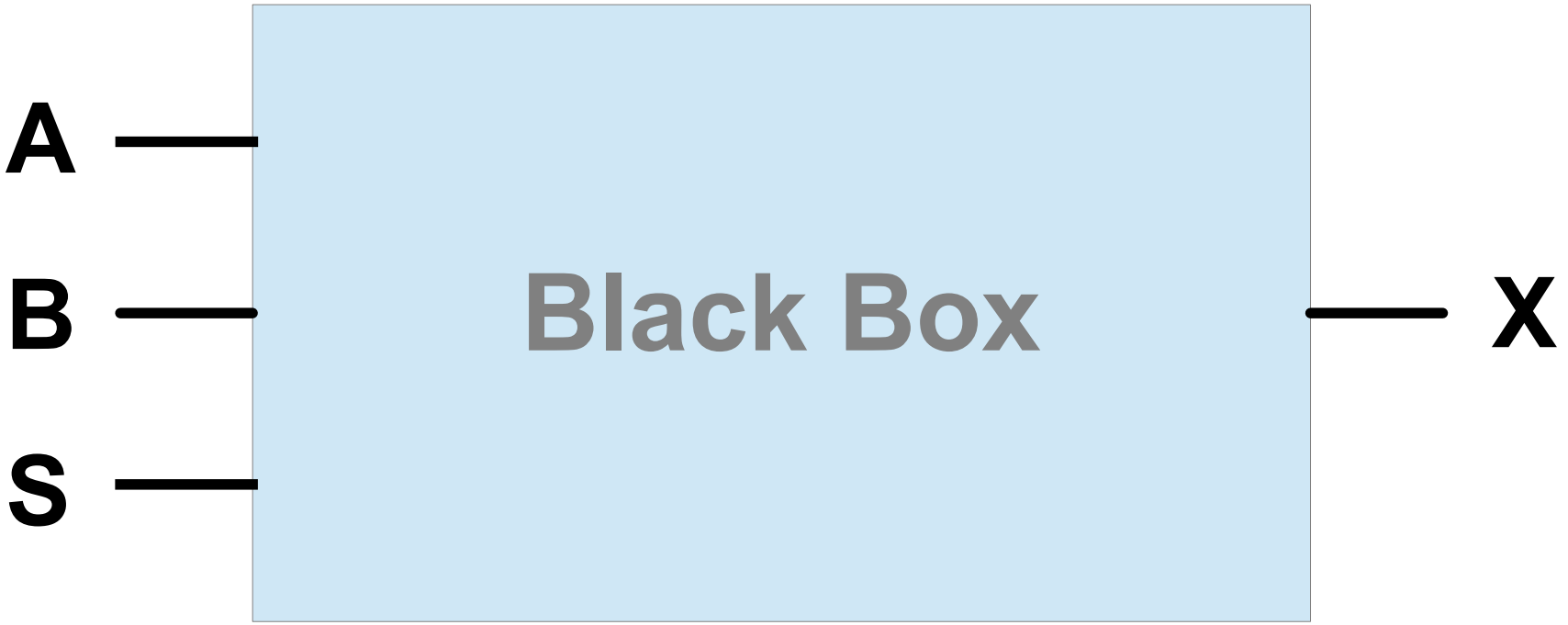
Combinational circuit

- Output determined by input

Design process

1. Specify semantics
   - Black Box input and output
   - Truth Table (input determines output)
2. From truth table to boolean formula
3. **Minimize boolean formula** (optional)
   - Boolean algebra
   - Karnaugh maps
4. From boolean formula to circuit

Two Way Multiplexer Design



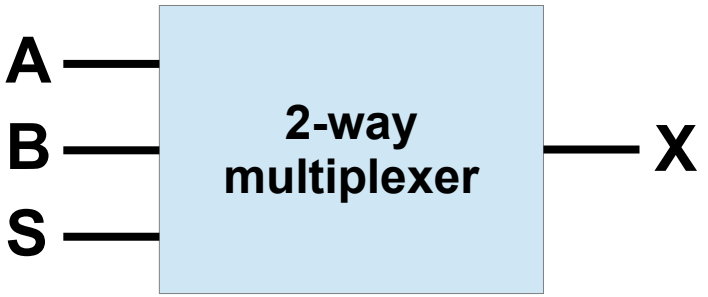
Informal semantics:

X = **A** — *if* **S** = 0

X = **B** — *if* **S** = 1

Two Way Multiplexer Truth Table

S	A	B	X
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

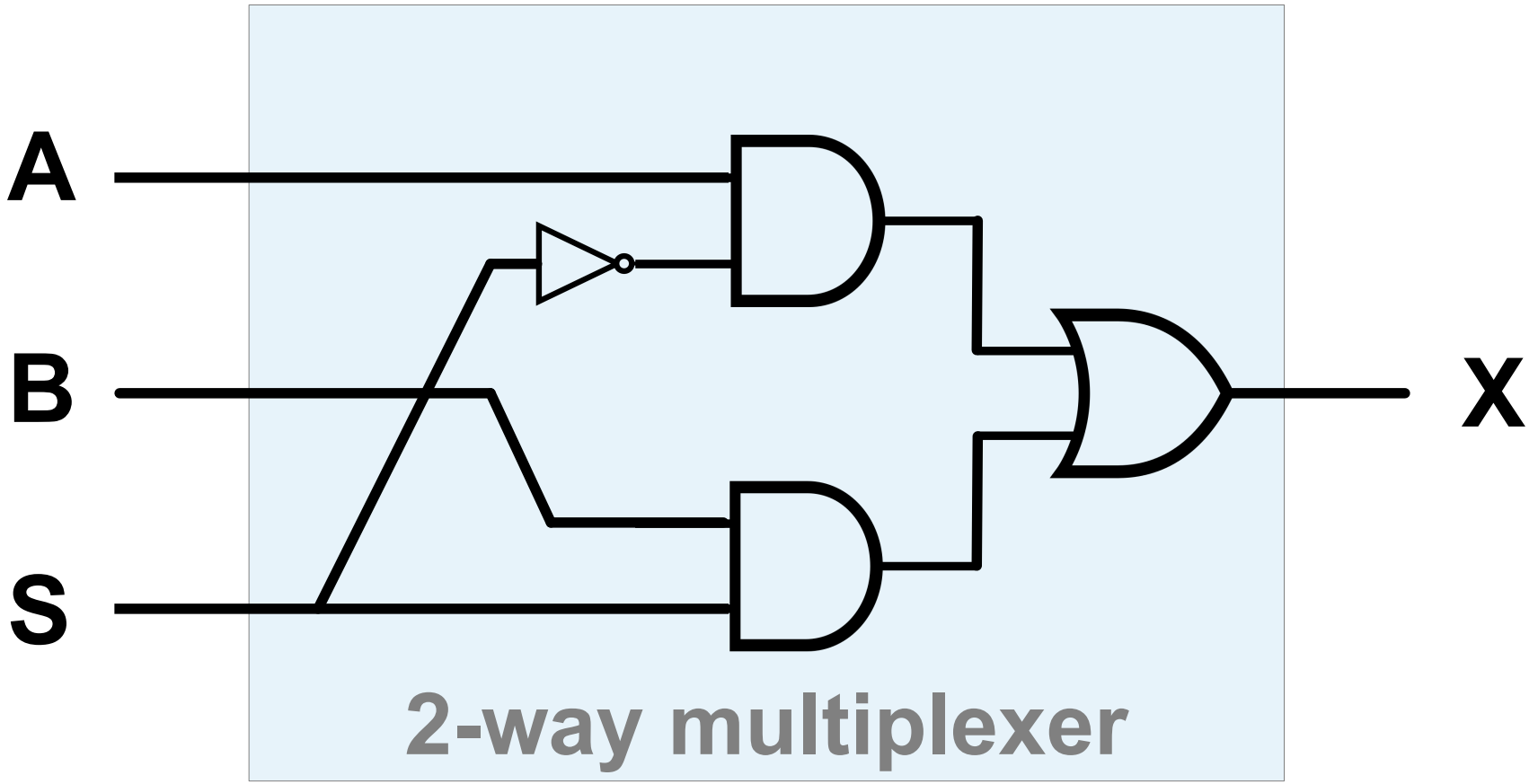


not S and A and not B, or
not S and A and B, or
S and not A and B, or
S and A and B

$X = \overline{S}A\overline{B} + \overline{S}AB + S\overline{A}B + SAB$

$= \overline{S}A + SB$

Two Way Multiplexer Circuit



$$X = \overline{S}A + SB$$

8 Great Computer Architecture Ideas

Design for ***Moore's Law***

Use ***abstraction*** to simplify design

Make the ***common case fast***

Performance *via* ***parallelism***

Performance *via* ***pipelining***

Performance *via* ***prediction***

Hierarchy of memories

Dependability *via* redundancy

Moore's "Law"

Moore's Observation (1965)

- # gates per chip doubles (about) every two years
- Compute power $\propto$ # gates per chip

What to do with increasing compute power?

- Until about 2000, faster (and bigger) uniprocessors
- Since 2003, more (simpler) processors per chip
  - Exploiting increasing parallelism isn't easy

Are we the end of Moore's Law?

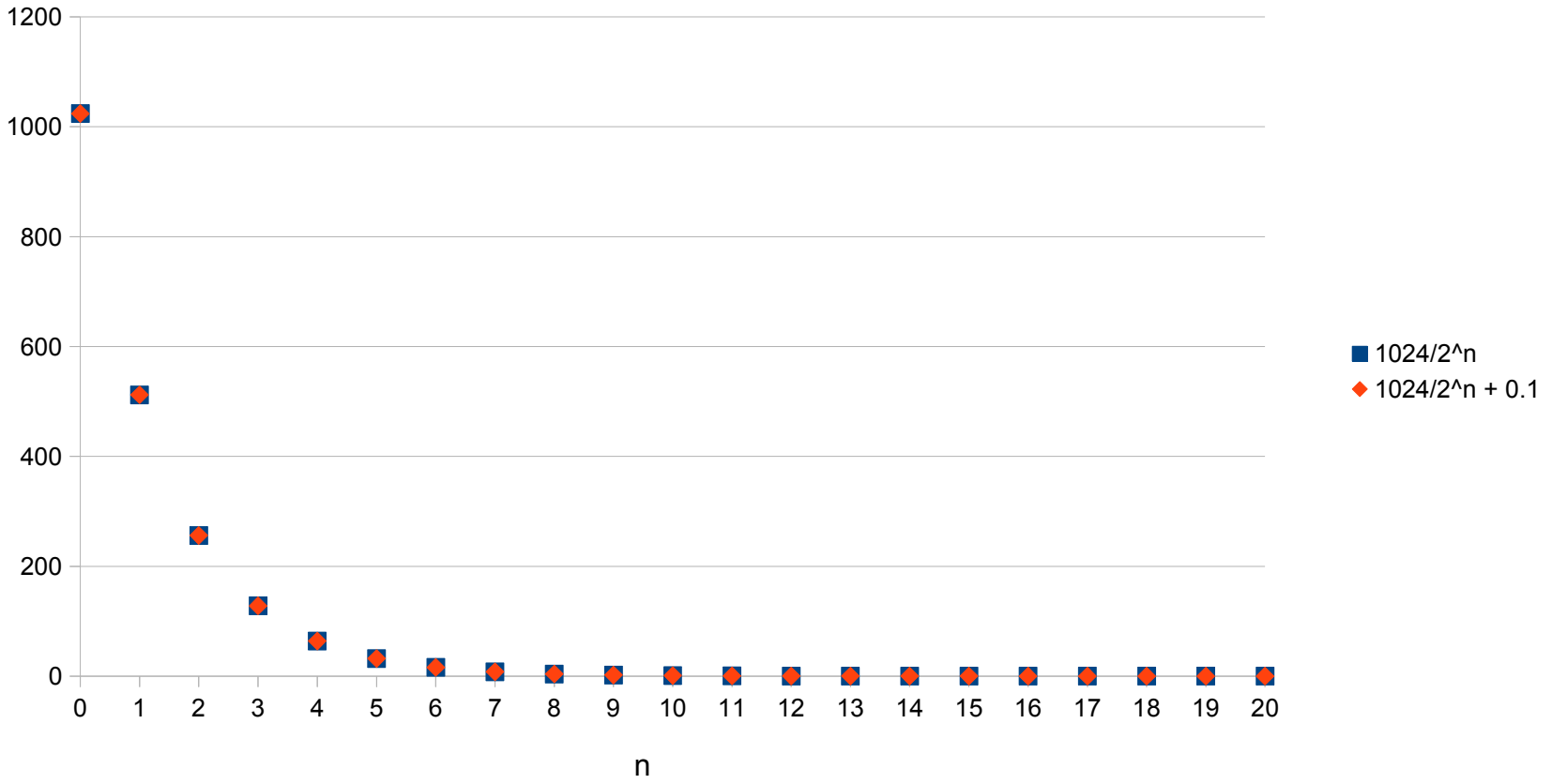
- Seems to be slowing down, can't continue forever
- However, it has been pronounced dead before

Moore’s Law

Processor Performance over Time

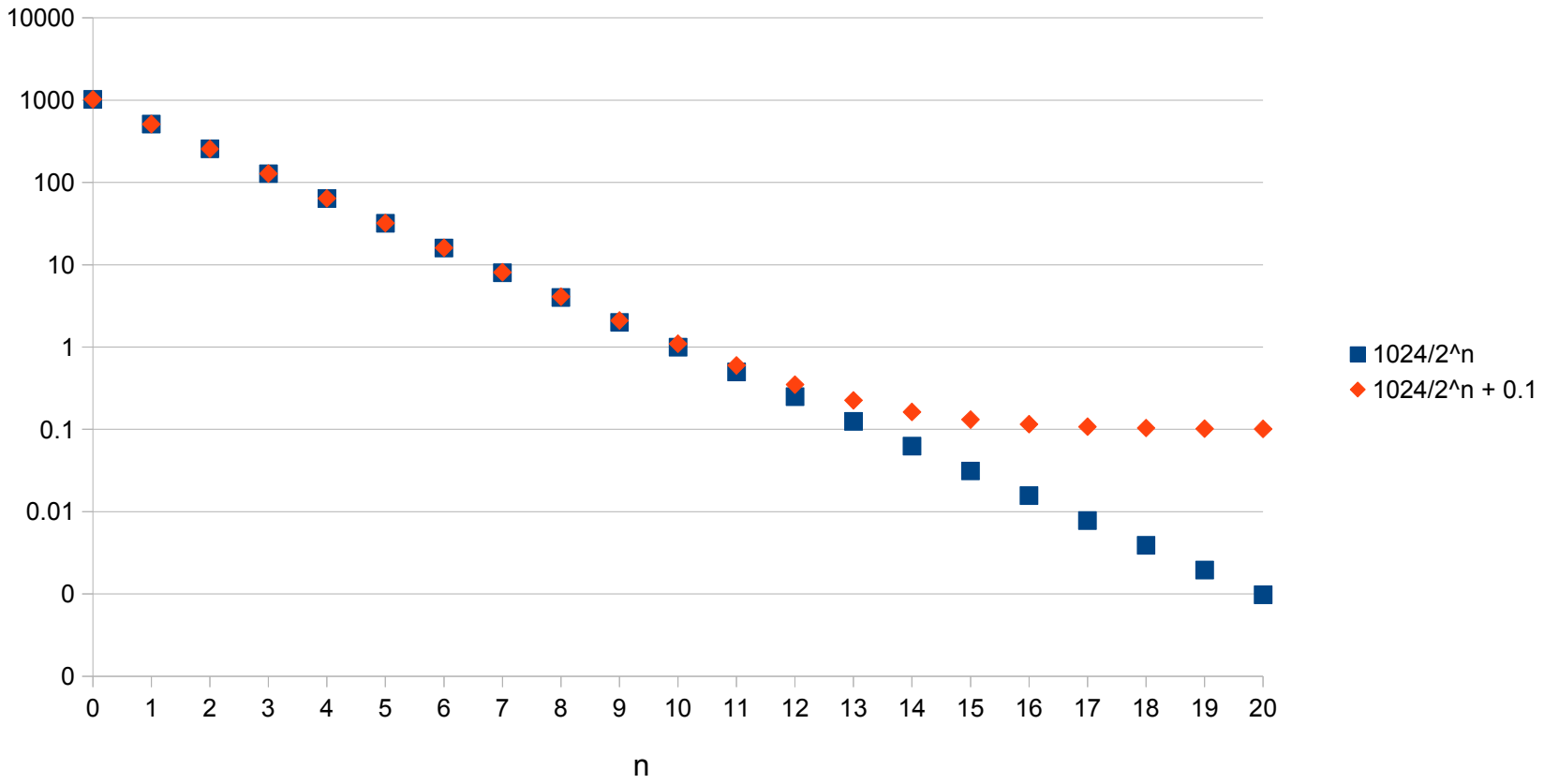
Alpern's Law

Exponential growth is ultimately ***unsustainable***

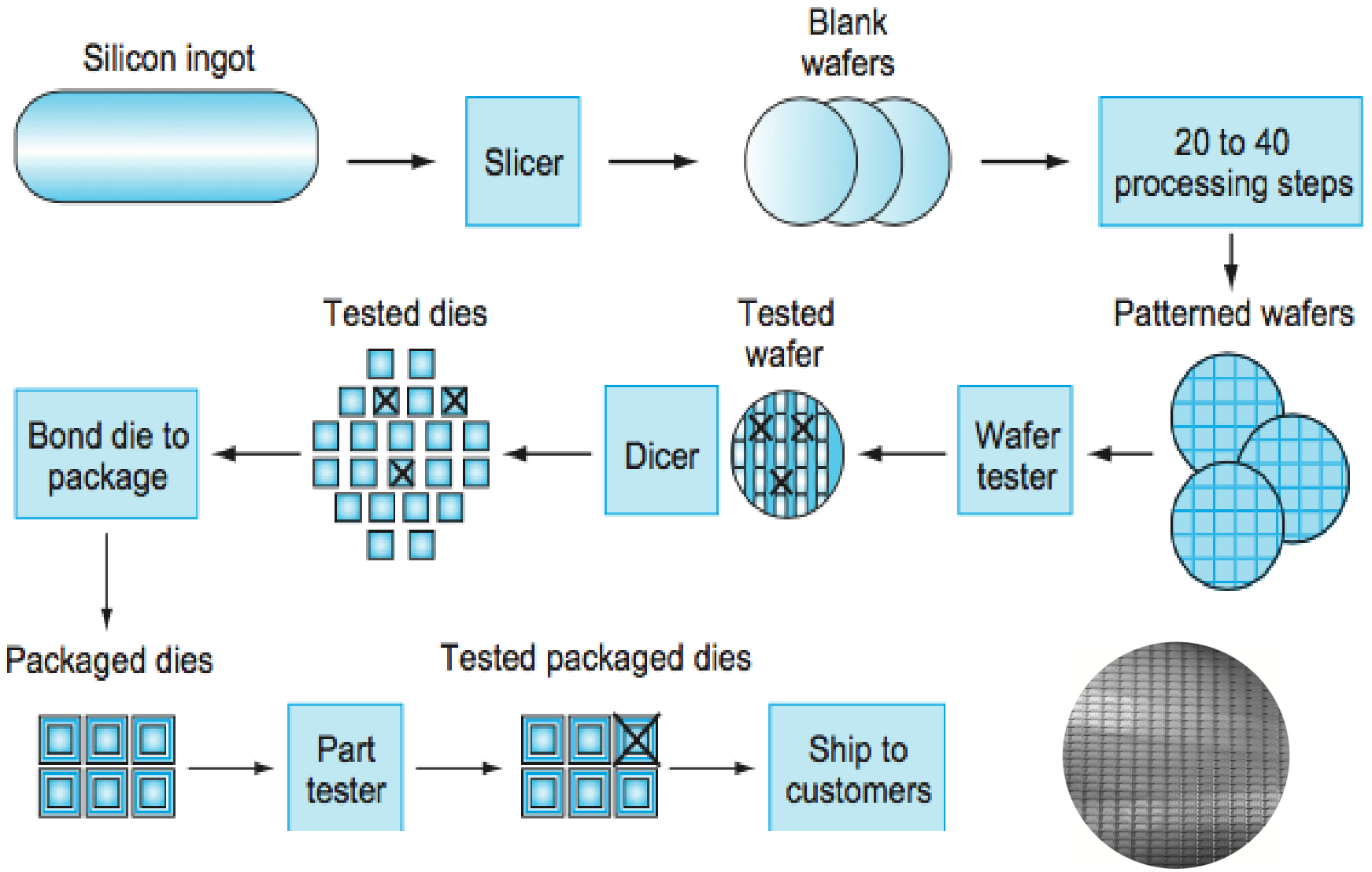


Alpern's Law

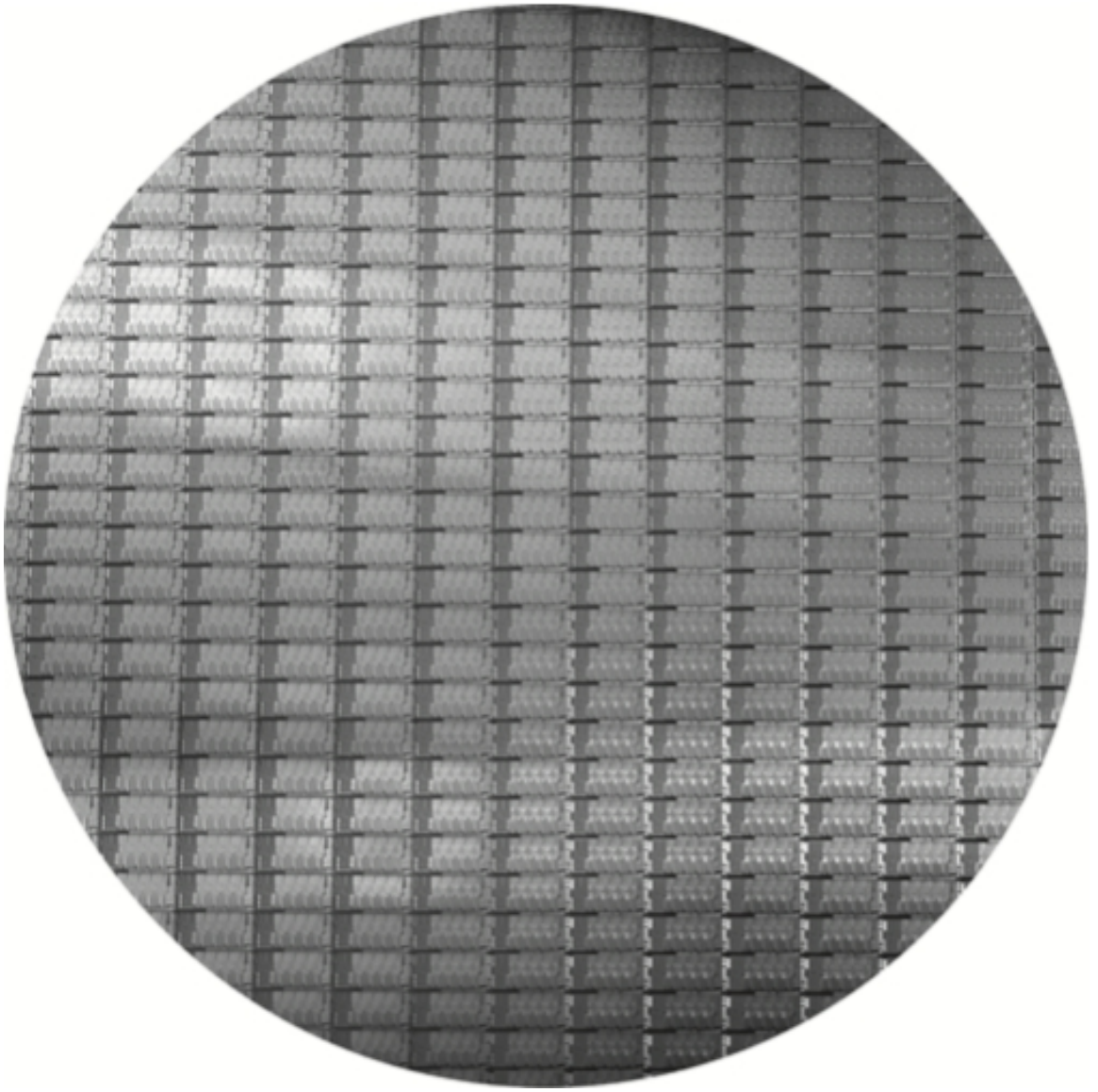
Exponential growth is ultimately ***unsustainable***



Chip Fabrication



Intel Core I7 Wafer



Integrated Circuit Fabriction Costs

$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{yield}}$$

$$\text{Dies per wafer} \approx \frac{\text{Wafer area}}{\text{Die area}}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

11.8 inch (300mm) patterned wafer

~325 (20.7 x 10.5 mm) dies per wafer

~23% of dies are defective (yield = ~0.77)

If each wafer costs $20,000

what is the fabrication cost of a chip (die)?

Understanding Performance

From *qualitative* to *quantitative* analysis

Statistical tools

- **Average** and **weighted average**

Performance equations

- ***Relative performance***
- ***CPU time equation***
- ***Amdahl's law***

Performance metrics (what to measure)

- What does “performance” mean?

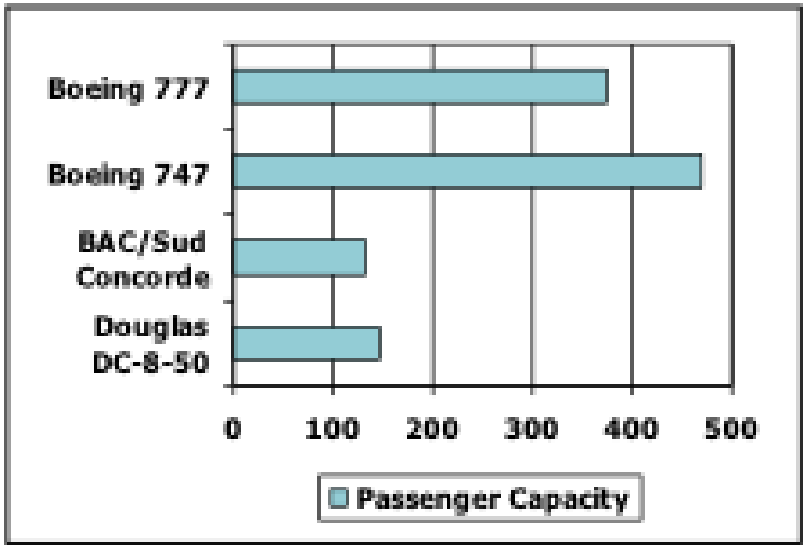
Performance Metrics

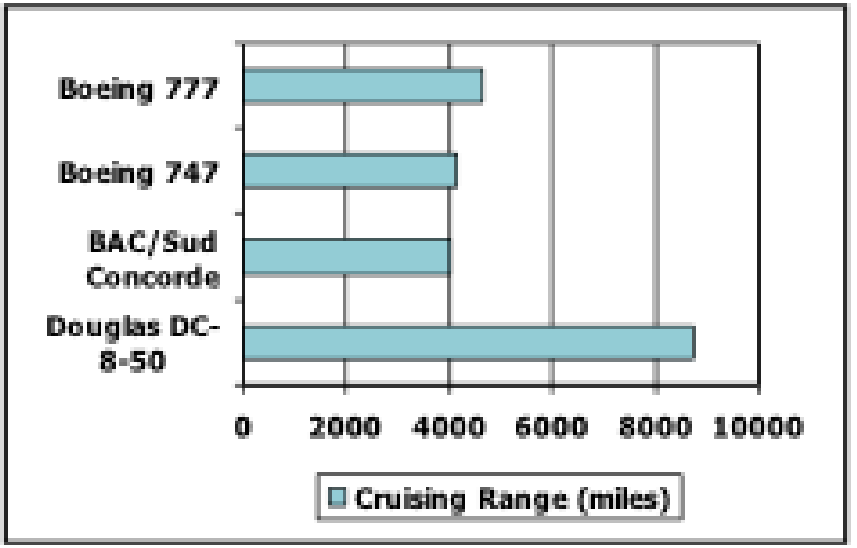
Different measures of airplane “performance”?

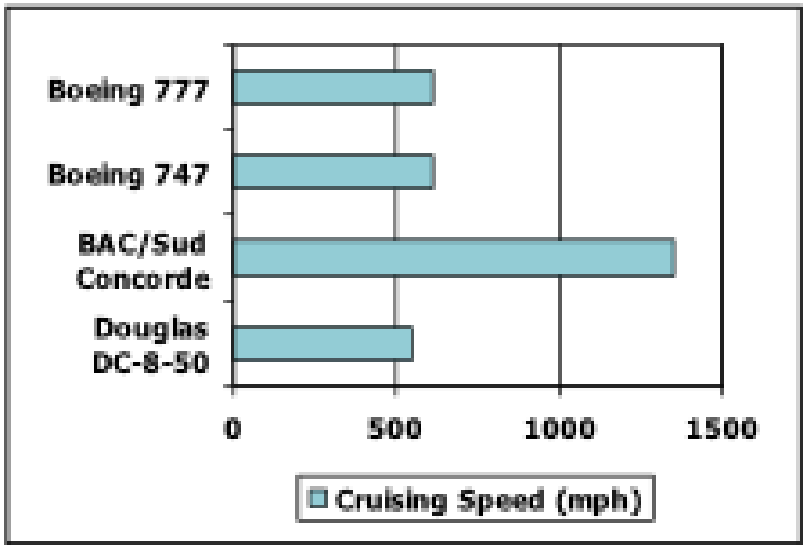
- ***Speed*** (mph) ?
- ***Range*** (miles) ?
- ***Capacity*** (passengers) ?
- ***Throughput*** (passengers miles per hour) ?

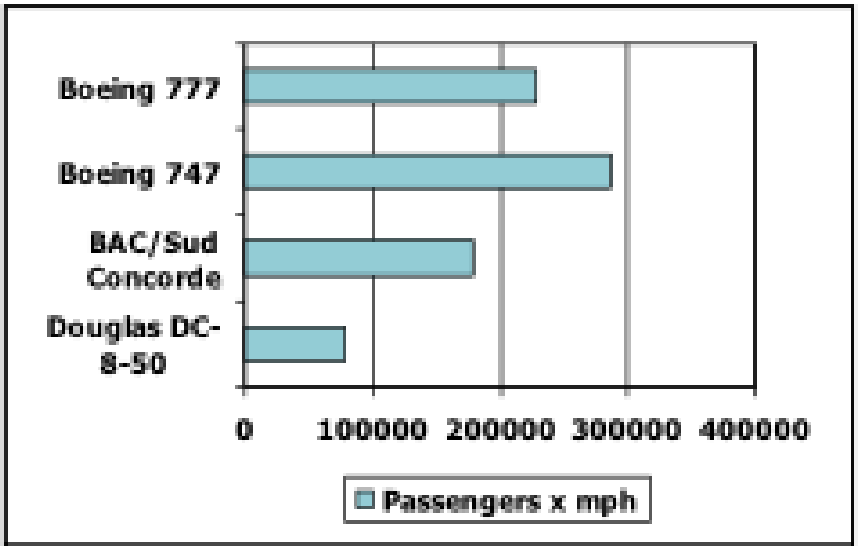
Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 777	375	4630	610	228,750
Boeing 747	470	4150	610	286,700
BAC/Sud Concorde	132	4000	1350	178,200
Douglas DC-8-50	146	8720	544	79,424

Airplane Performance Metrics









Computer Performance Metrics

Response (execution) time (seconds)

$\mathbf{CPU}_{time} + I/O_{time}$

Throughput (tasks per hour)

Availability (percent) $\frac{MTTF}{MTTF + MTTR}$

MTTF — Mean Time To Failure (years)

MTTR — Mean Time To Repair (minutes)

Execution energy (joules)

Throughput cost (tasks per hour per dollar)

. . .

Response Time Performance

Definition

$$\text{Performance}_X \equiv \frac{1}{\text{ExecutionTime}_X} \qquad P_X \equiv \frac{1}{E_X}$$

Better performance mean *shorter* execution time

Relative performance

X is ***n*** times *as fast as* Y if and only if

$$\boldsymbol{n} = \frac{P_X}{P_Y} = \frac{E_Y}{E_X}$$

Y takes ***n*** times as long as X to execute

Relative Performance

If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times as fast as B if

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Thus the performance ratio is

$$\frac{15}{10} = 1.5$$

and A is therefore 1.5 times as fast as B.

In the above example, we could also say that computer B is 1.5 times *slower than* computer A, since

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$

means that

$$\frac{\text{Performance}_A}{1.5} = \text{Performance}_B$$

Processor Performance

Program execution time = CPU_{time} + $I/0_{time}$

CPU_{time} will be our key metric of processor performance

We will return to I/O_{time} at the end of this course

CPU_{time} = # instructions • (average) $instruction_{time}$

$instruction_{time}$ = (average) cycles per instruction • $cycle_{time}$

$$cycle_{time} = \frac{\#\ \text{seconds}}{\text{cycle}} = \frac{1}{\text{clock}_{\text{rate}}}$$

$clock_{rate}$ measured in Hertz (cycles per second)

$$\text{CPU}_{\text{time}}(\text{execution}) = \frac{\#\ \text{instructions}}{\text{execution}} \cdot \frac{\#\ \text{cycles}}{\text{instruction}} \cdot \frac{\#\ \text{seconds}}{\text{cycle}}$$

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

Figure 1.15 shows the basic measurements at different levels in the computer and what is being measured in each case. We can see how these factors are combined to yield execution time measured in seconds per program:

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

Always bear in mind that the only complete and reliable measure of computer performance is time. For example, changing the instruction set to lower the instruction count may lead to an organization with a slower clock cycle time or higher CPI that offsets the improvement in instruction count. Similarly, because CPI depends on type of instructions executed, the code that executes the fewest number of instructions may not be the fastest.

Performance Equations

Definition of performance

performance: $P_x \equiv \frac{1}{T_x}$ ***relative performance:*** $\frac{P_x}{P_y} = \frac{T_y}{T_x}$

CPU time equation

$$T_{CPU}(\text{execution}) = \frac{\#\text{ instructions}}{\text{execution}} \cdot \frac{\#\text{ cycles}}{\text{instruction}} \cdot \frac{\#\text{ seconds}}{\text{cycle}}$$

Amdahl's law

$$T_{new} = \frac{\text{fraction effected} \cdot T_{old}}{\text{improvement}} + \text{fraction not effected} \cdot T_{old}$$

CPU_{time} Relative Performance

$$T_{CPU}(\text{execution}) = \frac{\#\,\text{instructions}}{\text{execution}} \cdot \frac{\#\,\text{cycles}}{\text{instruction}} \cdot \frac{\#\,\text{seconds}}{\text{cycle}}$$

$$T_X = \#\,\text{instructions}_X \cdot CPI_X \cdot \text{cycleTime}_X$$

$$= \frac{\#\,\text{instructions}_X \cdot CPI_X}{\text{clockRate}_X}$$

$$\textbf{cycleTime} = \frac{\textbf{1}}{\textbf{clockRate}}$$

$$\frac{P_X}{P_Y} = \frac{T_Y}{T_X} = \frac{\#\,\text{instructions}_Y \cdot CPI_Y \cdot \text{cycleTime}_Y}{\#\,\text{instructions}_X \cdot CPI_X \cdot \text{cycleTime}_X}$$

$$= \frac{\#\,\text{instructions}_Y \cdot CPI_Y \cdot \text{clockRate}_X}{\#\,\text{instructions}_X \cdot CPI_X \cdot \text{clockRate}_Y}$$