Computer Architecture - Performance Metrics

What is the Performance?

Plane	DC to Paris	Speed	Passengers	passengers X mph
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

Which of the planes has better performance

What is the Performance?

- The plane with the highest speed is **Concorde**
- The plane with the largest capacity is **Boeing 747**

- Time of Concorde vs. Boeing 747?
  - Concord is 1350 mph / 610 mph = 2.2 times faster

- Throughput of Concorde vs. Boeing 747 ?
  - Boeing is 286,700 pmph / 178,200 pmph = 1.6 times faster

- Boeing is 1.6 times faster in terms of throughput
- Concord is 2.2 times faster in terms of flying time

- When discussing processor performance, we will focus primarily on execution time for a single job - why?

Definitions of Time

- Time can be defined in different ways, depending on what we are measuring:
  - Response time : The time between the start and completion of a task. It includes time spent executing on the CPU, accessing disk and memory, waiting for I/O and other processes, and operating system overhead. This is also referred to as execution time.
  - Throughput :The total amount of work done in a given time.
  - CPU execution time : Total time a CPU spends computing on a given task (excludes time for I/O or running other programs). This is also referred to as simply CPU time.

Performance Definition

- For some program running on machine X,

  $\text{Performance} = 1 \,/\, \text{Execution time}_X$
- "X is n times faster than Y"

  $\text{Performance}_X \,/\, \text{Performance}_Y = n$

Problem:

- machine A runs a program in 20 seconds
- machine B runs the same program in 25 seconds
- how many times faster is machine A?

$$\frac{25}{20} = 1.25$$

Basic Measurement

- Comparing Machines
  - Metrics
    - Execution time
    - Throughput
    - CPU time
    - MIPS – millions of instructions per second

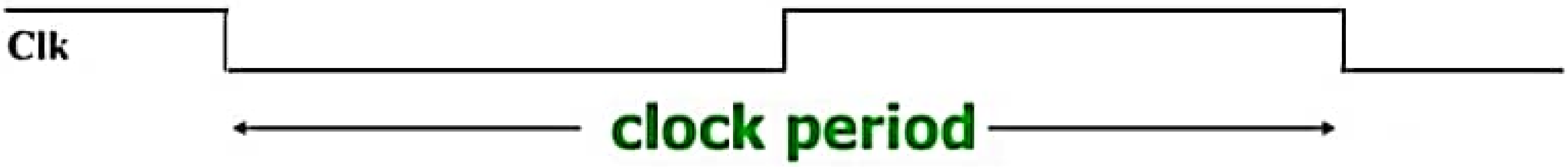
- Comparing Machines
  - Metrics
    - Execution time
    - Throughput
    - CPU time
    - MIPS – millions of instructions per second
    - MFLOPS – millions of floating point operations per second
- Comparing Machines Using Sets of Programs
  - Arithmetic mean, weighted arithmetic mean

- Execution time
- Throughput
- CPU time
- MIPS – millions of instructions per second
- MFLOPS – millions of floating point operations per second

- Comparing Machines Using Sets of Programs
  - Arithmetic mean, weighted arithmetic mean
  - Benchmarks

Computer

- A computer clock runs at a constant rate and determines when events take placed in hardware.



- The clock cycle time is the amount of time for one **clock period** to elapse (e.g. 5 ns).
- The clock rate is the inverse of the clock cycle time.
- For example, if a computer has a clock cycle time of 5 ns, the clock rate is:

1

when events take placed in hardware.

- The clock cycle time is the amount of time for one **clock period** to elapse (e.g. 5 ns).
- The clock rate is the inverse of the clock cycle time.
- For example, if a computer has a clock cycle time of 5 ns, the clock rate is:

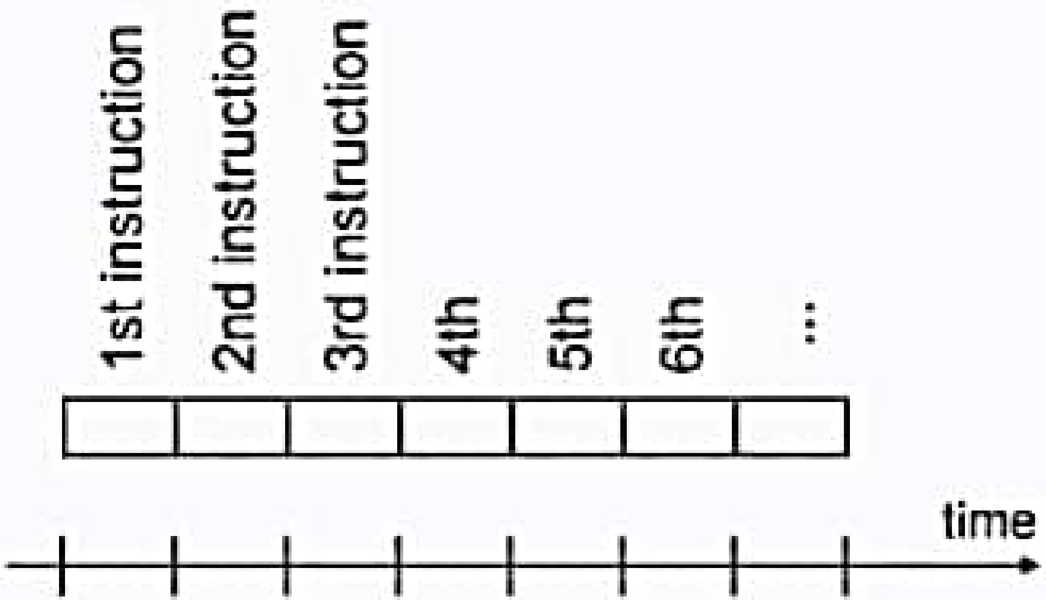
$$\frac{1}{5 \times 10^{-9} \text{ sec}} = 200 \text{ MHz}$$

Prefix	Symbol	Multiplier	
exa	E	10^{18}	1,000,000,000,000,000,000
peta	P	10^{15}	1,000,000,000,000,000
tera	T	10^{12}	1,000,000,000,000
giga	G	10^{9}	1,000,000,000
mega	M	10^{6}	1,000,000
kilo	k	10^{3}	1,000
hecto	h	10^{2}	100
deka	da	10^{1}	10
deci	d	10^{-1}	0.1
centi	c	10^{-2}	0.01
milli	m	10^{-3}	0.001
micro	μ	10^{-6}	0.000,001
nano	n	10^{-9}	0.000,000,001
pico micro micro	p $\mu\mu$	10^{-12}	0.000,000,000,001
femto	f	10^{-15}	0.000,000,000,000,001
atto	a	10^{-18}	0.000,000,000,000,000,001

peta	P	$10^{[illegible]}$	1,000,000,000,000,000
tera	T	10^{12}	1,000,000,000,000
giga	G	10^{9}	1,000,000,000
mega	M	10^{6}	1,000,000
kilo	k	10^{3}	1,000
hecto	h	10^{2}	100
deka	da	10^{1}	10
deci	d	10^{-1}	0.1
centi	c	10^{-2}	0.01
milli	m	10^{-3}	0.001
micro	μ	10^{-6}	0.000,001
nano	n	10^{-9}	0.000,000,001
pico micro micro	p $\mu\mu$	10^{-12}	0.000,000,000,001
femto	f	10^{-15}	0.000,000,000,000,001
atto	a	10^{-18}	0.000,000,000,000,000,001

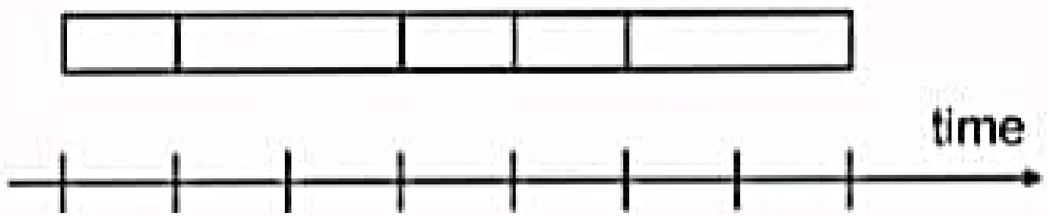
How Many Cycles are Required for Program?

- Could assume that # of cycles = # of instructions



- This assumption is incorrect, different instructions take different amounts of time on different machines.

Different Numbers of Cycles for Different Instructions



- Division takes more time than addition
- Floating point operations take longer than integer ones
- Accessing memory takes more time than accessing registers

- A given program will require
  - some number of instructions (machine instructions)
  - some number of clock cycles
  - some number of seconds
- We have a vocabulary that relates these quantities:
  - clock cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)
    - *a floating point intensive application might have a higher CPI*

Computing CPU Time

- The time to execute a given program can be computed as

  CPU time = CPU clock cycles x clock cycle time

- Since clock cycle time and clock rate are reciprocals

  CPU time = CPU clock cycles / clock rate

- The number of CPU clock cycles can be determined by

  CPU clock cycles = (instructions/program) x (clock cycles/instruction)

  = Instruction count x CPI

  which gives

  CPU time = Instruction count x CPI x clock cycle time

  CPU time = Instruction count x CPI / clock rate

- The units for CPU time are

- The time to execute a given program can be computed as

  CPU time = CPU clock cycles x clock cycle time

- Since clock cycle time and clock rate are reciprocals

  CPU time = CPU clock cycles / clock rate

  clock rate= 1/cloc cycl

- The number of CPU clock cycles can be determined by

  CPU clock cycles = (instructions/program) x (clock cycles/instruction)

  = Instruction count x CPI

  which gives

  CPU time = Instruction count x CPI x clock cycle time

  CPU time = Instruction count x CPI / clock rate

- The units for CPU time are

$$\text{CPU time} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{clock cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{clock cycle}}$$

CPU Time Example

- **Example 1:**
  - CPU clock rate is 1 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time?

$$45{,}000{,}000 * (1 / 1{,}000{,}000) = 45 \text{ seconds}$$

- **Example 2:**
  - CPU clock rate is 500 MHz

- **Example 1:**
  - CPU clock rate is 1 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time?

$$45,000,000 * (1 / 1,000,000) = 45 \text{ seconds}$$

- **Example 2:**
  - CPU clock rate is 500 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time

- Program takes 45 million cycles to execute
- What's the CPU time?

45,000,000 * (1 / 1,000,000) = 45 seconds

- **Example 2:**
  - CPU clock rate is 500 MHz
  - Program takes 45 million cycles to execute
  - What's the CPU time

45,000,000 * (1 / 500,000,000) = 0.09 seconds

Which factors are affected by each of the following?

	instr. Count	CPI	clock rate
Program	x		
Compiler	x	x	
Instr. Set Arch.	x	x	
Organization		x	x
Technology			x

CPI Example

- **Example:** Let assume that a benchmark has 100 instructions:

  25 instructions are loads/stores (each take 2 cycles)

  50 instructions are adds (each takes 1 cycle)

  25 instructions are square root (each takes 50 cycles)

What is the CPI for this benchmark?

instructions:

25 instructions are loads/stores (each take 2 cycles)

50 instructions are adds (each takes 1 cycle)

25 instructions are square root (each takes 50 cycles)

What is the CPI for this benchmark?

$$CPI = ((0.25 * 2) + (0.50 * 1) + (0.25 * 50)) = 13.5$$

Computing CPI

- The CPI is the average number of cycles per instruction.
- If for each instruction type, we know its frequency and number of cycles need to execute it, we can compute the overall CPI as follows:

$$CPI = \sum CPI \times F$$

- For example

Op	F	CPI	CPI x F	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%

overall CPI as follows:

$$CPI = \sum CPI \times F$$

- For example

Op	F	CPI	CPI x F	% Time
ALU	50%	1	.5	23%
Load	20%	5	1.0	45%
Store	10%	3	.3	14%
Branch	20%	2	.4	18%
Total	100%		2.2	100%

- Suppose we have two implementations of the same instruction set architecture (ISA).

  For some program,

  Machine A has a clock cycle time of **10 ns.** and a CPI of **2.0**
  Machine B has a clock cycle time of **20 ns.** and a CPI of **1.2**

- Which machine is faster for this program, and by how much?

Assume that # of instructions in the program is 1,000,000,000.

$\text{CPU Time}_A = 10^9 * 2.0 * 10 * 10^{-9} = 20$ seconds

$\text{CPU Time}_B = 10^9 * 1.2 * 20 * 10^{-9} = 24$ seconds

Machine A is faster

$$\frac{24}{20} = 1.2 \text{ times}$$

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C

The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

# of cycles for first code = (2 * 1) + (1 * 2) + (2 * 3) = 10 cycles

# of cycles for second code = (4 * 1) + (1 * 2) + (1 * 3) = 9 cycles

(respectively).

The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

# of cycles for first code = (2 * 1) + (1 * 2) + (2 * 3) = 10 cycles

# of cycles for second code = (4 * 1) + (1 * 2) + (1 * 3) = 9 cycles

10 / 9 = 1.11

CPI for first code = 10 / 5 = 2

CPI for second code = 9 / 6 =

2. CPI Example

- Assume a processor with instruction frequencies and costs
  - Integer ALU: 50%, 1 cycle
  - Load: 20%, 5 cycle
  - Store: 10%, 1 cycle
  - Branch: 20%, 2 cycle
- Which change would improve performance more?
  A: "Branch prediction" to reduce branch cost to 1 cycle?
  B: "Cache" to reduce load cost to 3 cycles?
- Compute CPI

	INT	LD	ST	BR	CPI
Base					
A					
B					

Another CPI Example

- Assume a processor with instruction frequencies and costs
  - Integer ALU: 50%, 1 cycle
  - Load: 20%, 5 cycle
  - Store: 10%, 1 cycle
  - Branch: 20%, 2 cycle
- Which change would improve performance more?
  - A. "Branch prediction" to reduce branch cost to 1 cycle?
  - B. "cache" to reduce load cost to 3 cycles?
- Compute CPI
  - Base = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*2 = 2 CPI
  - A = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*1 = 1.8 CPI
  - B = 0.5*1 + 0.2*3 + 0.1*1 + 0.2*2 = 1.6 CPI **(winner)**

- Compute CPI

	INT	LD	ST	BR	CPI
Base					
A					
B					

3 Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for a program, and machine B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which machine is faster for this program, and by how much?

4.

Instruction class	CPI for this instruction class
A	1
B	2
C	3

Code sequence	Instruction counts for instruction class		
	A	B	C
1	2	1	2

Example #1

- Computer A has a clock cycle time of 250 ps and a CPI of 2.0 for a program, and machine B has a clock cycle time of 500 ps and a CPI of 1.2 for the same program. Which machine is faster for this program, and by how much?

(1) CPU clock cyclesA=I x 2.0
CPU clock cyclesB=I x 1.2
I is the number of instructions for this program

(2) CPU timeA=I x 2.0 x 250 ps= 500 x I ps
CPU timeB=I x 1.2 x 500 ns = 600 x I ps

(3) CPU performanceA= 1 / timeA
CPU performanceB= 1 / timeB

(4) Speedup= performanceA/ performanceB=1.2