EEEN301 Embedded systems

Lecture 6 2023

Performance

The performance concepts

- Throughput: total amount of work done in a given time
- Response time: time between start and completion of a task.
- Applications usually have varied demand on throughput and response time.

- Example: do the following changes to a computer system increase throughput or decrease response time?

  - 1. Replace the processor in a computer with a faster one?

  - 2. Adding more processors to a computer system?

Performance vs. execution time

- To maximize performance, we want to minimize execution time

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- For two computers X and Y, if the performance of X is greater than that of Y, then

$$\text{Performance}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

- If X is n-times faster than Y, then

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n \quad \Rightarrow \quad \frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Measuring performance

- Time is the measure of computer performance
  - The computer that performs the same amount of work in the least time is the fastest.

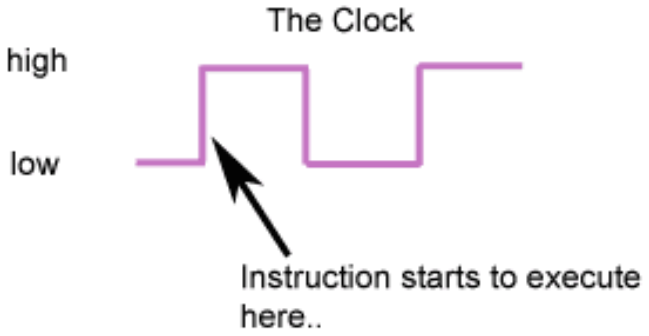
- Different time concepts
  - CPU time/ Clock time / response time / elapsed time

  - Elapsed time: The total time to complete a task, including disk accesses, memory accesses, I/O operations, operating system overhead, etc.

  - CPU time: the time the CPU spends on a task
    - Does not include time spent waiting for I/O or running other programs.
    - User CPU time: CPU time spent in the program
    - System CPU time: CPU time spent in the operating system performing tasks on behalf of the program

Other methods to measure performance

- Clock cycles: how fast the hardware can perform basic functions.
  - Almost all computers are constructed using a clock that determines when events take place in hardware.



  - Clock period: the time for a complete clock cycle.
  - Clock rate: the inverse of clock period.

0.1 s ⇨ 10 Hz

CPU performance

- CPU execution time

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

  - Since clock period (cycle time) and clock rate are inverses

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

- How can we improve performance based on the above formula?

  - A. Reducing the number of clock cycles required for a program
  - B. Decreasing the clock cycle time.
  - C. Increasing the clock rate
  - D. All of the above.

Example: improve performance

- An important program runs in 10 seconds on computer A
  - Computer A has a 2GHz clock.

- We are trying develop a computer B which will run the program in 6 seconds.
  - Due to the higher clock rate, computer B requires a different CPU design and requires 1.2 times as many clock cycles as computer A for the program.

- Question: what clock rate should computer B have in order to run the program in 6 seconds?

- The number of clock cycles required for A

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10 \text{ seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10 \text{ seconds} \times 2 \times 10^9 \frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \text{ cycles}$$

- CPU time for B then is

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

Instruction performance

- Average number of clock cycles per instruction (CPI):

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \text{Average clock cycles per instruction}$$

- Average clock cycles per instruction (CPI) provides a way of comparing two different implementations of the same instruction set architecture.

Instruction set architecture: an abstract interface between the hardware and software that encompasses all the information necessary to write a machine language program.
Instructions, registers, memory access, I/O, etc.

Example question

- Suppose we have two implementations of the same instruction set architecture.
  - unit ps = picosecond = 10^{-12} seconds = 0.000,000,000,001 seconds
  - Computer A has a clock cycle time of 250ps and a CPI of 2.0 for some program.
  - Computer B has a clock cycle time of 500ps and a CPI of 1.2 for the same program.
- Question: which computer is faster for this program and by how much?
  - A. Computer A is faster
  - B. Computer B is faster

- Each computer executes the same number (I) of instructions for the program.

$$\text{CPU clock cycles} = \text{I x CPI}$$

- Calculate the CPU time for each computer:

$$\text{CPU time}_A = \text{CPU clock cycles}_A \times \text{Clock cycle time}$$

$$= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}$$

$$\text{CPU time}_B = I \times 1.2 \times 500 \text{ ps} = 600 \times I \text{ ps}$$

- Computer A is faster by

$$\frac{\text{CPU performance}_A}{\text{CPU performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{600 \times I \text{ ps}}{500 \times I \text{ ps}} = 1.2$$

Example: comparing code segments

- A compiler designer is trying to decide between two code sequences for a particular computer

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

- Details of the two code sequences

Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

- Question: which code sequence will be faster?

Answer

- Sequence 1: 2+1+2=5 instructions
- Sequence 2: 4+1+1=6 instructions

- Calculate clock cycles

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$

$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

- Calculate CPI

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

Summary of performance analysis

- General formula

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

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

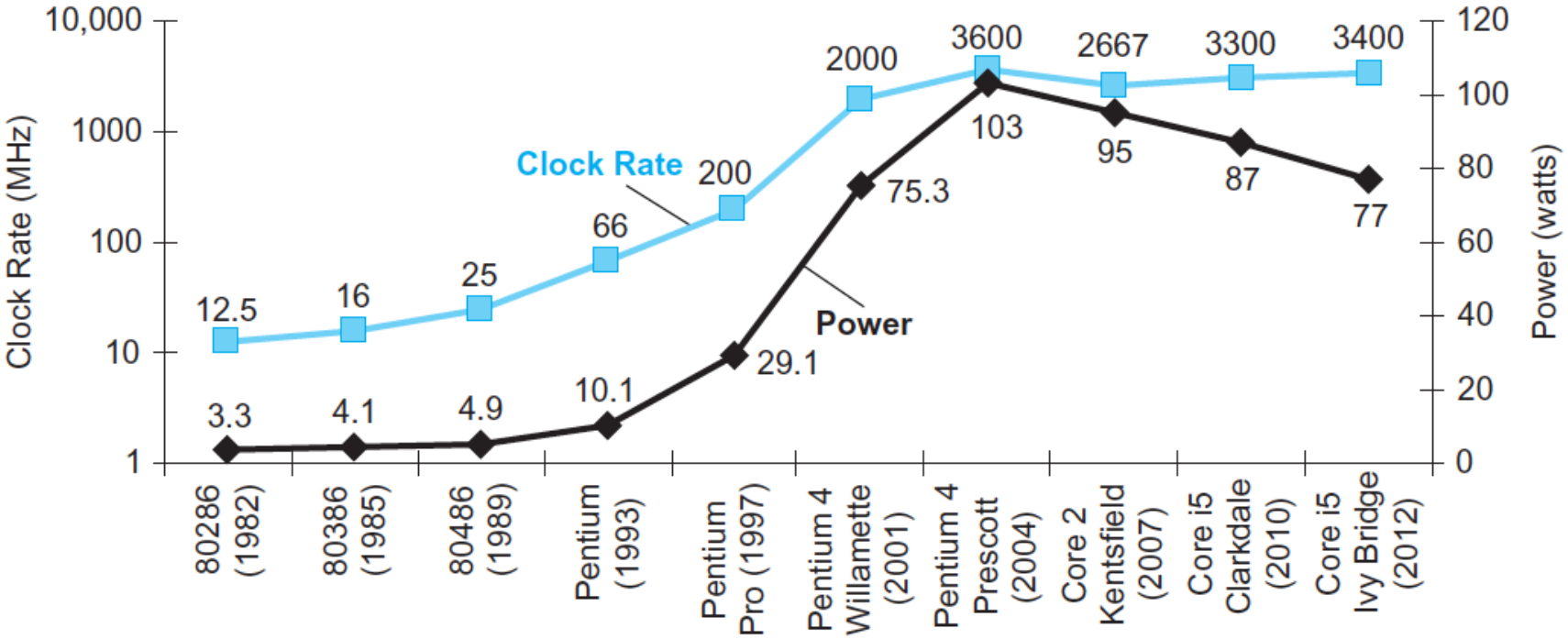
- The only complete and reliable measure of computer performance is time.

Understand program performance

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more floating-point operations, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.

The power wall

- Clock rate and power for Intel x86 microprocessors over eight generations and 30 years.



Quick question

- The energy consumption of a processor

$$Power \propto 1/2 \times Capacitive\ load \times Voltage^2 \times Frequency\ switched$$

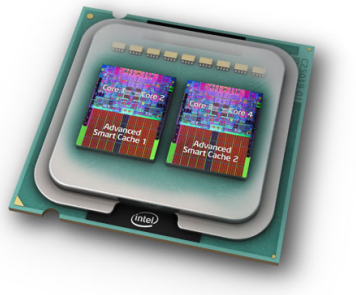
- Which approach is the most effective in reducing power consumption?
  - A. Reduce the size of each transistor
  - B. Reduce operating voltage
  - C. Reduce clock rate
  - D. Design simple processors

Problem of lowering the voltage

- Further lowering the voltage appears to make the transistors too leaky.
- About 40% of the power consumption is due to leakage.
- Also, limited by I/O signal integrity
- Solution:
  - Attach large heatsinks to devices to increase cooling
  - Turn off parts of the chip that are not used in a given clock cycle

Switch from uniprocessors to multiprocessors

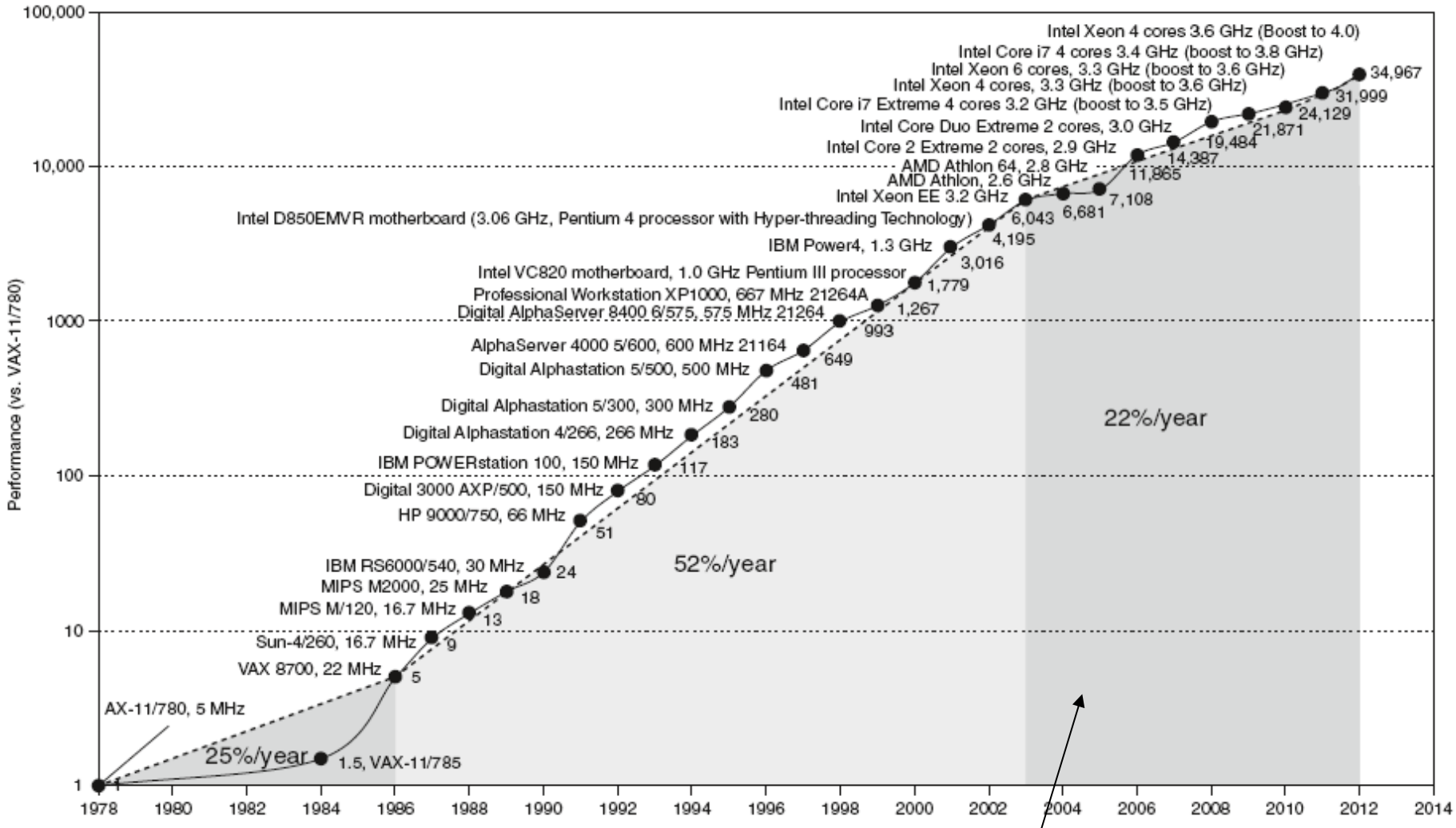
- All desktop and server companies are shipping microprocessors with multiple processors per chip
  - Multi-core processor
    - Dual core processor, quad core processor, etc.
  - Improve throughput rather than response time



- In the past, program performance can be doubled every 18 months without having to change a line of code.

- Today, to further improve performance, program needs to be explicitly designed to run on parallel processors.

Processor performance



Constrained by power, instruction-level parallelism, memory latency

Amdahl's law

- The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

6 second process + 6 second process = ?

50% improvement + no improvement = ?

Example

- Suppose that program runs in 100 seconds on a computer.
- Multiply operation responsible for 80 seconds of the total time.

- Question: how much should the multiplication operation be improved if the program needs to be run five times faster?

$$\text{Execution time after improvement} = \frac{80 \text{ seconds}}{n} + (100 - 80 \text{ seconds})$$

- Since the performance should be five times faster

$$20 \text{ seconds} = \frac{80 \text{ seconds}}{n} + 20 \text{ seconds}$$

$$0 = \frac{80 \text{ seconds}}{n}$$

- Can't be done!

Millions instructions per second

- A measurement of program execution speed based on the number of millions of instructions.

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6}$$

  - Faster computer means bigger MIPS.

- Problems with using MIPS for comparing computers
  - Different instruction set
  - MIPS varies between programs on the same computer

$$\text{MIPS} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

  - MIPS can vary independently from performance.

SPEC CPU Benchmark

- SPEC – Standard Performance Evaluation Cooperative
  - Funded and supported by a number of computer vendors to create standard sets of benchmarks for computer systems.

- SPEC CPU2006
  - CINT2006: 12 integer benchmarks
    - C compiler, chess program, quantum computer simulation, etc.
    - http://www.spec.org/cpu2006/CINT2006/
  - CFP2006: 17 floating-point benchmarks
    - Finite element modeling, particle method codes for molecular dynamics, sparse linear algebra code for fluid dynamics, etc.
    - http://www.spec.org/cpu2006/CFP2006/

- SPECratio:
  - dividing the execution time on a reference processor by the execution time on the measured computer.

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i \right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 for i7 Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
Σssj_ops/Σpower =		2,490

Fallacy: Low Power at Idle

- Look back at i7 power benchmark
  - At 100% load: 258W
  - At 50% load: 170W (66%)
  - At 10% load: 121W (47%)
- Google data center
  - Mostly operates at 10% – 50% load
  - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load
- Laptops typically use 'mobile' versions of processors that have better power management.
- The Apple M1 is an example of a vast improvement in power efficiency.

Quick exercise

- Which of the following can affect program performance
  - A. Your code
  - B. The compiler
  - C. The CPU
  - D. The I/O subsystem

Understanding Performance

- Algorithm
  - Determines number of operations executed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed