# Chapter 02 – Computer Evolution and Performance

Week – 03

# Topics to Cover

- Later Generations
- 2.2 Designing for Performance
- 2.3 Multicore Processors
- 2.4 The Evolution of Intel Architecture
- 2.6 Performance Assessment
- Examples

# Improvements that Came with IC Designs

1. Since the geometric sizes of components (transistors) are small, the time delays for signal propagation are short.
2. Components being small consume less power, and dissipate less heat, the circuit is thermally stable.
3. Every semiconductive component within an IC has virtually identical operating parameters, device mismatches are less.
4. This means that at high clock rates! All skew times, delay times, hold times... Everything is fixed and predictable!
5. The components being small can switch states at a faster pace thus providing a boost in speed/performance.

Skew time: Delay of clock signal for different components. Hold time: Amount of time for data to get stable.

# Later Generations (Beyond Third Generation)

- Beyond the third generation, there have been a number of later generations, based on advances in ‘integrated circuit’ technology.
- With the rapid pace of technology, the high rate of introduction of new products, and the improvements in hardware as well as software, the classification by generation becomes less clear and less meaningful. ($4^{th}$ generation is Microprocessor and $5^{th}$ is Artificial intel)
- In the later generations, two most important changes are made in:

1. Semiconductor memory
2. Microprocessors

# 1. Improvements in Semiconductor Memory

**Read**

- In 1970 Fairchild produced capacious semiconductor memory.
- This chip was about a sixteenth of an inch in diameter.
- It Holds 256 bits, chips can be connected to form larger memories.
- Non-destructive read (data is restored after a Read operation)
- Much faster than core memory. (70 billionth of a second to read a bit)
- Memory Capacity approximately doubles each year.
- Since 1970, the semiconductor memory has been through 13 generations, as of declining cost per bit and declining access time.
- 1k, 4k, 16k, 64k, 256k, 1M, 4M, 16M, 64M, 256M, 1G, 4G, 8G & so on

# 2. Improvements in Microprocessors

**Read**

- 1971 – Intel 4004 Cpu
  - First microprocessor
  - All CPU components on a single chip
  - 4 bit
- Followed in 1972 by 8008
  - 8 bit
  - Both CPU designed for specific applications
- 1974 - 8080
  - Intel's first general purpose microprocessor, faster, rich instruction set, large addressing capacity (data bus width increased)
- 1974 - 8086
  - Powerful, general purpose
  - 16-bit microprocessor
- 1981 - HP
  - 32-bit
  - Single-chip microprocessor
- 1985 - 80386
  - Intel's first 32-bit processor
  - Evolution goes on till Core-i

# 2.2 Designing For Performance

- Year by year, the cost of computer systems continues to drop dramatically, while the performance and capacity of those systems continue to rise equally dramatically.
- What is fascinating is that the basic building blocks for today's computer miracles are virtually the same as those of the IAS computer from over 60 years ago.
- Power in todays computer is the result of relentless pursuit of speed by processor chip manufacturers.
- The evolution of these chips continue to support 'Moore's law' claim.

# Computer Performance

- **Computer Performance** is the amount of work accomplished by a computer system.(Depends upon clock rate, cores,memory hierarchy)
- The word **Performance** means ‘how well is the computer doing the work it is supposed to do?’.
- It basically depends on the ‘response time’, ‘throughput’ and ‘execution time’ of a computer system.
- ‘Response time’ is how fast the computer is able to respond.
- ‘Throughput’ is the how quickly the instruction queue is being run.
- ‘Execution time’ is the amount of time a program takes to completion

# Microprocessor Speed

- As per Moore's law the chipmakers can generate a new generation of chips every three years – with four times as many transistors.
- In microprocessors, the addition of new circuits, and the speed boost that comes from reducing the distances between them, has improved performance four or five fold every three years.
- Greater speed in the execution of a given instruction could be gained by the use of more complex circuitry in the ALU, allowing sub operations to be carried out in parallel.
- Another way of increasing speed was to increase the width of the data path between main memory and the CPU. (reduce clock. Cycles)

# Techniques For Improved Processor Speed

- The raw speed of microprocessor will not achieve its potential unless it is fed a constant stream of work to do in the form of computer instructions. That is the processor **idle time** should be reduced.
- **Multitasking** means it could run multiple programs at the same time.
- **Superscalar** technology allows multiple instruction to execute in parallel. So the CPU executes several instruction during a clock cycle.

➢The organizational techniques for feeding the processor are following:

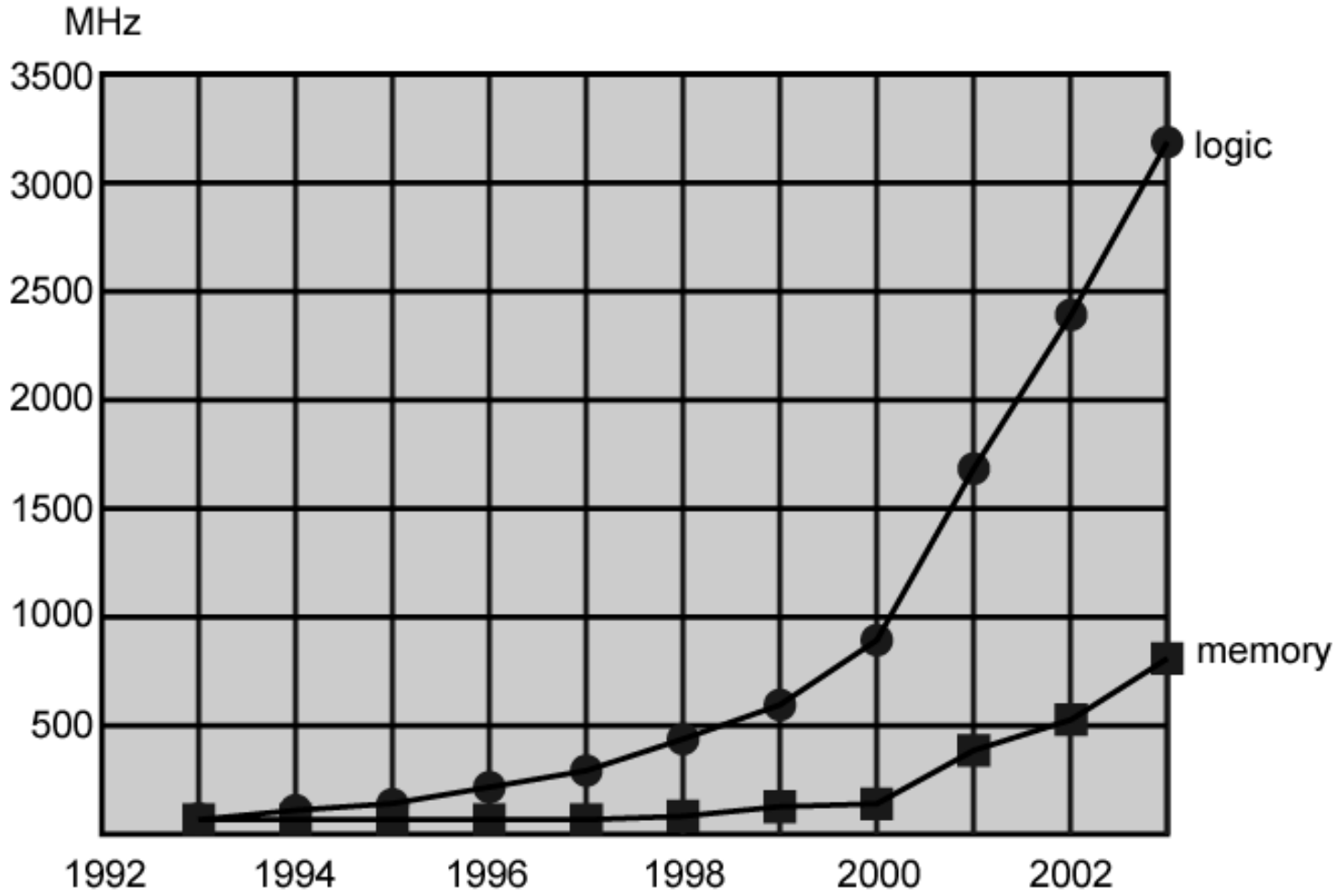
1. *Pipelining*
2. *Branch prediction*
3. *Data flow analysis*
4. *Speculative execution*

1. **Pipelining:** With pipelining, a processor can simultaneously work on multiple instructions. The processor overlaps operations e.g. while one instruction is being executed, the computer is decoding the next instruction. (instruction fetch → Decode → Execute)
2. **Branch prediction:** The processor looks ahead in the instruction coded fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next. This increases the amount of work available for the processor to execute. (if-then-else statements)
3. **Data flow analysis:** To determine those parts of a program to which a particular value assigned to a variable might propagate. A **data-flow analysis** is reaching definitions. This prevents unnecessary delay. Control Flow Graph
4. **Speculative execution:** Using branch prediction and data flow analysis, some processor speculatively (secretly) execute instruction ahead of their appearance in the program execution, holding the results in temporary locations. (parallel instruction computing)

# Performance Balance

- Processor speed has increased.
- Memory capacity has increased.
- Memory speed lags behind processor speed.
- This means that the speed with which data can be transferred between main memory and the processor has lagged badly.
- If memory or the pathway fails to keep pace with the processor's insistent processing demands, the processor stalls in a wait state, and valuable processing time is lost.
- The Key to tackle this problem is to **look for a performance balance** between processor component, main memory, I/O devices and buses.

# Logic and Memory Performance GAP



# Ways to Achieve ‘Performance Balance’ (Memory)

- Performance balance is a need to adjust the organization and architecture to compensate for the mismatch among the capabilities of the various components.
- Here are the solution of ‘performance balance’ in memory aspect:

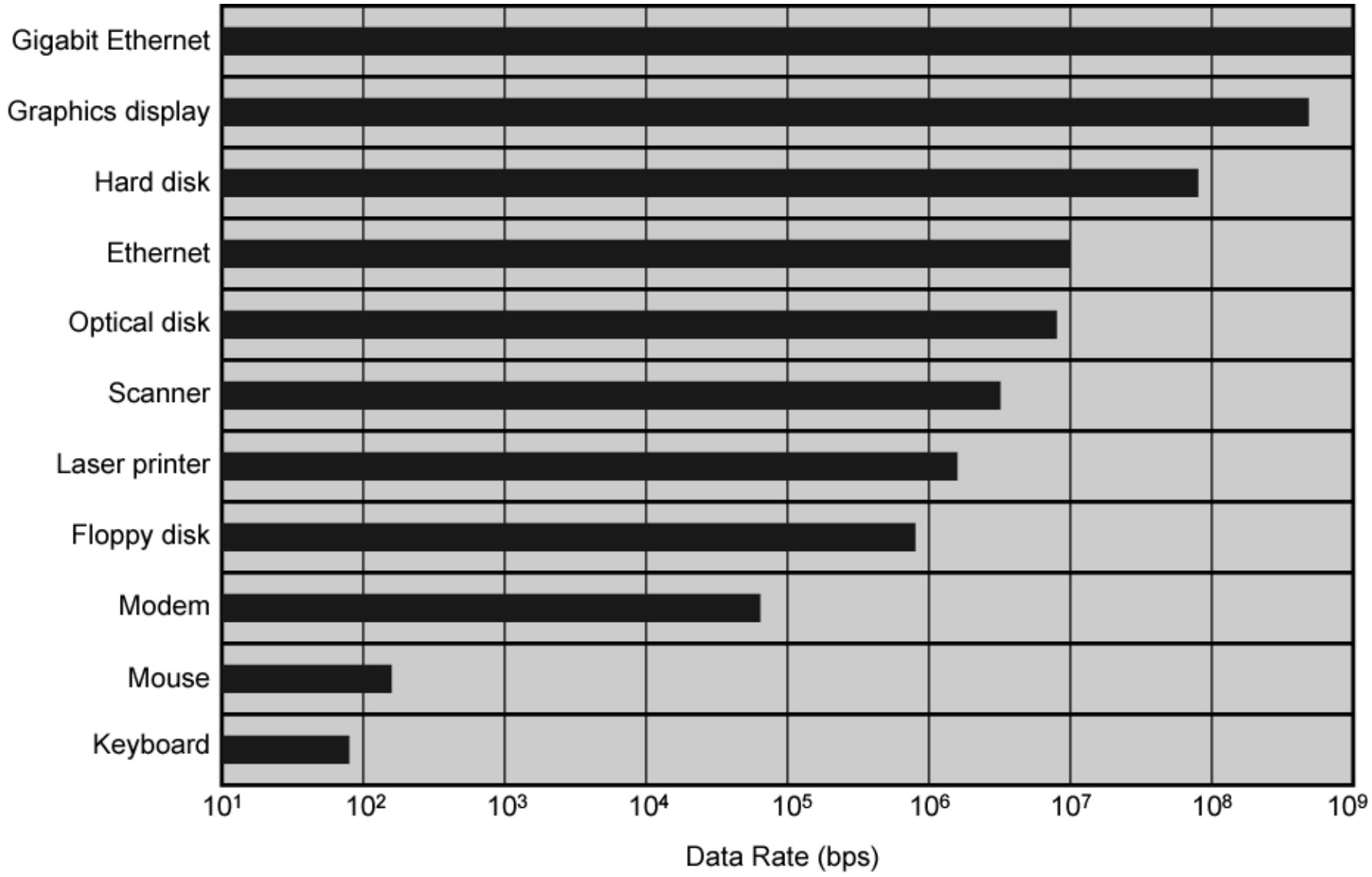
1. Increase the number of bits retrieved at one time by making DRAMs ‘wider’ rather than ‘deeper’.
2. Change the DRAM interface by including a cache as buffer memory.
3. Reduce the frequency of ‘memory access’ by adding more complex caches and caches on the processor chip.
4. Increase the interconnection bandwidth by using higher-speed buses and a hierarchy of buses.

# Ways to Achieve ‘Performance Balance’ (I/O)

- The Input/Output (I/O) devices create tremendous data throughput demands. **Throughput** is the amount of data moved in a given time.
- The processor can handle this data all at once, but there remains the problem of getting that data moved between processor & peripherals.
- Here are the solution of ‘performance balance’ in I/O aspect:

1. Include cache and buffer schemes. Once buffer is filled, send it all at once to CPU.
2. Use high speed interconnection buses and a hierarchy of buses.
3. Use-multiple-processor configurations to meet the I/O demands.

# Typical I/O Devices ‘Data Rates’



# HDW Approaches to Increasing the ‘Processor Speed’

- There are three approaches to achieving increased processor speed:

1. Increase the hardware speed of the processor due to shrinking the size of the logic gates on the processor chip, so that more gates can be packed together more tightly, thus the propagation time for signals is reduced.
2. Increase the size and speed of caches between the processor and main memory by dedicating a portion of the processor chip itself to the cache, cache access time drops significantly.
3. Change the processor organization and architecture that increases the effective speed of instruction execution by using multi-core CPUs, and by placing multiple ALUs to allow ‘parallel processing’.

# Problems with Increased Clock Speed & Logic Density

- Increasing the processor clock rate means that individual operations are executed more rapidly. (MHz to GHz transition)
- However, as clock speed and logic density increase, following problems become more significant:

1. **Power:** As the density of logic and the clock speed on a chip increase, so does the power density (watt/cm$^2$). And heat dissipation becomes a serious design issue.
2. **RC delay:** Delay of signal on a chip increases as the RC product increases. Thinner wire increase resistance, closer wire capacitance.
3. **Memory latency:** Memory speed lags processor speed.(Same issue)

# Increased Cache Capacity

- Typically two or three levels of cache between processor and main memory
- Chip density increased
  - More cache memory on chip
    - Enabling Faster cache access
- Pentium chip devoted about 10% of chip area to cache
- Pentium 4 devotes about 50%

# More Complex Execution Logic

- In logic, scaling implies area scaling, that is packing more and more circuits into a smaller area by means of materials and design innovations.
- The ability to transmit signals to all components as rapidly as possible, while minimizing signal losses in shrinking geometries, is critical for device scaling.
- so scaling focuses on obtaining progressively better performance from existing materials and design. (RC delay limits scaling – design limitation)
- Enable parallel execution of instructions
- Two important design approaches have been pipeline and superscalar
- **Pipeline** works like assembly line
  - Different stages of execution of different instructions occur at the same time along the pipeline
- **Superscalar** allows multiple pipelines within single processor
  - Instructions (thread) that do not depend on one another can be executed in parallel.

# Diminishing (become smaller) Returns

- Internal organization of processors has become complex
  - We can now get a great deal of parallelism
  - Further significant increases likely to be relatively modest
- Benefits from cache are reaching limit
- Increasing clock rate runs into power dissipation problem
  - Some fundamental physical limits are being reached

- Solution: Rather than building a more complex processor, use **multicore** computer chip.

(Half)

# 2.3 Multicore Processors

- Placing multiple processors on the same chip, with a large shared cache and same clock is referred to as multiple cores, or **multicore.**
- Within a processor the increase in performance is equal to the square root of the increase in complexity.
- If software can support multiple processors, doubling number of processors almost doubles performance.
- Strategy is to use two simpler processors on the chip rather than one more complex processor. Level-1 cache is dedicated to a processor.
- With two processors, larger caches are justified. Because the power consumption of memory cells on a chip is less than processing logic.

Multi-threading: operating system utilise multiple cores for processing an application.

# Features of Multicore Processors

- A **core** can be thought of as an individual processor. A dual-core processor, therefore has two internal processors, a quad-core model has four. More cores are useful for multi-tasking; for example, you can run two applications at the same time, each one having access to its own dedicated processor. (multi-threading= OS uses multiple cores)
- whether they're a Core i3, i5 and i7, the differences in performance come from which features are enabled or disabled, the clock speed and how many cores each one has.

| Model | Core i3 | Core i5 | Core i7 |
|---|---|---|---|
| Number of cores | 2 | 4 | 8 |
| Hyper-threading | No | Yes | Yes |
| Turbo boost | No | Yes | Yes |

Hyper-threading: Single processor Appears to the OS as two. Parallelism.

Turbo-boost: Technology that automatically allows cores to run faster than the base operating frequency, and thus Raising performance.

# Higher Clock Speed vs. More Cores?

- Ok, so you now understand the benefits of a higher clock speed and the performance boosts more cores can offer. Do you go for a processor with a lower clock speed but more cores? Or one with fewer cores but a higher clock speed?

- If possible, you want to go for the one with the highest clock speed and the highest amount of cores. Due to budgets, however, this isn't always possible and there is usually a trade-off between cores and clock speed.
- So which one is better?

# More cores, slower clock speed

Multi-threading: operating system utilise multiple cores for processing an application threads.

- **Pros**
  - Applications that support multi-threading will greatly benefit from having a higher number of cores at their disposal
  - Increasing the number of cores in your CPU is a cost-effective way of increasing performance
  - Multi-threading support for applications will continue to improve over time
  - You will be able to run more apps at once without seeing performance drops
  - Great for running multiple virtual machines
- **Cons**
  - Lower single-threaded performance than a higher clock speed processor

# Fewer cores, higher clock speed

- **Pros**
  - Better single threaded performance
  - Lower cost option
- **Cons**
  - Fewer cores to split between applications
  - Not as strong multi-threading performance
- The best thing to do in most cases is to look into the support your applications of choice provide for multi-threading. Following this you can decide whether you'd be better off with, more cores/clock speed.

# 2.4 The Intel x86 Evolution (1)

**Read**

- **8080**
  - first general purpose microprocessor
  - 8 bit data path
  - Used in first personal computer – Altair
- **8086** – 5MHz – 29,000 transistors
  - much more powerful
  - 16 bit
  - instruction cache, prefetch few instructions
  - 8088 (8 bit external bus) used in first IBM PC
- **80286**
  - 16 Mbyte memory addressable
  - up from 1Mb
- **80386**
  - 32 bit
  - Support for multitasking
- **80486**
  - sophisticated powerful cache and instruction pipelining
  - built in maths co-processor

# The Intel x86 Evolution (2)

Read

- **Pentium**
  - Superscalar
  - Multiple instructions executed in parallel
- **Pentium Pro**
  - Increased superscalar organization
  - Aggressive register renaming
  - branch prediction
  - data flow analysis
  - speculative execution
- **Pentium II**
  - MMX technology
  - graphics, video & audio processing
- **Pentium III**
  - Additional floating point instructions for 3D graphics

# The Intel x86 Evolution (3)

Read

- **Pentium 4**
  - Note Arabic rather than Roman numerals
  - Further floating point and multimedia enhancements
- **Core**
  - First x86 with dual core
- **Core 2**
  - 64 bit architecture
- **Core 2 Quad** – 3GHz – 820 million transistors
  - Four processors on chip

- x86 architecture dominant outside embedded systems
- Organization and technology changed dramatically
- Instruction set architecture evolved with backwards compatibility
- ~1 instruction per month added
- 500 instructions available
- See Intel web pages for detailed information on processors

# 2.6 Performance Assessment

- For new systems, performance is one of the key parameters to consider, along with cost, size, security, reliability and power consumption along with weight and physical dimensions.

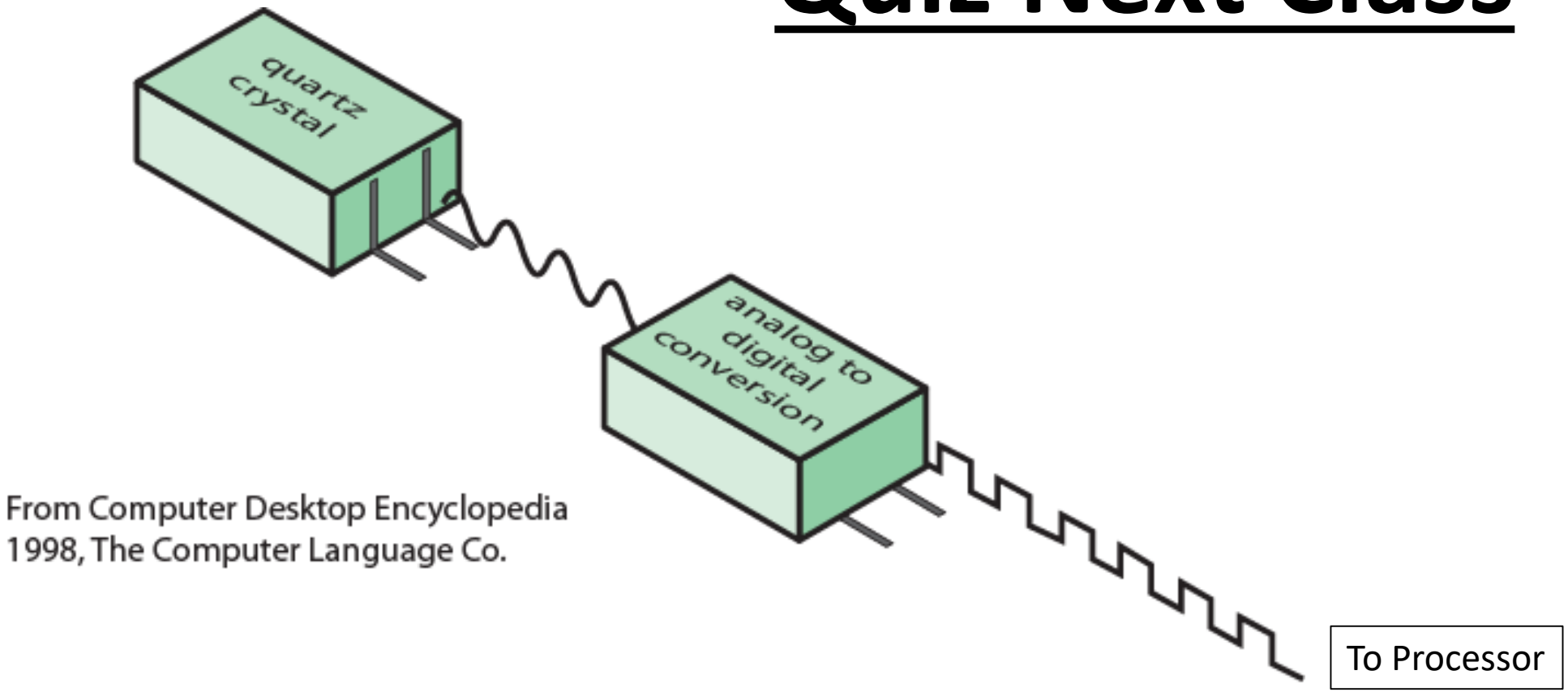
- ‘Application performance’ not only depends on the raw speed of the processor but also on the instruction set, choice of implementation language, efficiency of the compiler, and skill of the programmer.

# Clock Speed and Instructions per Second

- Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and so on, are governed by a **system clock,** which synchronizes all functions.
- All operations begin with the pulse of the clock, and the speed of the processor is dictated by the pulse frequency produced by the clock, measured in cycles per second, or Hertz (Hz).
- For example, a 1-GHz processor receives 1 billion pulses per second.
- The rate of pulses is known as the **clock rate,** or **clock speed.**
- One increment or pulse of the clock is referred to as a **clock cycle/tick**.
- The time between pulses is the **cycle time, τ (tau).**

# System Clock

**Quiz Next Class**

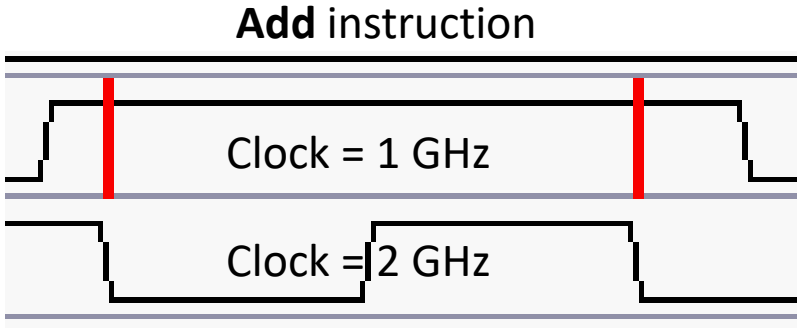



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

# Clock Speed Comparison of different CPUs

- Actions in the processor require signals to be sent from one processor element to another.
- Signals in CPU take time to settle down to 1 or 0 before being sent.
- Some signals may change at different speeds.
- Thus operations must be synchronized so that proper electrical signal (voltage) values are available for each operation.
- Moreover, the execution of an instruction involves a number of discrete steps, such as fetching the instruction from memory, decoding the instruction, loading and storing data and ALU operations
- Thus most instructions require multiple clock cycles to complete. 1/12
- Thus, a comparison of clock speeds on different CPUs is NOT a factor.

# Clock Speed and Performance




- We say a 2GHz system is twice as fast as a 1GHz system. But if an ADD instruction on a 1GHz system takes one clock cycle, whereas on a 2GHz system takes two clock cycles. Then the amount of time is same.
- This is one core reason why there's little point in comparing clock speeds across different processor architectures and families - **the amount of work done per clock cycle is different for each architecture**. (e.g. a 2GHz system has done half the work per cycle)
- Moreover when pipelining is used, multiple instructions are being executed simultaneously. Thus comparison of clock speed **!=** perform.
- So the relationship between clock speed and performance (measured in instructions per second) is different.

# Instruction Execution Rate (MIPS Rate)

- This term used for the measure of processor performance, is the rate at which instructions are executed referred to as the MIPS rate.

- It is expressed as Millions of instructions executed per second (MIPS).

- Heavily dependent on instruction set (Cisc & Risc design), compiler design, processor implementation, cache & memory hierarchy.

- We can express MIPS rate in terms of the clock rate and CPI as follows:

$$\text{MIPS rate} = \frac{I_C}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

# Cycles Per Instruction (CPI)

The processor time **T** needed to execute a given program can be expressed as:

$$T = I_c \times CPI \times \tau$$

- A processor is driven by a clock with a constant frequency **f**, or a constant cycle time **τ**, where **τ** =1/**f**.
- Define the 'instruction count' $\mathbf{I_c}$, for a program as the number of machine instructions executed for that program until it runs to completion.
- We calculate 'average cycles per instruction' (CPI) for a program.
- Let $\mathbf{CPI_i}$ be the number of cycles required for instruction type **i**.
- And $\mathbf{I_i}$ be the number of executed instructions of type **i** for a program.
- Then we can calculate an overall CPI as follows:

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

# Example (MIPS Rate)

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_c}$$

- Consider the execution of a program that results in the execution of 2 million instructions on a 400-MHz processor.
- The program consists of four major types of instructions.
- The instruction mix and CPI for each instruction type are given below:

| Instruction Type | CPI | Instruction Mix |
|---|---|---|
| Arithmetic and logic | 1 | 60% |
| Load/store with cache hit | 2 | 18% |
| Branch | 4 | 12% |
| Memory reference with cache miss | 8 | 10% |

- The average CPI when the program is executed on a uniprocessor with the above results is CPI = 0.6 + (2 X 0.18) + (4 X 0.12) + (8 X 0.1)

CPI = 2.24 ; since Ic = 100% = 1

- The corresponding MIPS rate is: $(400 \times 10^6)/(2.24 \times 10^6) \cong 178$.

# Problems (Chapter No. 2)

2.10. A benchmark program is run on a 40 MHz processor. The executed program consists of 100,000 instruction executions, with the following instruction mix and clock cycle count:

| Instruction Type | Instruction Count | Cycles per Instruction |
|---|---|---|
| Integer arithmetic | 45000 | 1 |
| Data transfer | 32000 | 2 |
| Floating point | 15000 | 2 |
| Control transfer | 8000 | 2 |

Determine the effective CPI, MIPS rate, and execution time for this program.

- Hint: First calculate the 'instruction mix'.

*CPI* = 1.55; MIPS rate = 25.8; Execution time = 3.87 ns.

# Problems (Chapter No. 2)

2.11. Consider two different machines, with two different instruction sets, both of which have a clock rate of 200 MHz. The following measurements are recorded on the two machines running a given set of benchmark programs:

| Instruction Type | Instruction Count (millions) | Cycles per Instruction |
|---|---|---|
| Machine A | | |
| Arithmetic and logic | 8 | 1 |
| Load and store | 4 | 3 |
| Branch | 2 | 4 |
| Others | 4 | 3 |
| Machine B | | |
| Arithmetic and logic | 10 | 1 |
| Load and store | 8 | 2 |
| Branch | 2 | 4 |
| Others | 4 | 3 |

a. Determine the effective CPI, MIPS rate, and execution time for each machine.

b. Comment on the results. Which machine is faster A or B?

# Preparatory Questions (Week-3)

**Q1. What are the improvements that came with IC design?**

**Q2. What are the organizational techniques for improving processor speed? Explain any four.**

**Q3. What does the term 'performance balance' mean in computers?**

**Q4. Write down the ways to achieve performance balance b/w CPU & memory.**

**Q5. Write down the ways to achieve performance balance w.r.t. I/O?**

**Q6. What are the problems associated with increased clock speed and logic density?**

**Q7. What is MIPS rate? What is its importance?**