COMPUTER ARCHITECTURE

EE-3009

OVERVIEW OF COURSE

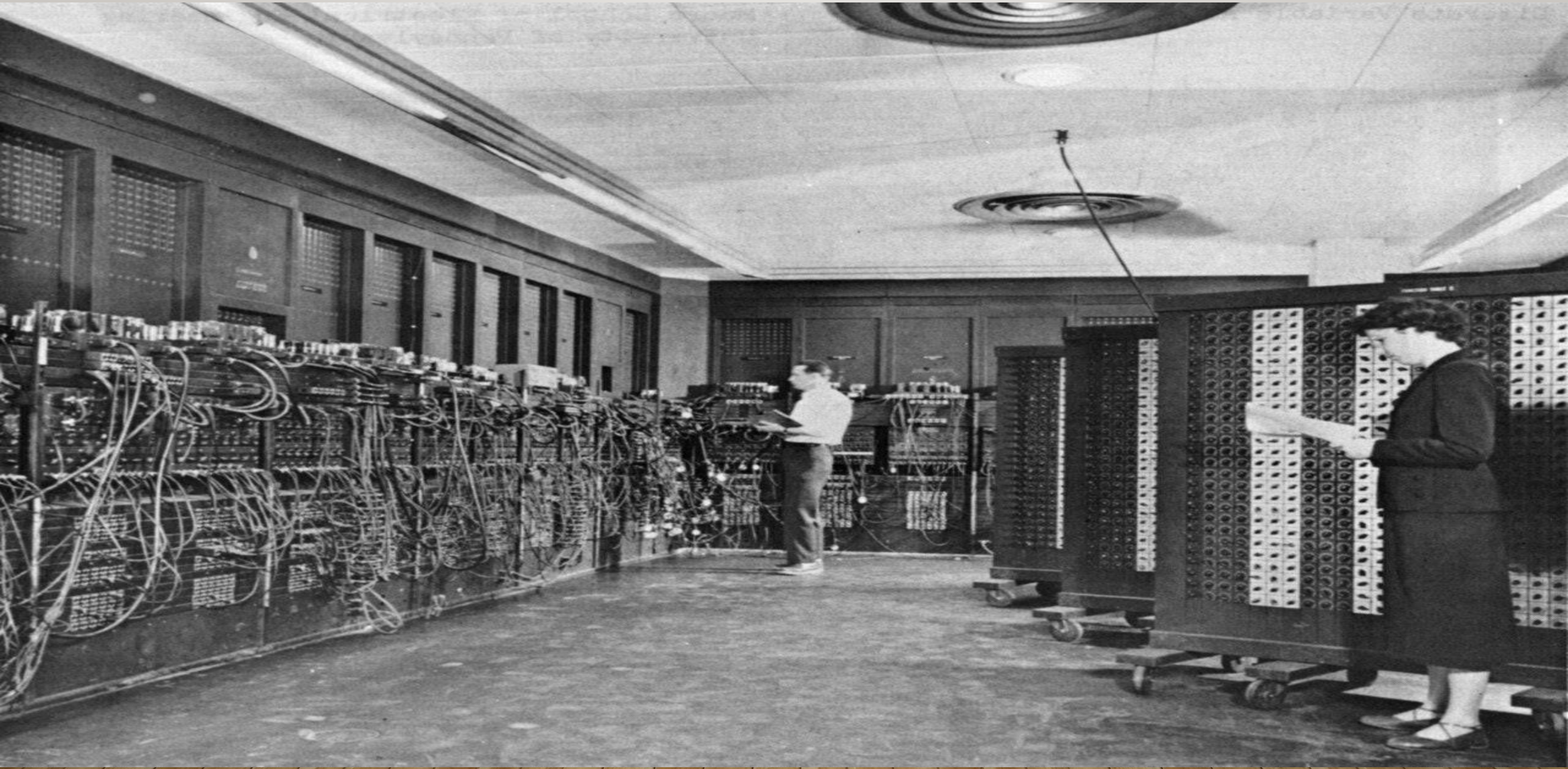
- Chapter 1 includes formulas for energy, static power, dynamic power, integrated circuit costs, reliability, and availability.
- Our hope is that these topics can be used through the rest of the course .
- In addition to the classic quantitative principles of computer design and performance measurement, it shows the slowing of performance improvement of general-purpose microprocessors, which is one inspiration for domain-specific architectures.

INTRODUCTION

- An iPod, a phone, an Internet mobile communicator… these are NOT three separate devices! And we are calling it iPhone! Today Apple is going to reinvent the phone. And here it is. (Steve Jobs, January 9, 2007)

- New information and communications technologies, in particular high-speed Internet, are changing the way companies do business, transforming public service delivery and democratizing innovation. With 10 percent increase in high speed Internet connections, economic growth increases by 1.3 percent. (The World Bank, July 28, 2009)

COMPUTERS BEFORE MICROPROCESSOR



WHAT IS A MICROPROCESSOR?

- A microprocessor is the predominant type of modern computer processor. It combines the components and function of a central processing unit (CPU) into a single integrated circuit (IC) or a few connected ICs.
- Like CPUs, microprocessors are commonly thought of as the "brain" of the computer. Unlike traditional CPUs, microprocessors integrate the arithmetic, logic and control circuits of more traditional CPUs into a single multipurpose, clock-driven and register-based digital circuit.

HISTORY OF MODERN COMPUTER

- Before microprocessors, computers used racks of ICs to accomplish the computer's main computing functions. In 1971, a significant shift occurred in computing technology with the introduction of the Intel 4004, the first commercially available microprocessor. This groundbreaking device, designed by Federico Faggin, was not just a response to a request for 12 custom microchips for a calculator from Japanese electronics maker Busicom. It was a landmark moment that revolutionized the way we think about computing, as Intel, instead of producing 12 individual chips, created a general-purpose logic device.

KEY MICROPROCESSOR STEPS

- **Fetch:** The microprocessor retrieves (or "fetches") instructions from computer memory. The fetch process can be initiated by automatic or manual input.
- **Decode:** The microprocessor "decodes" the instructions, essentially interpreting the input or command into a request and instigating a specific process or computation.
- **Execute:** Simply put, the microprocessor performs the required or requested operation.
- **Store:** The result of the execution is committed to the computer's memory.

MICROPROCESSOR COMPONENTS

- Microprocessors can complete these processes by combining the main components of a CPU into a singular circuit. The key components of a microprocessor are the following:
- **Arithmetic logic unit (ALU)**: The main logic unit of the CPU, this component performs logical operations, including mathematical calculations and data comparisons.
- **Control unit (CU):** The CU circuit interprets instructions and initiates their execution, directing the processor's basic operations.
- **Registers:** Registers provide small, fast memory storage used by a CPU to temporarily hold data and instructions during computational processes.
- **Cache memory:** Microprocessors and CPUs use cache memory, a high-speed form of memory located close to the CPU, to store frequently accessed data to accelerate performance.

MICROPROCESSOR COMPONENTS

- **Busses and bus interfaces:** Bus interfaces provide entry and exit points for data to travel across various groups of wires (referred to as busses), such as the address bus or data bus. Busses and interfaces physically connect different internal components, enabling and facilitating communication within the CPU and other peripherals like input/output (I/O) units.
- **Transistors:** One of the main building blocks of ICs, transistors are small semiconductors that regulate, amplify and generate electrical currents and signals. They can also act as simple switches or be combined to form logic gates. The number of transistors is a common indicator of microprocessor power.
- **Processor cores:** Individual processing units within microprocessors are known as cores. Modern processors frequently incorporate multiple cores (dual-core, quad-core) allowing for parallel processing by enabling the performance of multiple tasks simultaneously.
- **Clock:** Although not all microprocessors contain an internal clock, they are all clock-driven. Some rely on external clock chips, which are known for improved accuracy. Whether internal or external, a microprocessor's clock cycle determines the frequency at which it will carry out commands. Modern clock speeds are measured in megahertz (MHz) and gigahertz (GHz).

MICROPROCESSOR ARCHITECTURE

- The architecture of a microprocessor refers to various design and organization methodologies of the processor's various CPU components. These are the key architectural elements of a microprocessor:

- **Instruction Set Architecture (ISA):** The microprocessor's ISA defines the instruction set that the processor can perform. ISAs like the Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC) architectures provide various methods for data processing, offering varying levels of performance, reliability and speed suitable for different types of applications.

- **Data path:** A microprocessor's data path dictates the order in which data moves through the microprocessor's components (buses, ALU, registers), influencing overall performance.

MICROPROCESSOR ARCHITECTURE

- **Control path:** Similar to the data path, the control path element of a microprocessor's architecture instructs the sequence of operations and manages data transmission within the CPU.

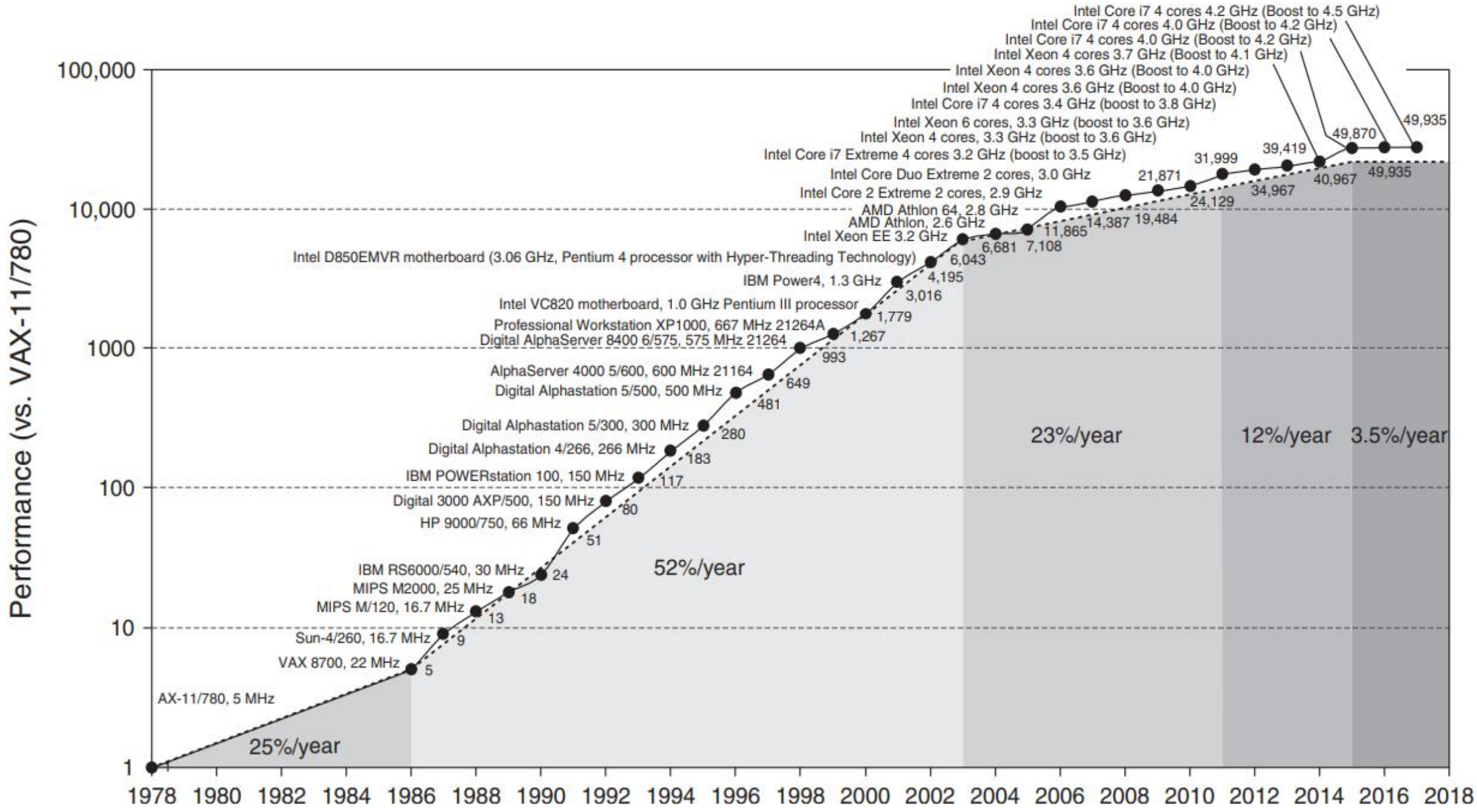
- **Memory hierarchy:** The memory hierarchy is a critical component of the processor's architecture, providing a structure for different levels of memory (cache, registers, RAM) to optimize for efficient data access and retrieval speed.

INSTRUCTION SET ARCHITECTURE (ISA)

- An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.
- The ISA provides the only way through which a user is able to interact with the hardware. It can be viewed as a programmer's manual because it's the portion of the machine that's visible to the assembly language programmer, the compiler writer, and the application programmer.

COMPUTER TECHNOLOGY

- Performance improvements:
  - Improvements in semiconductor technology
    - Feature size, clock speed
  - Improvements in computer architectures
    - Enabled by HLL compilers, UNIX
    - Lead to RISC architectures
  - Together have enabled:
    - Lightweight computers
    - Productivity-based managed/interpreted programming languages


Performance (vs. VAX-11/780)
100,000
10,000
1000
100
10
1
1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018
AX-11/780, 5 MHz
VAX 8700, 22 MHz
5
Sun-4/260, 16.7 MHz
9
MIPS M/120, 16.7 MHz
13
MIPS M2000, 25 MHz
18
IBM RS6000/540, 30 MHz
24
HP 9000/750, 66 MHz
51
Digital 3000 AXP/500, 150 MHz
80
IBM POWERstation 100, 150 MHz
117
Digital Alphastation 4/266, 266 MHz
183
Digital Alphastation 5/300, 300 MHz
280
Digital Alphastation 5/500, 500 MHz
481
AlphaServer 4000 5/600, 600 MHz 21164
649
Digital AlphaServer 8400 6/575, 575 MHz 21264
993
Professional Workstation XP1000, 667 MHz 21264A
1,267
Intel VC820 motherboard, 1.0 GHz Pentium III processor
1,779
IBM Power4, 1.3 GHz
3,016
Intel D850EMVR motherboard (3.06 GHz, Pentium 4 processor with Hyper-Threading Technology)
4,195
Intel Xeon EE 3.2 GHz
6,043
AMD Athlon, 2.6 GHz
6,681
AMD Athlon 64, 2.8 GHz
7,108
Intel Core 2 Extreme 2 cores, 2.9 GHz
11,865
Intel Core Duo Extreme 2 cores, 3.0 GHz
14,387
Intel Core i7 Extreme 4 cores 3.2 GHz (boost to 3.5 GHz)
19,484
Intel Xeon 4 cores, 3.3 GHz (boost to 3.6 GHz)
21,871
Intel Xeon 6 cores, 3.3 GHz (boost to 3.6 GHz)
24,129
Intel Core i7 4 cores 3.4 GHz (boost to 3.8 GHz)
31,999
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
34,967
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
39,419
Intel Xeon 4 cores 3.7 GHz (Boost to 4.1 GHz)
40,967
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
49,870
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
49,935
Intel Core i7 4 cores 4.2 GHz (Boost to 4.5 GHz)
49,935
25%/year
52%/year
23%/year
12%/year
3.5%/year

CLASSES OF COMPUTERS

- Not since the creation of the personal computer have we seen such striking changes in the way computers appear and in how they are used. These changes in computer use have led to five diverse computing markets, each characterized by different applications, requirements, and computing technologies.

Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Internet of things/ embedded

PERSONAL MOBILE DEVICE (PMD)

- Personal mobile device (PMD) is the term we apply to a collection of wireless devices with multimedia user interfaces such as cell phones, tablet computers, and so on. Cost is a prime concern given the consumer price for the whole product is a few hundred dollars. Although the emphasis on energy efficiency is frequently driven by the use of batteries, the need to use less expensive packaging—plastic versus ceramic—and the absence of a fan for cooling also limit total power consumption.
- Emphasis on energy efficiency and real-time

DESKTOP COMPUTING

- The first, and possibly still the largest market in dollar terms, is desktop computing. Desktop computing spans from low-end netbooks that sell for under $300 to high-end, heavily configured workstations that may sell for $2500.
- Throughout this range in price and capability, the desktop market tends to be driven to optimize price-performance. This combination of performance (measured primarily in terms of compute performance and graphics performance) and price of a system is what matters most to customers in this market, and hence to computer designers.

SERVERS

- **Servers** are specialized computers designed to provide services, resources, or data to other computers, devices, or clients on a network. They play a central role in modern IT infrastructure by supporting applications, hosting websites, storing data, and facilitating communication between devices.
- For servers, different characteristics are important. First, availability is critical.
- A second key feature of server systems is scalability.
- Finally, servers are designed for efficient throughput.

CLUSTERS/WAREHOUSE-SCALE COMPUTERS

- The growth of Software as a Service (SaaS) for applications like search, social networking, video viewing and sharing, multiplayer games, online shopping, and so on has led to the growth of a class of computers called clusters.
- Clusters are collections of desktop computers or servers connected by local area networks to act as a single larger computer. Each node runs its own operating system, and nodes communicate using a networking protocol.
- Fault Tolerance and Load Balancing .

INTERNET OF THINGS (IOT)

- refers to embedded computers that are connected to the Internet, typically wirelessly. When augmented with sensors and actuators, IoT devices collect useful data and interact with the physical world, leading to a wide variety of "smart" applications, such as smart watches, smart thermostats, smart speakers, smart cars, smart homes, smart grids, and smart cities.
- Embedded computers have the widest spread of processing power and cost. They include 8-bit to 32-bit processors that may cost one penny, and high-end 64-bit processors for cars and network switches that cost $100. Although the range of computing power in the embedded computing market is very large, price is a key factor in the design of computers for this space.

PARALLELISM

- Parallelism at multiple levels is now the driving force of computer design across all four classes of computers, with energy and cost being the primary constraints.

- In computer architecture, **parallelism** refers to the ability to perform multiple operations or tasks simultaneously, improving the overall speed and efficiency of processing. There are several types of parallelism, each at different levels of a computer system

CLASSES OF PARALLELISM

- **Data-Level Parallelism (DLP)** refers to the parallel execution of the same operation on multiple pieces of data simultaneously. In other words, it involves performing the same task on multiple data elements at once, rather than on just a single element at a time.
- This kind of parallelism is especially useful when you're working with large datasets, where the same operation needs to be applied to many pieces of data. Instead of processing one element after another, DLP enables the simultaneous processing of multiple elements, which significantly speeds up the computation.

EXAMPLES OF DLP

- Vector Addition
- Grayscale Conversion of an Image (image Processing)

CLASSES OF PARALLELISM

- **Task-Level Parallelism (TLP)** refers to the parallel execution of independent tasks or processes, where each task performs a different function or part of a program. Unlike **Data-Level Parallelism (DLP)**, where the same operation is applied to many pieces of data, task-level parallelism involves the concurrent execution of different tasks, which may not necessarily perform the same operation but contribute to completing the overall work.
- In task-level parallelism, each task is often treated as a separate unit of execution, and these tasks can run simultaneously on multiple processing units (e.g., cores or CPUs). This kind of parallelism is commonly used in multi-core or multi-processor systems, where each processor or core can handle different tasks independently.

EXAMPLE OF TASK-LEVEL PARALLELISM:

- Imagine a program that performs several steps to process data:
- **Task 1**: Read data from a file.
- **Task 2**: Clean the data (e.g., removing duplicates, filling missing values).
- **Task 3**: Perform calculations or transformations on the data.
- **Task 4**: Generate results and save them to a file.

TASK-LEVEL PARALLELISM VS. DATA-LEVEL PARALLELISM:

- **Data-Level Parallelism (DLP):** Involves performing the same operation on multiple data elements at the same time (e.g., vector processing or matrix operations).
- **Task-Level Parallelism (TLP):** Involves executing different tasks or functions simultaneously, often involving different types of operations.

CLASSES OF ARCHITECTURAL PARALLELISM:

- Instruction-Level Parallelism (ILP)
- Vector architectures/Graphic Processor Units (GPUs)
- Thread-Level Parallelism
- Request-Level Parallelism

FLYNN'S TAXONOMY

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
  - Vector architectures
  - Multimedia extensions
  - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)

DEFINING COMPUTER ARCHITECTURE

- "Old" view of computer architecture:
- Instruction Set Architecture (ISA) design
- i.e. decisions regarding:
- registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- Other aspects of computer design were called implementation, often insinuating that implementation is uninteresting or less challenging.

GENUINE COMPUTER ARCHITECTURE: DESIGNING THE ORGANIZATION AND HARDWARE TO MEET GOALS AND FUNCTIONAL REQUIREMENTS

- The implementation of a computer has two components: organization and hardware. The term organization includes the high-level aspects of a computer's design, such as the memory system, the memory interconnect, and the design of the internal processor or CPU (central processing unit—where arithmetic, logic, branching, and data transfer are implemented).
- The term microarchitecture is also used instead of organization. For example, two processors with the same instruction set architectures but different organizations are the AMD Opteron and the Intel Core i7. Both processors implement the 80 x86 instruction set, but they have very different pipeline and cache organizations.

GENUINE COMPUTER ARCHITECTURE

- The switch to multiple processors per microprocessor led to the term core also being used for processors. Instead of saying multiprocessor microprocessor, the term multicore caught on. Given that virtually all chips have multiple processors, the term central processing unit, or CPU, is fading in popularity.
- Hardware refers to the specifics of a computer, including the detailed logic design and the packaging technology of the computer. Often a line of computers contains computers with identical instruction set architectures and very similar organizations, but they differ in the detailed hardware implementation. For example, the Intel Core i7 and the Intel Xeon E7 are nearly identical but offer different clock rates and different memory systems, making the Xeon E7 more effective for server computers.

GENUINE COMPUTER ARCHITECTURE

- In this Course , the word **architecture** covers all three aspects of computer design—instruction set architecture, organization or microarchitecture, and hardware.

Functional requirements	Typical features required or supported
Application area	*Target of computer*
Personal mobile device	Real-time performance for a range of tasks, including interactive performance for graphics, video, and audio; energy efficiency (Chapters 2–5 and 7; Appendix A)
General-purpose desktop	Balanced performance for a range of tasks, including interactive performance for graphics, video, and audio (Chapters 2–5; Appendix A)
Servers	Support for databases and transaction processing; enhancements for reliability and availability; support for scalability (Chapters 2, 5, and 7; Appendices A, D, and F)
Clusters/warehouse-scale computers	Throughput performance for many independent tasks; error correction for memory; energy proportionality (Chapters 2, 6, and 7; Appendix F)
Internet of things/embedded computing	Often requires special support for graphics or video (or other application-specific extension); power limitations and power control may be required; real-time constraints (Chapters 2, 3, 5, and 7; Appendices A and E)

DEFINING COMPUTER ARCHITECTURE

- "Real" computer architecture:
- Specific requirements of the target machine
- Design to maximize performance within constraints:
- cost, power, and availability
- Includes ISA, microarchitecture, hardware

TRENDS IN TECHNOLOGY

- If an instruction set architecture is to prevail, it must be designed to survive rapid changes in computer technology.
- An architect must plan for technology changes that can increase the lifetime of a successful computer.
- To plan for the evolution of a computer, the designer must be aware of rapid changes in implementation technology. Five implementation technologies, which change at a dramatic pace, are critical to modern implementations:

INTEGRATED CIRCUIT LOGIC TECHNOLOGY

- Historically, transistor density increased by about 35% per year, quadrupling somewhat over four years.
- Increases in die size are less predictable and slower, ranging from 10% to 20% per year.
- The combined effect was a traditional growth rate in transistor count on a chip of about 40%–55% per year, or doubling every 18–24 months.
- This trend is popularly known as **Moore's Law**.

SEMICONDUCTOR **DRAM** (DYNAMIC RANDOM-ACCESS MEMORY)

- This technology is the foundation of main memory.
- The growth of DRAM has slowed dramatically, from quadrupling every three years as in the past.
- The 8-gigabit DRAM was shipping in 2014, but the 16-gigabit DRAM won't reach that state until 2019.

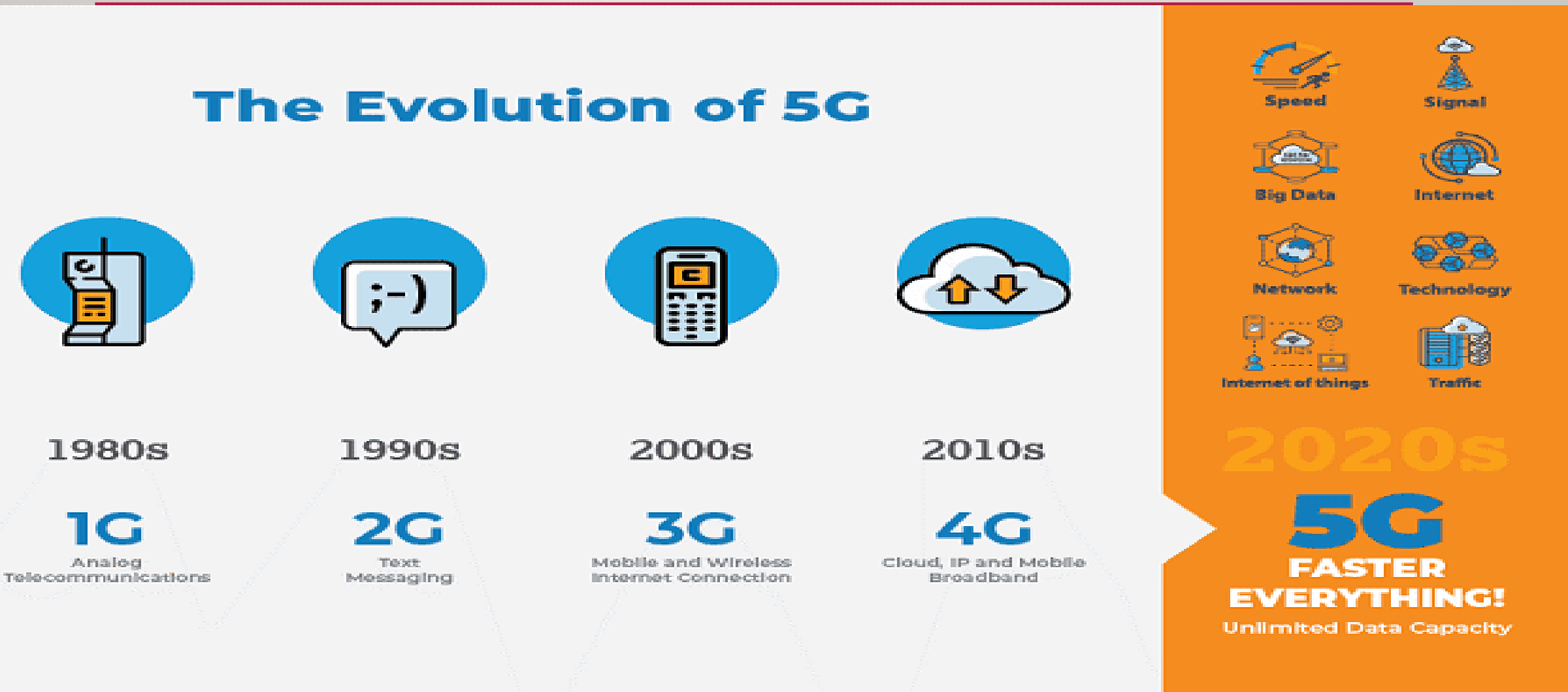
SEMICONDUCTOR FLASH

- electrically erasable programmable read-only memory.
- This nonvolatile semiconductor memory is the standard storage device in PMDs, and its rapidly increasing popularity has fueled its rapid growth rate in capacity.
- In recent years, the capacity per Flash chip increased by about 50%–60% per year, doubling roughly every 2 years.
- Currently, Flash memory is 8–10 times cheaper per bit than DRAM.

MAGNETIC DISK TECHNOLOGY

- Prior to 1990, density increased by about 30% per year, doubling in three years.
- It rose to 60% per year thereafter, and increased to 100% per year in 1996.
- Between 2004 and 2011, it dropped back to about 40% per year, or doubled every two years.
- Recently, disk improvement has slowed to less than 5% per year.
- This technology is central to server- and warehouse-scale storage.

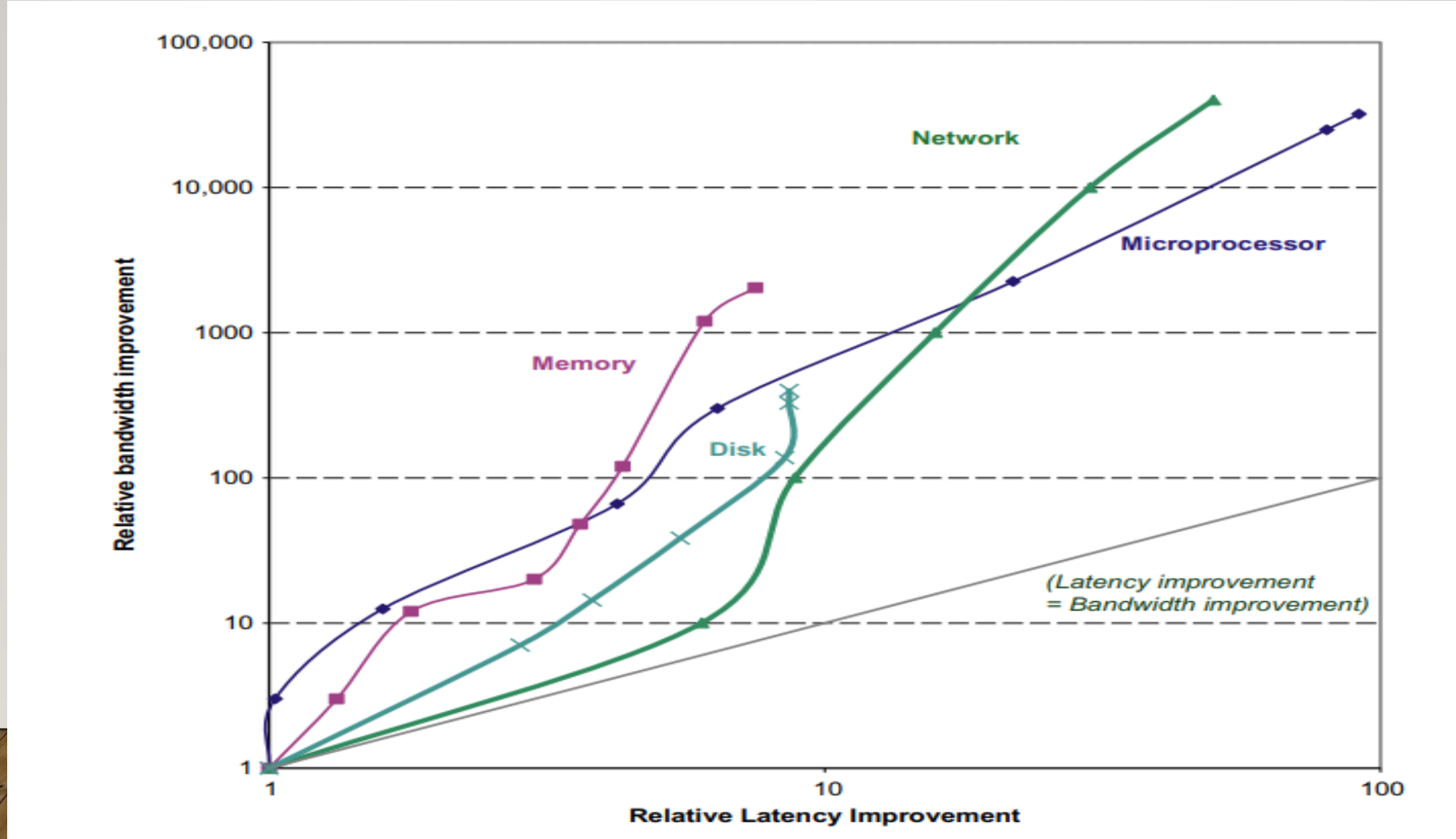
NETWORK TECHNOLOGY



PERFORMANCE TRENDS: BANDWIDTH OVER LATENCY

- Bandwidth or throughput is the total amount of work done in a given time, such as megabytes per second for a disk transfer.
- In contrast, latency or response time is the time between the start and the completion of an event, such as milliseconds for a disk access.
- A simple rule of thumb is that bandwidth grows by at least the square of the improvement in latency. Computer designers should plan accordingly.

PERFORMANCE TRENDS: BANDWIDTH OVER LATENCY



POWER AND ENERGY: A SYSTEMS PERSPECTIVE

- From the viewpoint of a system designer, there are three primary concerns.
- First, what is the maximum power a processor ever requires? Meeting this demand can be important to ensuring correct operation. For example, if a processor attempts to draw more power than a power-supply system can provide (by drawing more current than the system can supply), the result is typically a voltage drop, which can cause devices to malfunction. Modern processors can vary widely in power consumption with high peak currents; hence they provide voltage indexing methods that allow the processor to slow down and regulate voltage within a wider margin. Obviously, doing so decreases performance.

POWER AND ENERGY: A SYSTEMS PERSPECTIVE

- Second, what is the sustained power consumption? This metric is widely called the thermal design power (TDP) because it determines the cooling requirement. TDP is neither peak power, which is often 1.5 times higher, nor is it the actual average power that will be consumed during a given computation, which is likely to be lower still.
- A typical power supply for a system is typically sized to exceed the TDP, and a cooling system is usually designed to match or exceed TDP. Failure to provide adequate cooling will allow the junction temperature in the processor to exceed its maximum value, resulting in device failure and possibly permanent damage. Modern processors provide two features to assist in managing heat, since the highest power (and hence heat and temperature rise) can exceed the long-term average specified by the TDP. First, as the thermal temperature approaches the junction temperature limit, circuitry lowers the clock rate, thereby reducing power. Should this technique not be successful, a second thermal overload trap is activated to power down the chip.

POWER AND ENERGY: A SYSTEMS PERSPECTIVE

- The third factor that designers and users need to consider is energy and energy efficiency. Recall that power is simply energy per unit time: 1 watt =1 joule per second. Which metric is the right one for comparing processors: energy or power? In general, energy is always a better metric because it is tied to a specific task and the time required for that task.
- Whenever we have a fixed workload, whether for a warehouse-size cloud or a smartphone, comparing energy will be the right way to compare computer alternatives, because the electricity bill for the cloud and the battery lifetime for the smartphone are both determined by the energy consumed.

ENERGY AND POWER WITHIN A MICROPROCESSOR

- For CMOS chips, the traditional primary energy consumption has been in switching transistors, also called dynamic energy. The energy required per transistor is proportional to the product of the capacitive load driven by the transistor and the square of the voltage:

$$\text{Energy}_{\text{dynamic}} \propto \text{Capacitive load} \times \text{Voltage}^2$$

- This equation is the energy of pulse of the logic transition of 0->1->0 or 1->0->1. The energy of a single transition (0->1 or 1->0) is then

$$\text{Energy}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2$$

ENERGY AND POWER WITHIN A MICROPROCESSOR

- The power required per transistor is just the product of the energy of a transition multiplied by the frequency of transitions:

$$\text{Power}_{\text{dynamic}} \propto 1/2 \times \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency switched}$$

- For a fixed task, slowing clock rate reduces power, but not energy.
- Clearly, dynamic power and energy are greatly reduced by lowering the voltage, so voltages have dropped from 5 V to just under 1 V in 20 years. The capacitive load is a function of the number of transistors connected to an output and the technology, which determines the capacitance of the wires and the transistors.

EXAMPLE

- Some microprocessors today are designed to have adjustable voltage, so a 15% reduction in voltage may result in a 15% reduction in frequency. What would be the impact on dynamic energy and on dynamic power?

$$\frac{\text{Energy}_{\text{new}}}{\text{Energy}_{\text{old}}} = \frac{(\text{Voltage} \times 0.85)^2}{\text{Voltage}^2} = 0.85^2 = 0.72$$

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = 0.72 \times \frac{(\text{Frequency switched} \times 0.85)}{\text{Frequency switched}} = 0.61$$

POWER

- Power consumption increases as processor complexity increases
- Number of transistors increases
- Switching frequency increases
- Early microprocessors consumed about 1W
- 80386 microprocessors consumed about 2W
- 3.3GHz Intel Core i7 consumes about 130W
- Must be dissipated from a chip that is about 1.5cm × 1.5cm

MANAGING POWER FOR FURTHER EXPANSION

- Voltage cannot be reduced further .
- Power per chip cannot be increased because the air cooling limit has been reached.
- Therefore, clock frequency growth has slowed .
- Heat dissipation is now the major constraint on using transistors.

ENERGY EFFICIENCY STRATEGIES

- Modern microprocessors offer many techniques to try to improve energy efficiency despite flat clock rates and constant supply voltages:
- 1. ***Do nothing well***. Most microprocessors today turn off the clock of inactive modules to save energy and dynamic power. For example, if no floating-point instructions are executing, the clock of the floating-point unit is disabled. If some cores are idle, their clocks are stopped.

ENERGY EFFICIENCY STRATEGIES

- 2. ***Dynamic voltage and frequency scaling*** (DVFS) is the adjustment of power and speed settings on a computing device's various processors, controller chips and peripheral devices to optimize resource allotment for tasks and maximize power saving when those resources are not needed.
- If the current task is computationally intensive, meaning it requires the processor to consume more energy and run at a higher frequency, DVFS will ensure that the voltage and frequency increase to match the requirements. However, if the processor needs to run a low-priority or less computationally intensive task where high speed and throughput are not required, it will adjust accordingly.

ENERGY EFFICIENCY STRATEGIES

- ***Design for the typical case.***

- PMDs and laptops are often idle
- Use low power mode DRAM to save energy
- Spin disk at lower rate
- PCs use emergency slowdown if program execution causes overheating

ENERGY EFFICIENCY STRATEGIES

- ***Overclocking.***
- Intel started offering Turbo mode in 2008, where the chip decides that it is safe to run at a higher clock rate for a short time, possibly on just a few cores, until temperature starts to rise. For example, the 3.3 GHz Core i7 can run in short bursts for 3.6 GHz. Indeed, the highest-performing microprocessors each year since 2008 have all offered temporary overclocking of about 10% over the nominal clock rate.
- For single-threaded code, these microprocessors can turn off all cores but one and run it faster. Note that, although the operating system can turn off Turbo mode, there is no notification once it is enabled, so the programmers may be surprised to see their programs vary in performance because of room temperature.

ENERGY EFFICIENCY STRATEGIES

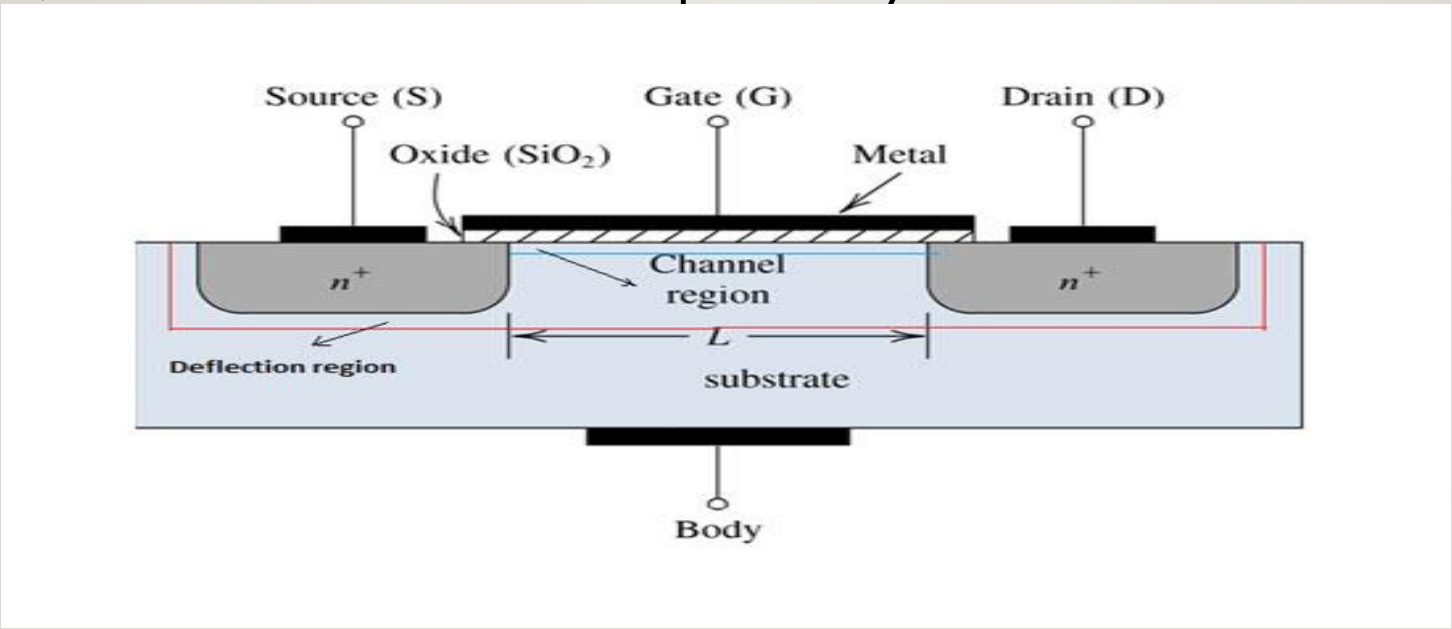
- ***Power gating***
- Powerstatic ∝ Currentstatic × Voltage
- Current flows in transistors even when idle: leakage current
- Leakage ranges from 25% to 50% of total power
- Power Gating turns off power to inactive modules

ENERGY EFFICIENCY STRATEGIES

- ***Race-to-halt***
- Processor is only part of system cost
- Use faster, less energy-efficient processor to allow the rest of the system to halt

DEPENDABILITY

- Historically, integrated circuits were one of the most reliable components of a computer. Although their pins may be vulnerable, and faults may occur over communication channels, the failure rate inside the chip was very low.



DEPENDABILITY

- Infrastructure providers started offering service level agreements (SLAs) or service level objectives (SLOs) to guarantee that their networking or power service would be dependable. For example, they would pay the customer a penalty if they did not meet an agreement of some hours per month. Thus an SLA could be used to decide whether the system was up or down.
- Systems alternate between two states of service with respect to an SLA:
- 1. Service accomplishment, where the service is delivered as specified.
- 2. Service interruption, where the delivered service is different from the SLA.

DEPENDABILITY

- Transitions between these two states are caused by failures (from state 1 to state 2) or restorations (2 to 1). Quantifying these transitions leads to the two main measures of dependability:
- Module reliability is a measure of the continuous service accomplishment (or, equivalently, of the time to failure) from a reference initial instant. Therefore the mean time to failure (MTTF) is a reliability measure.

DEPENDABILITY

- The reciprocal of MTTF is a rate of failures, generally reported as failures per billion hours of operation, or FIT (for failures in time). Thus an MTTF of 1,000,000 hours equals 10^9 /10^6 or 1000 FIT.
- Service interruption is measured as mean time to repair (MTTR).
- Mean time between failures (MTBF) is simply the sum of MTTF +MTTR.

$$\text{Module availability} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})}$$

EXAMPLE

- Assume a disk subsystem with the following components and MTTF:
- 10 disks, each rated at 1,000,000-hour MTTF
- 1 ATA controller, 500,000-hour MTTF
- 1 power supply, 200,000-hour MTTF
- 1 fan, 200,000-hour MTTF
- 1 ATA cable, 1,000,000-hour MTTF

SOLUTION

Using the simplifying assumptions that the lifetimes are exponentially distributed and that failures are independent, compute the MTTF of the system as a whole.

The sum of the failure rates is

$$\text{Failure rate}_{\text{system}} = 10 \times \frac{1}{1,000,000} + \frac{1}{500,000} + \frac{1}{200,000} + \frac{1}{200,000} + \frac{1}{1,000,000}$$

$$= \frac{10+2+5+5+1}{1,000,000 \text{ hours}} = \frac{23}{1,000,000} = \frac{23,000}{1,000,000,000 \text{ hours}}$$

or 23,000 FIT. The MTTF for the system is just the inverse of the failure rate

$$\text{MTTF}_{\text{system}} = \frac{1}{\text{Failure rate}_{\text{system}}} = \frac{1,000,000,000 \text{ hours}}{23,000} = 43,500 \text{ hours}$$

or just under 5 years.

EXERCISE

- You are working as an IT manager for a large data center that uses 500 hundred hard drives in its servers. Each hard drive has an **MTBF rating** of 1,000,000 hours (this is often provided by the manufacturer). You want to estimate the overall MTBF for the data center and predict the reliability of the system over time.

DEPENDABILITY

- Mean time to failure (MTFF) and MTBF are both important metrics that measure time and evaluate the performance of an asset. The difference lies in the type of asset, as MTBF applies to assets you can repair. In contrast, MTTF applies to assets that reach the maximum operational hours and are no longer repairable or functional. It calculates the entire life span of assets that aren't repairable.

MEASURING, REPORTING, AND SUMMARIZING PERFORMANCE

- When we say one computer is faster than another one is, what do we mean?
- The user of a cell phone may say a computer is faster when a program runs in less time, while an Amazon.com administrator may say a computer is faster when it completes more transactions per hour.
- The cell phone user wants to reduce response time—the time between the start and the completion of an event—also referred to as execution time. The operator of a WSC wants to increase throughput—the total amount of work done in a given time.

MEASURING, REPORTING, AND SUMMARIZING PERFORMANCE

- In comparing design alternatives, we often want to relate the performance of two different computers, say, X and Y. The phrase "X is faster than Y" is used here to mean that the response time or execution time is lower on X than on Y for the given task. In particular, "X is n times as fast as Y" will mean

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

Since execution time is the reciprocal of performance, the following relationship holds:

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

MEASURING, REPORTING, AND SUMMARIZING PERFORMANCE

- Execution time can be defined in different ways depending on what we count.
- The most straightforward definition of time is called ***wall-clock time***, ***response time***, or ***elapsed time***, which is the latency to complete a task, including storage accesses, memory accesses, input/output activities, operating system overhead—everything.
- CPU time recognizes this distinction and means the time the processor is computing, not including the time waiting for I/O or running other programs. (Clearly, the response time seen by the user is the elapsed time of the program, not the CPU time.)

BENCHMARKS

- Kernels (e.g. matrix multiply)
- Toy programs (e.g. sorting)
- Synthetic benchmarks (e.g. Dhrystone)
- Benchmark suites (e.g. SPEC06fp, TPC-C)

AMDAHL'S LAW

- The performance gain that can be obtained by improving some portion of a computer can be calculated using Amdahl's Law. Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.
- Amdahl's Law defines the speedup that can be gained by using a particular feature. What is speedup? Suppose that we can make an enhancement to a computer that will improve performance when it is used. Speedup is the ratio

AMDAHL'S LAW

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

AMDAHL'S LAW

- Speedup tells us how much faster a task will run using the computer with the enhancement contrary to the original computer. Amdahl's Law gives us a quick way to find the speedup from some enhancement, which depends on two factors:
- The fraction of the computation time in the original computer that can be converted to take advantage of the enhancement—For example, if 40 seconds of the execution time of a program that takes 100 seconds in total can use an enhancement, the fraction is 40/100. This value, which we call Fractionenhanced, is always less than or equal to 1.

AMDAHL'S LAW

- The improvement gained by the enhanced execution mode, that is, how much faster the task would run if the enhanced mode were used for the entire program—This value is the time of the original mode over the time of the enhanced mode. If the enhanced mode takes, say, 4 seconds for a portion of the program, while it is 40 seconds in the original mode, the improvement is 40/4 or 10. We call this value, which is always greater than 1, Speedupenhanced

EXAMPLE

- Suppose that we want to enhance the processor used for web serving. The new processor is 10 times faster on computation in the web serving application than the old processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

$$\text{Fraction}_{\text{enhanced}} = 0.4;\ \text{Speedup}_{\text{enhanced}} = 10;\ \text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

EXERCISE

- A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FSQRT) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FSQRT hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

SOLUTION

$$\text{Speedup}_{\text{FSQRT}} = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

THE PROCESSOR PERFORMANCE EQUATION

- Essentially all computers are constructed using a clock running at a constant rate. These discrete time events are called clock periods, clocks, cycles, or clock cycles.
- Computer designers refer to the time of a clock period by its duration (e.g., 1 ns) or by its rate (e.g., 1 GHz). CPU time for a program can then be expressed two ways:

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

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

THE PROCESSOR PERFORMANCE EQUATION

- By transposing the instruction count in the preceding formula, clock cycles can be defined as IC,CPI. This allows us to use CPI in the execution time formula:

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

- As this formula demonstrates, processor performance is dependent upon three characteristics: clock cycle (or rate), clock cycles per instruction, and instruction count. Furthermore, CPU time is equally dependent on these three characteristics; for example, a 10% improvement in any one of them leads to a 10% improvement in CPU time.

THE PROCESSOR PERFORMANCE EQUATION

- Unfortunately, it is difficult to change one parameter in complete isolation from others because the basic technologies involved in changing each characteristic are interdependent:
- ■ Clock cycle time—Hardware technology and organization
- ■ CPI—Organization and instruction set architecture
- ■ Instruction count—Instruction set architecture and compiler technology

THE PROCESSOR PERFORMANCE EQUATION

- In designing the processor, sometimes it is useful to calculate the number of total processor clock cycles as:

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

- where ICi represents the number of times instruction i is executed in a program and CPIi represents the average number of clocks per instruction for instruction i.

THE PROCESSOR PERFORMANCE EQUATION

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

$$\text{CPI} = \frac{\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i}{\text{Instruction count}} = \sum_{i=1}^{n} \frac{\text{IC}_i}{\text{Instruction count}} \times \text{CPI}_i$$

EXERCISE

- A 2.3GHz processor was used to execute a benchmark program with the following instruction mix and clock cycle count:

Instruction Type	Instruction Count	Clock cycle count (CPI)
Integer Arithmetic	42000	1
Data transfer	30000	2
Floating point	15000	3
Control transfer	8000	4

- Determine, Effective CPI , MIPS rate and Execution time for the program.

EXERCISE

- Your company has just bought a new Intel Core i5 dual core processor, and you have been tasked with optimizing your software for this processor. You will run two applications on this dual core, but the resource requirements are not equal. The first application requires 80% of the resources, and the other only 20% of the resources. Assume that when you parallelize a portion of the program, the speedup for that portion is 2.
- Given that 40% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation?

EXERCISE

- Given that 99% of the second application is parallelizable, how much speedup would this application observe if run in isolation?
- Given that 40% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it?
- Given that 99% of the second application is parallelizable, how much overall system speedup would you observe if you parallelized it?