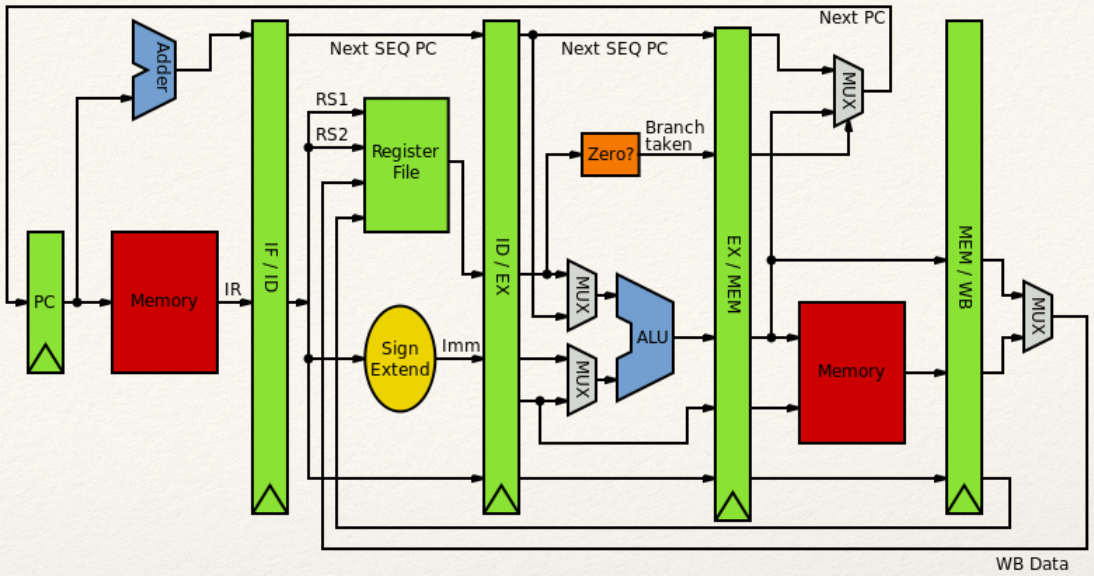


COMP2120 Computer Organization

Introduction

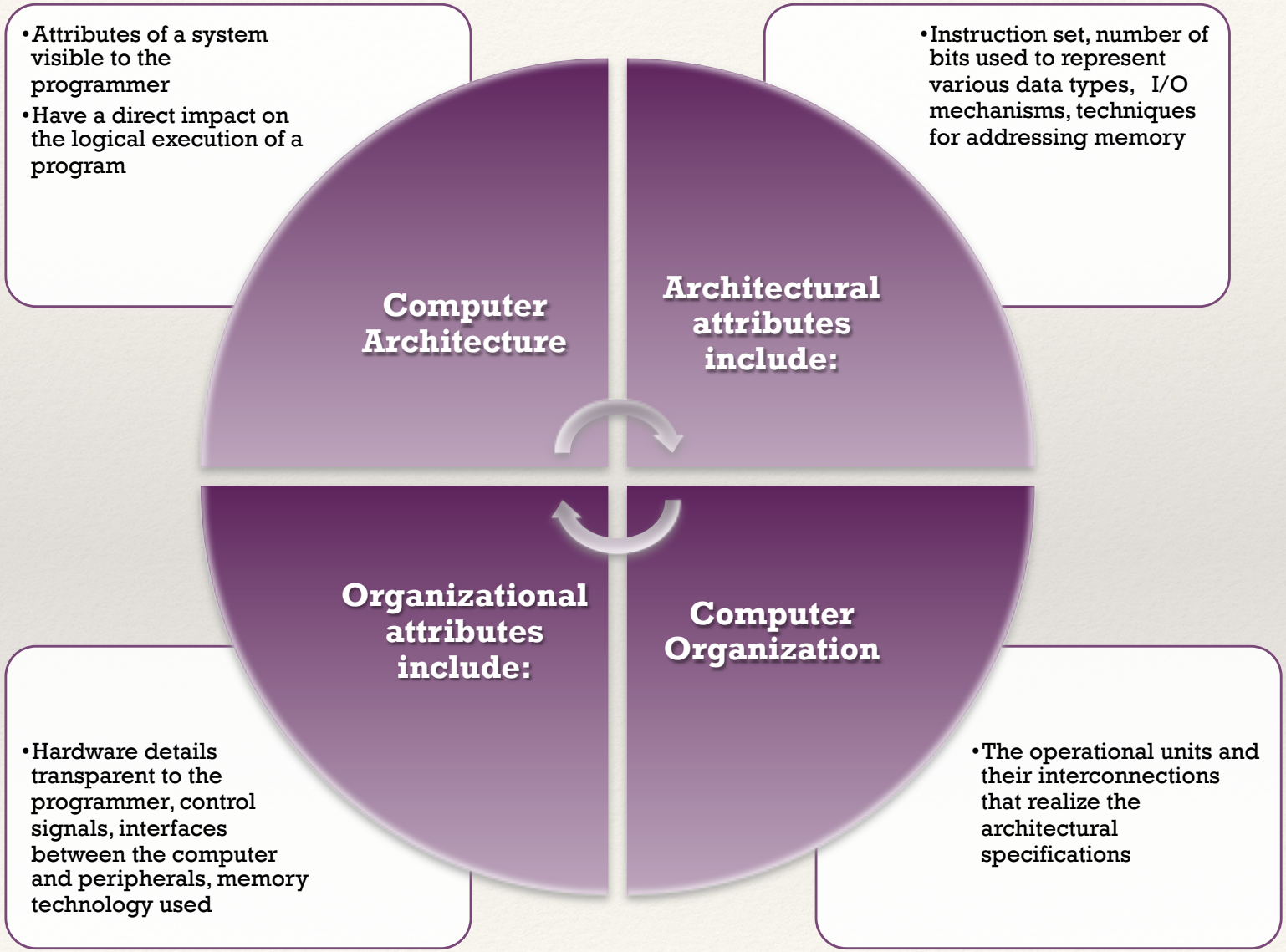
Dr. Ronald H.Y. Chung

THE UNIVERSITY OF HONG KONG

DEPARTMENT OF
COMPUTER SCIENCE

Chapter 1 - Basic Concepts and Computer Evolution

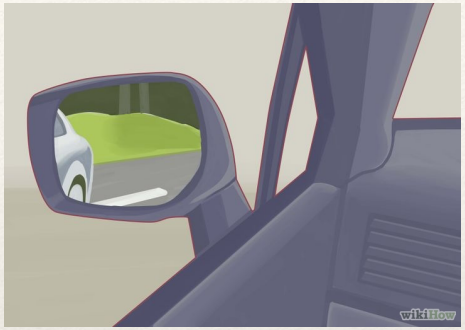
Computer Architecture & Computer Organization

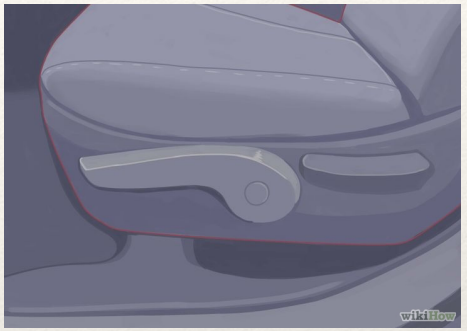


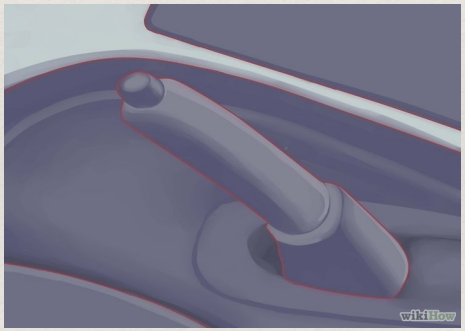
Analogy with Vehicle

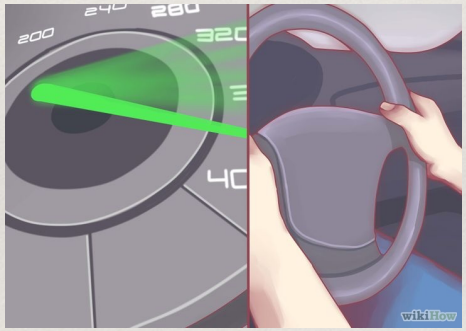
Source: http://www.wikihow.com/Drive-a-Car

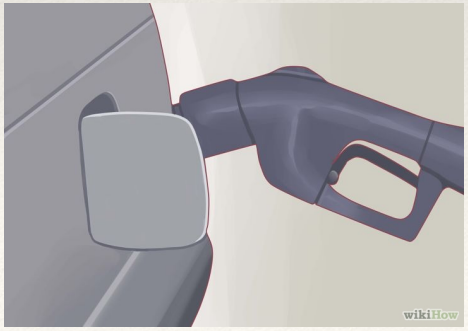
- Computer Architecture is similar to the architectural aspects of driving a Vehicle
  - How to drive a car
  - Interface visible to drivers

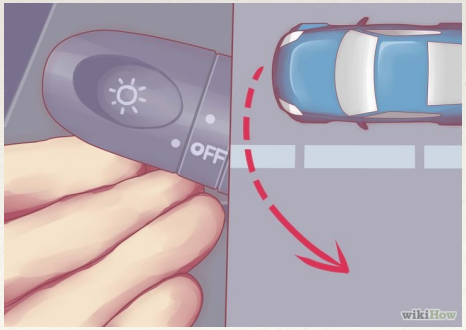


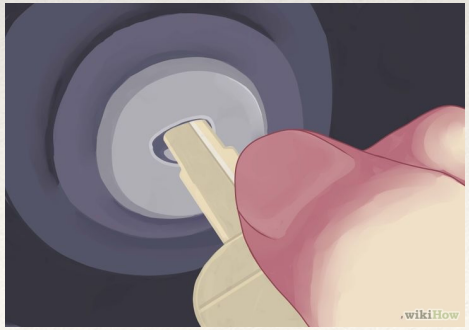


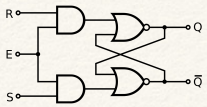








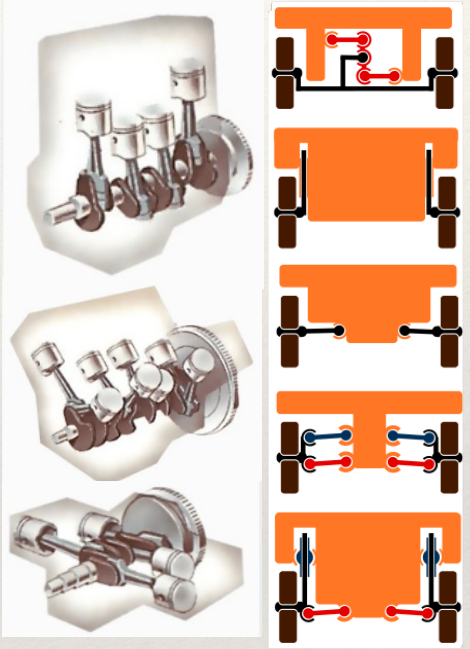


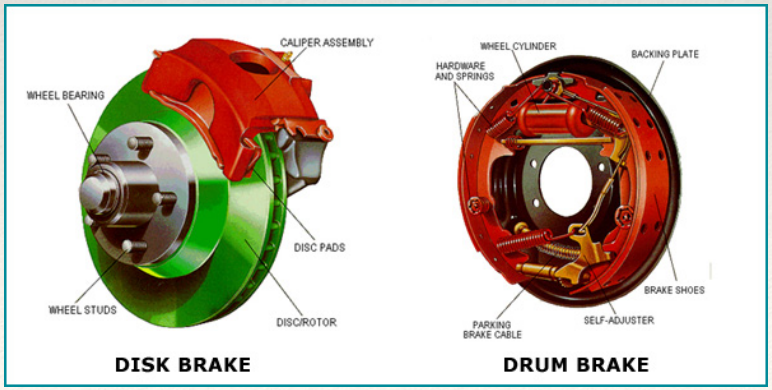


Analogy with Vehicle (Cont'd)

- Computer Organization is similar to the realization of the Vehicles
  - Engine
    - Diesel, Gasoline, Electrical?
    - In-line, V-8, Horizontally-opposed?
  - Suspension System
    - Passive, Semi-active, Active?
  - Braking System
    - Frictional, Electromagnetic?
    - Disk, Drum?

Sources:
https://en.wikipedia.org/wiki/Suspension_(vehicle)
http://www.howacarworks.com/basics/the-engine
http://motorist.org/articles/auto-braking-systems

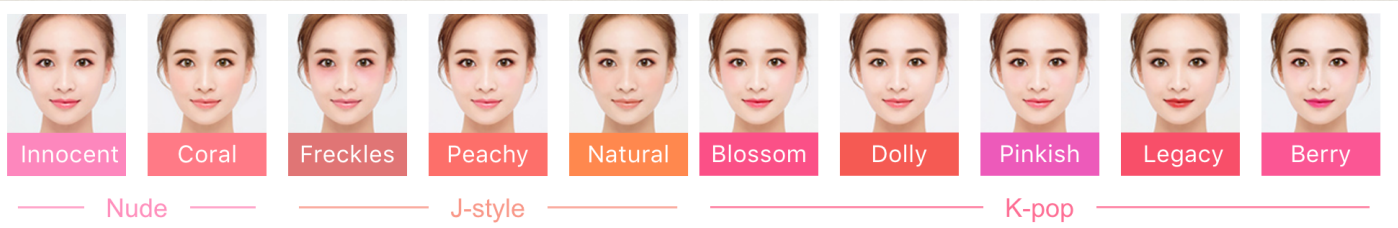




Analogy with Makeup Tools

- Computer Architecture is similar to the architectural aspects doing makeup
  - Tools available for different makeup effect

Source: http://www.befrassy.net/2015/01/whats-in-my-makeup-bag.html



Analogy with Makeup Tools (Cont'd)

- Computer Architecture is similar to the realisation of the makeup tools
  - Types of lipsticks
    - Matte Lipstick
    - Sheer Lipstick
    - Lip Crayon
    - Lip Oil
    - Lip Gloss
    - Cream Lipstick

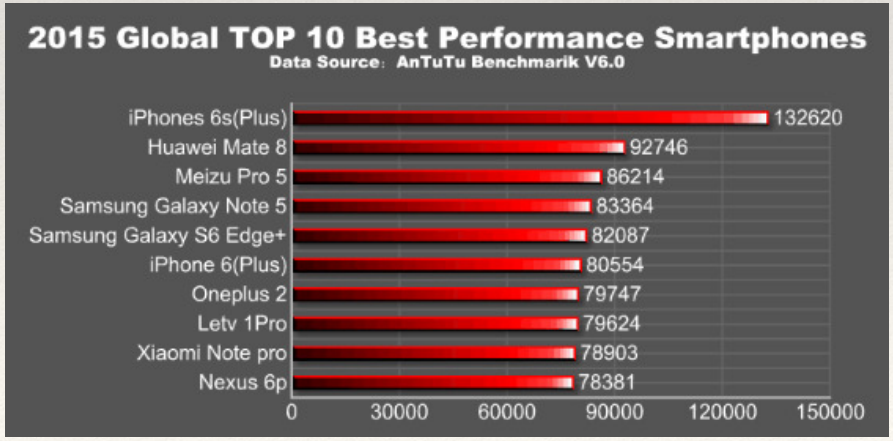
- Types of Foundations
  - Cream
  - Powder
  - Tinted
  - Mousse
  - Mineral

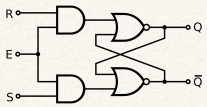
Importance of Computer Organization

- Example
  - "iPhone 6s performance thrashes high-spec Androids Huawei Mate 8, Samsung Note 5 & Nexus 6p", by Antutu
    - Apple's custom A9 chip comes with dual cores and 2GB RAM only
    - 64-bit vs 32-bit CPU
  - iOS also continues to be more efficient than Android in delivering performance from available hardware resources, and doing so with less RAM. Less installed system RAM helps contribute to longer battery life.
  - "With the exception of Apple and [Google subsiderary] Motorola," AnandTech observed, "literally every single OEM we've worked with ships (or has shipped) at least one device that has cheated in benchmarks."

APL0898, the Samsung version of the A9.

APL1022, the TSMC version of the A9.



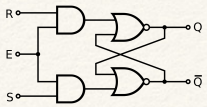


Sources: http://appleinsider.com/articles/16/01/13/antutu-iphone-6s-performance-thrashes-high-spec-androids-huawei-mate-8-samsung-note-5-nexus-6p

IBM System 370 Architecture

Source: https://en.wikipedia.org/wiki/IBM_System/370#/media/File:IBM_370-145_2.png

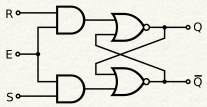
- IBM System/370 architecture
  - Was introduced in 1970
  - Included a number of models
  - Could upgrade to a more expensive, faster model without having to abandon original software
  - New models are introduced with improved technology, but retain the same architecture so that the customer's software investment is protected
  - Architecture has survived to this day as the architecture of IBM's mainframe product line



Structure and Function

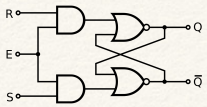
- Hierarchical system
  - Set of interrelated subsystems
- Hierarchical nature of complex systems is essential to both their design and their description
- Designer need only deal with a particular level of the system at a time
  - Concerned with structure and function at each level

- Structure
  - The way in which components relate to each other
- Function
  - The operation of individual components as part of the structure



Functions

- There are four basic functions that a computer can perform:
  - Data processing
    - Data may take a wide variety of forms and the range of processing requirements is broad
  - Data storage
    - Short-term
    - Long-term
  - Data movement
    - Input-output (I/O) - when data are received from or delivered to a device (peripheral) that is directly connected to the computer
    - Data communications – when data are moved over longer distances, to or from a remote device
  - Control
    - A control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions



Structure

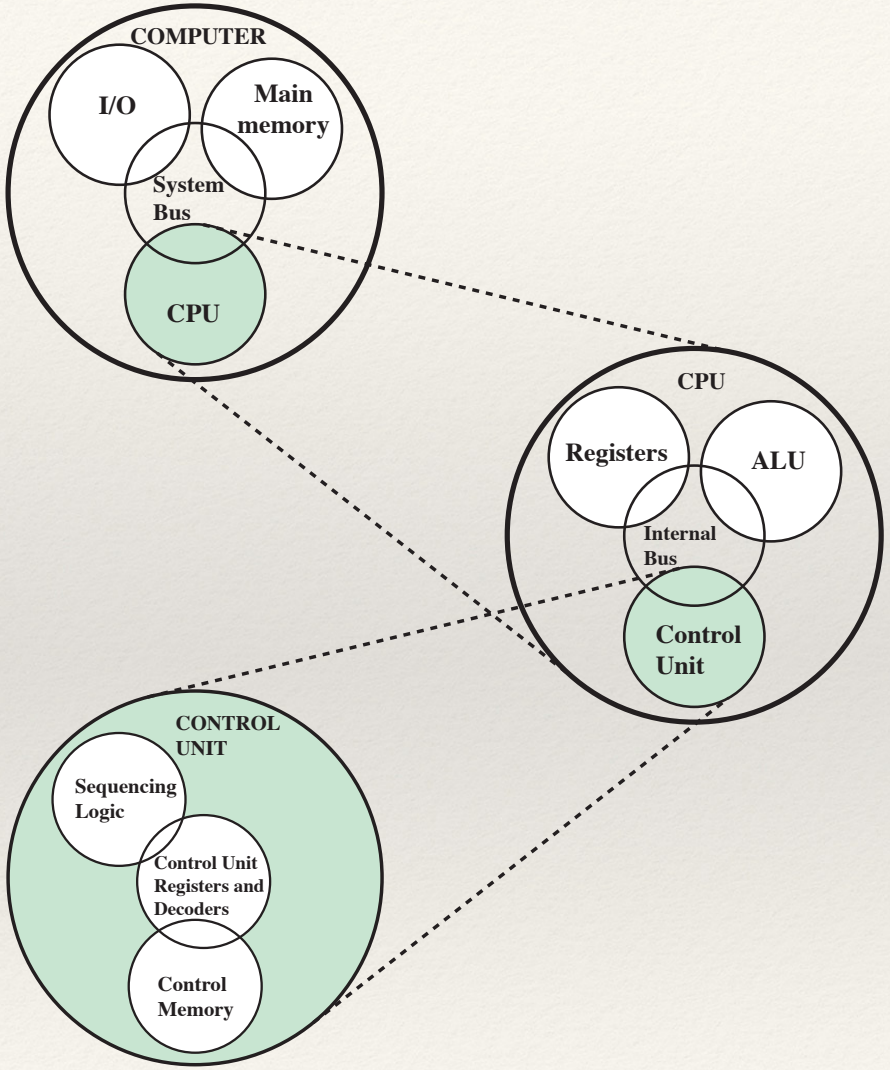
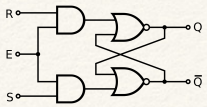


Figure 1.1 A Top-Down View of a Computer

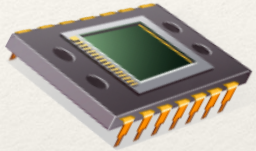
Four Main Structural Components

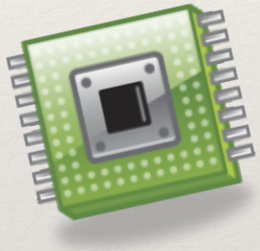
- CPU – controls the operation of the computer and performs its data processing functions
- Main Memory – stores data
- I/O – moves data between the computer and its external environment
- System Interconnection – some mechanism that provides for communication among CPU, main memory, and I/O

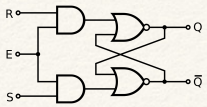


Major Structural Components of CPU

- Control Unit
  - Controls the operation of the CPU and hence the computer
- Arithmetic and Logic Unit (ALU)
  - Performs the computer's data processing function
- Registers
  - Provide storage internal to the CPU
- CPU Interconnection
  - Some mechanism that provides for communication among the control unit, ALU, and registers

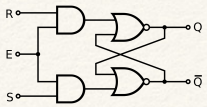






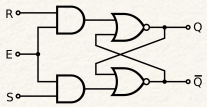
Multicore Computer Structure

- Central processing unit (CPU)
  - Portion of the computer that fetches and executes instructions
  - Consists of an ALU, a control unit, and registers
  - Referred to as a processor in a system with a single processing unit
- Core
  - An individual processing unit on a processor chip
  - May be equivalent in functionality to a CPU on a single-CPU system
  - Specialized processing units are also referred to as cores
- Processor
  - A physical piece of silicon containing one or more cores
  - Is the computer component that interprets and executes instructions
  - Referred to as a multicore processor if it contains multiple cores

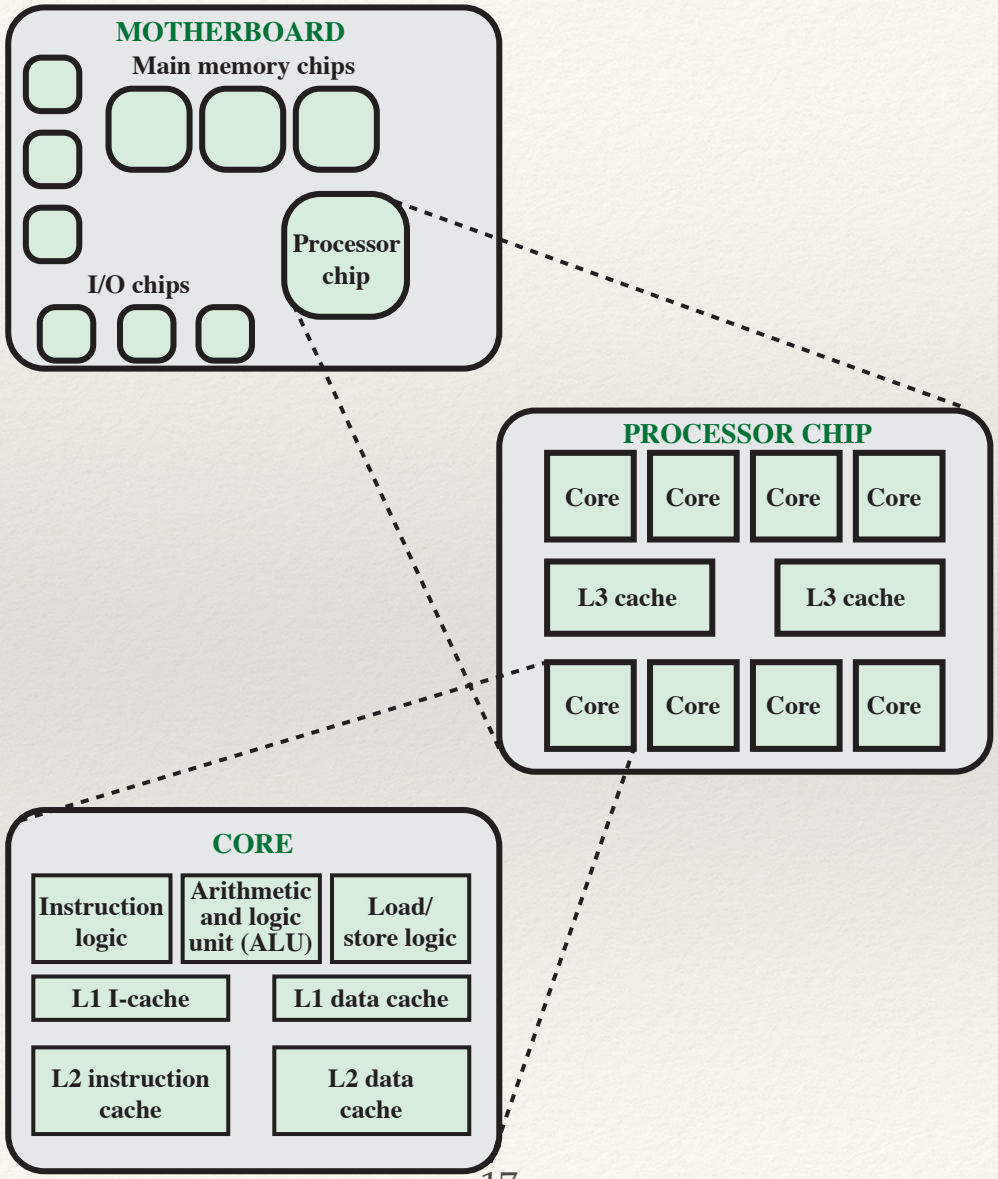


Cache Memory

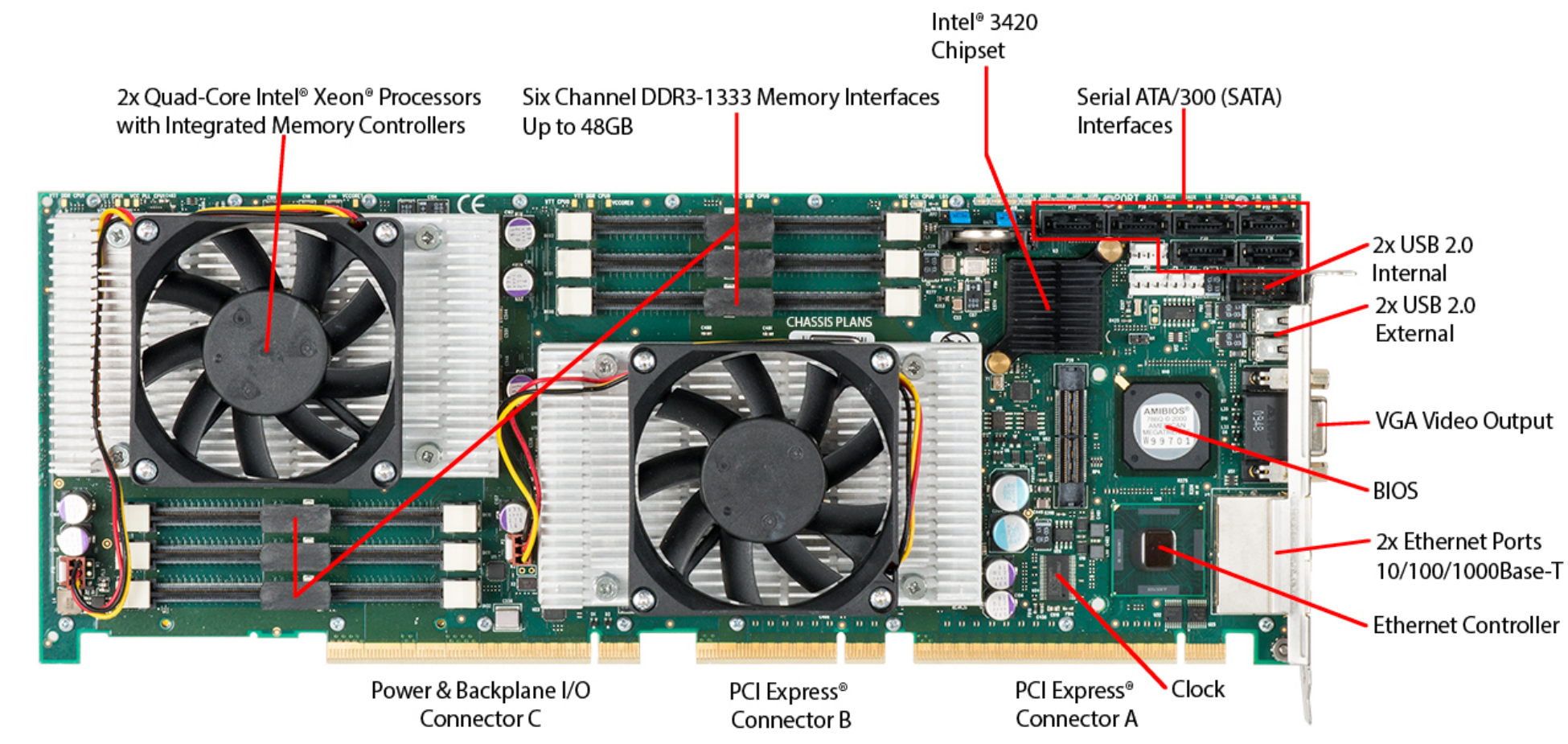
- Multiple layers of memory between the processor and main memory
- Is smaller and faster than main memory
- Used to speed up memory access by placing in the cache data from main memory that is likely to be used in the near future
- A greater performance improvement may be obtained by using multiple levels of cache, with level 1 (L1) closest to the core and additional levels (L2, L3, etc.) progressively farther from the core



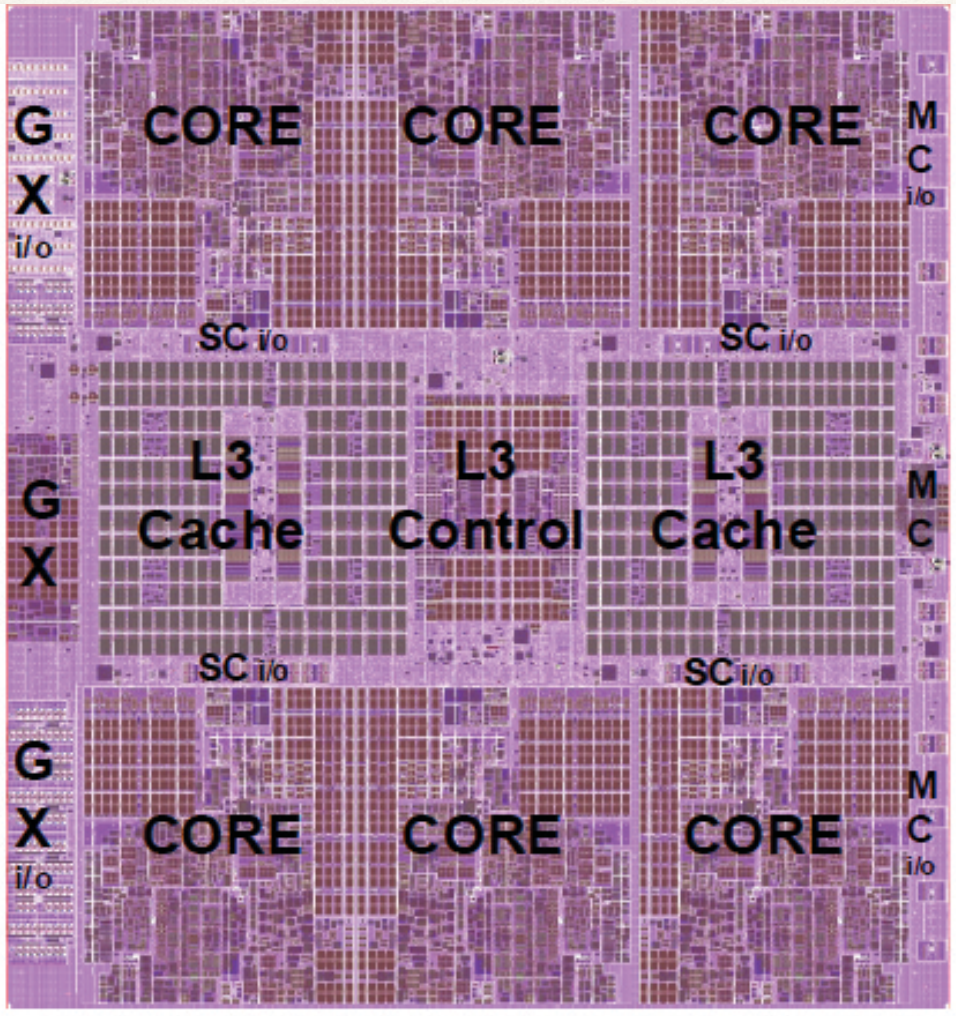
Major Elements of a Multicore Computer



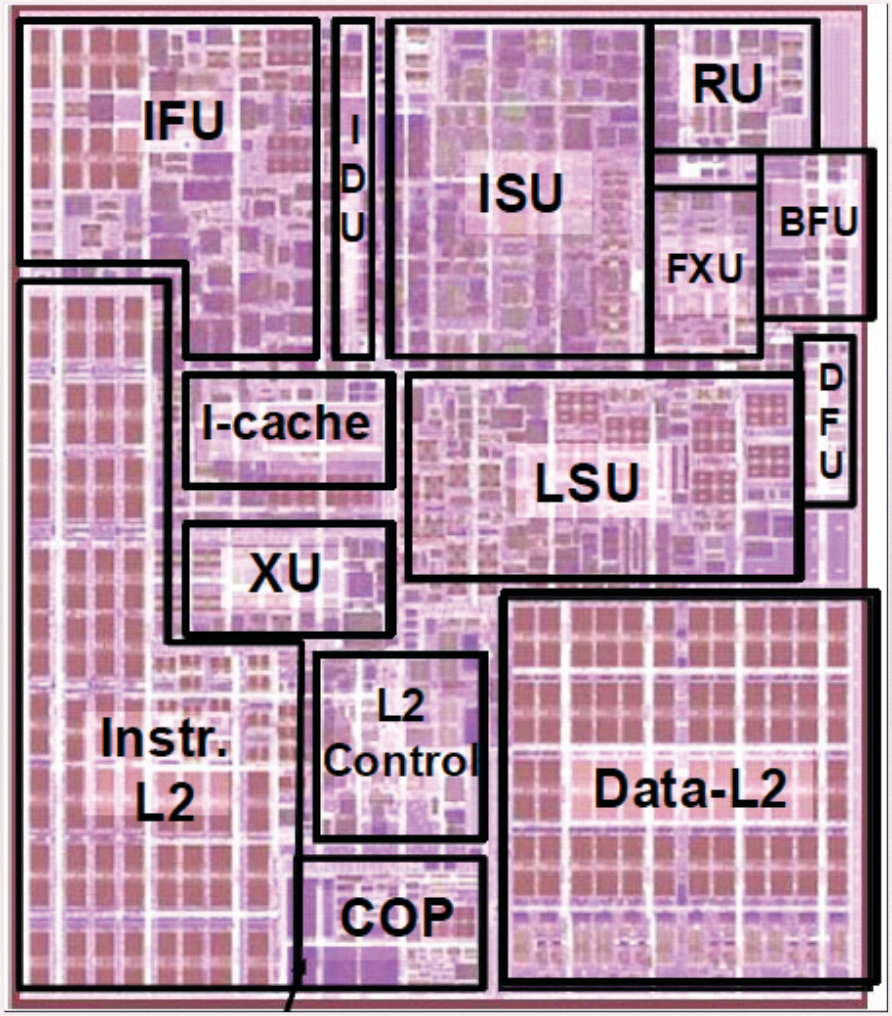
Motherboard with Two Intel Quad-Core Xeon Processors



zEnterprise EC12 Processor Unit



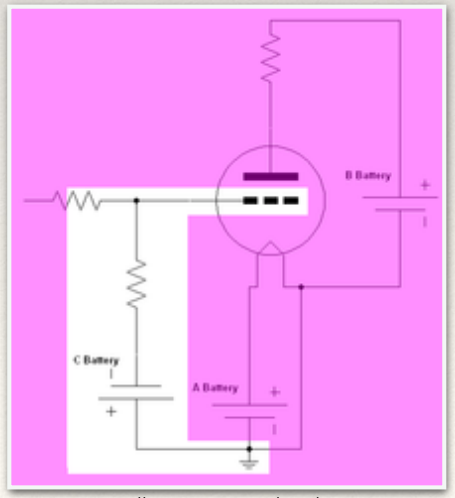
Chip Diagram



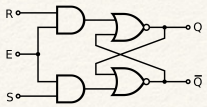
Core Layout

History of Computers - 1st Generation

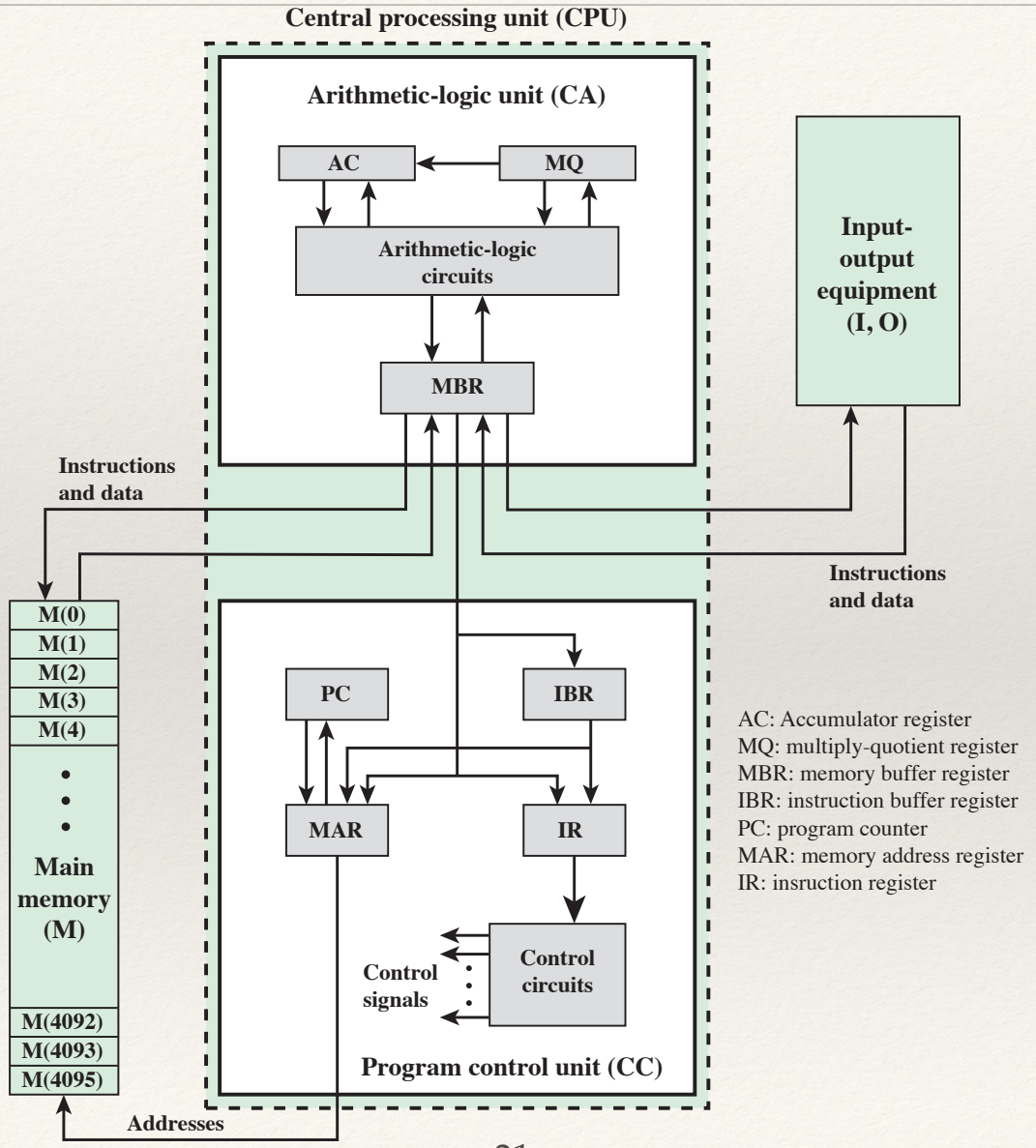
- ***Vacuum tubes*** were used for digital logic elements and memory
- IAS computer
  - Fundamental design approach was the stored program concept
    - Attributed to the mathematician John von Neumann
    - First publication of the idea was in 1945 for the EDVAC
  - Design began at the Princeton Institute for Advanced Studies
  - Completed in 1952
  - Prototype of all subsequent general-purpose computers



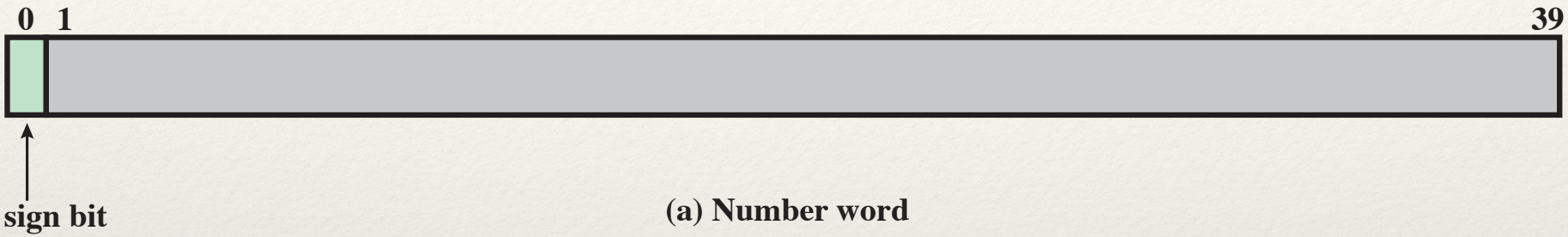
Source: https://en.wikipedia.org/wiki/Vacuum_tube



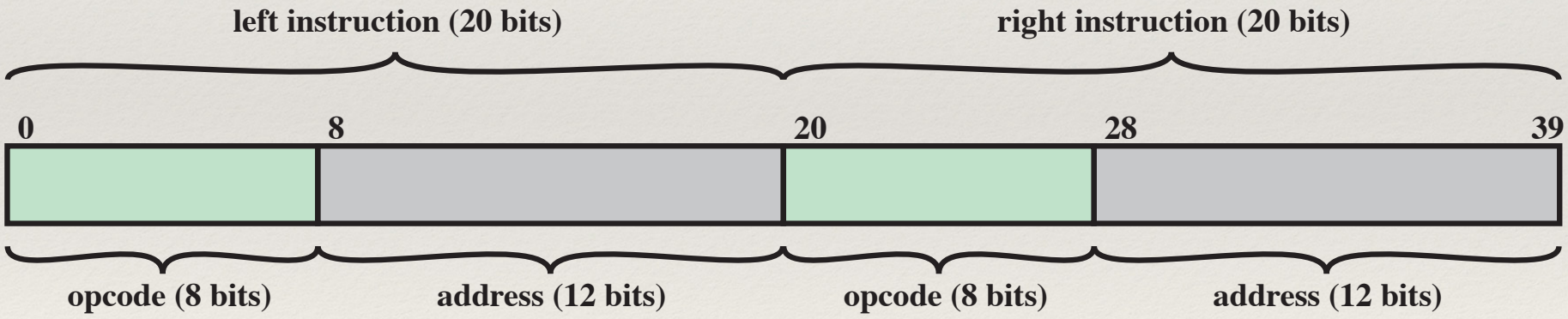
IAS Structure



IAS Memory Formats



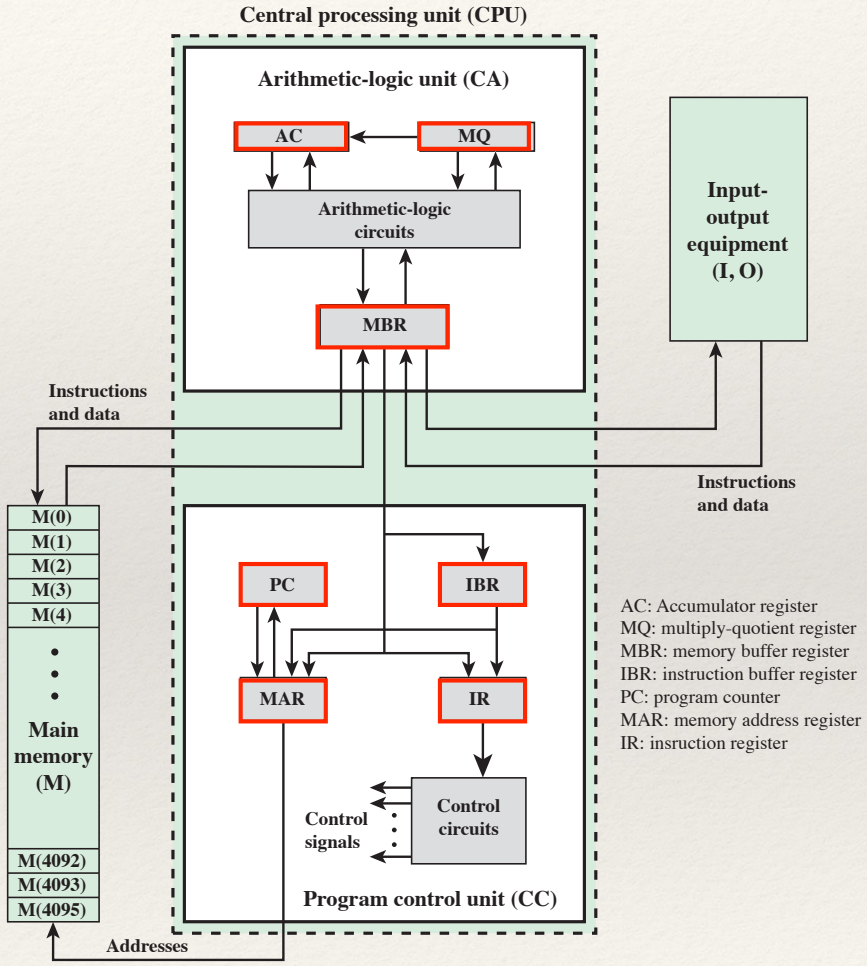
(a) Number word

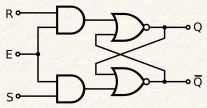


(b) Instruction word

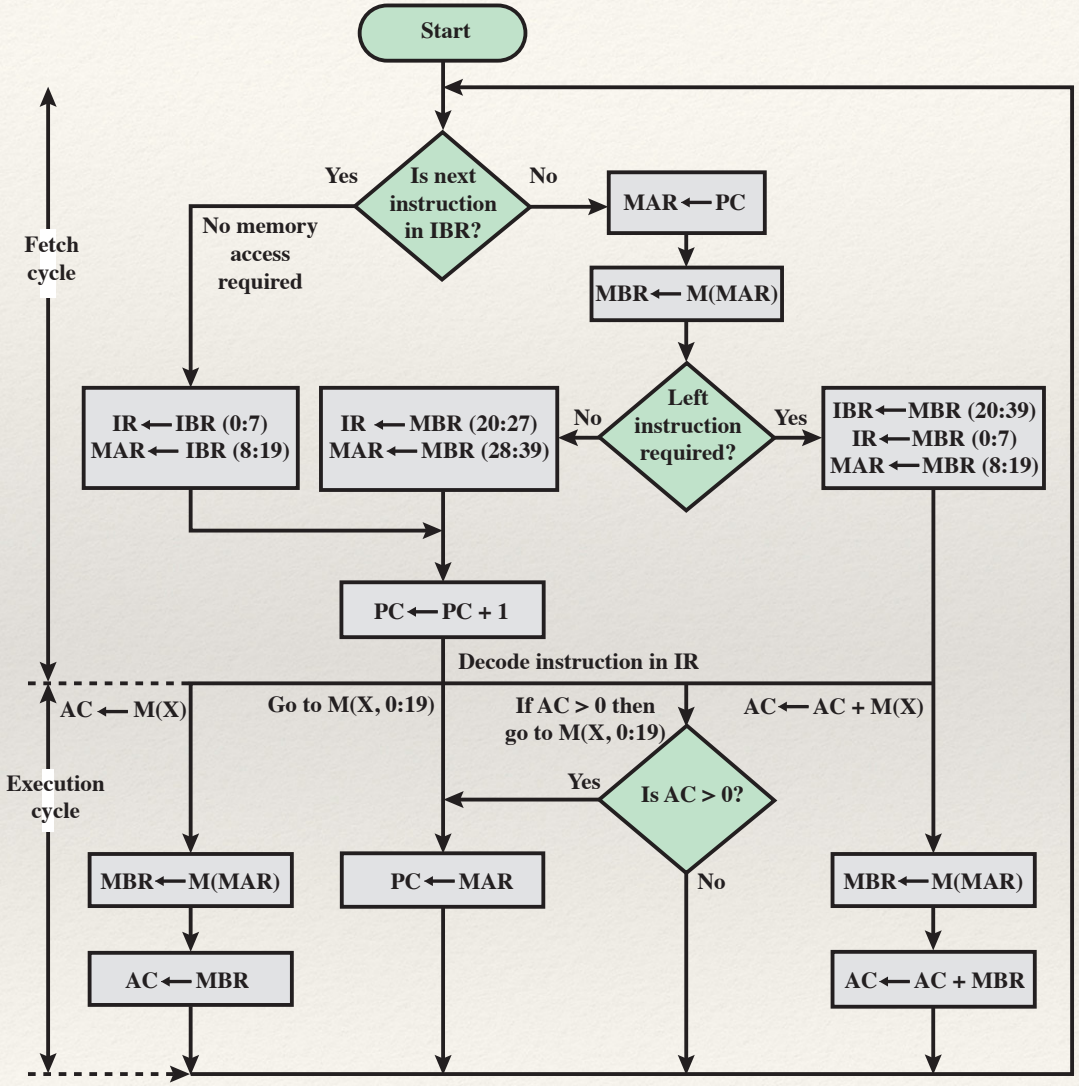
Registers

Memory buffer register (MBR)	• Contains a word to be stored in memory or sent to the I/O unit • Or is used to receive a word from memory or from the I/O unit
Memory address register (MAR)	• Specifies the address in memory of the word to be written from or read into the MBR
Instruction register (IR)	• Contains the 8-bit opcode instruction being executed
Instruction buffer register (IBR)	• Employed to temporarily hold the right-hand instruction from a word in memory
Program counter (PC)	• Contains the address of the next instruction pair to be fetched from memory
Accumulator (AC) and multiplier quotient (MQ)	• Employed to temporarily hold operands and results of ALU operations

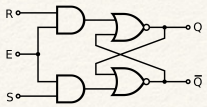




Partial Flowchart of IAS Operation

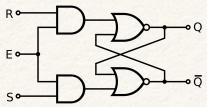


M(X) = contents of memory location whose address is X
(i:j) = bits i through j



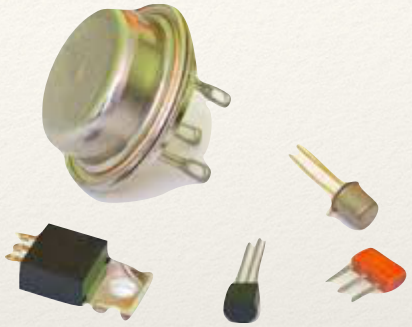
The IAS Instruction Set

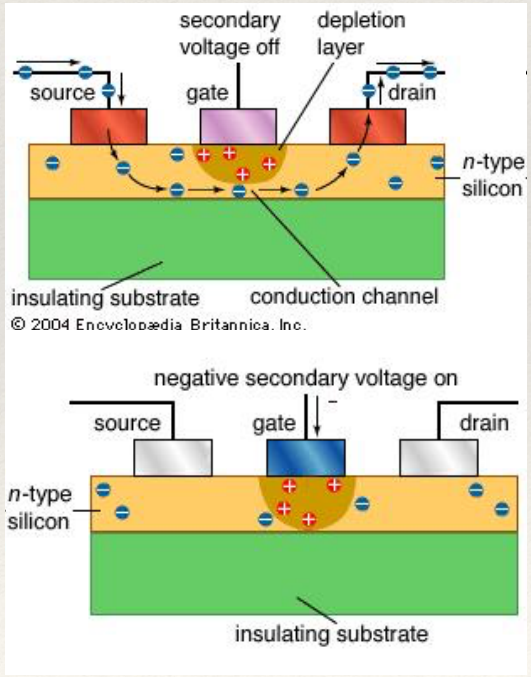
Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP+ M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
		JUMP+ M(X,20:39)	*If number in the accumulator is nonnegative, take next instruction from right half of M(X)*
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; i.e., shift left one bit position
	00010101	RSH	Divide accumulator by 2; i.e., shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits of AC



History of Computers: 2nd Generation

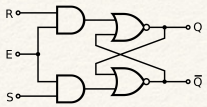
- Based on ***Transistors***
  - Smaller
  - Cheaper
  - Dissipates less heat than a vacuum tube
  - Is a ***solid state device*** made from silicon
  - Was invented at Bell Labs in 1947
  - It was not until the late 1950's that fully transistorized computers were commercially available





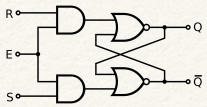
Computer Generations

Generation	Approximate Dates	Technology	Typical Speed (operations per second)
1	1946-1957	Vacuum tube	40,000
2	1957-1964	Transistor	200,000
3	1965-1971	Small and medium scale integration	1,000,000
4	1972-1977	Large scale integration	10,000,000
5	1978-1991	Very large scale integration	100,000,000
6	1991-	Ultra large scale integration	>1,000,000,000

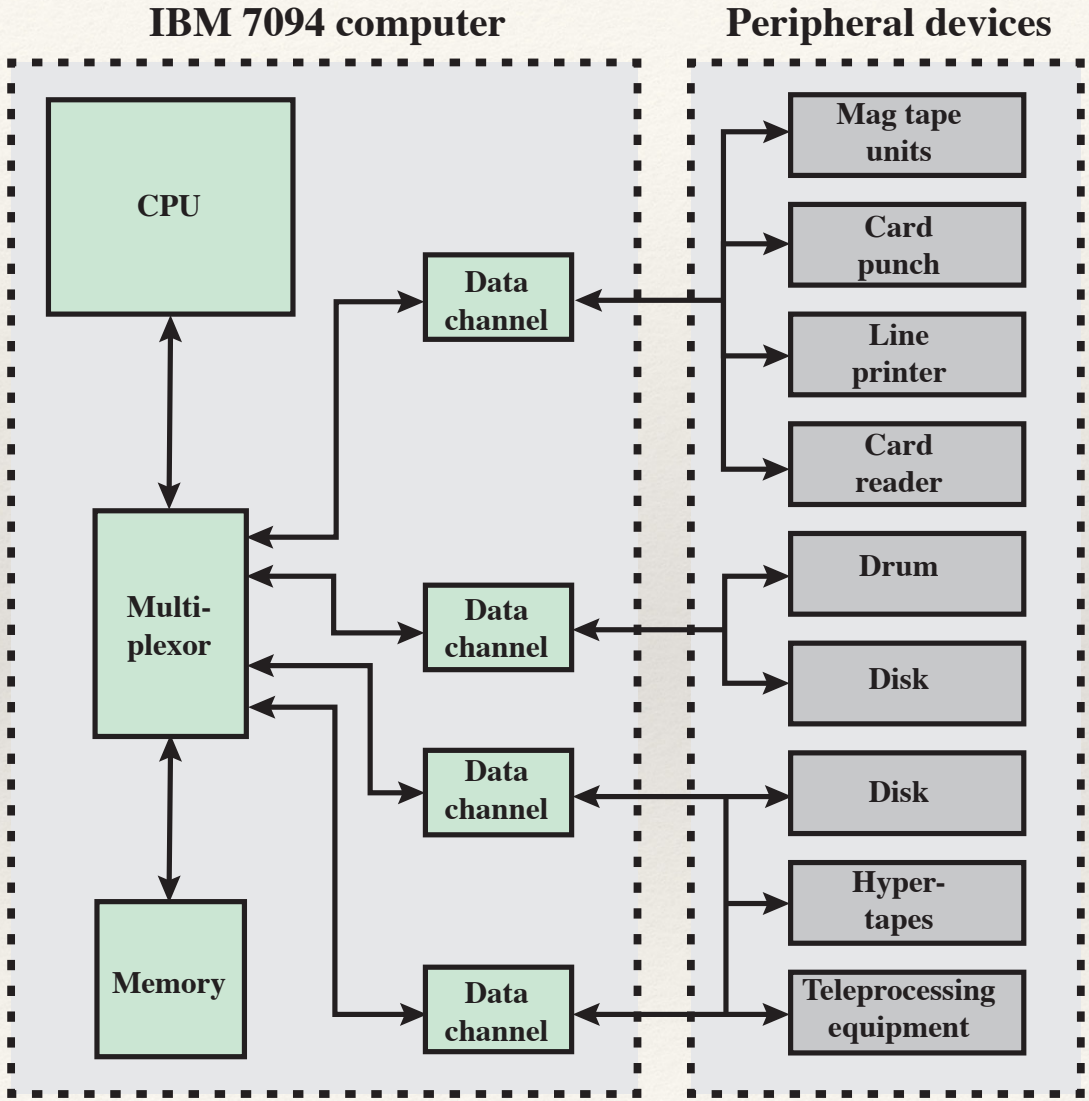


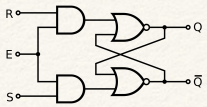
2nd Generation Computers

- Introduced:
  - More complex arithmetic and logic units and control units
  - The use of high-level programming languages
  - Provision of *system software* which provided the ability to:
    - Load programs
    - Move data to peripherals
    - Libraries perform common computations



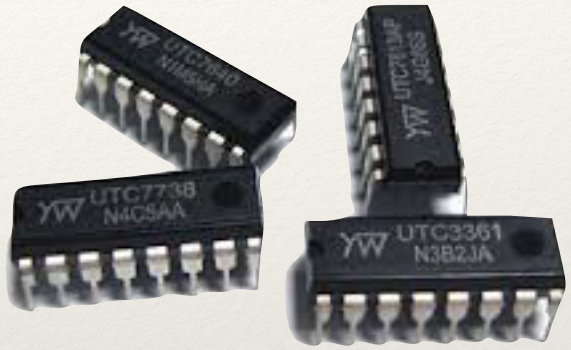
An IBM 7094 Configuration

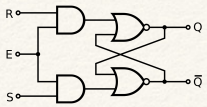




History of Computers: 3rd Generation

- Based on ***Integrated Circuits***
  - Invented in 1958
  - *Discrete component*
    - Single, self-contained transistor
    - Manufactured separately, packaged in their own containers, and soldered or wired together onto masonite-like circuit boards
    - Manufacturing process was expensive and cumbersome
  - The two most important members of the third generation were the IBM System/360 and the DEC PDP-8

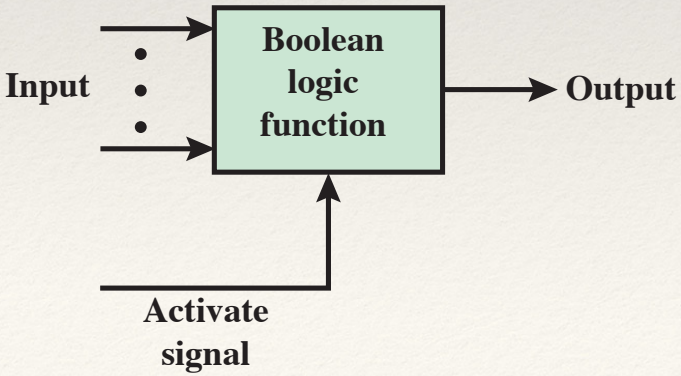


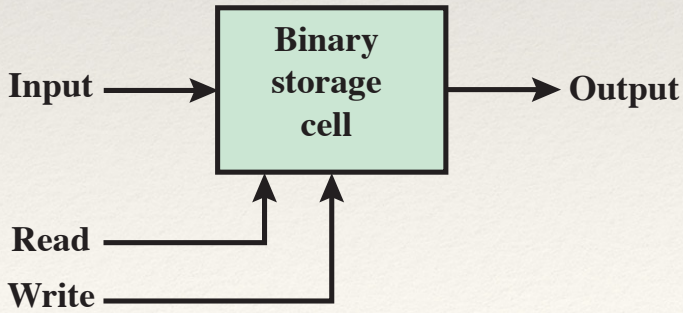


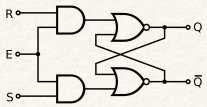
Integrated Circuits

- Data storage – provided by memory cells
- Data processing – provided by gates
- Data movement – the paths among components are used to move data from memory to memory and from memory through gates to memory
- Control – the paths among components can carry control signals
- A computer consists of gates, memory cells, and interconnections among these elements

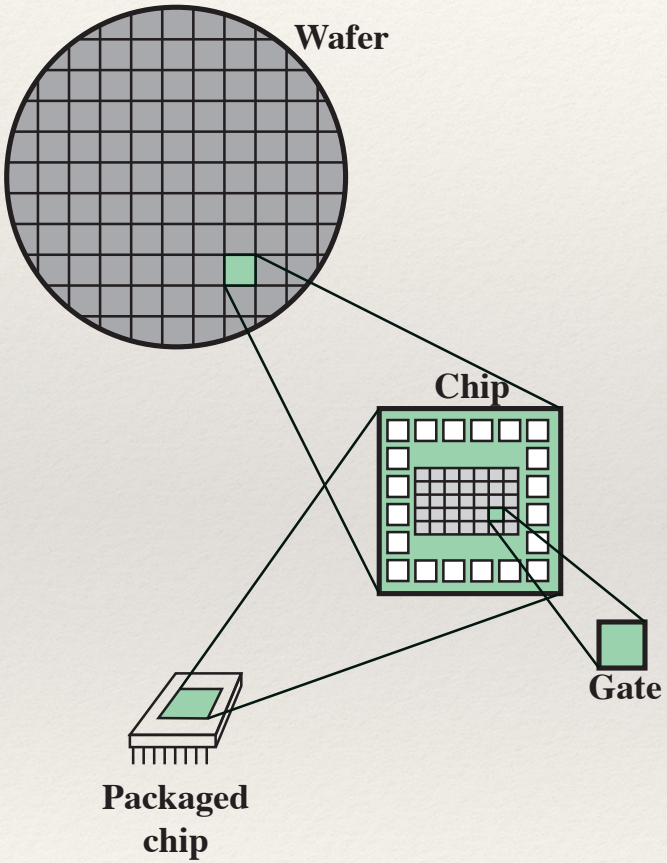
- The gates and memory cells are constructed of simple digital electronic components
- Exploits the fact that such components as transistors, resistors, and conductors can be fabricated from a semiconductor such as silicon
- Many transistors can be produced at the same time on a single wafer of silicon
- Transistors can be connected with a processor metallization to form circuits







Wafer, Chip, Gate and Transistor Count



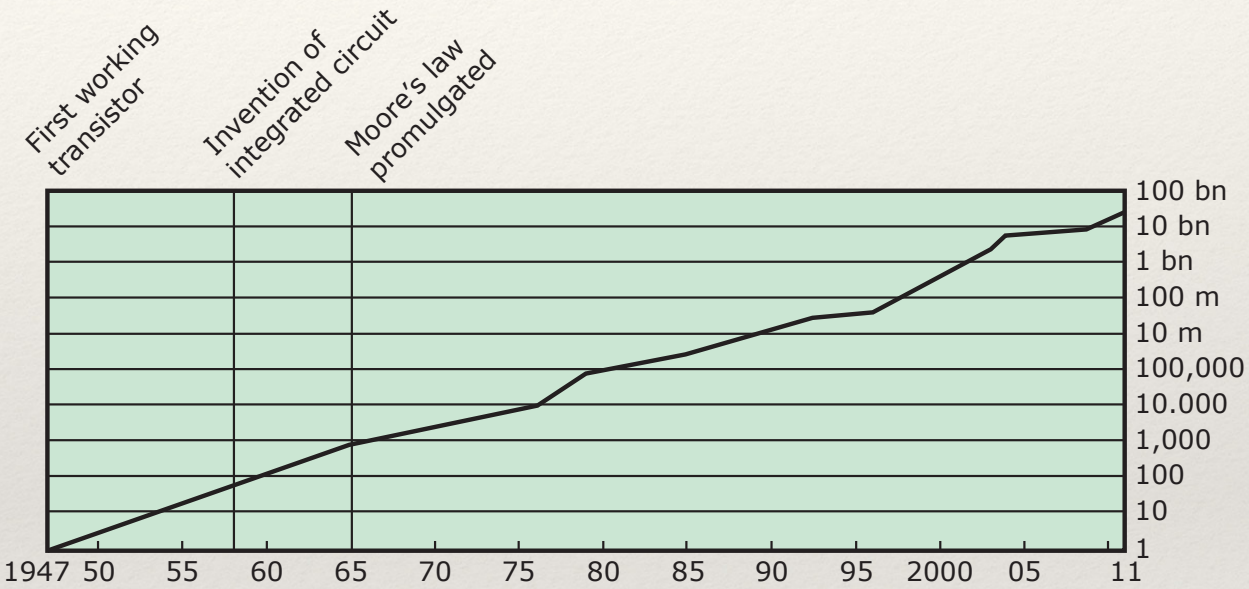
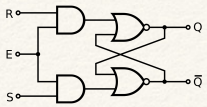


Figure 1.12 Growth in Transistor Count on Integrated Circuits (DRAM memory)



Moore's Law

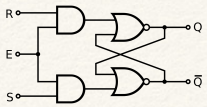
1965; Gordon Moore – co-founder of Intel

Observed number of transistors that could be put on a single chip was doubling every year

The pace slowed to a doubling every 18 months in the 1970's but has sustained that rate ever since

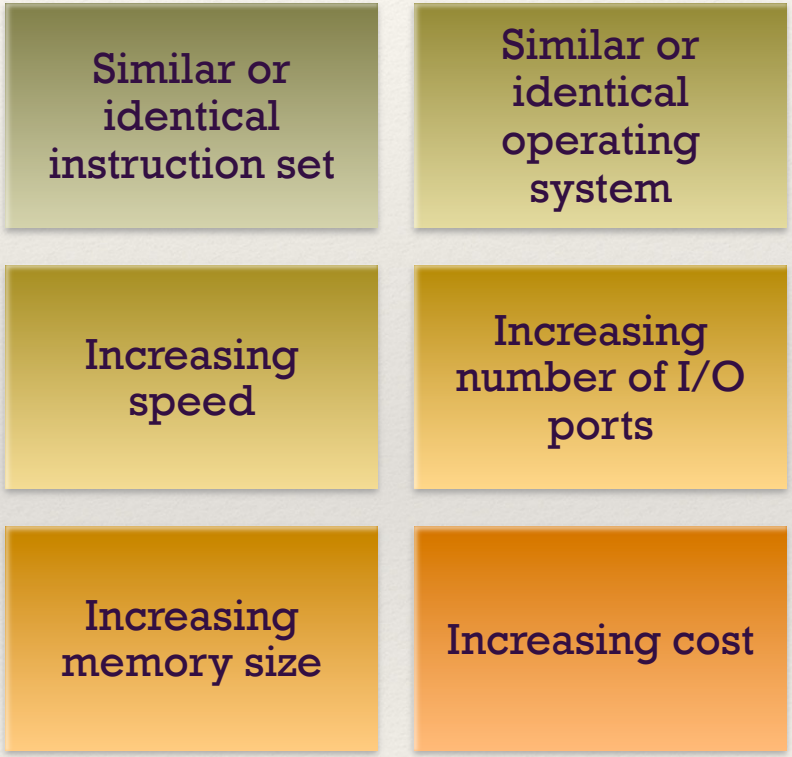
Consequences of Moore's law:

- The cost of computer logic and memory circuitry has fallen at a dramatic rate
- The electrical path length is shortened, increasing operating speed
- Computer becomes smaller and is more convenient to use in a variety of environments
- Reduction in power and cooling requirements
- Fewer interchip connections

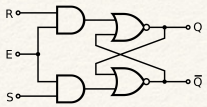


IBM System/360

- Announced in 1964
- Product line was incompatible with older IBM machines
- Was the success of the decade and cemented IBM as the overwhelmingly dominant computer vendor
- The architecture remains to this day the architecture of IBM's mainframe computers
- Was the industry's first planned family of computers
  - Models were compatible in the sense that a program written for one model should be capable of being executed by another model in the series

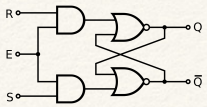


Family Characteristics



History of Computers: Later Generations

- Evolved based on introductions of LSI, VLSI and ULSI technologies
- Semiconductor Memory
  - vs memory that was constructed from tiny rings of ferromagnetic material
- Microprocessors
  - In 1971, Intel managed to deliver the first chip to contain *all* of the components of a CPU on a single chip, marking the born of microprocessor



Semiconductor Memory

In 1970 Fairchild produced the first relatively capacious semiconductor memory

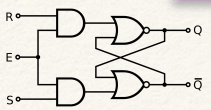
Chip was about the size of a single core	Could hold 256 bits of memory	Non-destructive	Much faster than core

In 1974 the price per bit of semiconductor memory dropped below the price per bit of core memory

There has been a continuing and rapid decline in memory cost accompanied by a corresponding increase in physical memory density	Developments in memory and processor technologies changed the nature of computers in less than a decade

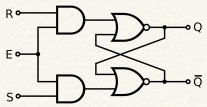
Since 1970 semiconductor memory has been through 13 generations

Each generation has provided four times the storage density of the previous generation, accompanied by declining cost per bit and declining access time



Microprocessors

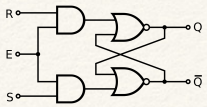
- The density of elements on processor chips continued to rise
  - More and more elements were placed on each chip so that fewer and fewer chips were needed to construct a single computer processor
- 1971 Intel developed 4004
  - First chip to contain all of the components of a CPU on a single chip
  - Birth of microprocessor
- 1972 Intel developed 8008
  - First 8-bit microprocessor
- 1974 Intel developed 8080
  - First general purpose microprocessor
  - Faster, has a richer instruction set, has a large addressing capability



Evolution of Intel Microprocessors (1)

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2,300	3,500	6,000	29,000	29,000
Feature size (μm)	10	8	6	3	6
Addressable memory	640 Bytes	16 KB	64 KB	1 MB	1 MB

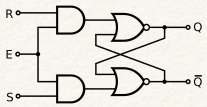
	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz - 12.5 MHz	16 MHz - 33 MHz	16 MHz - 33 MHz	25 MHz - 50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (μm)	1.5	1	1	0.8 - 1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB



Evolution of Intel Microprocessors (2)

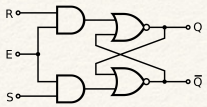
	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz - 33 MHz	60 MHz - 166 MHz,	150 MHz - 200 MHz	200 MHz - 300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 4960X
Introduced	1999	2000	2006	2013
Clock speeds	450 - 660 MHz	1.3 - 1.8 GHz	1.06 - 1.2 GHz	4 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1.86 billion
Feature size (nm)	250	180	65	22
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/15 MB L3
Number of cores	1	1	2	6



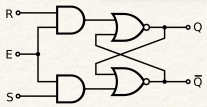
The Evolution of the Intel x86 Architecture

- Two processor families are the Intel x86 and the ARM architectures
- Current x86 offerings represent the results of decades of design effort on complex instruction set computers (CISCs)
- An alternative approach to processor design is the reduced instruction set computer (RISC)
- ARM architecture is used in a wide variety of embedded systems and is one of the most powerful and best-designed RISC-based systems on the market



Highlights of the Evolution of the Intel Product Line (1)

8080	8086	80286	80386	80486
• World's first general-purpose microprocessor • 8-bit machine, 8-bit data path to memory • Was used in the first personal computer (Altair)	• A more powerful 16-bit machine • Has an instruction cache, or queue, that prefetches a few instructions before they are executed • The first appearance of the x86 architecture • The 8088 was a variant of this processor and used in IBM's first personal computer (securing the success of Intel	• Extension of the 8086 enabling addressing a 16-MB memory instead of just 1MB	• Intel's first 32-bit machine • First Intel processor to support multitasking	• Introduced the use of much more sophisticated and powerful cache technology and sophisticated instruction pipelining • Also offered a built-in math coprocessor



Highlights of the Evolution of the Intel Product Line (2)

Pentium

- Intel introduced the use of superscalar techniques, which allow multiple instructions to execute in parallel

Pentium Pro

- Continued the move into superscalar organization with aggressive use of register renaming, branch prediction, data flow analysis, and speculative execution

Pentium II

- Incorporated Intel MMX technology, which is designed specifically to process video, audio, and graphics data efficiently

Pentium III

- Incorporated additional floating-point instructions
- Streaming SIMD Extensions (SSE)

Pentium 4

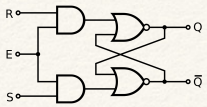
- Includes additional floating-point and other enhancements for multimedia

Core

- First Intel x86 micro-core

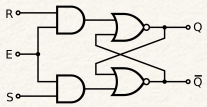
Core 2

- Extends the Core architecture to 64 bits
- Core 2 Quad provides four cores on a single chip
- More recent Core offerings have up to 10 cores per chip
- An important addition to the architecture was the Advanced Vector Extensions instruction set

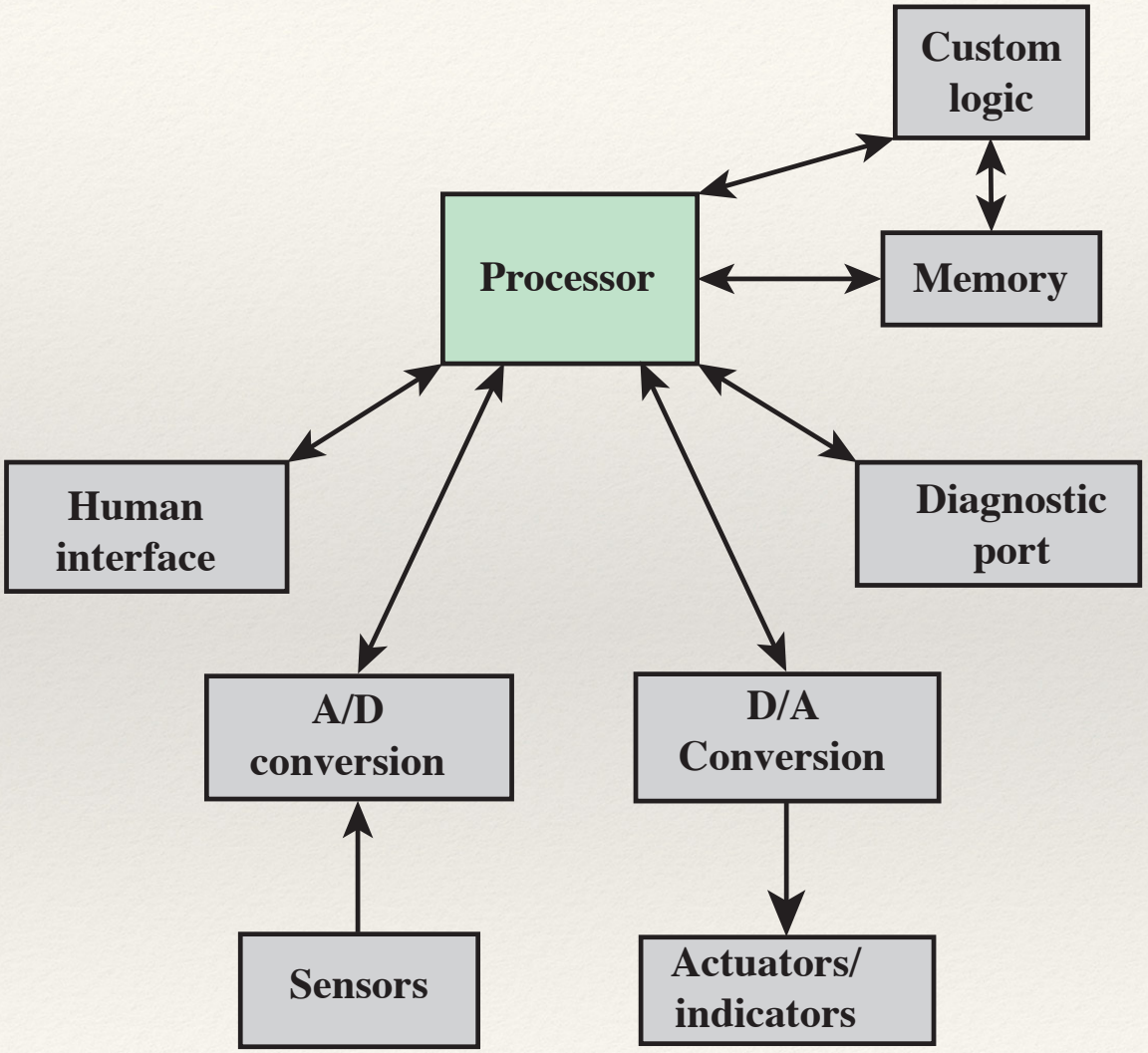


Embedded Systems

- The use of electronics and software within a product
- Billions of computer systems are produced each year that are embedded within larger devices
- Today many devices that use electric power have an embedded computing system
- Often embedded systems are tightly coupled to their environment
- This can give rise to real-time constraints imposed by the need to interact with the environment
- Constraints such as required speeds of motion, required precision of measurement, and required time durations, dictate the timing of software operations
- If multiple activities must be managed simultaneously this imposes more complex real-time constraints

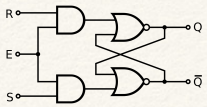


Organization of an Embedded System



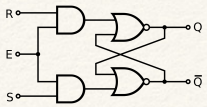
Internet of Things (IoT)

- Term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors
- Is primarily driven by deeply embedded devices
- Generations of deployment culminating in the IoT:
  - Information technology (IT)
    - PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people and primarily using wired connectivity
  - Operational technology (OT)
    - Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA, process control, and kiosks, bought as appliances by enterprise OT people and primarily using wired connectivity
  - Personal technology
    - Smartphones, tablets, and eBook readers bought as IT devices by consumers exclusively using wireless connectivity and often multiple forms of wireless connectivity
  - Sensor/actuator technology
    - Single-purpose devices bought by consumers, IT, and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems
- It is the fourth generation that is usually thought of as the IoT and it is marked by the use of billions of embedded devices

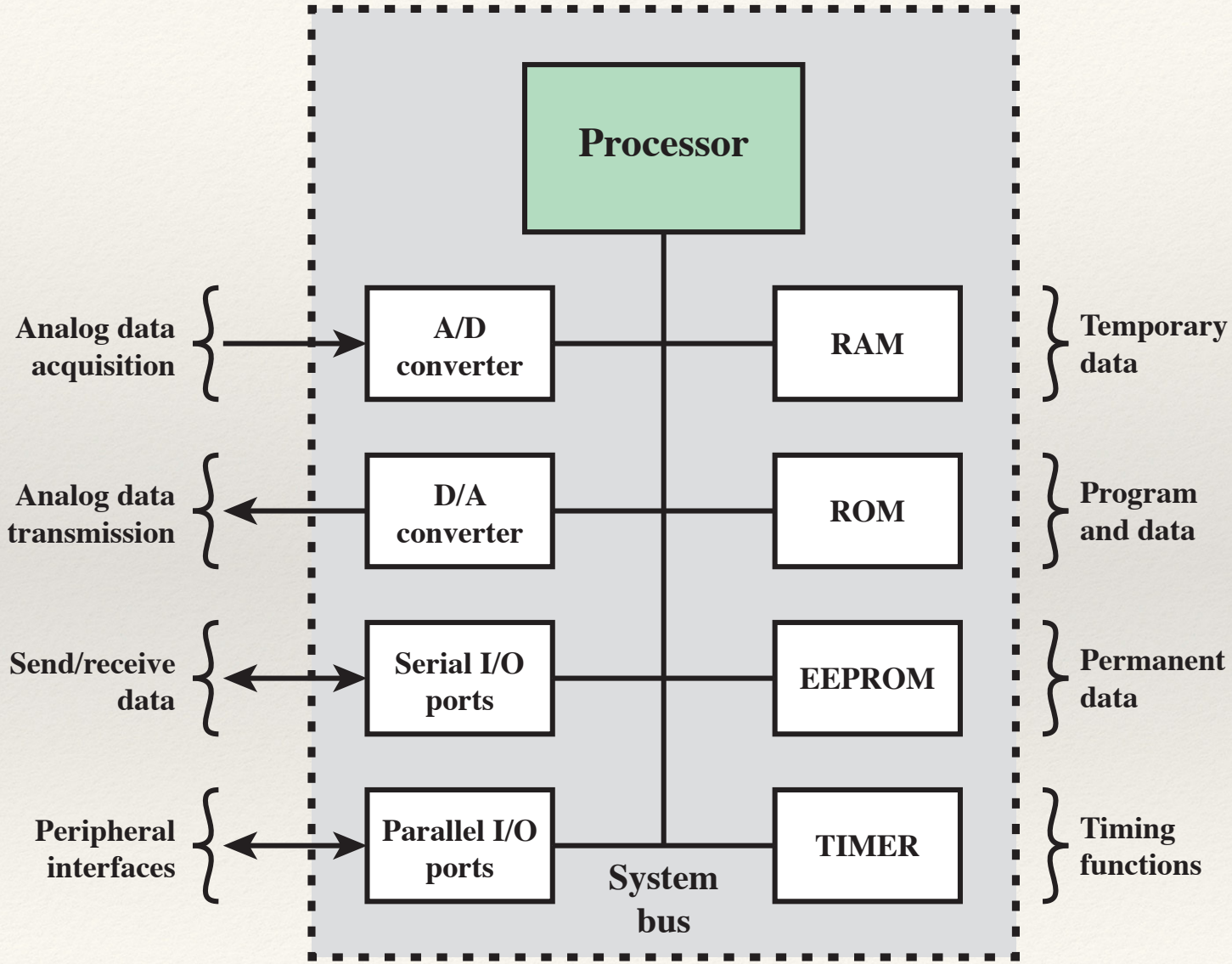


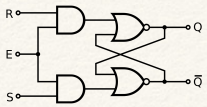
Approaches for Embedded Systems

- There are two general approaches to developing an embedded operating system (OS):
  - Take an existing OS and adapt it for the embedded application
  - Design and implement an OS intended solely for embedded use
- Application processors vs Dedicated Processor
  - Application processors
    - Defined by the processor's ability to execute complex operating systems
    - General-purpose in nature
    - An example is the smartphone – the embedded system is designed to support numerous apps and perform a wide variety of functions
  - Dedicated processor
    - Is dedicated to one or a small number of specific tasks required by the host device
    - Because such an embedded system is dedicated to a specific task or tasks, the processor and associated components can be engineered to reduce size and cost



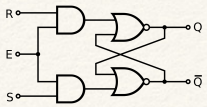
Typical Microcontroller Chip Elements



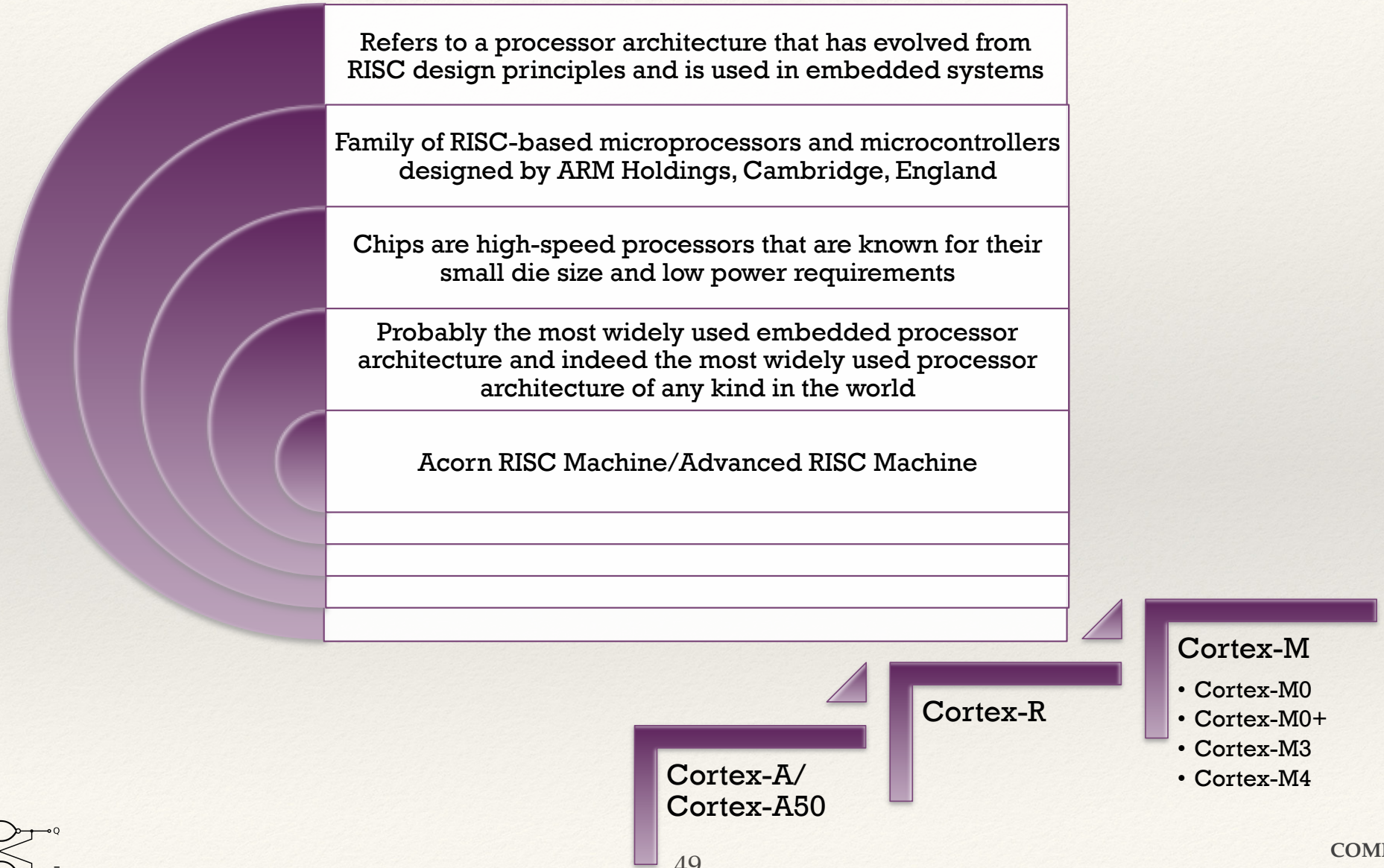


Deeply Embedded Systems

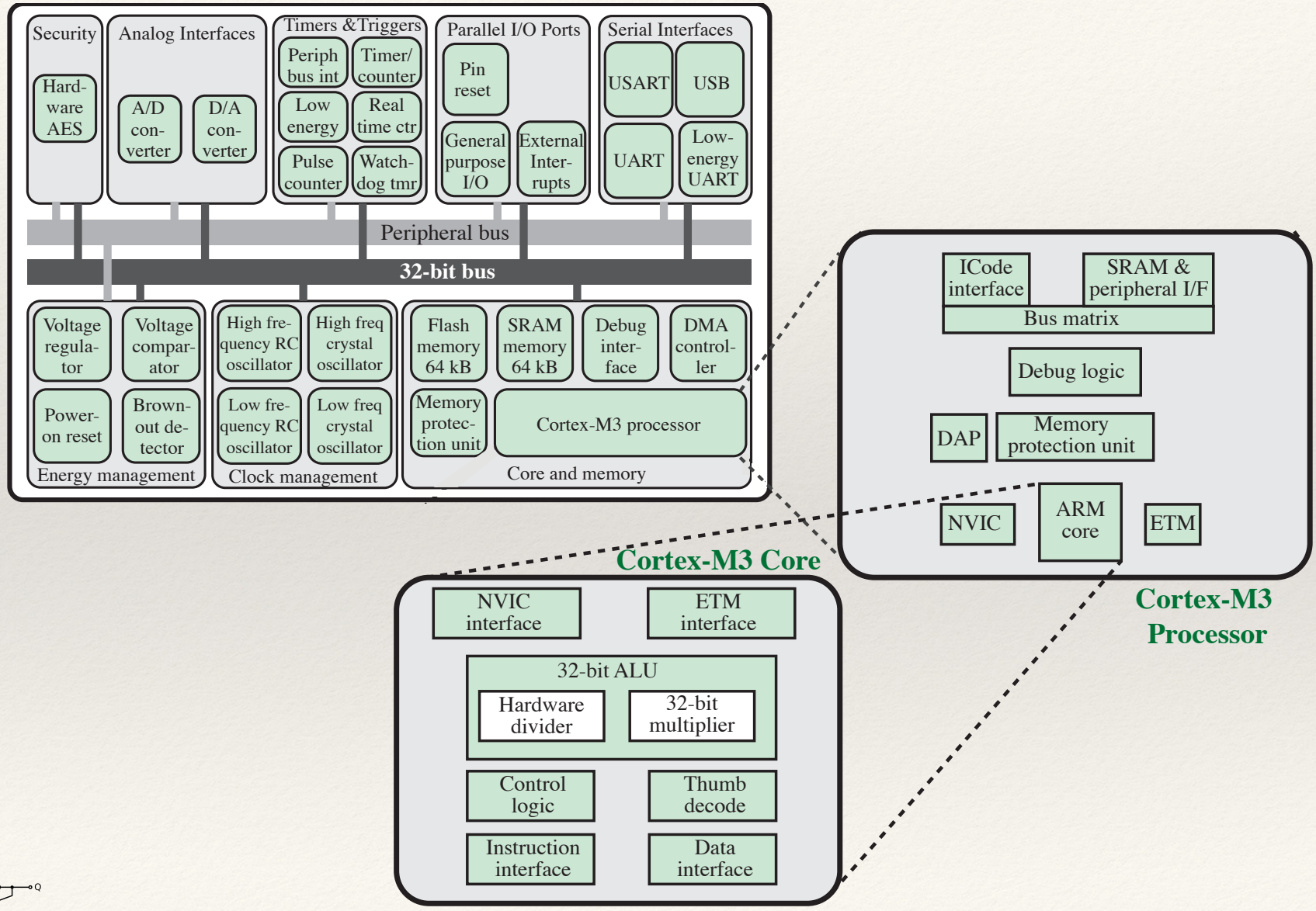
- Subset of embedded systems
- Has a processor whose behavior is difficult to observe both by the programmer and the user
- Uses a microcontroller rather than a microprocessor
- Is not programmable once the program logic for the device has been burned into ROM
- Has no interaction with a user
- Dedicated, single-purpose devices that detect something in the environment, perform a basic level of processing, and then do something with the results
- Often have wireless capability and appear in networked configurations, such as networks of sensors deployed over a large area
- Typically have extreme resource constraints in terms of memory, processor size, time, and power consumption



ARM

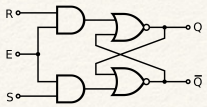

Refers to a processor architecture that has evolved from RISC design principles and is used in embedded systems
Family of RISC-based microprocessors and microcontrollers designed by ARM Holdings, Cambridge, England
Chips are high-speed processors that are known for their small die size and low power requirements
Probably the most widely used embedded processor architecture and indeed the most widely used processor architecture of any kind in the world
Acorn RISC Machine/Advanced RISC Machine
Cortex-A/ Cortex-A50
Cortex-R
Cortex-M
• Cortex-M0
• Cortex-M0+
• Cortex-M3
• Cortex-M4

Typical Microcontroller Chip based on Cortex-M3



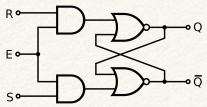
Cloud Computing

- National Institute of Standards and Technology (NIST) defines cloud computing as:
  - "A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction."
- You get economies of scale, professional network management, and professional security management
- The individual or company only needs to pay for the storage capacity and services they need
- Cloud provider takes care of security

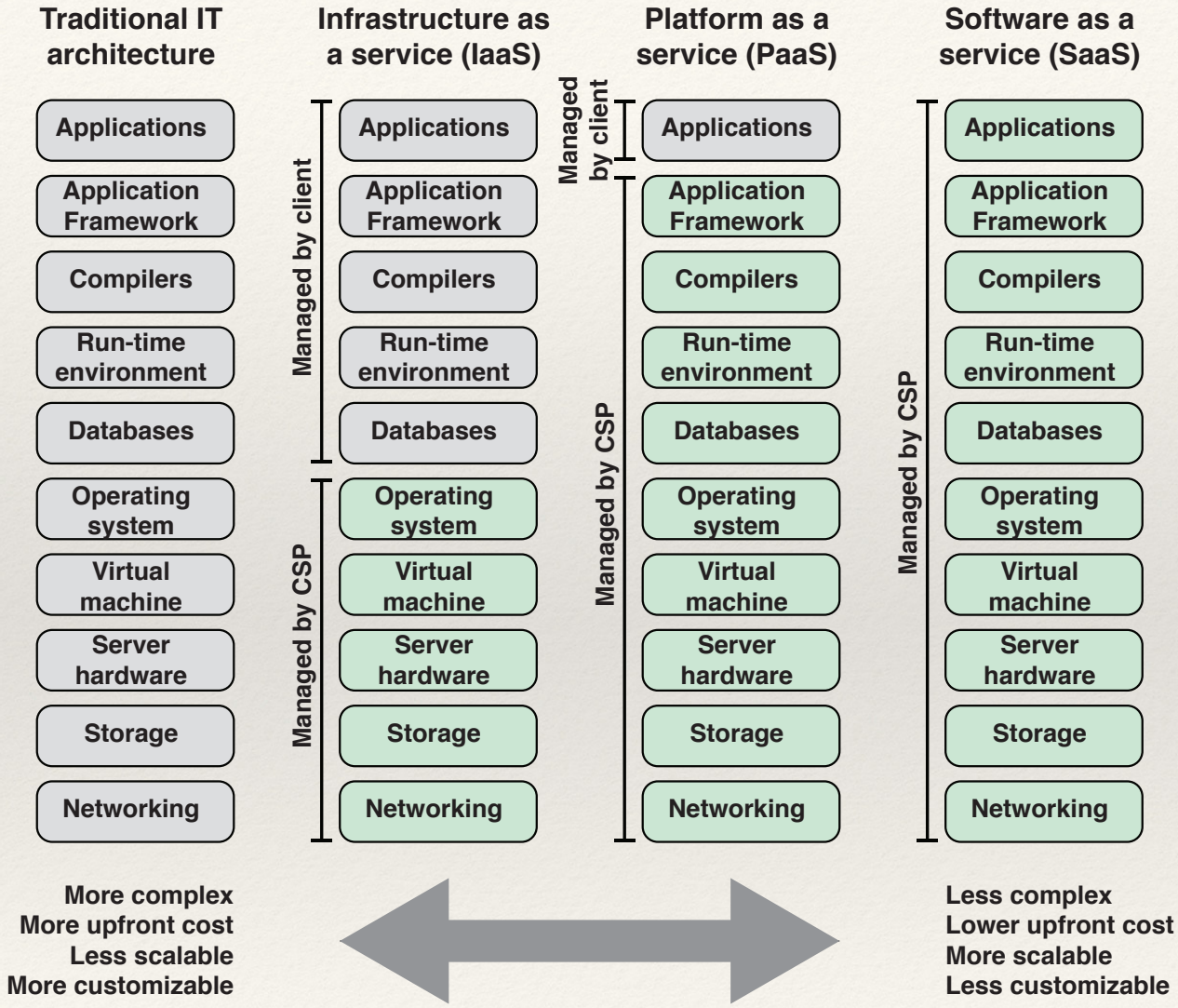


Cloud Computing (Cont'd)

- Cloud Networking
  - Refers to the networks and network management functionality that must be in place to enable cloud computing
  - One example is the provisioning of high-performance and/or high-reliability networking between the provider and subscriber
  - The collection of network capabilities required to access a cloud, including making use of specialized services over the Internet, linking enterprise data center to a cloud, and using firewalls and other network security devices at critical points to enforce access security policies
- Cloud Storage
  - Subset of cloud computing
  - Consists of database storage and database applications hosted remotely on cloud servers
  - Enables small businesses and individual users to take advantage of data storage that scales with their needs and to take advantage of a variety of database applications without having to buy, maintain, and manage the storage assets



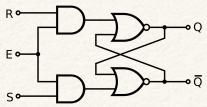
Alternative Information Technology Architectures



IT = information technology
CSP = cloud service provider

Chapter 1 - Summary

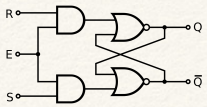
- Chapter 1 - Basic Concepts and Computer Evolution
  - Organization and architecture
  - Structure and function
  - Brief history of computers
    - The First Generation: Vacuum tubes
    - The Second Generation: Transistors
    - The Third Generation: Integrated Circuits
    - Later generations
  - The evolution of the Intel x86 architecture
  - Cloud computing
    - Basic concepts
    - Cloud services
  - Embedded systems
    - The Internet of things
    - Embedded operating systems
    - Application processors versus dedicated processors
    - Microprocessors versus microcontrollers
    - Embedded versus deeply embedded systems
  - ARM architecture
    - ARM evolution
    - Instruction set architecture
    - ARM products



Chapter 2 - Performance Issues

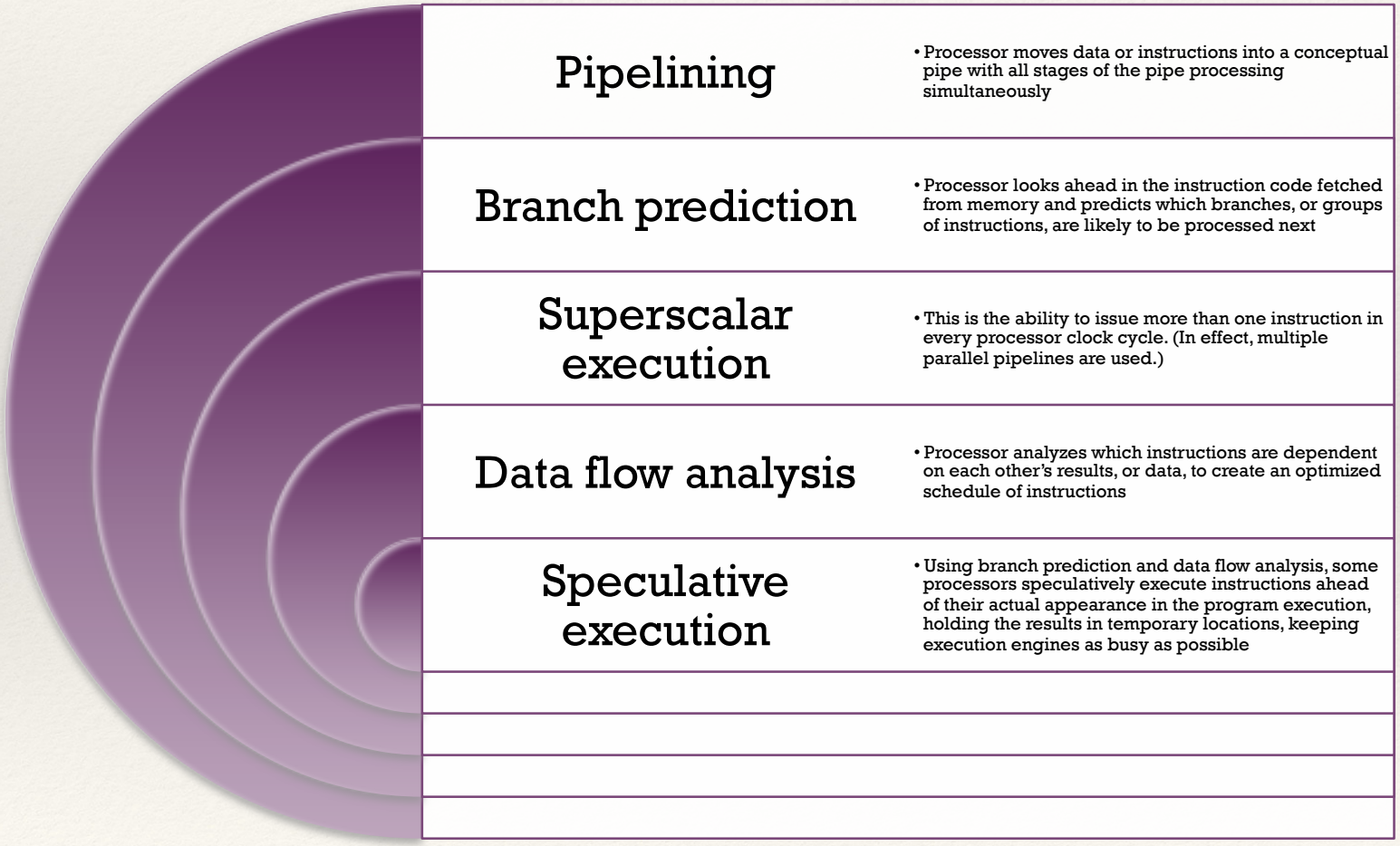
Designing for Performance

- The cost of computer systems continues to drop dramatically, while the performance and capacity of those systems continue to rise equally dramatically
- Today's laptops have the computing power of an IBM mainframe from 10 or 15 years ago
- Processors are so inexpensive that we now have microprocessors we throw away
- Desktop applications that require the great power of today's microprocessor-based systems include:
  - Image processing
  - Three-dimensional rendering
  - Speech recognition
  - Videoconferencing
  - Multimedia authoring
  - Voice and video annotation of files
  - Simulation modeling
- Businesses are relying on increasingly powerful servers to handle transaction and database processing and to support massive client/server networks that have replaced the huge mainframe computer centers of yesteryear
- Cloud service providers use massive high-performance banks of servers to satisfy high-volume, high-transaction-rate applications for a broad spectrum of clients

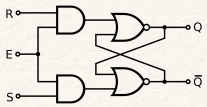


Microprocessor Speed

- Techniques built into contemporary processors include:

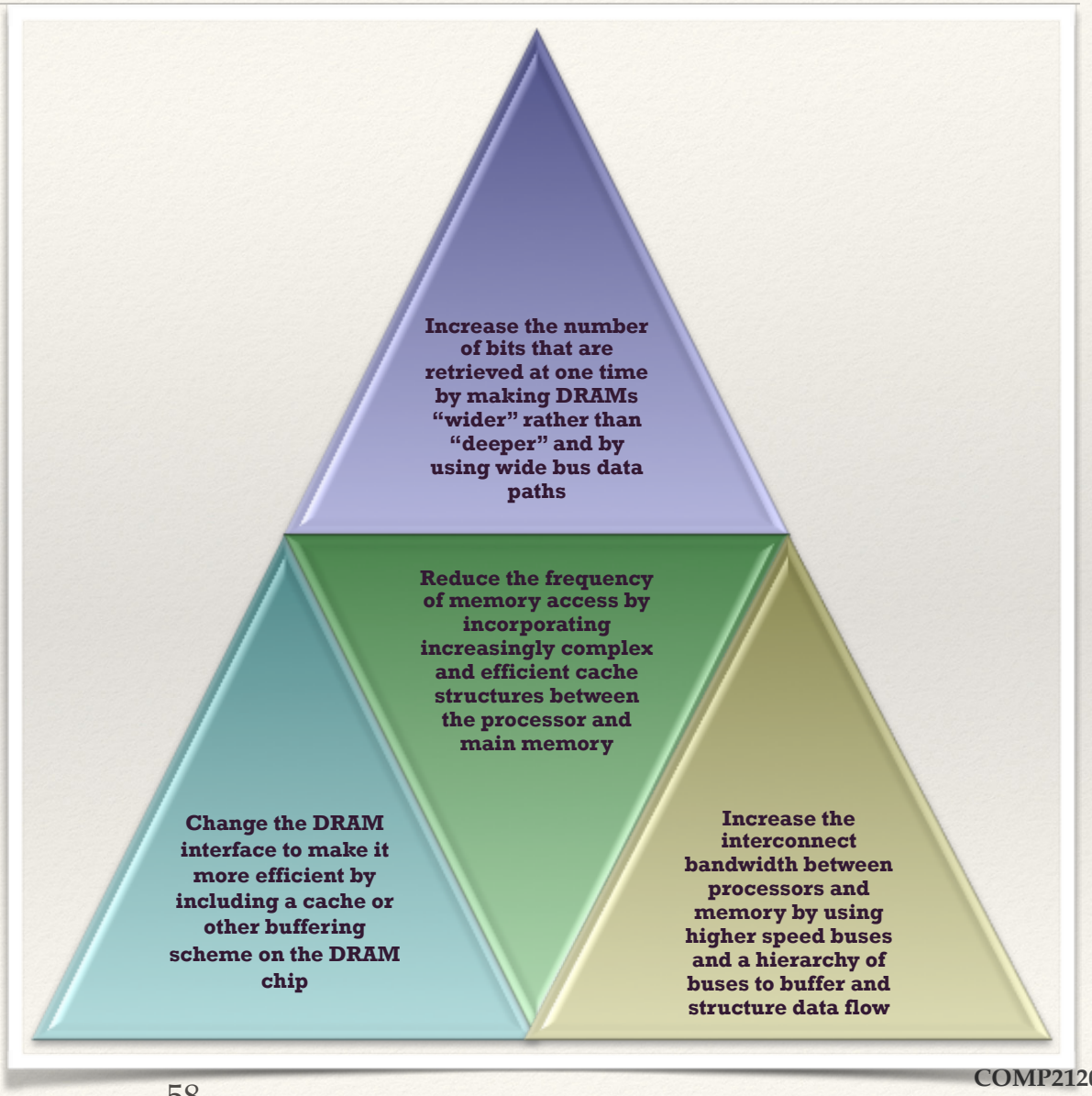


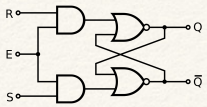
Pipelining	• Processor moves data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously
Branch prediction	• Processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next
Superscalar execution	• This is the ability to issue more than one instruction in every processor clock cycle. (In effect, multiple parallel pipelines are used.)
Data flow analysis	• Processor analyzes which instructions are dependent on each other's results, or data, to create an optimized schedule of instructions
Speculative execution	• Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations, keeping execution engines as busy as possible



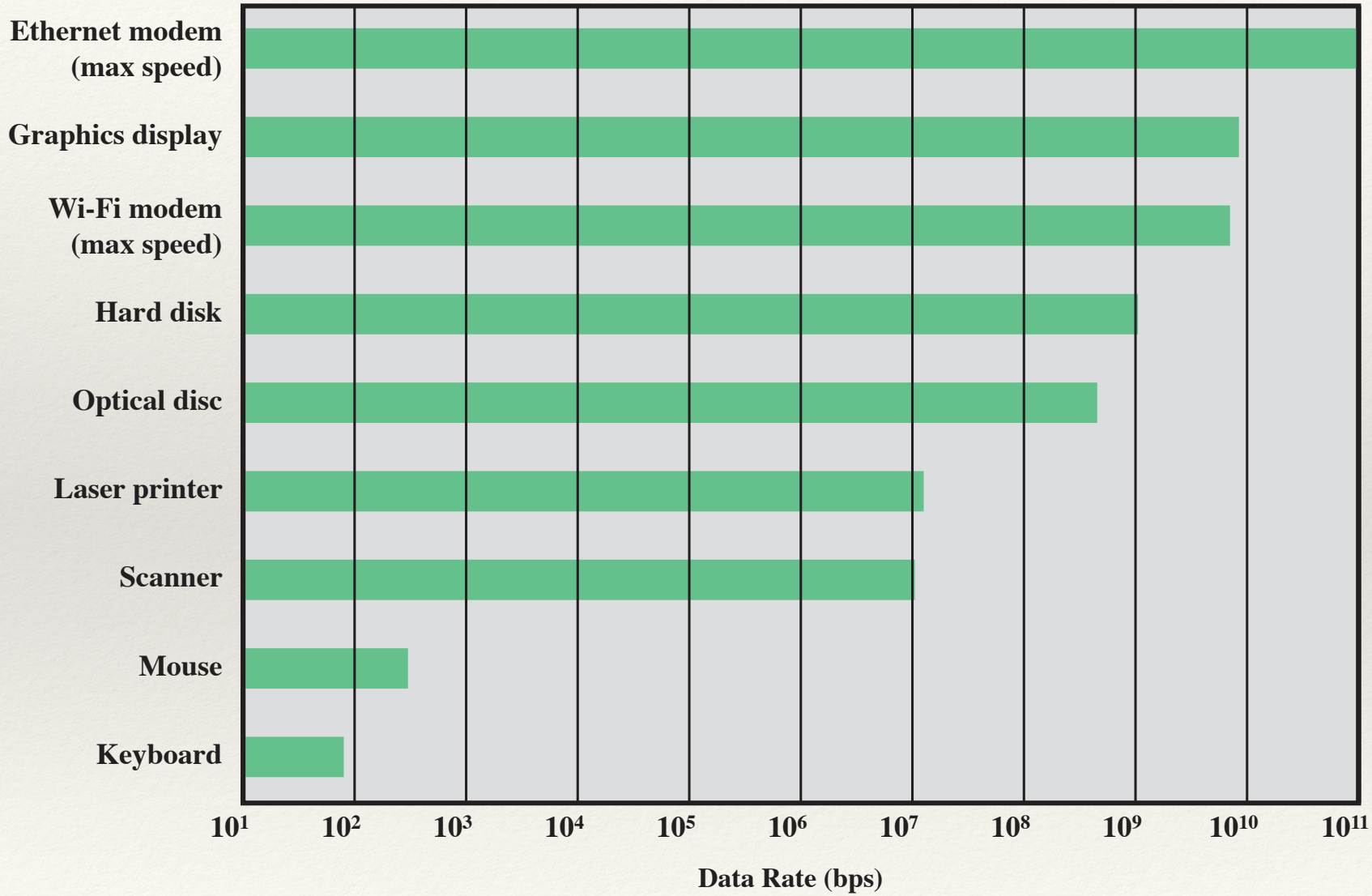
Performance Balance

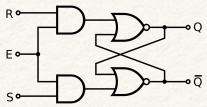
- Adjust the organization and architecture to compensate for the mismatch among the capabilities of the various components
- Architectural examples include:





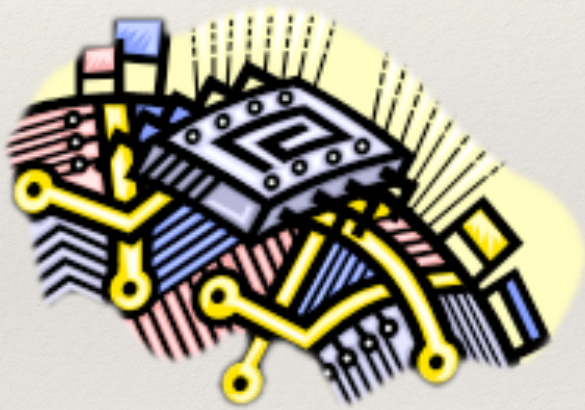
Typical I/O Device Data Rates

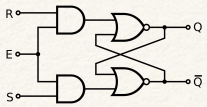




Improvements in Chip Organization and Architecture

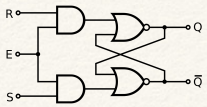
- Increase hardware speed of processor
  - Fundamentally due to shrinking logic gate size
    - More gates, packed more tightly, increasing clock rate
    - Propagation time for signals reduced
- Increase size and speed of caches
  - Dedicating part of processor chip
    - Cache access times drop significantly
- Change processor organization and architecture
  - Increase effective speed of instruction execution
  - Parallelism



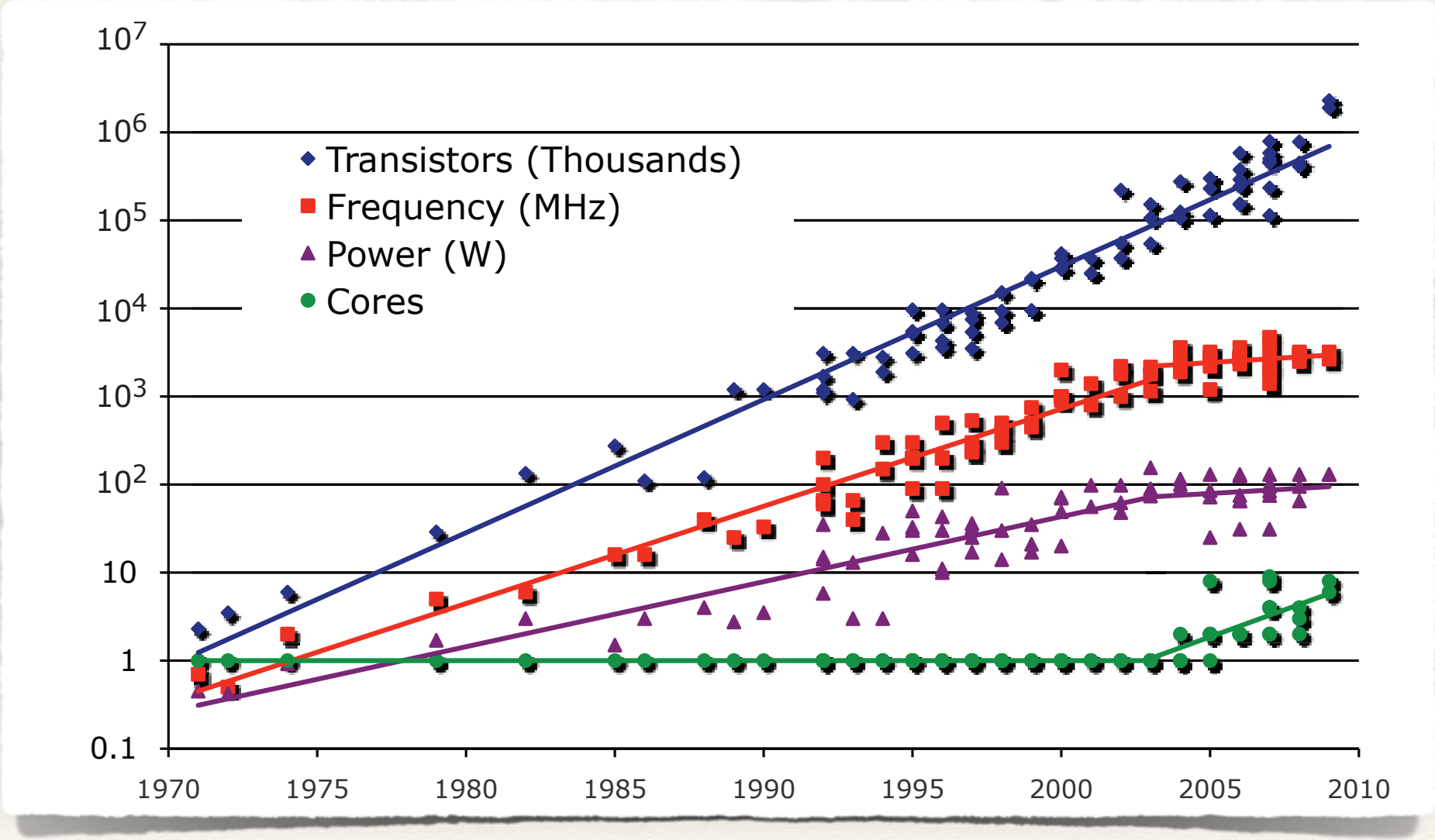


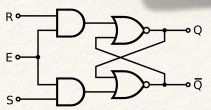
Problems with Clock Speed and Logic Density

- Power
  - Power density increases with density of logic and clock speed
  - Dissipating heat
- RC delay
  - Speed at which electrons flow limited by resistance and capacitance of metal wires connecting them
  - Delay increases as the RC product increases
  - As components on the chip decrease in size, the wire interconnects become thinner, increasing resistance
  - Also, the wires are closer together, increasing capacitance
- Memory latency
  - Memory speeds lag processor speeds

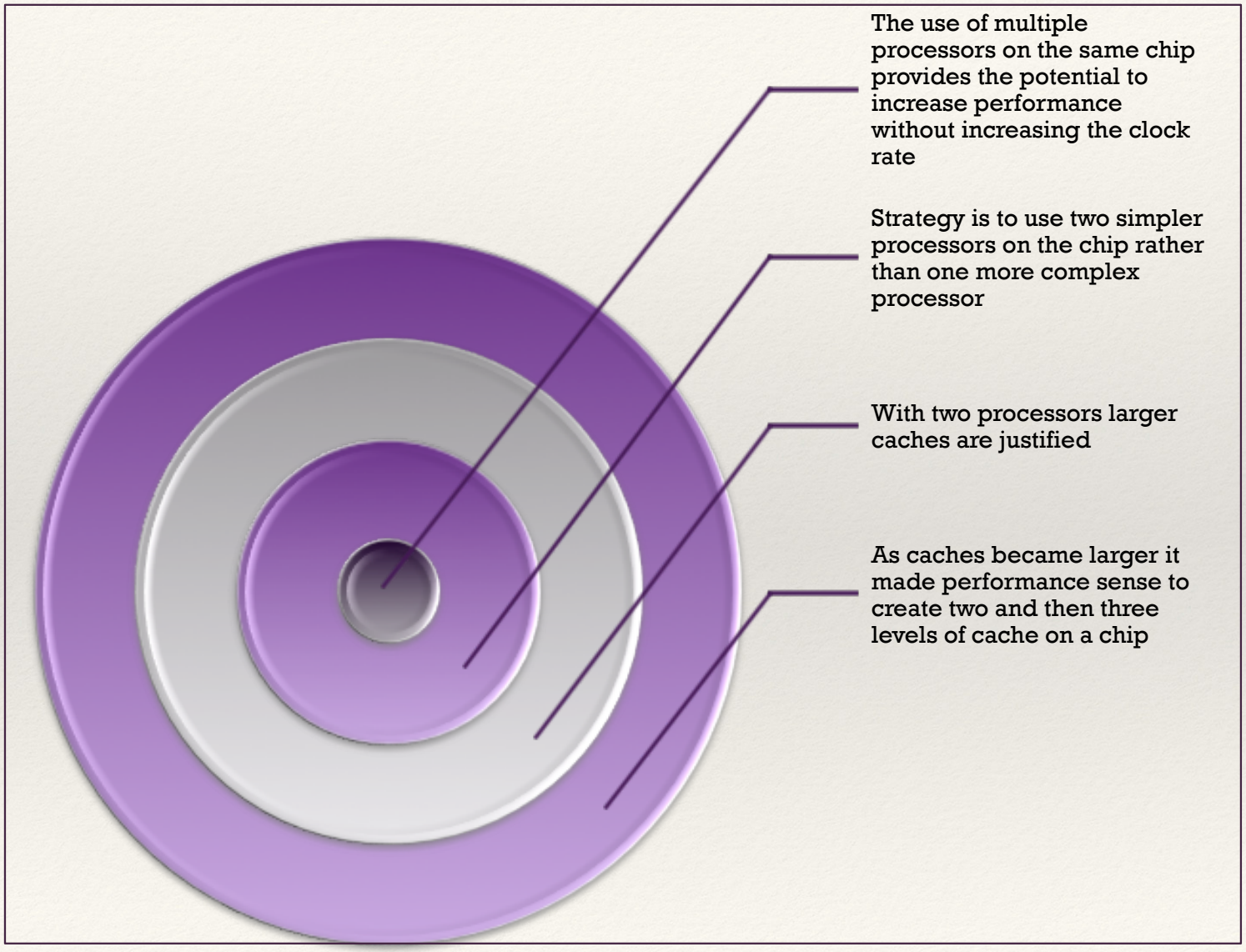


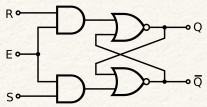
Processor Trends





Multicore

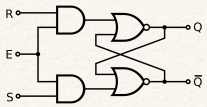

The use of multiple processors on the same chip provides the potential to increase performance without increasing the clock rate
Strategy is to use two simpler processors on the chip rather than one more complex processor
With two processors larger caches are justified
As caches became larger it made performance sense to create two and then three levels of cache on a chip


R
E
S
Q
$\bar{Q}$

Many Integrated Core (MIC)
Graphics Processing Unit (GPU)

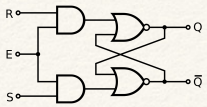
- ***MIC***
  - Leap in performance as well as the challenges in developing software to exploit such a large number of cores
  - The multicore and MIC strategy involves a homogeneous collection of general purpose processors on a single chip

- ***GPU***
  - Core designed to perform parallel operations on graphics data
  - Traditionally found on a plug-in graphics card, it is used to encode and render 2D and 3D graphics as well as process video
  - Used as vector processors for a variety of applications that require repetitive computations



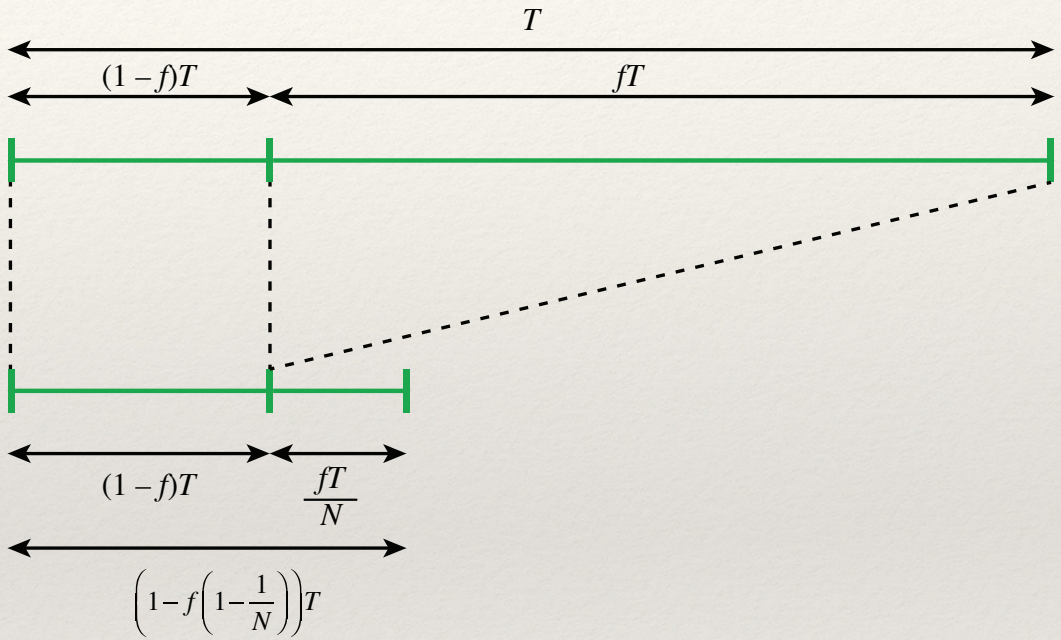
Amdahl's Law

- Gene Amdahl
- Deals with the potential speedup of a program using multiple processors compared to a single processor
- Illustrates the problems facing industry in the development of multi-core machines
  - Software must be adapted to a highly parallel execution environment to exploit the power of parallel processing
- Can be generalized to evaluate and design technical improvement in a computer system



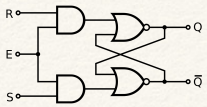
Amdahl's Law (Cont'd)

- Consider a program running on a single processor such that
  - A fraction $(1 - f)$ of the execution time involves code that is inherently serial; and
  - A fraction f that involves code that is infinitely parallelizable with no scheduling overhead.
- Let T be the total execution time of the program using a single processor
- Then the speedup using a parallel processor with N processors that fully exploits the parallel portion of the program is as follows:

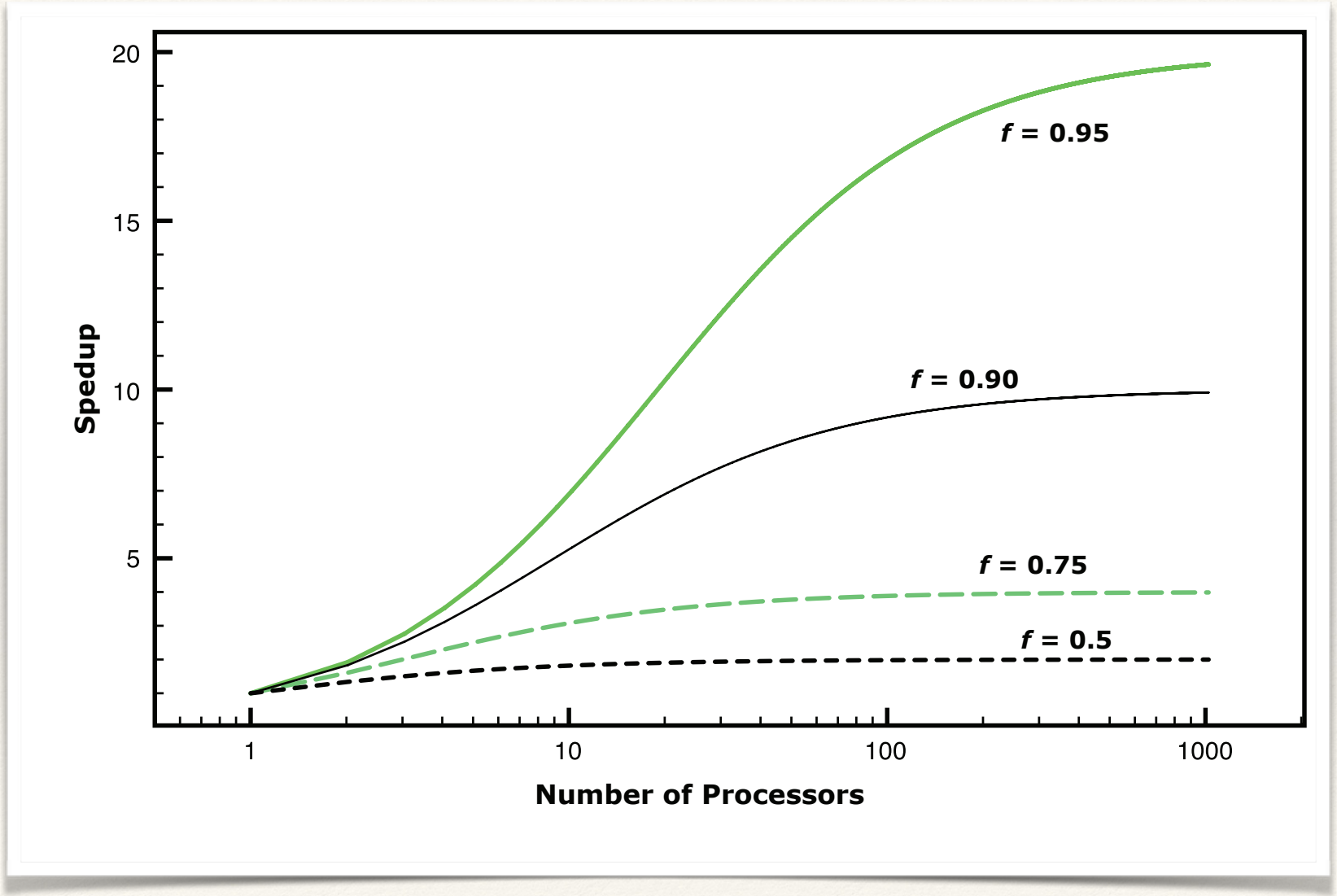


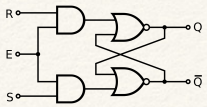
$$speedup = \frac{Time\ to\ execute\ program\ on\ a\ single\ processor}{Time\ to\ execute\ program\ on\ N\ parallelprocessors}$$

$$= \frac{T}{T(1-f) + Tf/N} = \frac{1}{(1-f) + f/N}$$



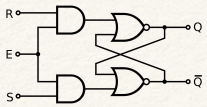
Amdahl's Law for Multiprocessors





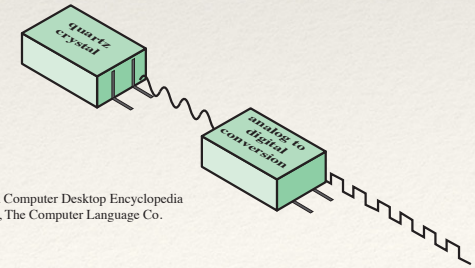
Little's Law

- Fundamental and simple relation with broad applications
- Can be applied to almost any system that is statistically in steady state, and in which there is no leakage
- Queuing system
  - If server is idle an item is served immediately, otherwise an arriving item joins a queue
  - There can be a single queue for a single server or for multiple servers, or multiple queues with one being for each of multiple servers
- Average number of items in a queuing system equals the average rate at which items arrive multiplied by the time that an item spends in the system
  - Relationship requires very few assumptions
  - Because of its simplicity and generality it is extremely useful

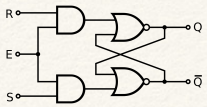


Clock Speed

- Operations performed by a processor are governed by a system clock
  - Instruction Fetch, Instruction Decode, Arithmetic operations, Logical Operations, etc.
- Clock signals are generated by a quartz crystal, which generates a constant sine wave when power is applied
- The wave is then converted into digital voltage pulse stream, the rate of pulses is known as ***clock rate***, or ***clock speed***
  - One pulse of the clock is referred to as a ***clock cycle***, or a ***clock tick***
- All operations begin with the pulse of the clock, the speed of a processor is thus dictated by the purse frequency, measured in cycles per second, or *Hertz*(*Hz*)
- A 1-GHz processor receives 1 billion pulses per second
- Is clock rate a good measurement for processor performance?

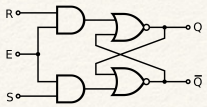


From Computer Desktop Encyclopedia
1998, The Computer Language Co.



Clock Speed (Cont'd)

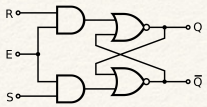
- Clock speed is not a good measure for performance comparison
  - The execution of an instruction involves a number of discrete steps, such as
    - Instruction fetch
    - Instruction decode
    - Loading of Data
    - Storing of Data
    - Arithmetic and Logical operations
  - Most instructions require multiple clock cycles to complete
    - Some require only a few cycles
    - Some may require up to dozens
  - Pipelining also affect the number of instructions that can be executed simultaneously



Instruction Execution Rate

- Consider a processor
  - Driven by a clock with constant cycle time τ (i.e. $\tau = 1/f$ where f is the clock rate)
- Assume a program require I_c number of machine instructions to runs from start to completion
  - Note that I_c refers to the instruction count required for execution, not the number of instructions in the object code of the program
  - The average cycles per instruction (*CPI*) for a program is an important parameter for determining the processor time T required to execute the program
    - $T = I_c \times CPI \times \tau$
- The number of clock cycles required for different instructions is not a constant
  - Load, Store, Branch, Arithmetic/Logical instructions may not consume the same number of clock cycles
  - Let CPI_i be the number of cycles required for instruction type i, and I_i be the number of executed instructions of type i for a given program, then the overall *CPI* can be calculated as follows:

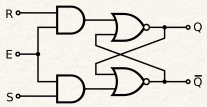
$$CPI = \frac{\sum_{i=1}^{n} (CPI_i \times I_i)}{I_c}$$



Instruction Execution Rate (Cont'd)

- Further refinement
  - Assume p is the number of processor cycles needed to decode and execute the instruction, and m is the number of memory references needed, and k is the ratio between memory cycle time and processor cycle time
  - Then
    - $CPI = p + (m \times k)$
    - $T = I_c \times [p + (m \times k)] \times \tau$
- Thus, (I_c, p, m, k, r) are the key five performance factors that determines execution time for a given program
  - These factors are influenced by four system attributes: ***Instruction Set Architecture***, ***Compiler Technology***, ***Processor Implementation***; and ***Cache and Memory Hierarchy***

	I_c	p	m	k	τ
Instruction set architecture	X	X			
Compiler technology	X	X	X		
Processor implementation		X			X
Cache and memory hierarchy				X	X



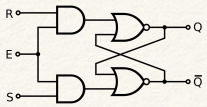
Instruction Execution Rate (Cont'd)

- A common measure of performance for a processor is the rate at which instructions are executed, expressed as millions of instructions per second (MIPS), referred to as the ***MIPS rate***.
  - $MIPS\ rate = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$

  Remark: $T = I_c \times CPI \times \tau \Rightarrow I_c/T = (1/\tau)/(CPI)$
- Example
  - Consider the execution of a program that results in execution of 2 million instructions on a 400-Mhz processor, with the following mix of instructions and CPI for each instruction type

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

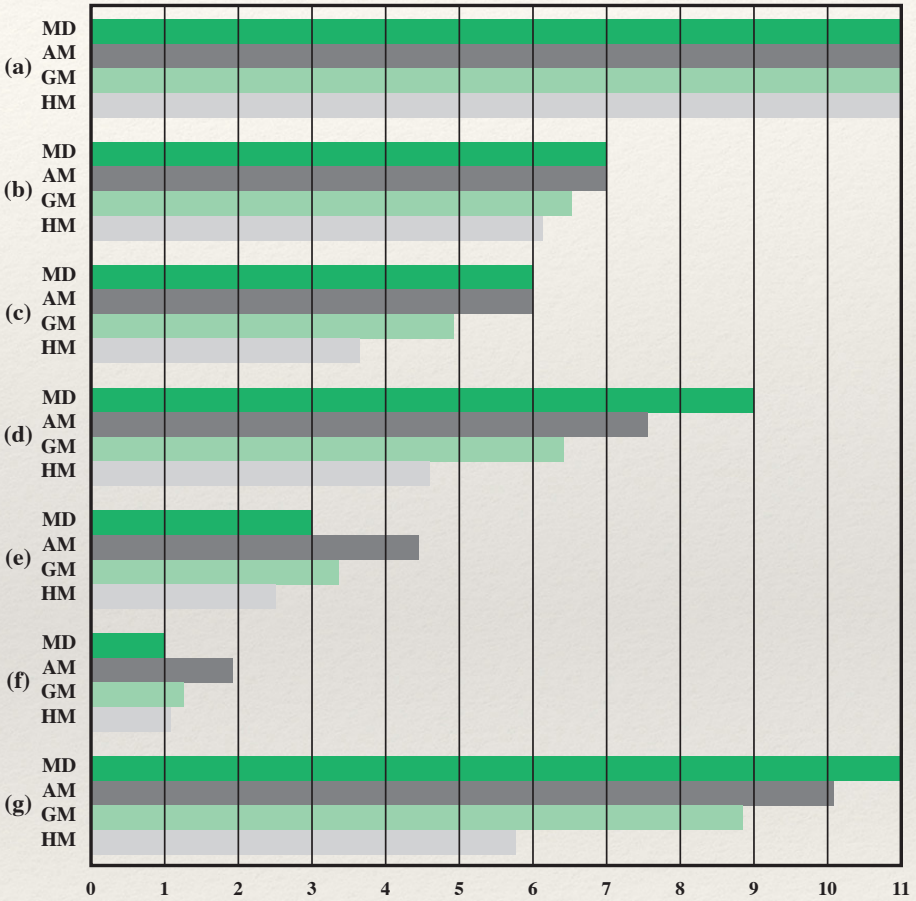
  - The average CPI when the program is executed on a uniprocessor is
    - $CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$
    - The corresponding MIPS rate is $(400 \times 10^6) / (2.24 \times 10^6) \approx 178$
- Another common performance measure deals only with floating-point instruction
  - i.e. Millions of Floating-Point Operations per second (MFLOPS)



Benchmarks

- It is common to use one signal number to denote computer performance
  - How to obtain this single number?
- The use of benchmarks to compare systems involves calculating the mean value of a set of data points related to execution time
- Three common formulas used for calculating mean:
  - Arithmetic
    - $$AM = \frac{1}{n} \sum_{i=1}^{n} x_i$$
  - Geometric
    - $$GM = \left(\prod_{i=1}^{n} x_i \right)^{\frac{1}{n}} = e^{\left(\frac{1}{n} \sum_{i=1}^{n} \ln(x_i) \right)}$$
  - Harmonic

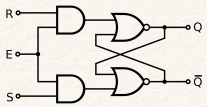
$$HM = \frac{n}{\sum_{i=1}^{n} \left(\frac{1}{x_i} \right)}, \quad x_i > 0$$



(a) Constant (11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)
(b) Clustered around a central value (3, 5, 6, 6, 7, 7, 7, 8, 8, 9, 11)
(c) Uniform distribution (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
(d) Large-number bias (1, 4, 4, 7, 7, 9, 9, 10, 10, 11, 11)
(e) Small-number bias(1, 1, 2, 2, 3, 3, 5, 5, 8, 8, 11)
(f) Upper outlier (11, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
(g) Lower outlier (1, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)

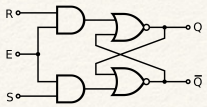
MD = median
AM = arithmetic mean
GM = geometric mean
HM = harmonic mean

$$AM \geqq GM \geqq HM$$



Which mean formula to use?

- Arithmetic Mean
  - AM is an appropriate measure if the sum of all the measurements is a meaningful and interesting value
  - Good candidate for comparing the execution time performance of several systems
  - It has an important property that it is directly proportional to the total time
    - If total time doubles, the mean value doubles
- Harmonic Mean
  - Consider we would like to compare performance based on the execution rates (e.g. MIPS or MFLOPS), does it make any sense to calculate the average rate?
    - The sum of rates would be a meaningless statistic
    - We prefer the mean to be inversely proportional to the total execution time

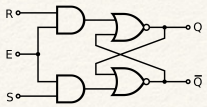


AM vs HM

- Suppose we have a set of n benchmark programs on a given system
  - Each program executes the same number of operations Z;
  - The execution times of each program on the system are $t_1, t_2, \ldots, t_n$.
- Then, the execution rate for each individual program is thus $R_i = Z / t_i$.

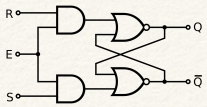
$$AM = \frac{1}{n} \sum_{i=1}^{n} R_i = \frac{1}{n} \sum_{i=1}^{n} \frac{Z}{t_i} = \frac{Z}{n} \sum_{i=1}^{n} \frac{1}{t_i}$$

$$HM = \frac{n}{\sum_{i=1}^{n} \frac{1}{R_i}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{Z/t_i}} = nZ \left(\frac{1}{\sum_{i=1}^{n} t_i} \right)$$



A Comparison of AM and HM Rates

	Computer A time (secs)	Computer B time (secs)	Computer C time (secs)	Computer A rate (MFLOPS)	Computer B rate (MFLOPS)	Computer C rate (MFLOPS)
Program 1 (10^8 FP ops)	2.0	1.0	0.75	50	100	133.33
Program 2 (10^8 FP ops)	0.75	2.0	4.0	133.33	50	25
Total execution time	2.75	3.0	4.75			
Arithmetic mean of times	1.38	1.5	2.38			
Inverse of total execution time (1/sec)	0.36	0.33	0.21			
Arithmetic mean of rates	(100M+100M)/2.75 = 72.72			91.67	75.00	79.17
Harmonic mean of rates				72.72	66.67	42.11



What about GM?

- One property of the GM that has made it appealing for benchmark analysis is that it provides consistent results when measuring the relative performance

$$GM = \left(\prod_{i=1}^{n} \frac{Z_i}{t_i} \right)^{1/n} = \frac{\left(\prod_{i=1}^{n} Z_i \right)^{1/n}}{\left(\prod_{i=1}^{n} t_i \right)^{1/n}}$$

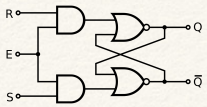
Table 2.3 A Comparison of Arithmetic and Geometric Means for Normalized Results

(a) Results normalized to Computer A

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (1.0)	1.0 (0.5)	0.75 (0.38)
Program 2	0.75 (1.0)	2.0 (2.67)	4.0 (5.33)
Total execution time	2.75	3.0	4.75
Arithmetic mean of normalized times	1.00 AM_A	1.58 $\leqq$ AM_B $\leqq$	2.85 AM_C
Geometric mean of normalized times	1.00 GM_A	1.15 $\leqq$ GM_B $\leqq$	1.41 GM_C

(b) Results normalized to Computer B

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (2.0)	1.0 (1.0)	0.75 (0.75)
Program 2	0.75 (0.38)	2.0 (1.0)	4.0 (2.0)
Total execution time	2.75	3.0	4.75
Arithmetic mean of normalized times	1.19 AM_A	1.00 $\geqq$ AM_B $\leqq$	1.38 AM_C
Geometric mean of normalized times	0.87 GM_A	1.00 $\leqq$ GM_B $\leqq$	1.22 GM_C



GM Measurement

Table 2.4 Another Comparison of Arithmetic and Geometric Means for Normalized Results

(a) Results normalized to Computer A

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (1.0)	1.0 (0.5)	0.20 (0.1)
Program 2	0.4 (1.0)	2.0 (5.0)	4.0 (10)
Total execution time	2.4	3.00	4.2
Arithmetic mean of normalized times	1.00 AM_A	2.75 AM_B	5.05 AM_C
Geometric mean of normalized times	1.00 GM_A	1.58 GM_B	1.00 GM_C

$AM_A \leqq AM_B \leqq AM_C$

$GM_A \leqq GM_B \geqq GM_C$

But $GM_A == GM_C$??

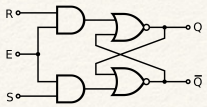
(b) Results normalized to Computer B

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (2.0)	1.0 (1.0)	0.20 (0.2)
Program 2	0.4 (0.2)	2.0 (1.0)	4.0 (2)
Total execution time	2.4	3.00	4.2
Arithmetic mean of normalized times	1.10 AM_A	1.00 AM_B	1.10 AM_C
Geometric mean of normalized times	0.63 GM_A	1.00 GM_B	0.63 GM_C

$AM_A \geqq AM_B \leqq AM_C$

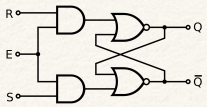
$GM_A \leqq GM_B \geqq GM_C$

But $GM_A == GM_C$??



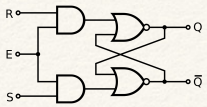
Benchmark Principles

- Desirable characteristics of a benchmark program:
  1. It is written in a high-level language, making it portable across different machines
  2. It is representative of a particular kind of programming domain or paradigm, such as systems programming, numerical programming, or commercial programming
  3. It can be measured easily
  4. It has wide distribution



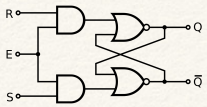
Benchmark Principles (Cont'd)

- Is MIPS or MFLOPS a good performance indicator?
  - Instruction Rate is not a valid means for comparing the performance of different architectures due to differences in instruction sets
  - Consider the high-level language statement: A = B + C
    - On complex instruction set computer (CISC), it can be achieved by one single instruction
      - add mem(B), mem(C), mem(A)
    - On RISC machine, the list of instructions to achieve the same could be
      - load mem(B), reg(1);
      - load mem(C), reg(2);
      - add reg(1), reg(2), reg(3);
      - store reg(3), mem(A)



SPEC Benchmarks

- A benchmark suite is a collection of programs, defined in a high-level language, that together attempt to provide a representative test of a computer in a particular application or system programming area.
- The best known collection of benchmark suites is defined and painted by the Standard Performance Evaluation Corporation (SPEC), an industry consortium
  - Widely used for comparison and research purpose
- SPEC has chosen to use the GM for performance comparison for the following reasons
  - GM gives consistent results regardless of which system is used as a reference
  - GM is less biased by outliers than the HM or AM
  - Distributions of performance ratios are better modelled by lognormal distribution



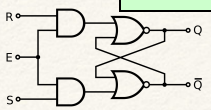
SPEC CPU2006

Integer Benchmarks

Benchmark	Reference time (hours)	Instr count (billion)	Language	Application Area	Brief Description
400.perlbench	2.71	2,378	C	Programming Language	PERL programming language interpreter, applied to a set of three programs.
401.bzip2	2.68	2,472	C	Compression	General-purpose data compression with most work done in memory, rather than doing I/O.
403.gcc	2.24	1,064	C	C Compiler	Based on gcc Version 3.2, generates code for Opteron.
429.mcf	2.53	327	C	Combinatorial Optimization	Vehicle scheduling algorithm.
445.gobmk	2.91	1,603	C	Artificial Intelligence	Plays the game of Go, a simply described but deeply complex game.
456.hmmer	2.59	3,363	C	Search Gene Sequence	Protein sequence analysis using profile hidden Markov models.
458.sjeng	3.36	2,383	C	Artificial Intelligence	A highly ranked chess program that also plays several chess variants.
462.libquantum	5.76	3,555	C	Physics / Quantum Computing	Simulates a quantum computer, running Shor's polynomial-time factorization algorithm.
464.h264ref	6.15	3,731	C	Video Compression	H.264/AVC (Advanced Video Coding) Video compression.
471.omnetpp	1.74	687	C++	Discrete Event Simulation	Uses the OMNet++ discrete event simulator to model a large Ethernet campus network.
473.astar	1.95	1,200	C++	Path-finding Algorithms	Pathfinding library for 2D maps.
483.xalancbmk	1.92	1,184	C++	XML Processing	A modified version of Xalan-C++, which transforms XML documents to other document types.

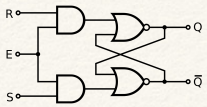
Floating-Point Benchmarks

Benchmark	Reference time (hours)	Instr count (billion)	Language	Application Area	Brief Description
410.bwaves	3.78	1,176	Fortran	Fluid Dynamics	Computes 3D transonic transient laminar viscous flow.
416.gamess	5.44	5,189	Fortran	Quantum Chemistry	Quantum chemical computations.
433.milc	2.55	937	C	Physics / Quantum Chromodynamics	Simulates behavior of quarks and gluons
434.zeusmp	2.53	1,566	Fortran	Physics / CFD	Computational fluid dynamics simulation of astrophysical phenomena.
435.gromacs	1.98	1,958	C, Fortran	Biochemistry / Molecular Dynamics	Simulate Newtonian equations of motion for hundreds to millions of particles.
436.cactusADM	3.32	1,376	C, Fortran	Physics / General Relativity	Solves the Einstein evolution equations.
437.leslie3d	2.61	1,273	Fortran	Fluid Dynamics	Model fuel injection flows.
444.namd	2.23	2,483	C++	Biology / Molecular Dynamics	Simulates large biomolecular systems.
447.dealII	3.18	2,323	C++	Finite Element Analysis	Program library targeted at adaptive finite elements and error estimation.
450.soplex	2.32	703	C++	Linear Programming, Optimization	Test cases include railroad planning and military airlift models.
453.povray	1.48	940	C++	Image Ray-tracing	3D Image rendering.
454.calculix	2.29	3,04`	C, Fortran	Structural Mechanics	Finite element code for linear and nonlinear 3D structural applications.
459.GemsFDTD	2.95	1,320	Fortran	Computational Electromagnetics	Solves the Maxwell equations in 3D.
465.tonto	2.73	2,392	Fortran	Quantum Chemistry	Quantum chemistry package, adapted for crystallographic tasks.
470.lbm	3.82	1,500	C	Fluid Dynamics	Simulates incompressible fluids in 3D.
481.wrf	3.10	1,684	C, Fortran	Weather	Weather forecasting model
482.sphinx3	5.41	2,472	C	Speech recognition	Speech recognition software.



Terms used in SPEC Documentation

- Benchmark
  - A program written in a high-level language that can be compiled and executed on any computer that implements the compiler
- System under test
  - This is the system to be evaluated
- Reference machine
  - This is a system used by SPEC to establish a baseline performance for all benchmarks
    - Each benchmark is run and measured on this machine to establish a reference time for that benchmark
- Base metric
  - These are required for all reported results and have strict guidelines for compilation
- Peak metric
  - This enables users to attempt to optimize system performance by optimizing the compiler output
- Speed metric
  - This is simply a measurement of the time it takes to execute a compiled benchmark
    - Used for comparing the ability of a computer to complete single tasks
- Rate metric
  - This is a measurement of how many tasks a computer can accomplish in a certain amount of time
    - This is called a throughput, capacity, or rate measure
    - Allows the system under test to execute simultaneous tasks to take advantage of multiple processors



SPEC Evaluation Procedure

- System under test shall run each program on the system three times
  - The median value is selected to account for variations in execution time that are not intrinsic to the program
    - e.g. disk access, OS kernel execution variations etc.
- Each of the program results is normalised by calculating the runtime ratio of the reference run time to the system run time
- GM of all the programs runtime ratios is calculated to yield the overall metric

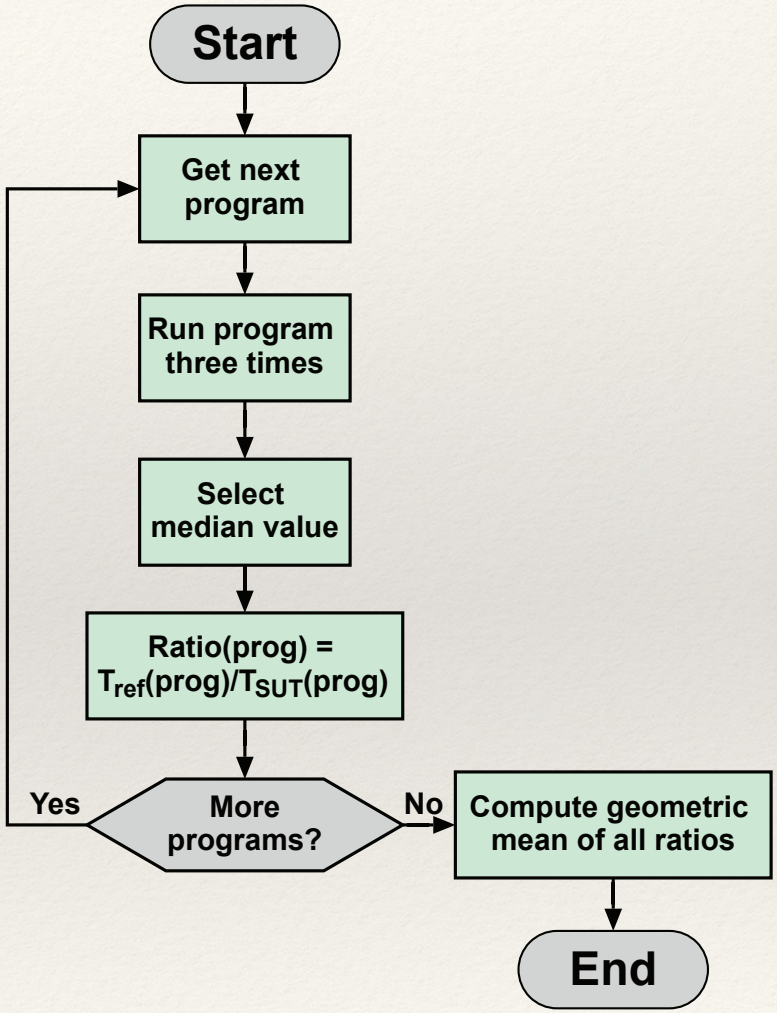


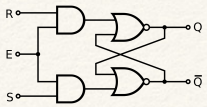
Figure 2.7 SPEC Evaluation Flowchart

Example SPEC CINT2006 Results

Table 2.7 Some SPEC CINT2006 Results

(a) Sun Blade 1000

Benchmark	Execution time	Execution time	Execution time	Reference time	Ratio
400.perlbench	**3077**	3076	3080	9770	3.18
401.bzip2	**3260**	3263	3260	9650	2.96
403.gcc	2711	2701	**2702**	8050	2.98
429.mcf	2356	**2331**	2301	9120	3.91
445.gobmk	3319	**3310**	3308	10490	3.17
456.hmmer	2586	**2587**	2601	9330	3.61
458.sjeng	3452	**3449**	3449	12100	3.51
462.libquantum	**10318**	10319	10273	20720	2.01
464.h264ref	5246	5290	**5259**	22130	4.21
471.omnetpp	2565	**2572**	2582	6250	2.43
473.astar	2522	**2554**	2565	7020	2.75
483.xalancbmk	2014	2018	**2018**	6900	3.42

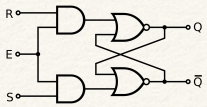


Example SPEC CINT2006 Results (Cont'd)

Table 2.7 Some SPEC CINT2006 Results

(b) Sun Blade X6250

Benchmark	Execution time	Execution time	Execution time	Reference time	Ratio	Rate
400.perlbench	497	497	**497**	9770	19.66	78.63
401.bzip2	613	614	**613**	9650	15.74	62.97
403.gcc	529	**529**	529	8050	15.22	60.87
429.mcf	**472**	472	473	9120	19.32	77.29
445.gobmk	637	637	**637**	10490	16.47	65.87
456.hmmer	446	446	**446**	9330	20.92	83.68
458.sjeng	**631**	632	630	12100	19.18	76.70
462.libquantum	**614**	614	614	20720	33.75	134.98
464.h264ref	830	**830**	830	22130	26.66	106.65
471.omnetpp	619	620	**619**	6250	10.10	40.39
473.astar	580	580	**580**	7020	12.10	48.41
483.xalancbmk	**422**	422	422	6900	16.35	65.40



Summary

- Chapter 2 - Performance Issues
  - Designing for performance
    - Microprocessor speed
    - Performance balance
    - Improvements in chip organization and architecture
  - Multicore
  - MICs
  - GPGPUs
  - Amdahl's Law
  - Little's Law
  - Basic measures of computer performance
    - Clock speed
    - Instruction execution rate
  - Calculating the mean
    - Arithmetic mean
    - Harmonic mean
    - Geometric mean
  - Benchmark principles
  - SPEC benchmarks