Akdeniz University
Computer Engineering Department

CSE206 Computer Organization

Week02: Computer Evolution

Assoc.Prof.Dr. Taner Danışman

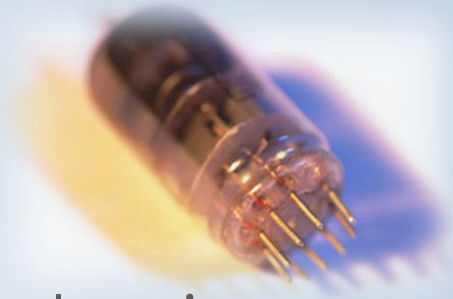
tdanisman@akdeniz.edu.tr

Course program (Textbook: Stalling 10th Edt.)

Week 1	10-Feb-25	Introduction	Ch1
Week 2	17-Feb-25	Computer Evolution	Ch2
Week 3	24-Feb-25	Computer Systems	Ch3
Week 4	3-Mar-25	Cache Memory, Direct Cache Mapping	Ch4
Week 5	10-Mar-25	Associative and Set Associative Mapping	Ch4
Week 6	17-Mar-25	Internal Memory, External Memory, I/O	Ch5-Ch6-Ch7
Week 7	24-Mar-25	Number Systems, Computer Arithmetic	Ch9-Ch10
Week 8	31-Mar-25	Midterm (Expected date, may change)	Ch1-...-Ch10
Week 9	7-Apr-25	Digital Logic	Ch11
Week 10	14-Apr-25	Instruction Sets	Ch12
Week 11	21-Apr-25	Addressing Modes	Ch13
Week 12	28-Apr-25	Processor Structure and Function	Ch14
Week 13	5-May-25	RISC, Inctruction Level Parallelism	Ch15-Ch16
Week 14	12-May-25	Assembly Language (TextBook : Assembly Language for x86 Processors)	Kip Irvine
Week 15	19-May-25	Assembly Language (TextBook : Assembly Language for x86 Processors)	Kip Irvine

History of Computers

First Generation: Vacuum Tubes



- ENIAC
  - Electronic Numerical Integrator And Computer
- Designed and constructed at the University of Pennsylvania
  - Started in 1943 – completed in 1946
  - By John Mauchly and John Eckert
- World's first general purpose electronic digital computer
  - Army's Ballistics Research Laboratory (BRL) needed a way to supply trajectory tables for new weapons accurately and within a reasonable time frame
  - Was not finished in time to be used in the war effort
  - Its first task was to perform a series of calculations that were used to help determine the feasibility of the hydrogen bomb

ENIAC Electronic Numerical Integrator And Computer

- Weighed 30 tons
- Occupied 1500 square feet of floor space
- Contained more than 18,000 vacuum tubes
- 140 kW Power consumption
- Capable of 5000 additions per second
- Decimal rather than binary machine
- Memory consisted of 20 accumulators, each capable of holding a 10 digit number
- Major drawback was the need for manual programming by setting switches and plugging/ unplugging cables

John von Neumann

EDVAC (Electronic Discrete Variable Computer)

- First publication of the idea was in 1945
- Stored program concept
  - Attributed to ENIAC designers, most notably the mathematician John von Neumann
- IAS computer
  - Princeton **I**nstitute for **A**dvanced **S**tudies
  - Prototype of all subsequent general-purpose computers
  - Completed in 1952

Structure of von Neumann Machine

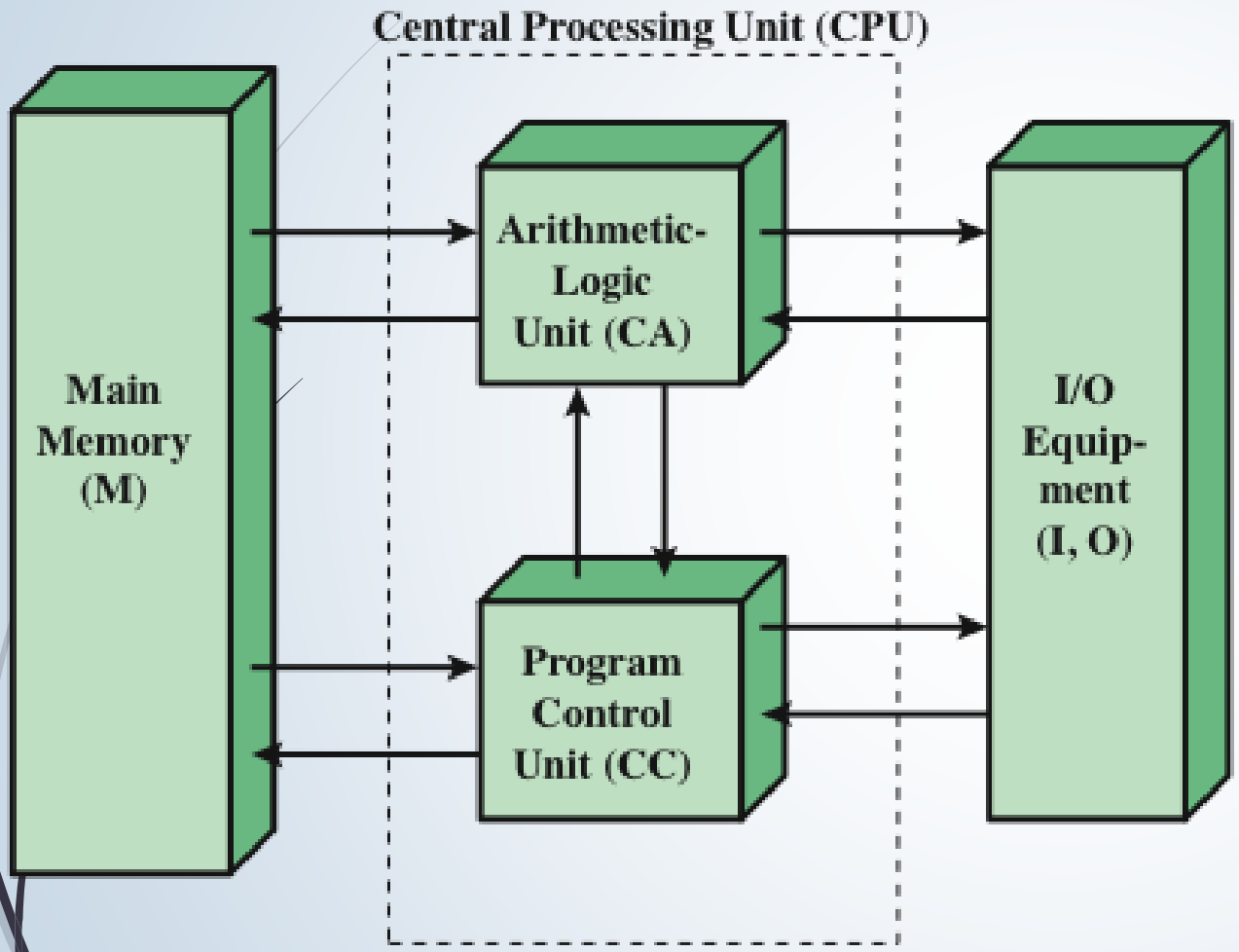


Figure 2.1 Structure of the IAS Computer

- A **main memory,** which stores both data and instructions

- An **arithmetic and logic unit (ALU)** capable of operating on binary data

- A **control unit,** which interprets the instructions in memory and causes them to be executed

- **Input/output (I/O)** equipment operated by the control unit

IAS Memory Formats

- Both data and instructions are stored there
- Numbers are represented in binary form and each instruction is a binary code

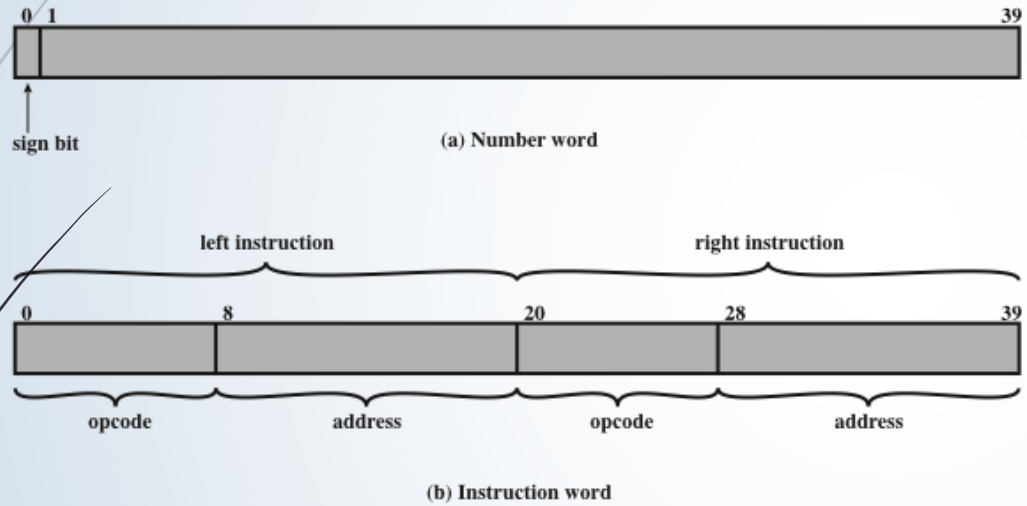


Figure 2.2 IAS Memory Formats

- The memory of the IAS consists of 1000 storage locations (called ***words***) of 40 bits each
- Numbers are represented in binary form, and each instruction is a binary code
- A word may also contain two 20-bit instructions, with each instruction consisting of an 8-bit operation code (**opcode**) specifying the operation to be performed and a 12-bit address designating one of the words in memory (numbered from 0 to 999)

There is no universal definition of the term **word**. In general, a word is an ordered set of bytes or bits that is the normal unit in which information may be stored, transmitted, or operated on within a given computer. Typically, if a processor has a fixed-length instruction set, then the instruction length equals the word length.

Structure of IAS Computer

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

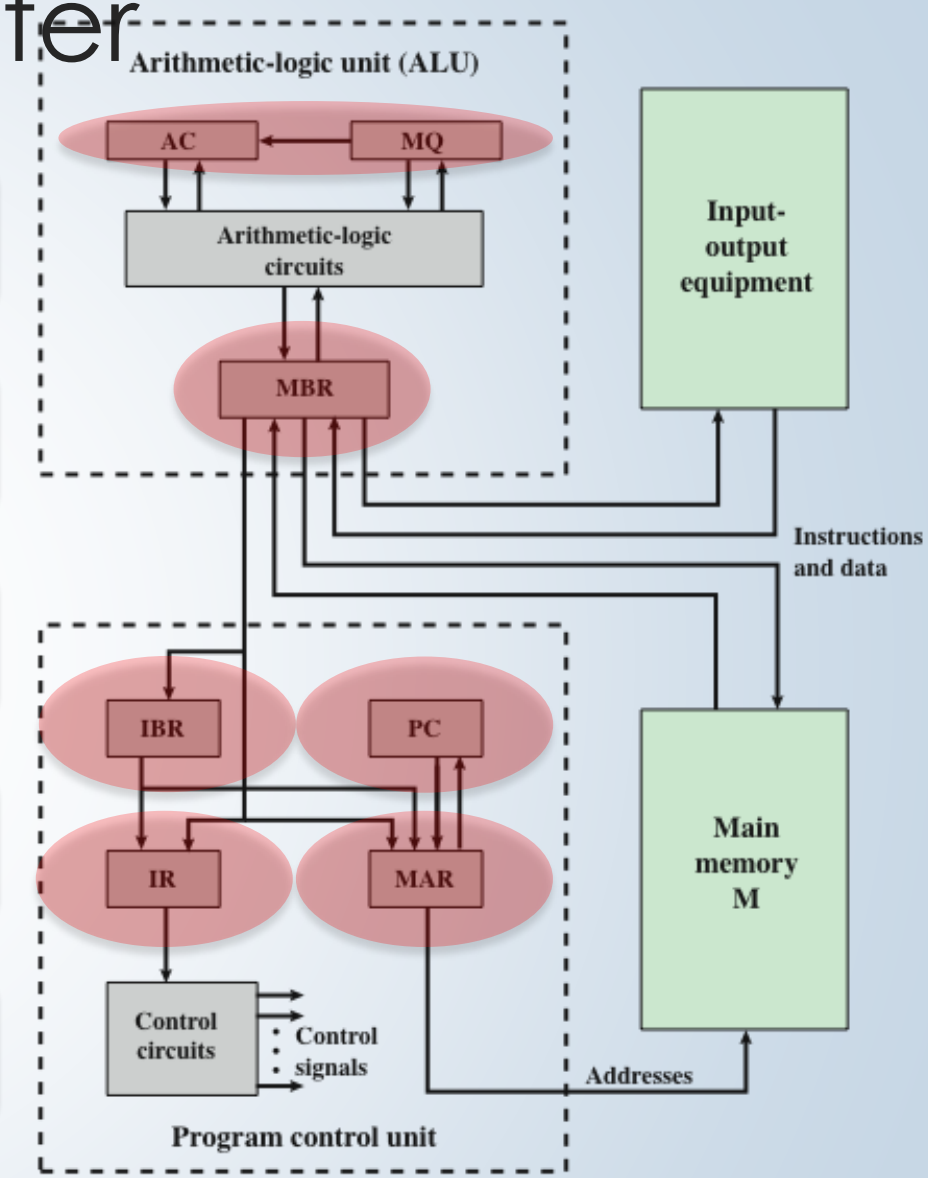


Figure 2.3 Expanded Structure of IAS Computer

IAS Operations

- The IAS operates by repetitively performing an **instruction cycle**
- Each instruction cycle consists of two subcycles
- During the **fetch cycle**, the **opcode** of the **next instruction** is loaded into the **IR** and the **address portion** is loaded into the **MAR**.
- Once the **opcode** is in the **IR**, the **execute cycle** is performed
- The IAS computer had a total of 21 instructions,

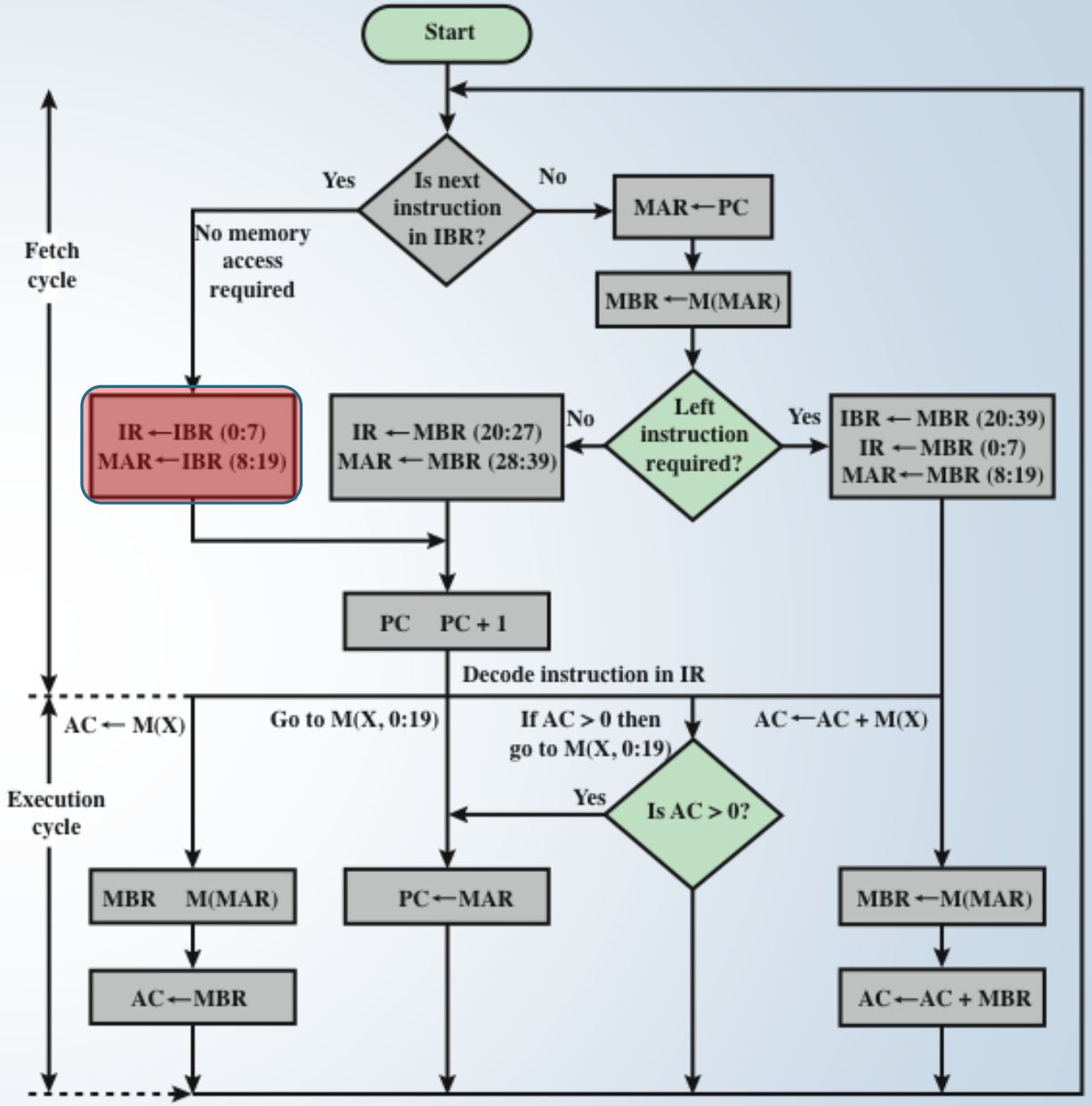


Figure 2.4 Partial Flowchart of IAS Operation

The IAS Instruction Set

Data transfer: Move data between memory and ALU registers or between two ALU registers.

Unconditional branch: Normally, the control unit executes instructions in sequence from memory. This sequence can be changed by a branch instruction, which facilitates repetitive operations.

Conditional branch: The branch can be made dependent on a condition, thus allowing decision points.

Arithmetic: Operations performed by the ALU.

Address modify: Permits addresses to be computed in the ALU and then inserted into instructions stored in memory. This allows a program considerable addressing flexibility.

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)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)

Instruction Type	Opcode	Symbolic Representation	Description
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

Machine Instruction Types

- Data Transfer: copy data from one location to another
- Arithmetic/Logic: use existing bit patterns to compute a new bit patterns
- Control: direct the execution of the program

Adding values stored in memory

Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop.

Figure 2.3 Dividing values stored in memory

Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

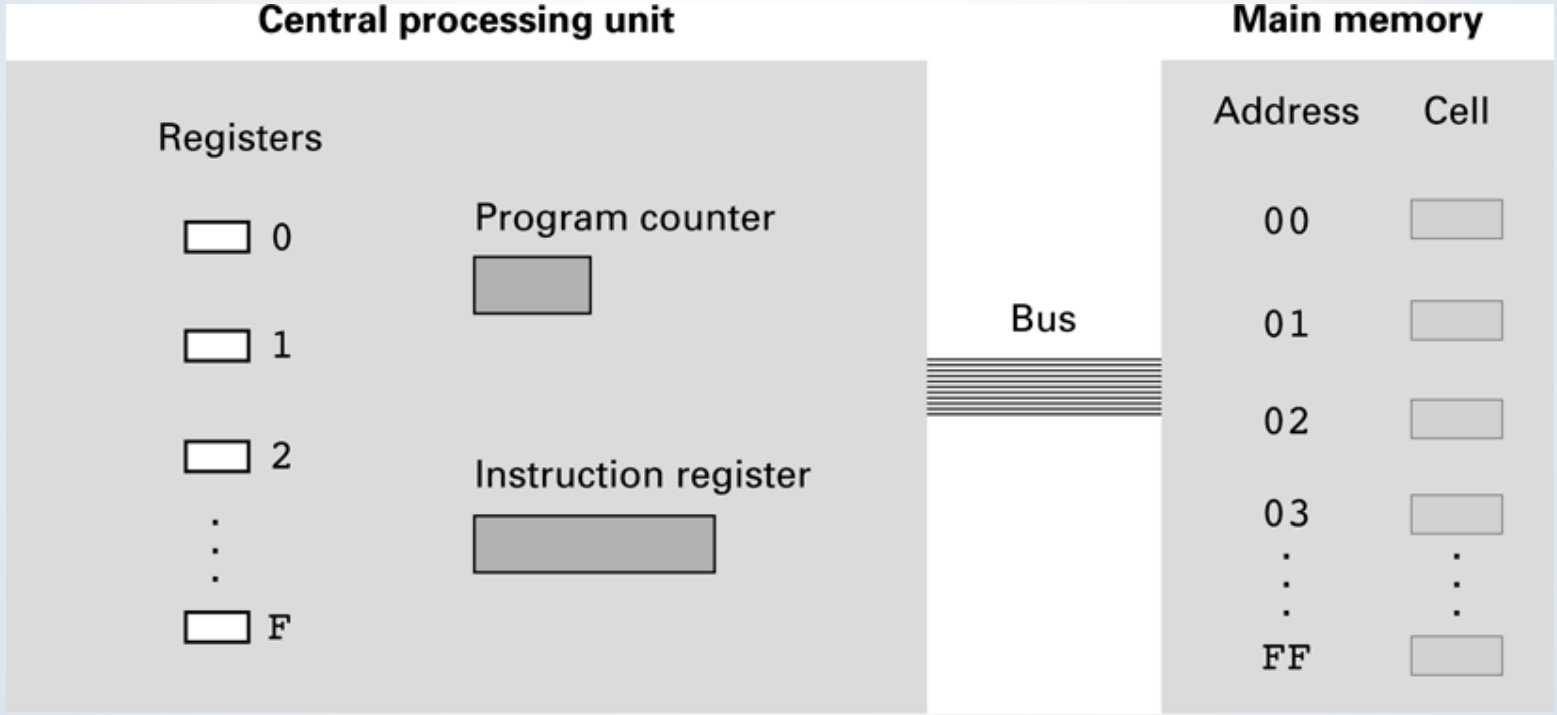
Step 3. If this second value is zero, JUMP to Step 6.

Step 4. Divide the contents of the first register by the second register and leave the result in a third register.

Step 5. STORE the contents of the third register in memory.

Step 6. STOP.

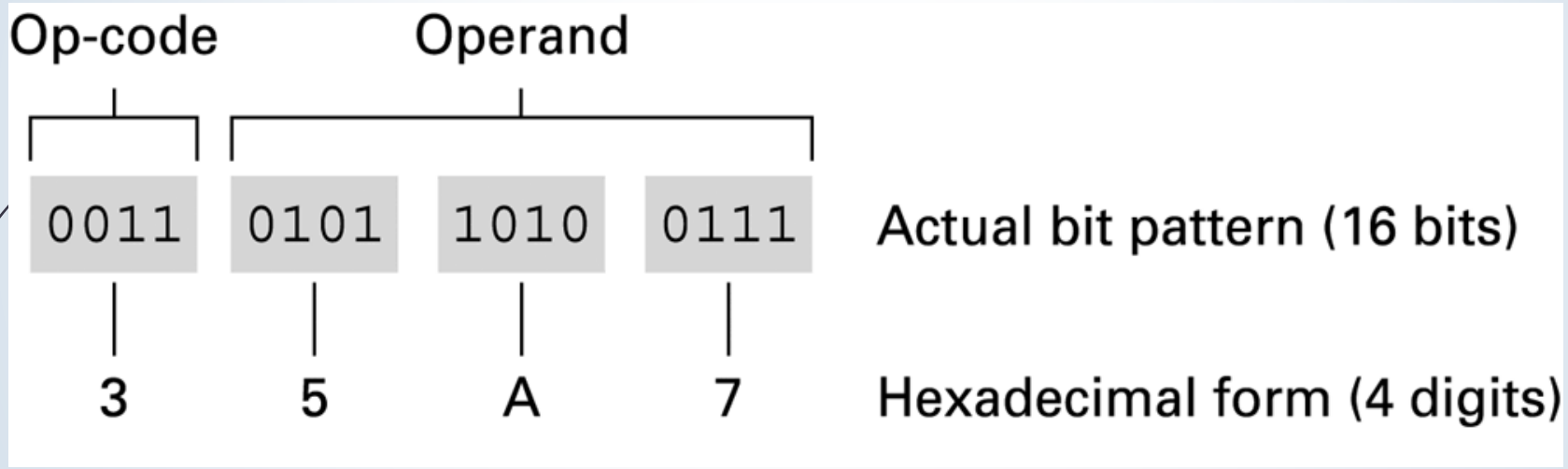
The architecture of the machine described in Appendix C



Parts of a Machine Instruction

- **Op-code:** Specifies which operation to execute
- **Operand:** Gives more detailed information about the operation
  - Interpretation of operand varies depending on op-code

The composition of an instruction for the machine in Appendix C



Instruction set

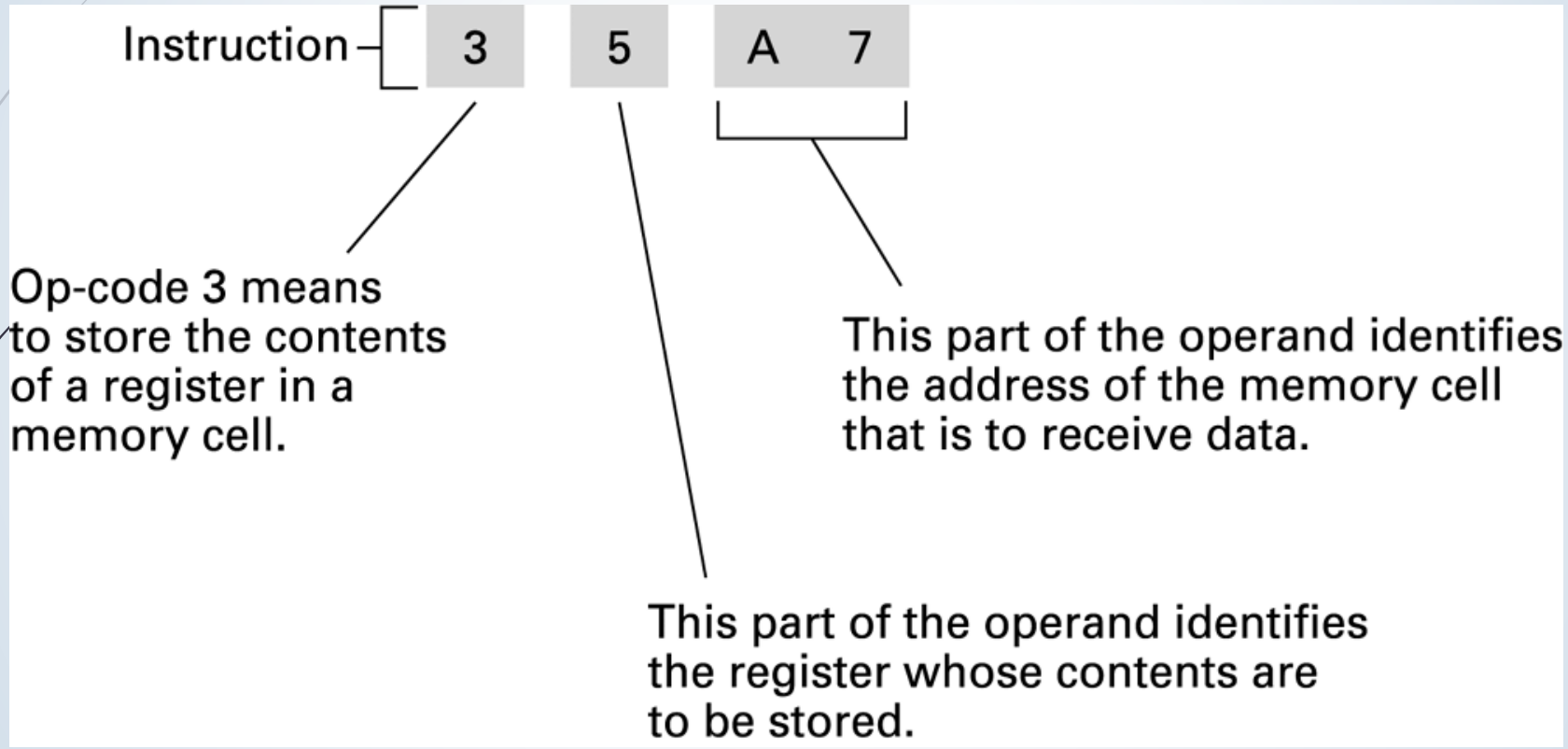
Op-code	Operand	Description
1	RXY	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	LOAD the register R with the bit pattern XY. *Example:* 20A3 would cause the value A3 to be placed in register 0.

Op-code	Operand	Description
3	RXY	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	0RS	MOVE the bit pattern found in register R to register S. *Example:* 40A4 would cause the contents of register A to be copied into register 4.
5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.
6	RST	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.

Instruction set

7	RST	OR the bit patterns in registers S and T and place the result in register R. *Example:* 7CB4 would cause the result of ORing the contents of registers B and 4 to be placed in register C.
8	RST	AND the bit patterns in registers S and T and place the result in register R. *Example:* 8045 would cause the result of ANDing the contents of registers 4 and 5 to be placed in register 0.
9	RST	EXCLUSIVE OR the bit patterns in registers S and T and place the result in register R. *Example:* 95F3 would cause the result of EXCLUSIVE ORing the contents of registers F and 3 to be placed in register 5.
A	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. *Example:* A403 would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion.
B	RXY	JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the program counter during the execute phase.) *Example:* B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the pattern 3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence.
C	000	HALT execution. *Example:* C000 would cause program execution to stop.

Decoding the instruction 35A7



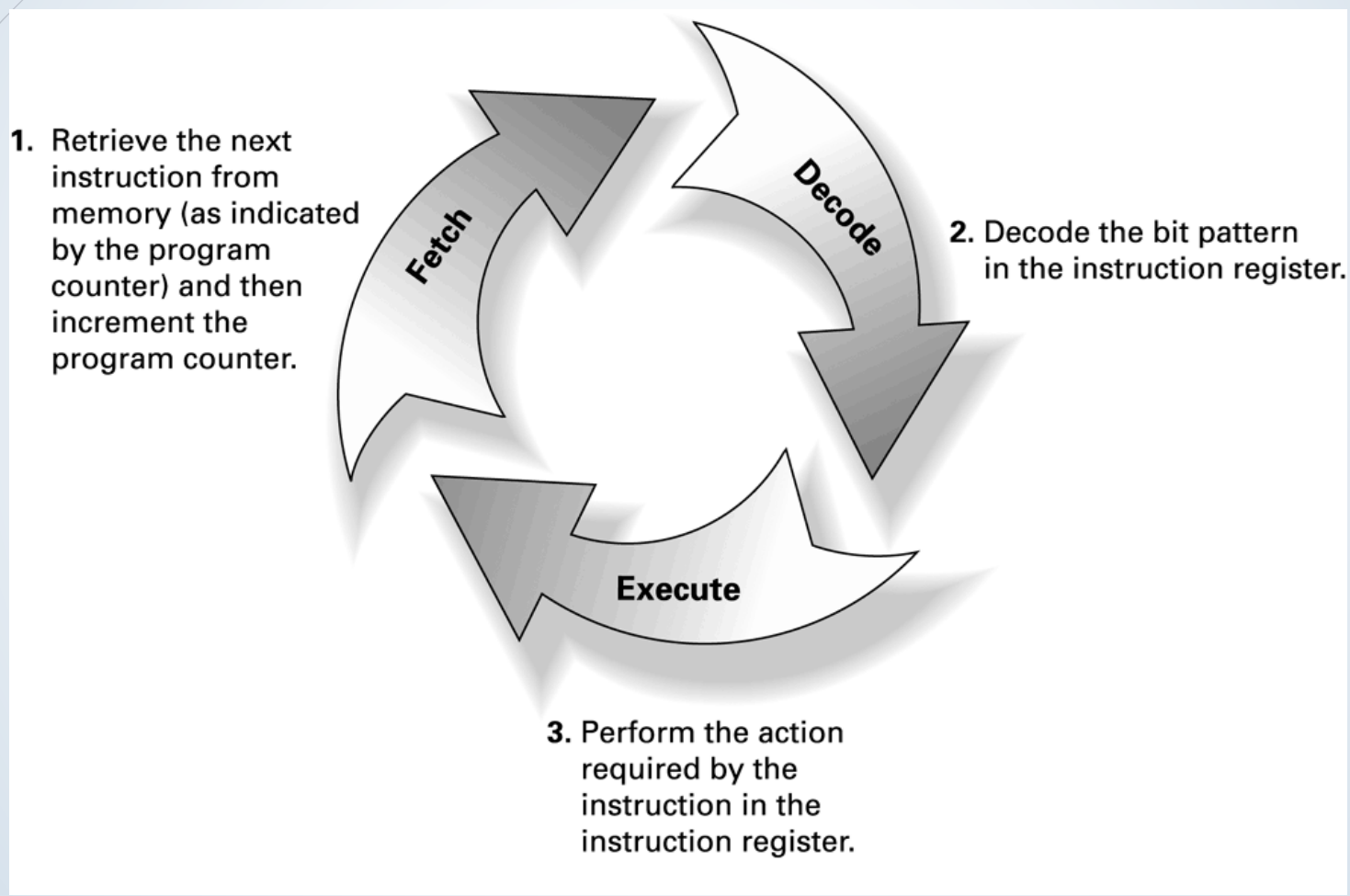
An encoded version of the instructions

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

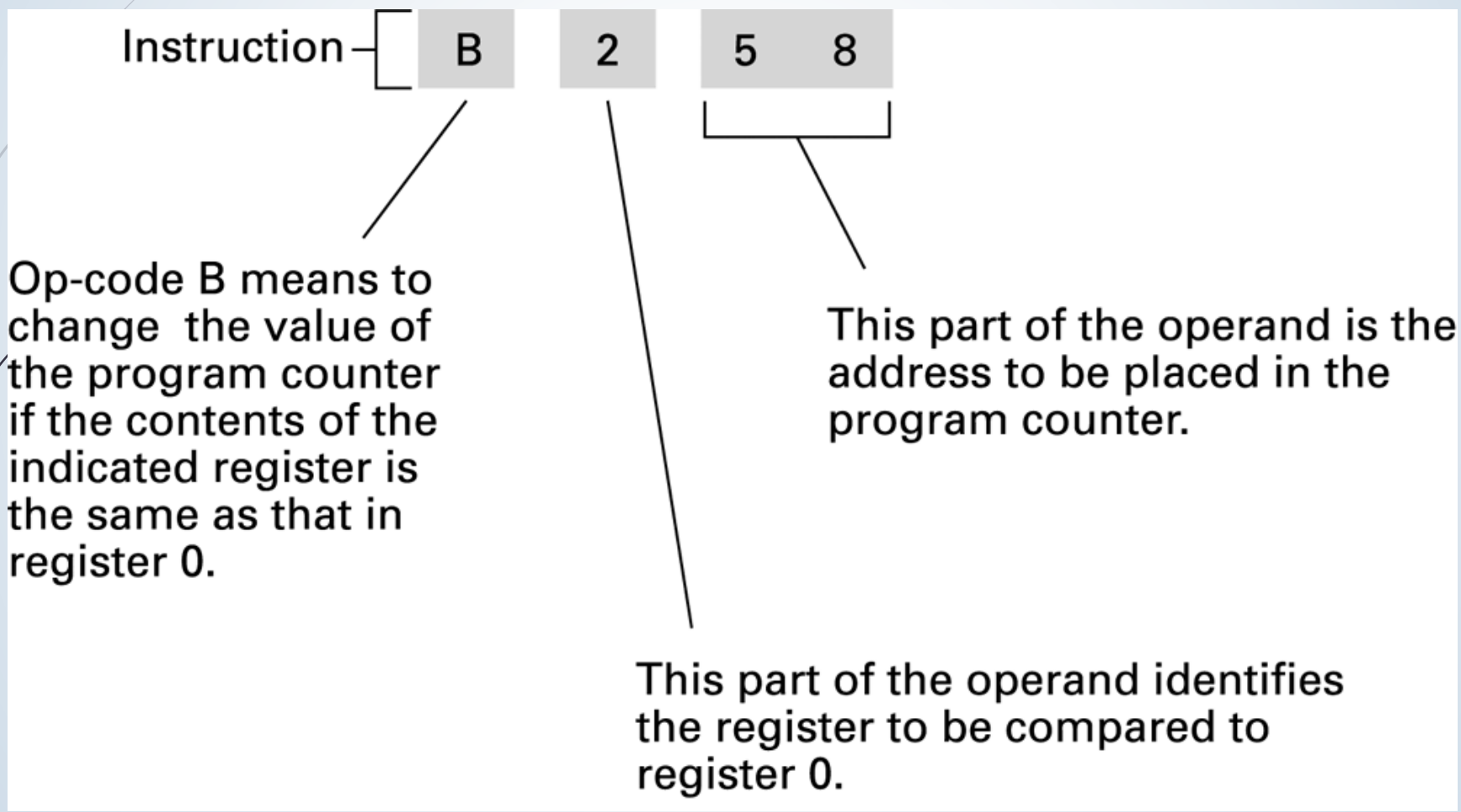
Program Execution

- Controlled by two special-purpose registers
  - Program counter: address of next instruction
  - Instruction register: current instruction
- Machine Cycle
  - Fetch
  - Decode
  - Execute

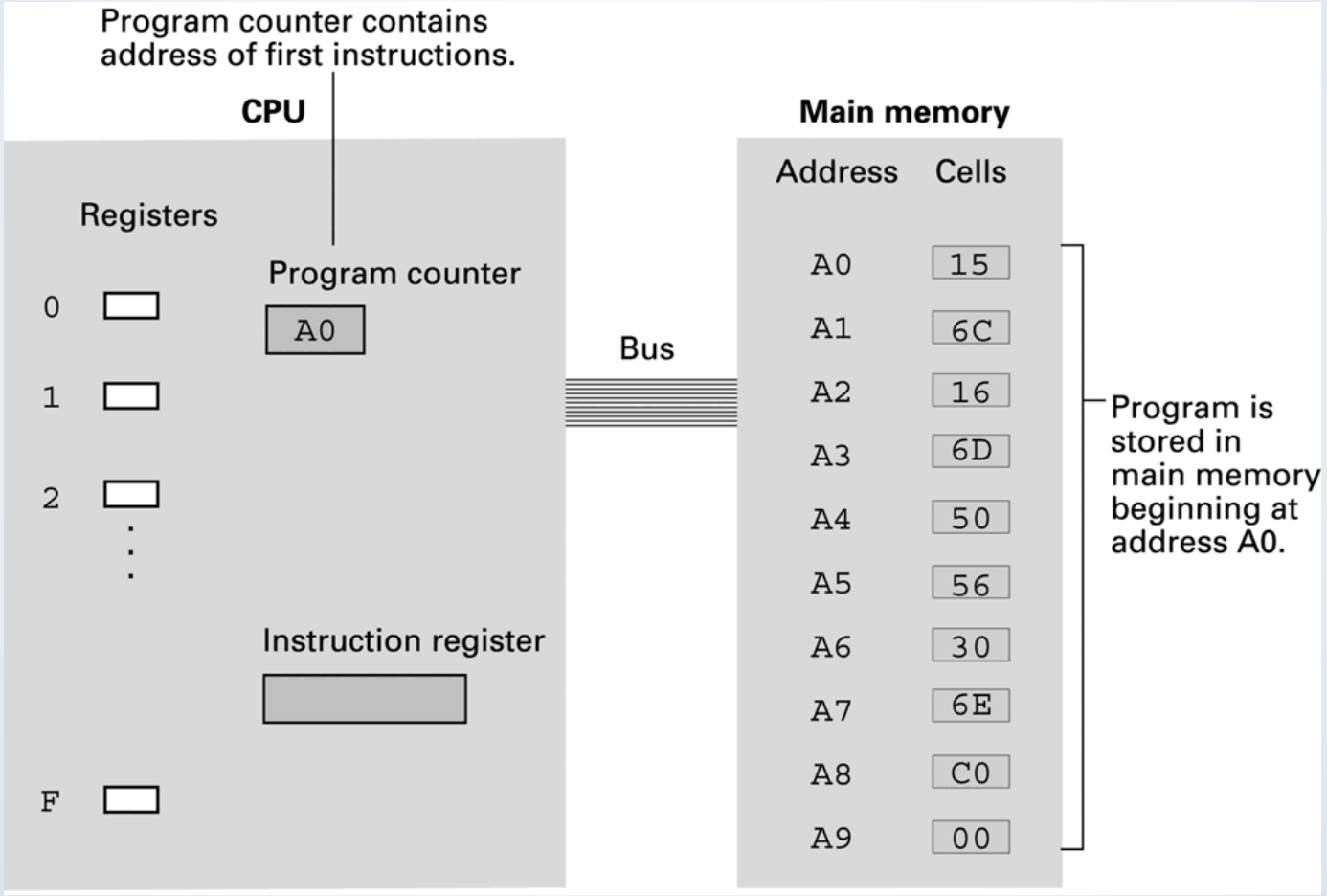
The machine cycle



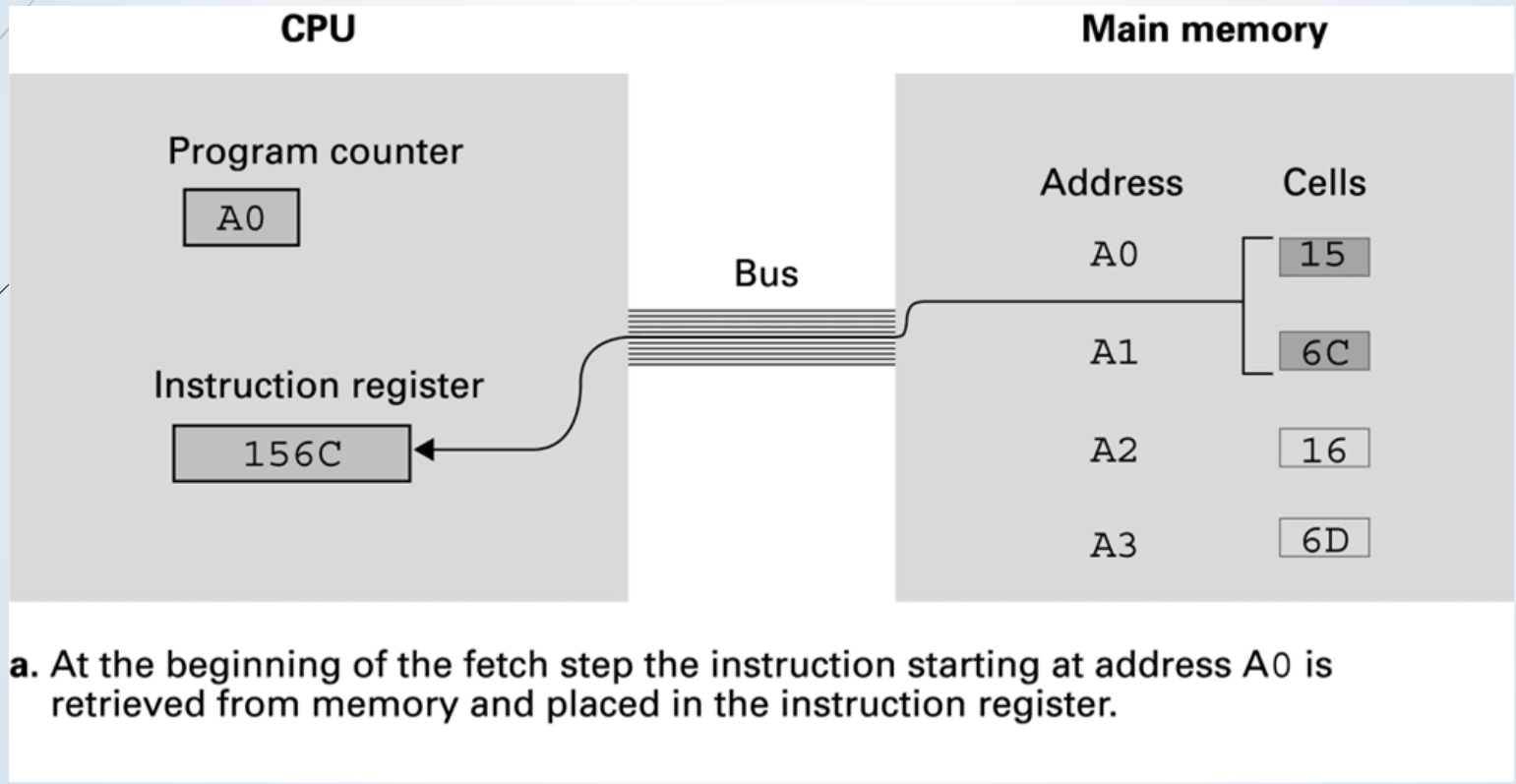
Decoding the instruction B258



The program stored in main memory ready for execution

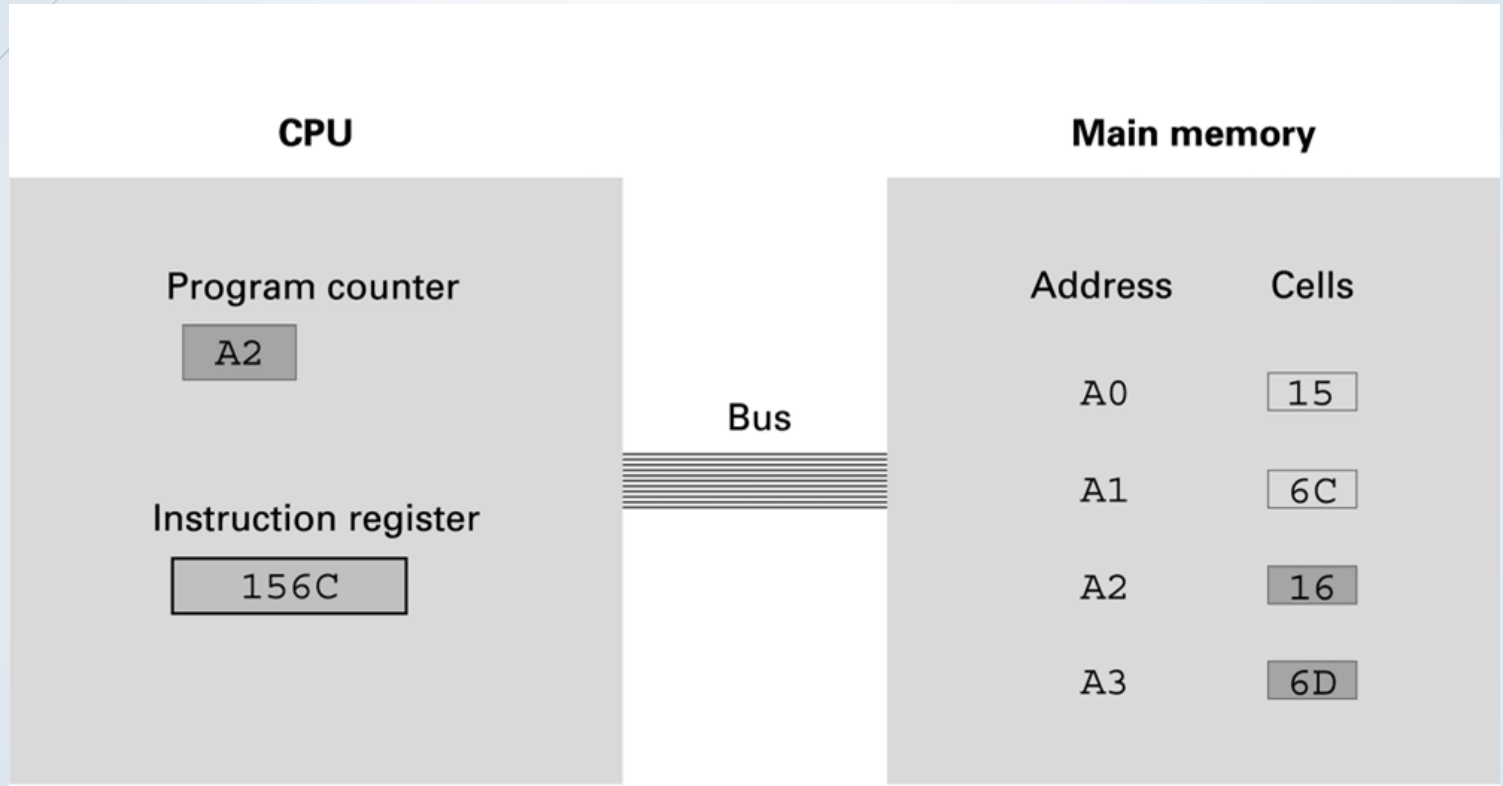


Performing the fetch step of the machine cycle



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

Performing the fetch step of the machine cycle (continued)



b. Then the program counter is incremented so that it points to the next instruction.

Commercial Computers

- 1947 – Eckert and Mauchly formed the Eckert-Mauchly Computer Corporation to manufacture computers commercially
- UNIVAC I (Universal Automatic Computer)
  - First successful commercial computer
  - Was intended for both scientific and commercial applications
  - Commissioned by the US Bureau of Census for 1950 calculations
- The Eckert-Mauchly Computer Corporation became part of the UNIVAC division of the Sperry-Rand Corporation
- UNIVAC II – delivered in the late 1950's
  - Had greater memory capacity and higher performance
- Backward compatible

IBM

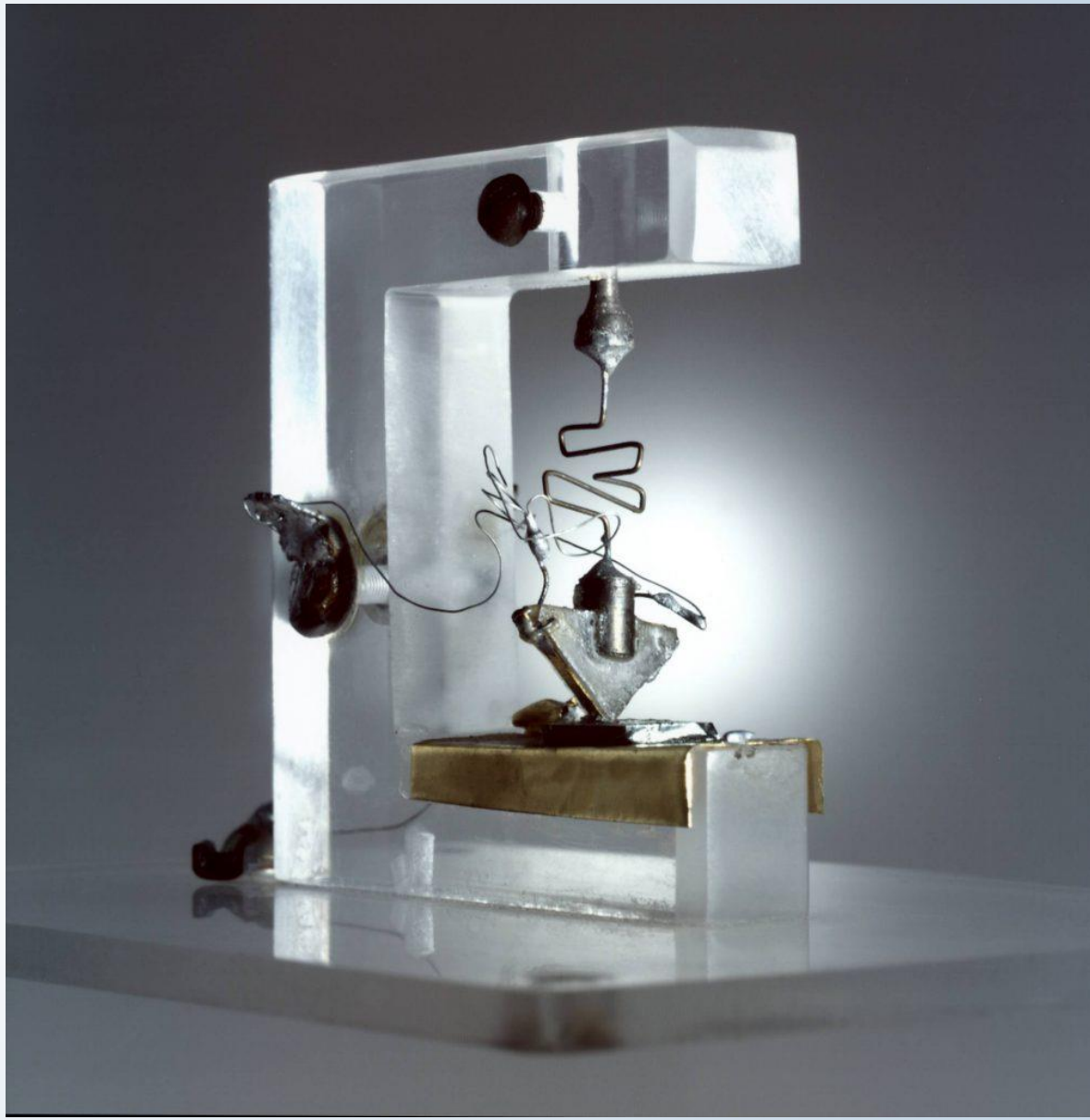
- Was the major manufacturer of punched-card processing equipment
- Delivered its first electronic stored-program computer (701) in 1953
  - Intended primarily for scientific applications
- Introduced 702 product in 1955
  - Hardware features made it suitable to business applications
- Series of 700/7000 computers established IBM as the overwhelmingly dominant computer manufacturer

Example members of the IBM 700/7000 Series

Model Number	First Delivery	CPU Tech-nology	Memory Technology	Cycle Time (μs)	Memory Size (K)	Number of Opcodes	Number of Index Registers	Hardwired Floating-Point	I/O Overlap (Channels)	Instruction Fetch Overlap	Speed (relative to 701)
701	1952	Vacuum tubes	Electrostatic tubes	30	2–4	24	0	no	no	no	1
704	1955	Vacuum tubes	Core	12	4–32	80	3	yes	no	no	2.5
709	1958	Vacuum tubes	Core	12	32	140	3	yes	yes	no	4
7090	1960	Transistor	Core	2.18	32	169	3	yes	yes	no	25
7094 I	1962	Transistor	Core	2	32	185	7	yes (double precision)	yes	yes	30
7094 II	1964	Transistor	Core	1.4	32	185	7	yes (double precision)	yes	yes	50

History of Computers Second Generation: 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	1958–1964	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

Second 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 and libraries
    - perform common computations
  - Appearance of the Digital Equipment Corporation (DEC) in 1957
  - PDP-1 was DEC's first computer
  - This began the mini-computer phenomenon that would become so prominent in the third generation

IBM 7094 Configuration

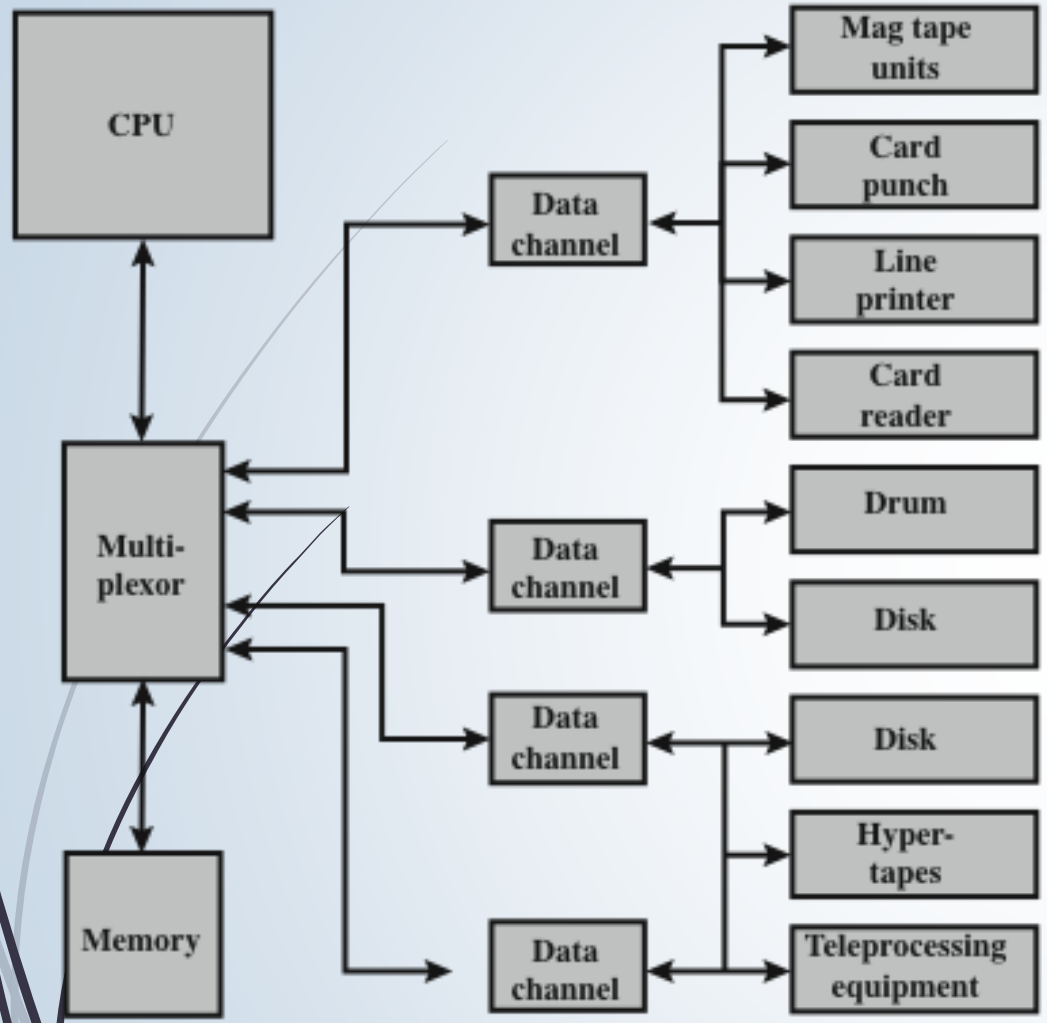


Figure 2.5 An IBM 7094 Configuration

- A **data channel** is an independent I/O module with its own processor and instruction set. In a computer system with such devices, the CPU does not execute detailed I/O instructions. Such instructions are stored in a main memory to be executed by a special-purpose processor in the data channel itself.
- The CPU **initiates** an I/O transfer by sending a control signal to the data channel, instructing it to execute a sequence of instructions in memory. The data channel performs its task independently of the CPU and signals the CPU when the operation is complete.
- **multiplexor**, schedules access to the memory from the CPU and data channels, allowing these devices to act independently.

History of Computers Third Generation: Integrated Circuits

- 1958 – the **invention** of the **integrated circuit**
- *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

Microelectronics

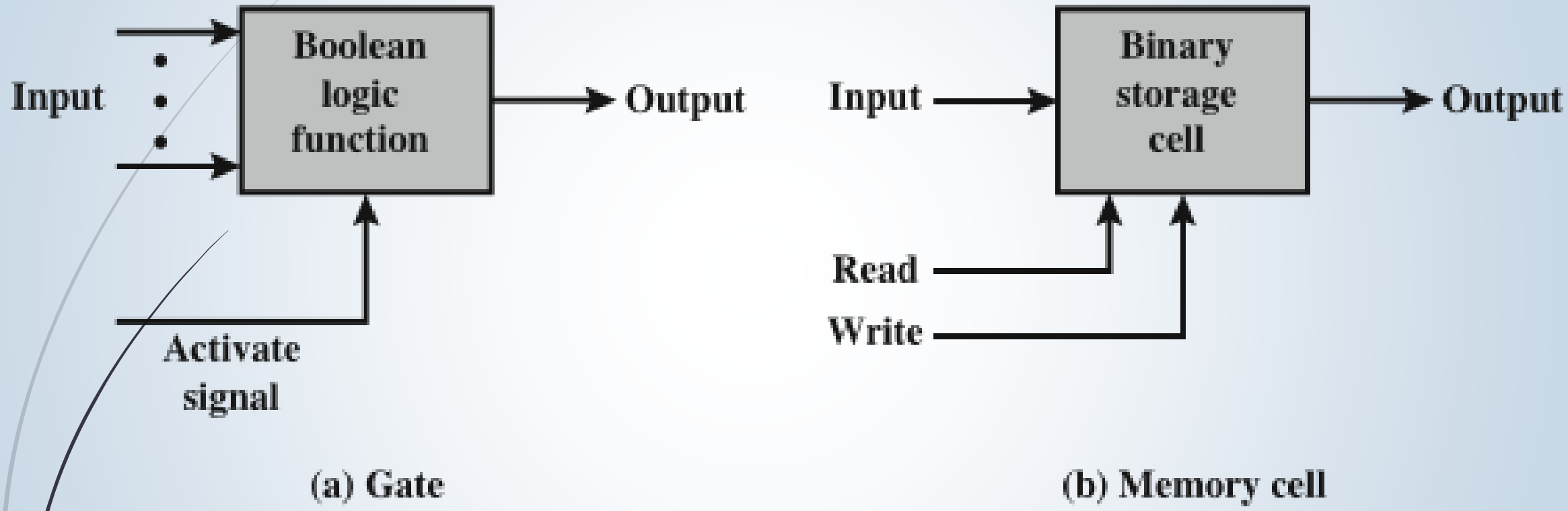


Figure 2.6 Fundamental Computer Elements

Wafer, Chip, and Gate Relationship

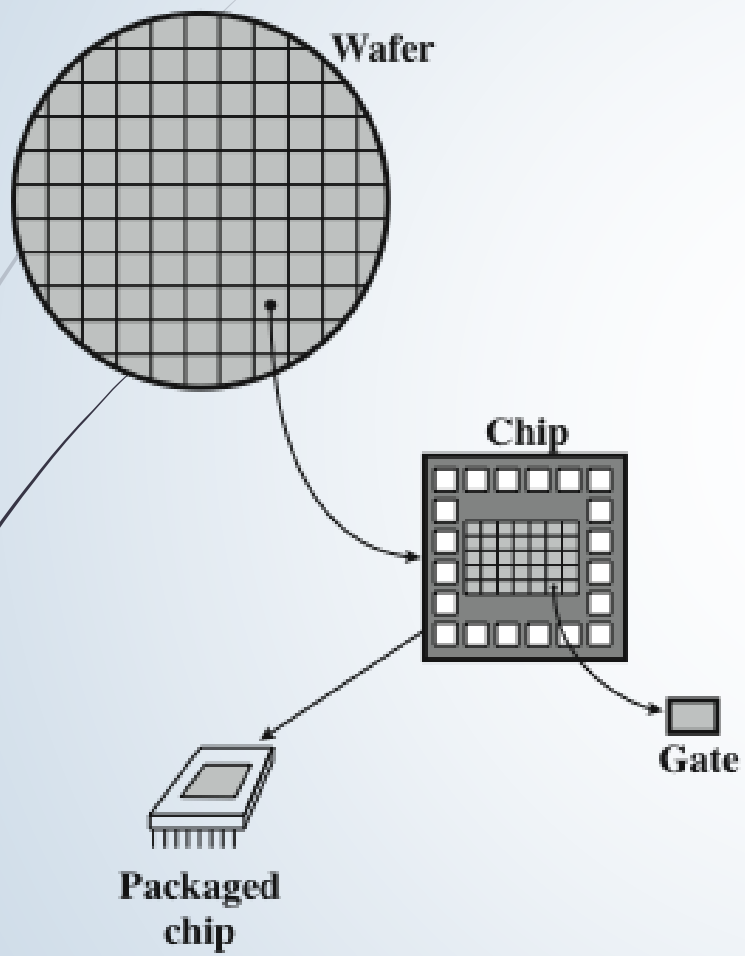


Figure 2.7 Relationship Among Wafer, Chip, and Gate

- A thin **wafer** of silicon is divided into a matrix of small areas, each a few millimeters square.
- The identical circuit pattern is fabricated in each area, and the wafer is broken up into **chips**.
- Each chip consists of many gates and/or memory cells plus a number of input and output attachment points.
- This chip is then packaged in housing that protects it and provides pins for attachment to devices beyond the chip.
- A number of these packages can then be interconnected on a printed circuit board to produce larger and more complex circuits.

Chip Growth

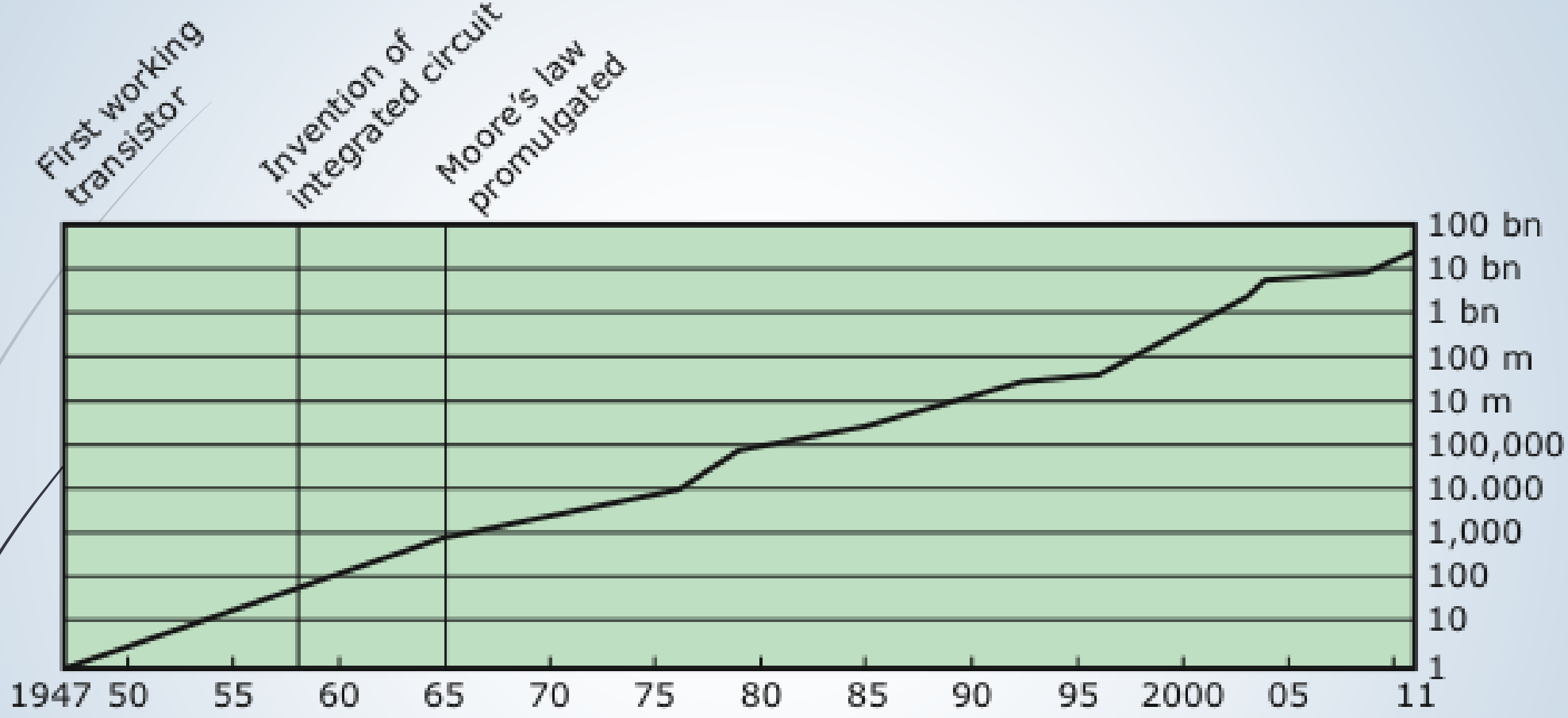


Figure 2.8 Growth in Transistor Count on Integrated Circuits (DRAM memory)

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

Table 2.4 Characteristics of the System/360 Family

Characteristic	Model 30	Model 40	Model 50	Model 65	Model 75
Maximum memory size (bytes)	64K	256K	256K	512K	512K
Data rate from memory (Mbytes/sec)	0.5	0.8	2.0	8.0	16.0
Processor cycle time μs)	1.0	0.625	0.5	0.25	0.2
Relative speed	1	3.5	10	21	50
Maximum number of data channels	3	3	4	6	6
Maximum data rate on one channel (Kbytes/s)	250	400	800	1250	1250

Similar or identical instruction set: In many cases, the exact same set of machine instructions is supported on all members of the family. So, a program that executes on one machine will also execute on any other.
Similar or identical operating system: The same basic operating system is available for all family members. In some cases, additional features are added to the higher-end members.
Increasing speed: The rate of instruction execution increases in going from lower to higher family members.
Increasing number of I /O ports: The number of I/O ports increases in going from lower to higher family members.
Increasing memory size: The size of main memory increases in going from lower to higher family members.
Increasing cost: At a given point in time, the cost of a system increases in going from lower to higher family members.

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

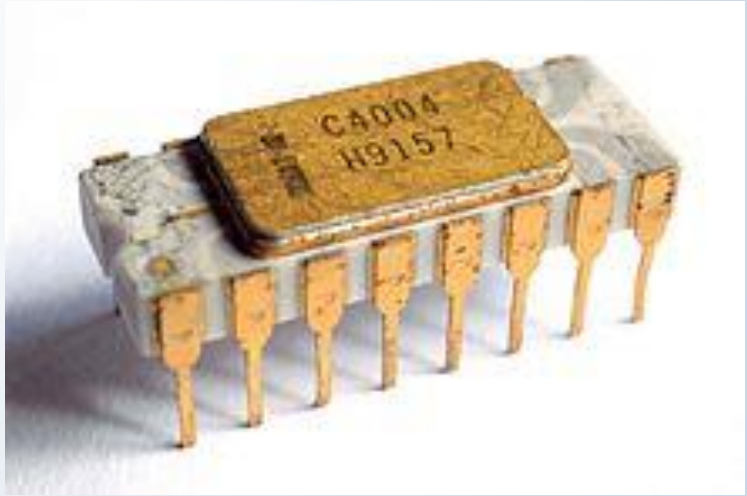
There has been a continuing and rapid decline in memory cost accompanied by a corresponding increase in physical memory density	Developments in memory and processor technologies changed the nature of computers in less than a decade

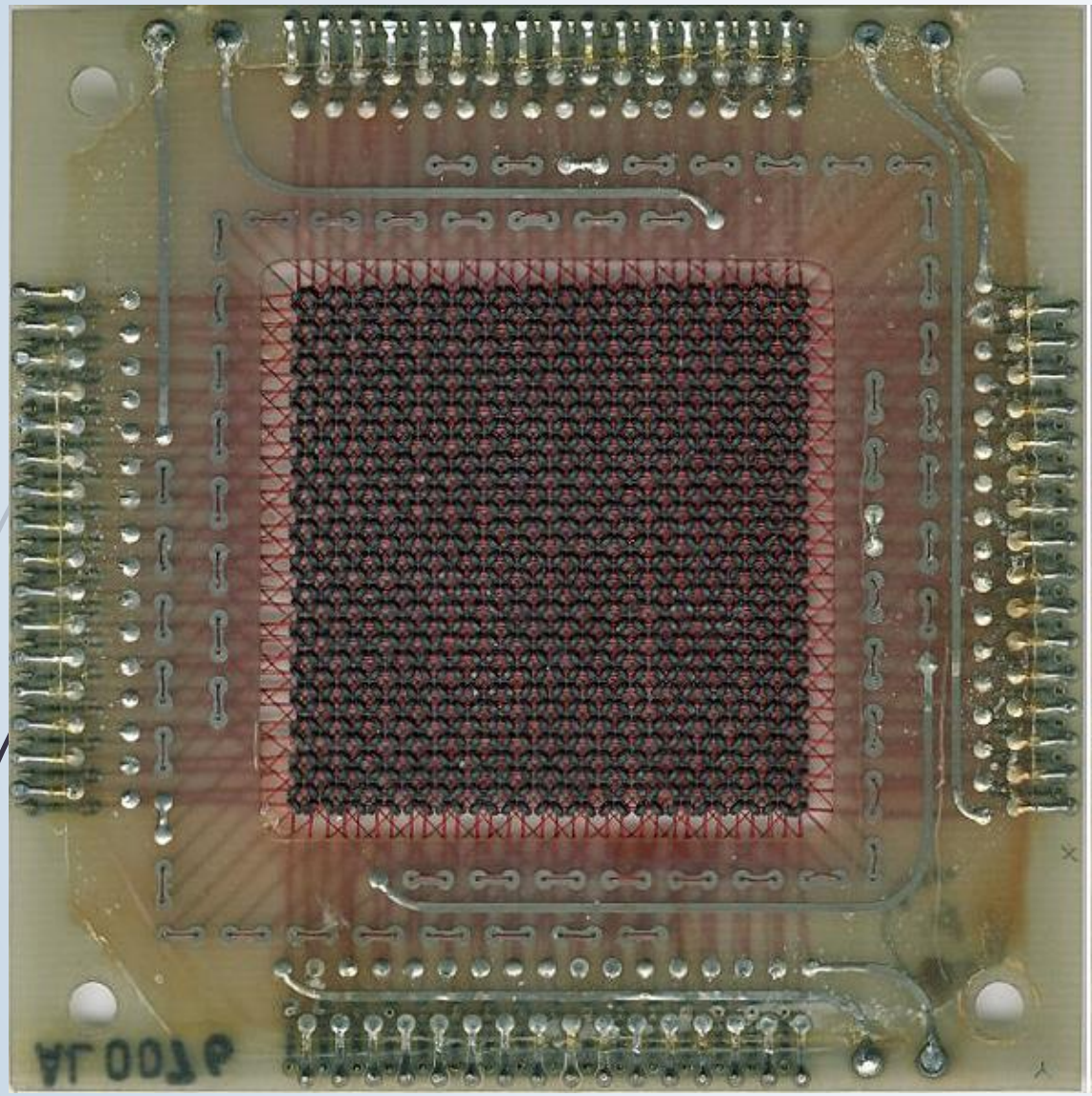
Since 1970 semiconductor memory has been through 13 generations

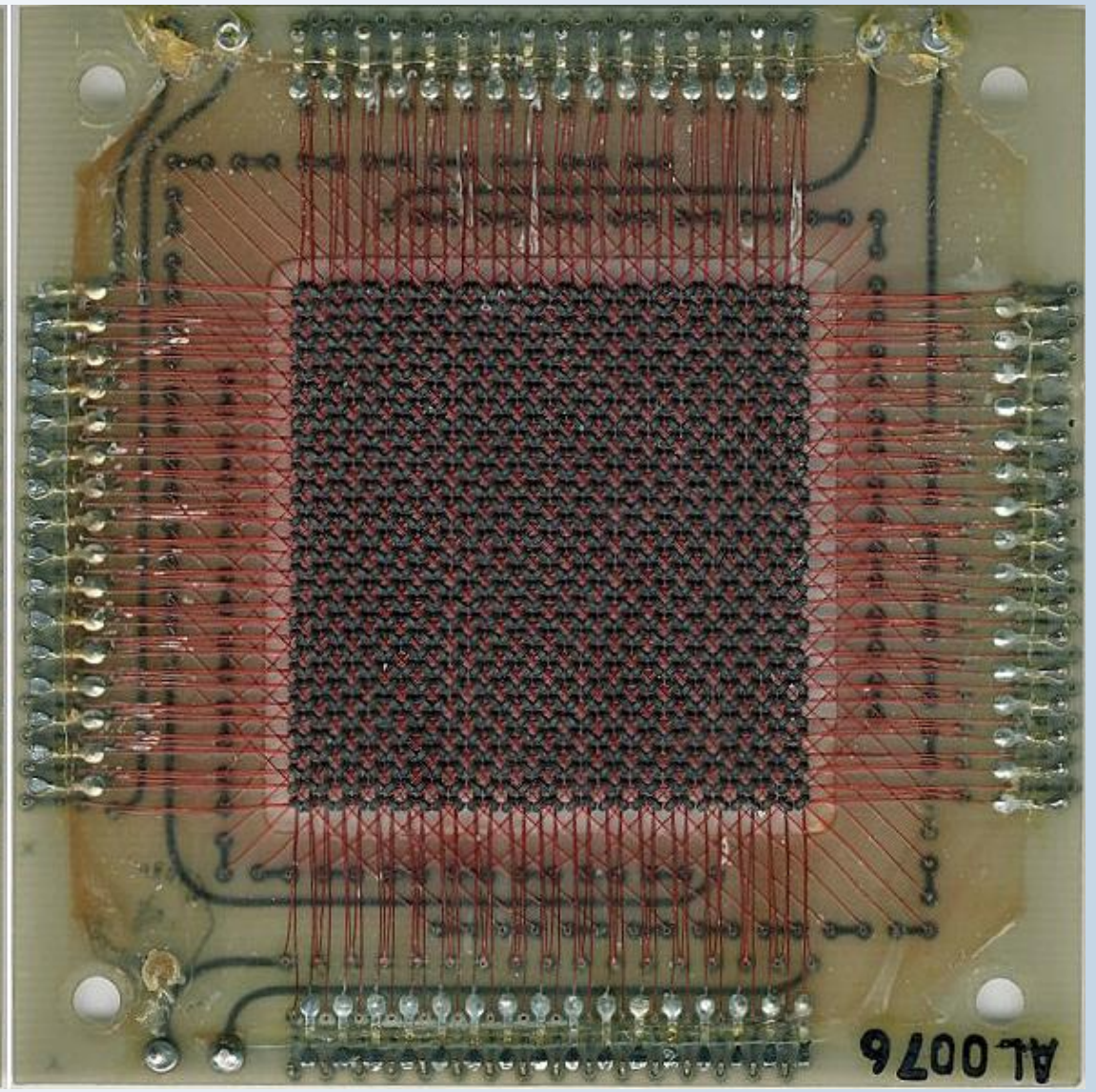
Each generation has provided four times the storage density of the previous generation, accompanied by declining cost per bit and declining access time

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






AL0076

Evolution of Intel Microprocessors

	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		6	3	6
Addressable memory	640 Bytes	16 KB	64 KB	1 MB	1 MB

a. 1970s Processors

	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

b. 1980s Processors

Evolution of Intel Microprocessors

	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

c. 1990s Processors

	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 990
Introduced	1999	2000	2006	2011
Clock speeds	450 - 660 MHz	1.3 - 1.8 GHz	1.06 - 1.2 GHz	3.5 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1170 million
Feature size (nm)	250	180	65	32
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 L2/12 MB L3

d. Recent Processors

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**

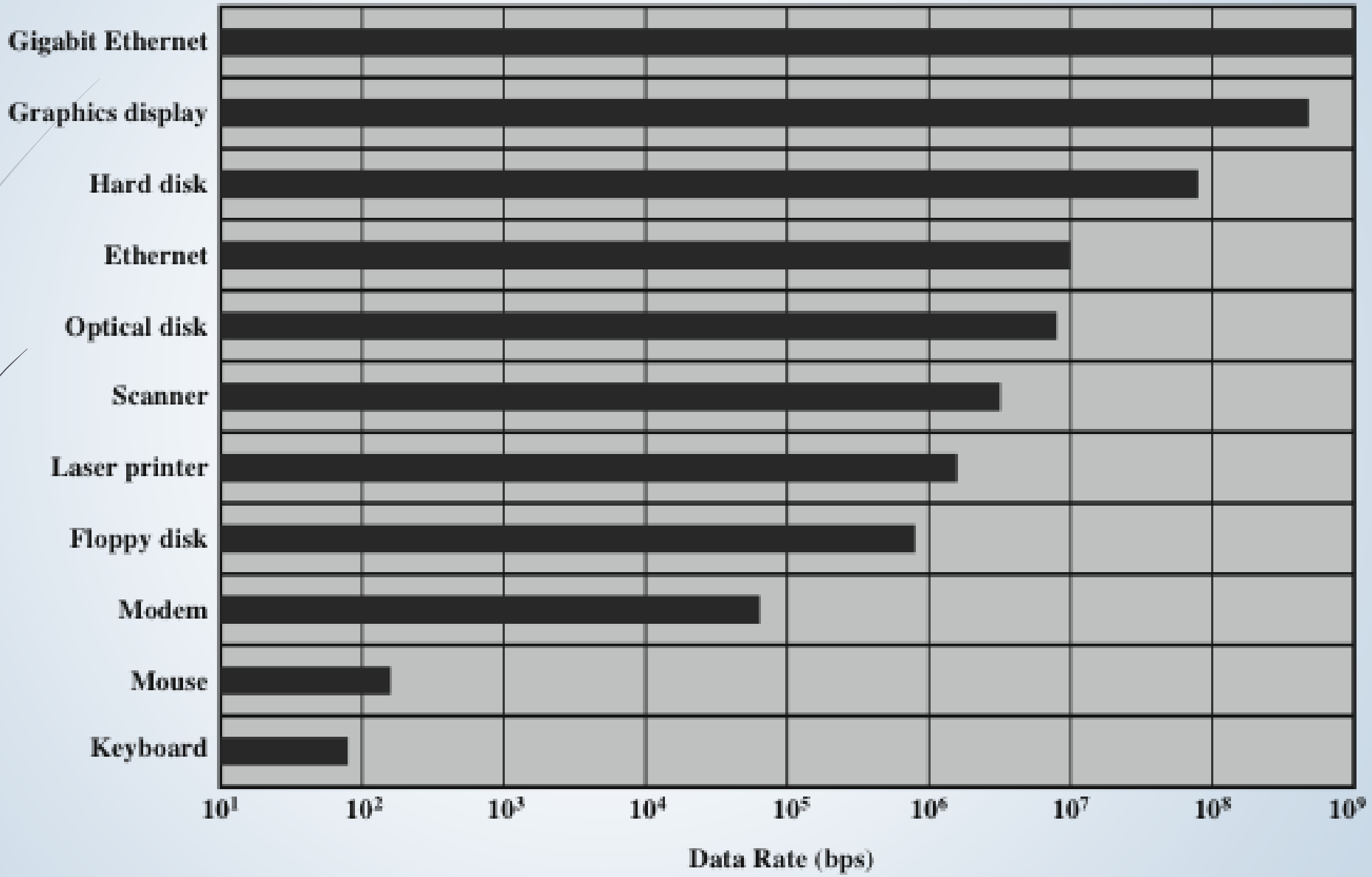
Data flow analysis

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**

Typical I/O Device Data Rates



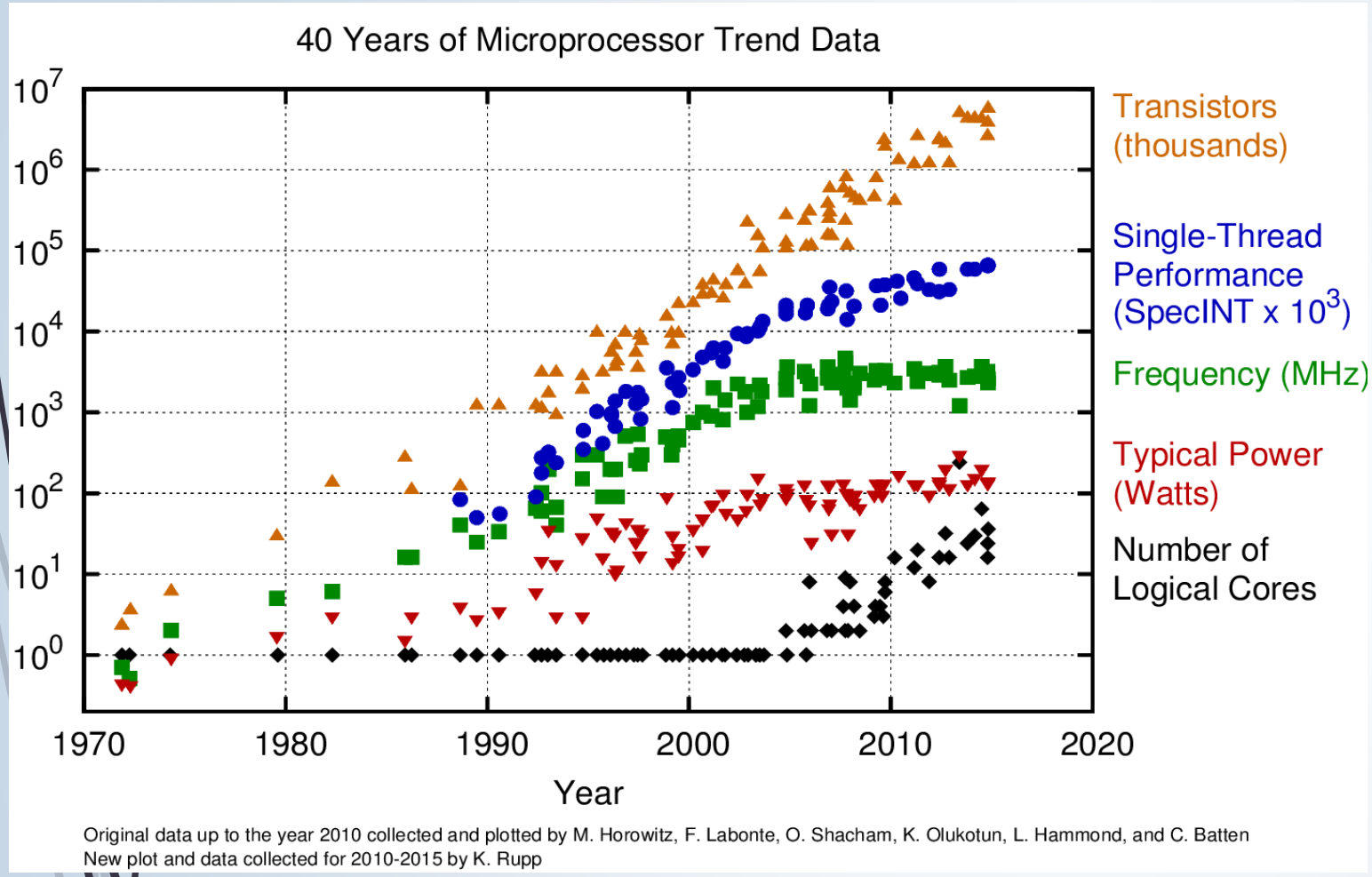
Improvements in Chip Organization and Architecture

- 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 Login Density

- Power
  - Power density (Watts/cm^2) 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 RC product increases
  - Wire interconnects thinner, increasing resistance
  - Wires closer together, increasing capacitance
- Memory latency
  - Memory speeds lag processor speeds

Processor Trends



- There are now typically two or three levels of cache between the processor and main memory.
- As chip density has increased, more of the cache memory has been incorporated on the chip, enabling faster cache access. For example, the original Pentium chip devoted about 10% of on-chip area to a cache. Contemporary chips devote over half of the chip area to caches.

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

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

- 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
- 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
- **GPUs used as vector processors** for a variety of applications that require repetitive computations.

x86 Architecture

- Results of decades of design effort on complex instruction set computers (CISCs)
- Excellent example of CISC design Intel
- Incorporates the sophisticated design principles once found only on mainframes and supercomputers
- An alternative approach to processor design is the reduced instruction set computer (RISC) ARM
- The 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
- In terms of market share Intel is ranked as the number one maker of microprocessors for non-embedded systems

x86 Evolution

- 8080
  - First general purpose microprocessor
  - 8-bit machine with an 8-bit data path to memory
  - Used in the first personal computer (Altair)
- 8086
  - 16-bit machine
  - Used an instruction cache, or queue
  - First appearance of the x86 architecture
- 8088
  - used in IBM's first personal computer
- 80286
  - Enabled addressing a 16-MByte memory instead of just 1 MByte
- 80386
  - Intel's first 32-bit machine
  - First Intel processor to support multitasking
- 80486
  - More sophisticated cache technology and instruction pipelining
  - Built-in math coprocessor

x86 Evolution - Pentium

Pentium

- Superscalar
- Multiple instructions executed in parallel

Pentium Pro

- Increased superscalar organization
- Aggressive register renaming
- Branch prediction
- Data flow analysis
- Speculative execution

Pentium II

- MMX technology
- Designed specifically to process video, audio, and graphics data

Pentium III

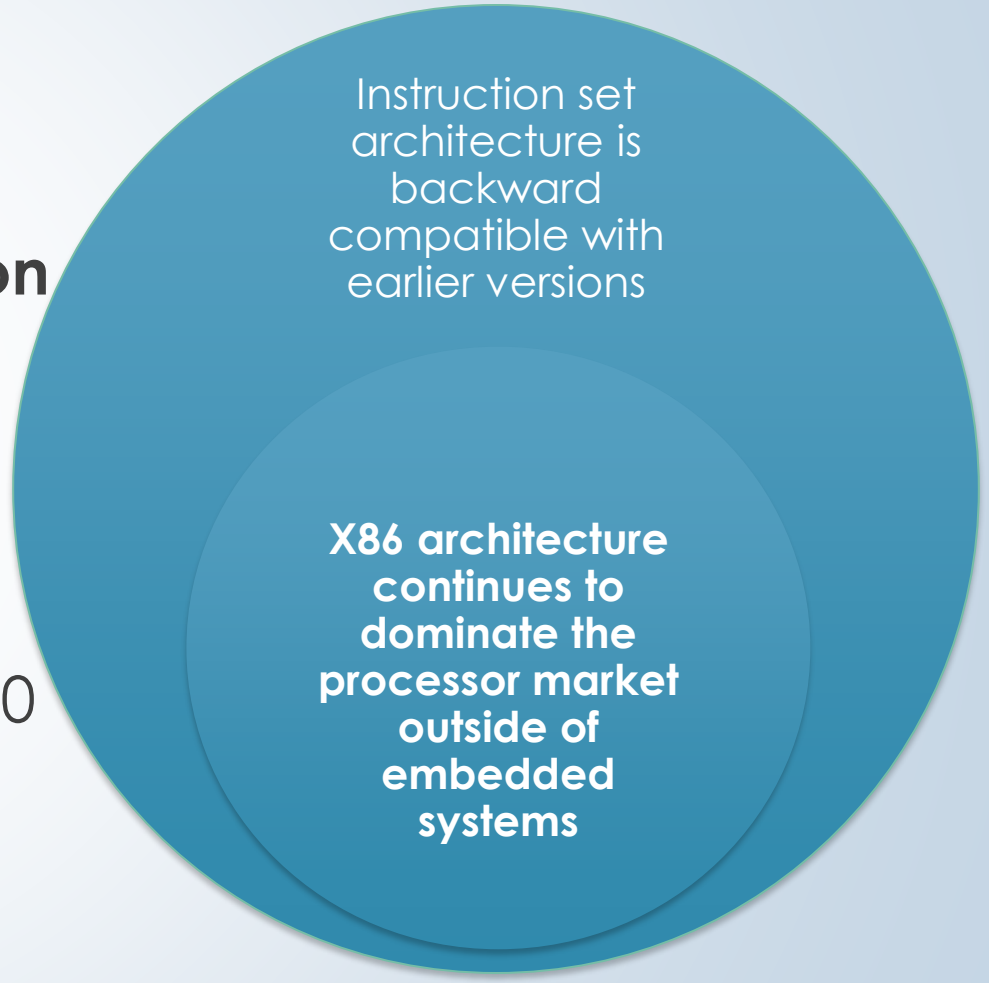
- Additional floating-point instructions to support 3D graphics software

Pentium 4

- Includes additional floating-point and other enhancements for multimedia

x86 Evolution (continued)

- Core
  - First Intel x86 microprocessor with a **dual core**, referring to the implementation of **two processors on a single chip**
- Core 2
  - Extends the architecture to **64 bits**
  - Recent Core offerings have up to 10 processors per chip



Examples of Embedded Systems and Their Markets

Market	Embedded Device
Automotive	Ignition system Engine control Brake system
Consumer electronics	Digital and analog televisions Set-top boxes (DVDs, VCRs, Cable boxes) Personal digital assistants (PDAs) Kitchen appliances (refrigerators, toasters, microwave ovens) Automobiles Toys/games Telephones/cell phones/pagers Cameras Global positioning systems
Industrial control	Robotics and controls systems for manufacturing Sensors
Medical	Infusion pumps Dialysis machines Prosthetic devices Cardiac monitors
Office automation	Fax machine Photocopier Printers Monitors Scanners

Embedded Systems Requirements and Constraints

Small to large systems, implying different cost constraints and different ns for optimization and reuse

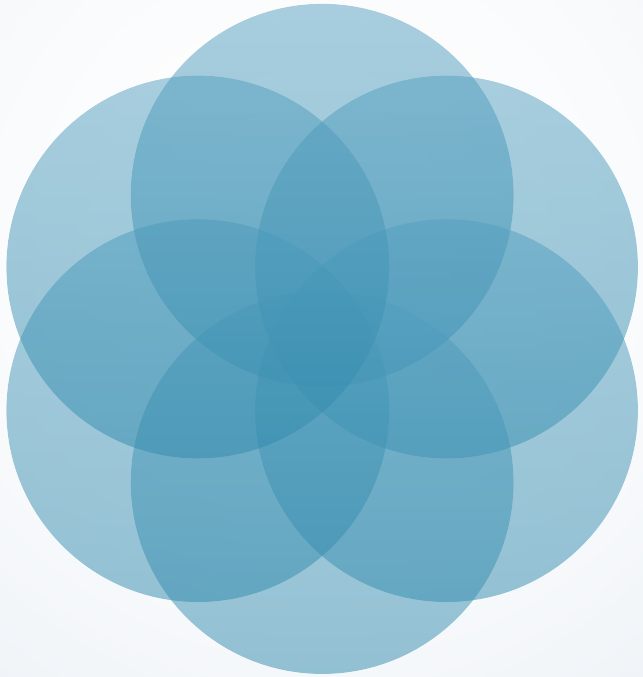
Relaxed to very strict requirements and combinations of different quality requirements with respect to safety, reliability, real-time and flexibility

Short to long life times

Different environmental conditions in terms of radiation, vibrations, and humidity

Different application characteristics resulting in static versus dynamic loads, slow to fast speed, compute versus interface intensive tasks, and/or combinations thereof

Different models of computation ranging from discrete event systems to hybrid systems



Acorn RISC Machine (ARM)

- Family of RISC-based microprocessors and microcontrollers
- Designs microprocessor and multicore architectures and licenses them to manufacturers
- Chips are high-speed processors that are known for their small die size and low power requirements
  - Widely used in PDAs and other handheld devices
  - Chips are the processors in iPod and iPhone devices
  - Most widely used embedded processor architecture
  - Most widely used processor architecture of any kind

ARM Evolution

Family	Notable Features	Cache	Typical MIPS @ MHz
ARM1	32-bit RISC	None	
ARM2	Multiply and swap instructions; Integrated memory management unit, graphics and I/O processor	None	7 MIPS @ 12 MHz
ARM3	First use of processor cache	4 KB unified	12 MIPS @ 25 MHz
ARM6	First to support 32-bit addresses; floating-point unit	4 KB unified	28 MIPS @ 33 MHz
ARM7	Integrated SoC	8 KB unified	60 MIPS @ 60 MHz
ARM8	5-stage pipeline; static branch prediction	8 KB unified	84 MIPS @ 72 MHz
ARM9		16 KB/16 KB	300 MIPS @ 300 MHz
ARM9E	Enhanced DSP instructions	16 KB/16 KB	220 MIPS @ 200 MHz
ARM10E	6-stage pipeline	32 KB/32 KB	
ARM11	9-stage pipeline	Variable	740 MIPS @ 665 MHz
Cortex	13-stage superscalar pipeline	Variable	2000 MIPS @ 1 GHz
XScale	Applications processor; 7-stage pipeline	32 KB/32 KB L1 512 KB L2	1000 MIPS @ 1.25 GHz

ARM Design Categories

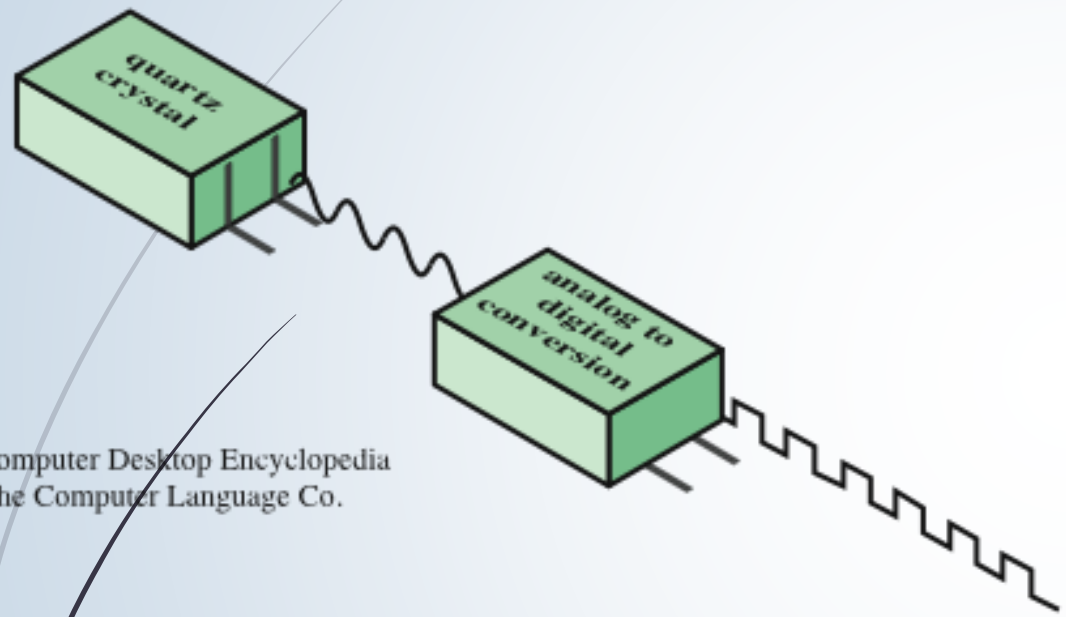
- ARM processors are designed to meet the needs of three system categories:

- Secure applications
  - Smart cards, SIM cards, and payment terminals

- Embedded real-time systems
  - Systems for storage, automotive body and power-train, industrial, and networking applications

- Application platforms
  - Devices running open operating systems including Linux, Palm OS, Symbian OS, and Windows CE in wireless, consumer entertainment and digital imaging applications

System Clock



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

Figure 2.13 System Clock

- Operations performed by a processor, such as fetching an instruction, decoding the instruction, performing an arithmetic operation, and so on, are governed by a system clock.
- Typically, all operations begin with the pulse of the clock. Thus, at the most fundamental level, the speed of a processor is dictated by the pulse frequency produced by the clock, measured in cycles per second, or Hertz (Hz).
- **For example, a 1-GHz processor receives 1 billion pulses per second.**
- The rate of pulses is known as the **clock rate**, or **clock speed**.

Benchmarks

For example, consider this high-level language statement:

A = B + C /* assume all quantities in main memory */

With a traditional instruction set architecture, referred to as a complex instruction set computer (CISC), this instruction can be compiled into one processor instruction:

add mem(B), mem(C), mem (A)

On a typical RISC machine, the compilation would look something like this:

```
load mem(B), reg(1);
load mem(C), reg(2);
add reg(1), reg(2), reg(3);
store reg(3), mem (A)
```

Desirable Benchmark Characteristics

- Written in a high-level language, making it portable across different machines
- Representative of a particular kind of programming style, such as system programming, numerical programming, or commercial programming
- Can be measured easily
- Has wide distribution

System Performance Evaluation Corporation (SPEC)

- Benchmark suite
  - A collection of programs, defined in a high-level language
  - Attempts to provide a representative test of a computer in a particular application or system programming area
- SPEC
  - An industry consortium
  - Defines and maintains the best known collection of benchmark suites
  - Performance measurements are widely used for comparison and research purposes
  - https://www.spec.org/

SPEC CPU2006

- Best known SPEC benchmark suite
- Industry standard suite for processor intensive applications
- Appropriate for measuring performance for applications that spend most of their time doing computation rather than I/O
- Consists of 17 floating point programs written in C, C++, and Fortran and 12 integer programs written in C and C++
- Suite contains over 3 million lines of code
- Fifth generation of processor intensive suites from SPEC

Amdahl's Law

- Gene Amdahl [AMDA67]
- 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

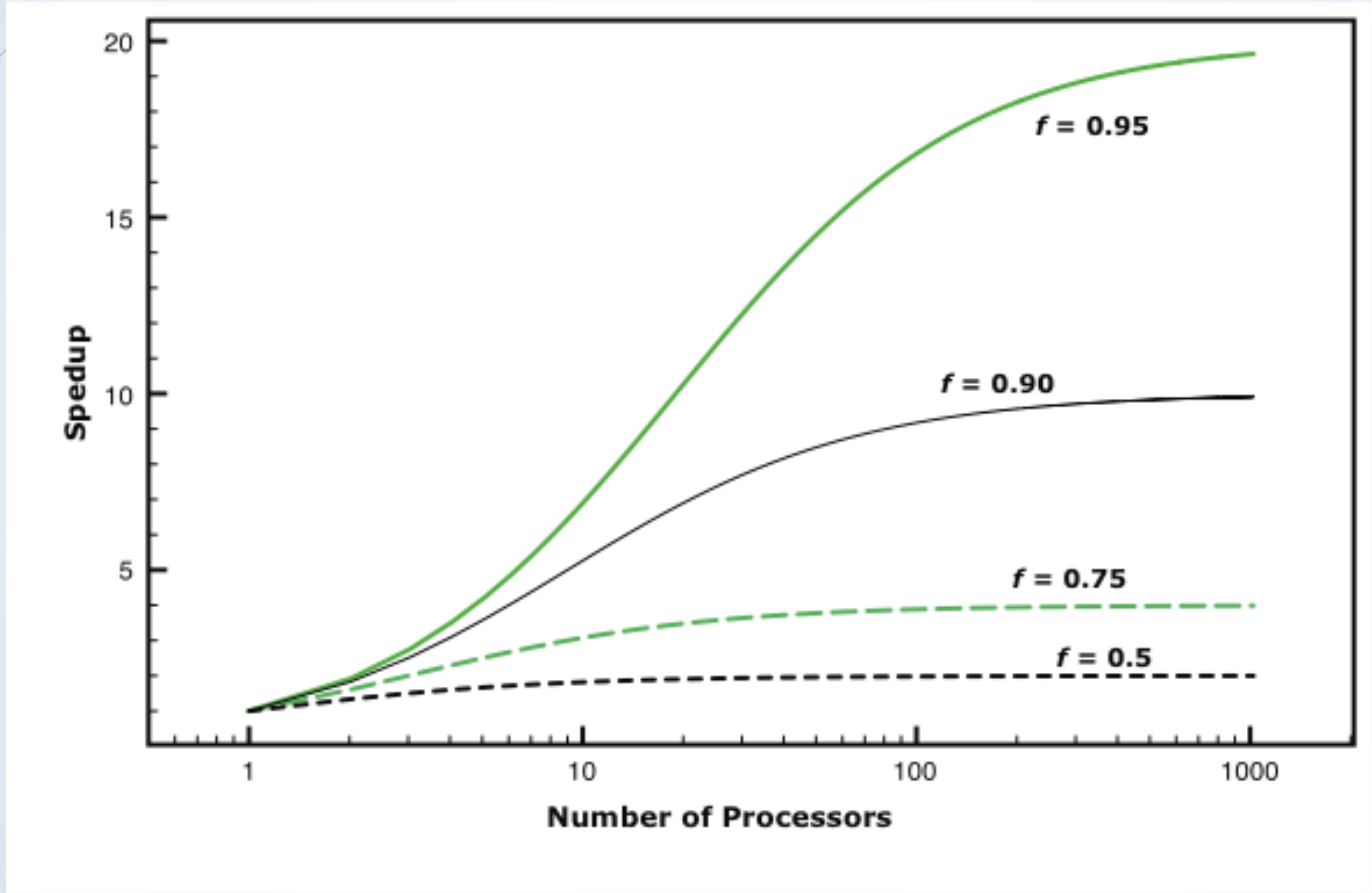


Figure 2.14 Amdahl's Law for Multiprocessors

Summary

- First generation computers
  - Vacuum tubes
- Second generation computers
  - Transistors
- Third generation computers
  - Integrated circuits
- Performance designs
  - Microprocessor speed
  - Performance balance
  - Chip organization and architecture

- Computer Evolution and Performance
- Multi-core
- MICs
- GPGPUs
- Evolution of the Intel x86
- Embedded systems
- ARM evolution
- Performance assessment
  - Clock speed and instructions per second
  - Benchmarks
  - Amdahl's Law