VIT-AP UNIVERSITY

Computer Organization & Architecture

ECE 2002

DR. S SARITHA

ASSISTANT PROFESSOR

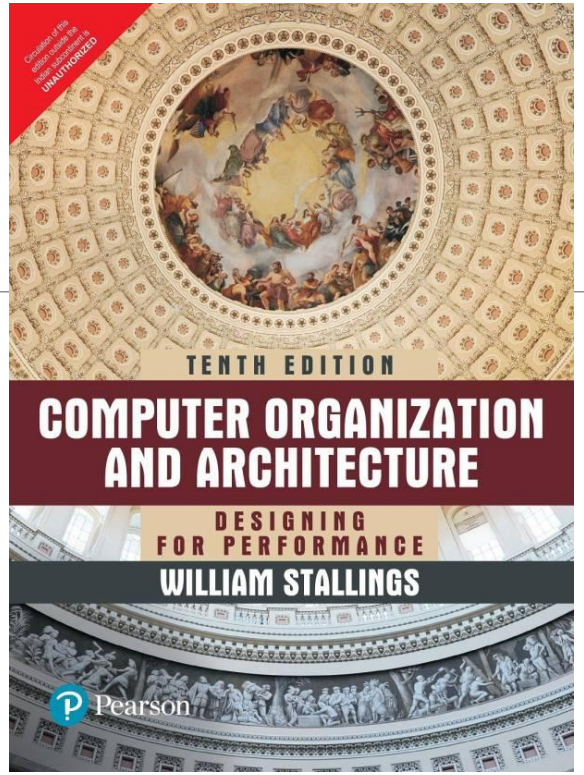
SENSE

Module 1: Computer evolution and arithmetic

- A Brief History of computers
- Basic structures of Computers: Computer Architecture vs. Computer Organization
- Functional units and Operational concepts
- RISC Vs CISC
- Registers
- Performance assessment, MIPS
- Bus and Bus organization
- Memory location and addresses
- Fixed and Floating point numbers, Signed numbers, Integer Arithmetic

Text Book:

William Stallings, Computer Organization and Architecture: Designing for Performance, Pearson Education, Tenth Edition,2013



Books:

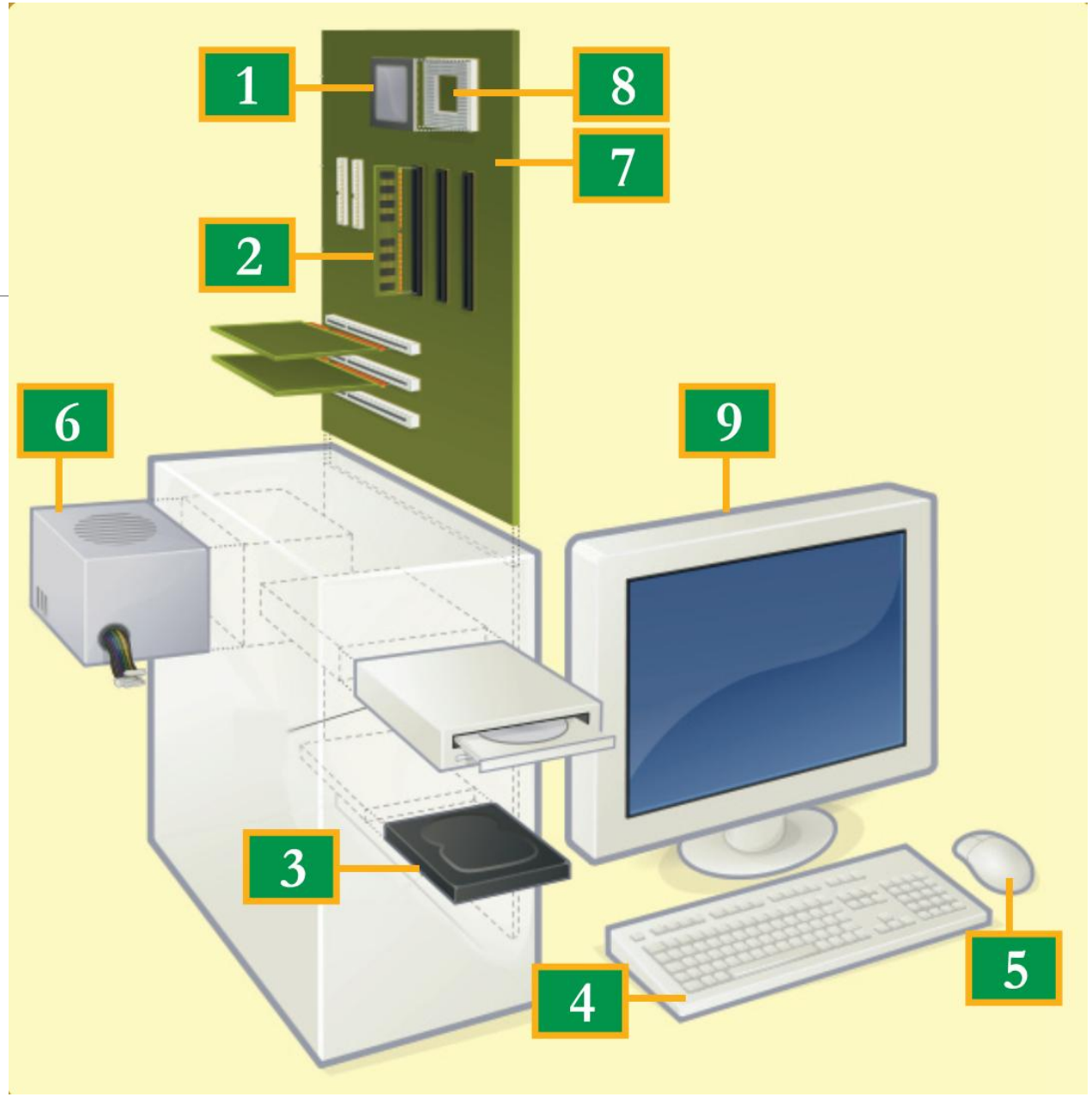
1. M. Morris Mano, Rajib Mall, Computer System Architecture, Pearson Education Third Edition,2017.
2. Carl Hamacher, Zvonkovranesic, Safwat Zaky , Computer Organization, McGraw Hill, Fifth Edition,2011.

Computer

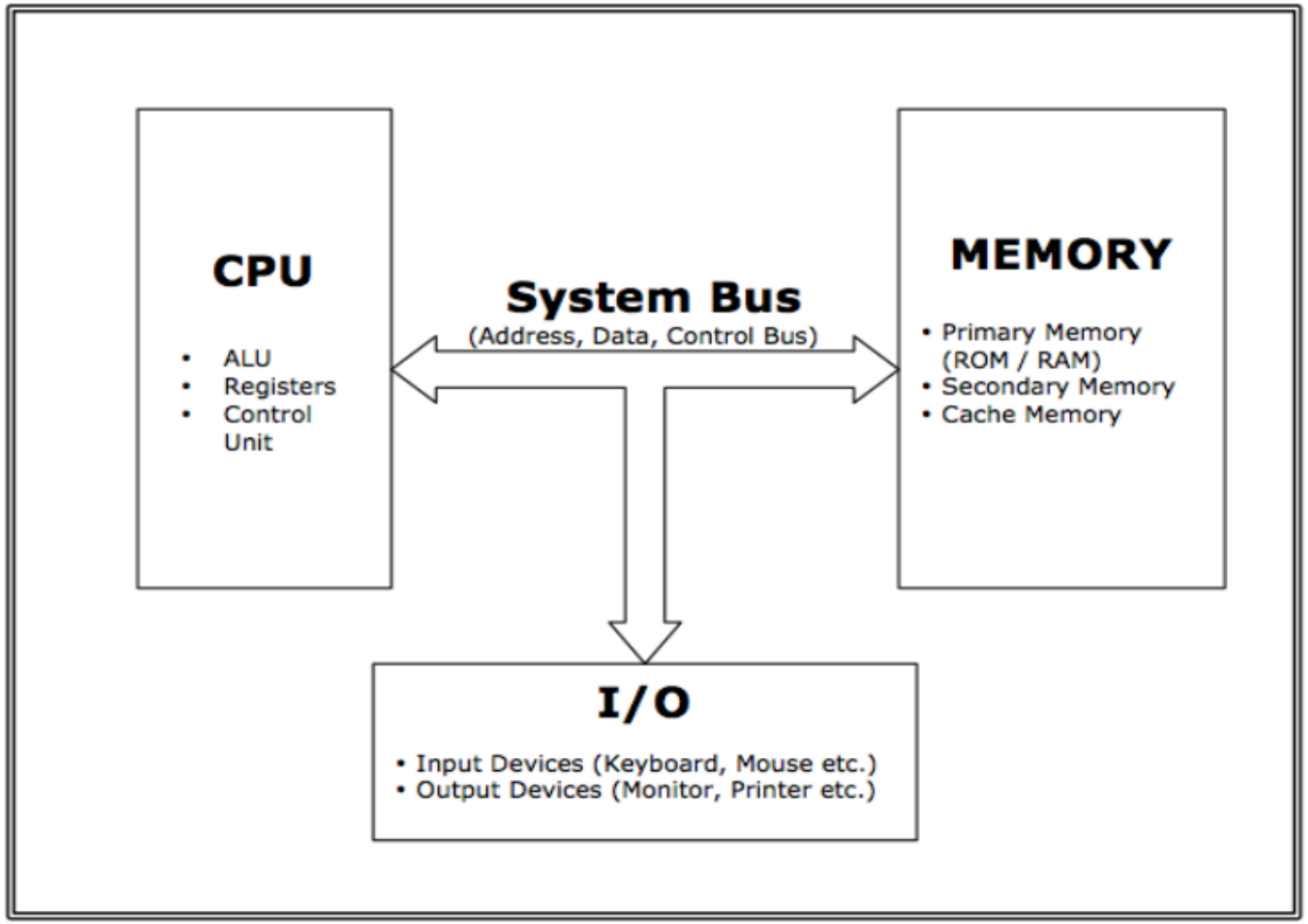
- CPU
  - Performs arithmetic and logical operations
- INPUT-OUTPUT devices (PERIPHERALS)
  - Keyboards, monitor, modem, mouse, joystick, speaker, printer, etc
- MEMORY
  - Primary-Directly accessible by the CPU
  - Secondary-external memory for storing

A typical Computer

1. Processor
2. RAM/Main memory
3. Hard disk
4. Keyboard
5. Mouse
6. Power supply
7. Motherboard
8. BIOS system
9. Monitor



COMPUTER SYSTEM



A computer system mainly consists of The CPU, Memory, I/O devices and the System bus used for interconnections

CPU

1) The Central Processing Unit is the **most important part** of the Computer.
2) It is also called the **microprocessor** or simply the **processor**.
3) It consists of the ALU, Registers, Control Unit etc.
4) All **programs are executed** in the CPU.
5) A program is a **set of instructions** stored in the memory.
6) The main function of the CPU is to **fetch, decode and execute** these instructions.
7) Instructions are **fetched from the memory** using the various **buses**.
8) Thereafter they are **decoded by the Control Unit** to analyze the Opcode.
9) Finally the instruction is **executed** to perform the **desired operation**.
10) This **execution** mainly involves the **ALU** and the **internal registers** of the processor.

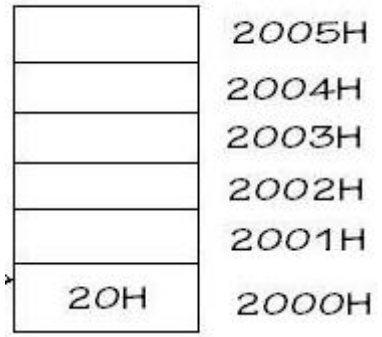
MEMORY

1) The memory is used to **store information**.
2) It mainly stores **programs and data**.
3) Memory has various **locations**.
4) Each location is identified by its own **unique address and contains some data**.
5) The most basic form of memory is called **Primary Memory**, which consists of **RAM and ROM**.
6) Then there are **secondary** storage devices such as **hard disk**.
7) There are **portable** storage devices like **CD, DVD, Pen drives** etc.

I/O

1) I/O devices are used for the **flow of information in and out** of the computer system.
2) **Input** devices such as **keyboard, mouse**, etc. are used to provide inputs into the computer.
3) They are used to **enter programs and data**.
4) **Output** devices such as **monitor and printer** are used to **generate results**.
5) Some devices such as a **touch-screen** can be used for **both input and output**.

System Bus



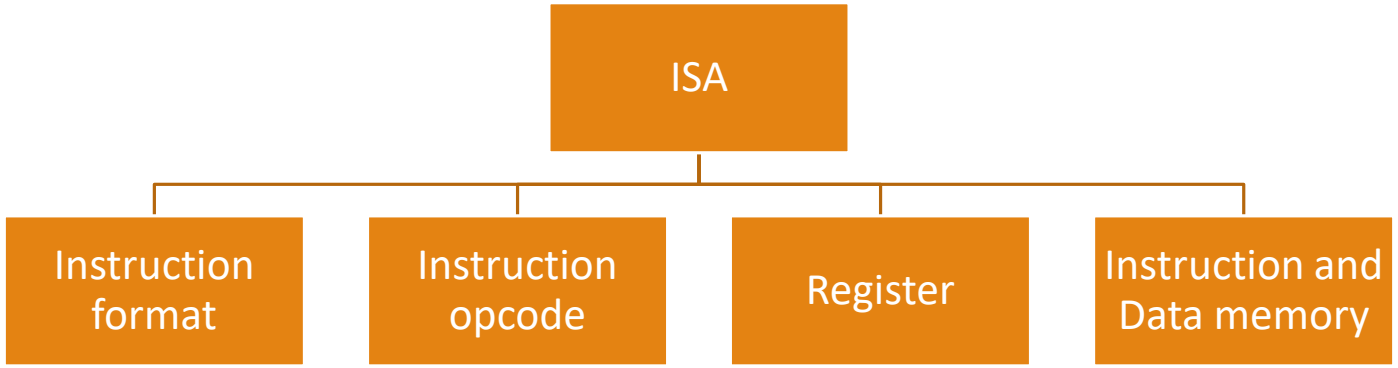
1) A bus is a set of **interconnecting lines used to carry information**.
2) **Size** of a bus means its **number of lines**.
3) An 8-bit bus has eight lines carrying **one bit each**.
4) There are three types of buses.
5) **Address Bus: It carries the address for the operation.**
   During any operation, the address bus identifies the location where the operation is performed.
   The size of the address bus determines the amount of Primary Memory that can be connected.
   Example: If address bus is 16-bit, we can connect 2^{16} = 64 KB Memory.
   Bigger the address bus, bigger is the memory.
6) **Data Bus: It carries data to and from the processor.**
   The size of data bus determines how much data can be transferred in one operation (cycle).
   Bigger the data bus, faster the processor, as it can transfer more data in one cycle.
7) **Control Bus: It Carries control signals like RD, WR etc.**
   These signals determine the kind of operation that will be performed on the system bus.

Architecture & Organization

Architecture is those attributes visible to the programmer

- Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
- e.g. Is there a multiply instruction?

Computer Architecture is also referred as Instruction set architecture (ISA) which has an algorithm to control various components.



Organization is how features are implemented by interconnecting the operational units to realize the specific architectural specifications.

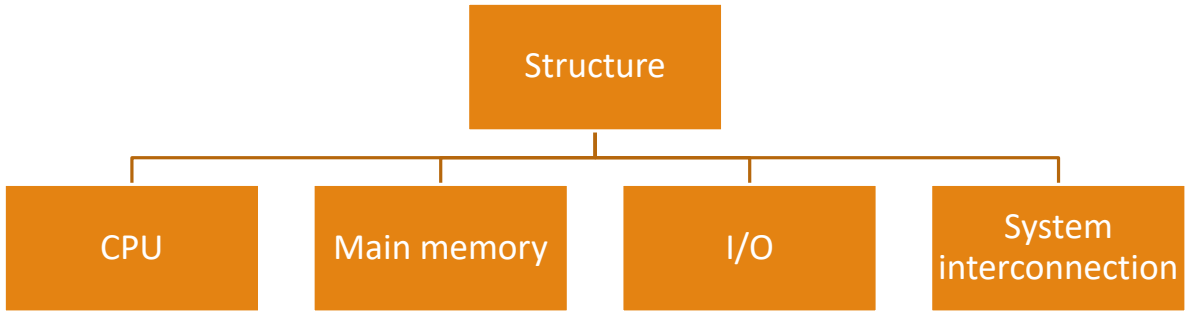
- Control signals, interfaces, memory technology.
- e.g. Is there a hardware multiply unit or is it done by repeated addition?

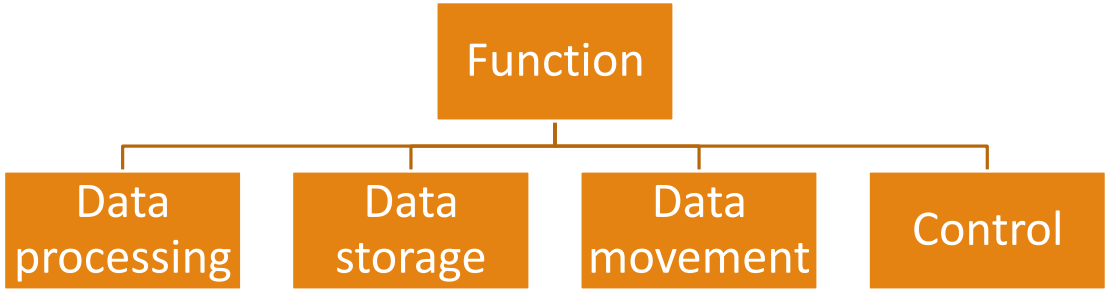
	Computer Architecture	Computer Organization
1.	Architecture describes what the computer does.	The Organization describes how it does it.
2.	Computer Architecture deals with the functional behavior of computer systems.	Computer Organization deals with a structural relationship.
3.	In the above figure, it's clear that it deals with high-level design issues.	In the above figure, it's also clear that it deals with low-level design issues.
4.	Architecture indicates its hardware.	Whereas Organization indicates its performance.
5.	As a programmer, you can view architecture as a series of instructions, addressing modes, and registers.	The implementation of the architecture is called organization.
6.	For designing a computer, its architecture is fixed first.	For designing a computer, an organization is decided after its architecture.
7.	Computer Architecture is also called Instruction Set Architecture (ISA).	Computer Organization is frequently called microarchitecture.
8.	Computer Architecture comprises logical functions such as instruction sets, registers, data types, and addressing modes.	Computer Organization consists of physical units like circuit designs, peripherals, and adders.

Structure & Function

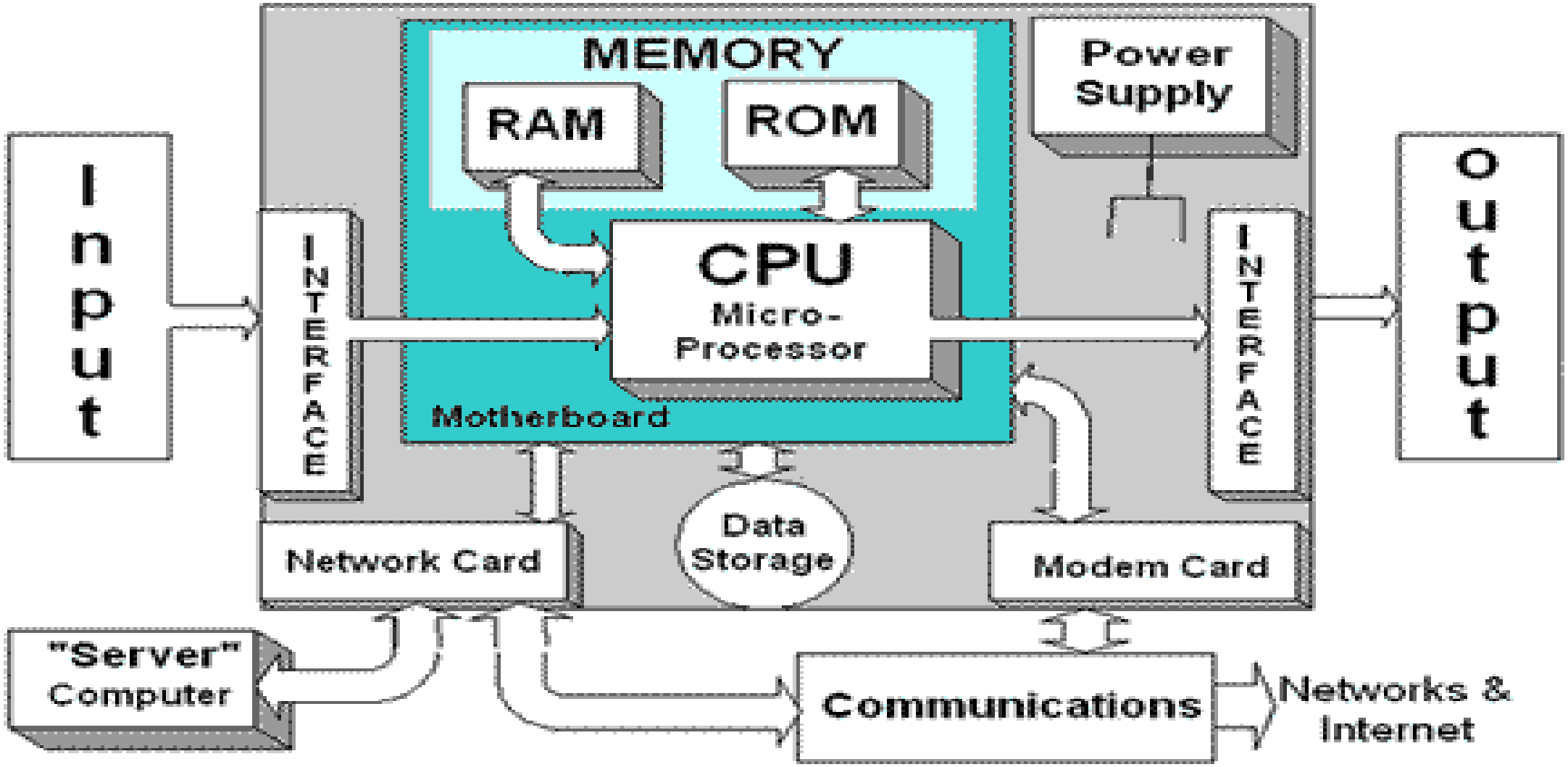
Structure is the way in which components relate to each other

Function is the operation of individual components as part of the structure





Computer Diagram



History of computers

First Generation: Vacuum Tubes

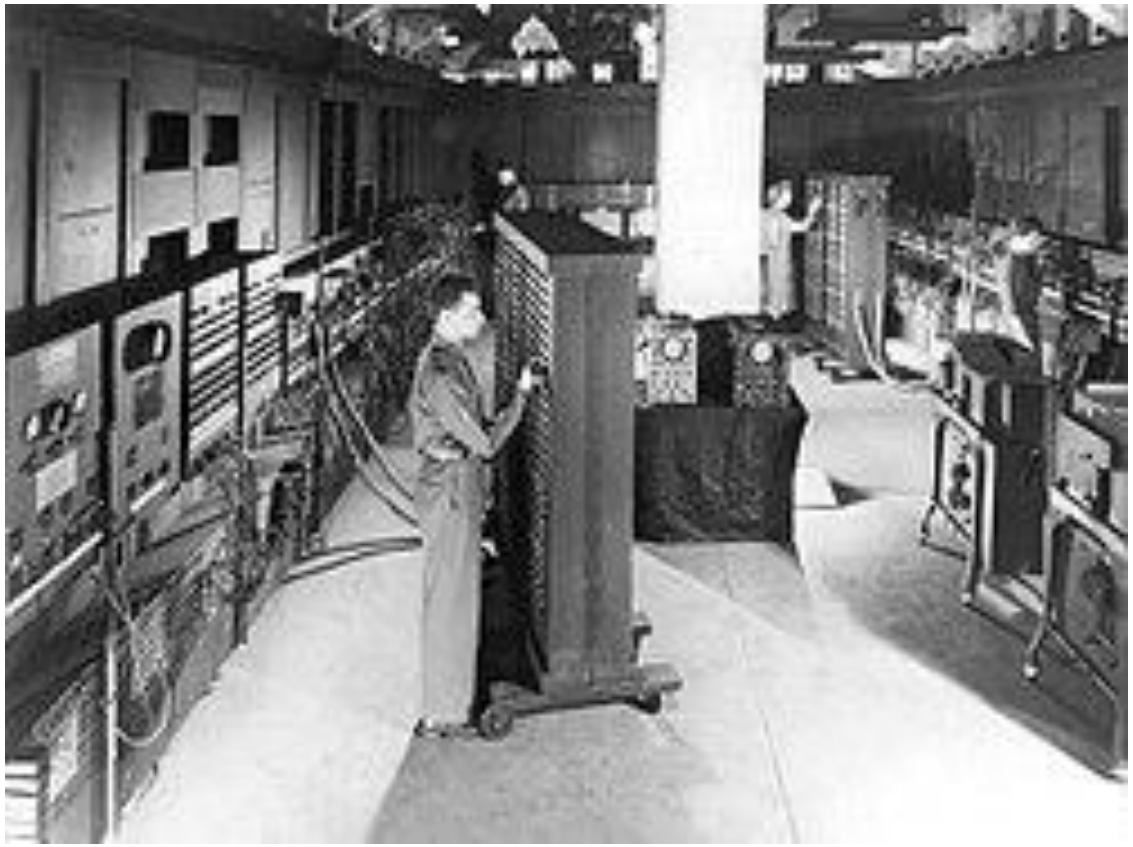
Second Generation: Transistors

Third Generation: Integrated circuits

Later generations: LSI and VLSI

First Gen: Vacuum tubes

- ENIAC (Electronic Numerical Integrator and Computer)
- Built in 1943 for WW-II
- Weighed 30 tons, occupying 1500 square feet of floor space, and containing more than 18,000 vacuum tubes
- Consumed 140 kilowatts of power
- Faster than any electromechanical computer, capable of 5000 additions per second



ENIAC

- It was a decimal computer, rather than a binary one
- Its memory consisted of 20 “accumulators,” each capable of holding a 10-digit decimal number
- A ring of 10 vacuum tubes represented each digit
- The major drawback is that it had to be programmed manually by setting switches and plugging and unplugging cables

Von Neumann machine

- Program could be represented in a form suitable for storing in memory alongside the data
- A computer could get its instructions by reading them from memory
- This program could be set or altered by setting the values of a portion of memory
- This idea, known as the *stored-program concept,* usually attributed to the mathematician John von Neumann
- Shortly, the design of a new stored program computer, referred to as the IAS computer, at the Princeton Institute for Advanced Studies
- Took 6 Years to build and is the prototype for all the subsequent general-purpose computers

Von Neumann/Turing

Stored program concept (memory comes to picture)

Main memory storing programs and data

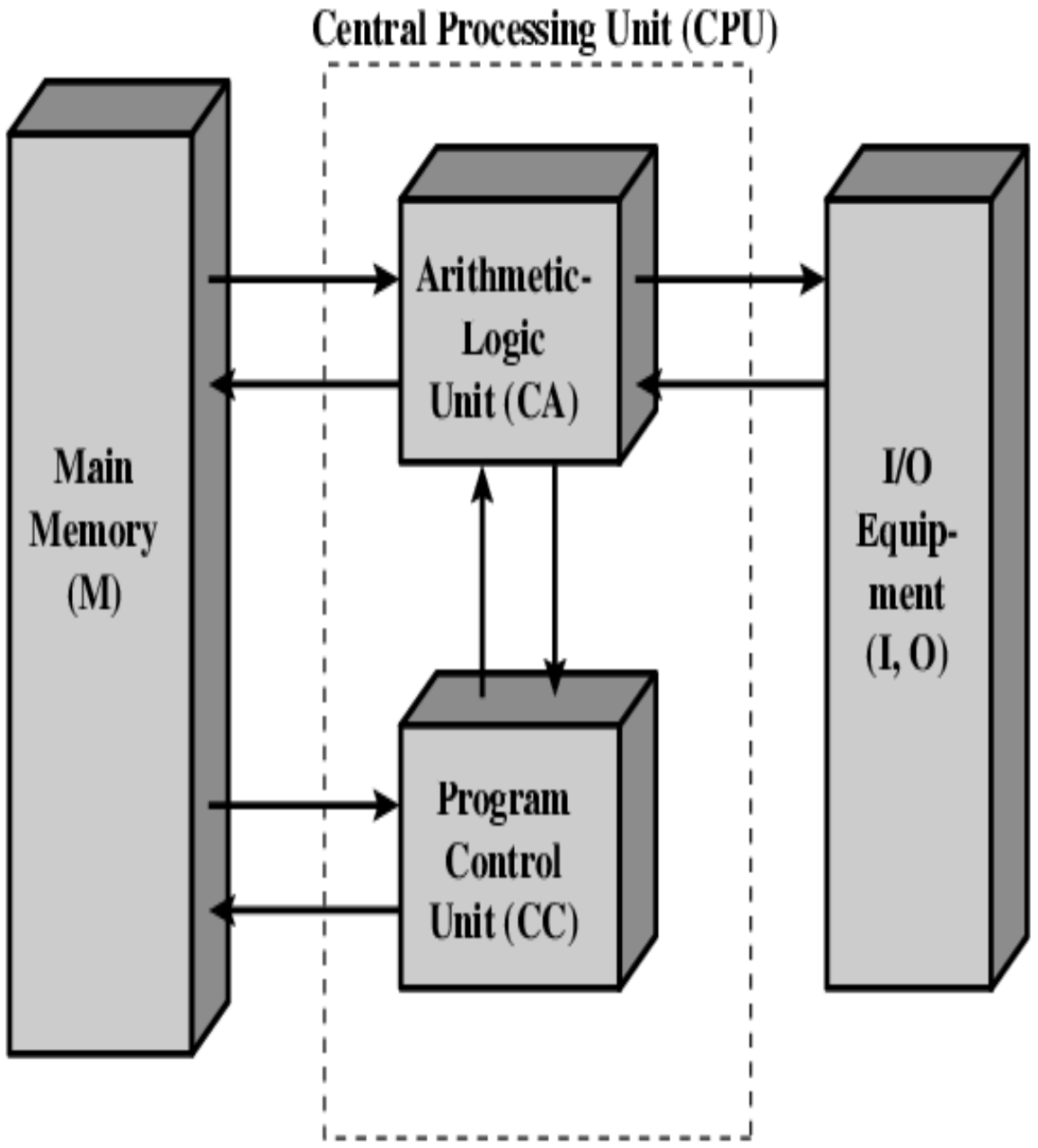
ALU operating on binary data

Control unit interpreting instructions from memory and executing

Input and output equipment operated by control unit

Princeton institute for advanced studies (IAS)

Completed 1952



CA: Central Arithmetic
CC: Central Control

Structure of von Neumann machine

Second GEN: Transistors

- Invented at Bell Labs in 1947; by 1950s → electronic revolution.
- Late 1950s → fully transistorized computers were commercially available
- Smaller, cheaper, and dissipates less heat; can be used in the same way as a vacuum tube to construct computers
- The second generation introduced more complex ALUs and CUs
- Use of high-level programming languages, and *system software* with the computer.

Table 2.3 Example members of the IBM 700/7000 Series

Model Number	First Delivery	CPU Tech-nology	Memory Tech-nology	Cycle Time (μs)	Memory Size (K)	Number of Opcodes	Number of Index Registers	Hardwired Floating-Point	I/O Overlap (Chan-nels)	Instruc-tion 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

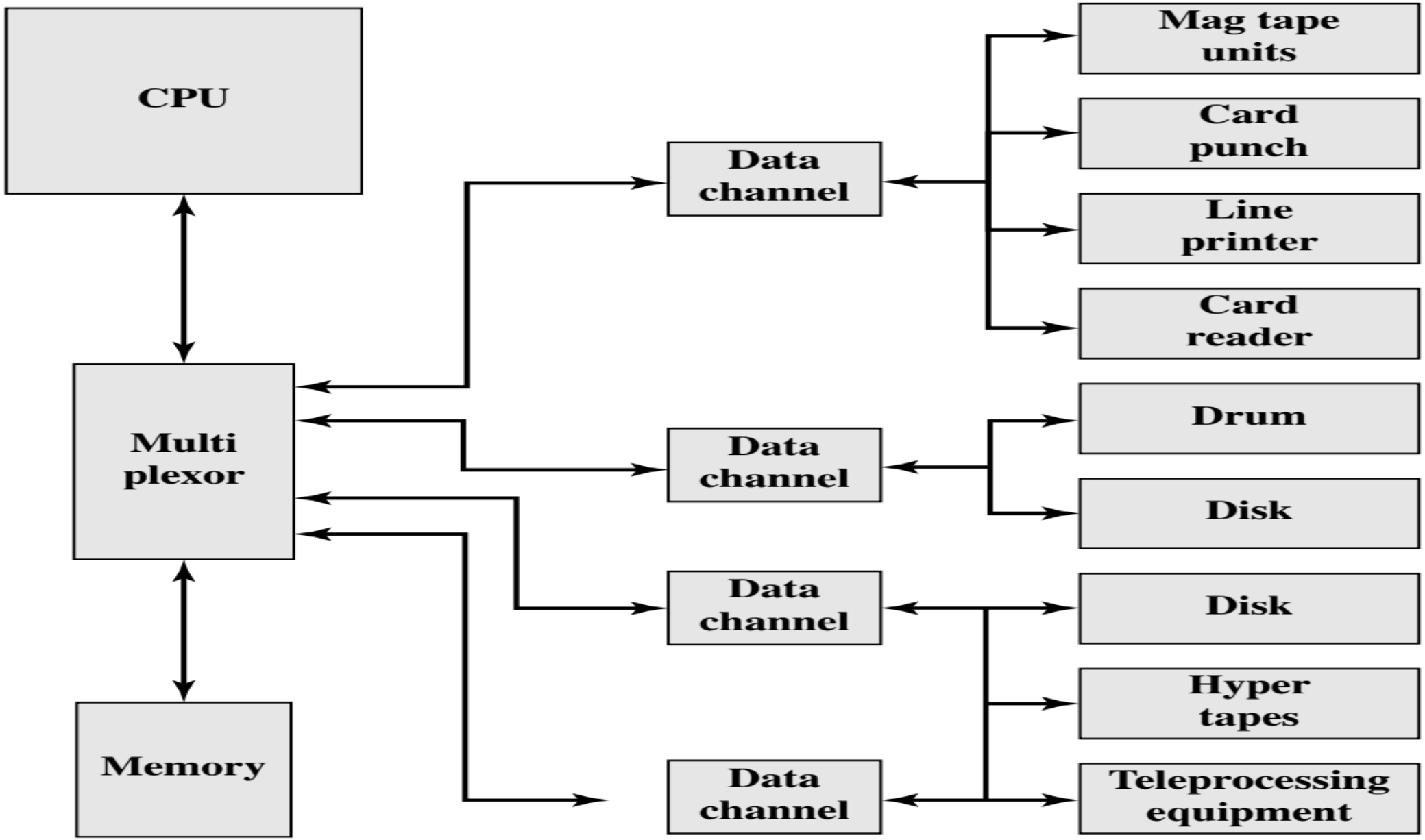


Figure 2.5 An IBM 7094 Configuration

Third gen: Integrated circuits

- A single, self-contained transistor is called a *discrete component*
- 1950s to early 1960s, electronic equipment was composed largely of discrete components—transistors, resistors, capacitors, and so on.
- Discrete components were manufactured separately → packaged in their own containers → soldered or wired together onto circuit boards → then installed in computers, oscilloscopes, and other electronic equipment.
- Whenever an electronic device needed a transistor replacement, a small piece of silicon had to be soldered to a circuit board; made the manufacturing process expensive and cumbersome.
- The Early second-generation computers contained about 10,000 transistors. This figure grew to the hundreds of thousands; manufacturing of newer, more powerful machines became increasingly difficult.

Third gen: Integrated circuits

- 1958 → invention of the Integrated circuit
- The integrated circuit exploits the fact that → components as transistors, resistors, and conductors can be fabricated from a semiconductor such as silicon.
- Fabrication of an entire circuit in a tiny piece of silicon; rather than assemble discrete components using separate pieces of silicon
- Many transistors can be produced at the same time on a single wafer of silicon; and can be connected with a process of metallization to form circuits

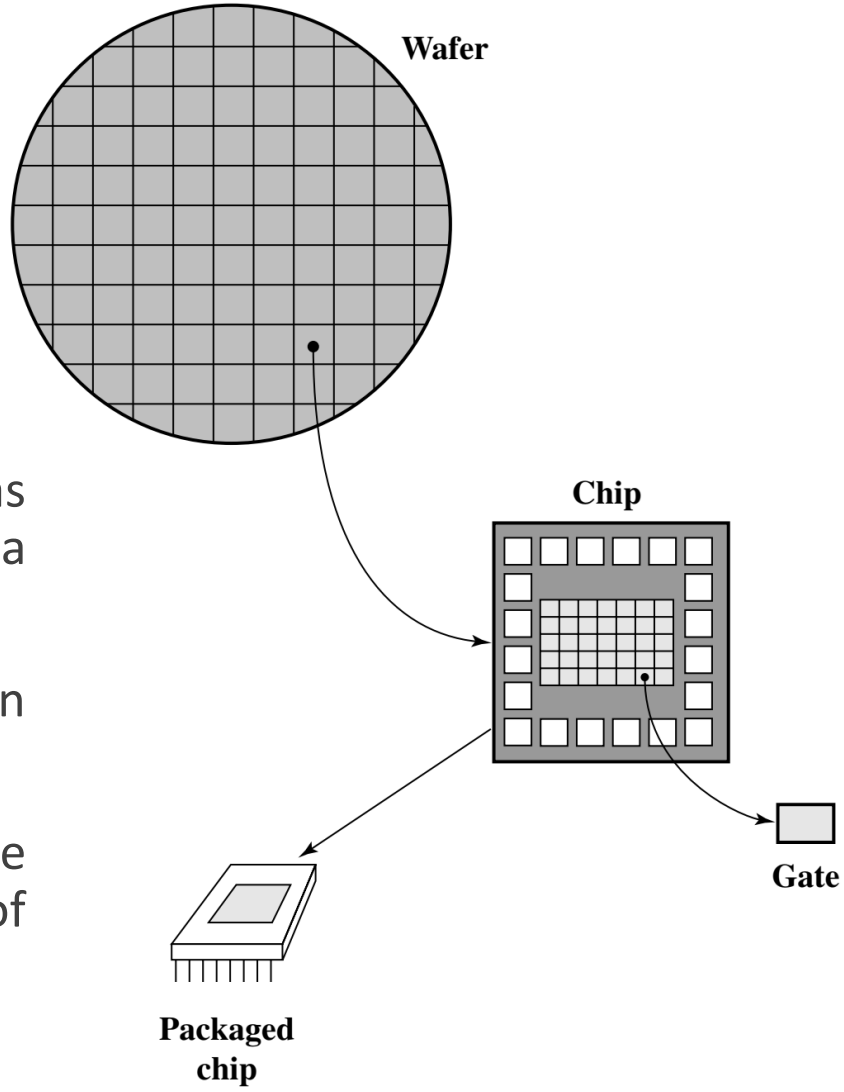


Figure 2.7 Relationship among Wafer, Chip, and Gate

Integrated circuits

- Initially, only a few gates or memory cells could be reliably manufactured and packaged together; known as SSI
- As time went on, it became possible to pack more and more components on the same chip.
- This figure reflects the famous Moore's law, which was propounded by Gordon Moore, cofounder of Intel, in 1965.

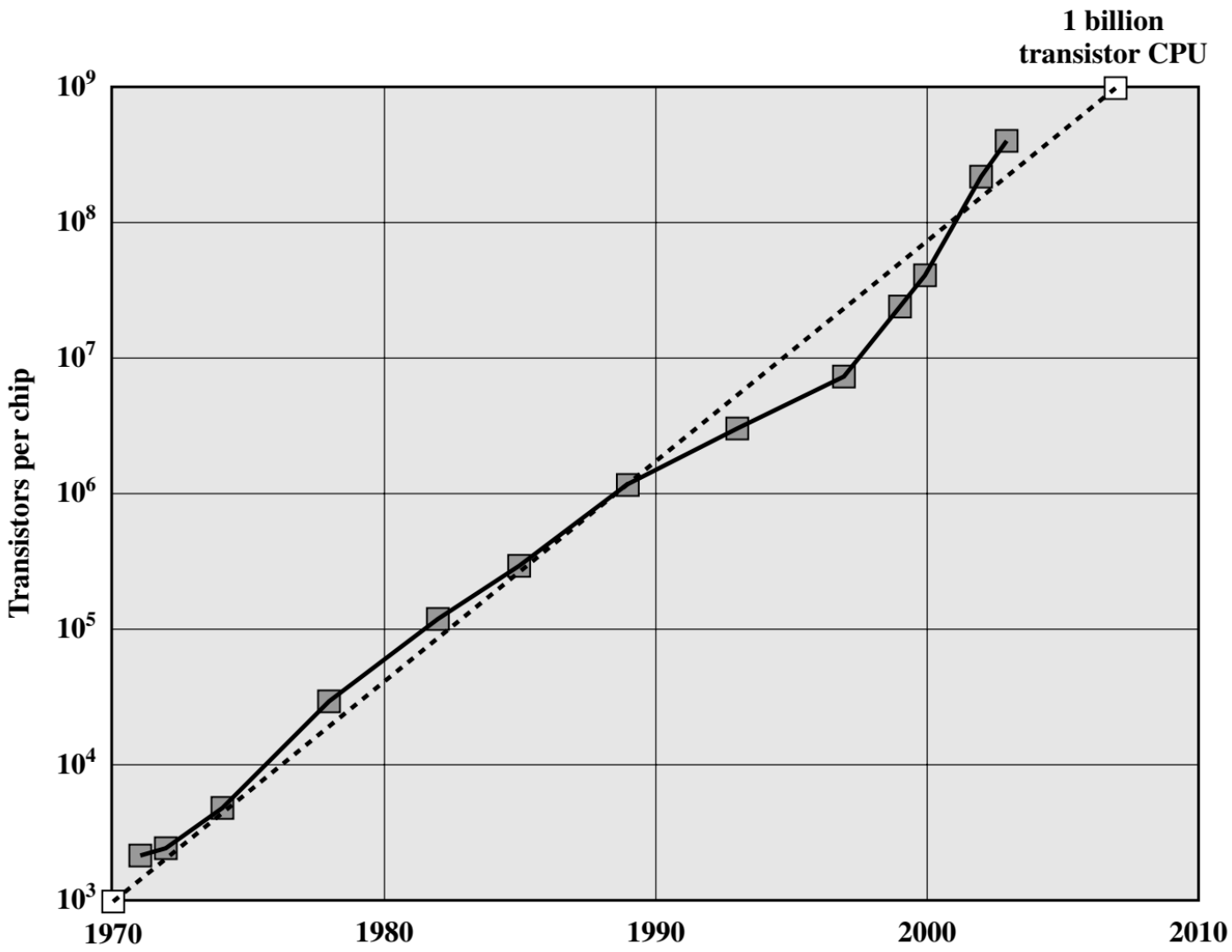


Figure 2.8 Growth in CPU Transistor Count [BOHR03]

Integrated circuits

- Moore observed that the #Transistors was doubling every year; correctly predicted that this pace would continue

- The pace continued year after year and decade after decade; it slowed to a doubling every 18 months in the 1970s but has sustained that rate ever since

- Consequences of Moore's law:
  - The cost of a chip has remained virtually unchanged during this period of rapid growth in density. This means that the cost of computer logic and memory circuitry has fallen at a dramatic rate.
  - Because logic and memory elements are placed closer together on more densely packed chips, the electrical path length is shortened, increasing operating speed.
  - The computer becomes smaller, making it more convenient to place in a variety of environments.
  - There is a reduction in power and cooling requirements.
  - The interconnections on the integrated circuit are much more reliable than solder connections. With more circuitry on each chip, there are fewer interchip connections

Later generations

Table 2.2 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

Later generations

- With the introduction of largescale integration (LSI), more than 1000 components can be placed on a single integrated circuit chip.
- Very-large-scale integration (VLSI) achieved more than 10,000 components per chip, while current ultra-large-scale integration (ULSI) chips can contain more than one million components.
- The first application of integrated circuit technology to computers was construction of the processor
- It was also found that this same technology could be used to construct memories

Later generations

- In the 1950s and 1960s, most computer memory was constructed from tiny rings of ferromagnetic materials
- Magnetized one way, a ring (called a *core*) represented a one; magnetized the other way, it stood for a zero.
- Magnetic-core memory was rather fast; it took as little as a millionth of a second to read a bit stored in memory.
- But it was expensive, bulky, and used destructive readout: The simple act of reading a core erased the data stored in it.
- It was therefore necessary to install circuits to restore the data as soon as it had been extracted.

Later generations

- Then, in 1970, Fairchild produced the first semiconductor memory.
- This chip, about the size of a single core, could hold 256 bits of memory.
- It was nondestructive and much faster than core. It took only 70 billionths of a second to read a bit.
- However, the cost per bit was higher than for that of core
- In 1974, the price per bit of semiconductor memory dropped below the price per bit of core memory.
- Following this, there has been a continuing and rapid decline in memory cost accompanied by a corresponding increase in physical memory density.
- This has led the way to smaller, faster machines with memory sizes of larger and more expensive machines from just a few years earlier.
- Developments in memory technology, together with developments in processor technology changed the nature of computers in less than a decade.

Microprocessors

- A breakthrough was achieved in 1971, when Intel developed its 4004.
- The 4004 was the first chip to contain *all* of the components of a CPU on a single chip: The microprocessor was born.
- The 4004 can add two 4-bit numbers and can multiply only by repeated addition.
- By today's standards, the 4004 is hopelessly primitive, but it marked the beginning of a continuing evolution of microprocessor capability and power.
- The next major step in the evolution of the microprocessor was the introduction in 1972 of the Intel 8008.
- This was the first 8-bit microprocessor and was almost twice as complex as the 4004.

Microprocessors

- The introduction, in 1974, of the Intel 8080.
- This was the first general-purpose microprocessor.
- The 4004 and the 8008 had been designed for specific applications, the 8080 was designed to be the CPU of a general-purpose microcomputer.
- Like the 8008, the 8080 is an 8-bit microprocessor.
- It was faster, has a richer instruction set, and has a large addressing capability.

Microprocessors

- About the same time, 16-bit microprocessors began to be developed.
- However, it was not until the end of the 1970s that powerful, general-purpose 16-bit microprocessors appeared. One of these was the 8086.
- The next step in this trend occurred in 1981, when both Bell Labs and Hewlett-Packard developed 32-bit, single-chip microprocessors.
- Intel introduced its own 32-bit microprocessor, the 80386, in 1985

Table 2.6 Evolution of Intel Microprocessors

(a) 1970s Processors

	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

(b) 1980s 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

(c) 1990s Processors

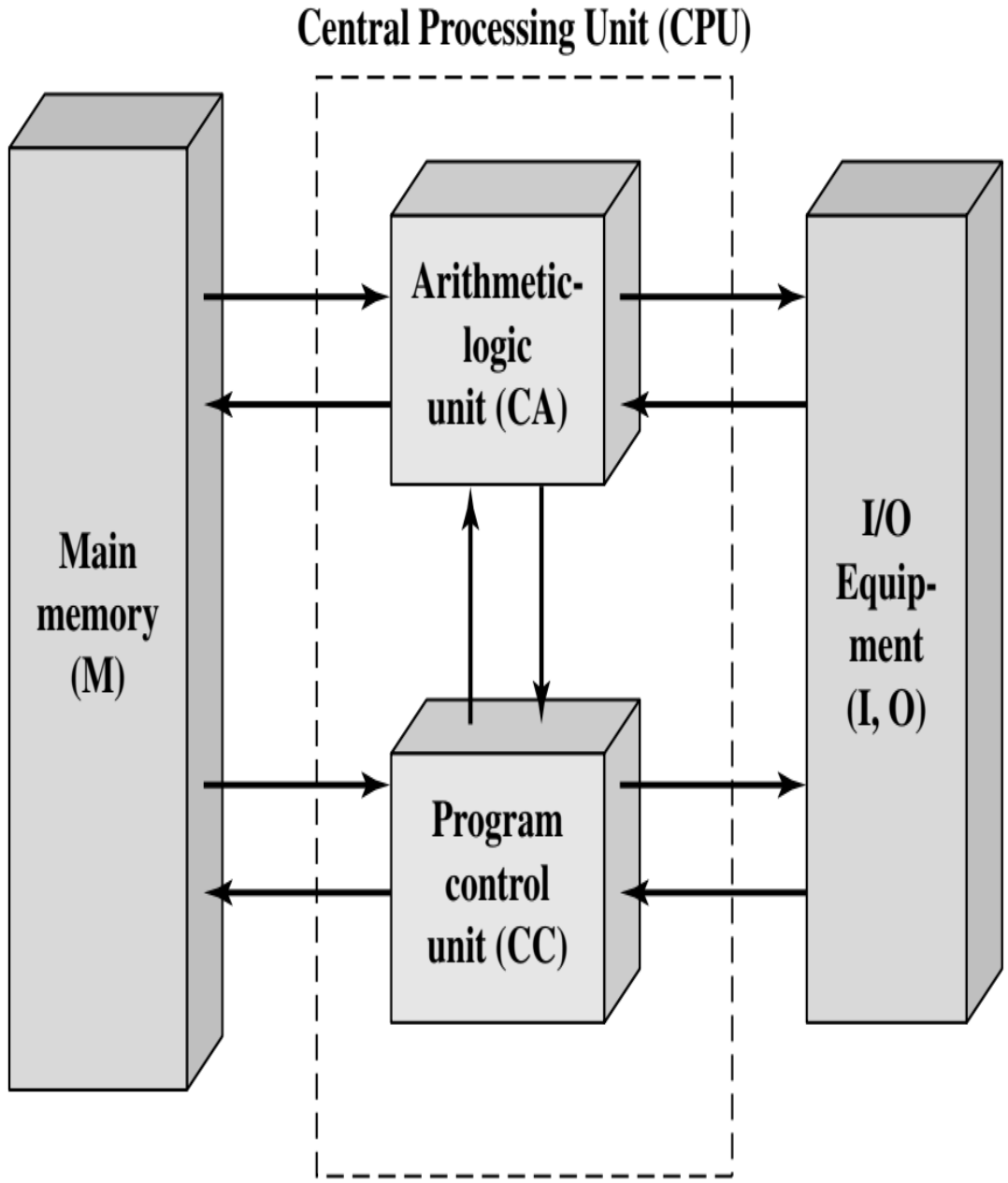
	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz–33 MHz	60 MHz–166 MHz,	150 MHz–200 MHz	200 MHz–300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (μm)	1	0.8	0.6	0.35
Addressable 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

(d) Recent Processors

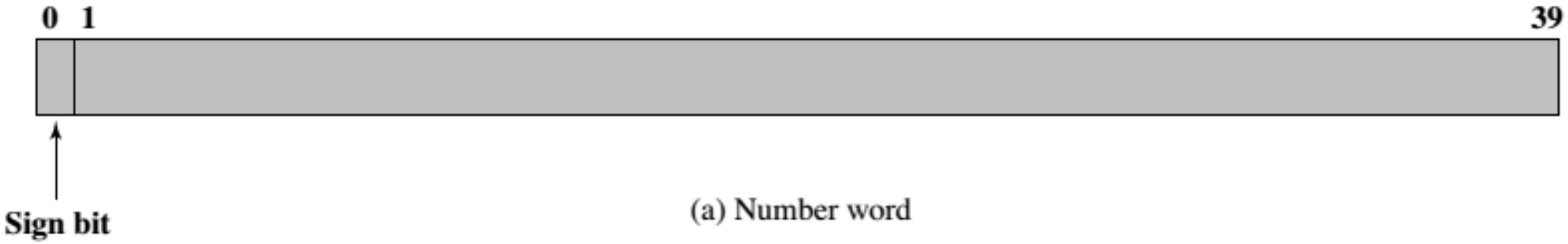
	Pentium III	Pentium 4	Core 2 Duo	Core 2 Quad
Introduced	1999	2000	2006	2008
Clock speeds	450–660 MHz	1.3–1.8 GHz	1.06–1.2 GHz	3 GHz
Bus sidth	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	820 million
Feature size (nm)	250	180	65	45
Addressable memory	64 GB	64 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	6 MB L2

IAS Computer

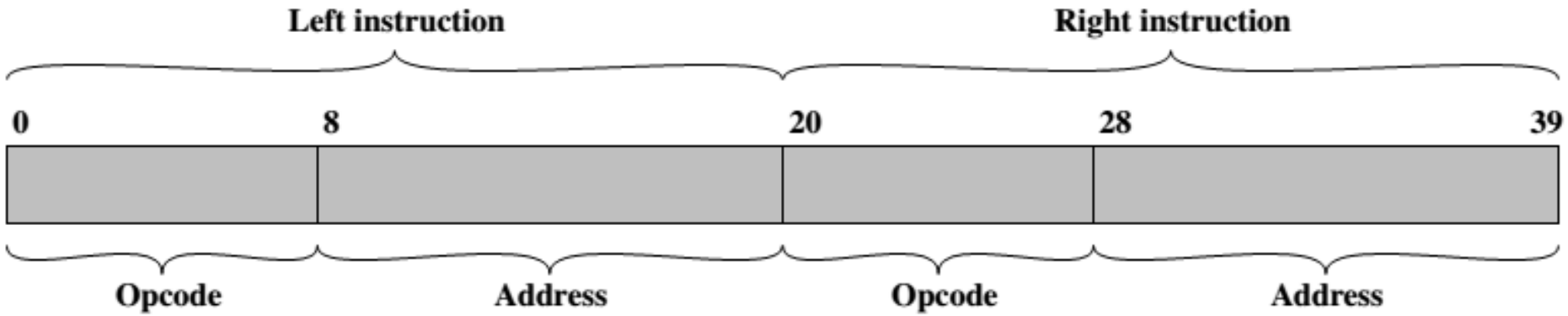
- A main memory, which stores both data and instructions
- An arithmetic and logic unit (ALU) capable of operating on binary data
- A control unit, which interprets the instructions in memory and causes them to be executed
- Input and output (I/O) equipment operated by the control unit



EXAMPLE



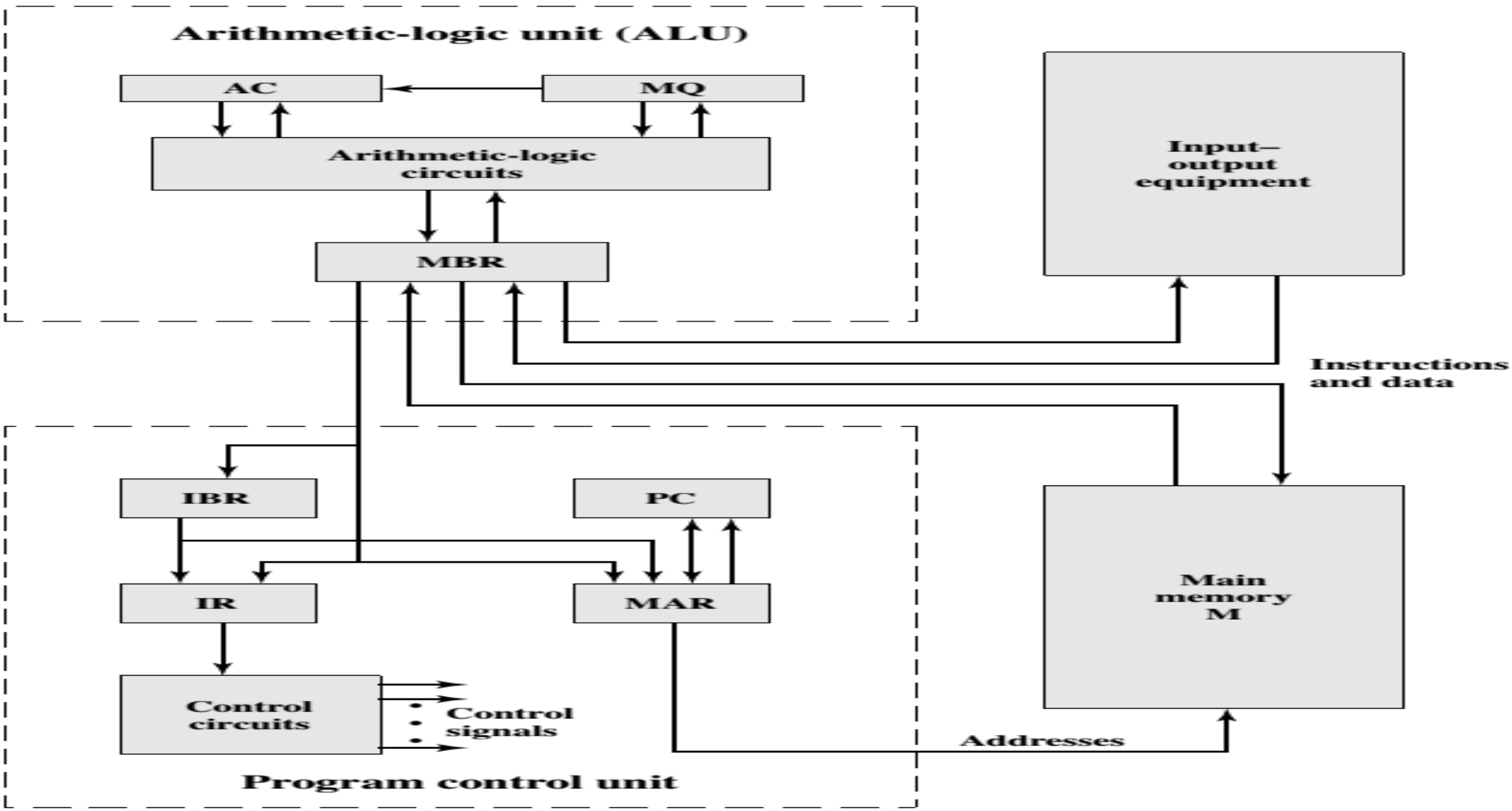
(a) Number word



(b) Instruction word

Working of IAS

- The memory of the IAS consists of 1000 storage locations, called words, of 40 binary digits (bits) each.
- Both data and instructions are stored there.
- Numbers are represented in binary form, and each instruction is a binary code.
- Each number is represented by a sign bit and a 39-bit value.
- 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).


Arithmetic-logic unit (ALU)
AC
MQ
Arithmetic-logic circuits
MBR
Input–output equipment
Instructions and data
IBR
PC
IR
MAR
Main memory M
Control circuits
Control signals
Addresses
Program control unit

Registers in IAS

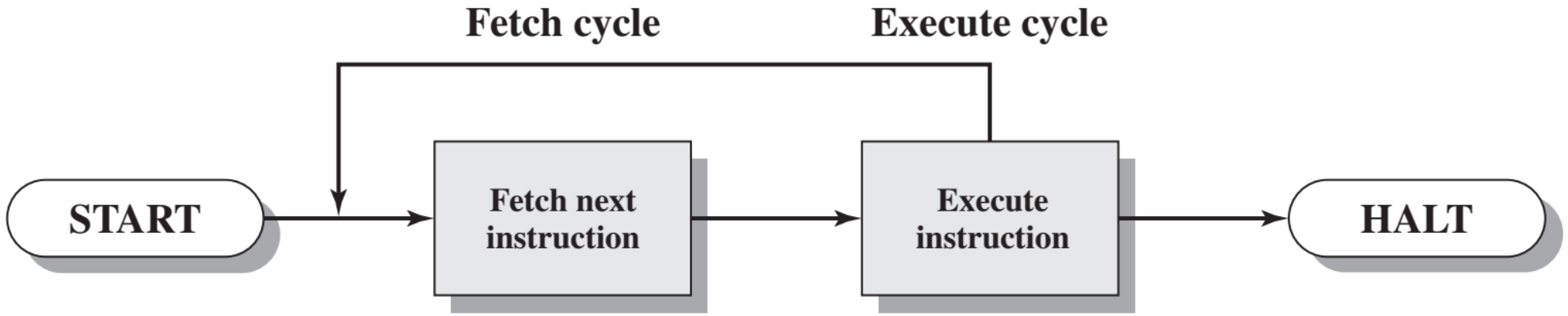
- **Memory buffer register (MBR):** Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.
- **Memory address register (MAR):** Specifies the address in memory of the word to be written from or read into the MBR.
- **Instruction register (IR):** Contains the 8-bit opcode instruction being executed.
- **Instruction buffer register (IBR):** Employed to hold temporarily the righthand 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 hold temporarily operands and results of ALU operations.

Registers in IAS

- **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 hold temporarily the righthand 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 hold temporarily operands and results of ALU operations.

Computer function

- Basic function performed by the computer is execution of the program → set of instructions stored in memory
- Instruction processing consists of two steps
  - Processor reads / Fetches
  - Executing the fetched instruction
- Processing required for a single instruction is known as instruction cycle



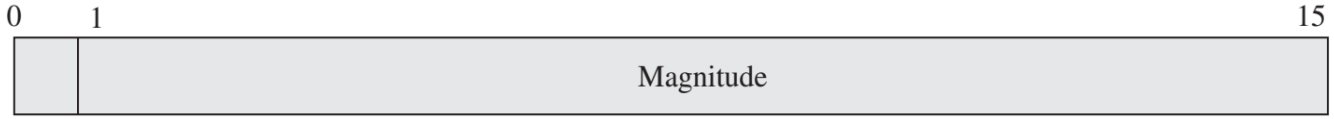
Instruction fetch and execute

- At the beginning of each instruction cycle, the processor fetches an instruction from memory
- A register called the program counter (PC) holds the address of the instruction to be fetched next
- Unless told otherwise, the processor always increments the PC after each instruction fetch so that it will fetch the next instruction in sequence i.e. the instruction located in the next higher memory address
- The fetched instruction is loaded into a register in the processor known as the instruction register (IR)
- The instruction contains bits that specify the action the processor needs to take
- The processor interprets the instruction and performs the required action

Example

- Suppose a processor contains a single data register, called an accumulator (AC)
- Both instructions and data are 16 bits long
- The instruction format provides 4 bits for the opcode → 16 opcodes and 4096 (4K) memory

(a) Instruction format



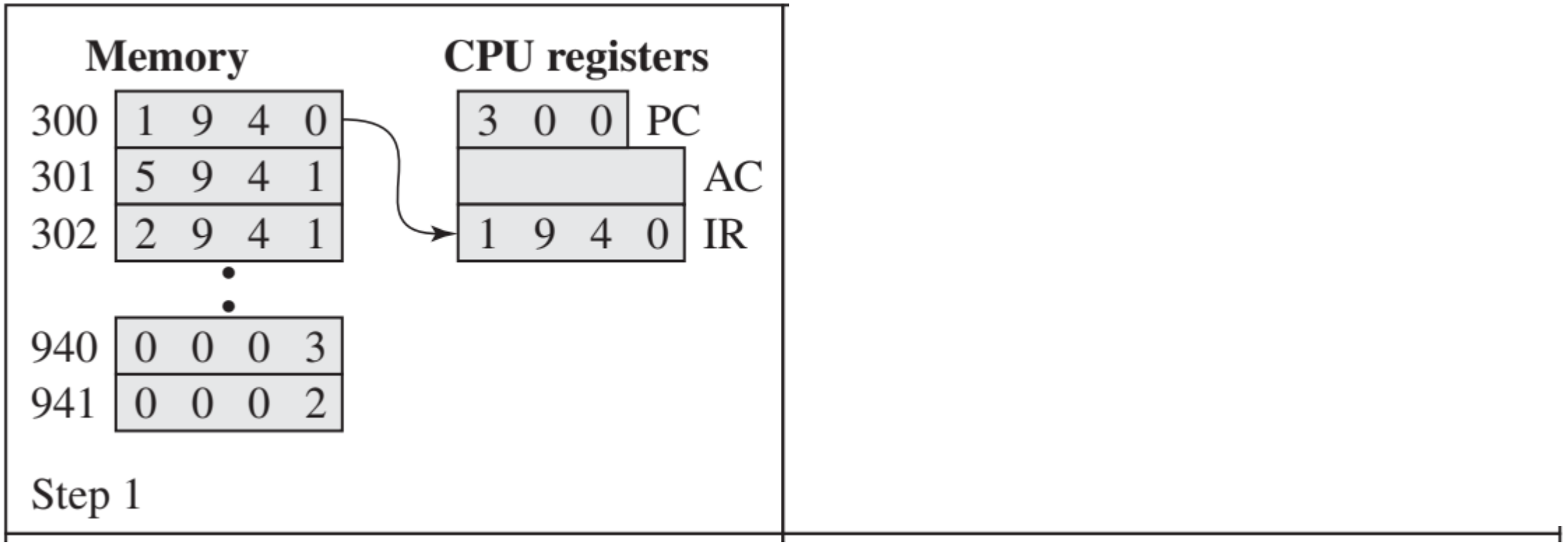
(b) Integer format

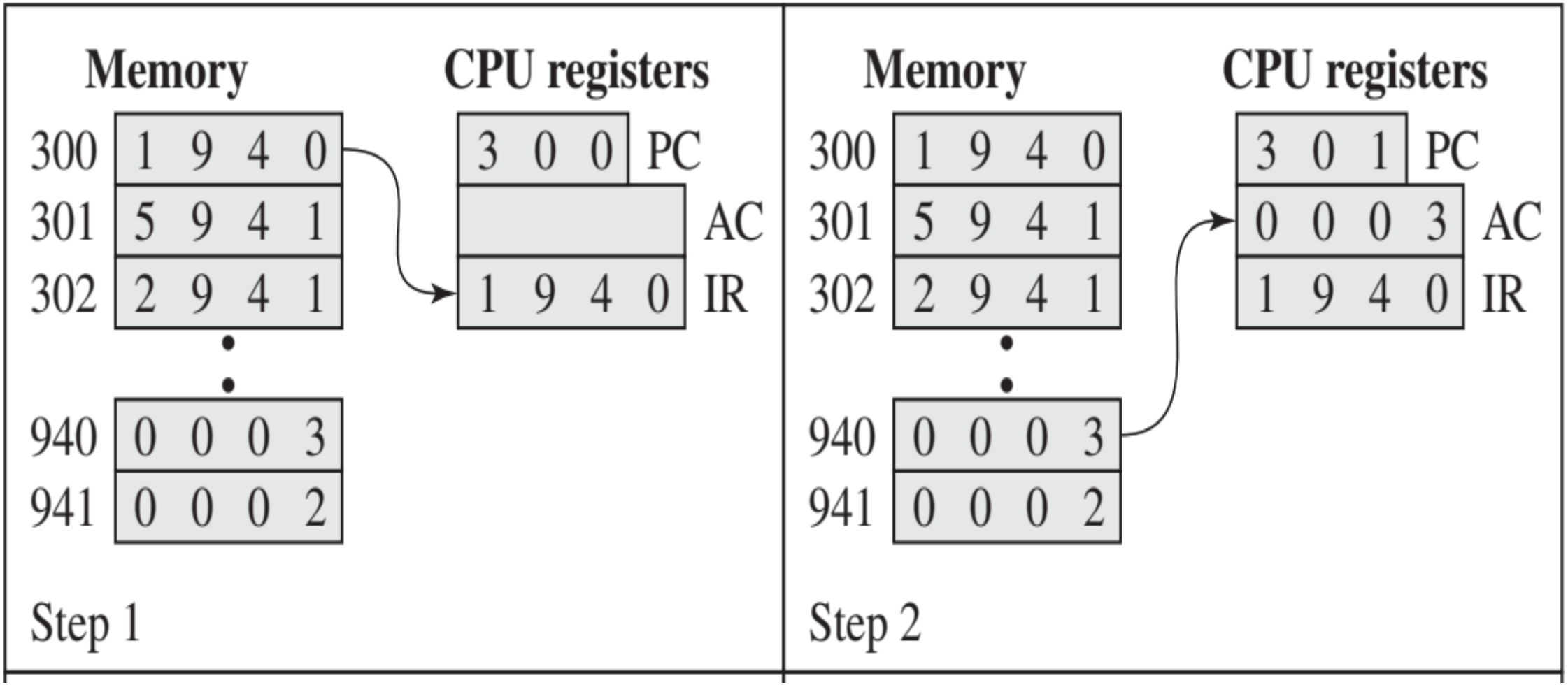
Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

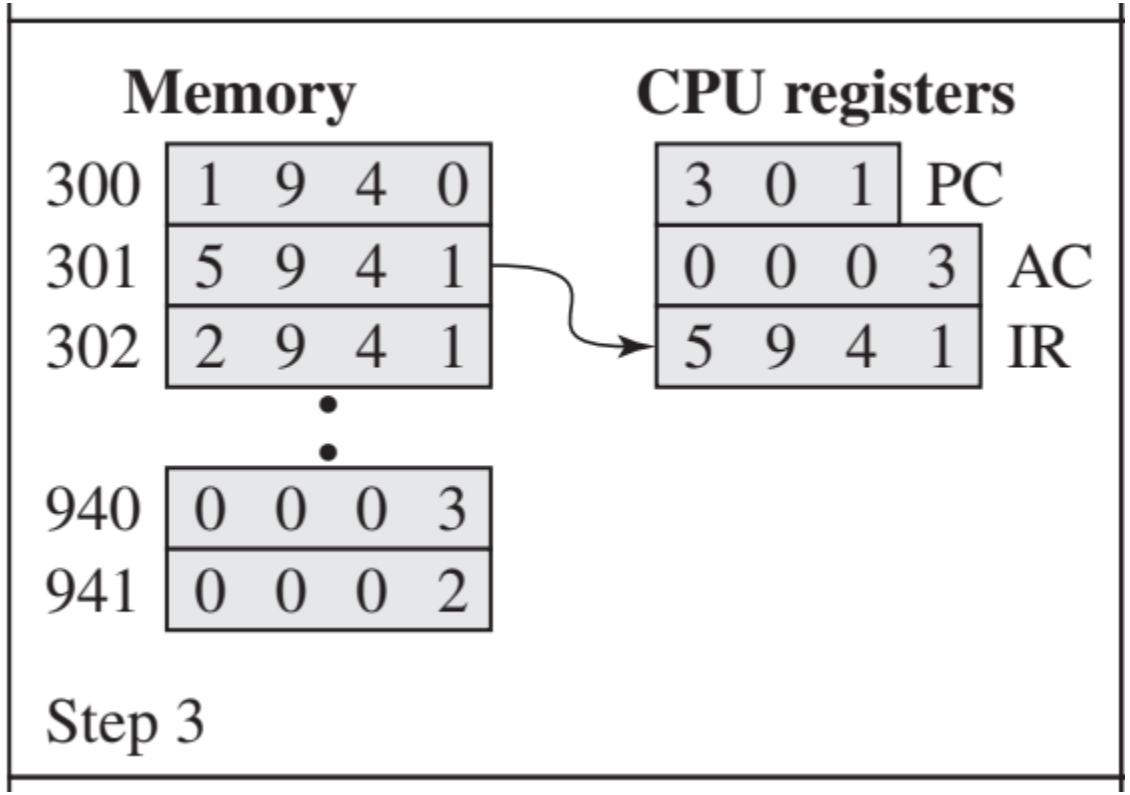
(c) Internal CPU registers

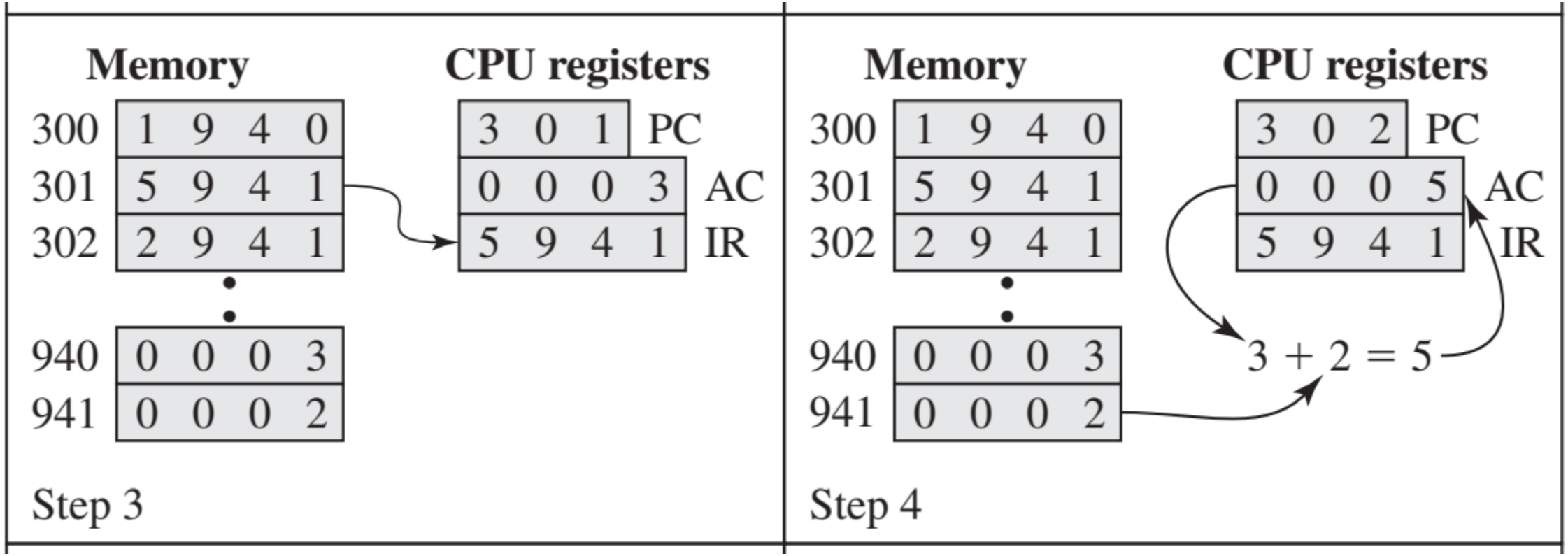
0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

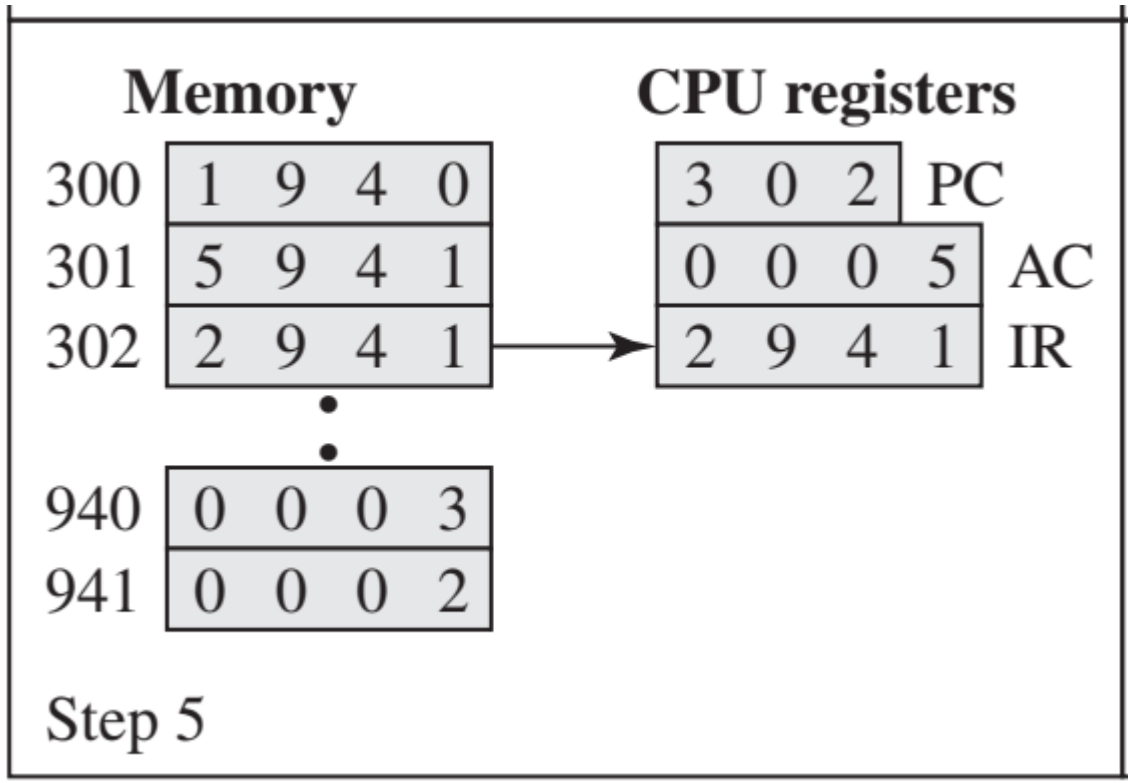
(d) Partial list of opcodes

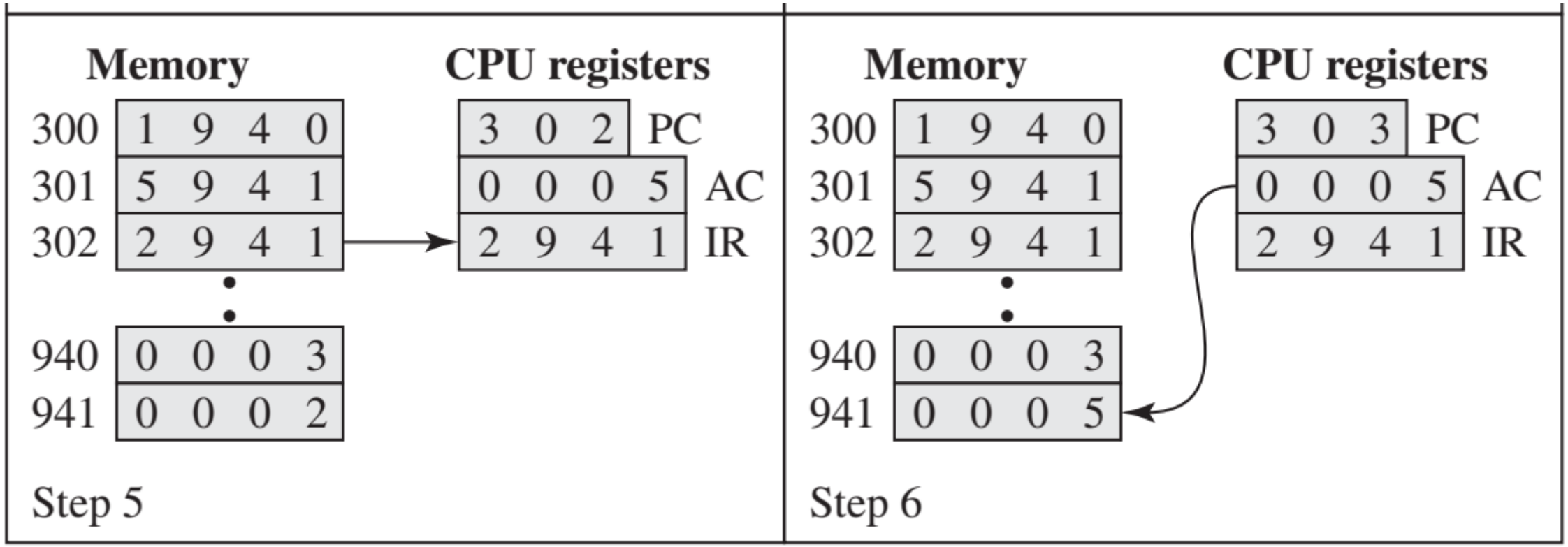

Memory
CPU registers
300
1 9 4 0
301
5 9 4 1
302
2 9 4 1
940
0 0 0 3
941
0 0 0 2
3 0 0
PC
AC
1 9 4 0
IR
Step 1


Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 0 PC
AC
1 9 4 0 IR
Step 1
Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
1 9 4 0 IR
Step 2


Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
5 9 4 1 IR
Step 3

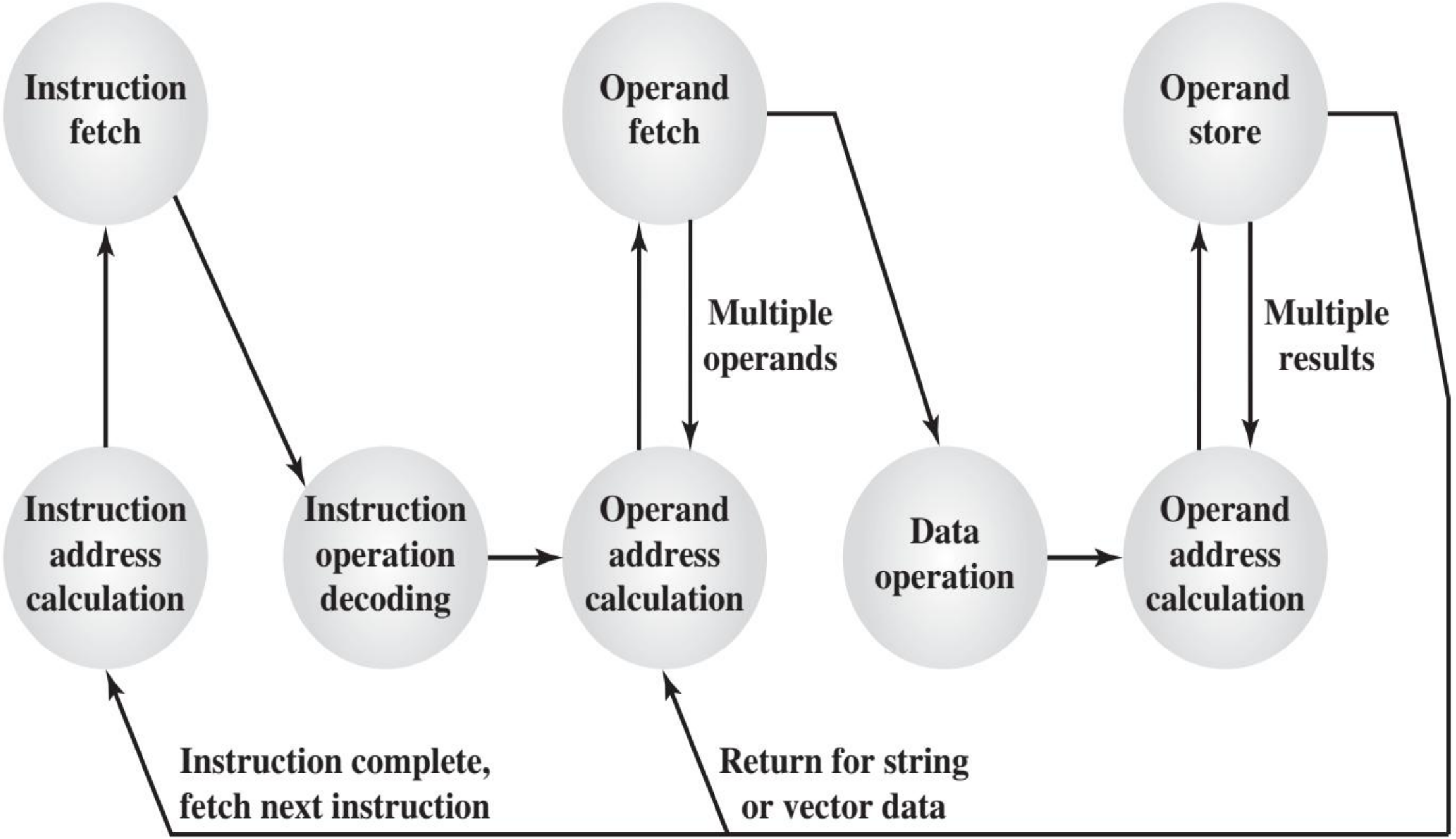

Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 1 PC
0 0 0 3 AC
5 9 4 1 IR
Step 3
Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
5 9 4 1 IR
3 + 2 = 5
Step 4


Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
2 9 4 1 IR
Step 5


Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 2
3 0 2 PC
0 0 0 5 AC
2 9 4 1 IR
Step 5
Memory
CPU registers
300 1 9 4 0
301 5 9 4 1
302 2 9 4 1
940 0 0 0 3
941 0 0 0 5
3 0 3 PC
0 0 0 5 AC
2 9 4 1 IR
Step 6

Example

- In this example, three instruction cycles, each consisting of a fetch cycle and an execute cycle, are needed to add the contents of location 940 to the contents of 941.
- With a more complex set of instructions, fewer cycles would be needed.
- Some older processors, for example, included instructions that contain more than one memory address.
- For example, the PDP-11 processor expressed symbolically as ADD B,A
  - Fetch the ADD instruction.
  - Read the contents of memory location A into the processor.
  - Read the contents of memory location B into the processor. In order that the contents of A are not lost, the processor must have at least two registers for storing memory values, rather than a single accumulator.
  - Add the two values.
  - Write the result from the processor to memory location A.


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

States

- **Instruction address calculation (IAC):** Determine the address of the next instruction to be executed.
- **Instruction fetch (if):** Read instruction from its memory location into the processor.
- **Instruction operation decoding (iod):** Analyze instruction to determine type of operation to be performed and operand(s) to be used.
- **Operand address calculation (oac):** If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand.
- **Operand fetch (of):** Fetch the operand from memory or read it in from I/O.
- **Data operation (do):** Perform the operation indicated in the instruction.
- **Operand store (os):** Write the result into memory or out to I/O.

States

- The **oac** state appears twice, because an instruction may involve a read, a write, or both.
- Example → ADD A,B results in the following sequence of states: iac, if, iod, oac, of, oac, of, do, oac, os.
- A single instruction can specify an operation to be performed on an array of numbers or a string of characters → involves repetitive operand fetch and store operations

IAS structure

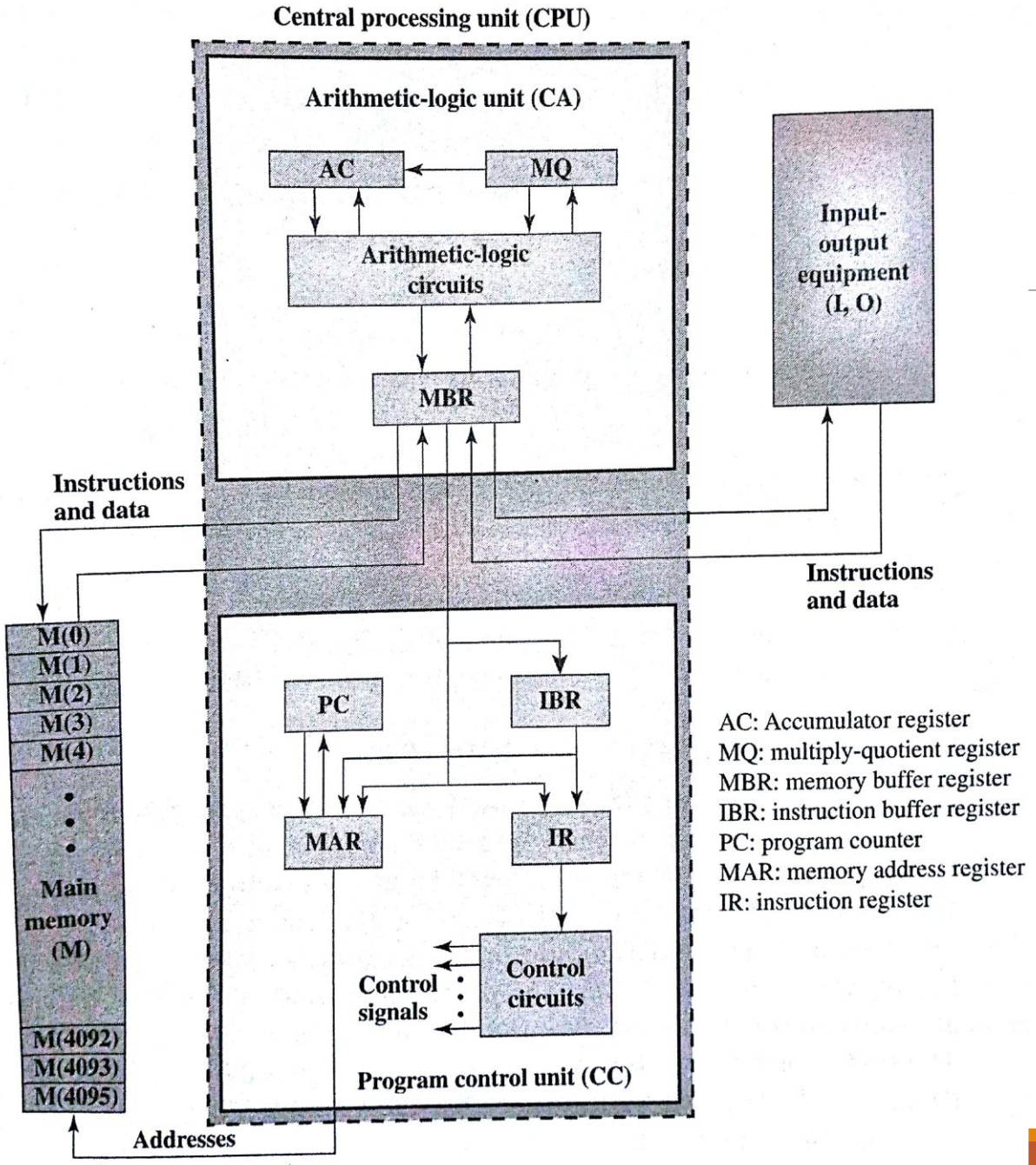
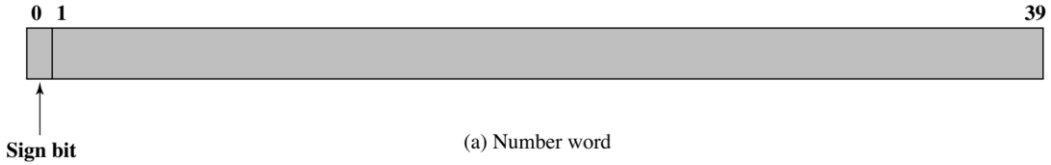
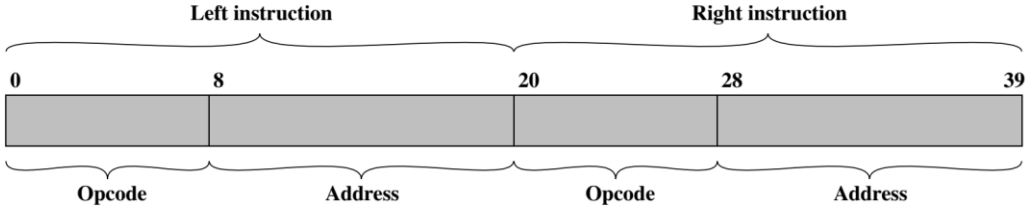


Figure 1.6 IAS Structure





Type	Opcode	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)
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

P1 Given the memory contents of the IAS computer shown below,

Address	Contents
08A	010FA210FB
08B	010FA0F08D
08C	020FA210FB

show the assembly language code for the program, starting at address 08A. Explain what this program does.

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)
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

P1

This program will store the absolute value of content at memory location 0FA into memory location 0FB.

Address	Contents
08A	LOAD M(0FA)
	STOR M(0FB)
08B	LOAD M(0FA)
	JUMP +M(08D)
08C	LOAD –M(0FA)
	STOR M(0FB)
08D	

P2

On the IAS, what would the machine code instruction look like to load the contents of memory address 2?

How many trips to memory does the CPU need to make to complete this instruction during the instruction cycle?

P2

OPCODE	OPERAND
00000001	000000000010

First, the CPU must make access memory to fetch the instruction. The instruction contains the address of the data we want to load. During the execute phase accesses memory to load the data value located at that address for a total of two trips to memory.

Problem

Write a program to add 5 numbers located at consecutive memory locations starting at address 500. Add another 5 numbers located at consecutive memory locations starting at address 600. Subtract the lower value from the higher value and store it at location having address 700.

Sol:

800: LOAD M(500) [Left Instr]

800: ADD M(501) [Right Instr]

801: ADD M(502) [Left Instr]

801: ADD M(503) [Right Instr]

802: ADD M(504) [Left Instr]

802: STOR M(701) [Right Instr]

803: LOAD M(600) [Left Instr]

803: ADD M(601) [Right Instr]

804: ADD M(602) [Left Instr]

804: ADD M(603) [Right Instr]

805: ADD M(604) [Left Instr]

805: SUB M(701) [Right Instr]

806: JUMP+ M(807,20:39) [Left Instr]

806: STOR M(702) [Right Instr]

807: LOAD -M(702) [Left Instr]

807: STOR M(700) [Right Instr]

IAS computer – Sample problem: There are 5 values at memory location 500. Find the greatest number among these values and store it in 700 mem location.

600 | 8

500	7
501	6
502	8
503	2
504	9

700 | 9

Ans:-

800: Load M(500) → AC = 7

800: SUB M(501) → AC = 7 − 6, AC = 1

801: Jump+ M(802, 20:39)

~~801: LOAd M(501)~~

~~802: Jump M(803, 0:19)~~

802: Load M(500) → AC = 7

803: Store M(600) → M(600) ⇐ AC 7

803: SUB M(502) → AC = 7 − 8 = −1

804: Jump+ M(805, 20:39)

804: Load M(502) → AC = 8

805: Jump M(806, 0:19)

805: Load M(600) ✗

806: Store M(600) → M(600) = 8

807: SUB M(503) → AC = 8 − 2 = 6

807: Jump+ M(808, 20:39)

~~8087: Load M(503)~~ ✗

808: Jump M(809, 0:19) ✗

808: Load M(600) AC = 8

809: Store M(600) M(600) = 8

809: SUB M(504) → AC = 8 − 9 = −1

810: Jump+ M(811, 20:39)

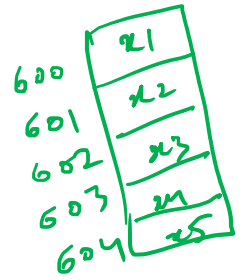
810: Load M(504) AC = 9

811: Jump M(812, 0:19)

811: Load M(600) ✗

812: Store M(700) → M(700) = AC = 9

IAS Computer - Sample problem 2:- Write a program to multiply 5 nos. stored consecutively at mem location 600 and store the results in mem location 700.



Ans:-

```
900: Load  M(600), MQ   (LI)
900: MUL   M(601)       (RI)
901: MUL   M(602)       (LI)
901: MUL   M(603)       (RI)
902: MUL   M(604)       (LI)
902: Store M(700)       (RI)
903: Load  MQ           (LI)
903: Store M(701)       (RI)
```

Q: Write an assembly language program for the IAS computer to find the MAC operation of 5 nos. located at 600h mem. location.

MAC → Multiply & Accumulate.

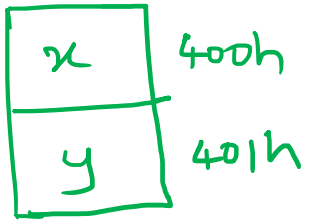
Ans:-

```
900: LOAD  MQ, M(600)
     MUL   M(601)
901: LOAD  MQ
     MUL   M(602)
902: STOR  M(700)
     LOAD  MQ
903: ADD   M(700)
     STOR  M(700)
904: LOAD  MQ, M(700)
     MUL   M(603)
905: LOAD  MQ
     ADD   M(700)
906: STOR  M(700)
     LOAD  MQ, M(700)
907: MUL   M(604)
     LOAD  MQ
908: ADD   M(700)
     STOR  M(700)
```

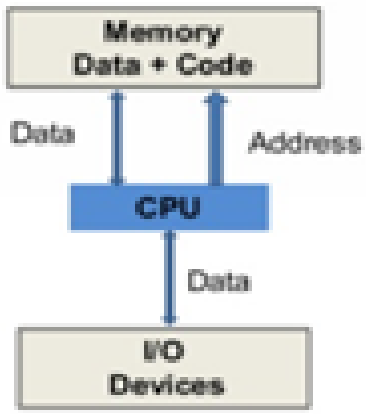
Assumption:- MUL instruction is assumed to produce only 40-bit values.

Q:- Write an ALP for the IAS Computer to find the largest among 2 nos.

Ans:-

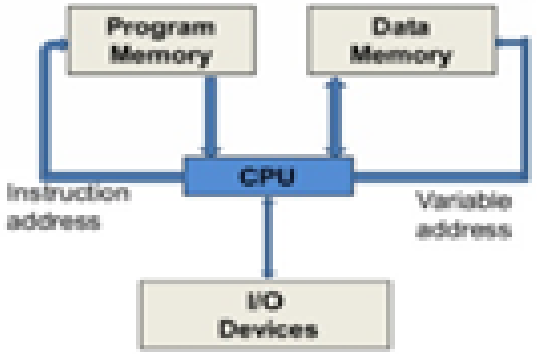



```
900: LOAD   M(400)
     SUB    M(401)
901: JumpT  M(903, 0:19)
     LOAD   M(401)
902: STOR   M(700)
     Jump   M(904 : 0:19)
903: LOAD   M(400)
     STOR   M(700)
904: EXIT
```



Von Neumann Machine

VON NEUMANN ARCHITECTURE	HARVARD ARCHITECTURE
It is ancient computer architecture based on stored program computer concept.	It is modern computer architecture based on Harvard Mark I relay based model.
Same physical memory address is used for instructions and data.	Separate physical memory address is used for instructions and data.
There is common bus for data and instruction transfer.	Separate buses are used for transferring data and instruction.
Two clock cycles are required to execute single instruction.	An instruction is executed in a single cycle.
It is cheaper in cost.	It is costly than von neumann architecture.
CPU can not access instructions and read/write at the same time.	CPU can access instructions and read/write at the same time.
It is used in personal computers and small computers because we need to change the program in RAM.	It is used in micro controllers and signal processing. In these applications the program is already dumped in to the ROM.



Harvard Machine

CISC and RISC refer to design principles and techniques, T=(N*S)/R

RISC

1. RISC: Reduced instruction set computers.

2. Simple instructions require a small number of basic steps to execute.

3. For a processor that has only simple instructions, a large number of instructions may by need to perform a given programming task. This could lead to a large value of N and a small value for S.

4. It is much easier to implement efficient pipelining in processors with simple instruction sets.

CISC

1. CISC: Complex instruction set computers.

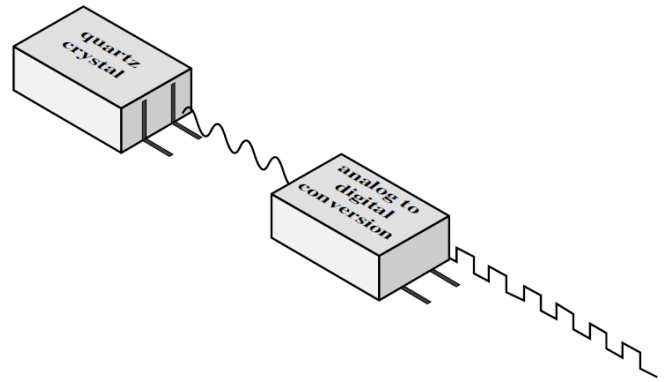
2. Complex instructions involve a large number of steps.

3. If individual instructions perform more complex operations, fewer instructions will be needed, leading to a lower value of N and a larger value of S.

4. Complex instructions combined with pipelining would achieve good performance

Performance Assessment

- Performance is one of the key parameters to consider, along with cost, size, security, reliability, and, in some cases power consumption.
- Application performance depends not just on the raw speed of the processor, but on the **instruction set**, choice of **implementation language**, **efficiency of the compiler**, and **skill of the programming.**
- Clock Speed
  - The System Clock: 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).
  - Clock signals are generated by a quartz crystal
  - The rate of pulses is known as the **clock rate**, or **clock speed**
  - One increment, or pulse, of the clock is referred to as a **clock cycle**, or a **clock tick**.
  - The time between pulses is the **cycle time**.
  - For example, a 1-GHz processor receives 1 billion pulses per second.



Performance Assessment

- Millions of instructions per second (MIPS) or MIPS rate
  - The System Clock: 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).

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

CPI: Cycle per instruction

f: Clock frequency (number of cycle per second)

Example: Consider a program having 500 million instructions, running on a 400 MHz processor. The program consists of three major types of instructions - ALU related, load/store, and branching. These instructions require 1, 2, and 4 CPI with a instruction mix of 60, 30, and 10% respectively in the program. Estimate the MIPS of the processor.

Performance Assessment

- Millions of instructions per second (MIPS) or MIPS rate
  - The System Clock: 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).

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

CPI: Cycle per instruction

f: Clock frequency (number of cycle per second)

Example: Consider a program having 500 million instructions, running on a 400 MHz processor. The program consists of three major types of instructions - ALU related, load/store, and branching. These instructions require 1, 2, and 4 CPI with a instruction mix of 60, 30, and 10% respectively in the program. Estimate the MIPS of the processor.

Solution: CPI=0.6x1+0.3x2+0.1x4=1.6

MIPS=$(400x10^6)/(1.6x10^6)$=250 MIPS

  - Similarly there are other performance parameters.

Performance Assessment :- (Sample problem 1) A program has 470 billion instructions. The processor has a clock period of 90ns. The CPI is 2, 1, 4, 3 for ALU, load, store, branching with a mix of 50%, 20%, 15%, 15% respectively. Find MIPS rate and also find the time reqd to execute the program.

Performance Assessment :- (Sample problem 1)

A program has 270 billion instructions. The processor has a clock period of 90ns. The CPI is 2, 1, 4, 3 for ALU, load, store, branching with a mix of 50%, 20%, 15%, 15% respectively. Find MIPS rate and also find the time reqd to execute the program.

Sol:-

$$CPI = 2 \times 0.5 + 1 \times 0.2 + 4 \times 0.15 + 3 \times 0.15$$

$$= 2.25$$

$$\text{MIPS rate} = \frac{f}{CPI \times 10^6} = \frac{111.11 \times 10^6}{2.25 \times 10^6}$$

$$= 49.382$$

$$t = \frac{270 \times 10^9 \times 10^3}{49.382222 \times 10^6}$$

$$\sim 54675 \text{ sec}$$

Performance Assessment of processor : (Sample problem 2)

A program has 195 trillion instructions. The cycles required for each instruction is different. Assume that there are 60% ALU instr, 10% load/store instr, 10% branch instruct, 10% string instructs, 10% I/O instr. The cycles required to execute these instructions are 3, 2, 1, 4, 5, respectively. Each cycle is of 20 μsec each.

Compute the MIPS rate & total exec time.

$$\text{MIPS rate} = \frac{f}{}$$

Performance Assessment of processor : (Sample problem 2)

A program has 195 trillion instructions. The cycles required for each instruction is different. Assume that there are 60% ALU instr, 10% load/store instr, 10% branch instruct, 10% string instructs, 10% I/O instr. The cycles required to execute these instructions are 3, 2, 1, 4, 5, respectively. Each cycle is of 20μsec ~~each~~ each. Compute the MIPS rate & total exec time.

Sol:- No. of inst = 195×10^{12}

$t = 20\ \mu sec$

$f = \frac{1}{20} \times 10^6$

$= 0.5 \times 10^5$

$= \boxed{50\ KHz}$

$CPI = 3 \times 0.6 + 2 \times 0.1 + 1 \times 0.1 + 4 \times 0.1 + 5 \times 0.1$

$= 1.8 + 0.1(12)$

$= \boxed{3}$

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

$t = \frac{195 \times 10^{12}}{16666}$

$\boxed{11.7 \times 10^9\ sec}$

$= \frac{50 \times 10^3}{3 \times 10^6}\ 100$

$= 0.016666667 \times 10^6$

$\boxed{16666.67}$ instr. per sec

Q:- Consider the MIPS rate to be '110'. Let the CPI of the instructions LC, ALU, Load/store, branching be 2, 3, 4 for 60%, 20% & 20% respectively. Calculate the maximum frequency with which the processor can run?.

Ans:-

$$MIPS = \frac{f}{CPI \times 10^6}$$

$$CPI = 2 \times 0.6 + 3 \times 0.2 + 4 \times 0.2$$

$$= 2.6$$

$$f = MIPS \times CPI \times 10^6$$

$$= 110 \times 2.6 \times 10^6$$

$$\boxed{f = 286\ MHz}$$

1. A 400 MHz processor was used to execute a benchmark program with the following instruction mix and clock cycle counts:

Instruction Type	Instruction Count	Clock Cycle Count
Integer arithmetic	450000	1
Data transfer	320000	2
Floating point	150000	2
Control transfer	80000	2

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

Ans:

1. A 400 MHz processor was used to execute a benchmark program with the following instruction mix and clock cycle counts:

Instruction Type	Instruction Count	Clock Cycle Count
Integer arithmetic	450000	1
Data transfer	320000	2
Floating point	150000	2
Control transfer	80000	2

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

Ans:

Effective CPI = (Total clock cycles needed to execute all instructions in the program) / (Instruction count of the entire program)

Here, Total clock cycles needed = (450000 x 1) + (320000 x 2) + (150000 x 2) + (80000 x 2)

= 1550000 cycles

Total Instruction count = 450000 + 320000 + 150000 + 80000 = 1000000

Hence, Effective CPI = 1550000/ 1000000 = 1.55

Execution time, T = Total instruction count x CPI x clock cycle duration

= Total instruction count x CPI x (1/clock frequency) ---- {since 1/clock frequency = clock duration)

= 1000000 x 1.55 x (1/400 x 10^6)

= 0.0038 s

MIPS = (clock frequency)/ (CPI x 10 ^ 6)

= (400 x 10^6)/ (1.55 x 10^6) = 258.06

2. Consider the execution of an object code with 2 x 10^6 instructions on a 400 MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles (CPI) needed for each instruction type are given below based on the result of a program trace experiment:

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%

a. Calculate the average CPI when the program is executed on a processor with the above trace results.
b. Calculate the corresponding MIPS rate based on CPI obtained in part (a).
c. Calculate the execution time of the program.

Ans:

No of arithmetic and logic instructions = 60% of 2 x 10^6 = 120 x 10^4

No of load/store with cache hit = 18% of 2 x 10^6 = 36 x 10^4

No of branch instructions = 12% of 2 x 10^6 = 24 x 10^4

No of memory reference with cache miss = 20 x 10^4

CPI = ((120 x 10^4 x 1) + (36 x 10^4 x 2) + (24 x 10^4 x 4) + (20 x 10^4 x 8))/ (2 x 10^6) = 2.24

MIPS = (clock frequency)/ (CPI x 10^6)

= (400 x 10^6)/ (2.24 x 10^6) = 178.57

T = 2 x 10^6 x 2.24 x (1/400 x 10^6) = 0.0112 s

2. Consider the execution of an object code with 2 x 10^6 instructions on a 400 MHz processor. The program consists of four major types of instructions. The instruction mix and the number of cycles (CPI) needed for each instruction type are given below based on the result of a program trace experiment:

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%

a. Calculate the average CPI when the program is executed on a processor with the above trace results.
b. Calculate the corresponding MIPS rate based on CPI obtained in part (a).
c. Calculate the execution time of the program.

A workstation uses a 1.5 GHz processor with a claimed 1000 MIPS rating to execute a given program mix. Assume a one cycle delay for each memory access.

a. What is the effective CPI of this computer?
b. Suppose the processor is being upgraded with a 3.0 GHz clock. However, even with faster cache two clock cycles are needed per memory access. If 30% of the instructions require one memory access and another 5% require two memory accesses per instruction, what is the performance of the upgraded processor with a compatible instruction set and equal instruction counts in the given program mix?

Ans:

a. MIPS = f/(CPI x 10^6)

CPI = f/MIPS x 10^6)

= (1.5 x 10^9)/ (1000 x 10^6)

= 1.5

b. 30% instructions take one memory access (hence, two clock cycles as two clock cycles are needed per memory access as per question)
Similarly 5% instructions take 2 memory accesses (i.e., 4 cycles)
Remaining 65% takes 1 cycle
Total no of clock cycles = (30 x 2) + (5 x 4) + (65 x 1) = 145 cycles

CPI = Total no of clock cycles/ Total no of instructions

= 145/ 100 = 1.45

MIPS = f/ (CPI x 10^6)

= (3 x 10^9)/ (1.45 x 10^6) = 2068.9

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

Instruction Type	Instruction Count	Cycles per Instruction
Integer arithmetic	45,000	1
Data transfer	32,000	2
Floating point	15,000	2
Control transfer	8000	2

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

Given,

Frequency =40 MHz

Clock time = $\frac{1}{40*10^6} sec$

Instruction type	Instruction count	cycle/instructions	% of instruction type
Integer arithmetic	45000	1	45%
Data transfer	32000	2	32%
Floating point	15000	2	15%
Control Transfer	8000	2	8%
Total	100000		100%

$$\mathbf{CPI} = \sum_i (\mathbf{Fraction\ of\ instruction\ type}\ i \times \mathbf{Cycles\ needed\ by\ instruction\ type}\ i)$$

$$\implies \mathbf{CPI} = 0.45 * 1 + (0.32 + 0.08 + 0.15) * 2 = 1.55$$

$$\mathbf{Execution\ time} = \mathbf{CPI} * \mathbf{total\ number\ of\ instructions} * \mathbf{Clock\ time}$$

$$= 1.55 * 10^5 * \frac{1}{40*10^6}\ \text{sec}$$

$$= 3.875\ ms$$

$$\mathbf{MIPS} = \frac{\mathbf{Clock\ Frequency}}{\mathbf{CPI}*10^6}$$

$$= \frac{40*10^6}{1.55*10^6}$$

$$= 25.8$$

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

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

a. Determine the effective *CPI*, MIPS rate, and execution time for each machine.
b. Comment on the results.