Brief history of And Generation of Computer computer

1- *Knowledge characteristic Generation of Computer*

2- Knowledge of type **Generation of Computer**

3-evolution of electronic circuits

4- Difference between type of **Generation of Computer**

5- Knowledge Classification of compute

1-1 FIRST GENERATION

- The computers of the first generation (1951-1958) were physically very large machines characterized by the vacuum tube (fig. 1-1). Because they used vacuum tubes, they were very unreliable, required a lot of power to run, and produced so much heat. Compared to today's computers, they had slow input and output devices, were slow in processing, and had small storage capacities (magnetic cores) . Many of the internal processing functions were measured in thousandths of a second (millisecond).Using such a machine language was efficient for the computer but difficult for the programmer (MARK-I). The first generation computers were welcomed by Government and Universities.

THE VON NEUMANN MACHINE

- The task of entering and altering programs for the ENIAC was extremely tedious. The programming process could be facilitated if the **program** could be represented in a form suitable for storing in memory alongside the data. Then, a computer could get its instructions by reading them from memory,
- and a program could be set or altered by setting the values of a portion of memory. This idea, known as the ***stored-program concept***. computer, the EDVAC (Electronic Discrete Variable Computer). In 1946, von Neumann and his colleagues began the design of a new stored program computer, referred to as the **IAS** computer, at the Princeton Institute for Advanced Studies. The IAS computer, although not completed until 1952, is the prototype of all subsequent general-purpose computers. The IAS computer, although not completed until 1952, is the prototype of all subsequent general-purpose computers.

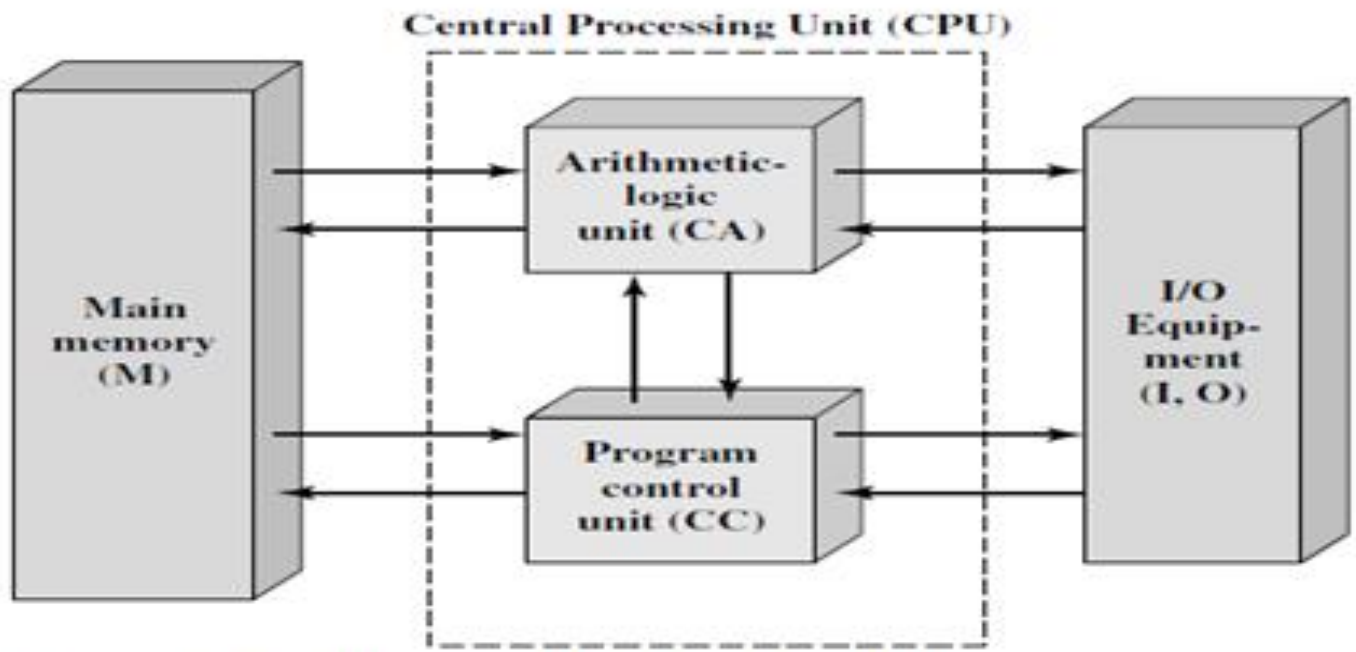

Figure 2.1 Structure of the IAS Computer

- It consists of
- • A main memory, which stores both data and instructions[1]
- • 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
-
- The memory of the IAS consists of **1000** storage locations, called ***words,*** of 40 binary digits (bits) each.2 Both data and instructions are stored there. Numbers are represented in binary form, and each instruction is a binary code.

- Figure 2.2 illustrates these formats. 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).
- The control unit operates the IAS by fetching instructions from memory and executing them one at a time.

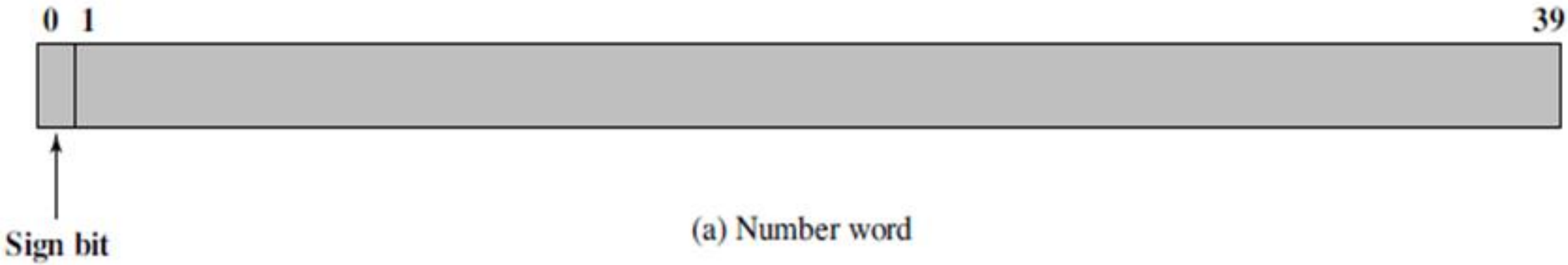

Figure 2.2 IAS Memory Formats

- The IAS operates by repetitively performing an ***instruction cycle;*** .Each instruction cycle consists of two sub cycles. During the ***fetch cycle,*** the opcode of the next instruction is loaded into the IR and the address portion is loaded into the MAR. This instruction may be taken from the IBR, or it can be obtained from memory by loading a word into the MBR, and then down to the IBR, IR, and MAR.
  - Once the opcode is in the IR, the ***execute cycle*** is performed. Control circuitry interprets the opcode and executes the instruction by sending out the appropriate control signals to cause data to be moved or an operation to be performed by the ALU. The IAS computer had a total of 21 instructions, which are listed in Table 2.1.

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator A
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location
	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 nex struction from left half of M(X)
	00010000	JUMP+ M(X,20:39)	If number in the accumulator is nonnegative, take nex 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 res 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 positic
	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