Computer Organization and Architecture

CH 01 & 02

Organization and Architecture

- A distinction between *computer architecture* and *computer organization.*
- **Computer architecture** refers to the attributes of a system visible to a programmer - have a direct impact on the logical execution of a program.
  - instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory.
- **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications.
  - hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

Content

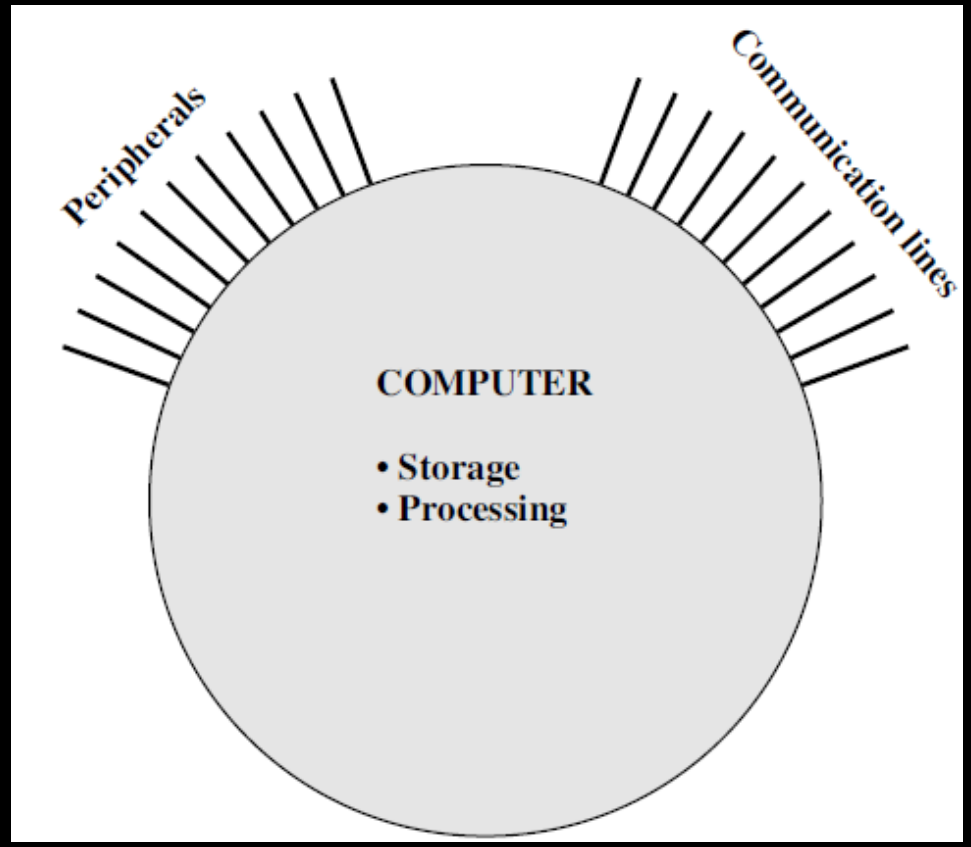
CH 00 & CH 01
Structure and Function
A Brief History of Computers

Structure and Function

- Basic functions that a computer can perform:
  - Data processing
  - Data storage
  - Data movement
  - Control

Structure and Function

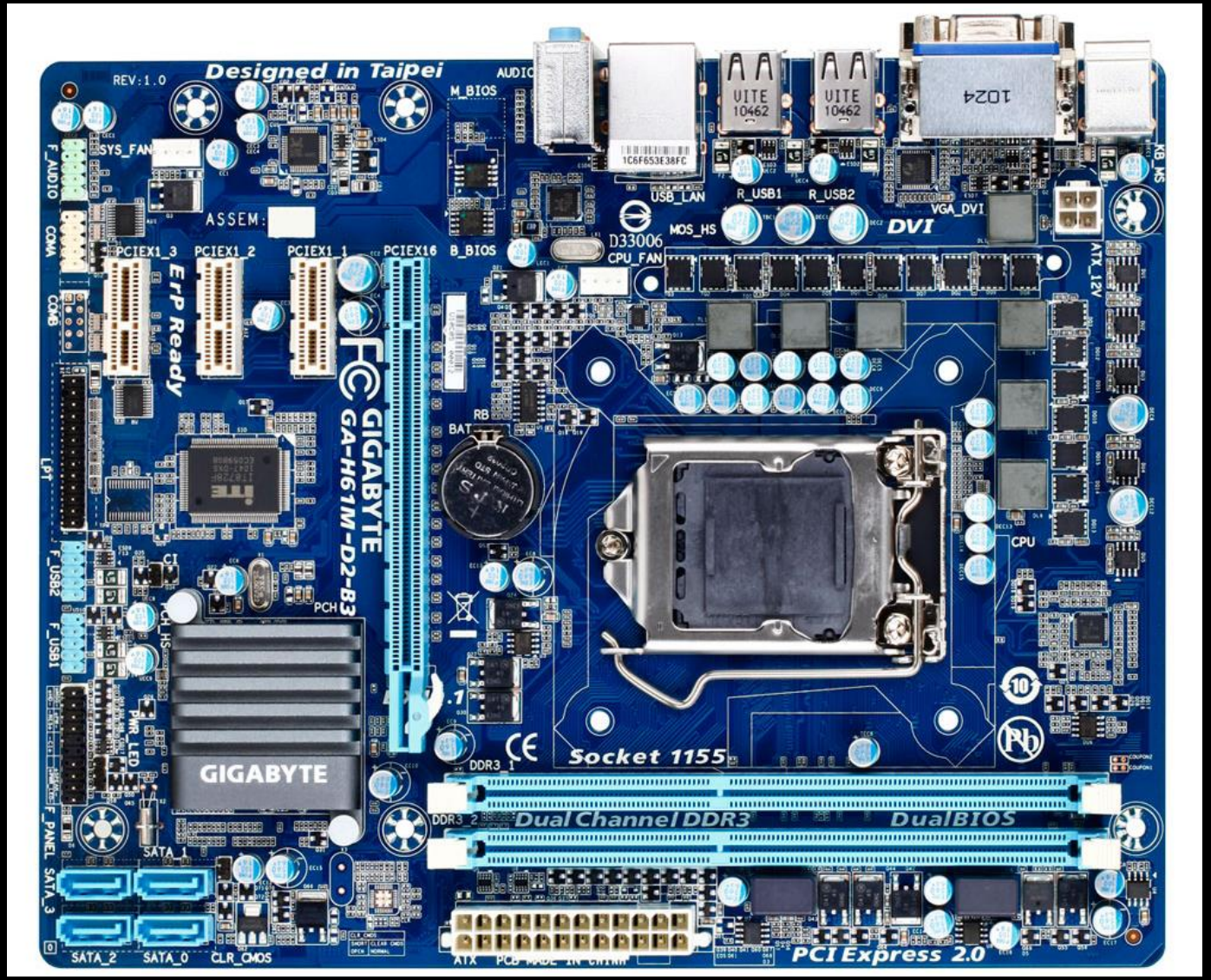
- The simplest possible depiction of a computer. The computer interacts in some fashion with its external environment.



Structure and Function

- The simplest possible depiction of a computer. The computer interacts in some fashion with its external environment.

Structure and Function



Structure and Function

- Our focus is on:
  - **Central processing unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor.
  - **Main memory:** Stores data.
  - **I/O:** Moves data between the computer and its external environment.
  - **System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O.
- A common example of system interconnection is by means of a system bus, consisting of several conducting wires to which all the other components attach.

Structure and Function

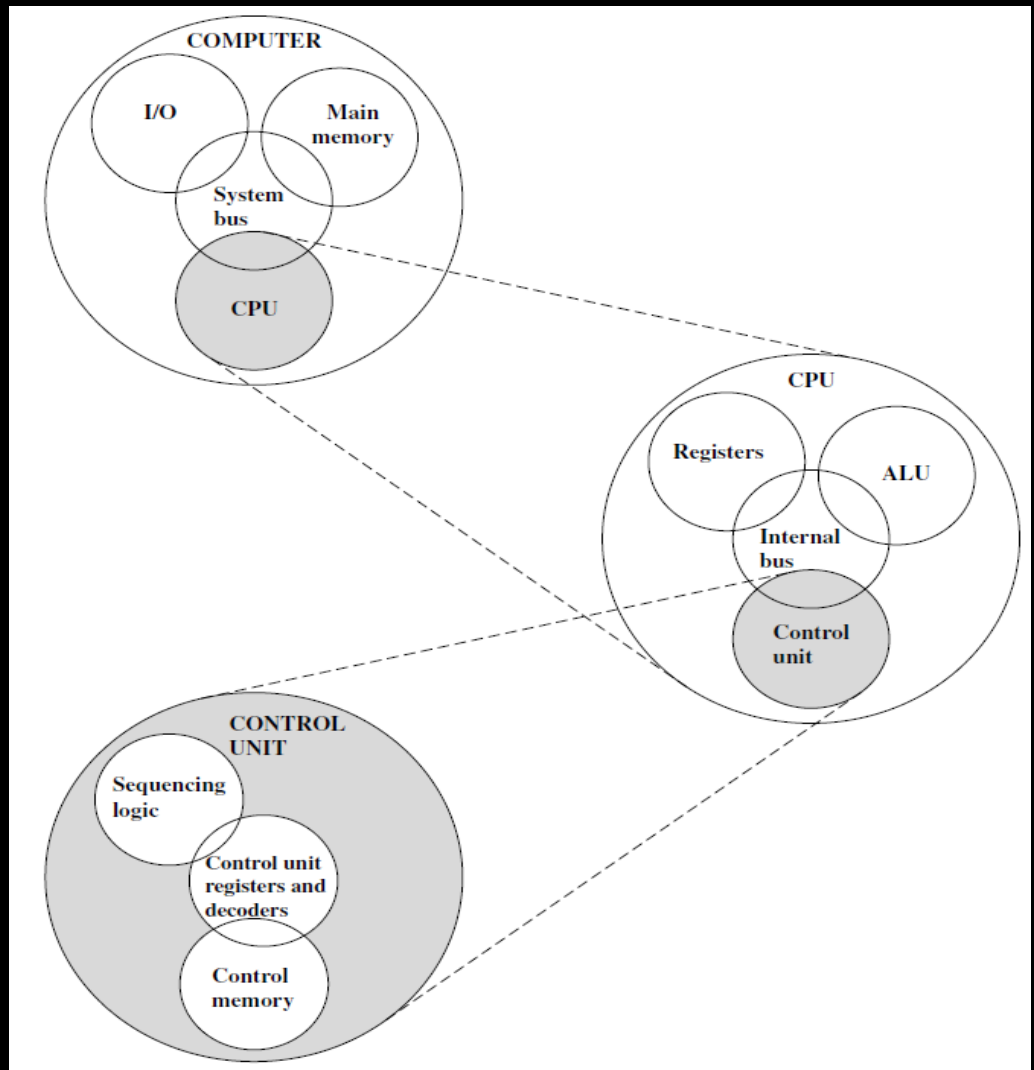
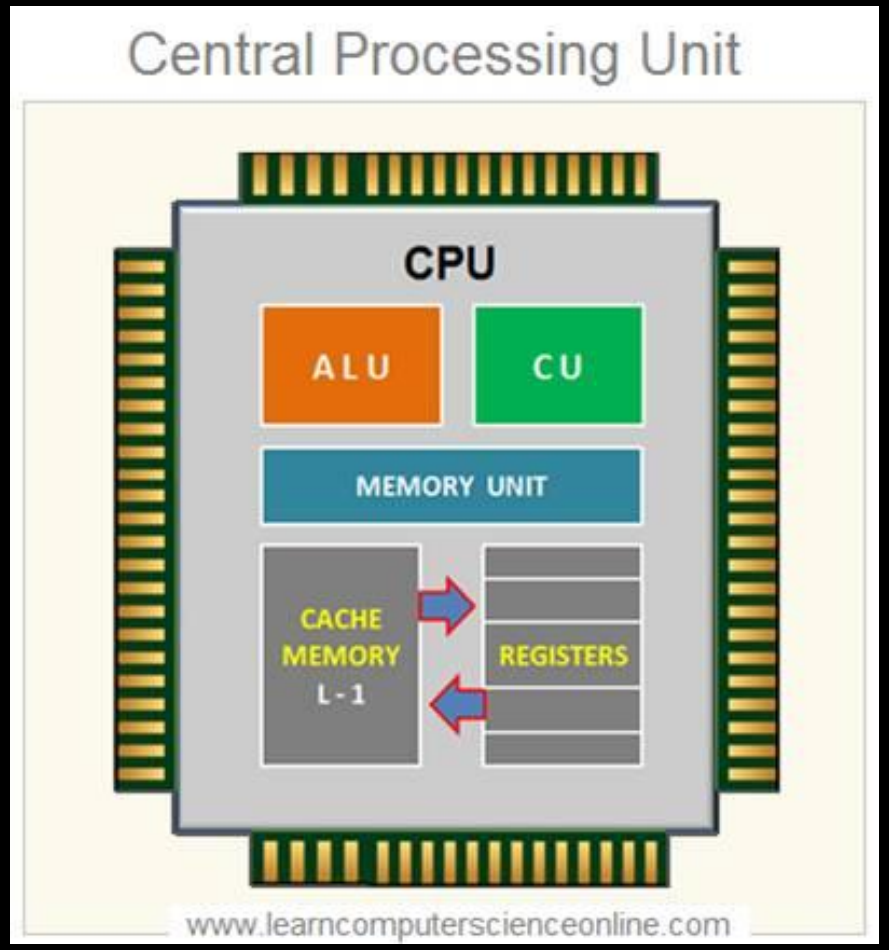


Figure 1.4 The Computer: Top-Level Structure

Structure and Function

Structure and Function

- Components of CPU:
  - **Control unit:** Controls the operation of the CPU and hence the computer
  - **Arithmetic and logic unit (ALU):** Performs the computer's data processing functions
  - **Registers:** Provides storage internal to the CPU
  - **CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers

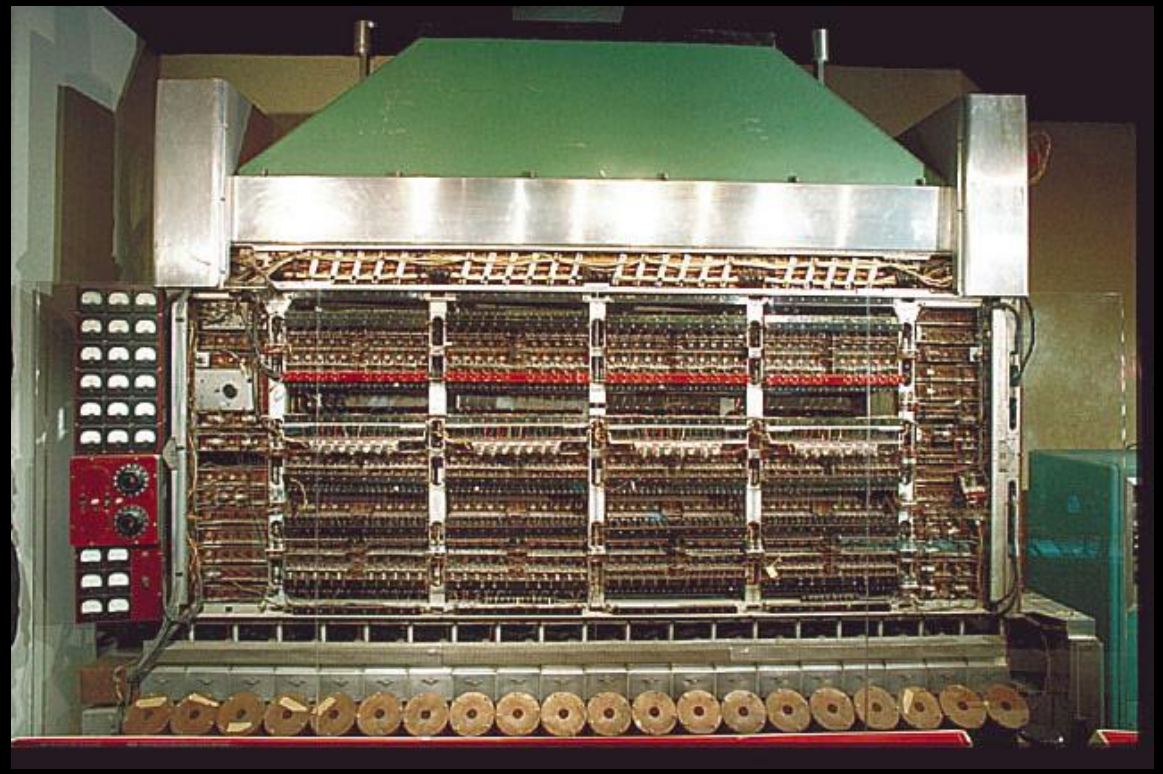


Content

CH 00 & CH 01
Structure and Function
A Brief History of Computers

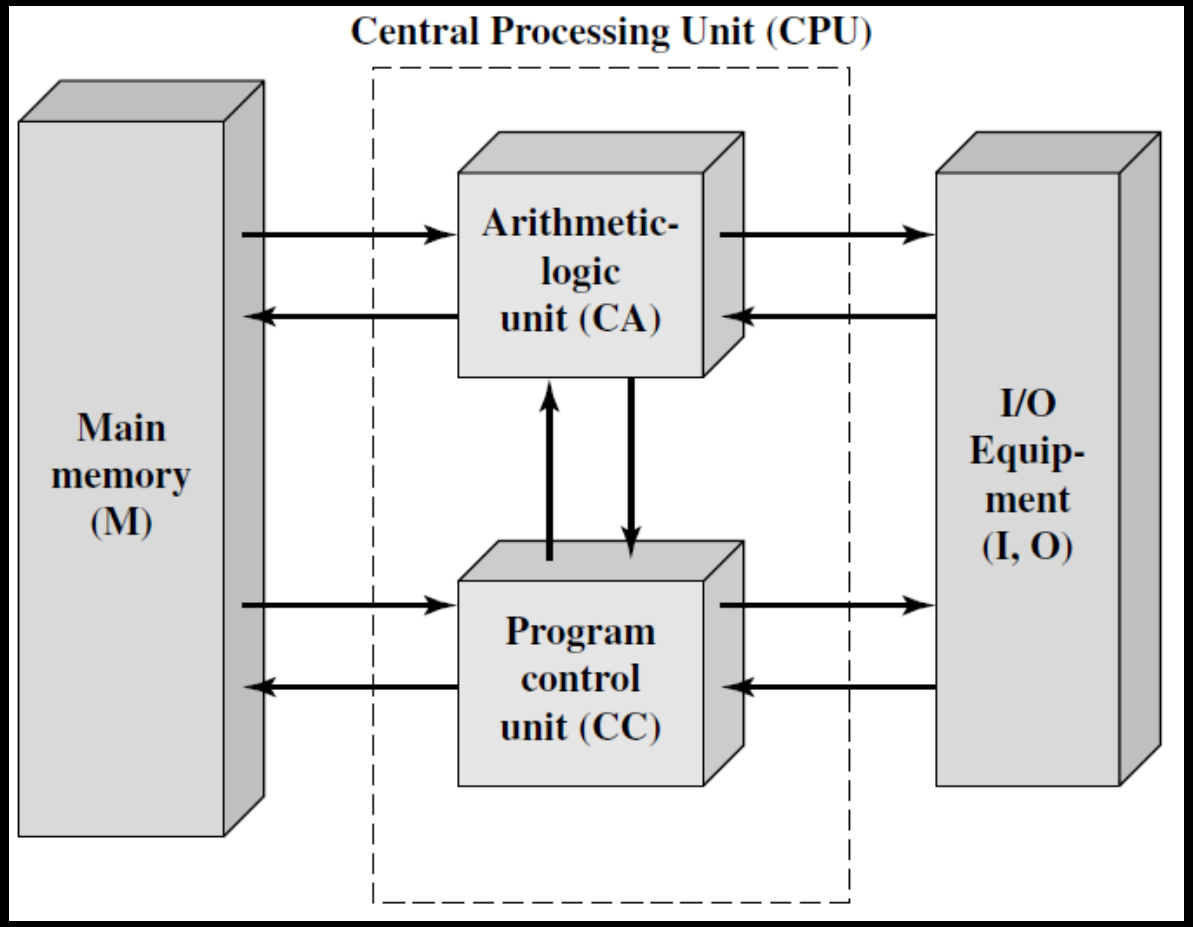
A Brief History of Computers

- Von Neumann designed of a new stored program computer, referred to as the IAS computer.
  - prototype of all subsequent general-purpose computers.



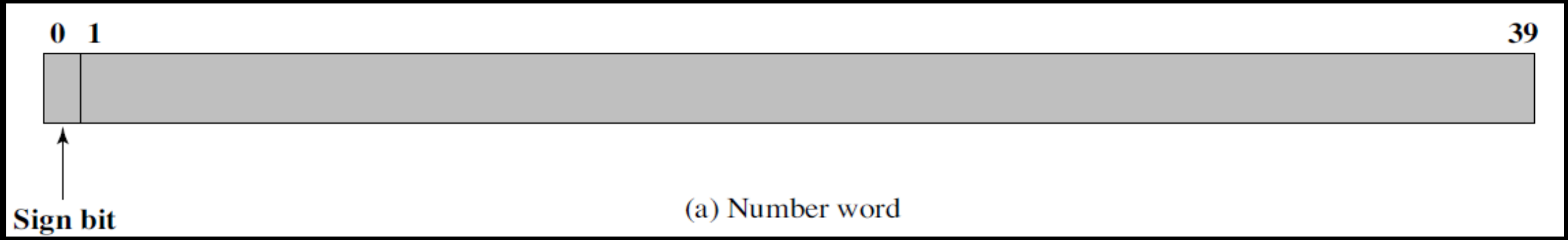
A Brief History of Computers

- Structure of the IAS computer
  1. Main memory, stores both data and instructions
  2. Arithmetic and logic unit (ALU), operating on binary data
  3. Control unit, interprets the instructions in memory to be executed
  4. Input and output (I/O) equipment operated by the control unit



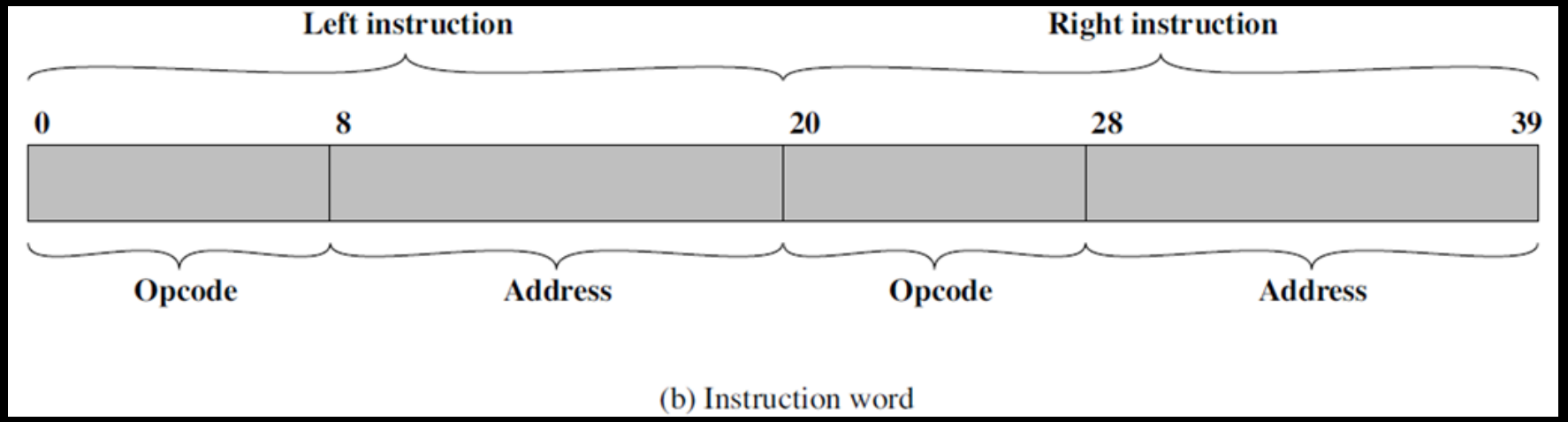
A Brief History of Computers

- The memory in IAS consists of 1000 *words* – a word is a storage location.
  - Each word stores 40 bits. It can store data and instructions.
- If a word stores a number, the first bit is the sign bit



A Brief History of Computers

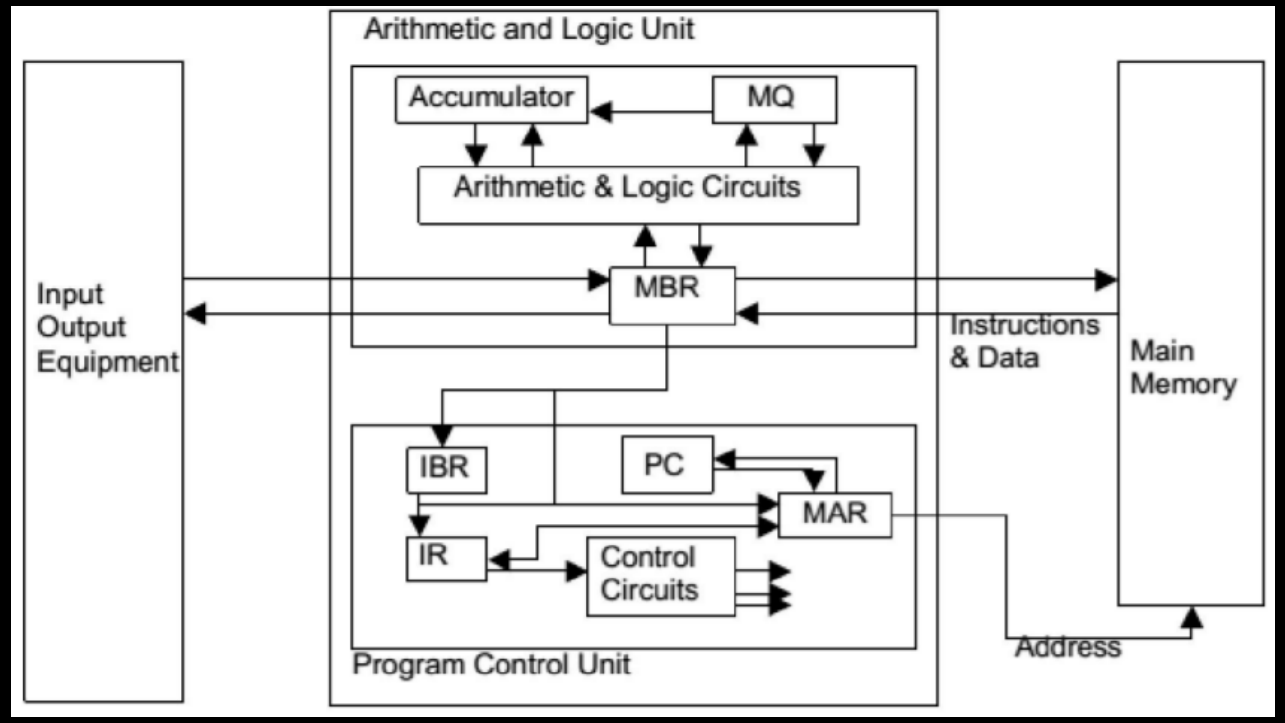
- The memory in IAS consists of 1000 *words* – a word is a storage location.
  - Each word stores 40 bits. It can store data and instructions.
- If a word stores an instruction, a word is divided into two 20-bit instructions.
  - 8-bit operation code (opcode) and 12-bit address designating one of the words in memory (numbered from 0 to 999).



(b) Instruction word

A Brief History of Computers

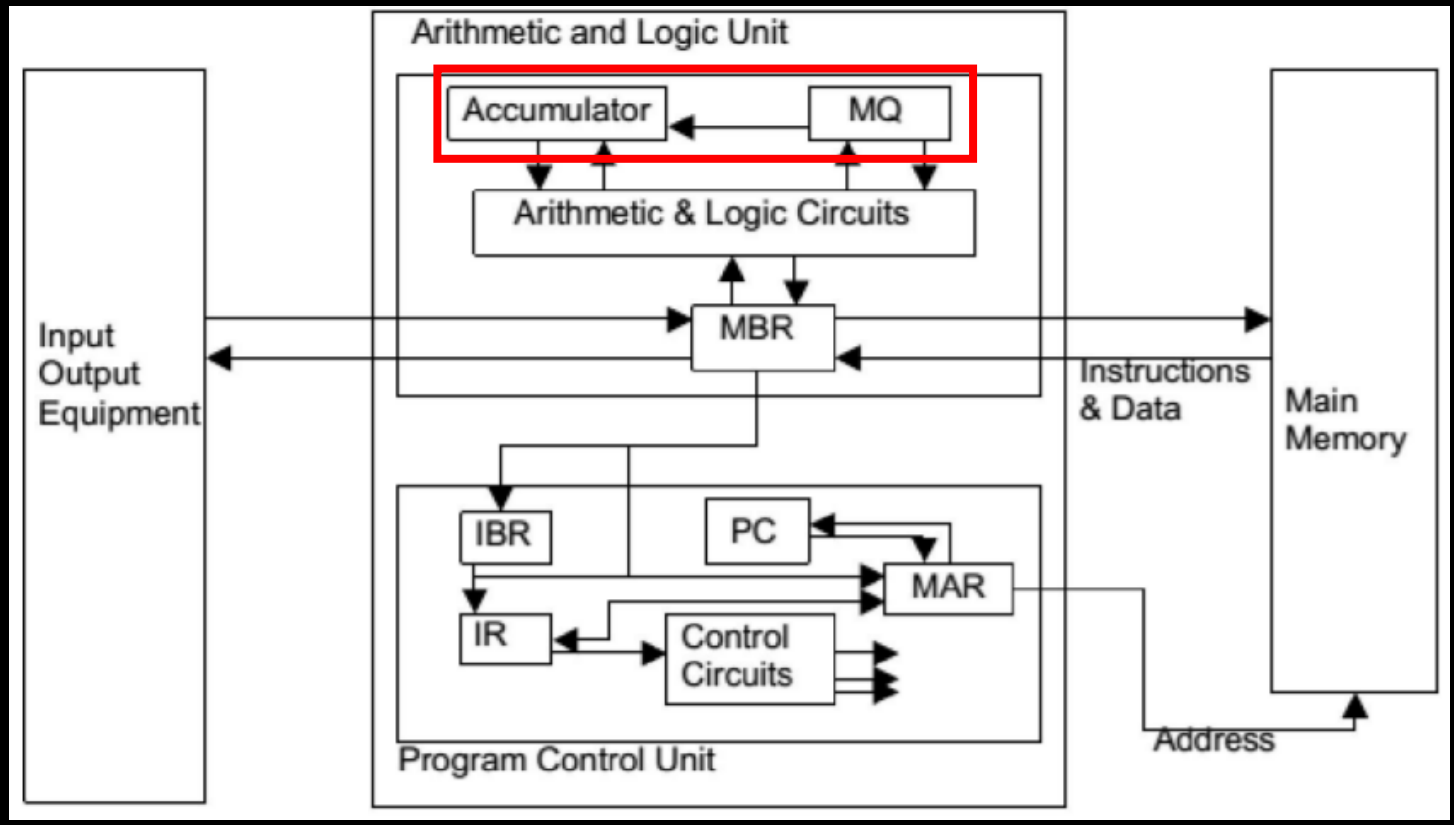
- The control unit fetches instructions from memory and executes them one at a time.
- Both the control unit and the ALU contain storage locations, called *registers.*



A Brief History of Computers

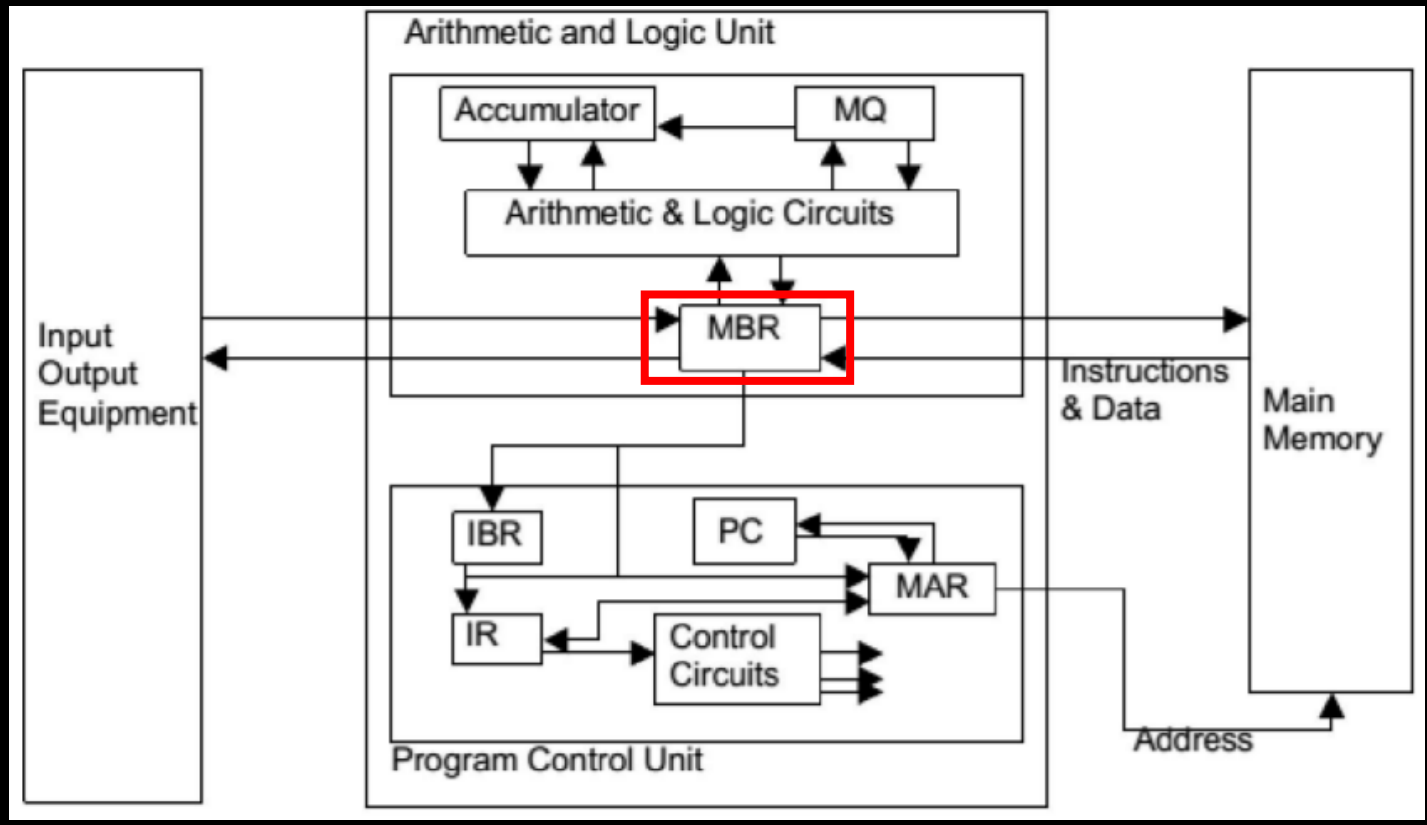
- ALU registers:
  - **Accumulator (AC) and multiplier quotient (MQ):** Hold temporarily operands and results of ALU operations.

  For example, the result of multiplying two 40-bit numbers is an 80-bit number; the most significant 40 bits are stored in the AC and the least significant in the MQ.



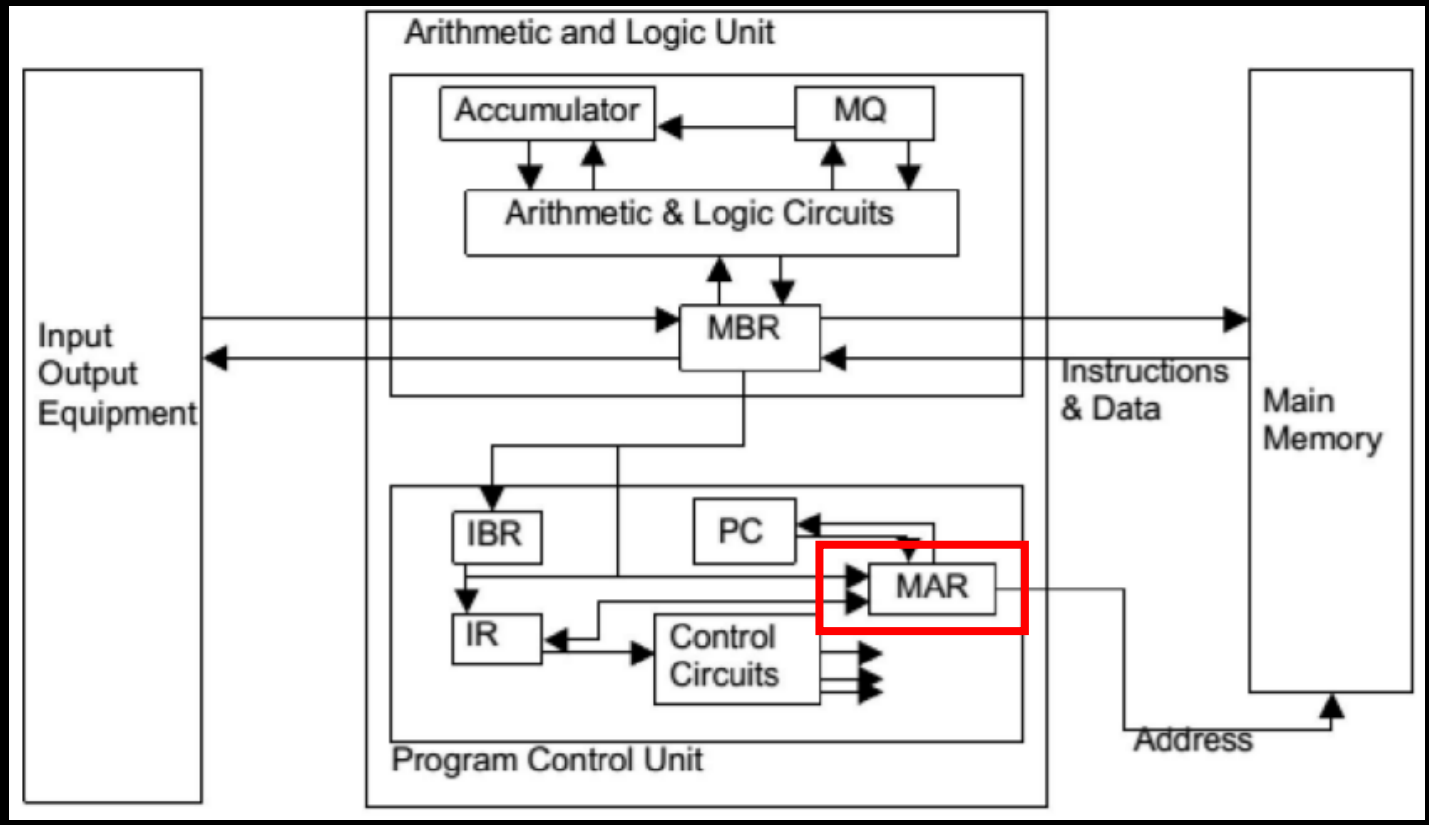
A Brief History of Computers

- ALU registers:
  - **Memory buffer register (MBR):** Contains a word to be stored in memory or sent to the I/O unit, or is used to receive a word from memory or from the I/O unit.



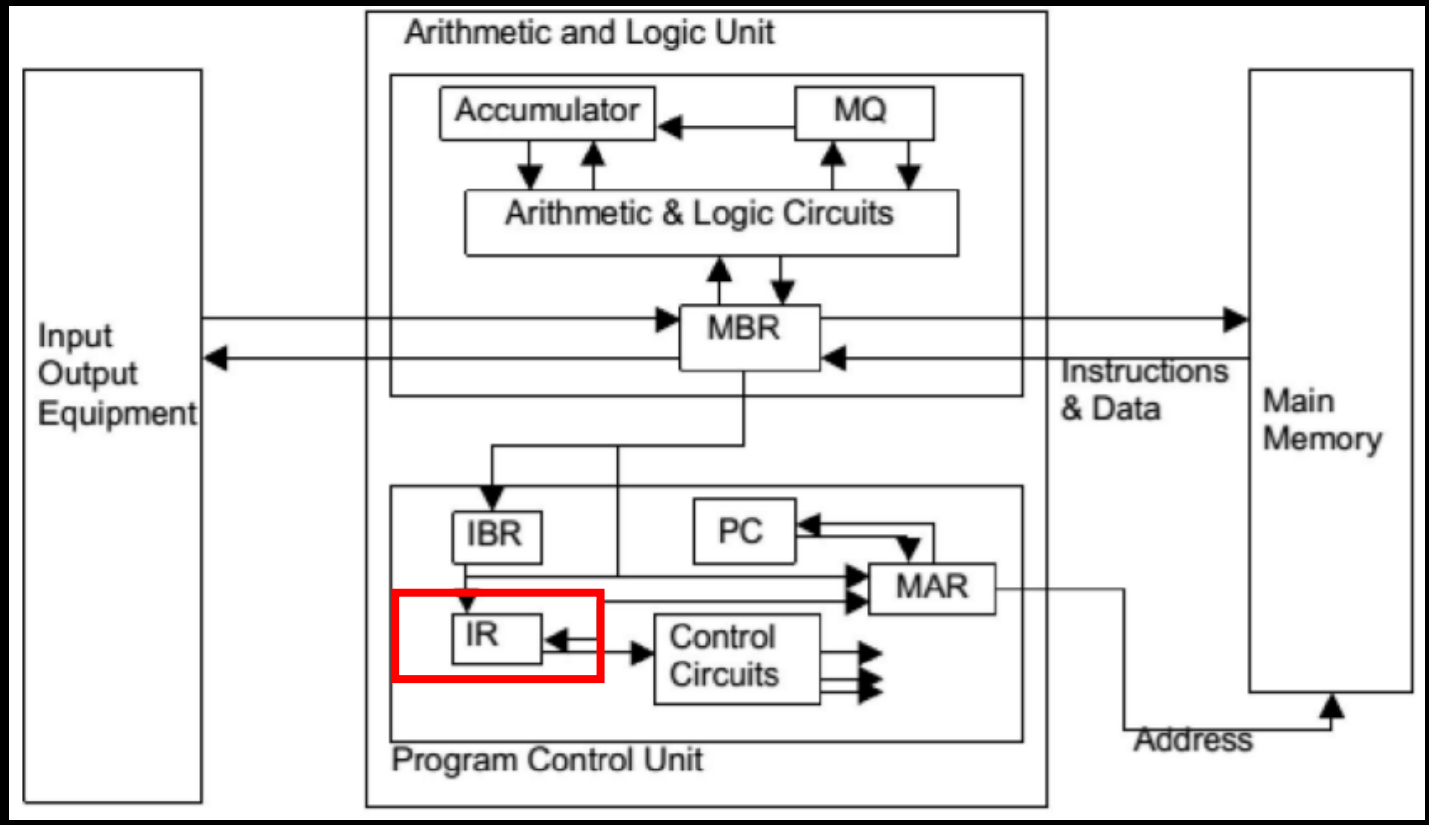
A Brief History of Computers

- Control unit registers:
  - **Memory address register (MAR):** Specifies the address in memory of the word to be written from or read into the MBR.



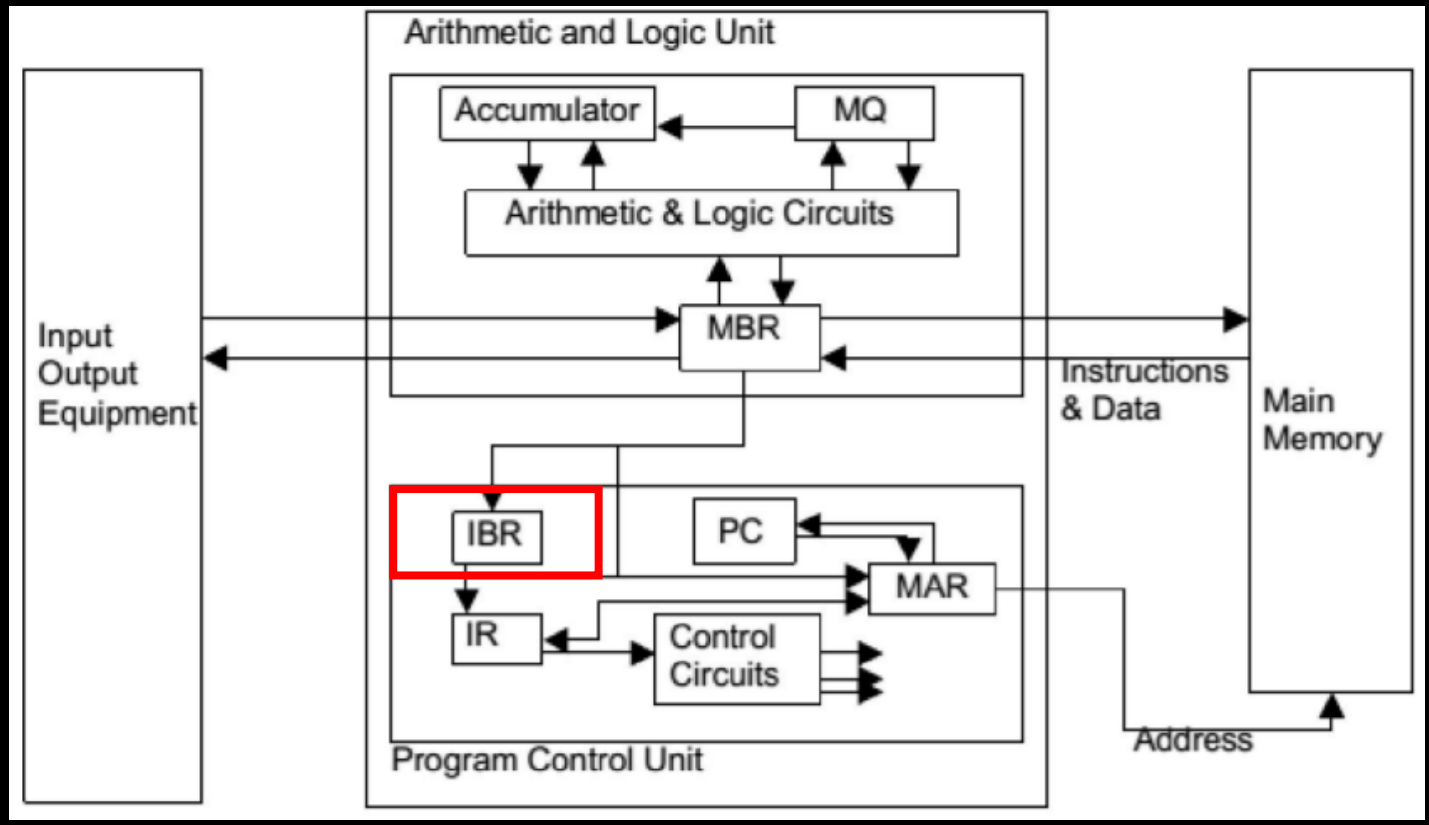
A Brief History of Computers

- Control unit registers:
  - **Instruction register (IR):** Contains the 8-bit opcode instruction being executed.



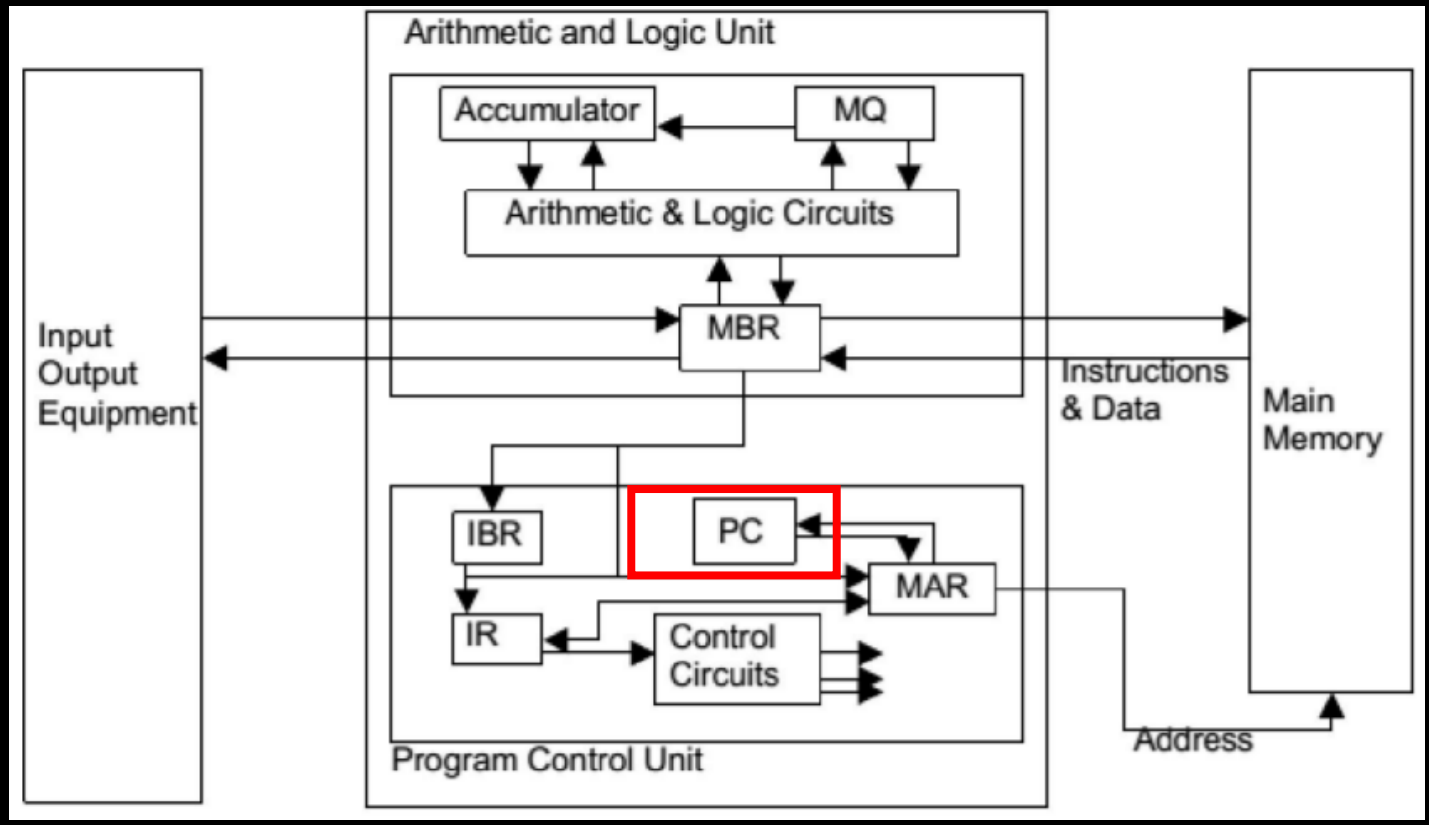
A Brief History of Computers

- Control unit registers:
  - **Instruction buffer register (IBR):** Employed to hold temporarily the righthand instruction from a word in memory.



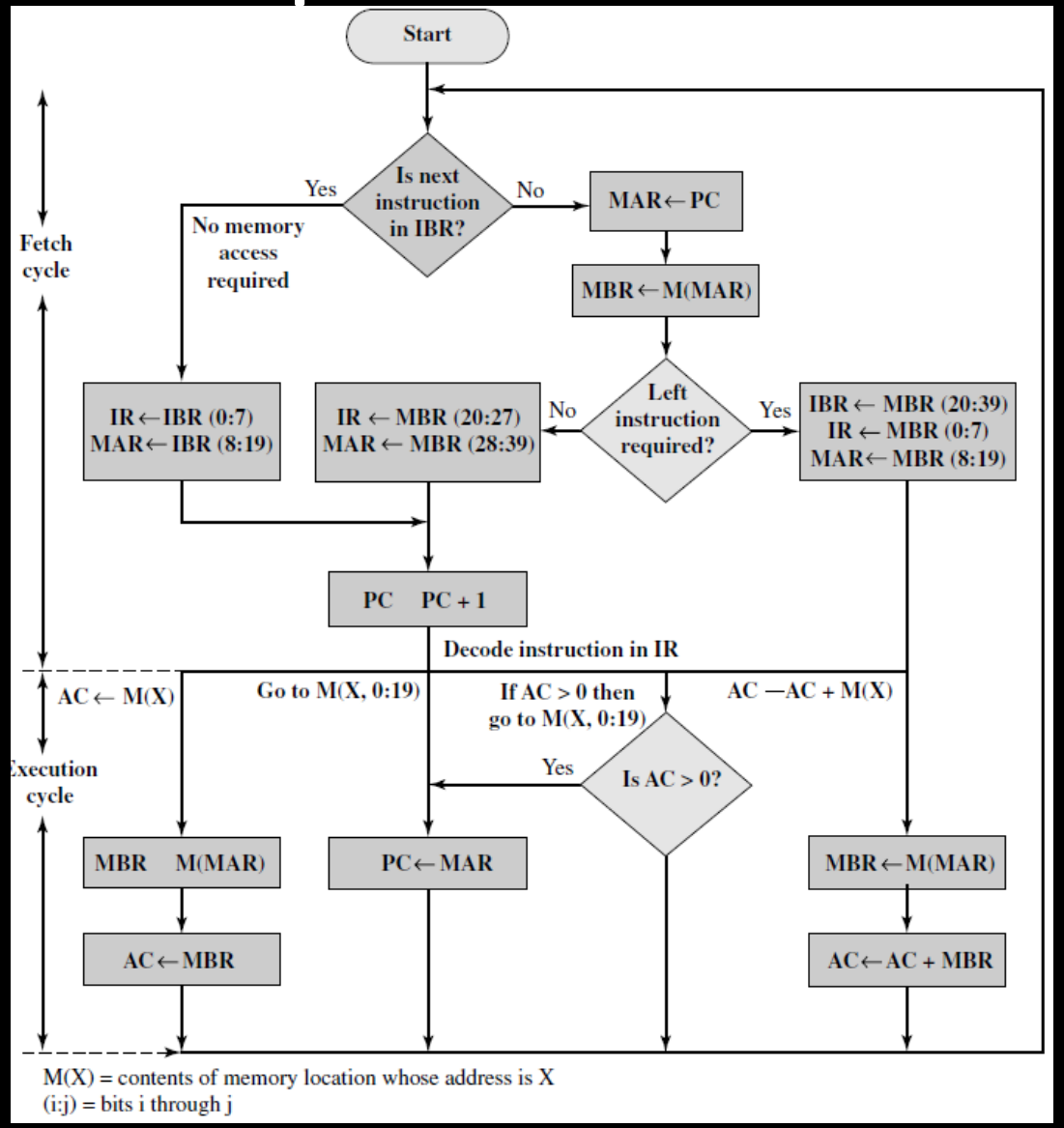
A Brief History of Computers

- Control unit registers:
  - **Program counter (PC):** Contains the address of the next instruction-pair to be fetched from memory.



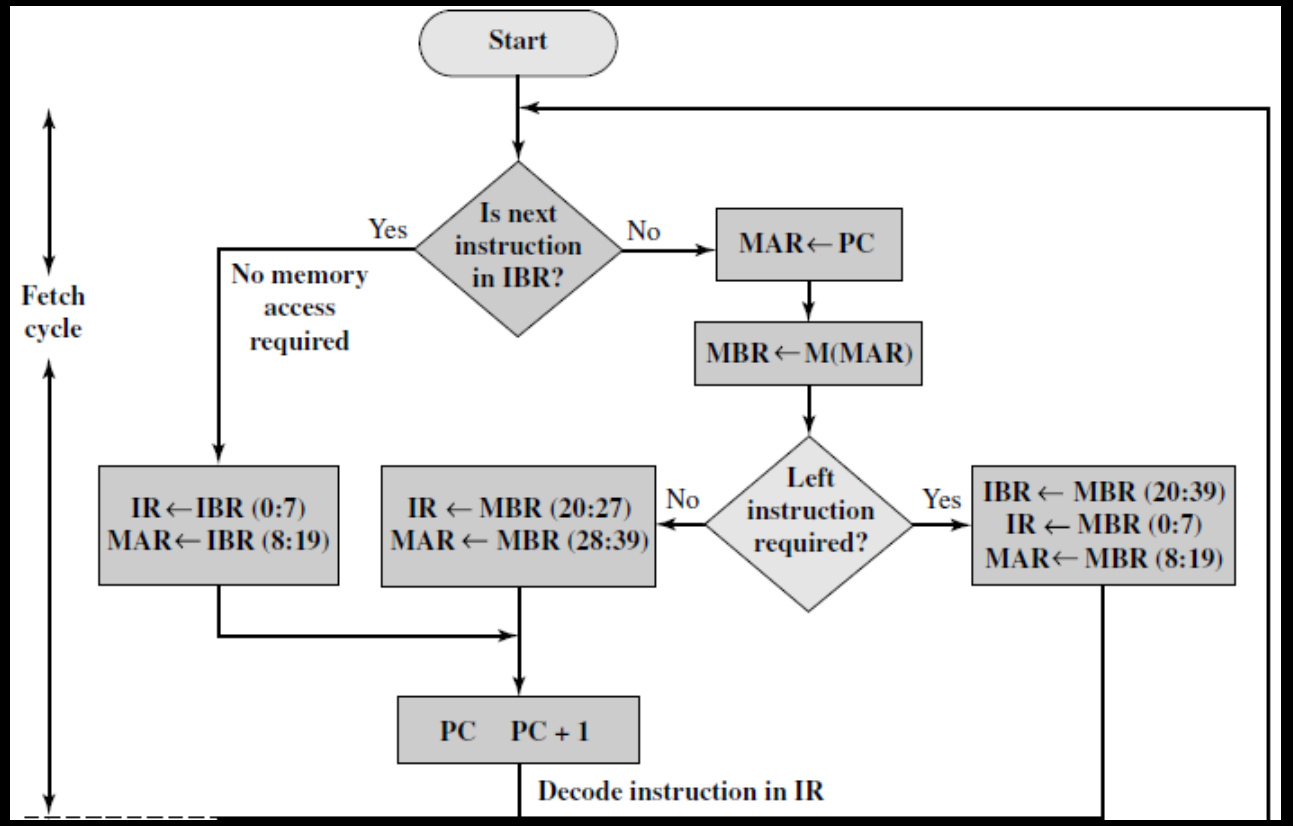
A Brief History of Computers

- The IAS operates by repetitively performing an *instruction cycle.*
- Each instruction cycle consists of:
  - fetch cycle
  - execute cycle



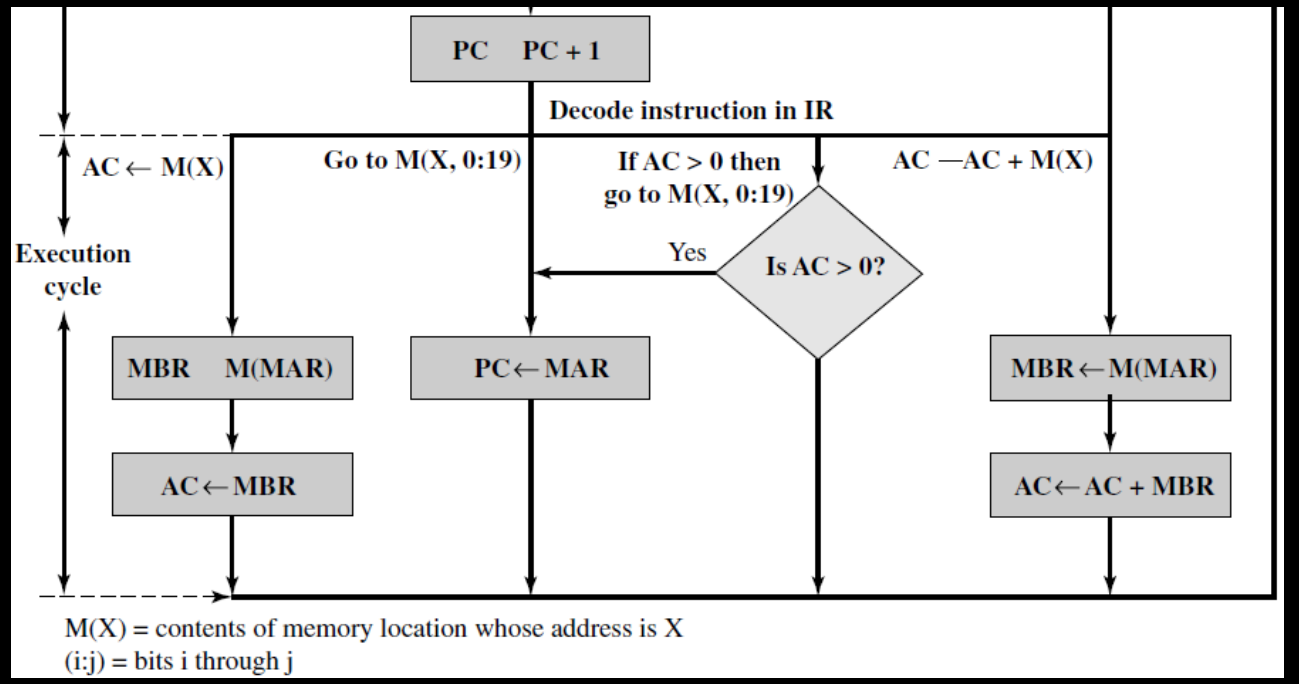
A Brief History of Computers

- During the fetch cycle:
  1. the opcode of the next instruction is loaded into the IR.
  2. 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.



A Brief History of Computers

- During the execution cycle:
  - 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.



A Brief History of Computers

- The IAS computer had a total of 21 instructions

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)

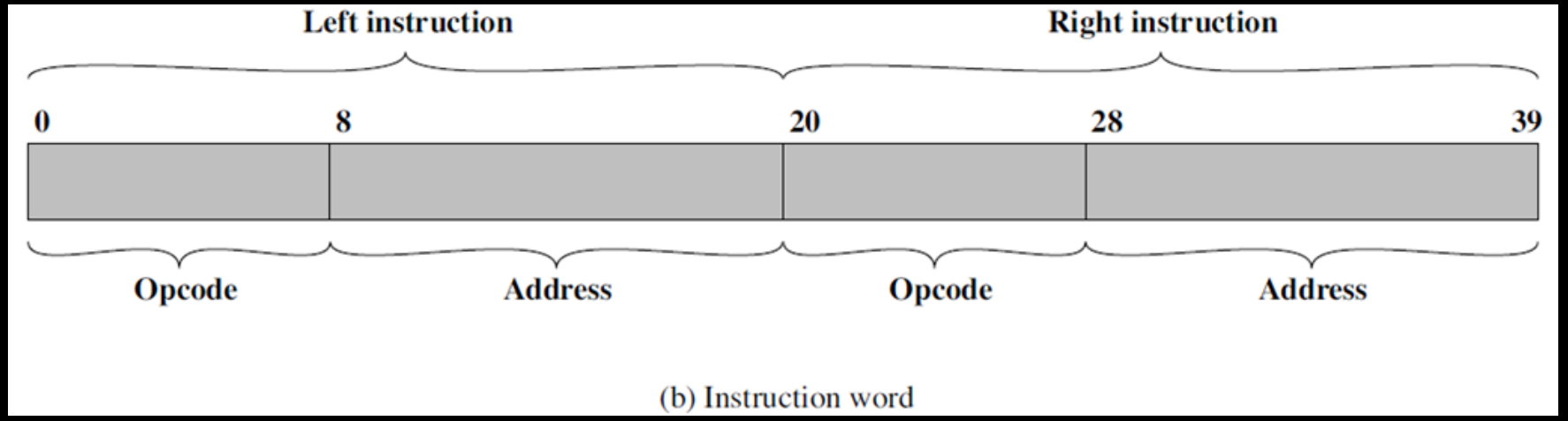
A Brief History of Computers

- The IAS computer had a total of 21 instructions

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

A Brief History of Computers

- The opcode portion (first 8 bits) specifies which of the 21 instructions is to be executed.
- The address portion (remaining 12 bits) specifies which of the 1000 memory locations is to be involved in the execution of the instruction.



(b) Instruction word

Exercises

BEFORE WE START…

There are 3 key steps to solve any problem

1. **Memorizes the equations/formulas**
2. **Identify the given input**
3. **Identify the required output/results**

Exercises

2.1. Let $A = A(1), A(2), \ldots, A(1000)$ and $B = B(1), B(2), \ldots, B(1000)$ be two vectors (one-dimensional arrays) comprising 1000 numbers each that are to be added to form an array C such that $C(I) = A(I) + B(I)$ for $I = 1, 2, \ldots, 1000$. Using the IAS instruction set, write a program for this problem. Ignore the fact that the IAS was designed to have only 1000 words of storage.

Exercises

We have 3 vectors A, B, and C. Suppose vector A starts at address 1001.

To sum each element from A and B and store the result in C, we need a loop.

The ISA instruction set doesn't define a loop instruction explicitly, instead, we use counter variables and $JUMP$ instruction to control execution flow.

1. N = 999, counter to go through each element in each vector (stop when -1)
2. X = 1, to specify increase/decrease index of the vector
3. Y = 1000, to move between the vectors

Exercises

Steps:

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1
   to get the previous element.
6. If N is nonnegative, update counters,
   go to step 2.
7. If N is negative, halt the program.

TIP: trace the content of the AC register and memory.

Exercises

AC:

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000

Exercises

AC: A(2000)

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)

Exercises

AC: A(2000)+B(3000)

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)

Exercises

AC: A(2000)+B(3000)

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)

Exercises

AC: 999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)

Exercises

AC: 998

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)

Exercises

AC: 998

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)

Exercises

AC: 998

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	999
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)

Exercises

AC: 998

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)

Exercises

AC: 999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)

Exercises

AC: 1999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(2000)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)

7R	ADD M(2)

Exercises

AC: 1999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)

Exercises

AC: 2999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(3000)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)
8R	ADD M(2)

Exercises

AC: 2999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(2999)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)
8R	ADD M(2)
9L	STOR M(3, 28:39)

Exercises

AC: 3999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(2999)
4L	STOR M(4000)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)
8R	ADD M(2)
9L	STOR M(3, 28:39)
9R	ADD M(2)

Exercises

AC: 3999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index 2000 from A.
3. Sum the element at index 3000 from B.
4. Store the result of $A + B$ in C at index 4000.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(2999)
4L	STOR M(3999)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)
8R	ADD M(2)
9L	STOR M(3, 28:39)
9R	ADD M(2)
10L	STOR M(4, 8:19)

Exercises

AC: 3999

1. Define counter variables: 999, 1, and 1000
2. Get the element at index ~~2000~~ from A.
3. Sum the element at index ~~3000~~ from B.
4. Store the result of $A + B$ in C at index ~~4000~~.
5. Get the counter N, decrement by 1 to get the previous element.
6. If N is nonnegative, update counters, go to step 2.
7. If N is negative, halt the program.

Add.	Instruction
0	998
1	1
2	1000
3L	LOAD M(1999)
3R	ADD M(2999)
4L	STOR M(3999)
4R	LOAD M(0)
5L	SUB M(1)
5R	JUMP+ M(6, 20:39)
6L	JUMP M(6, 0:19)
6R	STOR M(0)
7L	ADD M(1)
7R	ADD M(2)
8L	STOR M(3, 8:19)
8R	ADD M(2)
9L	STOR M(3, 28:39)
9R	ADD M(2)
10L	STOR M(4, 8:19)
10R	JUMP M(3, 8:19)

Exercises

Location	Instruction	Comments
0	999	Constant (count N)
1	1	Constant
2	1000	Constant
3L	LOAD M(2000)	Transfer A(I) to AC
3R	ADD M(3000)	Compute A(I) + B(I)
4L	STOR M(4000)	Transfer sum to C(I)
4R	LOAD M(0)	Load count N
5L	SUB M(1)	Decrement N by 1
5R	JUMP+ M(6, 20:39)	Test N and branch to 6R if nonnegative
6L	JUMP M(6, 0:19)	Halt
6R	STOR M(0)	Update N
7L	ADD M(1)	Increment AC by 1
7R	ADD M(2)	
8L	STOR M(3, 8:19)	Modify address in 3L
8R	ADD M(2)	
9L	STOR M(3, 28:39)	Modify address in 3R
9R	ADD M(2)	
10L	STOR M(4, 8:19)	Modify address in 4L
10R	JUMP M(3, 0:19)	Branch to 3L

Exercises

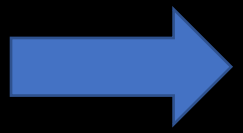
2.4. Given the memory contents of the IAS computer shown below, show the assembly language code for the program, starting at address 08A. Explain what this program does.

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

Exercises

Split the hex code into 2-hex left opcode, 3-hex left address, 2-hex right opcode, and 3-hex right address (each hex is 4 bits).

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

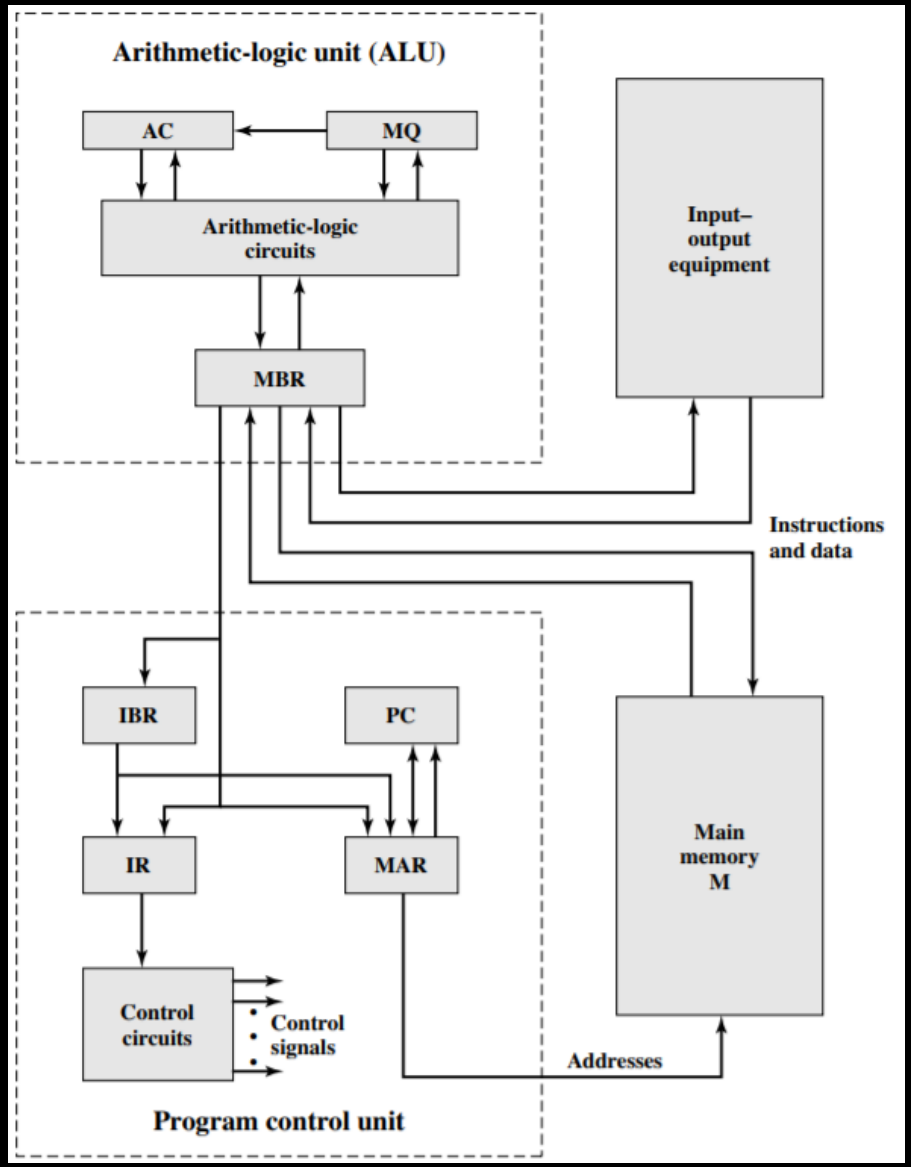


Address	Contents
08A	LOAD M(0FA) STOR M(0FB)
08B	LOAD M(0FA) JUMP +M(08D)
08C	LOAD –M(0FA) STOR M(0FB)
08D	

The program stores the positive value of content at memory address 0FA into memory address 0FB

Exercises

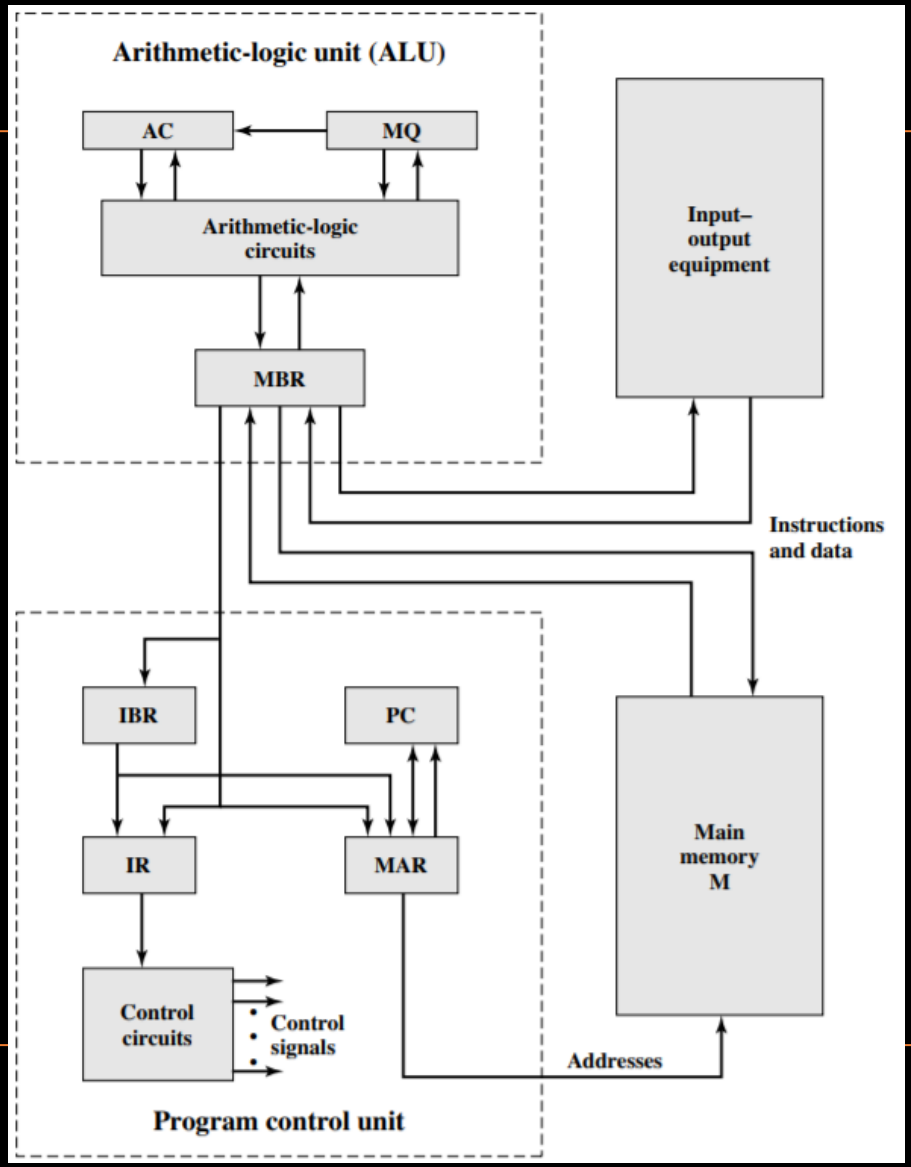
2.5. In Figure 2.3, indicate the width, in bits, of each data path (e.g., between AC and ALU).



Exercises

2.5. In Figure 2.3, indicate the width, in bits, of each data path (e.g., between AC and ALU).

All the data paths to/from MBR are 40 bits.
All the data paths to/from MAR are 12 bits.
All the data paths to/from IR are 8 bits.
All the data paths to/from IBR are 20 bits.
All the data Paths to/from AC is 40 bits.
All the data Paths to/from MQ is 40 bits.



TASK

- What is *Intel x86 architecture* and *ARM* architecture?
  - What is the difference?
- What is CMOS?