Lecture.3

# Computer Evolution and Performance

## ENIAC - background

- Electronic Numerical Integrator And Computer
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
- Used until 1955

# ENIAC - details

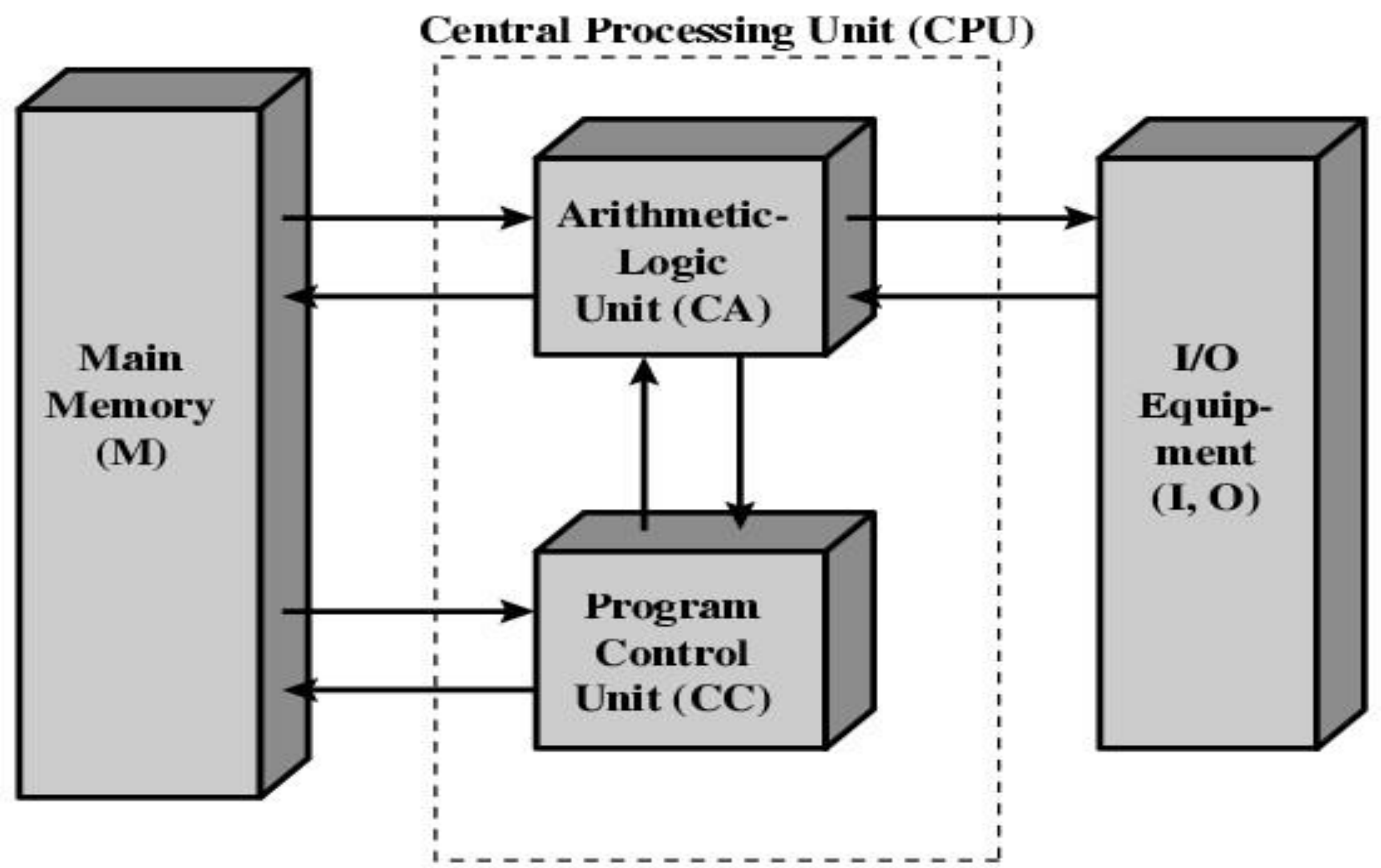
- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second

## von Neumann/Turing

- Stored Program concept
- **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**) computer
- Completed 1952

# Structure of von Neumann machine


Central Processing Unit (CPU)
Main Memory (M)
Arithmetic-Logic Unit (CA)
Program Control Unit (CC)
I/O Equip-ment (I, O)

# IAS - details

1000 x 40 bit words, each word representing

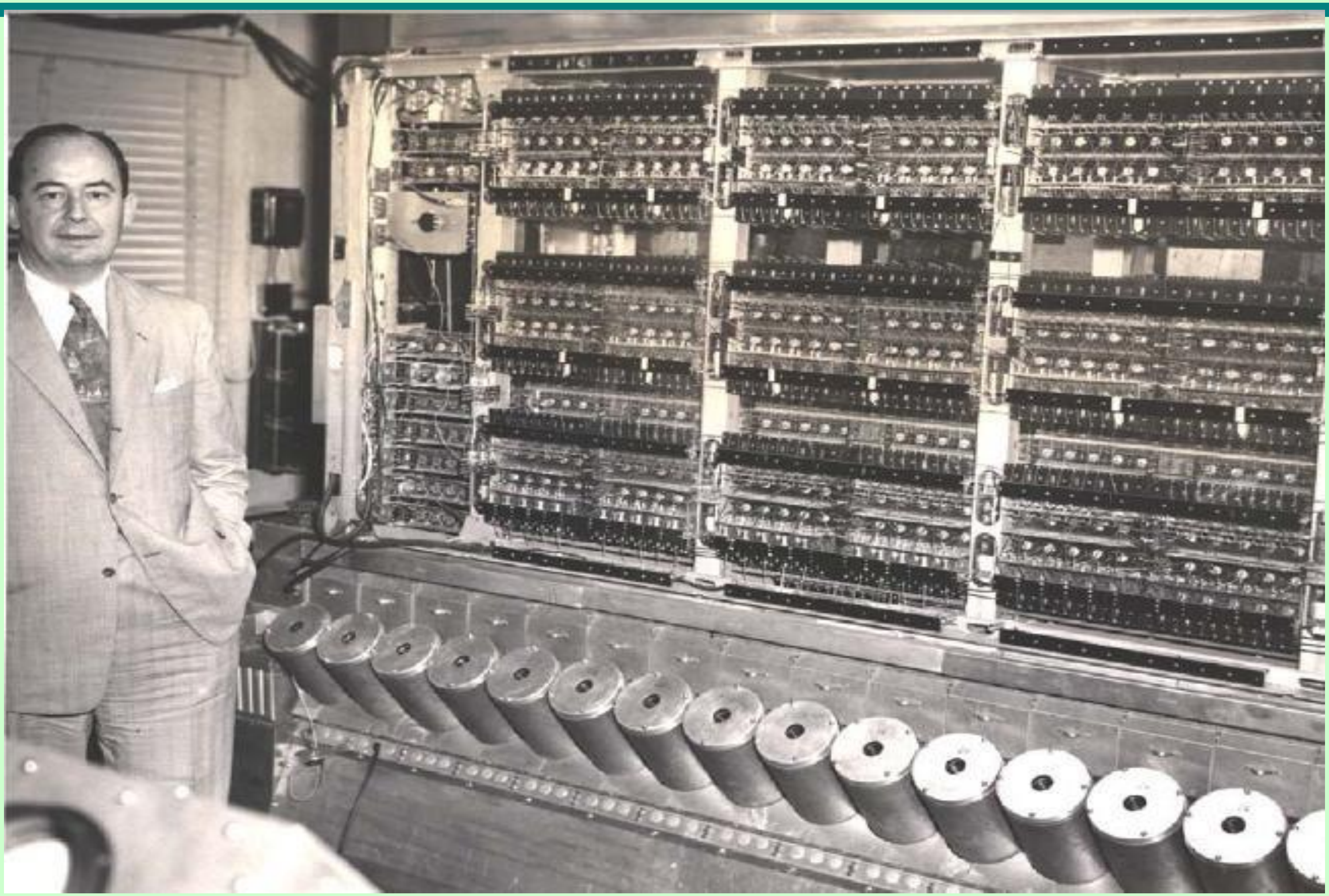
- One 40-bit binary number — Two 20-bit instructions:
  - 8 bits opcode
  - 12 bits address

Set of registers (storage in CPU)

- Memory Buffer Register
- Memory Address Register
- Instruction Register
- Instruction Buffer Register

- Program Counter
- Accumulator
- Multiplier Quotient

# John von Neumann and the IAS machine, 1952



# IAS organization

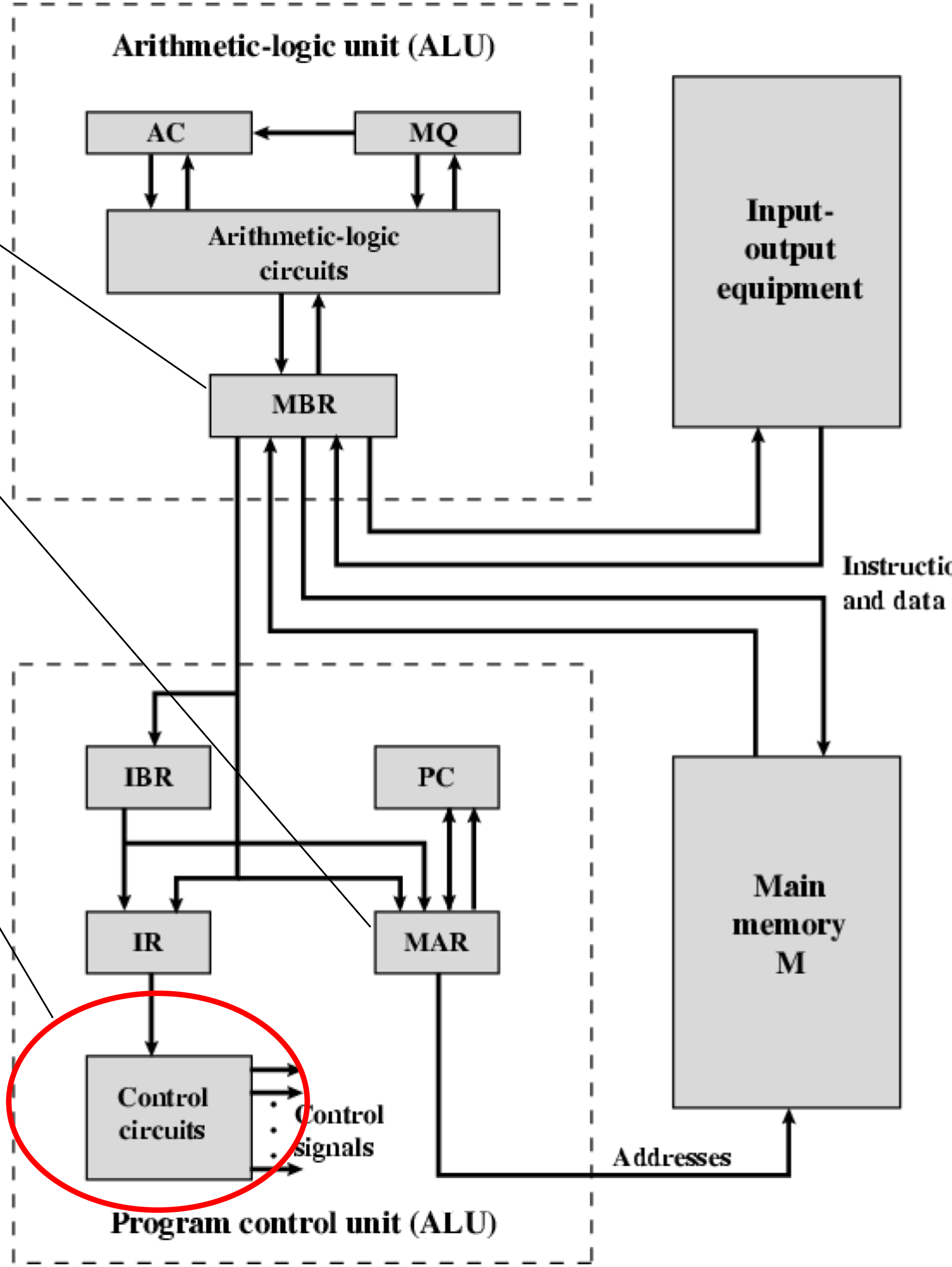
Memory Buffer Register either sends data to or receives data from Mem. or I/O

Memory Address Register specifies which Mem. location will be read or written next

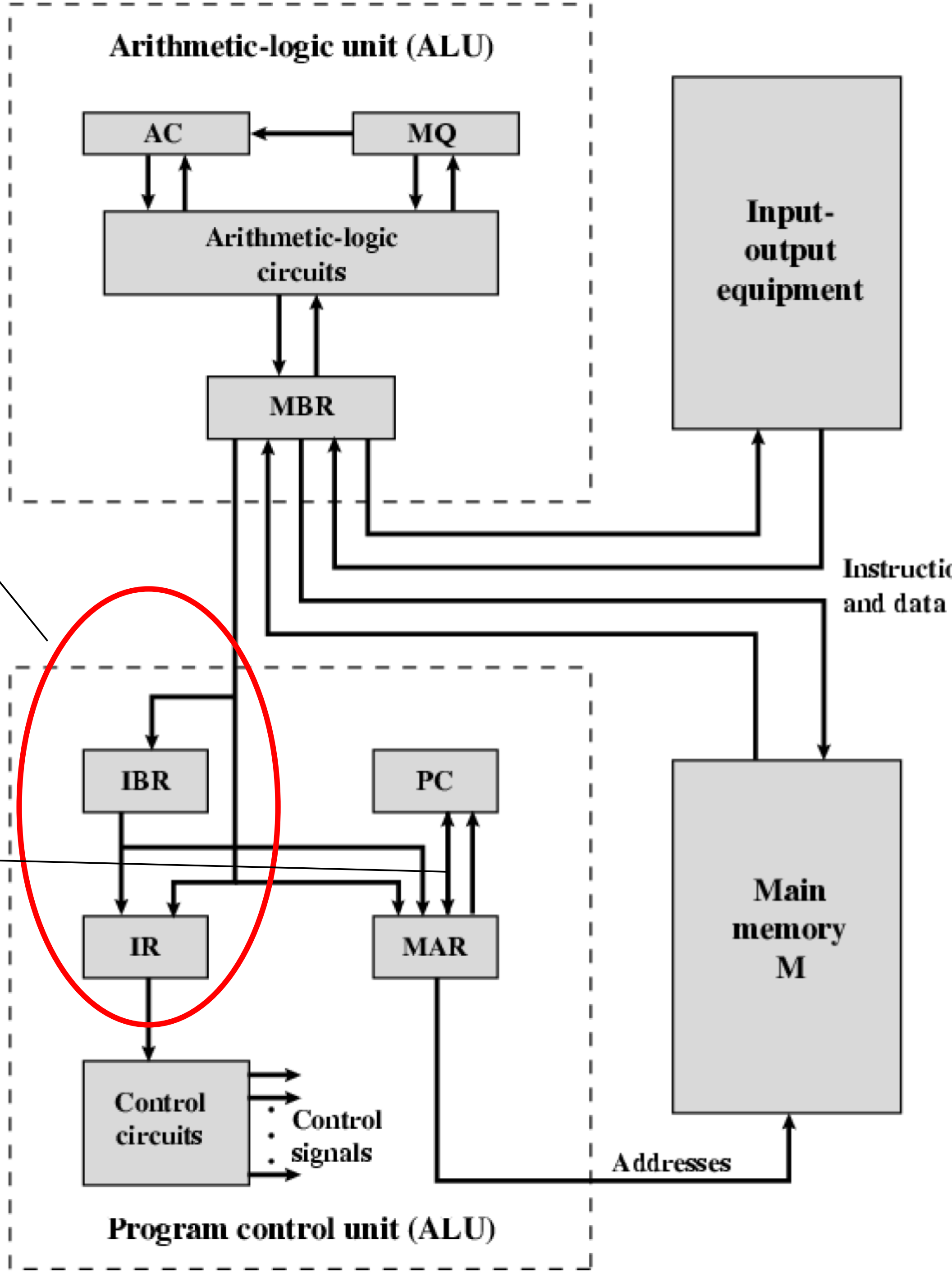
Control signals are set by the opcode part of the instruction bits.

Examples:

- Bring a new instruction from Mem. (fetch)
- Perform an addition (execute)



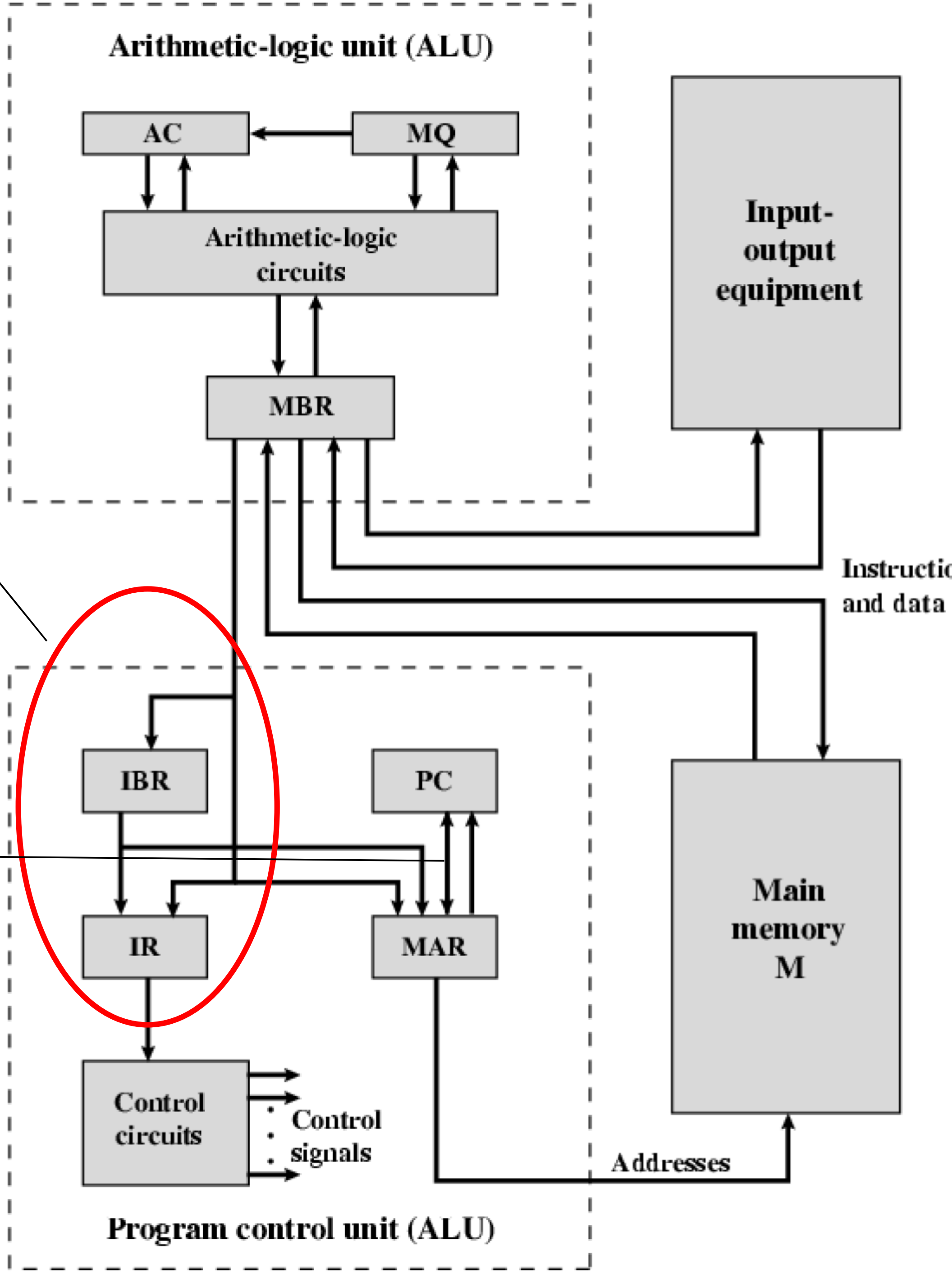
# IAS organization



# IAS organization

Hint: 2 instructions are stored in each memory word

Hint: the next instruction can be found either sequentially, or through a branch (jump)



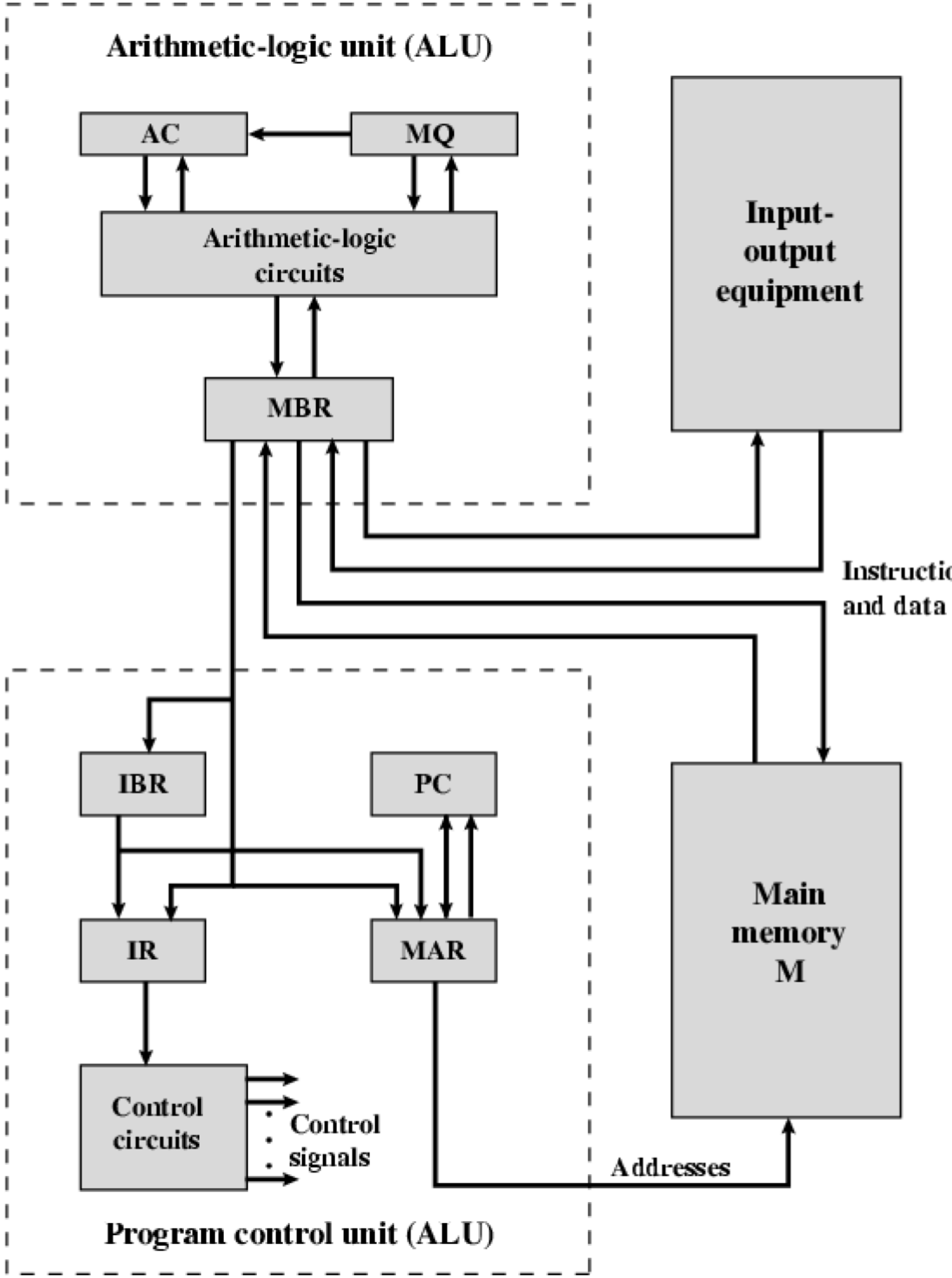
# IAS – The FETCH-EXECUTE Cycle

two step→ manner:FETCH load the binary code of the instr. from Memory (or IBR)

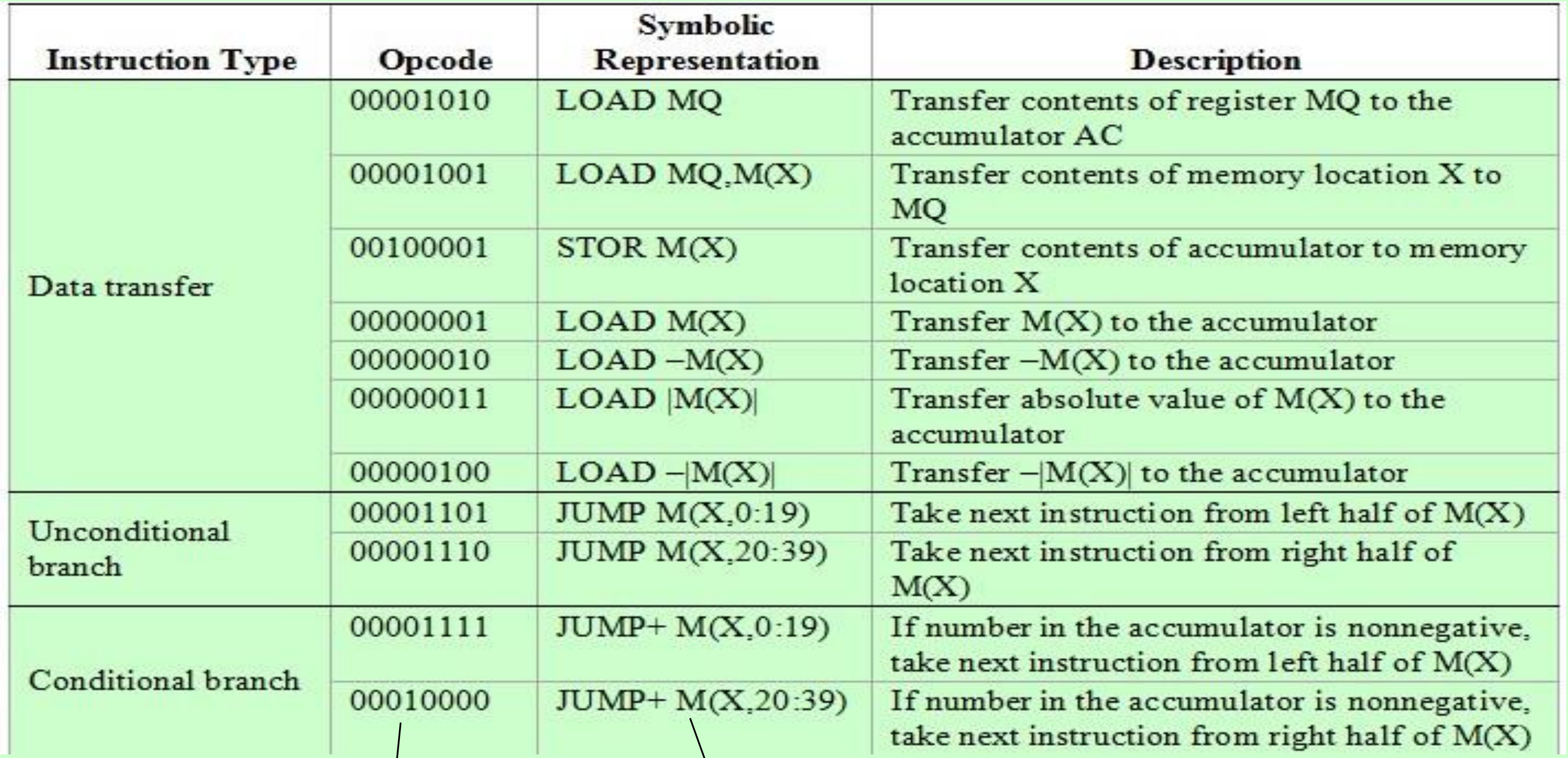
- —Opcode goes into IR
- —Address goes into MAR

- EXECUTE → send appropriate control signals to do what the instr. needs to do

# IAS FETCH-EXECUTE cycle



# IAS – instruction set (architecture)

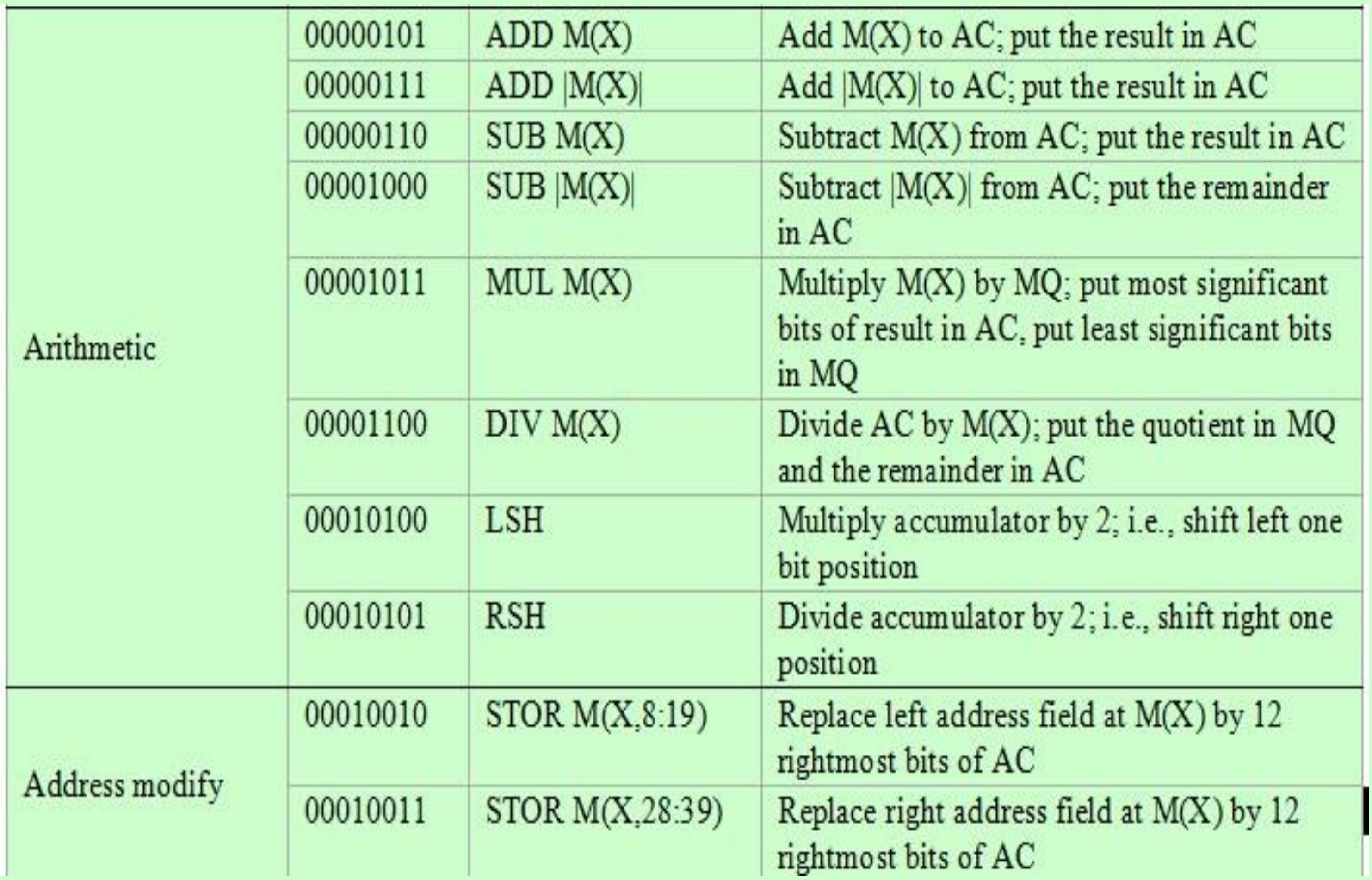


| Instruction Type | Opcode | Symbolic Representation | Description |
|---|---|---|---|
| Data transfer | 00001010 | LOAD MQ | Transfer contents of register MQ to the accumulator AC |
| | 00001001 | LOAD MQ,M(X) | Transfer contents of memory location X to MQ |
| | 00100001 | STOR M(X) | Transfer contents 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) |

Specifies one of 21 instructions

There was no assembly language back then!

# IAS – Instruction Set (continued)



| | | | |
|---|---|---|---|
| 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 |