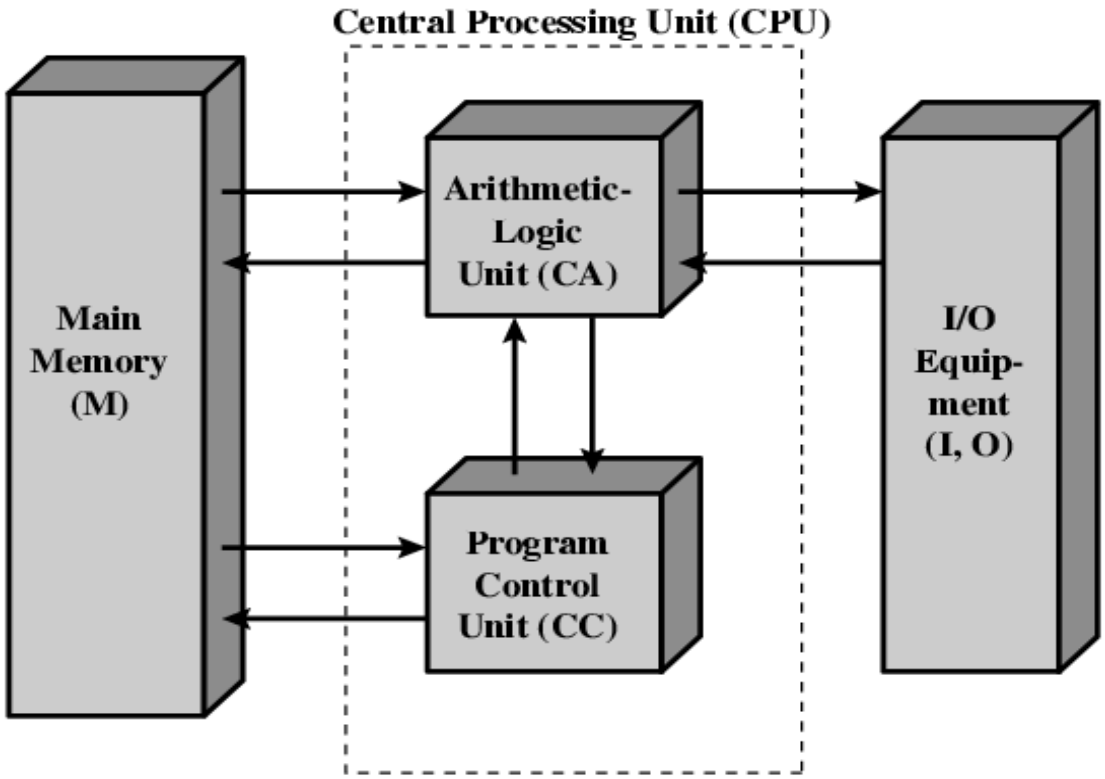
IAS Machine

Organization of the von Neumann machine

IAS – a prototype developed by John Von Neumann in 1946 at Princeton University. (Institute for Advanced Study)

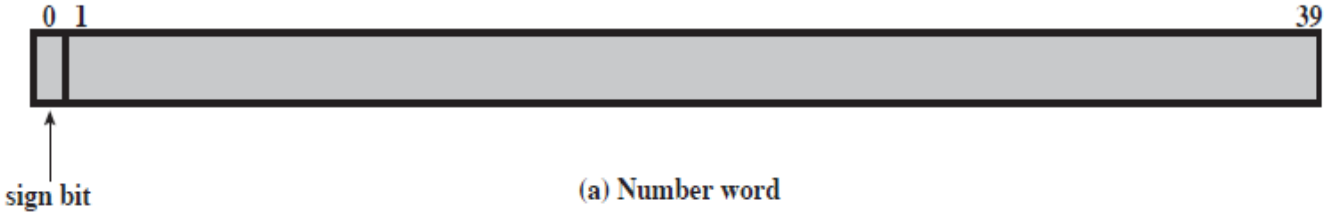
Structure of Von Neumann machine(IAS Computer)



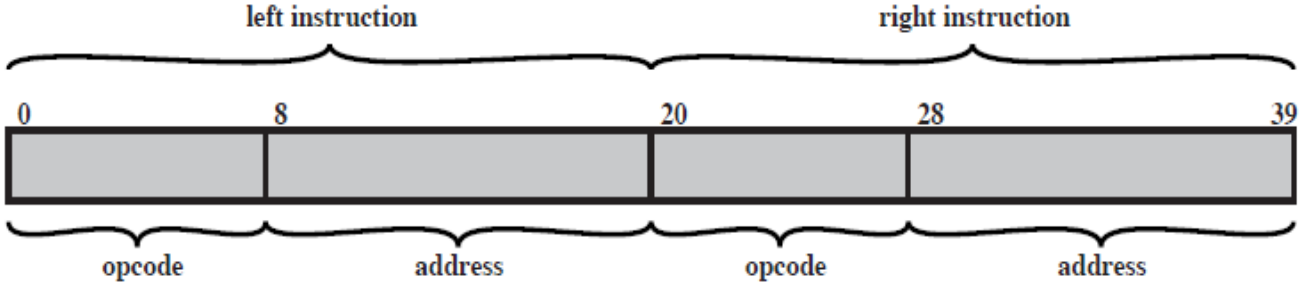
IAS – Contd..

Memory Formats

1



(a) Number word



(b) Instruction word

IAS – memory format

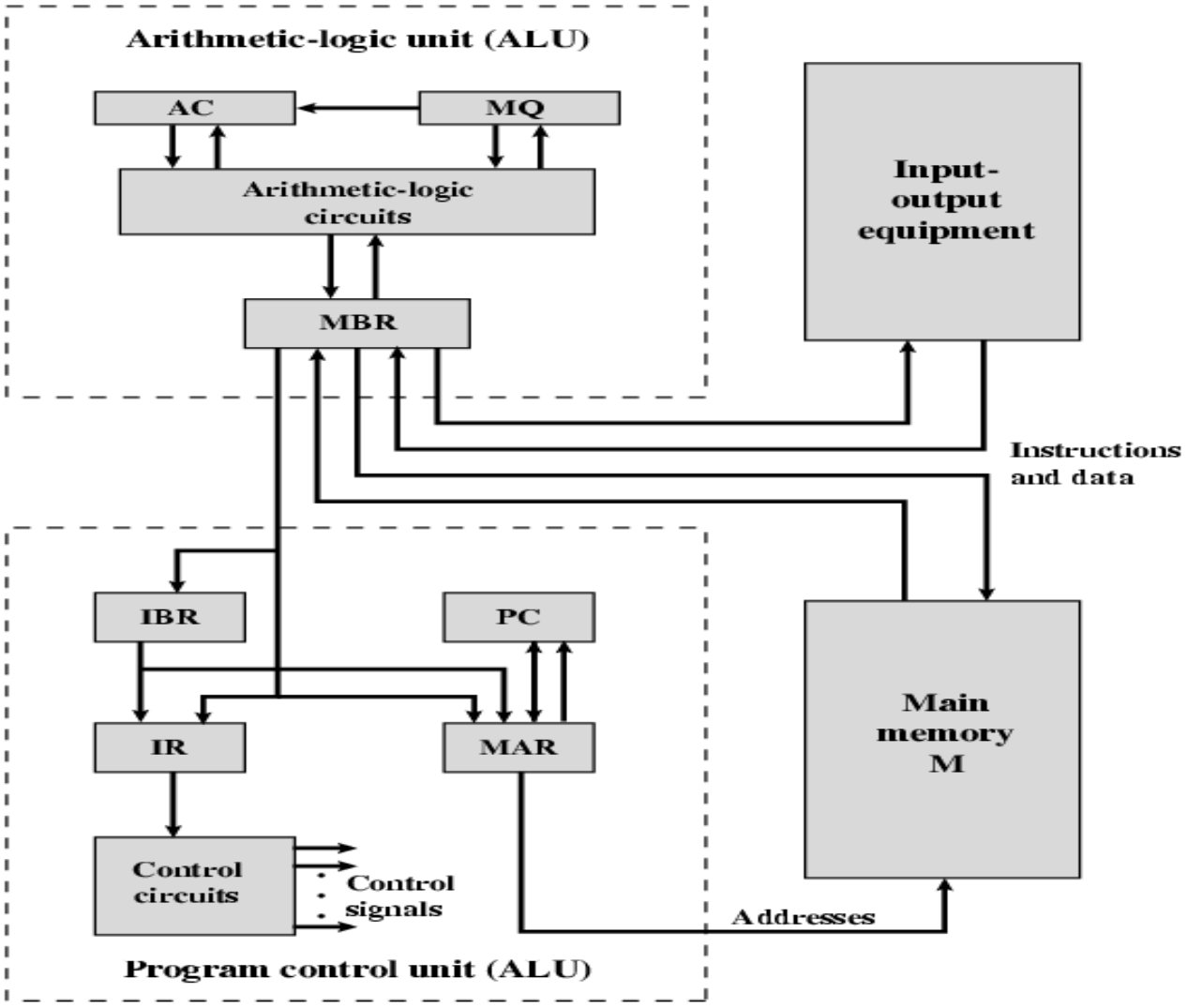
- 1000 x 40 bit words (1000 storage locations of 40 binary bits each)
  - Binary number(both data and instructions are stored here)
- Number Format:
  - Each number is represented by a sign bit and a 39 bit value.
- Instruction Format:
  - A word may contain 20 bit instruction with each instruction consisting 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 (0 to 999)

IAS – Contd..

IAS – Total 21 Instructions

- Data Transfer
- Unconditional Branch Instruction
- Conditional Branch Instruction
- Arithmetic
- Address Modify Instruction

Expanded Structure of IAS



Expanded structure of IAS computer

- **Set of registers (storage in CPU)**
  - **Memory Buffer Register (MBR)**
    - Contains a word to be stored in memory or sent to the I/O unit, or it 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 right hand instruction from a word in memory
  - **Program Counter (PC)**
    - Contains the address of the next instruction pair to be fetched from memory
  - **Accumulator (AC) & Multiplier Quotient (MQ)**
    - Employed to hold temporarily the right hand instruction from a word in memory. For eg. 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.

IAS Operation - Flowchart

MAR←PC

MBR←M[MAR]

IBR←MBR<20..39>

IR←MBR<0..7>

MAR←MBR<8..19>

MBR←M[MAR]

AC←MBR

IR←IBR<0..7>

MAR←IBR<8..19>

PC←PC+1

PC←MAR

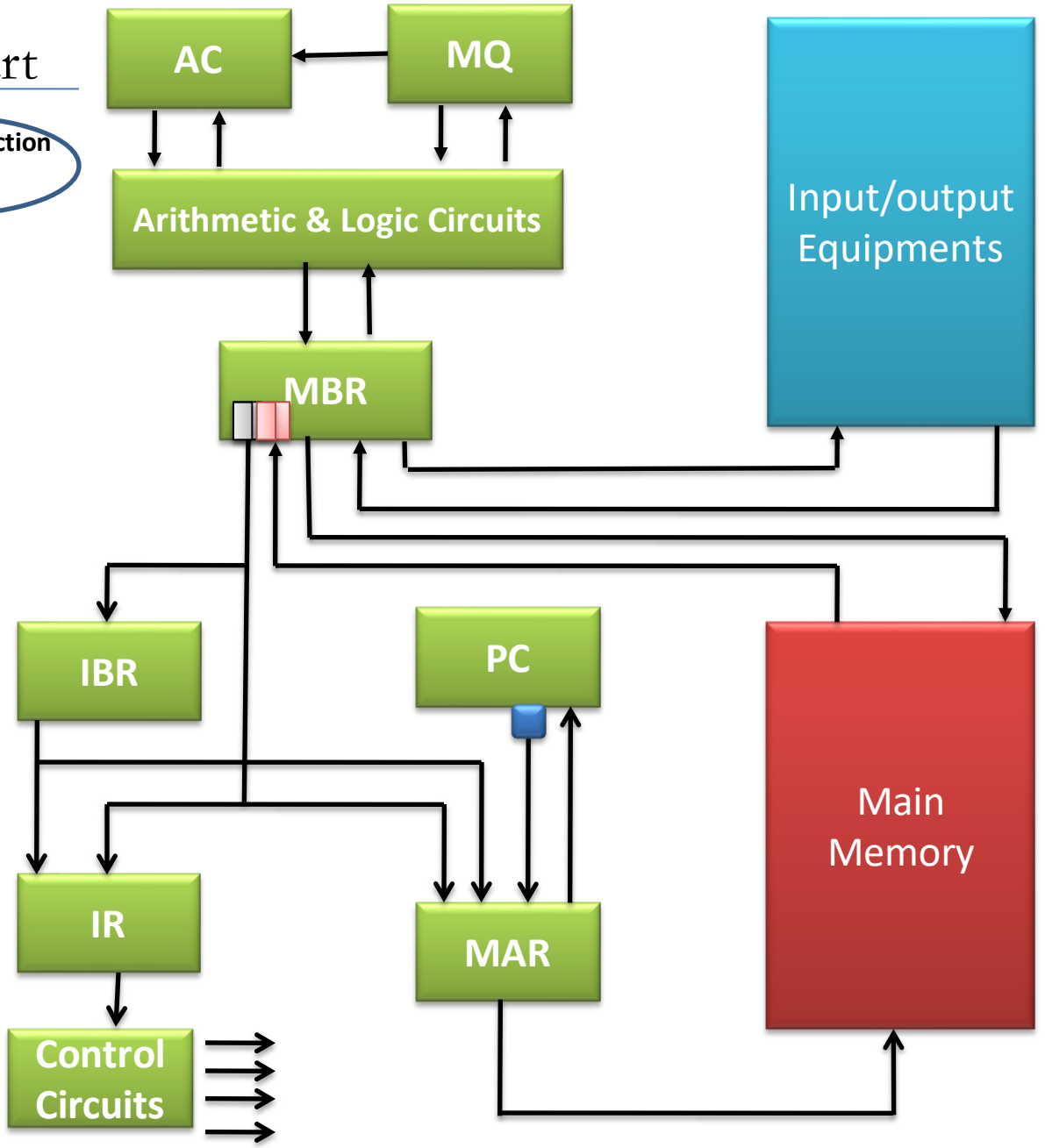
MBR←M[MAR]

AC← MBR

Is next instruction in IBR?

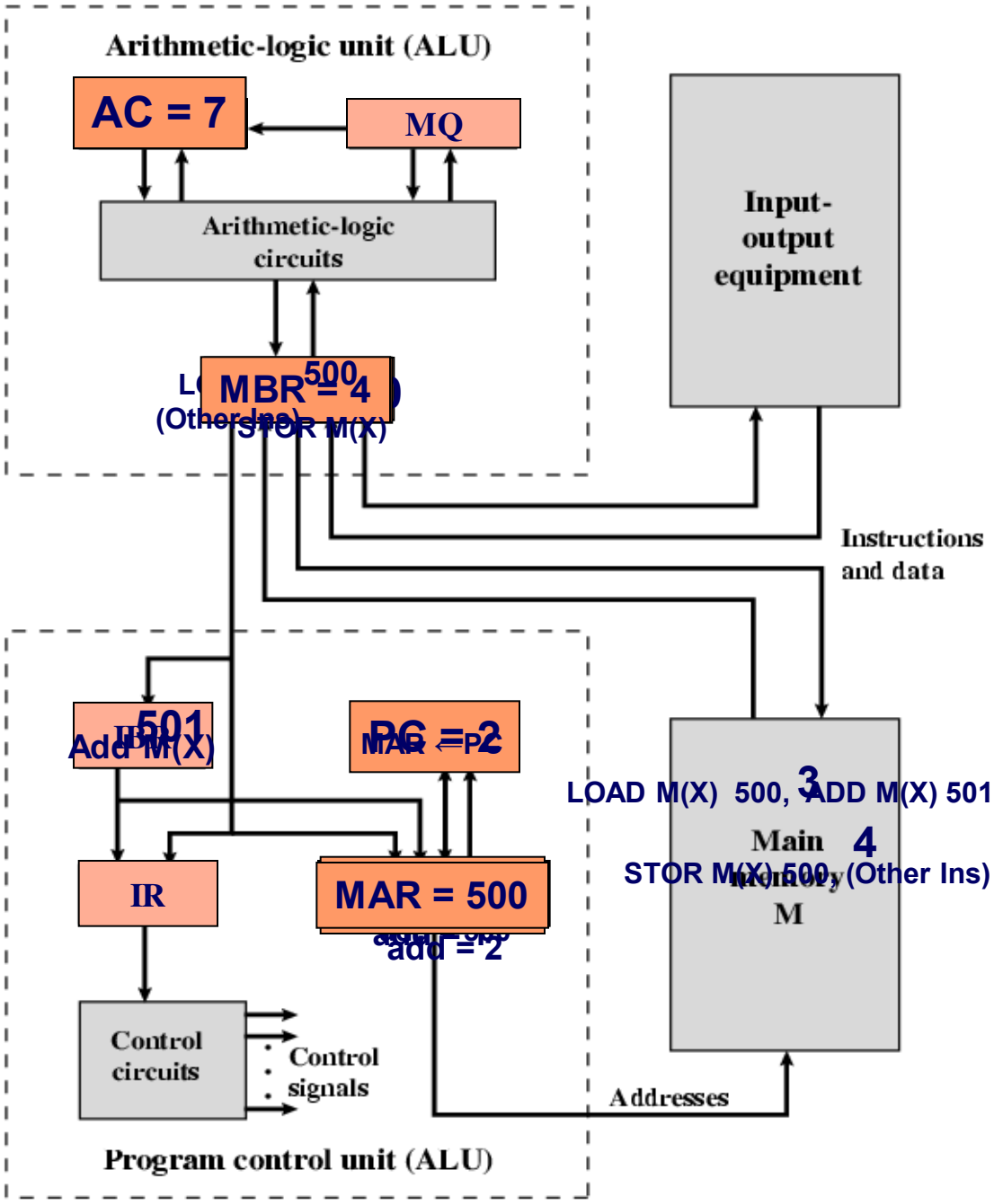
No

yes



MEMORY
1. LOAD M(X) 500, ADD M(X) 501
2. STOR M(X) 500, (Other Ins)
.....
500. 3
501. 4

PC	2
MAR	500
MBR	STOR M(X) 500, (Other Ins)
IR	STOR M(X)
IBR	(Other Ins)
AC	7

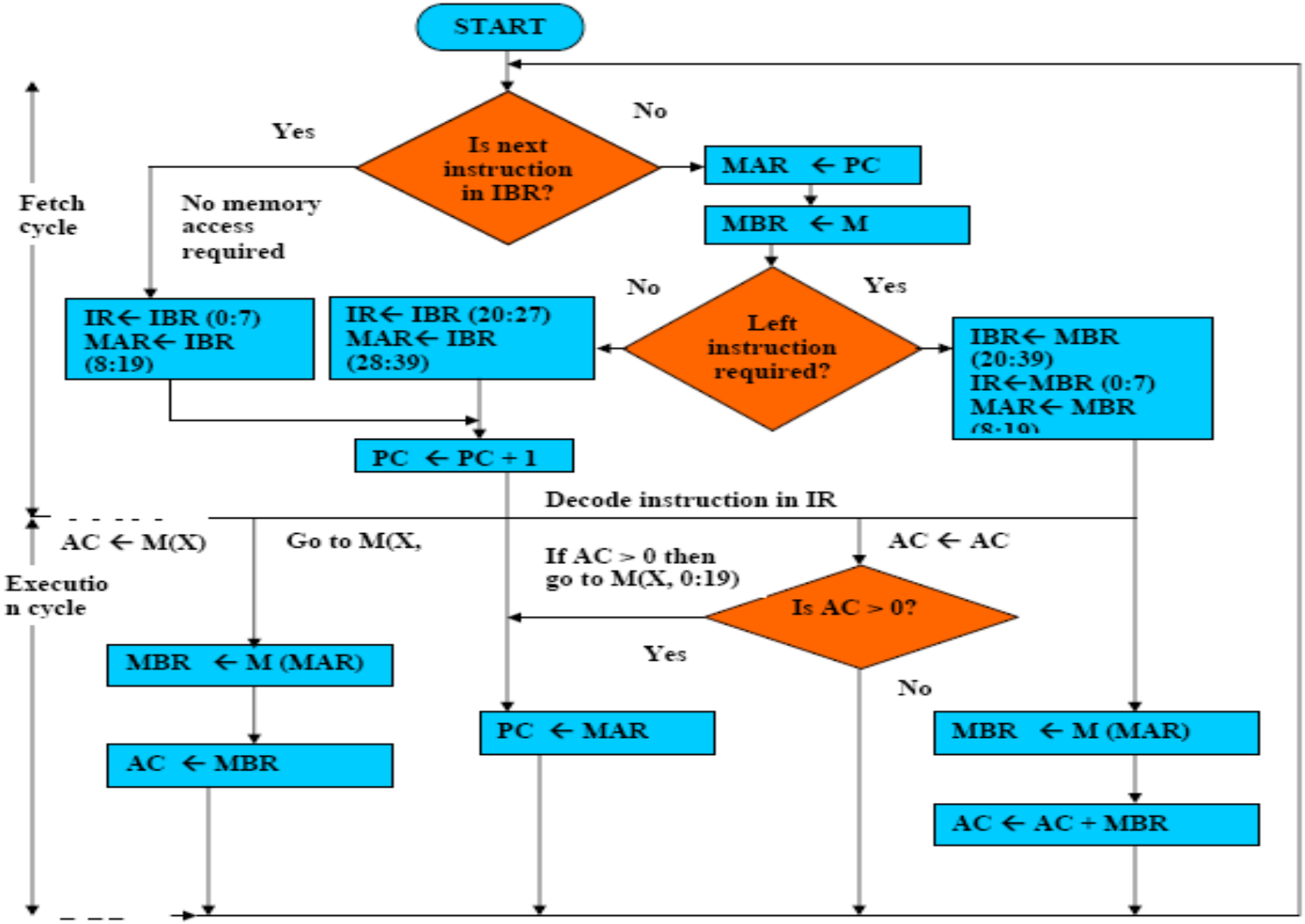


Register transfer operation for addition operation

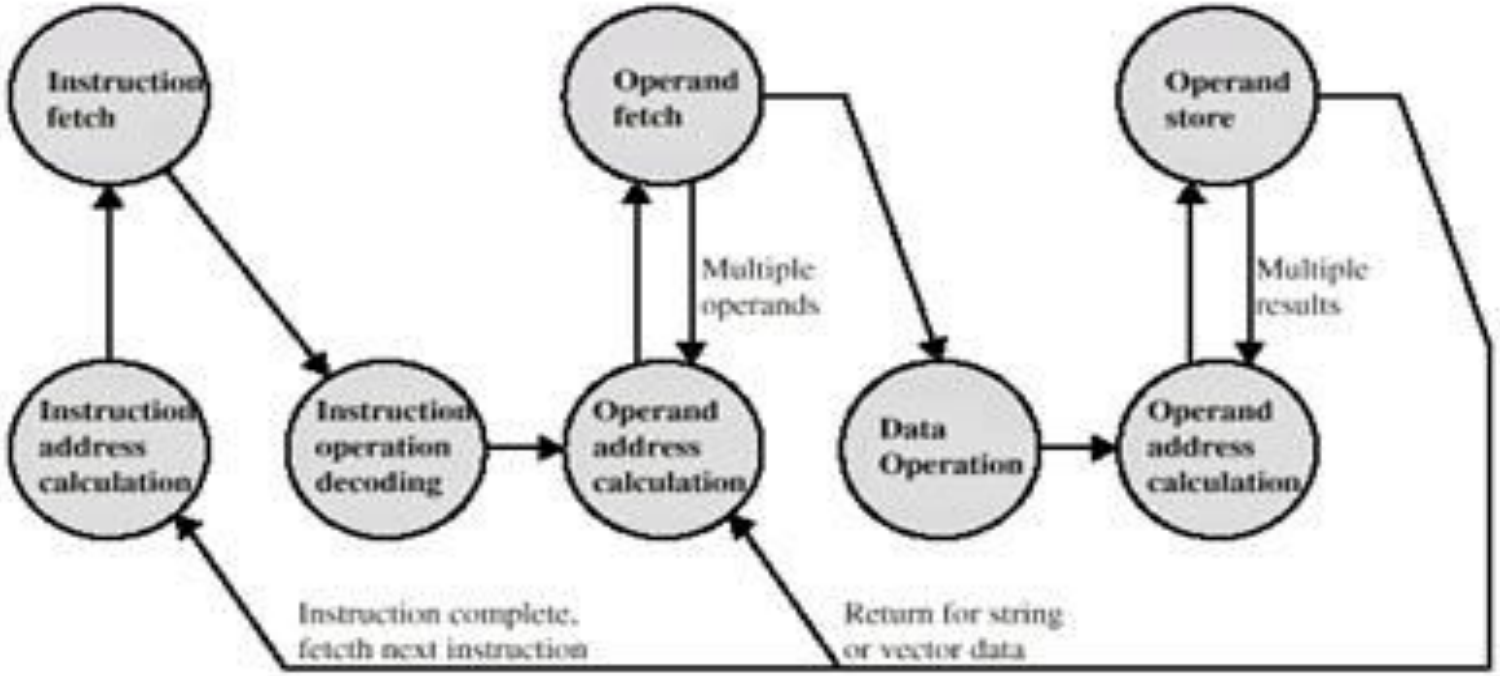
1. LOAD M(X) 500, ADD M(X) 501

- **Register transfer operations: (PC = 1)**
  - **MAR ← PC**
  - **MBR ← M[MAR]**
  - **IBR ← MBR[20:39]**
  - **IR ← MBR[0:7]**
  - **MAR ← MBR[8:19]**
  - **MBR ← M[MAR]**
  - **AC ← MBR**
  - **IR ← IBR[0:7]**
  - **MAR ← IBR[8:19]**
  - **MBR ← M[MAR]**
  - **AC ← AC + MBR**

Fetch / Execute Cycle



Instruction Cycle State Diagram



Instruction set: Collection of instructions that the CPU can execute

What an Instruction set should specify?

- Which Operation to perform (Opcode)
- Where to find the operand or operands (CPU registers, main memory or I/O port)
- Where to put the result, if there is result
- Where to find the next instruction

IAS Instruction set

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)

IAS Instruction set Contd..

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

- Write an ALP for the given expression using Von Neumann instruction set and write the register transfer notation for the same. Assume that B,C and D are available in the memory locations 450, 451 and 452 respectively.

- A = (B + C) * D
- X = A + B/2

ALP

- A = (B + C) * D

- Load M [450], Add M [451]
- Store M [453], Load MQ, M [453]
- Mul M [452], Store M[453]
- Load MQ, Store M[454]

- Complete the Register Transfer Notation

ALP

- X = A+ B/2

- Load M [450], RSH
- ADD M [451], Store M[452]

- Complete the Register Transfer Notation

ALP

If A<0

A = A+B

Else

A = A-B

AP

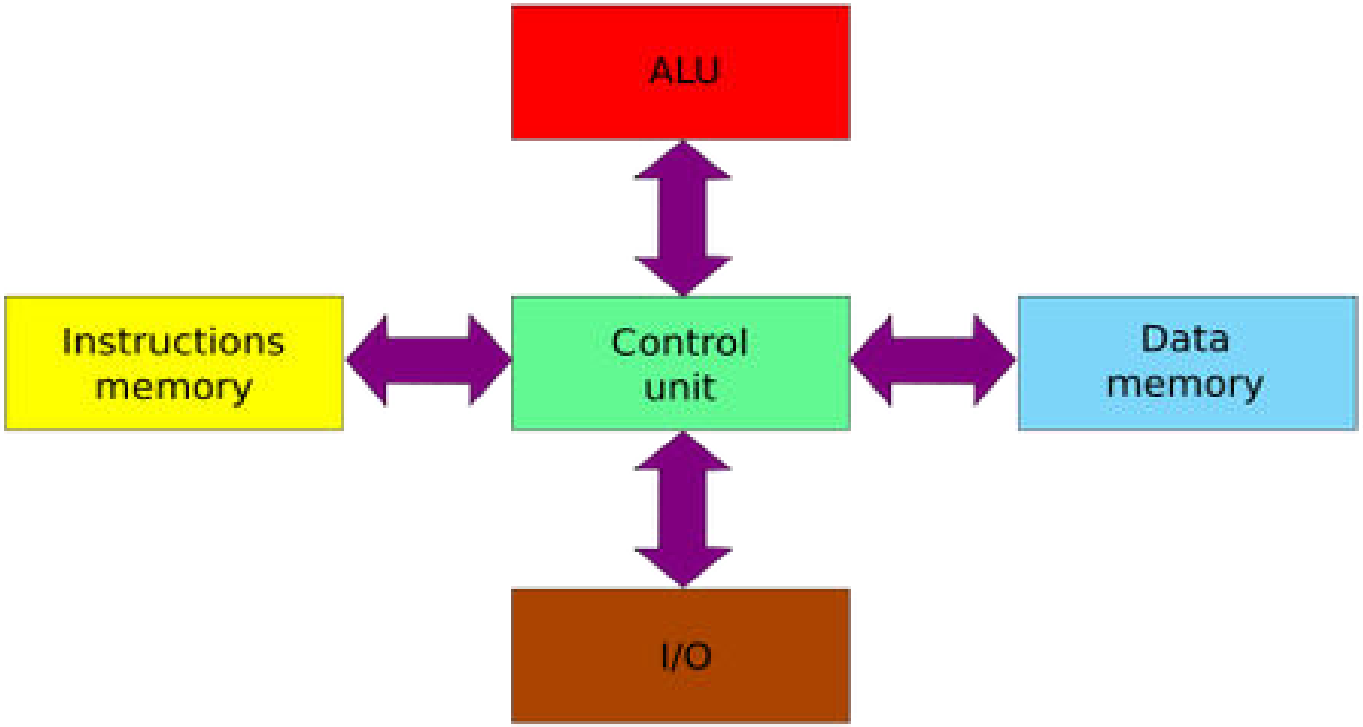
Assume M[450] = A
M[451] = B

100: Load M [450], JUMP +M[500, 0:19]
101: SUB M[451] , RHI

500: ADD M[451], RHI

Harvard Architecture

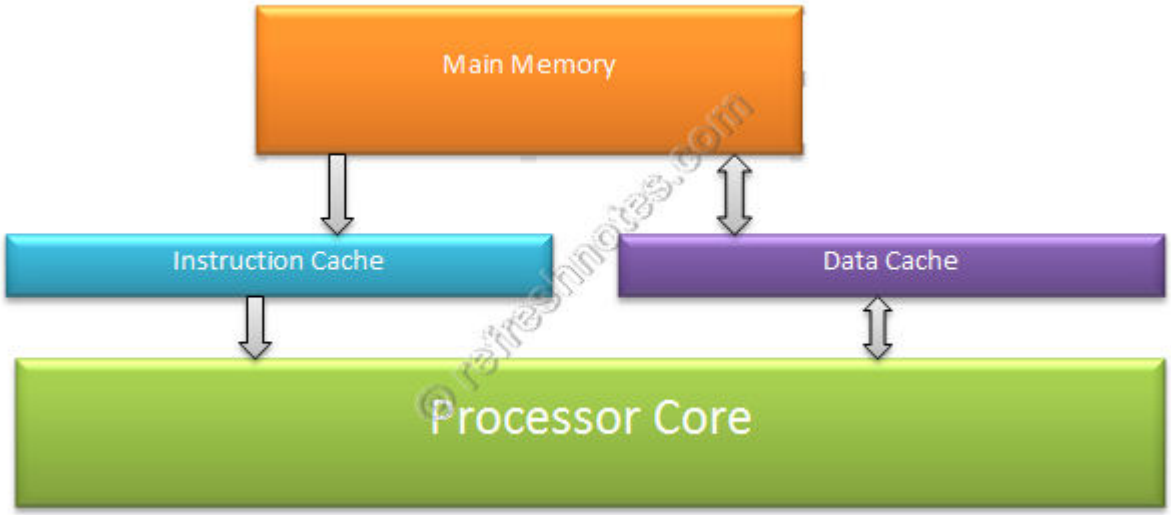
- The Harvard architecture is a computer architecture with physically separate storage and signal pathways for instructions and data.



Von Neumann Vs. Harvard Architecture

Von Neumann	Harvard Architecture
Based on Stored Program Concept	Modern Architecture based on Harvard Mark I Model
Same physical memory for instruction and data	Separate memory space for instruction and data
Processor takes two cycles to execute an instruction	Processor takes one cycle to execute an instruction
Simpler, cheaper control unit design	Complicated design, as it uses two buses
Data transfer and instruction fetch are not simultaneous	Data transfer and instruction fetch are simultaneous
Used in PCs, laptops etc.	Used in microcontrollers….

Modified Harvard Architecture



- Split Cache Architecture