Digital System Design

Slide – 11

SAP-I

Introduction

- **SAP – Simple As Possible Computer**
- First stage in the evolution towards modern computers
- very basic model of a microprocessor explained by Albert Paul Malvino
- Main purpose is to introduce all the crucial ideas behind computer operations
- Being a simple computer, also covers many advanced concepts
- A bus organized computer
- All registers connected to the W bus with the help of tri state buffers

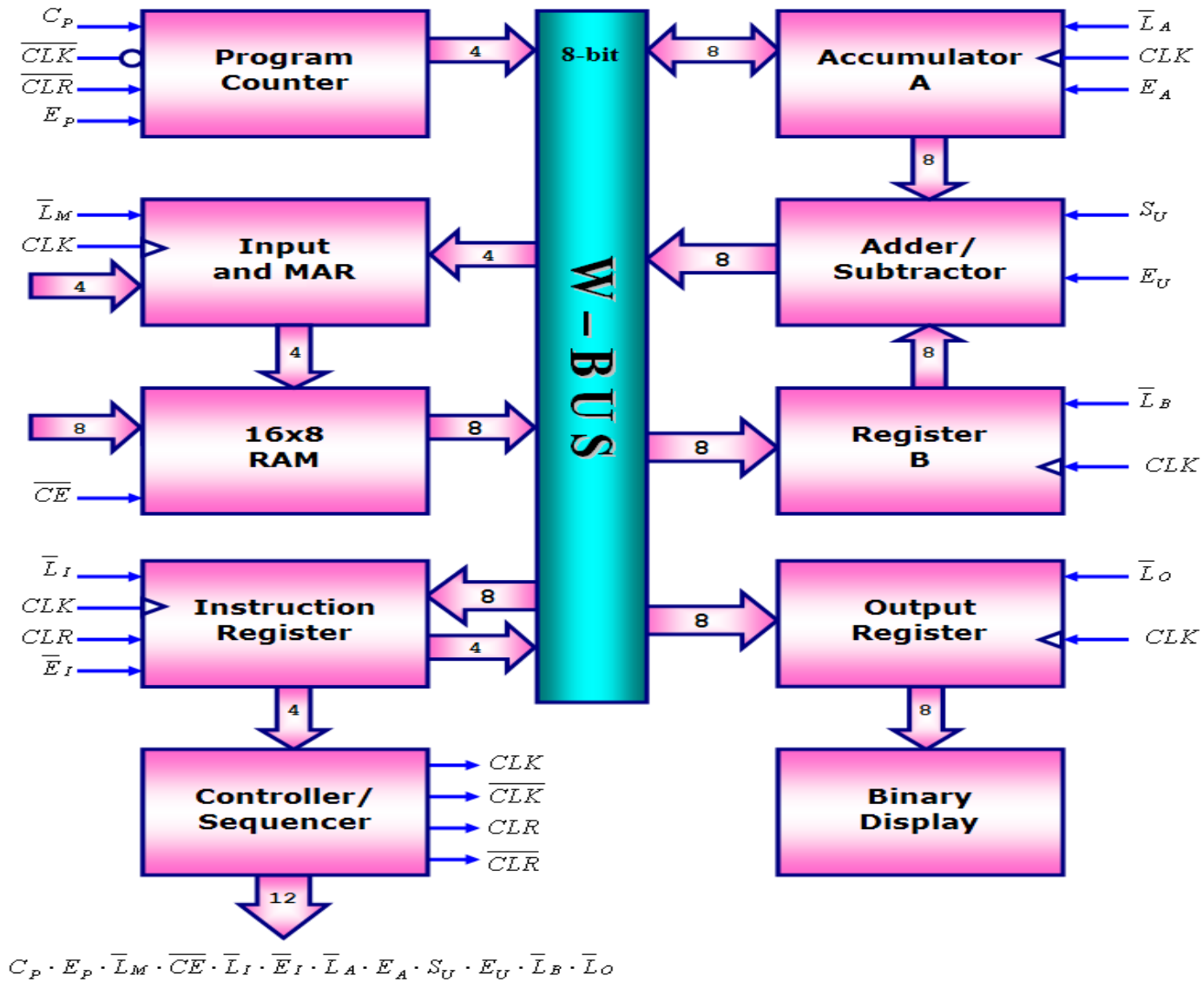
Introduction (Contd.)

- primary purpose is to develop a basic understanding of -
  I. how a microprocessor works
  II. interacts with memory and other parts of the system like input and output

- Instruction set is very limited and simple

SAP – I Architecture

Block Diagram of SAP-I Architecture



SAP – I Architecture (Contd.)

Main Features:

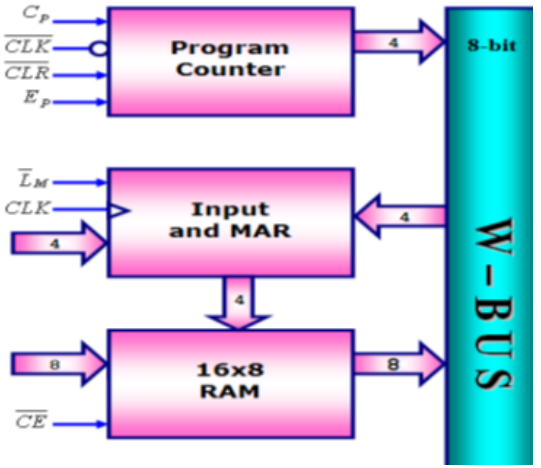
- One output device with 8 LEDs
- 16 bytes of RAM (i.e. 16 memory locations with 8 bits each)
- 5 instructions
    I. 3 with 1 operand
    II. 2 with implicit operands
- Accumulator Architecture
    I. Accumulator
    II. Out Register
    III. B Register
    IV. Memory Address Register (MAR)
    V. Instruction Register (IR)

SAP – I Architecture (Contd.)

- 8 bit "W" bus
- 4 bit program counter, only counts up, it starts counting from 0 and counts up to 15
- 4 bit Memory Address Register (MAR)
- 16 Byte Memory (RAM)
- 8 bit (1 Byte) Instruction Register (IR)
- 6 cycle controller with 12 bit microinstruction word
- 8 bit Accumulator
- 8 bit B Register
- 8 bit adder/ subtractor
- 8 bit Output Register

SAP – I Architecture (Contd.)

Program Counter:

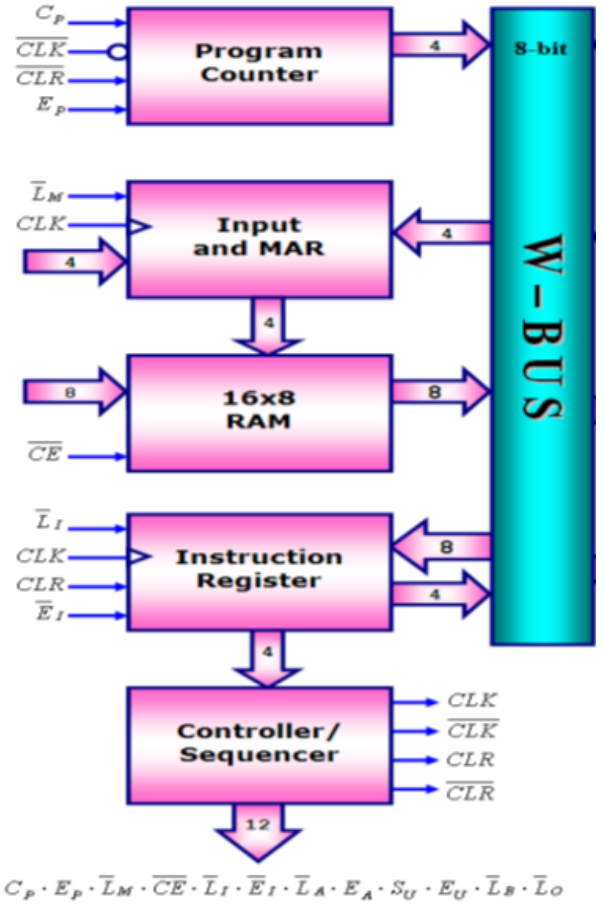


- Instructions to be executed are placed at the starting addresses of memory, e.g. the first instruction of a program will be placed at binary address 0000, the second at address 0001
- Now to execute one instruction, first step is to generate the address at which this instruction is placed in memory
- So this address is generated by (4 bit) Program Counter , that counts from 0000 to 1111 (for total of 16 memory locations)
- If the value of program counter is 0100 , then the instruction at address at 4 will be executed next
- program counter is like a pointer register; it points to the address of next instruction to be fetched and executed

SAP – I Architecture (Contd.)

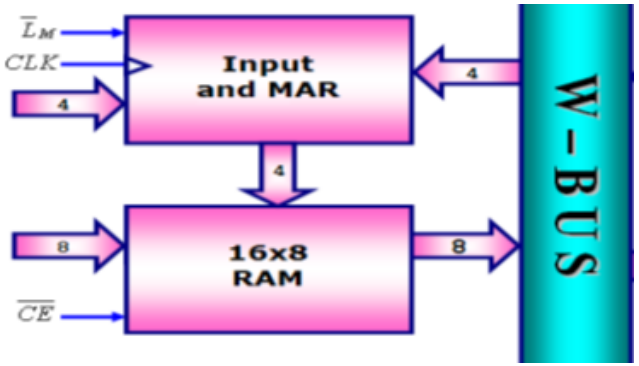
Input and Memory Address Register (MAR):

- The MAR stores the (4 bit) address of data and instruction which are placed in memory.
- When SAP-I is *Running Mode*, the (4 bit) address is generated by the Program Counter which is then stored into the MAR through W bus.
- A bit later, the MAR applies this 4 bit address to the RAM, where Data or instruction is read from RAM.



SAP – I Architecture (Contd.)

The RAM:

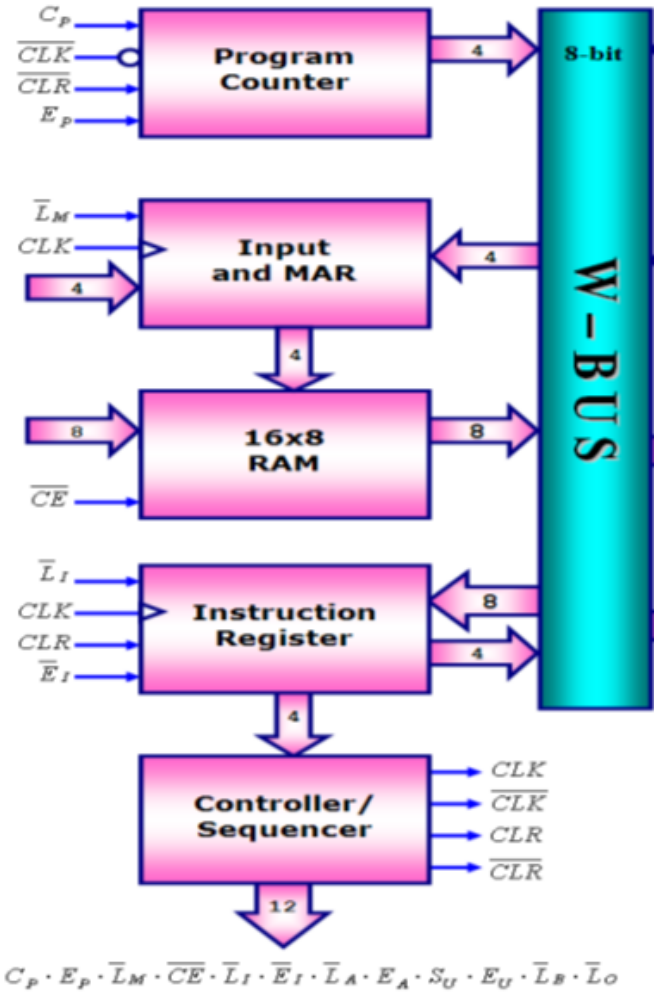


- In initial design, the RAM is a 16 x 8 static TTL RAM
- It means there are 16 memory locations (from 0 to 15) and each location contains an 8 bit of data or instruction.
- You can program the RAM by means of the switches to be used for address and data. This allows you to store a program and data in the memory before a computer run
- During a computer run , the RAM receives 4 bit addresses from the MAR and a read operation is performed
- in this way, the instruction or data stored in the RAM is placed on the W bus for use in some other part of the computer.

SAP – I Architecture (Contd.)

Instruction Register:

- When the instruction is placed at W bus from memory, the Instruction Register stores this instruction on the next positive clock edge.
- Part of the control Unit
- The contents of the instruction register are split into two nibbles:
  - The upper nibble goes directly to the block labeled "Controller sequencer”
  - The lower nibble is read onto the W bus when needed



SAP – I Architecture (Contd.)

Adder/Subtractor:

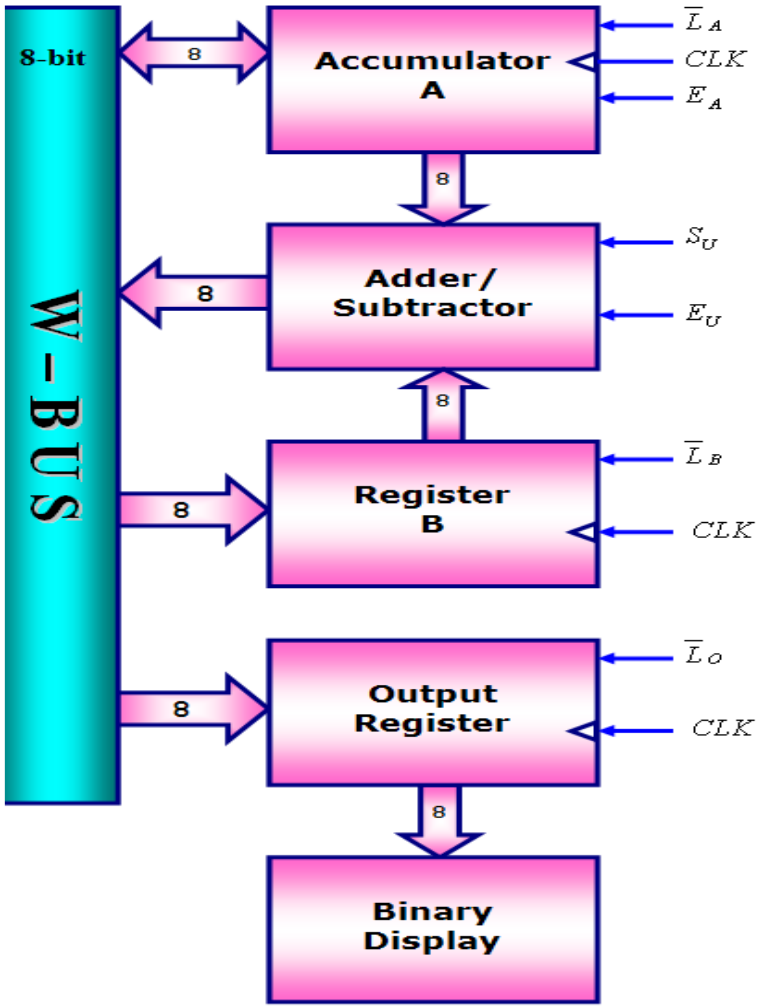
- SAP 1 uses a 2 's complement adder subtractor
- When input S_u is low (logic 0), the sum,

$$S = A + B$$

- When S_u is high (logic 1), the sum,

$$S = A + B' + 1$$

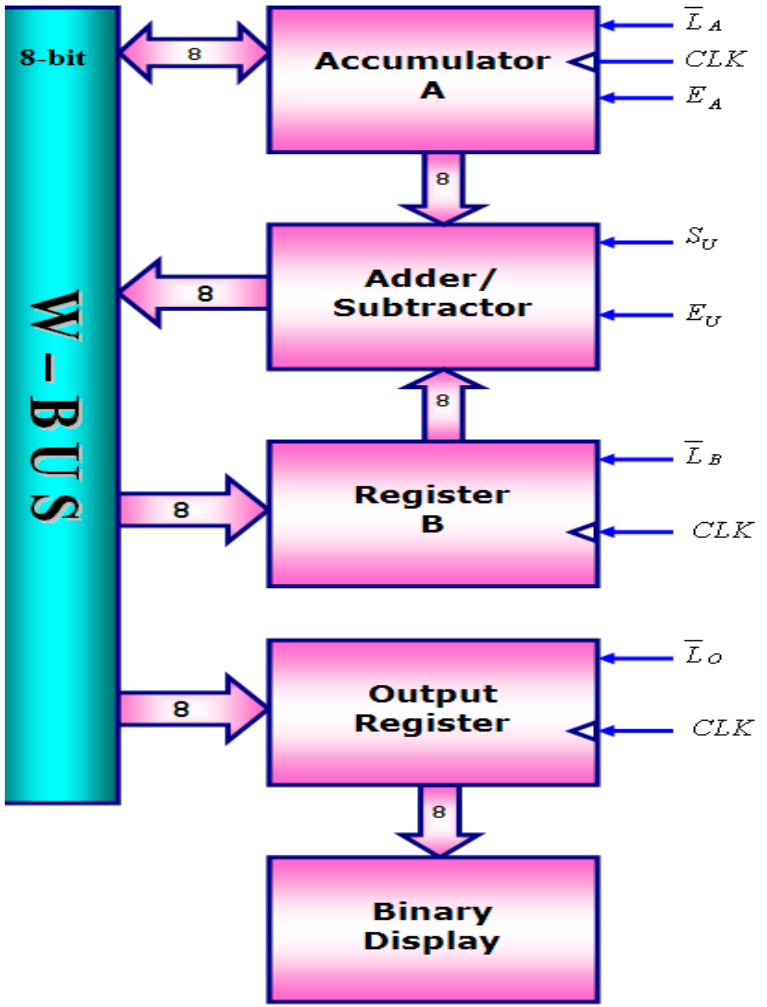
- The Adder subtractor is asynchronous and its contents change as soon as the input changes.
- When E_U is high, these contents appear on the W bus.



SAP – I Architecture (Contd.)

Accumulator:

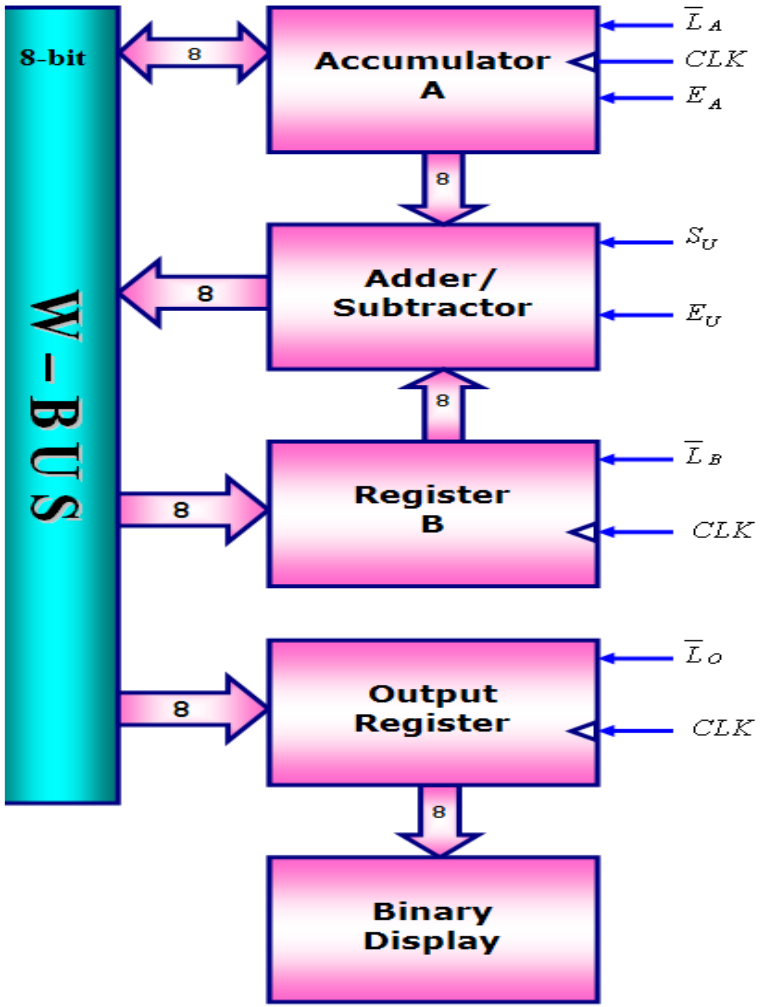
- To add/sub two 8 bit numbers A and B, the accumulator register stores the number A.
- The Accumulator has two outputs.
  I. One output goes to the adder/ subtractor
  II. Other goes to the W through tri state buffers
- It also stores the output of adder/subtractor, when L_A is low.
- It's value appears on W bus when E_A is high, which can then be read by output register.



SAP – I Architecture (Contd.)

B Register:

- A buffer register
- To add/sub two 8 bit numbers A and B, the B register stores the number B
- Supplies the number to be added or subtracted from the contents of accumulator to the adder/ subtractor
- When data is available at W bus and L_b goes low, B register loads that data.



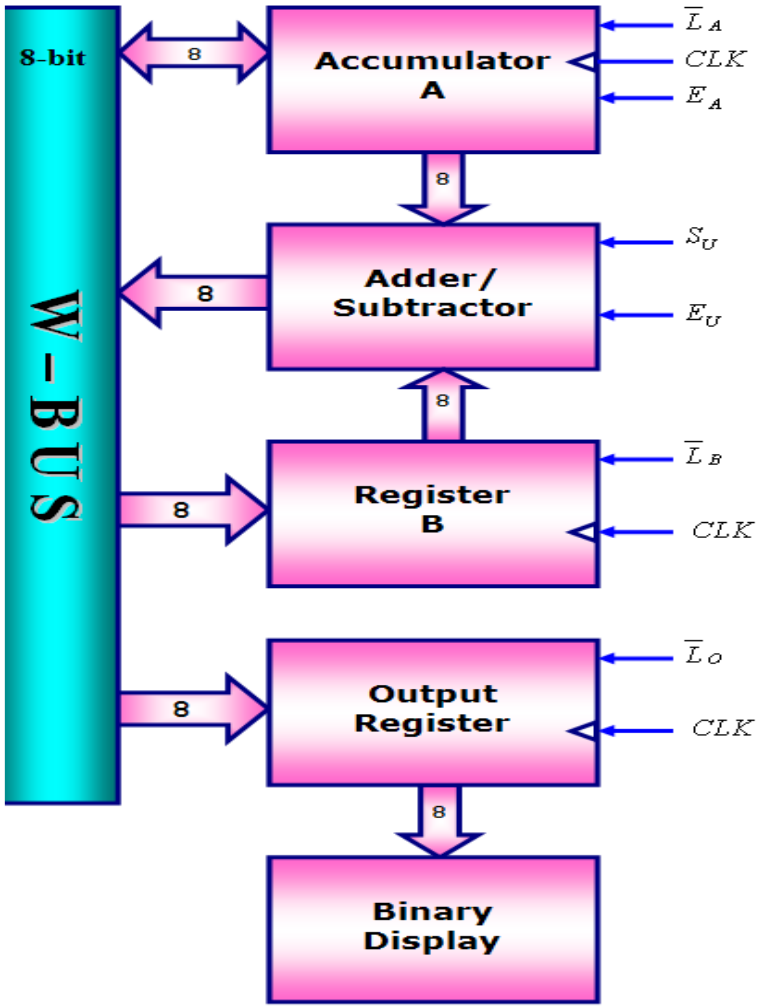
SAP – I Architecture (Contd.)

Output Register:

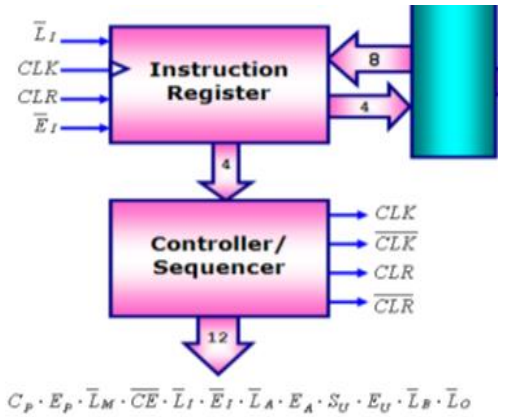
- At the end of an arithmetic operation the accumulator contains the word representing the answer
- Then answer stored in the accumulator register and loaded into the output register through W bus
- This is done in the next positive clock edge when **E_A is high and L_O is low**

Binary Display:

- Now this value can be displayed to the outside world with the help of LEDs or 7 Segments.



SAP –I Architecture (Contd.)



Controller Sequencer:

- Before each computer run, a $\overline{CLR}$ signal is sent to the PC and a CLR signal to the IR.
- This resets the PC to 0000 and wipes out the last instruction in the IR.
- A clock signal is sent to the buffer registers, which synchronizes the operation of the computer
- 12 bits coming out of the Controller Sequencer form a word that controls the rest of the computer.
- 12 wires carrying the control word are called the **Control Bus**.
- Control word has the format:

$$CON = \quad C_P \quad E_P \quad \overline{L}_M \quad \overline{CE} \quad \overline{L}_1 \quad \overline{E}_1 \quad \overline{L}_A \quad E_A \quad S_U \quad E_U \quad \overline{L}_B \quad \overline{L}_O$$

SAP –I Instruction Set

- Computer is a useless hardware until it is programmed
- This means loading step by step instructions into the memory before the start of a computer run
- Before you can program a computer, however, you must learn its instruction set, the basic operations it can perform.
- The SAP 1 instruction set follows:

SAP-1 INSTRUCTION SET		
Mnemonics	**Operation**	**Description**
LDA	ACC ← RAM[MAR]	Load RAM data into accumulator
ADD	ACC ← ACC + B	Add RAM data to accumulator
SUB	ACC ← ACC – B	Subtract RAM data from accumulator
OUT	OUT ← ACC	Load accumulator data into output register
HLT	CLK ← 0	Stop processing

SAP –I Instruction Set (Contd.)

1. LDA Instruction:

- LDA stands for “load the accumulator”
- A complete LDA instruction includes the hexadecimal address of the data to be loaded
- For example, LDA 8H means load the accumulator with the contents of memory location 8H
- Therefore, given RAM[8] = 1111 0000
- The execution of LDA 8H results in ACC = 1111 0000
- Similarly, LDA FH means "load the accumulator with the contents of memory location FH”

SAP –I Instruction Set (Contd.)

2. ADD Instruction:

- ADD 9H means: “add the data of memory location 9H with data of accumulator and save the result in accumulator”
- Suppose, ACC = 0000 0010 , RAM[9] = 0000 0011
- During the execution of ADD 9H:
  - First data at RAM address 9 is loaded into the B register to get B = 0000 0011
  - and instantly the adder/ subtracter forms the sum of A and B, SUM = 0000 0101
  - Second, this sum is loaded into the accumulator to get ACC = 0000 0101
- Similarly, the execution of ADD FH adds data at RAM address 15 to the accumulator and save the answer back in accumulator overwriting the previous value
- The negative numbers are stored in 2’s complement form

SAP –I Instruction Set (Contd.)

3. SUB Instruction:

- SUB 9H means: “subtract the data of memory location 9H from the data of accumulator and save the result in accumulator”
- Suppose, ACC = 0000 0011 , RAM[9] = 0000 0010
- During the execution of SUB 9H:
  - First data at RAM address 9 is loaded into the B register to get B = 0000 0010
  - and instantly the adder/ subtracter forms the diff. of A and B, Diff. = 0000 0001
  - Second, this diff. is loaded into the accumulator to get ACC = 0000 0001

SAP –I Instruction Set (Contd.)

4. OUT Instruction:

- The instruction OUT tells the SAP-1 computer to transfer the accumulator contents to the output port.
- After OUT has been executed, you can see the answer to the problem being solved on LEDs display.
- OUT is complete by itself; that is, you do not have to include an address when using OUT because the instruction does not involve data in the memory.

SAP –I Instruction Set (Contd.)

5. HLT Instruction:

- HLT stands for halt . This instruction tells the computer to stop processing data so it stops the clock
- HLT marks the end of a program
- You must use a HLT instruction at the end of every SAP 1 program; otherwise, you get computer trash (meaningless answers caused by runaway processing)
- HLT is complete by itself, you do not have to include a RAM word when using HLT because this instruction does not involve the memory

SAP –I Memory Reference Instructions

- LDA, ADD, and SUB are called memory-reference instructions because they use data stored in the memory.
- OUT and HLT, on the other hand, are not memory reference instructions because they do not involve the data stored in the memory.

Mnemonics:

- LDA, ADD, SUB, OUT, and HLT are the instruction set for SAP-1. Abbreviated instructions like these are called mnemonics (memory aids). Mnemonics are popular in computer work because they remind you of the operation that will take place when the instruction is executed

Op Codes of SAP - I

- To load instruction and data words into the SAP-1 memory , we have to use some kind of code that the computer can interpret.
- The number 0000 stands for LDA, 0001 for ADD, 0010 for SUB, 0000 for OUT, and 1111 for HLT.
- Because this code tells the computer which operation to perform, it is called an operation code (op code).
- Assembly language involves working with mnemonics when writing a program.
- Machine language involves working with strings of 0s and 1s.

TABLE 2, SAP-1 OP CODES

Mnemonics	Op Code
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

Example:

Program in Assembly		Program in Machine Language		
Address	Contents	Address	Contents in Binary	Contents in Hexadecimal
0H	LDA 9H	0000	0000 1001	09H
1H	ADD AH	0001	0001 1010	1AH
2H	ADD CH	0010	0001 1100	1CH
3H	SUB BH	0011	0010 1011	2BH
4H	OUT	0100	1110 1111	EFH
5H	HLT	0101	1111 1111	FFH
6H	FFH	0110	1111 1111	FFH
7H	FFH	0111	1111 1111	FFH
8H	FFH	1000	1111 1111	FFH
9H	10H	1001	0001 0000	10H
AH	18H	1010	0001 1000	18H
BH	14H	1011	0001 0100	14H
CH	20H	1100	0010 0000	20H
DH	FFH	1101	1111 1111	FFH
EH	FFH	1110	1111 1111	FFH
FH	FFH	1111	1111 1111	FFH

The 8080 and 8085 Microprocessors:

- The 8080 was the first widely used microprocessor.
- It has 72 instructions. The 8085 is an enhanced version of the 8080 with essentially the same instruction set (both are designed by Intel Corp.).
- The SAP-1 instructions are upward compatible with the 8080/8085 instruction set.
- In other words, the SAP-1 instructions LDA, ADD, SUB, OUT, and HLT are 8080/8085 instructions.
- Learning SAP instructions is getting you ready for the 8080 and 8085, two widely used microprocessors.

Fetch Cycle:

- The *control unit* is the key to a computer's automatic operation. The control unit generates the control words that fetch and execute each instruction.
- While each instruction is fetched and executed, the computer passes through different *timing states* (*T* states), time intervals during which register contents change.
- Ring Counter has an output of

$$T = T_6T_5T_4T_3T_2T_1$$

- At the beginning of a computer run, the ring word is

$$T = 000001 = T_1$$

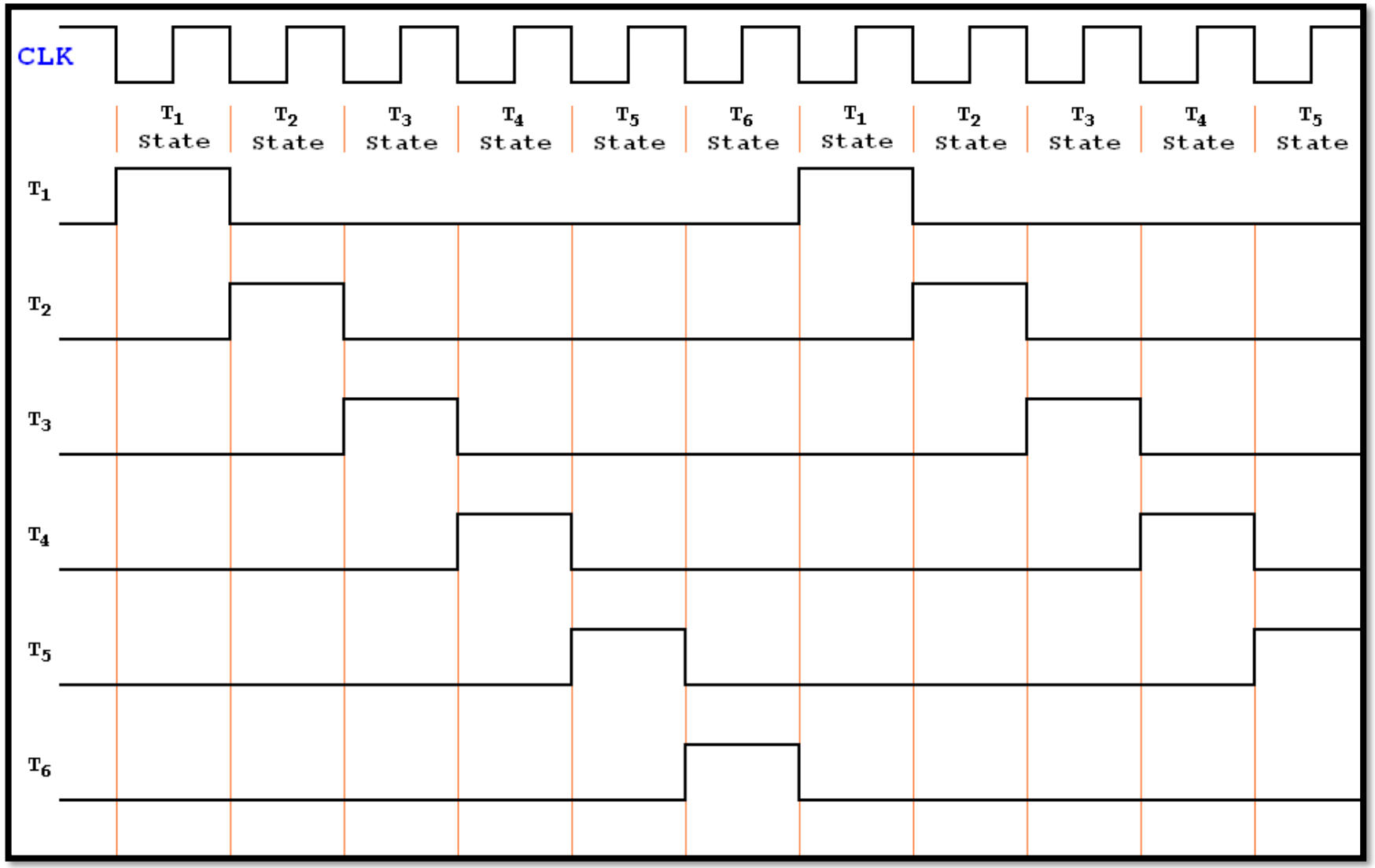
- Successive clock pulses produce, ring words of:

$$T = 000010 = T_2$$
$$T = 000100 = T_3$$
$$T = 001000 = T_4$$
$$T = 010000 = T_5$$
$$T = 100000 = T_6$$

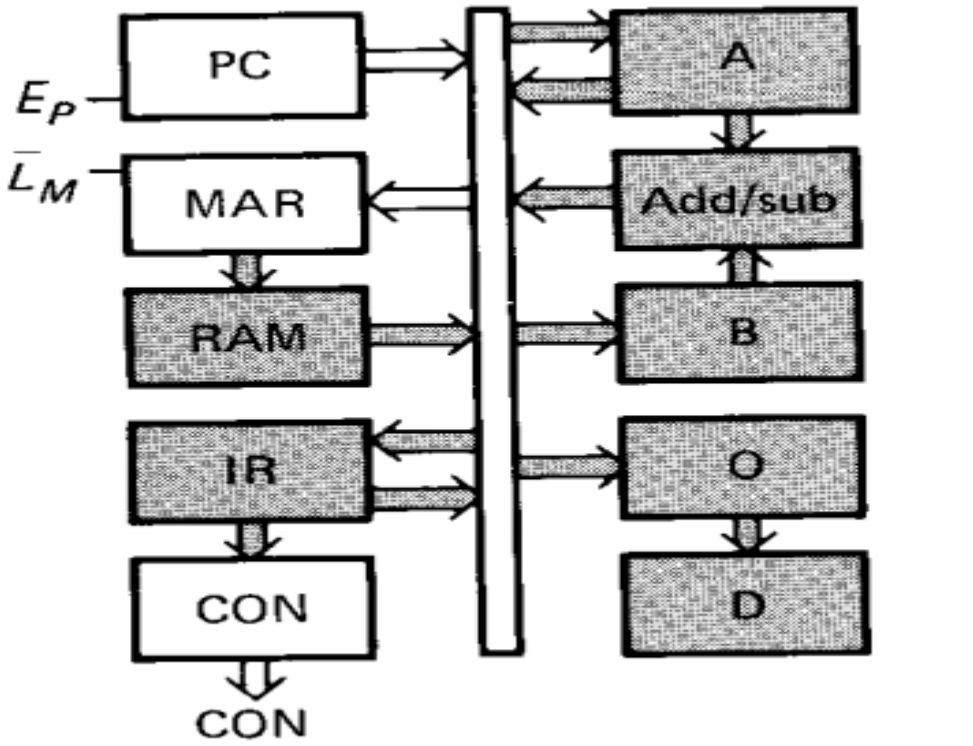
Fetch Cycle and Ring Counter:

- Then, the ring counter resets to 000001, and the cycle repeats.
- Each ring word represents one T state.
- The initial state T1 starts with a negative clock edge and ends with the next negative clock edge.
- During this T state, the T1 bit out of the ring counter is high.
- During the next state, T2 is high; the following state has a high T3; then a high T4; and so on.
- The ring counter produces six T states. **Each instruction is fetched and executed during these six T states.**
- A positive CLK edge occurs midway through each T state.

Ring Counter Timing Diagram:



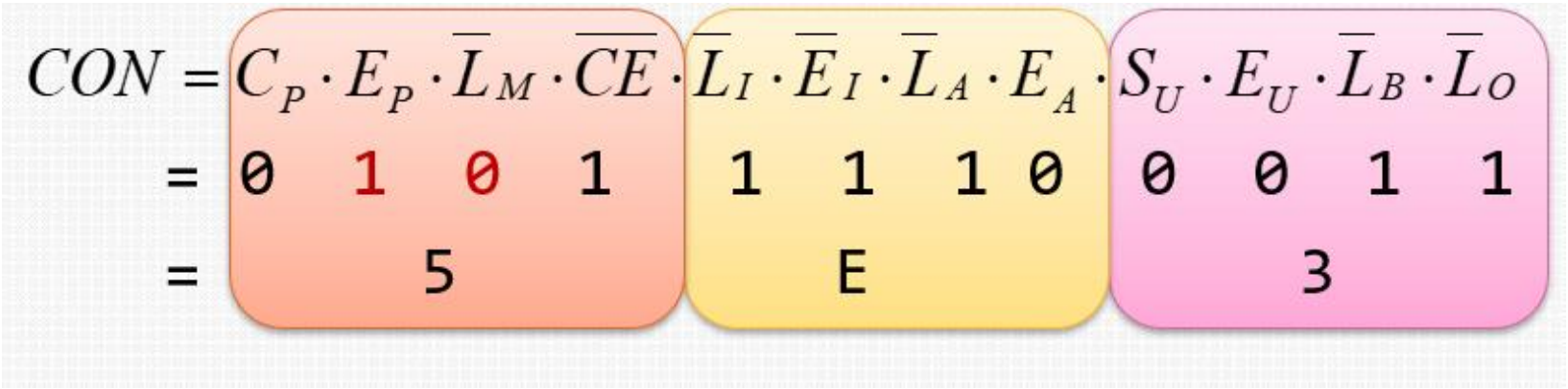
T_1 State:



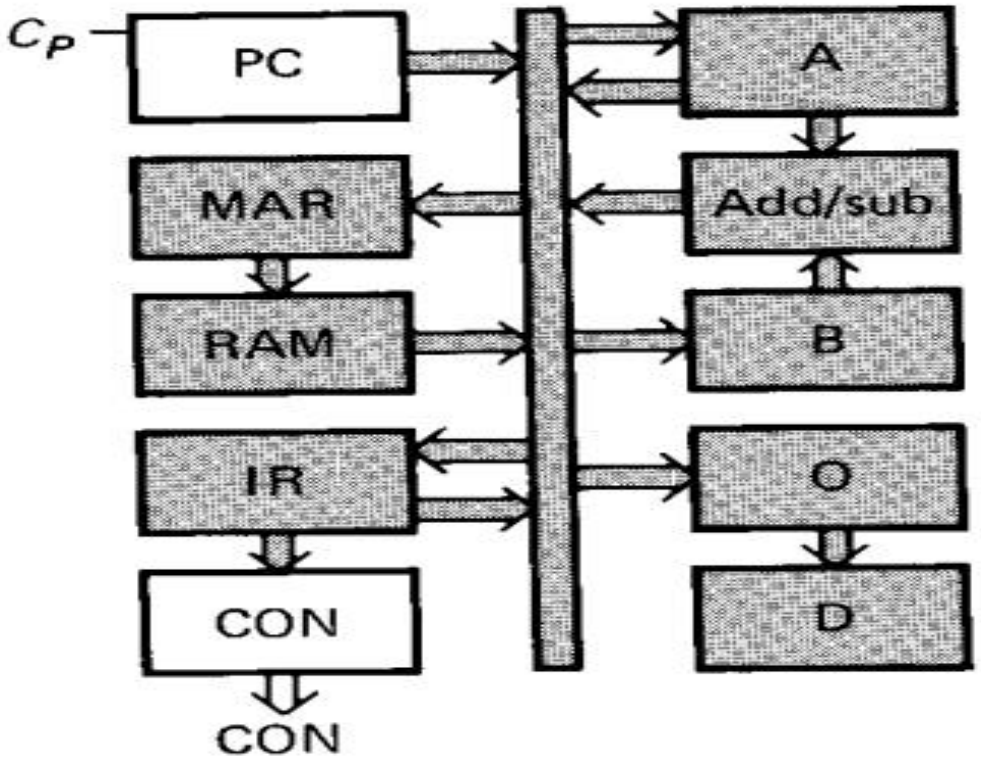
Light→ Active, Dark → Inactive

Address State (T = 000001 = 1 = T_1)

- The T_1 state is called the *address state* because the address in the program counter (PC) is transferred to the memory address register (MAR) during this state.
- During the address state, E_P and L'_M are active; all other control bits are inactive.
- This means that the controller-sequencer is sending out a control word of 5E3H during this state



T_2 State:



Light→ Active, Dark → Inactive

Increment State (T = 000010 = 2 = T_2)

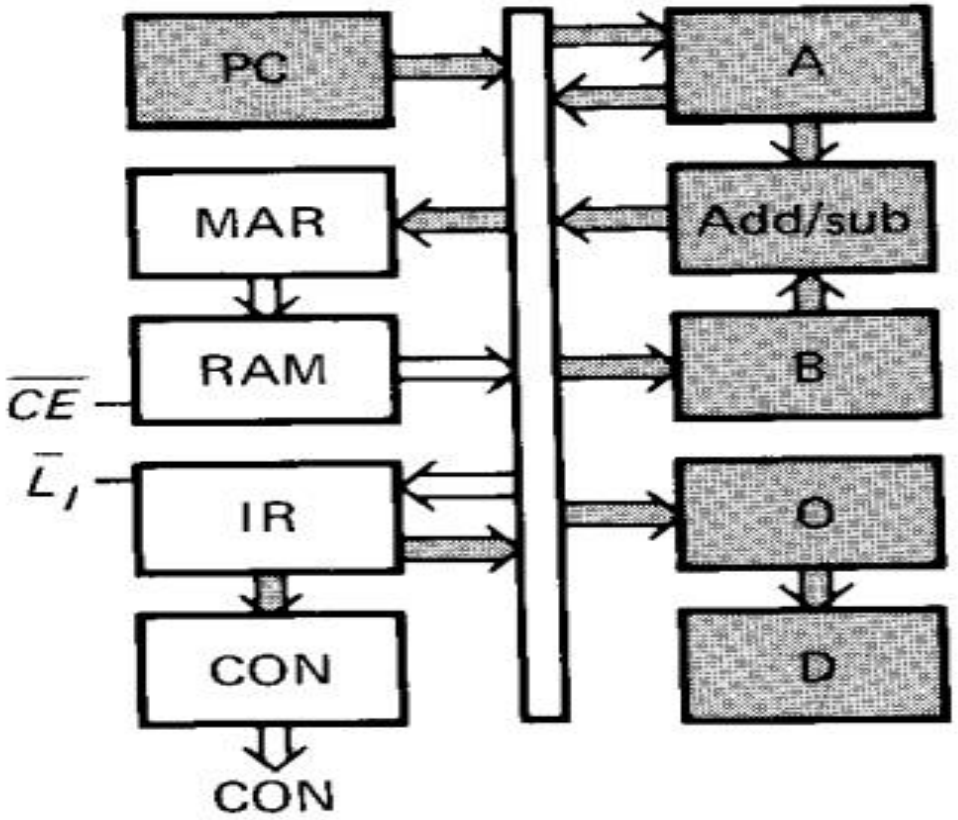
- The T_2 state is called the *increment state* because the program counter is incremented.
- During the increment state, the controller-sequencer is producing a control word of BE3H
- Only the C_P bit is active in this state.

$$CON = C_P \cdot E_P \cdot \overline{L}_M \cdot \overline{CE} \cdot \overline{L}_I \cdot \overline{E}_I \cdot \overline{L}_A \cdot E_A \cdot S_U \cdot E_U \cdot \overline{L}_B \cdot \overline{L}_O$$

$$= \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1$$

$$= \quad \text{B} \qquad\qquad \text{E} \qquad\qquad \text{3}$$

T_3 State:



Light→ Active, Dark → Inactive

Memory State (T = 000100 = 4 = T_3)

- The *T_3* state is called the *memory state* because the addressed RAM instruction is transferred from the memory to the instruction register.
- The only active control bits during this state are *CE'* and *L_I*, and the word out of the controller-sequencer is 263H

$$CON = C_P \cdot E_P \cdot \overline{L}_M \cdot \overline{CE} \cdot \overline{L}_I \cdot \overline{E}_I \cdot \overline{L}_A \cdot E_A \cdot S_U \cdot E_U \cdot \overline{L}_B \cdot \overline{L}_O$$

$$= \quad 0 \quad 0 \quad 1 \quad 0 \qquad 0 \quad 1 \quad 1 \quad 0 \qquad 0 \quad 0 \quad 1 \quad 1$$

$$= \qquad\qquad 2 \qquad\qquad\qquad 6 \qquad\qquad\qquad 3$$

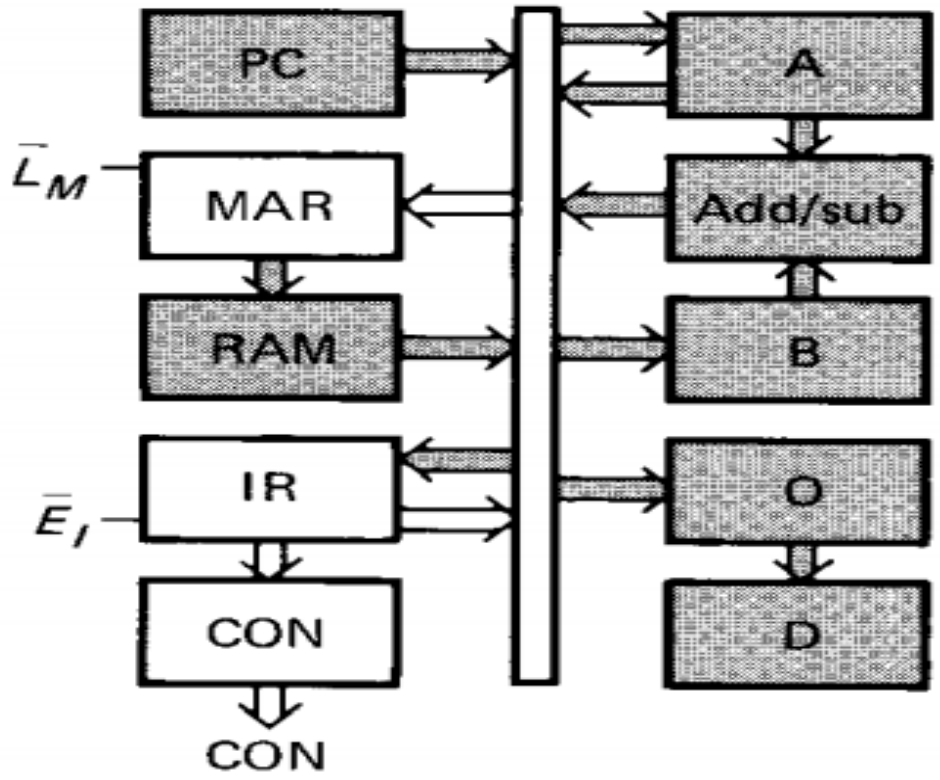
Fetch Cycle:

- The *address*, *increment*, and *memory states* are called the *fetch cycle* of SAP-l.
- During the *address state*, E_P and L_M arc active; this means that the program counter sets up the MAR via the W bus.
- A positive clock edge occurs midway through the address state; this loads the MAR with the contents of the PC.
- During the *increment state*, C_P is the only active control bit.
- This sets up the program counter to count positive clock edges. Halfway through the increment state, a positive clock edge hits the program counter and advances the count by 1.
- During the *memory state*, CE' and L'_I are active. The addressed RAM word sets up the instruction register via the W bus. Midway through the memory state, a positive clock edge loads the instruction register with the addressed RAM word.

Execution Cycle:

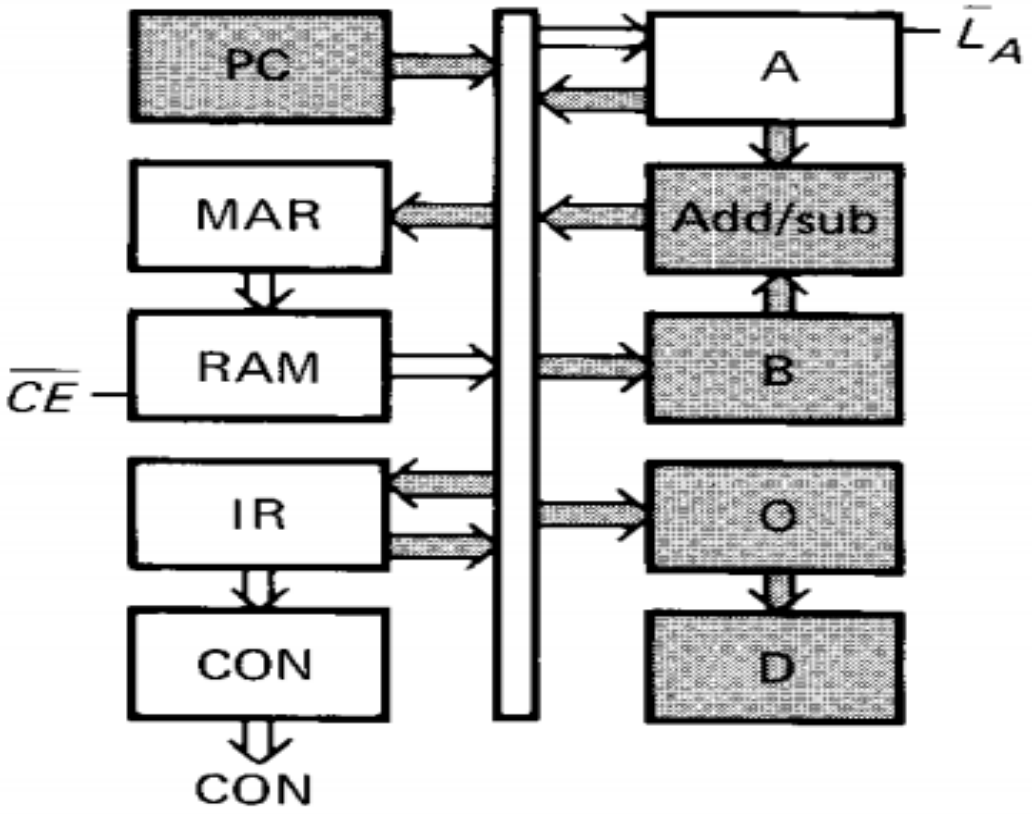
- The next three states (T_4, T_5, and T_6) are the *execution cycle* of SAP-1.
- The register transfers during the execution cycle depend on the particular instruction being executed.
- For instance, LDA 9H requires different register transfers than ADD BH.
- Followings are the *control routines* for different SAP-1 instructions.

T_4 for LDA:



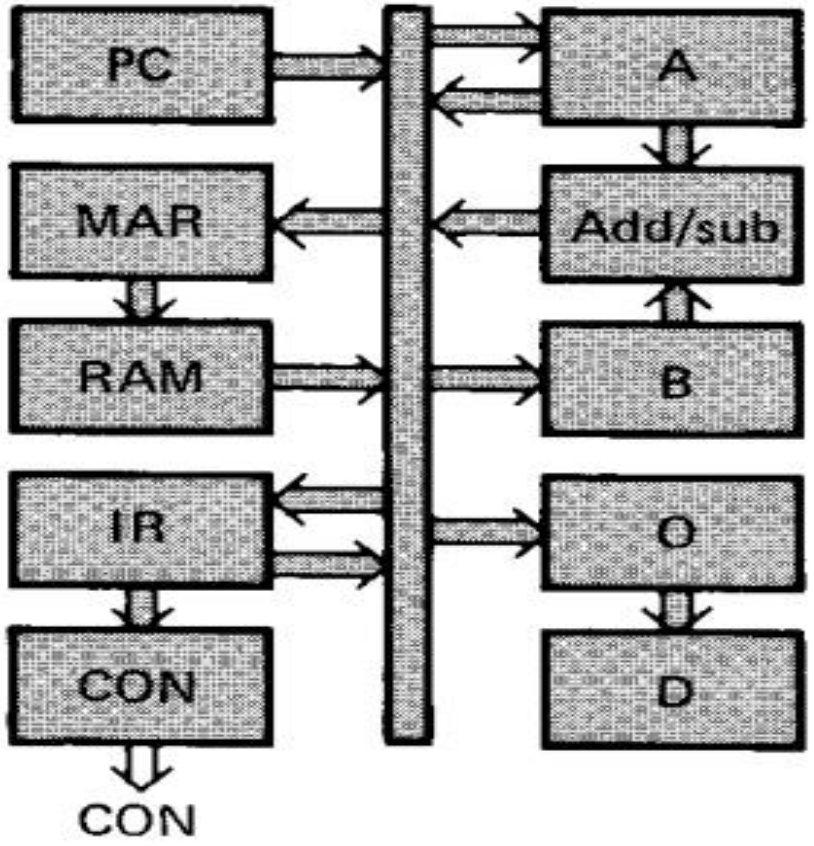
Light→ Active, Dark → Inactive

T_5 for LDA:



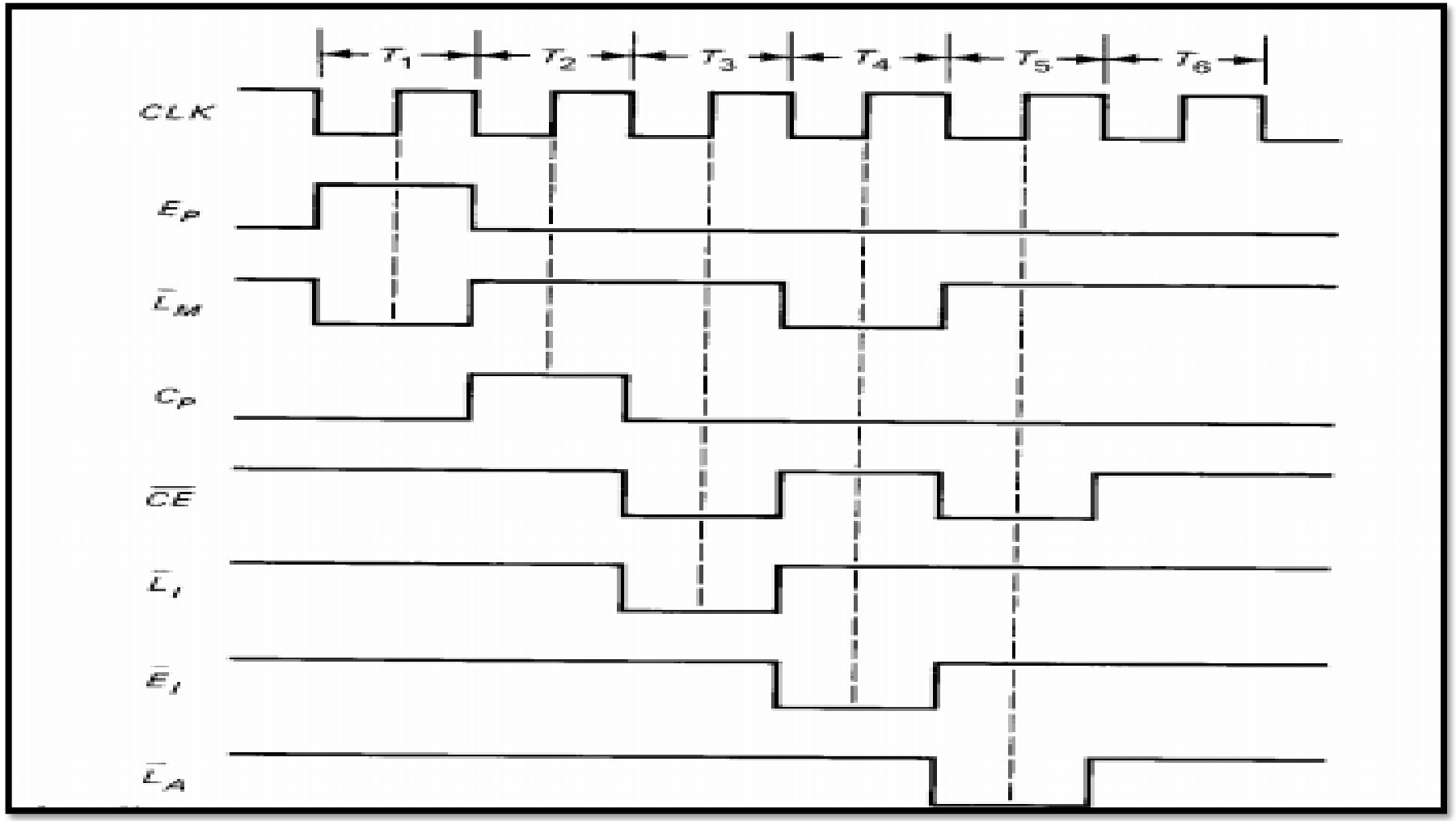
Light→ Active, Dark → Inactive

T_6 for LDA:

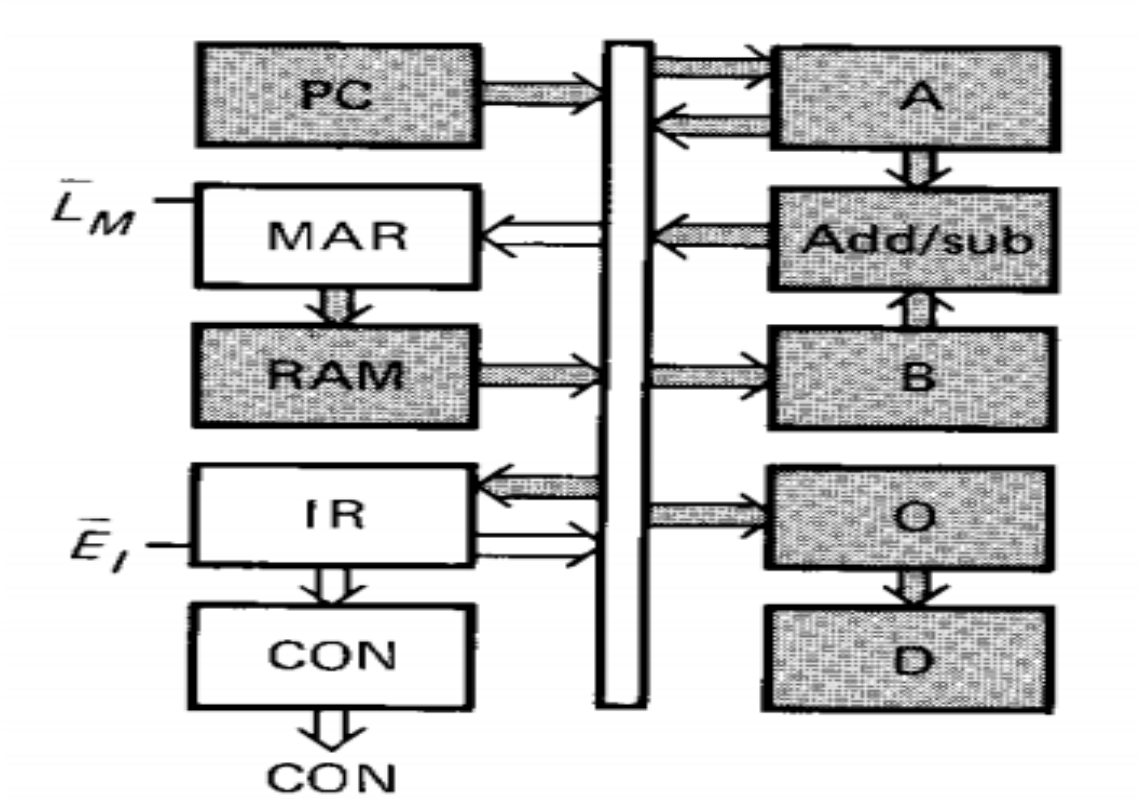


Light→ Active, Dark → Inactive

Timing Diagram for LDA:

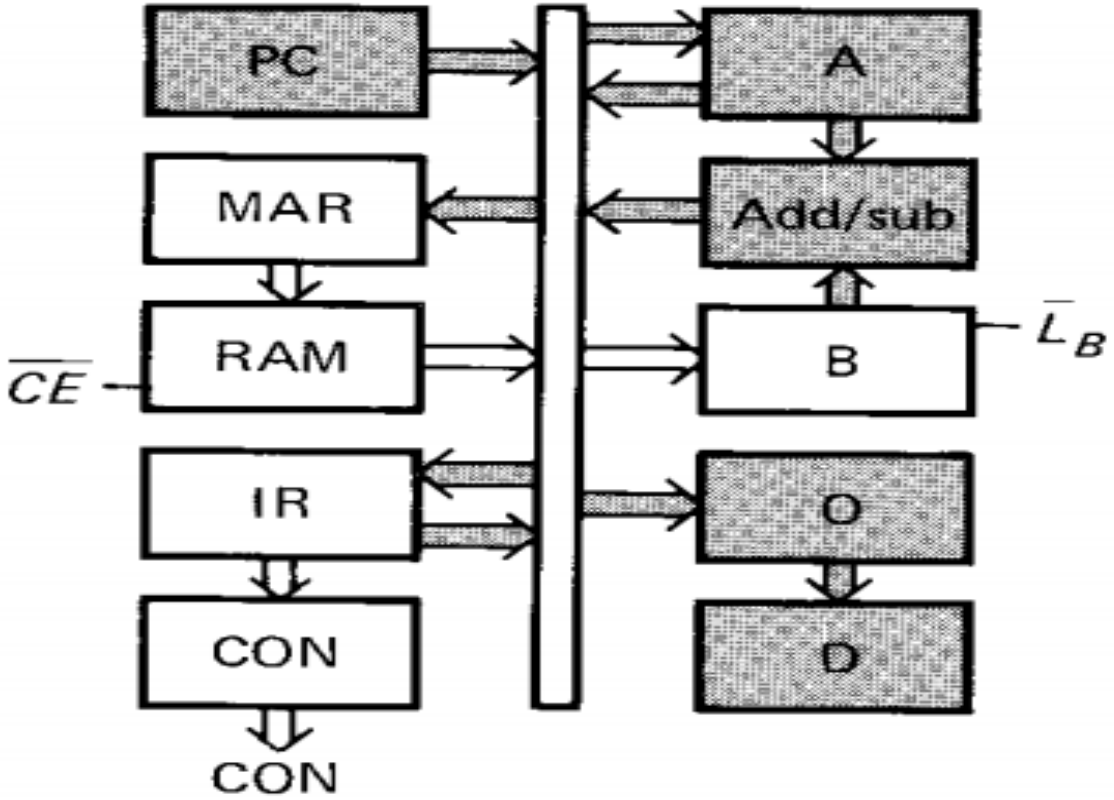


T_4 for SUB and ADD:



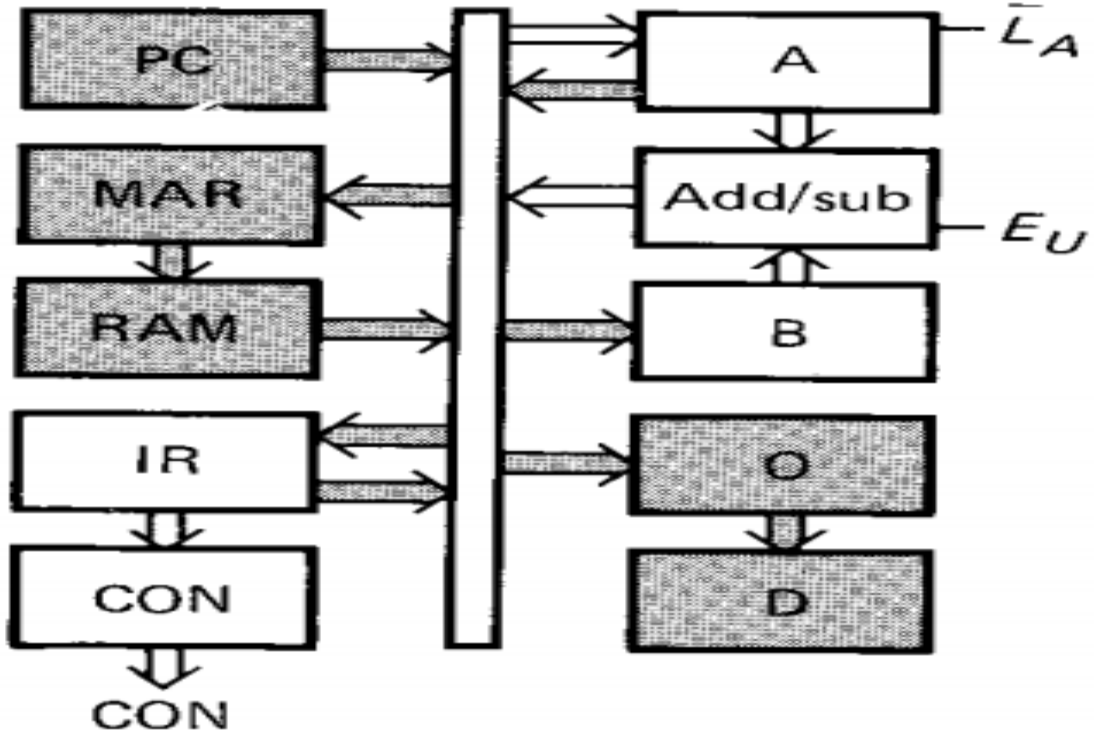
Light→ Active, Dark → Inactive

T_5 for SUB and ADD:



Light→ Active, Dark → Inactive

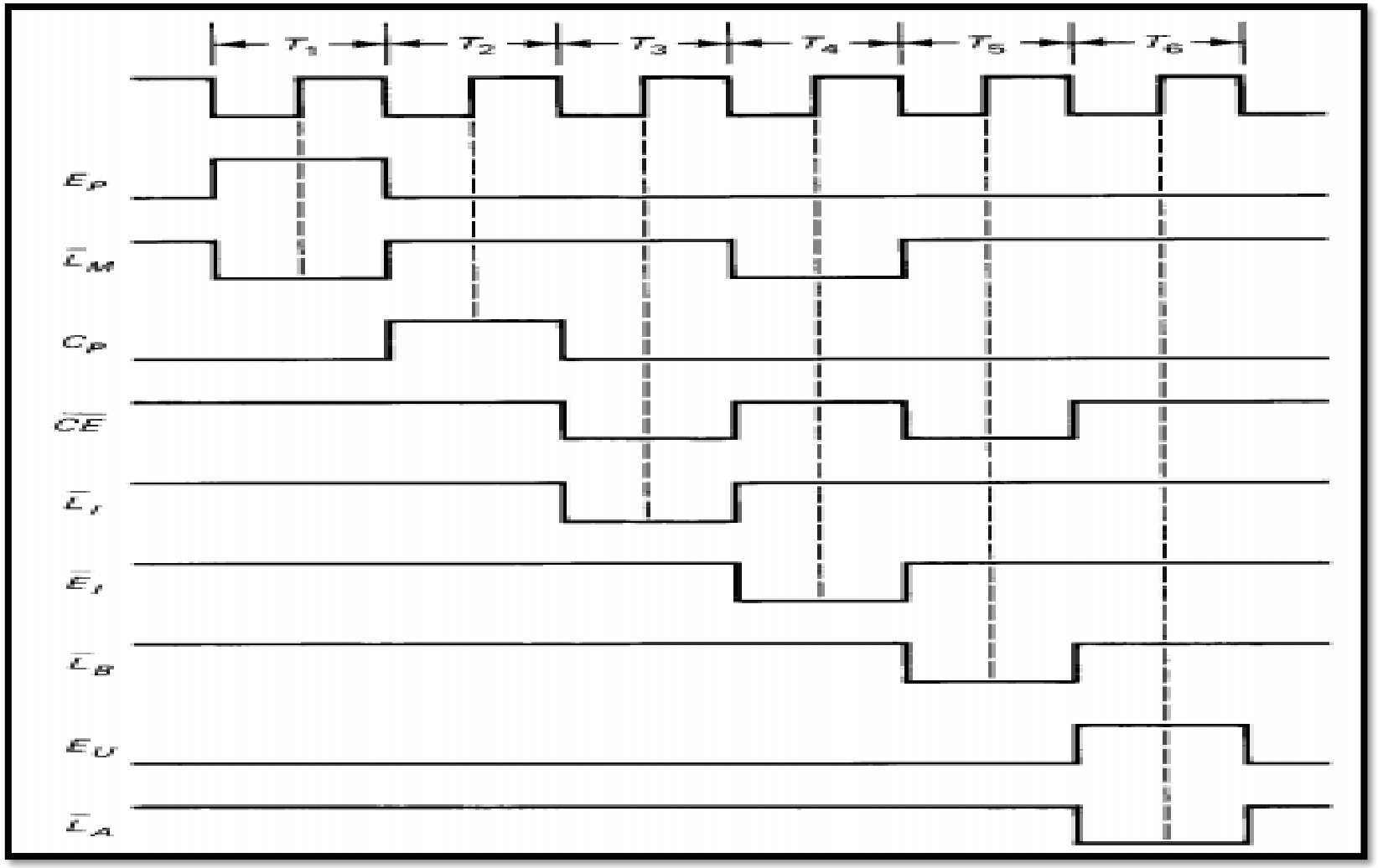
T_6 for SUB and ADD:



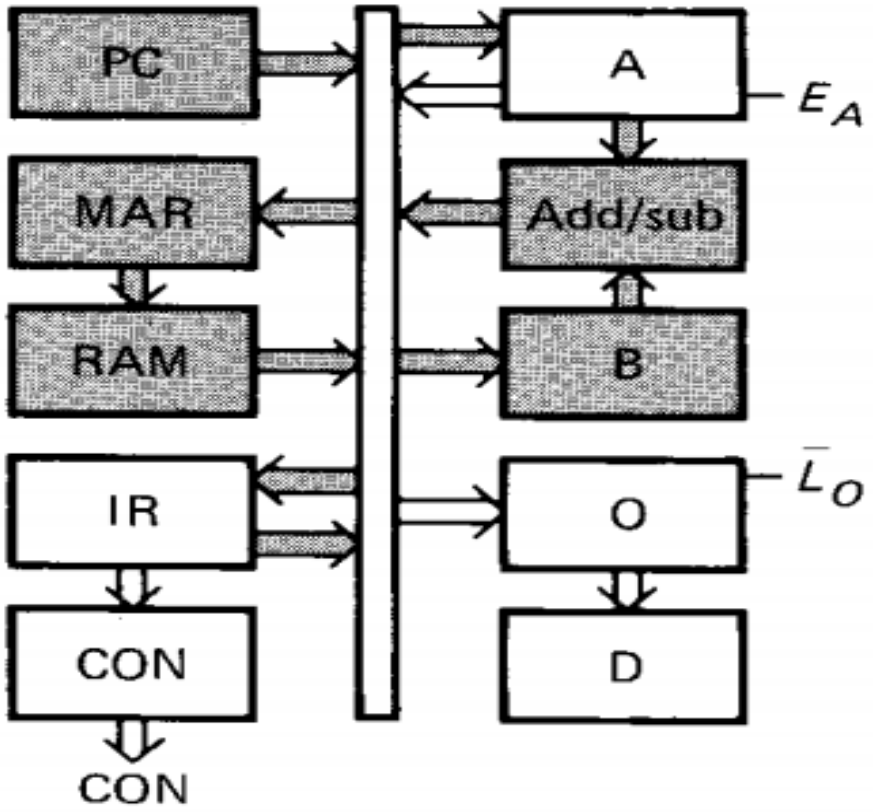
***For Sub, S_u and E_u both must be active

Light→ Active, Dark → Inactive

Timing Diagram for ADD and SUB:



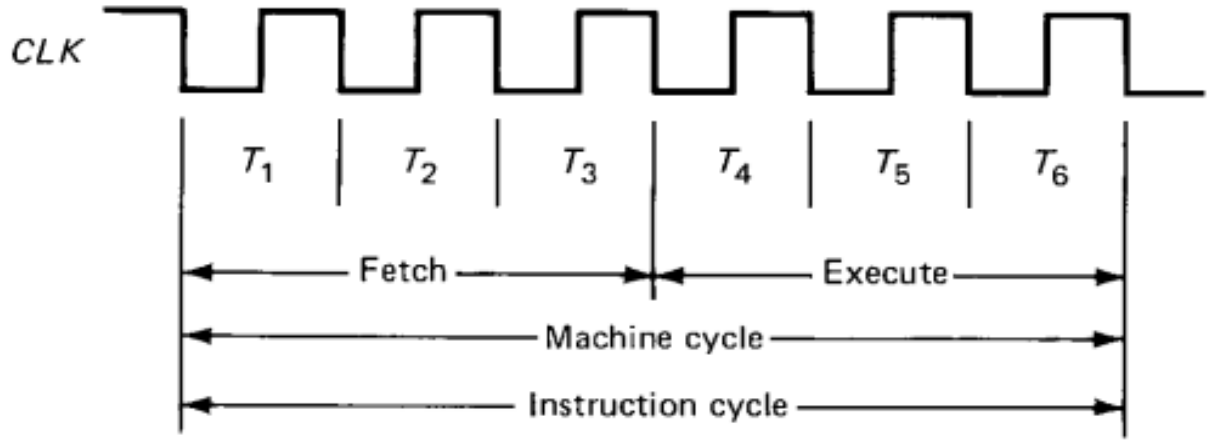
T_4 for OUT:



Light→ Active, Dark → Inactive

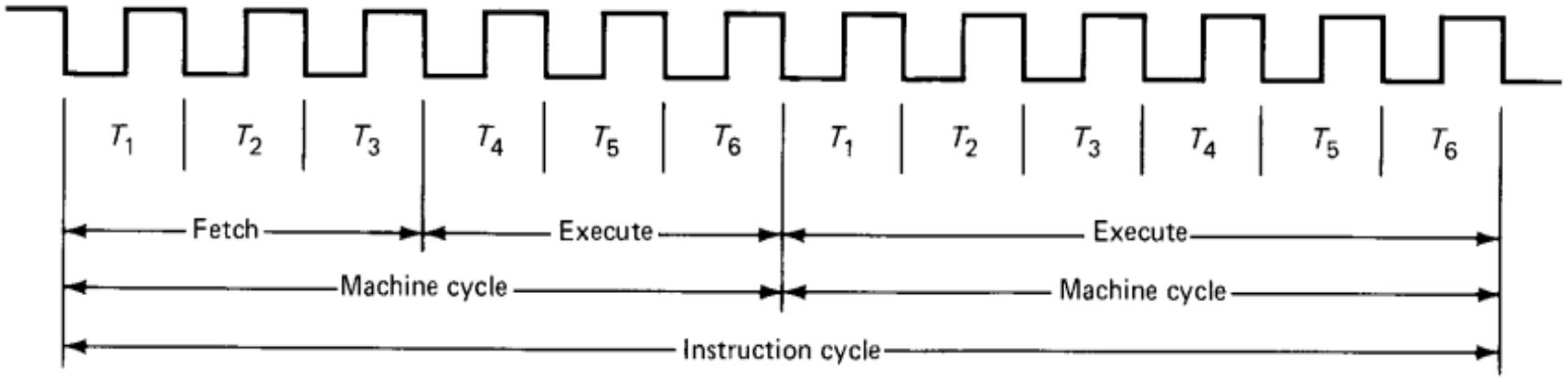
Machine Cycle:

- SAP1 has 6 T-states. These six T-states are called machine cycle.



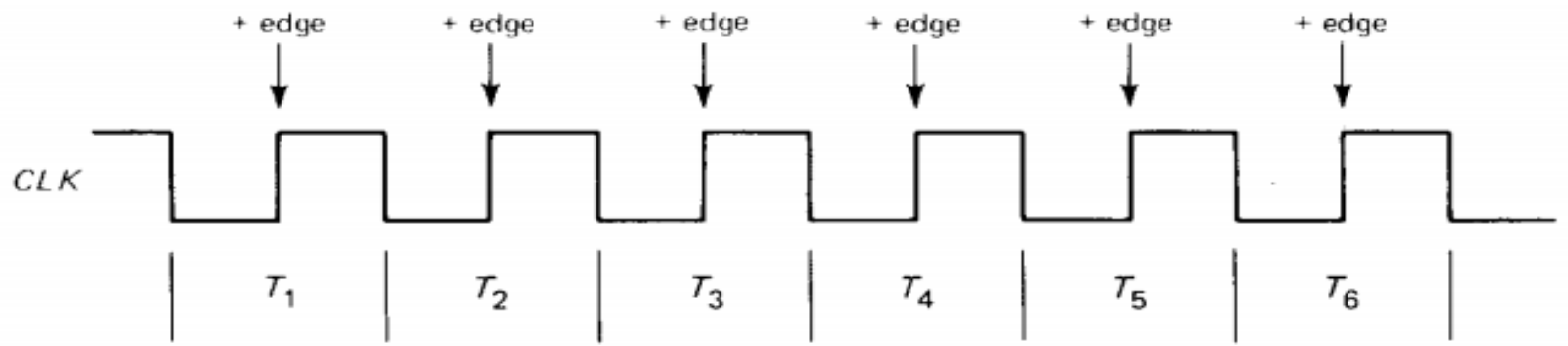
Instruction Cycle:

- The number of T-states needed to fetch and execute an instruction is called instruction cycle.
- In SAP-1 machine cycle = instruction cycle.
- In SAP-2 and other microcomputers the instruction cycle may equal two or more machine cycle.



The positive clock edge occurs halfway through each state. Why this is important?

- Register reliable loading has two conditions: setup time and hold time
- First half: Setup time, Second Half: Hold Time
- To ensure valid data on W-bus (waiting first half)



Micro Instructions:

- The controller-sequencer sends out control words, on during each *T* state or clock cycle.
- These words are like directions telling the rest of the computer what to do.
- Because it produces a small step in the data processing, each control word is called a *micro-instruction*.

Macro Instructions:

- The instructions we have been programming with (LDA, ADD, SUB, . . .) are sometimes called macro-instructions to distinguish them from micro-instructions.
- Each SAP-1 macroinstruction is made up of three microinstructions. For example, the LDA macroinstruction consists of the three microinstructions shown in the next Table.

Macro and Micro Instructions:

This table shows the SAP-1 macro-instruction and the micro-instructions needed to carry it out.

Fetch and Execute Cycle of SAP-1				
Macro Inst.	**T State**	**Micro Operation**	**Active**	**CON**
All Instructions	T_1	MAR ← PC	L'_M, E_P	5E3H
	T_2	PC← PC+1	C_P	BE3H
	T_3	IR ← RAM[MAR]	CE', L'_I	263H
LDA	T_4	MAR ← IR(3…0)	L'_M, E'_I	1A3H
	T_5	ACC ← RAM[MAR]	CE', L'_A	2C3H
	T_6	None	None	3E3H
ADD	T_4	MAR ← IR(3…0)	L'_M, E'_I	1A3H
	T_5	B ← RAM[MAR]	CE', L'_B	2E1H
	T_6	ACC ← ACC+B	L'_A, E_U	3C7H
SUB	T_4	MAR ← IR(3…0)	L'_M, E'_I	1A3H
	T_5	B ← RAM[MAR]	CE', L'_B	2E1H
	T_6	ACC ← ACC – B	L'_A, S_U, E_U	3CFH
OUT	T_4	OUT ← ACC	E_A, L'_O	3F2H
	T_5	None	None	3E3H
	T_6	None	None	3E3H
HLT	T_4	None	HLT '	263H

SAP-1

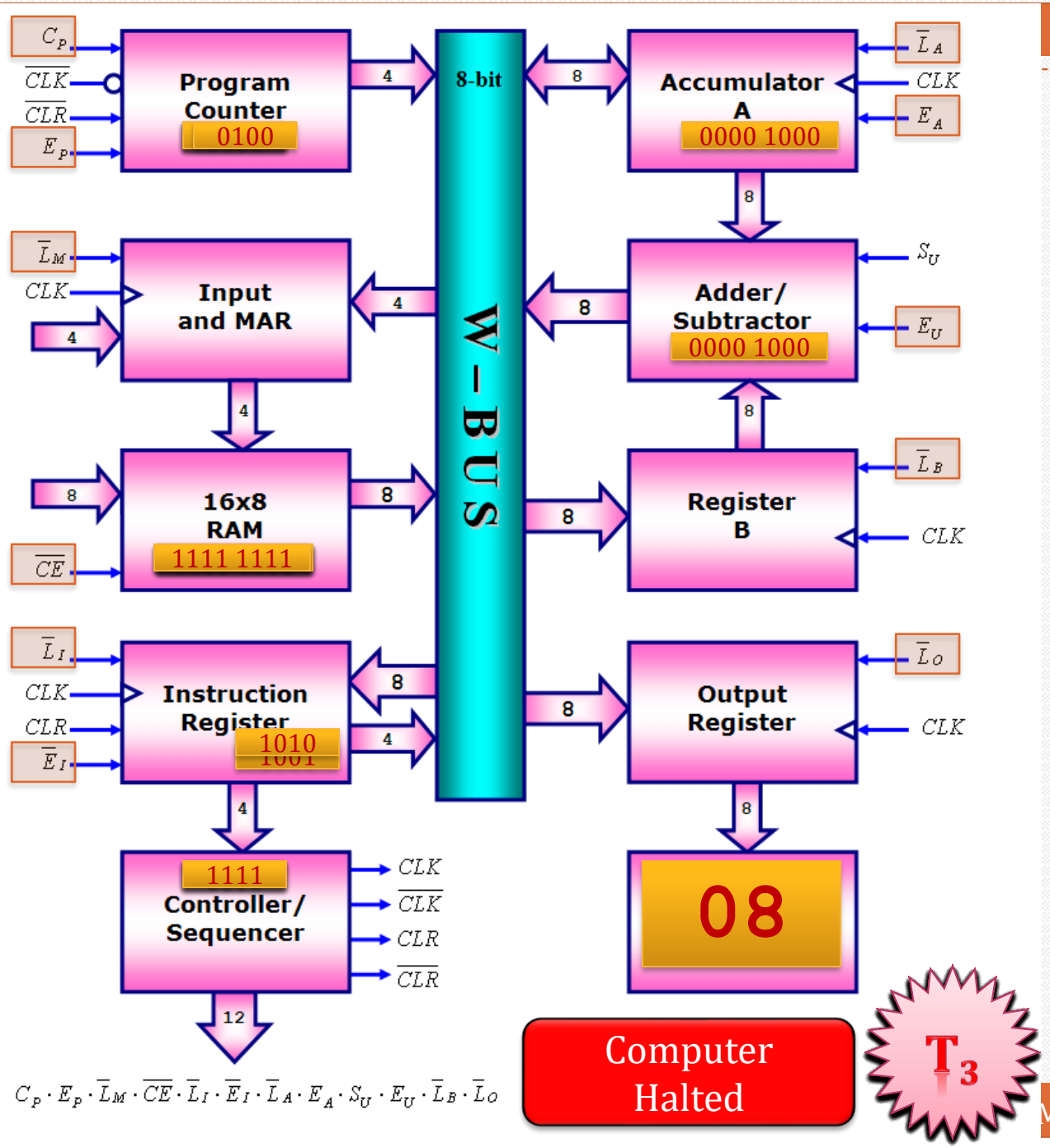
Simulation of Program

LDA AH

ADD AH

OUT

HLT



Acknowledgement

1. Engr. Rashid Farid Chishti

Lecturer, Faculty of Engineering & Technology

International Islamic university Islamabad.

2. Md. Iftekharul Islam Sakib

Lecturer, CSE, BUET.

Best of Luck!