ARCHITECTURE OF 8085 MICROPROCESSOR (8 BIT)

The 8085A is an 8 bit general purpose microprocessor capable of addressing 64K of memory. The main components of the 8085 include the ALU, timing and control unit, Instruction register and decoder, Register array. These are linked by an internal data bus.

The ALU

The arithmetic/logic unit performs the computing functions; it includes the accumulator the temporary register, the arithmetic and logic circuits, and five flags. The temporary register is used to hold the data during an arithmetic/logic operation. The result of the operation is stored in the accumulator and flags are set or reset according to the result of operation. Flags generally reflect data conditions in the accumulator.

Timing and Control Unit

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals.

Instruction Register and Decoder

The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequences of events to follow (what operation is to be performed).

Accumulator

The accumulator (Register A) is an 8 bit register that is part of the arithmetic/logic unit(ALU). This register is accessible to user and holds one of the operands and the result itself.
8085 is accumulator based microprocessor because the operation of 8085 depends on the accumulator. Almost all the arithmetic and logic functions are performed on the data are in the accumulator

Registers B, C, D, E

These four 8 bit registers are accessible to the programmer. These can be used individually as 8 bit individual registers or in pairs as BC and DE as 16 bit registers.

Register H & L

Register H & L can be used as 8 bit registers or as pairs. However, these can also be used for indirect addressing.

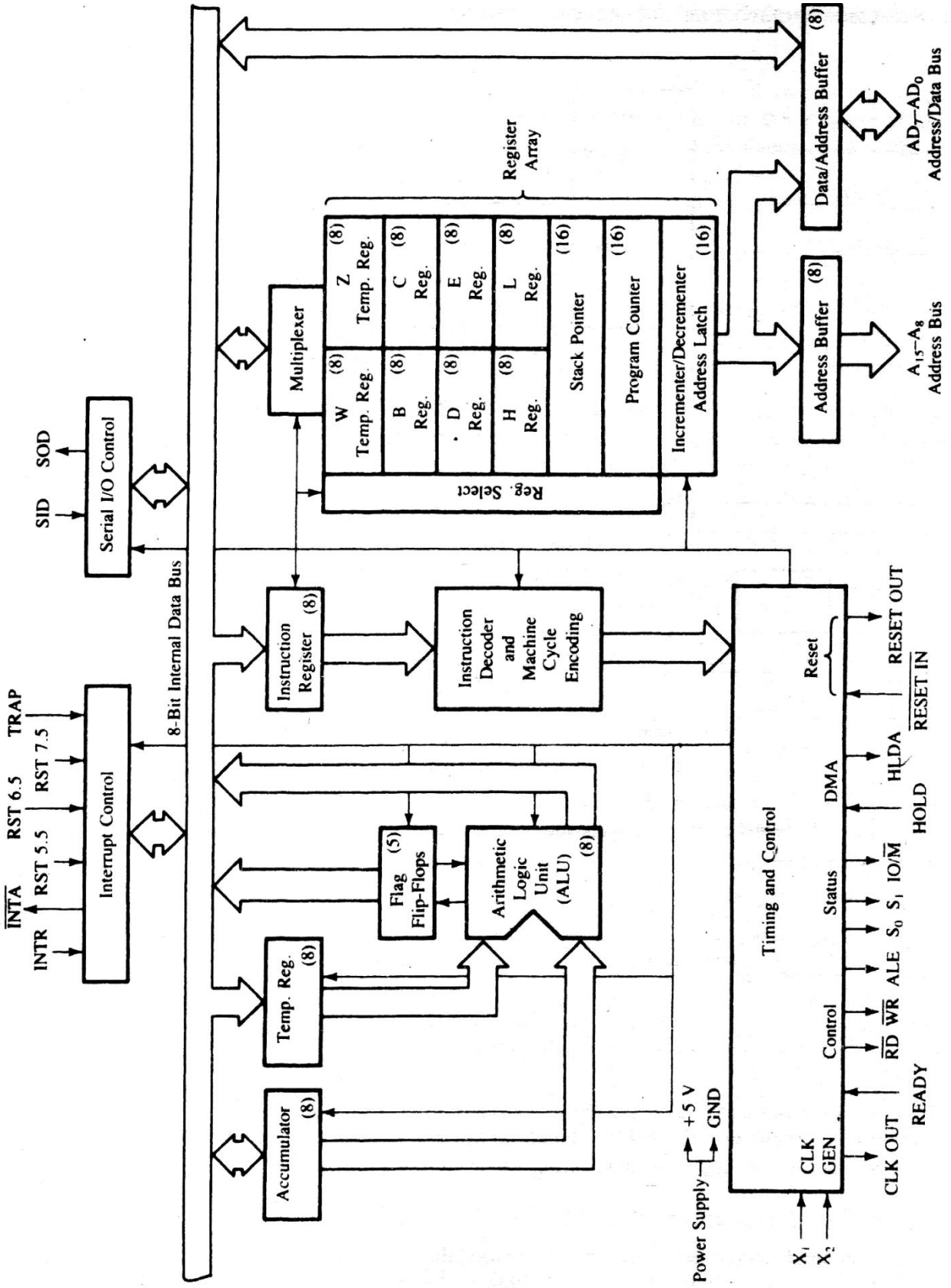

Fig: Functional Block Diagram of 8085A

Program Counter (PC) and Stack Pointer (SP)

These are 16 bit registers used to hold addresses. The size of these registers is 16 bits because the memory addresses are 16 bits.

The computer program consists of the sequence of coded instructions. These are stored in memory. The microprocessor uses the PC register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. PC is incremented by one when the machine code is fetched.

A stack is an area of memory set aside for the purpose of storing data. The location of the memory is defined by the 16 bit register called SP. When a data is stored its value is decreased by 1 while it is increased by 1 when the data is retrieved.

Flags

It consists of a register with five flip flops, each holding a specific status of the output of the operation of ALU. The register is known as Flag Register and each flip flop is called flag. The states of the flags indicate the result of arithmetic and logic operation, which in turn is used for decision making processes. There are no decision instructions for AC flag. The different flags are:

- S – Sign Flag: Is set if the MSB of the result of last operation is 1. (Also called negative, since 1 in the MSB position of two"s complement number)
- Z – Zero Flag: Is set if the ALU operation results in 0, and flag is reset if the result is not 0. Operation of the flag is affected by accumulator as well as that of other registers.
- AC – Auxiliary Carry Flag: If the arithmetic operation generates a carry from lower nibble to upper nibble then this flag is set. It is useful while performing BCD arithmetic.
- P – Parity Flag: Is set if the result has an even numbers of 1s while reset if the number of 1 is odd.
- CY – Carry Flag: Is set if the arithmetic operation results in a carry and is reset if otherwise.

The bit positions reserved for these flags in the flag register are as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

ADDRESSING MODES

Instructions are commands to the microprocessor to perform a certain task. The instructions consist of Op-code and data called operand. The operand may be source only, destination only or both of them.
For example:

MOV A, M

Destination, Source

LDA 3000H

↑

Source

STA 3000H

↑

Destination

In these instructions, the source can be register, input port and a memory location. Similarly, a destination port can be a register and output port or a memory location.The various format of specifying the operand are called addressing modes. The way operands are chosen during program execution is dependent on the addressing mode. Addressing mode specifies the rule for interpreting and modifying the address of the instruction before the operand is actually referenced. The 8085 instruction set has six different addressing modes.

- Direct Addressing
- Register Indirect Addressing
- Register Direct Addressing
- Implied Addressing
- Immediate Addressing

Direct Addressing (Absolute addressing mode):

This addressing mode is called direct because the effective address of the operand is specified directly in the instruction. Instructions using this mode may contain 2 or 3 bytes, with first byte as the Op-code followed by 1 or 2 bytes of address of data. Loading and storing in memory use 2 bytes of address while IN and OUT have one byte address.

For example:

LDA 9089H

This instruction loads the accumulator with the contains of the memory location having a 16 bit address of 9089H. This type of addressing is also called absolute addressing.

IN and OUT instructions are the examples of the instructions which have only 1-byte address.

For example:

IN FCH

Register Indirect Addressing:

In the mode, the address part of the instruction specifies the memory location whose content is the address of the operand. So in this type of addressing mode, it is the address of the address rather than the address itself is part of the instruction. In 8085, wherever the instruction uses HL pointer the address is called indirect addressing.

For example:

MOV A, M

Here, the content of the memory location whose address is specified by the content

of the register H & L is moved to the accumulator. It is called indirect addressing because the register contains the address of the memory location. If register H & L contains 90H, 89H respectively, then the content of the memory location 9089H will be transferred to Register A.

Register Direct Addressing:

Register direct addressing mode means that a register is the source of an operand for an instruction.
For example:

MOV A, B
This instruction moves the contents of the register B to the accumulator. Here, register B contains data rather than the address of the data. So, it is similar to direct addressing.

Implied or Inherent Addressing:

The instructions of this mode do not have operands.
For example:

EI (Enable Interrupt),
STC (set the carry flag),
NOP (No operation)

Immediate Addressing:

This is the simplest form of addressing. The operand is present in instruction in this mode is used to define and use constants of set initial values of variables. In this mode, the instruction will operate on immediate hexadecimal number. The operand may be 8 bit data or 16 bit data.
For example:

MVI B, 05H
This 2- bytes instruction with the 1st byte as op-code, loads Register B with 8 bit data.
LXI B, 5009H
This is 3- bytes instruction with the 1st byte as op-code followed by 2 bytes of data. This instruction loads Register B with 50H and Register C with 09H.

INSTRUCTION WORD SIZE

During the execution of an instruction different actions are exercised. The preliminary action: op-code fetch is same for all the instructions however the latter action depends on the size of the instruction. A microprocessor can handle only a fixed number of instructions in case of 8085 it can handle a maximum of 2^8 i.e. 256 instructions. The different instructions and their equivalent op-codes are available in the 8085 instruction sheet.

Depending on the byte size the instruction set is classified into the following three groups.

- One byte Instructions: A 1byte instruction includes the op-code and the operand in the same byte. The operand will be default for that instruction. These instructions require one memory locations. E.g.: MOV C,A , ADD B, CMA etc.

  Op-code Operand

```
MOV     C, A
ADD     B
CMA
```

- Two byte Instructions: In a 2byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Thus, two memory locations are required. E.g.: MVI A, 32H, ADI F2H etc.

Op-code	Operand
MVI A	32H
ADI	F2H

- Three byte Instruction: In three byte instruction first byte will be op-code where as second and third byte will be the operands. Operands may be data or address depending upon the instruction. E.g.: LXI B, 4455H, LDA 2050H, JMP 2085H etc.

Op-code	Operand
LXI B	4455H
LDA	4050H
JMP	2085H

INSTRUCTION GROUPS OF 8085

An "instruction" is a binary pattern designed to perform a specific function. The list of entire instructions is called the instruction set. The instruction set determines what function the microprocessor can perform. The 8085 instruction set can be classified into the following five categories:

- Data transfer group
- Arithmetic group
- Logical group
- Branching group
- Miscellaneous group

Data transfer group instructions:

The data transfer group is the longest group of instructions in 8085 and consists of 88 different op-codes. This group of instructions copy data from a source location to the destination location, without modifying the contents of the source. The transfer of data may be between the registers or between register and memory or between an I/O device and accumulator. None of these instructions modify the flags.

- Load register (immediate instruction): loads the designated register with the operand.

Some of the instructions are:

MVI A, data	MVI B, data	MVI C, data	MVI D, data
MVI E, data	MVI H, data	MVI L, data	

- Load memory (immediate): loads the memory location whose address is contained in registers H and L with the operand
  Instruction MVI M, data

- Load register pairs (Immediate): loads the register pair with 2 bytes of data. The register pairs are B>B&C, D>D&E, H>H&L. The instructions are:
  LXI B, data LXI D, data LXI H, data

- Load Accumulator (direct): loads the accumulator with the contents of memory location specified by the 2 byte address. The instruction is LDA 2050H.
  Instructions LDA, address

- Load Accumulator (indirect): load the accumulator with the contents of memory location whose address is contained in the designated register pair (only register pair B & D are used). The instructions are LDAX B & LDAX D.
  Instruction LDAX B

- Move register: this instruction loads the first specified register with the contents of the second specified register. The instructions are MOV A, B , MOV C, D.
  Instruction MOV A, B

- Move to memory from designated registers: load the memory location whose address is contained in register H-L with the contents of the register. The instructions are MOV M,A , MOV M,D.
  Instruction MOV M, reg

- Move to designated registers from memory: loads the designated register with the contents of the memory location whose address is specified by the contents of register H&L. The instructions are MOV A,M , MOV D,M.

- Store Accumulator contents (indirect address): loads the memory location whose address is contained in the designated register pair (only register pair B & D are used) with the contents of the accumulator. The instructions are STAX B, STAX D.

- Store Accumulator Contents (direct): loads the memory location whose address is specified by the operand. The instruction is STA operand.

- Exchange contents of H&L with D&E: This instruction is used to exchange the contents of register H with the contents of register D and the contents of register L with the contents of register E. The instruction is XCHG.

- Load H & L directly: This instruction loads the registers H&L with the contents of the memory location whose address is the operand of the instruction. (Register L contains the value at the address and Register H contains the value at the address+1)

- Store H&L directly: This instruction is similar to load instruction and stores the contents of H & L at the designated memory address.

Arithmetic group instructions:

There are different instructions for arithmetic operations in 8085. The arithmetic operations performed by 8085 are addition, subtraction, increment and decrement. All the addition and subtraction operations are performed with accumulator. However, increment and decrement can be performed in any register and affect the flags except carry flag.

For addition and subtraction the following features exist:

- The instructions assume implicitly that the accumulator is one of the operands.
- The flags get modified according to the data conditions of the result.
- The result is placed in the accumulator.
- The execution of the instruction does not affect the operand register.

The different instructions are:

ADD register: Add the contents of a register with the accumulator
ADI data: Add immediate 8 bit data with the accumulator
ADC register: Add the contents of a register with the accumulator and the
carry ADI data: Add immediate 8 bit data with the accumulator and the
carry SUB register: Subtract the contents of register from accumulator
SBB register: Subtract the contents of register and borrow from accumulator
SUI data: Subtract the immediate data from the accumulator
SBI data: Subtract the immediate data and borrow from accumulator
INR register: Increment the contents of register by 1
DCR register: Decreases the contents of register by 1

Logical group instructions:

A microprocessor is programmable logic device. All the operations of the microprocessor is carried out via the hardwired logic circuits. The logical functions that 8085 microprocessor can perform are AND, OR, Ex-OR and NOT. The different instructions are:

ANA register: Logical AND the contents of register with
accumulator ANI data: Logical AND the immediate data with
accumulator ORA register: Logical OR the contents of register with
accumulator ORI data: Logical OR the immediate data with
accumulator
XRA register: Logical exclusive OR the contents of register with accumulator
XRI data: Logical exclusive OR the data with accumulator
CMA: Complements the contents of accumulator

The following features hold true for all logic instructions:

- The accumulator is one of the operands
- Except the instruction CMA, all instruction reset carry flag
- They modify the Z, P, A, S flags according to the data conditions of result
- They place the result in accumulator
- The contents of operand registers are not affected.

Branching group instructions:

The branch instructions allow the microprocessor to change the sequence of a program,

either on certain conditions or unconditionally. The microprocessor is a sequential machine i.e. the instructions are executed from memory location sequentially. Branch instructions cause the microprocessor to execute from another memory locations. The branch instructions can be categorized as:

- Jump Instruction
- Call and Return Instruction
- Restart Instruction

Jump Instruction:

The jump instructions specify the memory location explicitly. These are 3 byte instructions where the first byte is op-code and other bytes represent the 16 bit address. The jump may be conditional or unconditional.

An unconditional jump enables the programmer to set up continuous loops without depending on any type of conditions. The instruction is JMP data. The jump location can also be specified using a label if the exact memory location which a program sequence should be directed, is not known. However, it should be considered that a single label cannot be used for different memory locations.

A conditional jump instruction allows the microprocessor to make decision based on certain conditions indicated by the flags. After a arithmetic or logical operation, flags are set or reset to reflect the data conditions. These instructions check flag conditions and make decision to change the sequence of program. The flags tested for conditional jump are zero, carry, sign and parity. The different instructions are:

JC: Jump on carry
JNC: Jump on if no carry
JZ: Jump on zero
JNZ: Jump on no zero
JP: Jump on plus
JM: Jump on minus
JPE: Jump on even parity
JPO: Jump on odd parity

Call and return Instructions:

Call and return instructions are used for using subroutine. A subroutine is a group of instruction that performs a subtask of repeated occurrence. The subroutine is written as a separate unit, apart from the main program and the control from program is transferred to subroutine when the call instruction is encountered. After the completion of the subroutine the control is returned back to the main program.

The call instruction and return instructions can also be set conditionally or unconditionally. The different call and return instructions are:

CALL: Unconditional call instruction
RET: Unconditional return instruction
CC, CNC, CZ, CNZ, CM, CP, CPE, CPO: Conditional Call
RC, RNC, RZ, RNZ, RM, RP, RPE, RPO: Conditional Return

Restart Instruction:

The 8085 instruction set includes eight RST (Restart instructions). These are 1 byte call instructions and transfer the program execution to a specific location as listed in table below:

Restart Instructions	Hex. Code	Call Location in Hex
RST0	C7	0000H
RST1	CF	0008H
RST2	D7	0010H
RST3	DF	0018H
RST4	E7	0020H
RST5	EF	0028H
RST6	F7	0030H
RST7	FF	0038H

Actually these restart instructions are inserted through additional hardware and for this purpose the INTA signal of 8085A is used. These instructions are part of interrupt process.

Miscellaneous group instruction:

All other instructions in the instruction set fall on this group. These instructions include stack operation instructions, input/output operations and interrupt operations of 8085 microprocessor. The different instructions are.

PUSH, POP, DI, EI, HLT, etc….

UNIT-3: INSTRUCTION CYCLE

RESISTER TRANSFER LANGUAGE (RTL)

During the execution of an instruction different actions are exercised. These operations are expressed using a language. Such type of language, which is basically used to express the transfer of data among the registers, is called Register Transfer Language (RTL). If a data is transferred from register B to register A the in RTL

register A ← register B

During the execution of an instruction we have fetch cycle and execute cycle. The operations performed during fetch cycle are:

- The PC contains the address of the next instruction to be executed. As first operation of fetch cycle, the contents of program counter will be transferred to the memory address register (MAR). The memory address register then uses the address bus to transmit its contents that specifies the address of the memory location from where the instruction code is to be fetched. Let t1 be period of this operation.

  MAR ← PC
- Now as soon as the control unit issues the memory read signal, the contents of the addressed memory location specified by MAR will be transferred to the memory buffer register (MBR). Let t2 be the period of this operation.

  MBR ← Memory

- Now the contents of MBR will be transferred to the instruction register and the program counter (PC) gets incremented to fetch another instruction and the fetching cycle is thus complete. Here two operations take place within a single time unit t3. Let t3 be the time required by the CPU for this operation.

  IR ← (MBR)
  PC ← PC+1

The control unit of the computer maintains the timing sequence of the all of the above operations that constitute the fetch cycle. The operations are sequenced on the basis of the single time period called the period of the clock. The RTL for fetch cycle is:

t1: MAR ← PC
t2: MBR ← Memory
t3: IR ← (MBR)
PC ← PC +1

All three time units are of equal duration. A time unit is defined by a regularly spaced clock pulses. The operations performed within this single unit of time are called micro-operations. Since each micro-operation specifies the transfer of data into or out of a register, such type of language is called Register Transfer Language.

After the fetch of the instruction is complete, the execution cycle starts. Different instructions are executed in different fashions. Let us consider execution of a simple instruction MOV A,B. The first step needed is to obtain the data from the location B. For this the address field of instruction register indicating the address of memory location will

be transferred to address bus through the Memory Address Register (MAR).

When the control unit issues a memory read signal, the contents of location B will be output to the MBR. Now the address of memory location A is loaded in MAR. Finally, after the memory write signal from the control unit is issued the data is copied to memory location A. Using RTL language and defining above operations in the form of micro operation the execute cycle is represented in following time units.

t1: MAR ← (IR (Address of B))
t2: MBR ← (B)
t3: MAR ← (IR (Address of A))
t4: A ← MBR

INSTRUCTION AND MACHINE CYCLE

Consider a simple instruction: MOV A, B which states that data from register B is to copied to register A. The instructions are all stored in the memory. The computer spends certain period of time on Fetching, Decoding and Executing this instruction.

Instruction Cycle: It is defined as the time required to complete execution of an instruction.

Machine cycle: It is defined as the time required to complete one operation of accessing memory, I/O, or acknowledging an external request.

T-state: It is defined as one subdivision of the operation performed in one clock period.

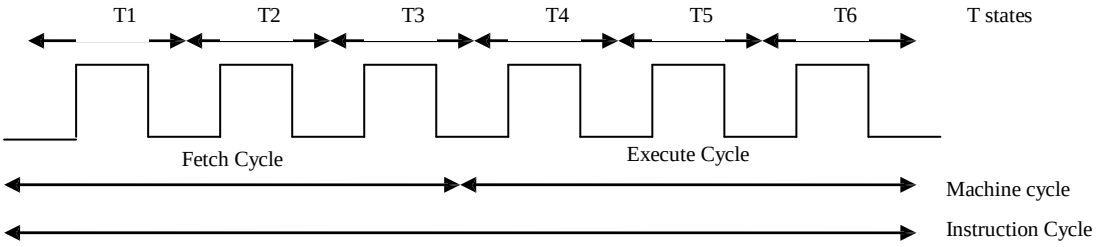

Fig Instruction cycle

APPENDIX: RTL

Instruction	MVI A, data
RTL (execute cycle)	t1: MAR ← PC
	t2: MBR ← (MAR) , PC++
	t3: register A ← MBR

Instruction	MVI M, data
RTL	t1: MAR ← PC
	t2: MBR ← (MAR), PC++
	t3: MAR ← registers H&L
	t4: (MAR) ← MBR

Instruction	LXI B, data
RTL	t1: MAR ← PC
	t2: MBR ← (MAR), PC++
	t3: register C ← MBR
	t4: MAR ← PC
	t5: MBR ← (MAR), PC++
	t6: register B ← MBR

Instructions	LDA, address
RTL	t1: MAR ← PC
	t2: MBR ← (MAR++ MAR)
	t3: MAR ← MBR
	t4: MBR ← (MAR)
	t5: register A ← MBR

Instruction	LDAX B
RTL	t1: MBR ← register B & register C
	t2: MAR ← MBR
	t3: MBR ← (MAR)
	t4: register A ← MBR

Instruction	MOV A, B
RTL	t1: MAR ← (Address of B)
	t2: MBR ← (B)
	t3: MAR ← (Address of A)
	t4: A ← MBR