UNIT-II

Page No.:
Date :

Assembly Language Programming

8085 Instruction set :-

An instruction is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions called the Instruction set, determines what function the microprocessor can perform.

The 8085 instructions can be classified into the following five functional categories :- data transfer (copy) operations, arithmetic operations, logical operations, branching operations and machine control operations.

Instruction classifications :-

(1) Data Transfer (Copy) Operations :-

This group of instructions copies data from a location called a source to another location called a destination, without modifying the contents of the source. In technical manuals, the term data transfer is used for this copying function.

The various types of data transfer (copy) are listed below together with examples of each type :

Types	Examples
▫ Between Registers	Copy the contents of register B into register D.
▫ Specific data byte to a register or a memory location	Load register B with the data byte 32H.
▫ Between a memory location and a register	from the memory location 2000H to register B.
▫ Between an I/O device and the accumulator	from an input keyboard to the accumulator.

(2) Arithematic Operations :- These instructions perform arithematic operations such as addition, subtraction, increment and decrement.

▫ Addition :- Any 8-bit number or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (for ex:- the contents of register B cannot be added directly to the contents of register C). The

Instruction DAD is an exception; it adds 16 bit data directly in register pairs

- **Subtraction:-** Any 8-bit number or the the contents of a register or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's complement and the results, if negative are expressed in 2's compl No two other registers can be subtracted directly.

- **Increment/Decrement:-** The 8-bit contents of a register or a memory location can be incremented or decremented by 1. Similarly, the 16 bit contents of a register pair (such as BC) can be incremented or decremented by 1. These incremented and decremented operations differ from addition and subtraction in an important way i.e. they can be performed in any one of the registers or in a memory location.

(B) Logical Operations:- These instructions perform various logical operations with the contents of the accumulator.

- **AND, OR, EX-OR:-** Any 8-bit number or the

Date :

contents of a register or of a memory location can be logically ANDed, ORed or EX-ORed with the contents of the accumulator. The results are stored in the accumulator.

- **Rotate :-** Each bit in the accumulator can be shifted either left or right to the next position.

- **Compare :-** Any 8-bit number or the contents of a register or a memory location can be compared for equality, greater than or less than with the contents of the accumulator.

- **Complement :-** The contents of the accumulator can be complemented all 0's are replaced by 1's and all 1's are replaced by 0's.

(4) Branching Operations :-

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

- **Jumps :-** Conditional jumps are an importa

aspect of the decision-making process in programming. These instructions test for a certain condition (ex- zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called unconditional jump.

- **Call, Return and Restart:-** These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The Conditional Call and Return instructions also can test conditional flags.

(5) **Machine Control Operations:** These instructions control machine functions such as Halt, Interrupt or do nothing.

ADDRESSING MODES IN 8085 MICROPROCESSOR:-

The way of specifying data to be operated by an instruction is called addressing mode.

Types of Addressing Modes:-

There are 5 types of addressing modes:-

(1) Immediate Addressing Mode:-

In immediate addressing mode the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes.

Example:-

MVI B, 45 (move the data 45H immediately to register B)

LXI H, 3050 (load the H-L pair with the operand 3050H immediately)

JMP address (jump to the operand address immediately)

(2) Register Addressing Mode:-

In register addressing mode, the data to be operated is available inside the registers and registers are operand. Therefore the operation is performed within various registers of the microprocessor.

Page No.:
Date :

Examples:-
MOV A, B (move the contents of register B to register A)
ADD B (add contents of registers A and B and store the result in register A)
INR A (increment the contents of register A by one)

(3) Direct Addressing Mode:- In direct addressing mode, the data to be operated is available inside a memory location and that memory location is directly specified as an operand. The operand is directly available in the instruction itself.

Examples:-
LDA 2050 (load the contents of memory location into accumulator A)
LHLD address (load contents of 16-bit memory location into H-L register pair)
IN 35 (read the data from port whose address is 35)

(4) Register Indirect Addressing Mode:- In register indirect addressing mode, the data to be operated is available inside a memory location and

that memory location is indirectly specified by a register pair

Examples:-

MOV A, M (Move the contents of the memory location pointed by the H-L pair to the accumulator)

LDAX B (move contents of B-C register to the accumulator)

STAX B (store accumulator contents in memory pointed by register pair B-C)

5. Implied / Implicit Addressing Mode:-

In implied / implicit addressing mode the operand is hidden and the data to be operated is available in the instruction itself.

Examples:-

CMA (finds and stores the 1's complement of the contents of accumulator A in A)

RRC (rotate accumulator A right by one bit)

RLC (rotate accumulator A left by one bit)

Page No.:
Date:

STACK AND SUBROUTINES :-

STACK:- The stack in an 8085 microcomputer system can be described as a set of memory locations in the R/W memory, specified by a programmer in a main program. These memory locations are used to store binary information (bytes) temporarily during the execution of a program.

The beginning of the stack is defined in the program by using the instruction **LXI SP,** which loads a 16-bit memory address in the stack pointer register of the microprocessor. Once the stack location is defined, storing of data bytes begins at the memory ~~location~~ address that is one less than the address in the stack pointer register.

The size of the stack is limited only by the available memory.

★ Data bytes in the register pairs of the microprocessor can be stored on the stack (two at a time) in reverse order (decreasing memory address) by using the instruction **PUSH**.

★ Data bytes can be transferred from the stack to respective registers by using the instruction **POP**.

The stack is shared by the programmer and the microprocessor. The programmer can store and retrieve the contents of a register pair by using PUSH & POP instructions. Similarly, the microprocessor automatically stores the contents of the program counter when a Subroutine is called.

INSTRUCTIONS :-

Opcode	Operand	
LXI	SP, 16-bit	↳ Load stack pointer ↳ Load the stack pointer register with a 16-bit address
PUSH	Rp	Store Register Pair on stack ↳ This is a 1-byte instruction
POP	Rp	Retrieve Register Pair from stack ↳ This is a 1-byte instruction

SUBROUTINE :-

A subroutine is a group of instructions written separately from the main program to perform a function that occurs repeatedly in the main program. For example, if a time delay is

required between three successive events, three delays can be written in the main program. To avoid repetition of the same delay instruction, the Subroutine technique is used.

The 8085 microprocessor has Two instructions to Implement subroutines: CALL (call a subroutine) and RET (return to main program from a subroutine)

CALL:- The CALL instruction is used in the main program to call a subroutine.

RET:- The RET instruction is used at the end of the subroutine to return to the main program.

When a subroutine is called the contents of the program counter, which is the address of the instruction following the CALL instruction, is stored on the stack and the program execution is transferred to the subroutine address. When a RET instruction is executed at the end of the subroutine, the memory address stored on the stack is retrived and the sequence of execution is resumed in the main program.

Page No.:
Date :

INSTRUCTIONS :-

Opcode	Operand	
CALL	16 bit memory address of a subroutine	Call Subroutine Unconditionally ↳ That is a 3-byte instruction that transfers the program sequence to a subroutine address. ↳ Saves the contents of the program counter on the stack. ↳ Decrements the stack pointer register by two ↳ Jump unconditionally to the memory location specified by the second and third bytes. The second byte specifies a line number and the third byte specifies a page number. ↳ This instruction is accompained by a return instruction in the subroutine.
RET		Return from Subroutine unconditionally

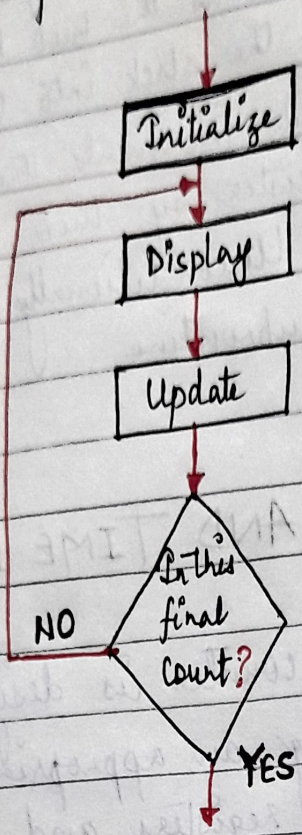
↳ This is a 1-byte instruction
↳ Inserts the two bytes from the top of the stack into the program counter and increments the stack pointer register by two
↳ Unconditionally returns from a subroutine.

COUNTERS AND TIME DELAYS :

Counter :- A counter is designed simply by loading an appropriate number into one of the registers and using the INR (Increment by one) or the DCR (Decrement by one) instructions. A loop is established to update the count and each count is checked to determine whether it has reached the final number, if not, the loop is repeated.

The flowchart, shown in figure. However, this counter has on major drawback, the counting is performed at such high speed that only the last count can be observed. To observe counting, there must be an appropriate time delay between counts.

figure:- flowchart of a Counter

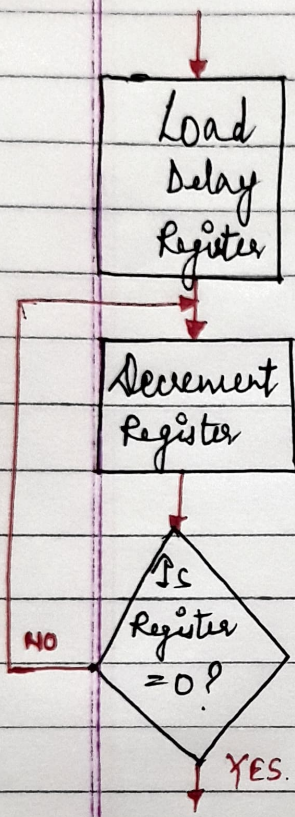


Time Delay :- The procedure used to design a specific delay is similar to that used to set up a counter. A register is loaded with a number, depending on the time delay required and then the register is decremented until it reaches zero by setting up a loop with a conditional jump instruction. The loop causes the delay, depending upon the clock period of the system.

* Time Delay Using One Register :-

The flowchart

in figure shows a Time-delay loop. A count is loaded in a register and the loop is executed until the count reaches zero. The set of instructions necessary to set up the loop is also shown in figure



Label	Opcode	Operand	Comments	T-states
	MVI	C,ffH	;load register C	7
LOOP:	DCR	C	;Decrement C	4
	JNZ	LOOP	;Jump back to ;decrement C	10/7

figure :- Time Delay loop :- flowchart and Instructions

The last column in figure shows the T-states (clock periods) required by the 8085 microprocessor to execute each instruction. The instruction MVI requires seven clock periods. An 8085-based microcomputer with 2 MHz clock frequency will execute the instruction MVI in 3.5 μs as follows:-

Clock frequency of the system $f = 2$ MHz

Clock period $T = 1/f = 1/2 \times 10^{-6} = 0.5$ μs

Time to execute MVI = 7 T-states $\times$ 0.5 = 3.5 μs

The following notations are used in the description of the instructions.

R	= 8085 8-bit register	(A, B, C, D, E, H, L)
M	= Memory register (location)	
Rs	= Register source	
Rd	= Register destination	(A, B, C, D, E, H, L)
Rp	= Register pair	(BC, DE, HL, SP)
( )	= Contents of	

1. **Data Transfer (Copy) Instructions.** These instructions perform the following six operations.
   - Load an 8-bit number in a register
   - Copy from register to register
   - Copy between I/O and accumulator
   - Load 16-bit number in a register pair
   - Copy between register and memory
   - Copy between registers and stack memory

	Mnemonics	Examples	Operation
1.1	MVI R,** 8-bit	MVI B, 4FH	Load 8-bit data (byte) in a register
1.2	MOV Rd, Rs**	MOV B, A MOV C, B	Copy data from source register Rs into destination register Rd
1.3	LXI Rp,** 16-bit	LXI B, 2050H	Load 16-bit number in a register pair
1.4	OUT 8-bit (port address)	OUT 01H	Send (write) data byte from the accumulator to an output device
1.5	IN 8-bit (port address)	IN 07H	Accept (read) data byte from an input device and place it in the accumulator
1.6	LDA 16-bit	LDA 2050H	Copy the data byte into A from the memory specified by 16-bit address
1.7	STA 16-bit	STA 2070H	Copy the data byte from A into the memory specified by 16-bit address
1.8	LDAX Rp	LDAX B	Copy the data byte into A from the memory specified by the address in the register pair
1.9	STAX Rp	STAX D	Copy the data byte from A into the memory specified by the address in the register pair

*These instructions are explained and illustrated in Chapters 6 and 7. The complete instruction set is explained alphabetically in Appendix F for easy reference; the appendix also includes three lists of instruction summaries arranged according to the functions, hexadecimal sequence of machine codes, and alphabetical order.
**The letters R, Rd, Rs, Rp represent generic registers. In the 8085 instructions, these are replaced by registers such as A, B, C, D, E, H, and L or register pairs.

1.10	MOV R, M	MOV B, M	Copy the data byte into register from the memory specified by the address in HL register
1.11	MOV M, R	MOV M, C	Copy the data byte from the register into memory specified by the address in HL register

2. Arithmetic Instructions. The frequently used arithmetic operations are:

- Add
- Subtract
- Increment (Add 1)
- Decrement (Subtract 1)

Mnemonics		Examples	Operation
2.1	ADD R	ADD B	Add the contents of a register to the register to the contents of A
2.2	ADI 8-bit	ADI 37H	Add 8-bit data to the contents of A
2.3	ADD M	ADD M	Add the contents of memory to A; the address of memory is in HL register
2.4	SUB R	SUB C	Subtract the contents of a register from the contents of A
2.5	SUI 8-bit	SUI 7FH	Subtract 8-bit data from the contents of A
2.6	SUB M	SUB M	Subtract the contents of memory from A; the address of memory is in HL register
2.7	INR R	INR D	Increment the contents of a register
2.8	INR M	INR M	Increment the contents of memory, the address of which is in HL
2.9	DCR R	DCR E	Decrement the contents of a register
2.10	DCR M	DCR M	Decrement the contents of a memory, the address of which is in HL
2.11	INX Rp	INX H	Increment the contents of a register pair
2.12	DCX Rp	DCX B	Decrement the contents of a register pair

3. Logic and Bit Manipulation Instructions. These instructions include the following operations:

- AND
- OR
- X-OR (Exclusive OR)
- Compare
- Rotate Bits

Mnemonics		Examples	Operation
3.1	ANA R	ANA B	Logically AND the contents of a register with the contents of A

3.2	ANI 8-bit	ANI 2FH	Logically AND 8-bit data with the contents of A
3.3	ANA M	ANA M	Logically AND the contents of memory with the contents of A; the address of memory is in HL register
3.4	ORA R	ORA E	Logically OR the contents of a register with the contents of A
3.5	ORI 8-bit	ORI 3FH	Logically OR 8-bit data with the contents of A
3.6	ORA M	ORA M	Logically OR the contents of memory with the contents of A; the address of memory is in HL register
3.7	XRA R	XRA B	Exclusive-OR the contents of a register with the contents of A
3.8	XRI 8-bit	XRI 6AH	Exclusive-OR 8-bit data with the contents of A
3.9	XRA M	XRA M	Exclusive-OR the contents of memory with the contents of A; the address of memory is in HL register
3.10	CMP R	CMP B	Compare the contents of register with the contents of A for less than, equal to, or greater than
3.11	CPI 8-bit	CPI 4FH	Compare 8-bit data with the contents of A for less than, equal to, or greater than

4. Branch Instructions. The following instructions change the program sequence.

4.1	JMP 16-bit address	JMP 2050H	Change the program sequence to the specified 16-bit address
4.2	JZ 16-bit address	JZ 2080H	Change the program sequence to the specified 16-bit address if the Zero flag is set
4.3	JNZ 16-bit address	JNZ 2070H	Change the program sequence to the specified 16-bit address if the Zero flag is reset
4.4	JC 16-bit address	JC 2025H	Change the program sequence to the specified 16-bit address if the Carry flag is set
4.5	JNC 16-bit address	JNC 2030H	Change the program sequence to the specified 16-bit address if the Carry flag is reset
4.6	CALL 16-bit address	CALL 2075H	Change the program sequence to the location of a subroutine
4.7	RET	RET	Return to the calling program after completing the subroutine sequence

5. Machine Control Instructions. These instructions affect the operation of the processor.

		HLT	Stop processing and wait
5.1	HLT		
5.2	NOP	NOP	Do not perform any operation

This set of instructions is a representative sample; it does not include various instructions related to 16-bit data operations, additional jump instructions, and conditional Call and Return instructions.