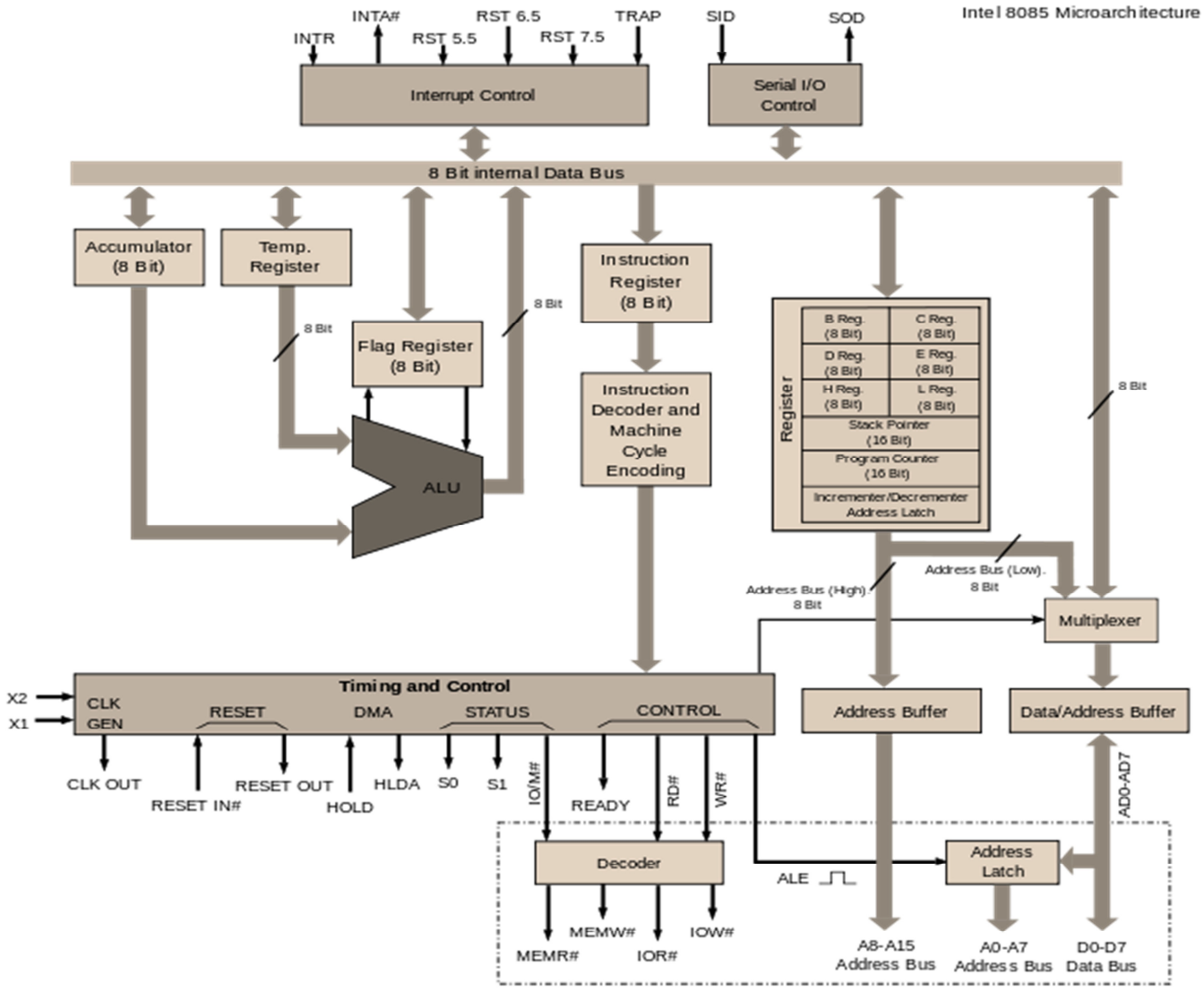
THE 8085 MICROPROCESSOR

1. Introduction

The 8085 microprocessor was made by Intel in mid 1970s. It was binary compatible with 8080 microprocessor but required less supporting hardware thus leading to less expensive microprocessor systems. It is a general purpose microprocessor capable of addressing 64k of memory. The device has 40 pins, require a +5V power supply and can operate with 3 MHz single phase clock. It has also a separate address space for up to 256 I/O ports. The instruction set is backward compatible with its predecessor 8080 even though they are not pin-compatible.

2. 8085 Internal Architecture



The 8085 has a 16 bit address bus which enables it to address 64 KB of memory, a data bus 8 bit wide and control buses that carry essential signals for various operations. It also has a built in register array which are usually labeled A(Accumulator), B, C, D, E, H, and L. Further special-purpose registers are the 16-bit Program Counter (PC), Stack Pointer (SP), and 8-bit flag register F. The microprocessor has three maskable interrupts (RST 7.5, RST 6.5 and RST 5.5), one Non-Maskable interrupt (TRAP), and one externally serviced interrupt (INTR).

- ***Control Unit:***
  Generates signals within microprocessor to carry out the instruction, which has been decoded.

- ***Arithmetic Logic Unit:***
  The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR', etc. Uses data from memory and from Accumulator to perform arithmetic and always stores the result of operation in the Accumulator.

- ***Registers:***
  The 8085 microprocessor includes six registers, one accumulator, and one flag register. In addition, it has two 16-bit registers: the stack pointer and the program counter. The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations.

- ***Accumulato*r:**
  The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

- ***Flag Registers:***
  The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

- ***Program Counter (PC)****:*
  This is a 16 bit register which points the next instruction to execute.

- ***Stack Pointer (SP):***
  The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

- ***Instruction Register / Decoder:***
  This is a temporary store for the current instruction of a program. Latest instruction is sent to here from memory prior to execution. Decoder then takes instruction and 'decodes' or interprets the instruction. Decoded instruction is then passed to next stage.

- ***Memory Address Register (MAR):***
  Holds addresses received from PC for E.g: of next program instruction. MAR feeds the address bus with address of the location of the program under execution.

- ***Control Generator:***
  Generates signals within microprocessor to carry out the instruction which has been decoded. In reality it causes certain connections between blocks of the processor to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

- ***Register Selector:***
  This block controls the use of the register stack. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

3. 8085 Pin Diagram

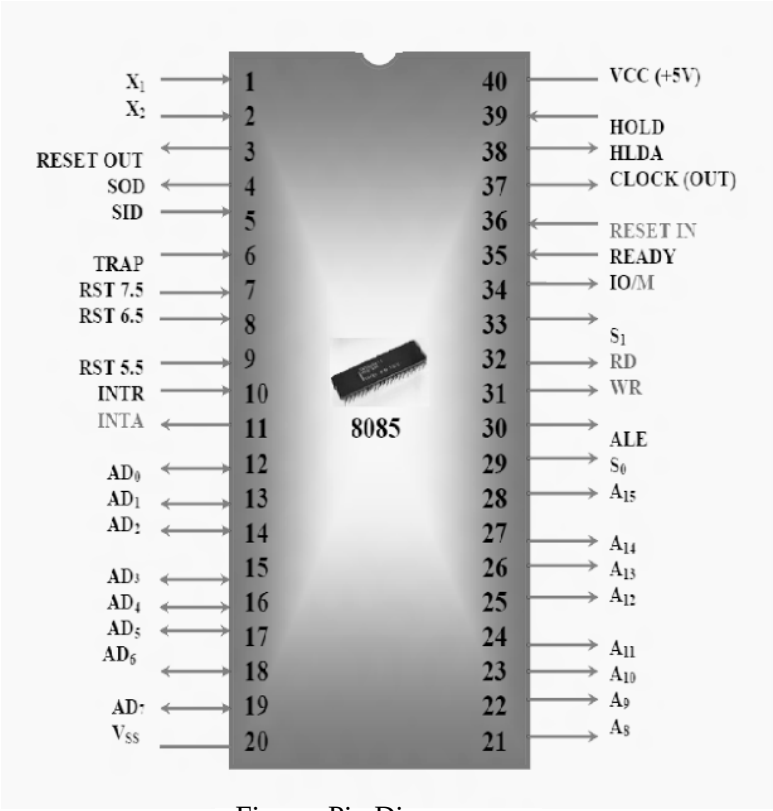


Figure: Pin Diagram

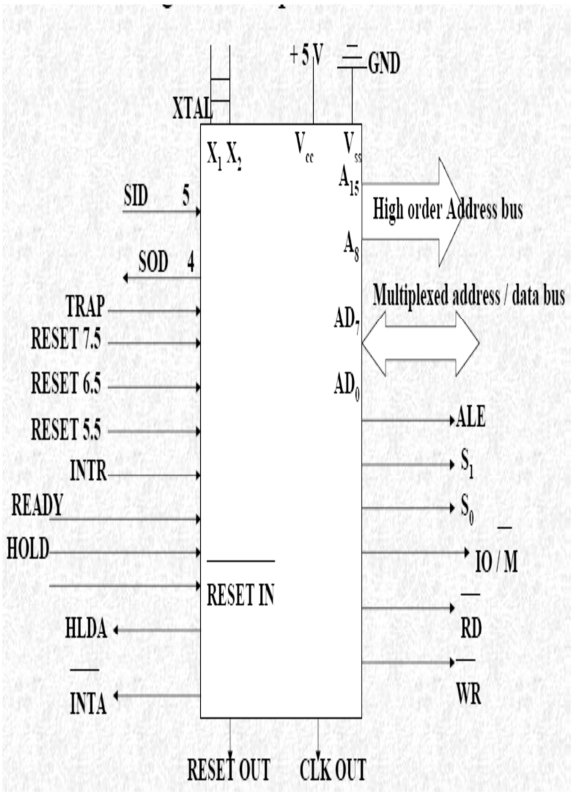


Figure: Pin group Diagram

8085 Pin Description:

- ***AD0-AD7:*** multiplexed address (lower 8-bit address) and data bus.
- ***A8-A15:*** upper 8-bit address.
- ***ALE (Output):*** Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals.
- ***SO, S1 (Output):*** Data Bus Status: Encoded status of the bus cycle:

S1	S0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

- ***RD (Output 3state):*** READ (low active); indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.
- ***WR (Output 3state):*** WRITE (low active); indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.
- ***READY (Input):*** If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

- ***HOLD (Input):*** HOLD; indicates that another Master is requesting the use of the address and data buses. The CPU, upon receiving the Hold request, will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

- ***HLDA (Output):*** HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycles after HLDA goes low.

- ***INTR (Input):*** INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only using the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

- ***INTA (Output):*** INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

- **RST 5.5, RST 6.5, RST 7.5 (Inputs):** RESTART INTERRUPTS; These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.
  RST 7.5 ~~ Highest Priority
  RST 6.5
  RST 5.5 ~~ Lowest Priority
  The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

- ***TRAP (Input):*** Trap interrupt is a non-maskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

- ***RESET IN (Input):*** Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip flops. None of the other flags or registers (except the instruction register) are affected The CPU is held in the reset condition as long as Reset is applied.

- ***RESET OUT (Output):*** Can be used as a system RESET. The signal is synchronized to the processor clock.

- ***X1, X2 (Input):*** Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

- ***CLK (Output):*** Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

- ***IO/M (Output):*** IO/M indicates whether the Read/Write is to memory or l/O Tri-stated during Hold and Halt modes.

- ***SID (Input):*** Serial input data line: The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

- ***SOD (output):*** Serial output data line: The output SOD is set or reset as specified by the SIM instruction.
- ***Vcc :*** +5 volt supply.
- ***Vss :*** Ground Reference

4. 8085 Addressing modes

The method by which the address of source of data or the address of destination of result is given in the instruction is called Addressing Modes. The term addressing mode refers to the way in which the operand of the instruction is specified. Intel 8085 uses the following addressing modes:

a. Direct Addressing Mode
b. Register Addressing Mode
c. Register Indirect Addressing Mode
d. Immediate Addressing Mode
e. Implicit Addressing Mode

a. Direct Addressing Mode:

In this mode, the address of the operand is given in the instruction itself.

LDA 2500H ; H Load the contents of memory location 2500 H in accumulator.

- LDA is the operation.
- 2500 H is the address of source.
- Accumulator is the destination.

b. Register Addressing Mode:

In this mode, the operand is in general purpose register.

MOV A, B ; Move the contents of register B to A.

- MOV is the operation.
- B is the source of data.
- A is the destination.

c. Register Indirect Addressing Mode:

In this mode, the address of operand is specified by a register pair.

MOV A, M ; Move data from memory location specified by H-L pair to accumulator.

- MOV is the operation.
- M is the memory location specified by H-L register pair.
- A is the destination.

d. Immediate Addressing Mode:

In this mode, the operand is specified within the instruction itself.

MVI A, 05H ; Move 05 H in accumulator.

- MVI is the operation.
- 05 H is the immediate data (source).
- A is the destination.

e. Implicit Addressing Mode:

If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.

CMA ; Complement accumulator.

- CMA is the operation.
- A is the source.
- A is the destination.

5. 8085 Instruction Set

Mnemonic	*Description*	*Clock Cycles*
• ***MOVE, LOAD, AND STORE***		
MOV r1, r2	Move register to register (r1 <= r2)	4
MOV M, r	Move register to memory (M <= r)	7
MOV r, M	Move memory to register (r <=M)	7
MVI r, 8 bit data	Move immediate register (r <= 8 bit data)	7
MVI M, 8 bit data	Move immediate memory (M<= 8 bit data)	10
LXI B, 16 bit data	Load immediate register Pair B & C (BC <=Data 16 bit)	10
LXI D, 16 bit data	Load immediate register Pair D & E (DE <=Data 16 bit)	10
LXI H, 16 bit data	Load immediate register Pair H & L (HL <=Data 16 bit)	10
LXI SP, 16 bit data	Load immediate stack pointer (SP <=Data 16 bit)	10
STAX B	Store A indirect ([BC] <=A)	7
STAX D	Store A indirect ([DE] <= A)	7
LDAX B	Load A indirect (A <= [BC])	7
LDAX D	Load A indirect (A <= [DE])	7
STA 16 bit address	Store A direct ([16 bit address] <=A)	13
LDA 16 bit address	Load A direct (A<= [16 bit address])	13
SHLD 16 bit address	Store H & L direct	16
LHLD 16 bit address	Load H & L direct	16
XCHG	Exchange D & E, H & L registers	4
• ***STACK OPERATION***		
PUSH B	Push register Pair B & C on stack	12
PUSH D	Push register Pair D & E on stack	12
PUSH H	Push register Pair H & L on stack	12
PUSH PSW	Push A and Flags on stack	12
POP B	Pop register Pair B & C off stack	10
POP D	Pop register Pair D & E off stack	10
POP H	Pop register Pair H & L off stack	10
POP PSW	Pop A and Flags off stack	10
XTHL	Exchange top of stack H & L	16
SPHL	Move H & L to stack pointer	6
• ***JUMP***		
JMP 16 bit address	Jump unconditional	10
JC 16 bit address	Jump on carry	7/10
JNC 16 bit address	Jump on no carry	7/10
JZ 16 bit address	Jump on zero	7/10
JNZ 16 bit address	Jump on no zero	7/10
JP 16 bit address	Jump on positive	7/10
JM 16 bit address	Jump on minus	7/10
JPE 16 bit address	Jump on parity even	7/10
JPO 16 bit address	Jump on parity odd	7/10
PCHL	Move H & L to program counter	6

- ***CALL***

CALL 16 bit address	Call unconditional	18
CC 16 bit address	Call on carry	9/18
CNC 16 bit address	Call on no carry	9/18
CZ 16 bit address	Call on zero	9/18
CNZ 16 bit address	Call on no zero	9/18
CP 16 bit address	Call on positive	9/18
CM 16 bit address	Call on minus	9/18
CPE 16 bit address	Call on parity even	9/18
CPO 16 bit address	Call on parity odd	9/18

- ***RETURN***

RET	Return	10
RC	Return on carry	6/12
RNC	Return on no carry	6/12
RZ	Return on zero	6/12
RNZ	Return on no zero	6/12
RP	Return on positive	6/12
RM	Return on minus	6/12
RPE	Return on parity even	6/12
RPO	Return on parity odd	6/12

- ***RESTART***

RST	Restart (RST 0 up to RST 7)	12

- ***INPUT/OUTPUT***

IN 8 bit address	Input	10
OUT 8 bit address	Output	10

- ***INCREMENT AND DECREMENT***

INR r	Increment register ([r] <= [r]+1)	4
DCR r	Decrement register ([r] <= [r]-1)	4
INR M	Increment memory	10
DCR M	Decrement memory	10
INX B	Increment B & C registers ([BC] <= [BC]+1)	6
INX D	Increment D & E registers	6
INX H	Increment H & L registers	6
INX SP	Increment stack pointer	6
DCX B	Decrement B & C ([BC] <= [BC]-1)	6
DCX D	Decrement D & E	6
DCX H	Decrement H & L	6
DCX SP	Decrement stack pointer	6

- ***ADD***

ADD r	Add register to A ([A] <= [A]+[r])	4
ADC r	Add register to A with carry ([A] <= [A]+[r]+carry Flag)	4
ADD M	Add memory to A ([A] <= [A]+[M])	7
ADC M	Add memory to A with carry ([A] <= [A]+[M]+carry Flag)	7

ADI 8 bit data	Add immediate to A ([A] <= [A]+ 8 bit data)	7
ACI 8 bit data	Add immediate to A with carry ([A]<=[A]+ 8 bit data+carry Flag)	7
DAD B	Add B & C to H & L ([HL] <= [HL]+[BC])	10
DAD D	Add D & E to H & L ([HL] <= [HL]+[DE])	10
DAD H	Add H & L to H & L	10
DAD SP	Add stack pointer to H & L	10

- ***SUBTRACT***

SUB r	Subtract register from A ([A] <= [A]-[r])	4
SBB r	Subtract register from A with borrow ([A] <= [A]-[r]-carry Flag)	4
SUB M	Subtract memory from A	7
SBB M	Subtract memory from A with borrow	7
SUI 8 bit data	Subtract immediate from A ([A] <= [A]- 8 bit data)	7
SBI 8 bit data	Subtract immediate from A with borrow	7

- ***LOGICAL***

ANA r	AND register with A ([A] <= [A] AND [r])	4
XRA r	Exclusive OR register with A ([A] <= [A] XOR [r])	4
ORA r	OR register with A ([A] <= [A] OR [r])	4
CMP r	Compare register with A (carry, zero flag chage)	4
ANA M	AND memory with A	7
XRA M	Exclusive OR Memory with A	7
ORA M	OR memory with A	7
CMP M	Compare memory with A	7
ANI 8 bit data	AND immediate with A ([A] <= [A] AND 8 bit data)	7
XRI 8 bit data	Exclusive OR immediate with A	7
ORI 8 bit data	OR immediate with A	7
CPI 8 bit data	Compare immediate with A	7

- ***ROTATE***

RLC	Rotate A left	4
RRC	Rotate A right	4
RAL	Rotate A left through carry	4
RAR	Rotate A right through carry	4

- ***SPECIALS***

CMA	Complement A ([A] <= 1's complement [A])	4
STC	Set carry	4
CMC	Complement carry	4
DAA	Decimal adjust A (binary to BCD conversion)	4

- ***CONTROL***

EI	Enable Interrupts	4
DI	Disable Interrupts	4
NOP	No-operation	4
HLT	Halt (Power down)	5
RIM	Read Interrupt Mask	4
SIM	Set Interrupt Mask	4

6. Introduction to MICROFRIEND DYNA-85

Microfriend DYNA-85 is an introduction to a low cost trainer and development kit. It was developed to assist the novice to get familiar with INTEL 8085 microprocessor in a user friendly environment.

a. System Overview

- **Central Processing Unit:**
  MICROFRIEND DYNA - 85 i s based on the Intel 8085a high performance CPU operating at 3 mhz.
- **Memory:**
  Powerful system monitor has been provided on a 2732 eprom covering 4k bytes. This monitor includes all standard commands, codes, functions and utility subroutines. A 6116 battery backup ram (2k) is provided on the board for inputting and executing programs. Three 28 pin sockets are provided for memory chips so that further expansion of ram/eprom is possible up to a maximum of 56k .
- **Hex keypad / display interface:**
  A keypad with 21 keys and 6 digits led seven segments display is provided for interaction with the system using 8279 keyboard /display controller.

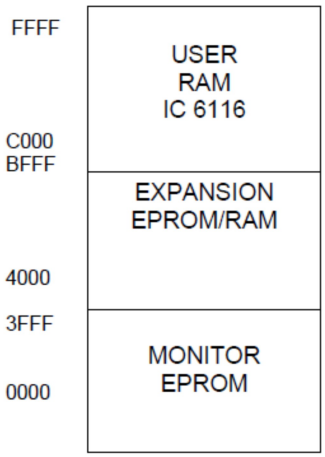
b. System commands overview

The hex keypad mode supports the following commands:

- **RESET:** Provides hardware reset, display shows "friend" on pressing this key.
- **VI:** Vector interrupt key, activate rst 7.5 vectored interrupt.
- **SET:** Allows the user to examine & modify the contents of ram and only examination of contents is possible in case of eprom.
- **INR:** Increments memory address presently displayed in the address field of display.
- **DCR:** Decrements memory address presently displayed in the address field of display.
- **REG:** Allows the user to examine contents of cpu registers & modify them if necessary.
- **GO:** Allows the user to load the program counter by the desired memory address which is the starting address of the program to be executed.
- **EXEC:** Used to start the execution of go or code command.
- **CODE:** Used for selecting one of the coded subroutines in the monitor.

c. System Memory

The system memory is also as important as the CPU itself, because this is where the system program resides and the CPU takes its instruction from the program. The memory is of two types ROM and RAM i.e. READ ONLY MEMORY & RANDOM ACCESS MEMORY.



- **0000H TO 3FFFH :**

  Monitor EPROM socket.Monitor 2732 is located at 000H to 0FFFH and is mapped at 1000H-1FFFH,2000H-2FFFH and 3000-3FFFh also.

- **4000H to BFFFH :**
  This Socket is used for user expansion of EPROM and RAM. EPROMs like 2716 / 2732 / 2732 / 2764 / 27128 / 27256 or RAMs like 6116 / 6264 / 62256 can be installed by suitable strappings.

- **C000H to FFFFH :**
  User RAM socket.The 2K user RAM IC 6116 is located at F800HFFFFH. This 2K memory is folded after every 2K bytes from C000H to FFFFH.In this socket 6264 can also be used.

d. Entering a Program and Execution Procedure

While using the built in keyboard, we can work only in the Hexadecimal number system. When the system is switched on the display shows, sign on message 'F r l E n d'. This indicates that the system is reset and the monitor expects a command from you.

- **Program input procedure:**

  RES => **SET**=> Starting address (COOOH onward) => **INR/DCR** (write HEX converted program into Memory, INR and DCR is for increment, decrement of memory location)

- **Program Execution procedure:**

  RES => **GO** => Starting address (COOOH, initial address of program) => **EXEC** (if program is executed properly, display shows 'E').

- **Register value checking procedure:**

  RES => **REG** => **A** (or any Register) => **INR/DCR** (to other registers' value).

EXAMPLE 1 :

Key Pressed	Address Field	Data Field
RES	F r I E	n d
SET		
F	0 0 0 F	
0	0 0 F 0	
0	0 F 0 0	
0	F 0 0 0	
INR	F 0 0 0	X X
3	F 0 0 0	0 3
E	F 0 0 0	3 E
INR	F 0 0 1	X X
DCR	F 0 0 0	3 E
EXEC	F	

e. 8085 Mnemonic Codes

Hex	mnemonic	Hex	mnemonic	Hex	mnemonic	Hex	mnemonic
CE	ACI 8-Bit	3F	CMC	2B	DCX H	01	LXI B,16-Bit
8F	ADC A	BF	CMP A	3B	DCX SP	11	LXI D,16-Bit
88	ADC B	B8	CMP B	F3	DI	21	LXI H,16-Bit
89	ADC C	B9	CMP C	FB	EI	31	LXI SP,16-Bit
8A	ADC D	BA	CMP D	76	HLT	7F	MOV A A
8B	ADC E	BB	CMP E	DB	IN 8-Bit	78	MOV A B
8C	ADC H	BC	CMP H	3C	INR A	79	MOV A C
8D	ADC L	BD	CMP	04	INR B	7A	MOV A D
8E	ADC M	BE	CMP M	0C	INR C	7B	MOV A E
87	ADD A	D4	CNC 16-Bit	14	INR D	7C	MOV A H
80	ADD B	C4	CNZ 16-Bit	1C	INR E	7D	MOV A L
81	ADD C	F4	CP 16-Bit	24	INR H	7E	MOV A M
82	ADD D	EC	CPE 16-Bit	2C	INR L	47	MOV B A
83	ADD E	FE	CPI 8-Bit	34	INR M	40	MOV B B
84	ADD H	E4	CPO 16-Bit	03	INX B	41	MOV B C
85	ADD L	CC	CZ 16-Bit	13	INX D	42	MOV B D
86	ADD M	27	DAA	23	INX H	43	MOV B E
C6	ADI 8-Bit	09	DAD B	33	INX SP	44	MOV B H
A7	ANA A	19	DAD D	DA	JC 16-Bit	45	MOV B L
A0	ANA B	29	DAD H	FA	JM 16-Bit	46	MOV B M
A1	ANA C	39	DAD SP	C3	JMP 16-Bit	4F	MOV C A
A2	ANA D	3D	DCR A	D2	JNC 16-Bit	48	MOV C B
A3	ANA E	05	DCR B	C2	JNC 16-Bit	49	MOV C C
A4	ANA H	0D	DCR C	F2	JP 16-Bit	4A	MOV C D
A5	ANA L	15	DCR D	EA	JPE 16-Bit	4B	MOV C E
A6	ANA M	1D	DCR E	E2	JPO 16-Bit	4C	MOV C H
E6	ANA 8-Bit	25	DCR H	CA	JZ 16-Bit	4D	MOV C L
CD	CALL 16-Bit	2D	DCR L	3A	LDA 16-Bit	4E	MOV C M
DC	CC 16-Bit	35	DCR M	0A	LDAX B	57	MOV D A
FC	CM 16-Bit	0B	DCX B	1A	LDAX D	50	MOV D B
2F	CMA	1B	DCX D	2A	LHLD 16-Bit	51	MOV D C

Hex	mnemonic	Hex	mnemonic	Hex	mnemonic	Hex	mnemonic
52	MOV D D	71	MOV M C	E5	PUSH H	9E	SBB M
53	MOV D E	72	MOV M D	F5	PUSH PSW	DE	SBI 8-Bit
54	MOV D H	73	MOV M E	17	RAL	22	SHLD 16-Bit
55	MOV D L	74	MOV M H	1F	RAR	30	SIM
56	MOV D M	75	MOV M L	D8	RC	F9	SPHL
5F	MOV E A	3E	MVI A 8-Bit	C9	RET	32	STA 16-Bit
58	MOV E B	06	MVI B 8-Bit	20	RIM	02	STAX B
59	MOV E C	OE	MVI C 8-Bit	07	RLC	12	STAX D
5A	MOV E D	16	MVI D 8-Bit	F8	RM	37	STC
5B	MOV E E	1E	MOV E 8-Bit	D0	RNC	97	SUB A
5C	MOV E H	26	MVI H 8-Bit	C0	RNC	90	SUB B
5D	MOV E L	2E	MVI L 8-Bit	F0	RP	91	SUB C
5E	MOV E M	36	MVI M 8-Bit	E8	RPE	92	SUB D
67	MOV H A	00	NOP	E0	RPO	93	SUB E
60	MOV H B	B7	ORA A	0F	RRC	94	SUB H
61	MOV H C	B0	ORA B	C7	RST 0	95	SUB L
62	MOV H D	B1	ORA C	CF	RST 1	96	SUB M
63	MOV H E	B2	ORA D	D7	RST 2	D6	SUI 16-Bit
64	MOV H H	B3	ORA E	DF	RST 3	EB	XCHG
65	MOV H L	B4	ORA H	E7	RST 4	AF	XRA A
66	MOV H M	B5	ORA L	EF	RST 5	A8	XRA B
6F	MOV L A	B6	ORA M	F7	RST 6	A9	XRA C
68	MOV L B	F6	ORI 8-Bit	FF	RST 7	AA	XRA D
69	MOV L C	D3	OUT 8-Bit	C8	RZ	AB	XRA E
6A	MOV L D	E9	PCHL	9F	SBB A	AC	XRA H
6B	MOV L E	C1	POP B	98	SBB B	AD	XRA L
6C	MOV L H	D1	POP D	99	SBB C	AE	XRA M
6D	MOV L L	E1	POP H	9A	SBB D	EE	XRI 8-Bit
6E	MOV L M	F1	POP PSW	9B	SBB E	E3	XTHL
77	MOV M A	C5	PUSH B	9C	SBB H		
70	MOV M B	D5	PUSH D	9D	SBB L		

f. Programming Example:

- **Copy Data of Register B to Register C.**

Address	*HexCode*	*Mnemonic*	*Comment*
C000H	06H	MVI B	Store data 50H in Register B
C001H		50H	
C002H	48H	MOV C, B	Copy data of Register B to C
C003H	76H	HLT	Stop program

Output:
At B=50H
At C=50H

Lab1 Exercises:

I. Add two 8-bit numbers.
II. Subtract two 8-bit numbers.
III. Add two 16-bit numbers.

Method for I and II: The numbers to be added or subtracted are stored in memory locations. First number is brought to accumulator and the second number in the memory is added or subtracted from it. If borrow is generated, the result stored in the accumulator is complemented and a 1 is added to it. Finally, the result and carry or borrow are stored in memory locations.

Method for III: The numbers to be added are stored in two 16 bit registers. They are added and the resultant sum and carry are stored in memory locations.

Lab2 Exercises:

I. Write Assembly language programs which reflect addressing modes of 8085 microprocessor (Note: 5 programs, each for one addressing mode) and verify each in DYNA-85 kit.

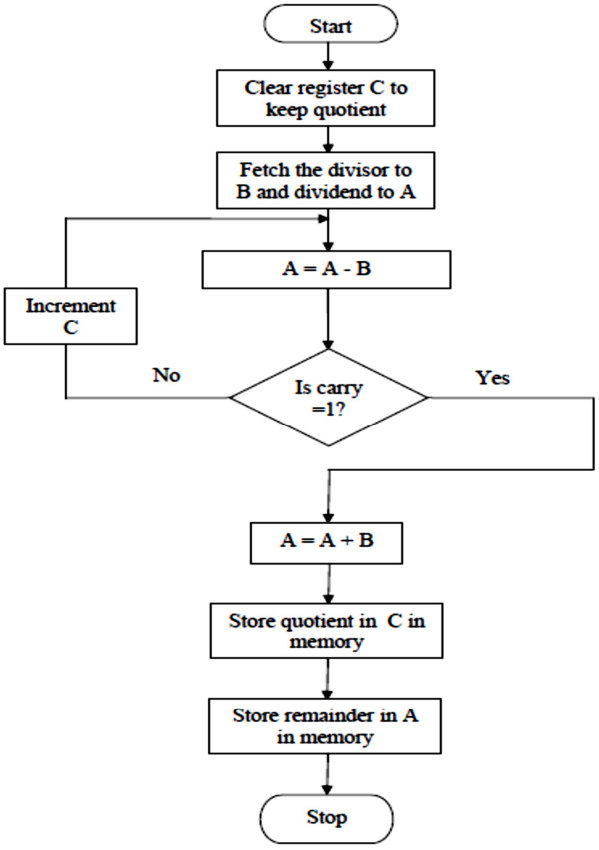
Hints:

- Add data 05H immediately with the content of Register A (contain 30H) and evaluate flag register (*Immediate Addressing Mode*).
- Add 05H stored in Register C with 30H stored in Register D. Then, AND result with FFH stored in Register B (*Register Addressing Mode*).
- Copy data of location from CODE H to FOOD H using *Direct Addressing Mode*.
- Copy data of location from CODE H to FOOD H using *Register Indirect Addressing Mode*.
- Complement 05H data (*Implied Addressing Mode*).

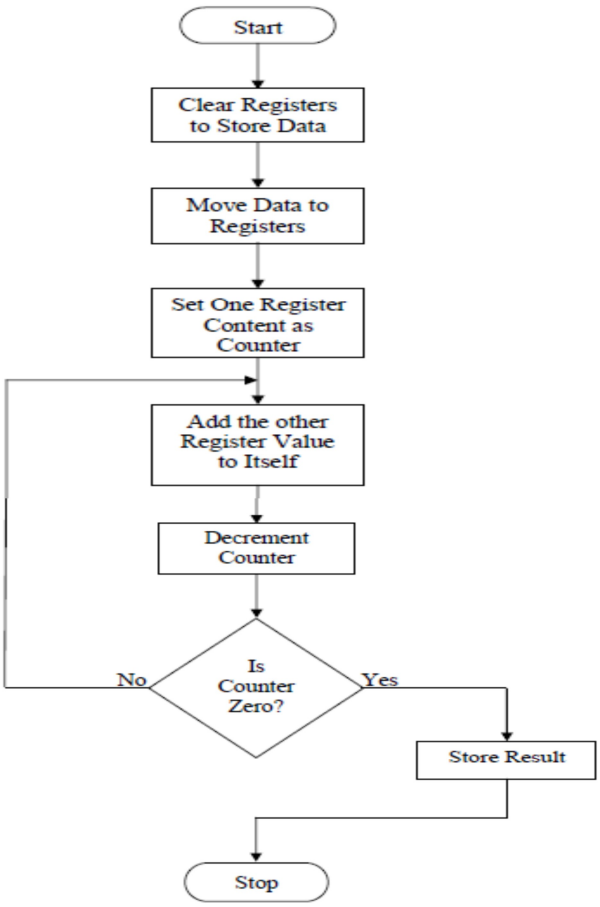
Lab3 Exercises:

I. Divide two 8-bit numbers.
II. Multiply two 8 bit numbers.
III. Sort 10 numbers stored in consecutive memory locations in ascending order.
IV. Sort 10 numbers stored in consecutive memory locations in descending order.
V. Find out the largest of ten 8 bit numbers.
VI. Find out the smallest of ten 8 bit numbers.

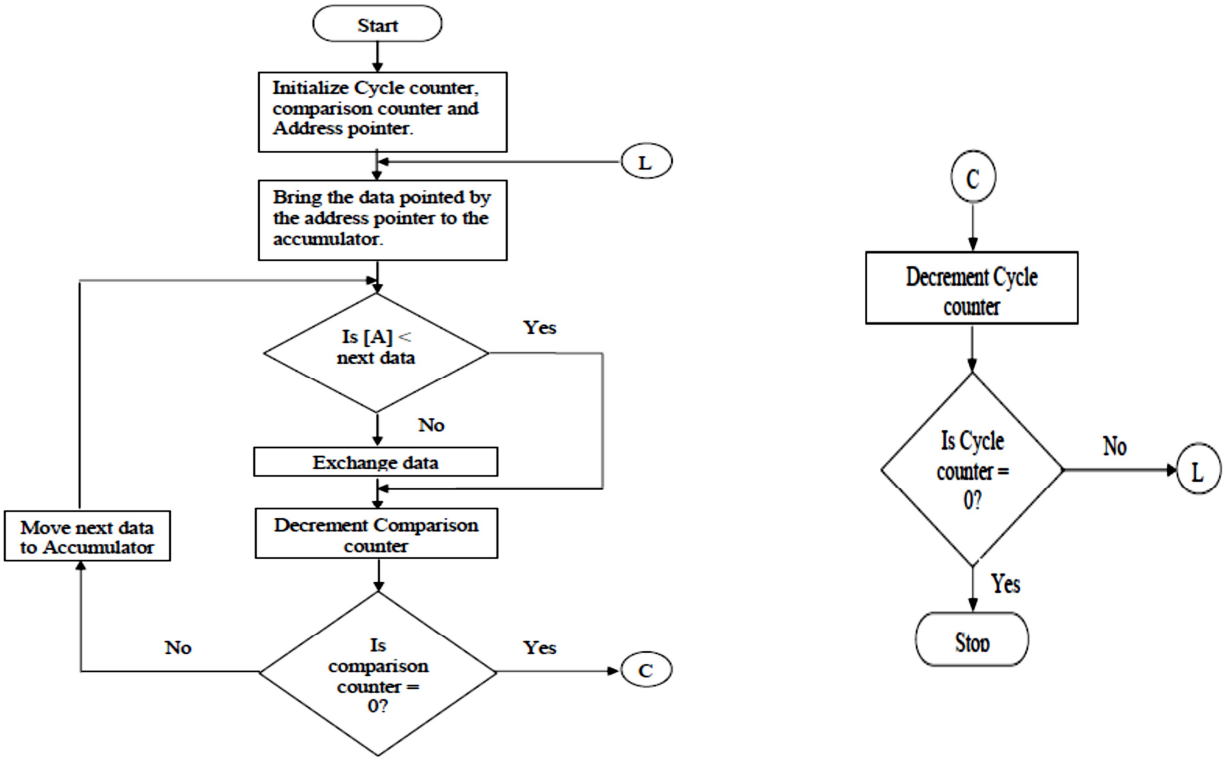
Hints:



Flow Chart for LAB3 Q.I



Flow Chart for LAB3 Q.II



Flow Chart for LAB3 Q.III