MICROPROCESSOR SYSTEMS

8085 Microprocessor

Zuhaib A. Shaikh,
Asst. Prof., CSE Deptt.,QUEST
Web: zuhaib-shaikh.neocities.org

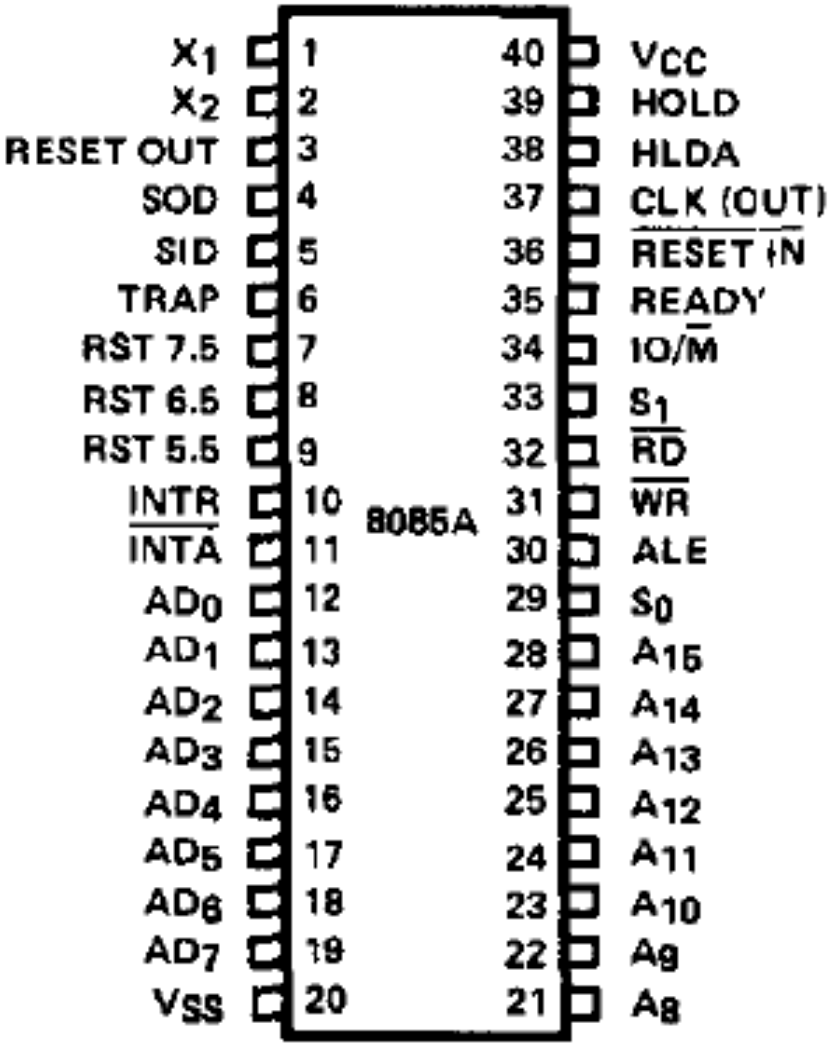
The Intel ® 8085 Microprocessor

- Intel introduced its first microprocessor in 1971
- 4004 was a 4-bit microprocessor
- 8008 in the same year was an 8-bit microprocessor
- Intel in 1974 introduced 8080, an 8-bit microprocessor having 16-bit address line
- Intel 8085 is an enhancement over 8080 having:
  - Clock, control and interrupt control with in CPU IC
  - Two added instructions
  - +5v power supply
- The generic microprocessor studied was a simplified version of 8080/8085
- 8085 will have some more registers, control lines and features

The Intel ® 8085 Pin Diagram and Functions

- It's a 40-pin DIP
- Due to less pins available, the data bus is multiplexed with lower address lines (AD_0-AD_7)
- To multiplex means first select one and then select other
- MPU first uses these lines as lower byte of address and then send/receive data on theses lines
- To inform peripherals about address/data, MPU uses a control signal Address Latch Enable (30)
- Address / Data lines may be tri-state
- Power pins (20, 40)
- Clock circuitry (1,2)

8085 Pin Diagram (Fig.)



The Intel ® 8085 Pin Diagram and Functions (Contd.)

- When RESETIN* (36) goes low, PC resets to 0000h
- RD* and WR* are same as of generic microprocessor
- RESETOUT (3) is a signal to peripherals that it is being reset
- CLK (37)
- INTR (10) can be enabled / disabled by software
- Beside the regular INTR, 8085 has four other interrupt inputs
  - TRAP (causes MPU to jump to a routine at specific address)
  - RST 7.5 (causes MPU to jump to a routine at specific address)
  - RST 6.5 (causes MPU to jump to a routine at specific address)
  - RST 5.5 (causes MPU to jump to a routine at specific address)
  - INTR being at the lowest priority interrupt
  - When INTR is received, the MPU sends INTA* signal to the device and device then sends a code which decides the address to jump at

The Intel ® 8085 Pin Diagram and Functions (Contd.)

- SOD, SID (4,5)
  - Serial output data and Serial Input data
  - SID loads 1 bit into MSB of Accumulator with RIM instruction
  - SOD is set/reset by the MPU SIM instruction
- READY input (35)
  - Comes from the peripheral device indicating that it is ready to send / receive the data
  - If this is LOW, MPU can wait or it can proceed with read / write tasks
  - It is used when slow devices communicate with the MPU
- HOLD input (39) and HLDA output (37)
  - Used when a device wants to use DMA
  - MPU sends HLDA and relieves control over Buses and control signals

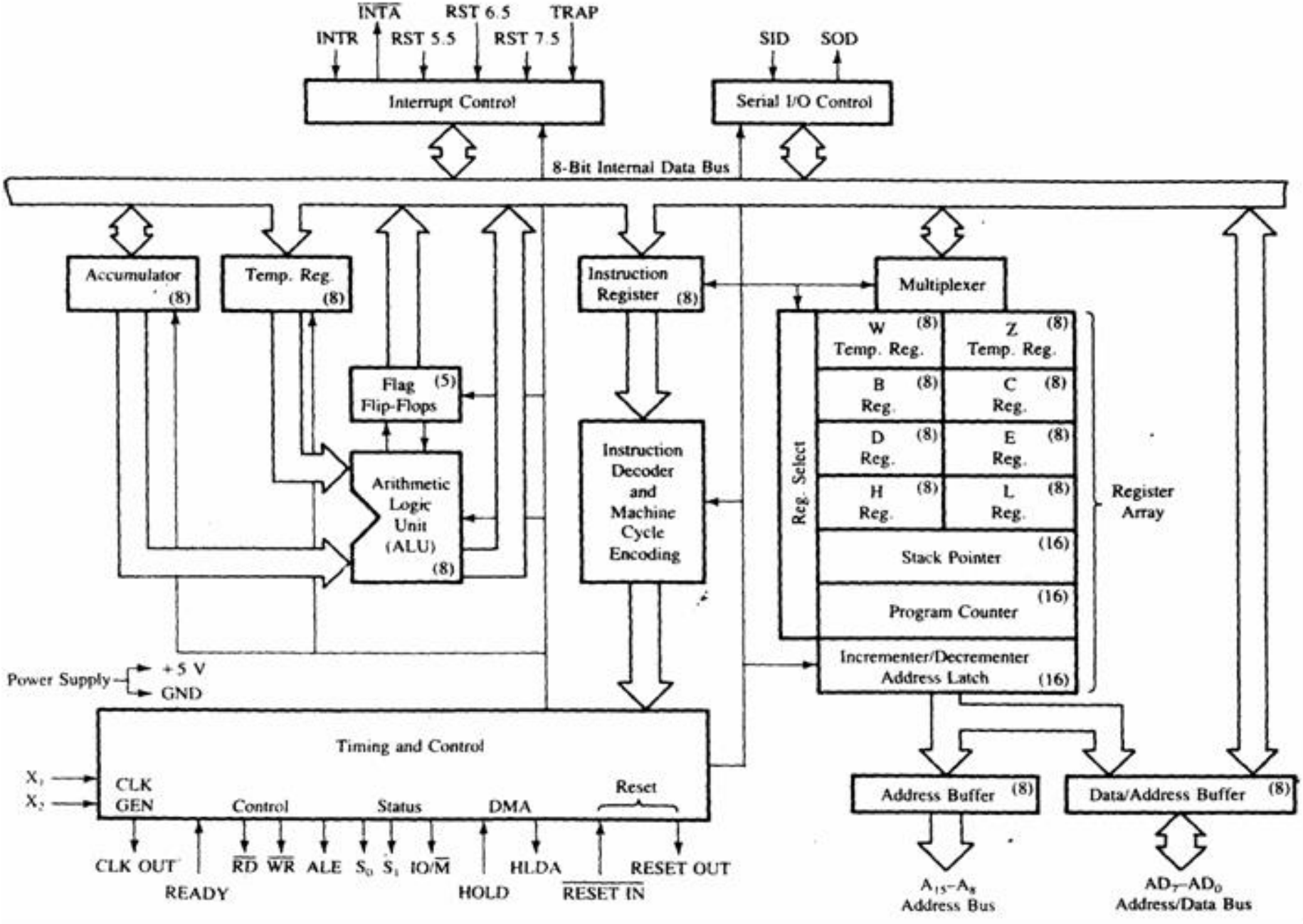
The Intel ® 8085 Pin Diagram and Functions (Contd.)

- IO/M* output (34)
  - Control signal to indicate from where read / write operations is being done (I/O or memory)
- S0 and S1 outputs (29,33)
  - Used to select one of the machine cycle

8085 Control Signals			Machine Cycle Status
I/O M*	S_1	S_0	
0	0	1	Memory Write
0	1	0	Memory Read
1	0	1	I/O Write
1	1	0	I/O Read
0	1	1	Opcode Fetch
1	1	1	Interrupt acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

The Intel ® 8085 Architecture

- 16-bit program counter
- Address bus (A8-A15) and dual purpose Address/Data bus
- Internal 8-bit bi directional data bus is responsible for communication between accumulator and rest of the registers
- Serial input and Serial output (SOD and SID)
- Interrupt control circuitry
- Timing and Control sections
- Instruction Decoder and Instruction Register


INTA
INTR
RST 5.5
RST 6.5
RST 7.5
TRAP
Interrupt Control
SID
SOD
Serial I/O Control
8-Bit Internal Data Bus
Accumulator (8)
Temp. Reg. (8)
Flag Flip-Flops (5)
Arithmetic Logic Unit (ALU) (8)
Instruction Register (8)
Instruction Decoder and Machine Cycle Encoding
Multiplexer
W Temp. Reg. (8)
Z Temp. Reg. (8)
B Reg. (8)
C Reg. (8)
D Reg. (8)
E Reg. (8)
H Reg. (8)
L Reg. (8)
Reg. Select
Register Array
Stack Pointer (16)
Program Counter (16)
Incrementer/Decrementer Address Latch (16)
Power Supply
+5 V
GND
Timing and Control
X1
X2
CLK GEN
Control
Status
DMA
Reset
CLK OUT
READY
RD WR ALE S0 S1 IO/M
HOLD
HLDA
RESET IN
RESET OUT
Address Buffer (8)
Data/Address Buffer (8)
A15–A8 Address Bus
AD7–AD0 Address/Data Bus

The Intel ® 8085 Architecture (Contd.)

- Registers:
  - 8085 has eight registers each of 8-bits
    - Accumulator
    - B and C, D and E, H and L
    - Flags Register
  - 16-bits Registers
    - Program counter
    - Stack Pointer
- Flags:
  - Five active flags
    - B7-Sign (S), B6-Zero (Z), B4-Auxiliary Carry (AC), B2-Parity (P), B0-Carry (C)

The Intel ® 8085 Architecture (Contd.)

- Stack Pointer
  - Stores the address of the last byte of the stack
  - Stack is the Read/Write memory
  - It is incremented when push, decremented when pop
- ALU
  - Arithmetic, Logical and Rotate operations are performed
- Internal Clock generator
  - Internal clock is driven by a crystal of 6-10 MHz
  - External CLK is one half of the internal clock

The Intel ® 8085 Architecture (Contd.)

- Serial input and output
  - RIM (Read Interrupt Mask) instruction transfers one bit from SID to MSB of accumulator
  - SIM (Set Interrupt Mask) instruction sets / resets SOD to what ever value of bit 7 of accumulator.
- Interrupts
  - TRAP
    - Highest priority interrupt (Non-mask able). Processor saves PC and goes to 0024h
  - RST 7.5 (jumps to 003Ch)
  - RST 6.5 (jumps to 0034h)
  - RST 5.5 (jumps to 002Ch)
  - INTR (Lowest priority)

Interrupts & Interrupt handler addresses

Name	Priority	Address Branched to when interrupt occurs
TRAP	1	24h
RST7.5	2	3Ch
RST6.5	3	34h
RST5.5	4	2Ch
INTR	5	(1)*

(1)* Depends upon the number provided by the 8259 or other circuitry to the MPU when interrupt is acknowledged.

Addressing Modes

- The way operands are specified is Addressing Mode
- 8085 uses 5 addressing modes
  - Implied
    - No register or memory location involved. Like STC, NOP, HLT
  - Register
    - Both operands are in MPU registers like ADD C
  - Immediate
    - Data immediately follows in the next byte to op code (MVI)
  - Direct
    - 3-byte instructions. 1st byte op code. Next 2 bytes address like LDA
  - Register Indirect
    - Uses register pair to address memory like ADD M

8085 Instruction Set

- Intel 8085 uses stored program concepts
  - Programs are stored in Program Memory
  - Data are stored in Data memory
- In program memory the program is stored as:
  - Sequence of 1,2 or 3-byte instructions
- The first byte of instruction is always the op-code
- The next byte (s) if present tells about the operands
- Op-code + data are in binary form
- Op-codes are fixed by the manufacturers in the design of chip
- The set of instructions (op-codes) along with some additional information is called *Instruction Set* of microprocessor

8085 Instruction Set (Contd.)

- Intel ® 8085 instruction set is divided into following group
  - Data Transfer Group
    - Moves data between register and register - memory
  - Arithmetic Group
    - Addition, Subtraction, Increment, Decrement register and memory
  - Logic Group
    - Performs Logical AND, OR, XOR, Caparison, Rotation, complement
  - Branch Group
    - Conditional and un conditional Jumps, Call and return
  - Stack, I/O and Control Group

8085 Instruction Set (Contd.)

- Intel ® 8085 instruction set
- A total of 246 instructions can be executed by 8085
- The instruction set of 8080 and 8085 is same except two added instructions in 8085 (RIM and SIM)

8085 Instruction Set (Contd.)

	Mnemonic	Op Code	Description
A	ADD A	87	Add A to A (double A)
	ADD E	80	Add B to A
	ADD C	81	Add C to A
	ADD D	82	Add D to A
	ADD E	83	Add E to A
	ADD H	84	Add H to A
	ADD L	85	Add L to A
	ADD M	86	Add memory LOC (H&L) to A
	ADI v	C6	Add immediate data v to A
	ADC A	8F	Add A to A with carry (double A with carry)
	ADC B	88	Add B to A with carry
	ADC C	89	Add C to A with carry
	ADC D	8A	Add D to A with carry
	ADC E	8B	Add E to A with carry
	ADC H	8C	Add H to A With carry
	ADC L	8D	Add L to A with carry
	ADC M	8E	Add memory LOC (H & L) to A with carry
	ACI v	CE	Add immediate data v to A with carry
	ANA A	A7	Test A and clear carry
	ANA B	A0	AND B with A
	ANA C	A1	AND C with A
	ANA D	A2	AND D with A
	ANA E	A3	AND E with A
	ANA H	A4	AND H with A
	ANA L	A5	AND L with A
	ANA M	A6	AND memory LOC (H & L) with A
	ANI v	E6	AND immediate data v with A
C	CALL aa	CD	Call subroutine at address aa
	CZ aa	CC	If zero. CALL at address aa
	CNZ	C4	U not zero, CALL at address aa
	CP aa	F4	If plus, CALL at address aa
	CM aa	FC	If minus, CALL at address aa
	CC aa	DO	If carry. CALL at address no
	CNC aa	D4	If no carry. CALL at address an
	CPE aa	EC	If even parity. CALL at address aa
	CPO aa	E4	If odd parity. CALL at address aa
	CMA	2F	Complement A
	CMC	3F	Complement carry
	CMP A	BF	Set zero flag
	CMP B	B8	Compare A with B
	CMP C	B9	Compare A with C
	CMP D	BA	Compare A with D
	CMP E	BB	Compare A with E
	CMP H	BC	Compare A with H
	CMP L	BD	Compare A with L
	CMP M	BE	Compare A with memory LOC (H & L)
	CPI v	FE	Compare A with immediate data p
D	DAA	27	Decimal adjust A
	DAD E	09	Add B&C to H&L
	DAD D	19	Add D&E to H&L
	DAD H	29	Add H&L to H&L (double H&L)
	DAD SP	39	Add SP to H&L
	DCR A	3D	Decrement A
	DCR B	05	Decrement B
	DCR C	0D	Decrement C
	DCR D	15	Decrement D
	DCR E	1D	Decrement E
	DCR H	25	Decrement H
	DCR L	2D	Decrement L
	DCR M	35	Decrement memory LOC (H & L)
	DCX B	0B	Decrement B & C
	DCX D	1B	Decrement D & E
	DCX H	2B	Decrement H & L

	Mnemonic	Opcode (hex)	Description
D	DCX SP	3B	Decrement SP
	DI	F3	Disable interrupts
E	EI	FB	Enable interrupts
H	HLT	76	Halt until interrupt
I	IN u	0B	Input from device ii
	INR A	3C	Increment A
	INR B	04	Increment B
	INR C	0C	Increment C
	INR D	14	Increment D
	INR B	1C	Increment E
	INR H	24	Increment H
	INR L	2C	Increment L
	INR M	34	Increment memory LOC (H & L)
	INX B	03	Incrément B & C
	INX D	13	Incrément D & E
	INX H	23	Increment H & L
	INX SP	33	Increment SP
J	JMP no	C3	Jump to address aa
	JZ aa	CA	If zero, JMP to address aa
	JNZ no	C2	If not zero, JMP to address a
	JP no	F2	It plus, JMP to address aa
	JM an	FA	If minus, JMP to address aa
	JC aa	PA	IC carry, JMP to address aa
	JNC aa	P2	If no carry, JMP to address aa
	JPE an	EA	II even parity, JMP to address a
	JPO an	E2	If odd panty, JMP to address aa
L	LDA aa	3A	Load A from address aa
	LDAX B	0A	Load A from memory LOC (B & C)
	LDAX D	1A	Load A from memory LOC (D&E)
	LHLD aa	2A	Load H & L from address aa
	LXI B, vv	01	Load B & C with immediate data vv
	LXI D. vv	11	Load D & E with immediate data vv
	LXIH, vv	21	Load H & L with immediate data vv
	LXI SP, vv	31	Load SP with immediate data vv
M	MOV A, B	78	Move B to A
	MOV A, C	79	Move C to A
	MOV A, D	7A	Move D to A
	MOV A, E	78	Move E to A
	MOV A, H	7C	Move H to A
	MOV A, L	7D	Move L to A
	MOVA, M	7E	Move memory LOC (H&L) to A
	MOV B, A	47	Move A to B
	MOV B, C	41	Move C to B
	MOV B, D	42	Move D to B
	MOV B, E	43	Move B to B
	MOV B, H	44	Move H to B
	MOV B, L	45	Move L to B
	MOV B, M	46	Move memory LOC (H&L) to B
	MOV C, A	4F	Move A to C
	MOV C, B	48	Move B to C
	MOV C, D	4A	Move D to C
	MOV C, E	48	Move E to C
	MOV C, H	4C	Move H to C
	MOV C, L	4D	Move L to C
	MOV C, M	4E	Move memory LOC (H & L) to C
	MOV D, A	57	Move A to D
	MOV D, B	50	Move B to D
	MOV D, C	51	Move C to
	MOV D, E	53	Move E to D
	MOV D, H	54	Move H to D
	MOV D, L	55	Move L to D
	MOV D, M	56	Move memory LOC (H&L) to D

8085 Instruction Set (Contd.)

	Mnemonic	Op code (hex)	Description
M	MOV E, A	5F	Move A to E
	MOV E, B	58	Move B to E
	MOV E, C	59	Move C to E
	MOV E, D	5A	Move D to E
	MOV E, H	5C	Move H to E
	MOV E, L	5D	Move L to E
	MOV E, M	5E	Move memory LOC (H & L) to E
	MOV H, A	67	Move A to H
	MOV H, B	60	Move B to H
	MOV H, C	61	Move C to H
	MOV H, D	62	Move D to H
	MOV H, E	63	Move E to H
	MOV H, L	65	Move L to H
	MOV H, M	66	Move memory LOC(H&L) to H
	MOV L, A	6F	Move A to L
	MOV L, B	68	Move B to L
	MOV L, C	69	Move C to L
	MOV L, D	6A	Move D to L
	MOV L, E	6B	Move E to L
	MOV L, H	6C	Move H to L
	MOV L, M	6E	Move memory LOC (H & L) to L
	MOV M, A	77	Move A to memory LOC (H & L)
	MOV M, B	70	Move B to memory LOC (H & L)
	MOV M, C	71	Move C to memory LOC (H & L)
	MOV M, D	72	Move D to memory LOC (H & L)
	MOV M, E	73	Move E to memory LOC (H & L)
	MOV M, H	74	Move H to memory LOC (H & L)
	MOV M, L	75	Move L to memory LOC (H & L)
	MVI A, v	3E	Move immediate data v to A
	MVI B, v	06	Move immediate data v to B
	MVI C, v	0E	Move immediate data v to C
	MVI D, v	16	Move immediate data v to D
	MVI E, v	1E	Move immediate data v to E
	MVI H, v	26	Move immediate data v to H
	MVI L. v	2E	Move immediate data v to L
N	NOP	00	No operation
O	ORA A	B7	Test A and clear carry
	ORA B	B0	OR B with A
	ORA C	B1	OR C with A
	ORA D	B2	OR D with A
	ORA E	B3	OR E with A
	ORA H	B4	OR H with A
	ORA L	B5	OR L with A
	ORA M	B6	OR memory LOC (H & L) with A
	ORI v	F6	OR immediate data v with A
	OUT v	03	Output A to device v
P	PCHL	E9	Jump to memory LOC contained in (H & L)
	POP B	C1	Pop B&C from stack
	POP D	D1	Pop D & E from stack
	POP H	E1	Pop H & L from stack
	POP PSW	F1	Pop A and flags from stack
	PUSH B	CS	Push B & C onto stack
	PUSH D	05	Push D & E onto stack
	PUSH H	ES	Push H & L onto stack
	PUSH PSW	F5	Push A and flags Onto stack
R	RAL	17	Rotate CY+ A left
	RAR	IF	Rotate CY + A right
	RLC	07	Rotate A left and into carry
	RRC	OF	Rotate A right and into carry
	RIM	20	Read interrupt mask (8085 only)
	RET	C9	Return from subroutine
	RZ	CS	If zero, return from subroutine

	Mnemonic	Op code (hex)	Description
R	RNZ	co	If not zero, return from subroutine
	RP	FO	1f plus, return from subroutine
	RM	F8	If minus, return from subroutine
	RC	D8	If carry. return from subroutine
	RNC	DO	If no carry, return from subroutine
	RPE	E8	If even panty, return from subroutine
	RPO	EO	If odd panty, return from subroutine
	RST 0	C7	Restart subroutine at address 00H
	RST 1	CF	Restart subroutine at address 08H
	RST 2	D7	Restart subroutine at address 1OH
	RST 3	DF	Restart subroutine at address 18H
	RST 4	E7	Restart subroutine at address 20H
	RST S	EF	Restart subroutine at address 28H
	RST 6	FI	Restart subroutine at address 30H
	RST 7	FT	Restart subroutine at address 38H
S	SIM	30	Set interrupt mask (8085 only)
	SPPHL	F9	Load SP from H & L
	SHLD aa	22	Store H & L at memory LOC aa
	STA aa	32	Store A at memory LOC aa
	STAX B	02	Store A at memory LOC (B & C)
	STAX D	12	Store A at memory LOC (D & E)
	STC	37	Set carry flag
	SUB A	97	Clear A
	SUB B	90	Subtract B from A
	SUB C	91	Subtract C from A
	SUB D	92	Subtract D from A
	SUB E	93	Subtract E from A
	SUB H	94	Subtract H from A
	SUB L	95	Subtract L from A
	SUB M	96	Subtract contents of memory LOC (H & L) from A
	SUI v	D6	Subtract immediate data v from A
	SBB A	9F	Set A to minus carry
	SBB B	98	Subtract B from A with borrow
	SBB C	99	Subtract C from A with borrow
	SBB D	9A	Subtract D from A with borrow
	SBB E	9B	Subtract E from A with borrow
	SBB H	9C	Subtract H from A with borrow
	SBB L	9D	Subtract L from A with borrow
	SBB M	9E	Subtract memory LOC (H & L) from A with borrow
	SBI v	22	Store H & L at memory LOC an
X	XCHG	EB	Exchange D & E with H & L
	XTHL	E3	Exchange top of stack with H & L
	XRAA	AF	Clear A
	XRA B	A8	Exclusive OR B with A
	XRA C	A9	Exclusive OR C with A
	XRA D	AA	Exclusive OR D with A
	XRA E	AD	Exclusive OR E with A
	XRA H	AC	Exclusive OR H with A
	XRA L	AD	Exclusive OR L with A
	XRA M	AE	Exclusive OR memory LOC (H & L) with A
	XRI v	EE	Exclusive OR immediate data v with A

Data Transfer Instructions

- Includes:
  - Data transfer from register to register
  - Data transfer from register to memory
  - Data transfer from memory to register
  - Moving immediate number into register
- Destination and Source operands

Data Transfer Instructions

- MOV r1, r2 (Register)
- MOV r, M (Register Indirect)
- MOV M, r (Register Indirect)
- MVI r, data (Immediate)
- MVI M, data (Immediate / Register Indirect)
- LXI rp, data16 (Immediate)
- LDA addr (Direct)
- STA addr (Direct)
- LHLD addr (Direct)
- SHLD addr (Direct)
- LDAX rp (Register Indirect)
- STAX rp (Register Indirect)
- XCHG (Register)

Arithmetic Instructions

- Add r (Register)
- Add M (Register Indirect)
- ADI data (Immediate)
- ADC r (Register)
- ADC M (Register Indirect)
- ACI data (Immediate)
- SUB r (Register)
- SUB M (Register Indirect)
- SUI data (Immediate)
- SBB r (Register)
- SBB M (Register Indirect)
- SBI data (Immediate)
- INR r (Register)
- INR M (Register Indirect)
- DCR r (Register)
- DCR M ((Register Indirect)
- INX rp (Register)
- DCX rp (Register)
- DAD rp (Register)

Arithmetic Instructions

- Addition and Subtraction operations affects all flags
- Increment and Decrement operations affect all flags except carry (INX and DCX affect none)
- DAD affects only carry flag

Logical Instructions

- ANA r (Register)
- ANA M (Register Indirect)
- ANI data (Immediate)
- XRA r (Register)
- XRA M (Register Indirect)
- XRI data (Immediate)
- ORA r (Register)
- ORA M (Register Indirect)
- ORI data (Immediate)
- CMP r (Register)
- CMP M (Register Indirect)
- CPI data (Immediate)
- RLC
- RRC
- RAL
- RAR
- CMA
- CMC
- STC

Branch Instructions

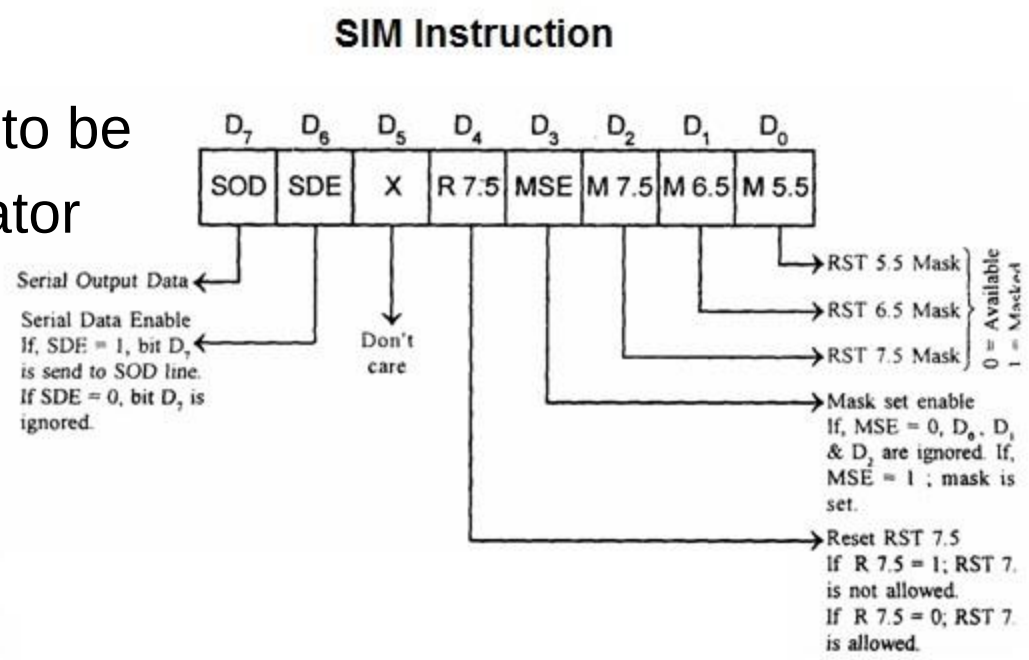
- JMP addr (Immediate)
- Jcondition addr (Immediate)
- CALL addr (Immediate)
- Ccondition addr (Immediate/ Register Indirect)
- RET (Register Indirect)
- Rcondition (Register Indirect)
- RST n (Register Indirect)
- PCHL (Register)

Other Instructions

- PUSH rp (Register Indirect)
- PSUH PSW (Register Indirect)
- POP rp (Register Indirect)
- POP PSW (Register Indirect)
- XTHL (Register Indirect)
- SPPHL (Register)
- IN port (Direct)
- OUT port (Direct)
- EI
- DI
- HLT
- NOP
- RIM
- SIM

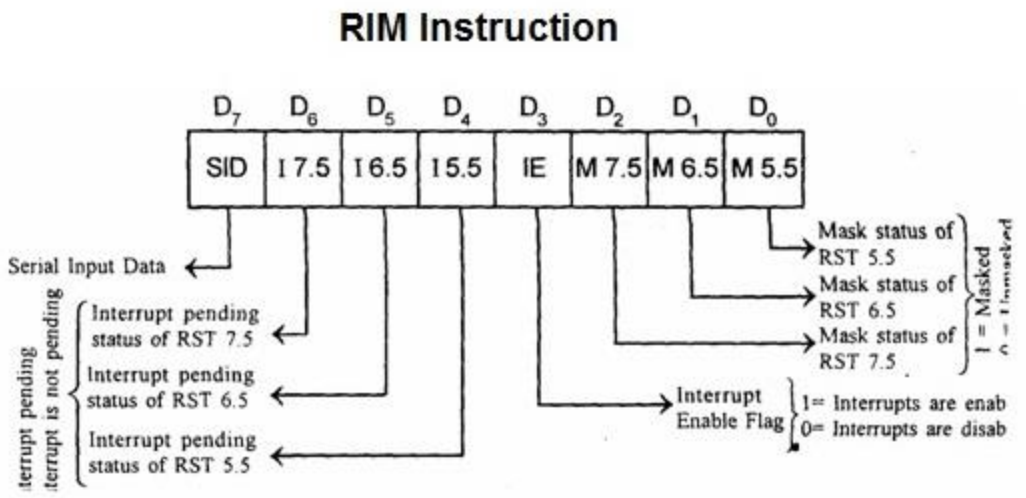
RIM & SIM

- RIM & SIM have two functions:
  - Serial Data
  - Interrupt Masking
- SIM
  - For serial data out
  - It requires the data to be loaded in accumulator



RIM & SIM

- RIM
  - For serial data in
  - It requires the data to be loaded in accumulator



Programming 8085 MPU

- The programming model of any microprocessor is:
  - Instruction set
  - Register set
  - Internal Architecture
  - Memory Organization
- We will use assembly language to program 8085 processor
- An assembly language program consists of instruction (usually one per line) each having at most 4 fields
  - **Label**: Its optional and used to identify instruction line
  - **Op code**: Always present. Represents the operation to be performed
  - **Operand**: Present most of the time. Represents values
  - **Comments**: Optional. Gives additional information about the instruction

Programming 8085 MPU (Contd.)

- Consider the following example of an assembly language instruction

```
DATA:     MOV A, M     ;Input data to accumulator
ADD B     ;Add accumulator with B
MVI B, 01h  ;Initialize the register B with 01h
XRA A     ;Clear Accumulator
JMP LOOP  ;Make a jump
```

Programming 8085 MPU (Contd.)

- Flowchart
  - Graphical Representation of the program using special shape symbols
- Source Program
  - Listing of assembly language instructions
- Assembler
  - A program that lets you write the program in assembly language and then converts it into sequence of machine codes

Programming 8085 MPU (Contd.)

- Program (Converting ASCII to BCD)
- Program:

```
IN 01h          ; Input ASCII data from port 01h
ANI 0Fh         ; AND with 0Fh to make BCD
STA 2040h       ; Store the result in memory (2040h)
HLT             ; Stop
```

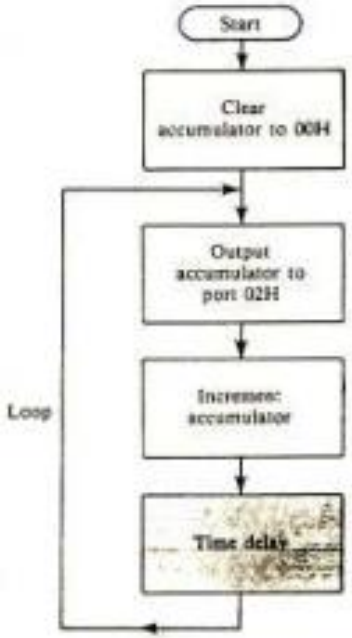
Programming 8085 MPU (Contd.)

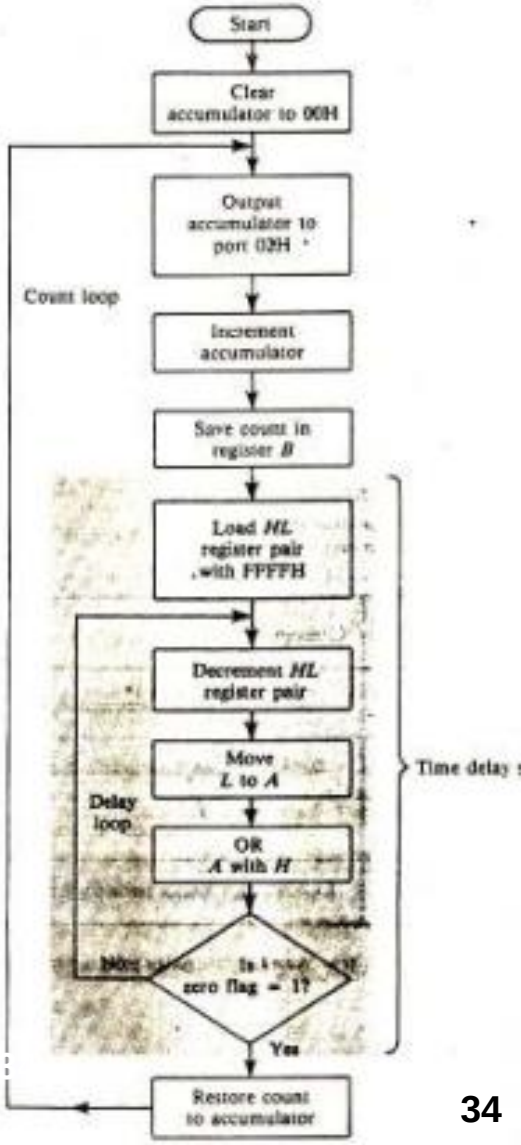
- Program (Display Binary Count to port)
- Program code:

```
XRA A              ; Clear Accumulator
LOOP: OUT 02h      ; Output acc. To port 02h
INR A              ; Increment accumulator
JMP LOOP           ; Jump back to repeat
```

Programming 8085 MPU (Contd.)

- Program (Display Binary Count to port) Modified
- Program counts and displays so fast
- A delay routine provides appropriate time interval for a count to be seen and observed





Program code:

```
XRA A                ; Clear Accumulator
COUNT: OUT 02h       ; Output acc. To port 02h
INR A                ; Increment accumulator
MOV B, A             ; Store count an B Register
LXI H, FFFFh         ; Initialize HL pair with a big number
DELAY: DCX H         ; Decrement HL by 1
MOV A, L             ; Move L to A
ORA H                ; Logically OR H and L to see if HL=0?
JNZ DELAY            ; If not zero then repeat with delay
MOV A, B             ; If yes then repeat next count
JMP COUNT                    ;
```

Programming 8085 MPU (Contd.)

- Program (Adding two 3-byte numbers)
- The accumulator can only handle 8-bit data
- In order to add 24-bit numbers, we first divide them into 8-bit groups and then add them starting from LSB (recording carry)
- The 8085 has special instruction “Add with Carry” to handle such situation
- The procedure
  - Add the LSB of both numbers and record the sum
  - Add middle byte of 1st number, middle byte of 2nd numbers and the carry generated in step 1. Store the partial sum
  - Add the MSB of 1st number, MSB of 2nd number and the previous carry and record the sum

Program Code

```
LXI H 2020H     ; Initialize HL register pair
MOV A, M        ; Move LSB of first number into accumulator
INX H           ; Increment pointer
ADD M           ; Add LSBs of both numbers
STA 2026H       ; Store the LSB of sum at memory location 2026h
INX H           ; Increment memory pointer (2022h)
MOV A, M        ; Move the middle byte of first number into accumulator
INX H           ; Increment memory pointer
ADC M           ; Add the middle bytes with carry
STA 2027H       ; Store the sum of middle bytes at memory location 2027h
INX H           ; Increment the memory pointer
MOV A, M        ; Move the MSB of first number into accumulator
INX H           ; Increment memory pointer
ADC M           ; Add the MSB of both numbers with carry
STA 2028        ; Store the result of addition at memory location 202h
HLT             ; Stop
```

Questions