Basics of 8085 Microprocessor

Microprocessor

- With the advent of LSI and VLSI technology it became possible to build the entire CPU on a single chip IC
- A CPU built into a single LSI/VLSI chip is called a microprocessor
- A digital computer using microprocessor as its CPU is called a microcomputer

Microprocessor

- The term micro initiates its physical size; not it’s computing power
- Today the computing power of a powerful microprocessor approaches that a CPU on earlier large computer
- The main sections of a microprocessor are: ALU, timing and control unit, accumulator, general purpose and special purpose registers

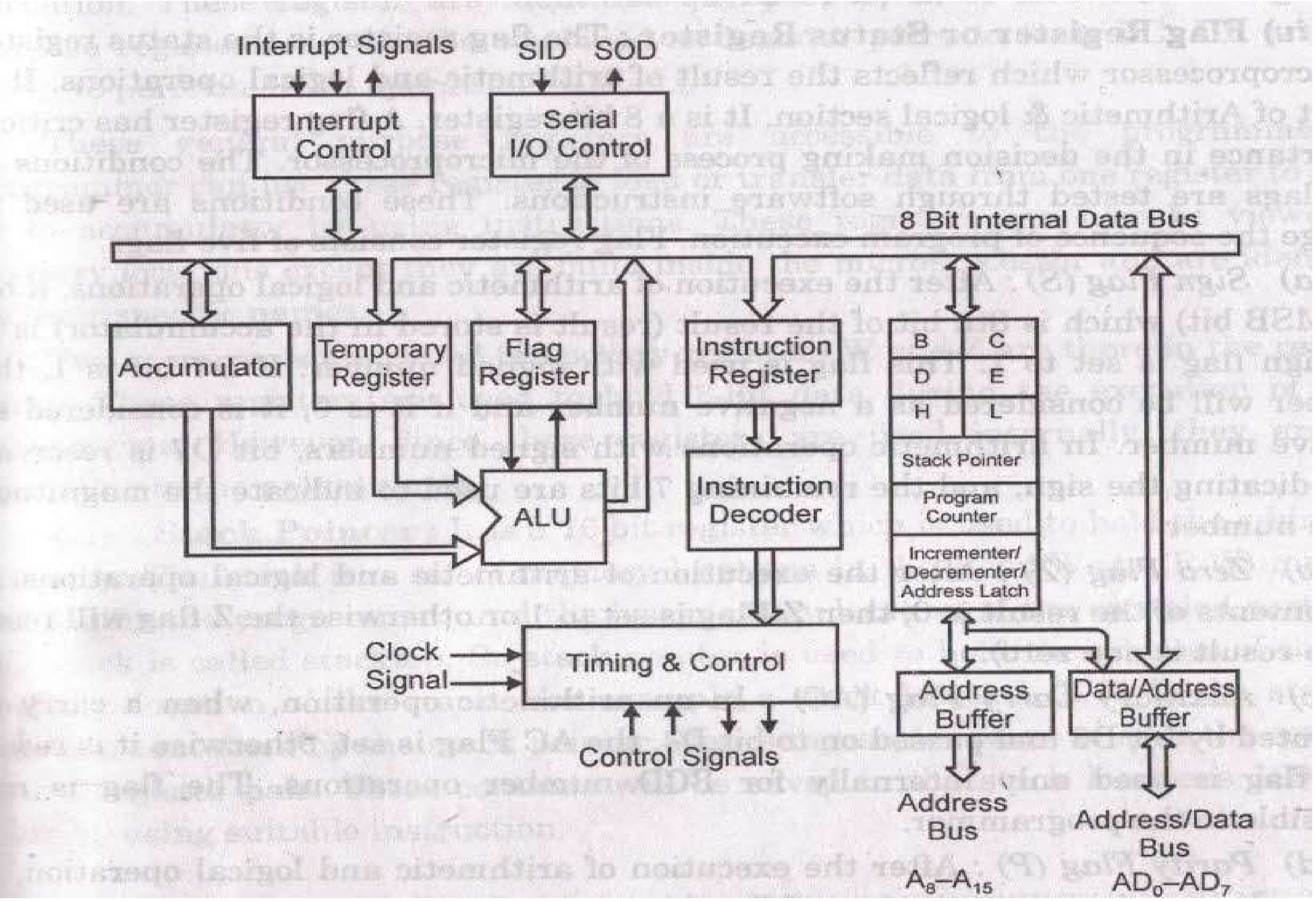
History of 8085

1971 –Intel 4004	- 4 bit µp
1972 –Intel 8008	- 8 bit µp
1973 –Intel 8080	- 8 bit µp
1974 –Motorolla 6800	- 8 bit µp
1976 –Zilog 80	- 8 bit µp
1976 –Intel 8085	- 8 bit µp

8085 Microprocessor

- Intel 8085 is an 8-bit, N-channel Metal Oxide semiconductor (NMOS) microprocessor
- It is a 40 pin IC package fabricated on a single Large Scale Integration (LSI) chip
- The Intel 8085 uses a single +5V DC supply for its operation
- Its clock speed is about 3MHz
- The clock cycle is of 320 ns
- The time for the clock cycle of the Intel 8085 is 200 ns
- It has 80 basic instructions and 246 opcodes

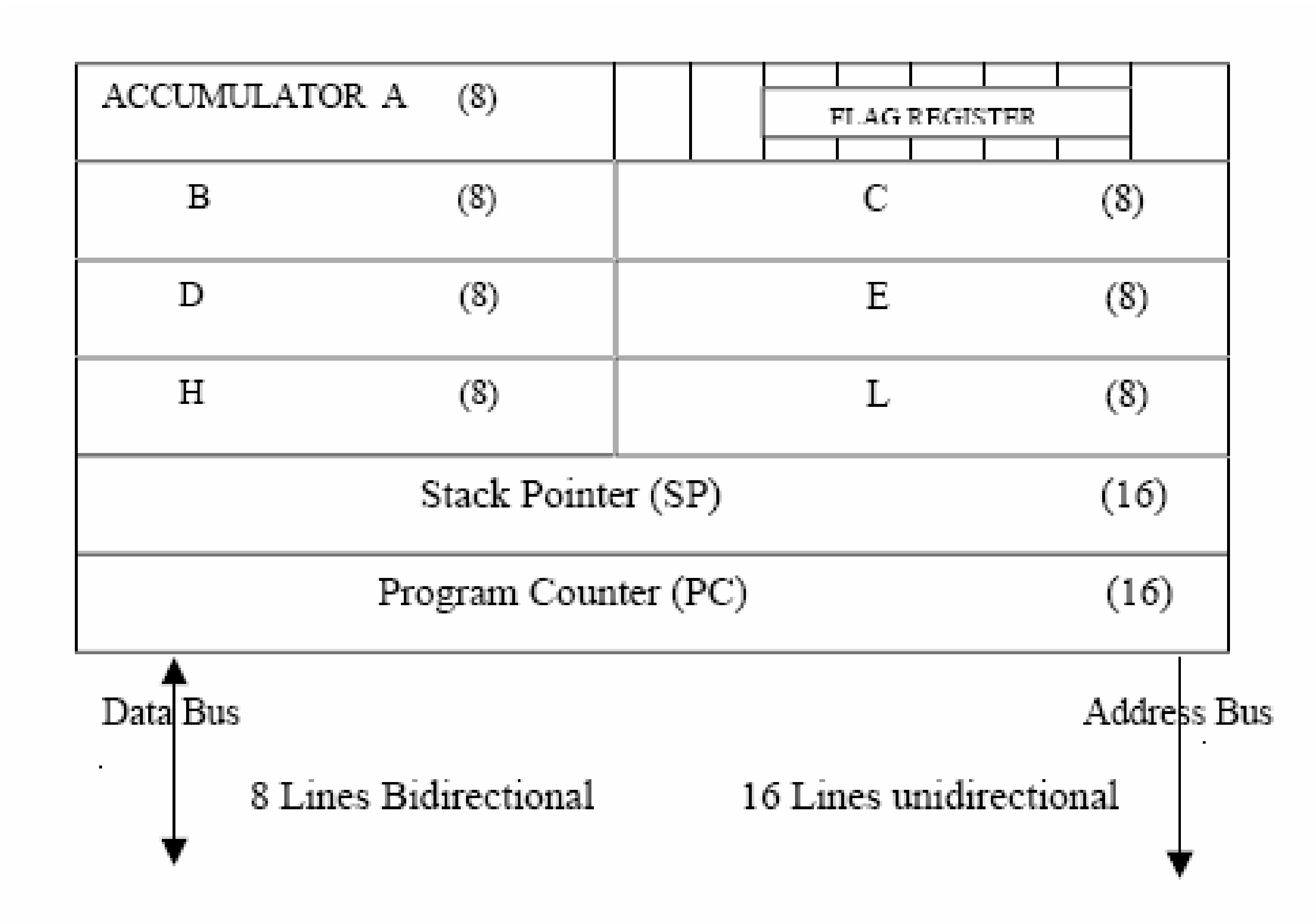
8085 Architecture



ALU

- The ALU performs the following arithmetic and logical operations.
  - Addition
  - Subtraction
  - Logical AND
  - Logical OR
  - Logical EXCLUSIVE OR
  - Complement (logical NOT)
  - Increment (add 1)
  - Decrement (subtract 1)
  - Left shift
  - Clear

Register Set



ACCUMULATOR A (8)

FLAG REGISTER

B (8)

C (8)

D (8)

E (8)

H (8)

L (8)

Stack Pointer (SP) (16)

Program Counter (PC) (16)

Data Bus

8 Lines Bidirectional

Address Bus

16 Lines unidirectional

General Registers

- 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
- The programmer can use these registers to store or copy data into the registers by using data copy instructions
- The HL register pair is also used to address memory locations
- In other words, HL register pair plays the role of memory address register

Accumulator & Pointers

- The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU)
- Program Counter – store address of next instruction to be executed.
- Stack Pointer – store the address of stacktop, the last filled location of a Stack.

Instruction Register/Decoder

- The instruction register and the decoder are considered as a part of the ALU
- The instruction register is a temporary storage for the current instruction of a program
- The decoder decodes the instruction and establishes the sequence of events to follow

Flags

- 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

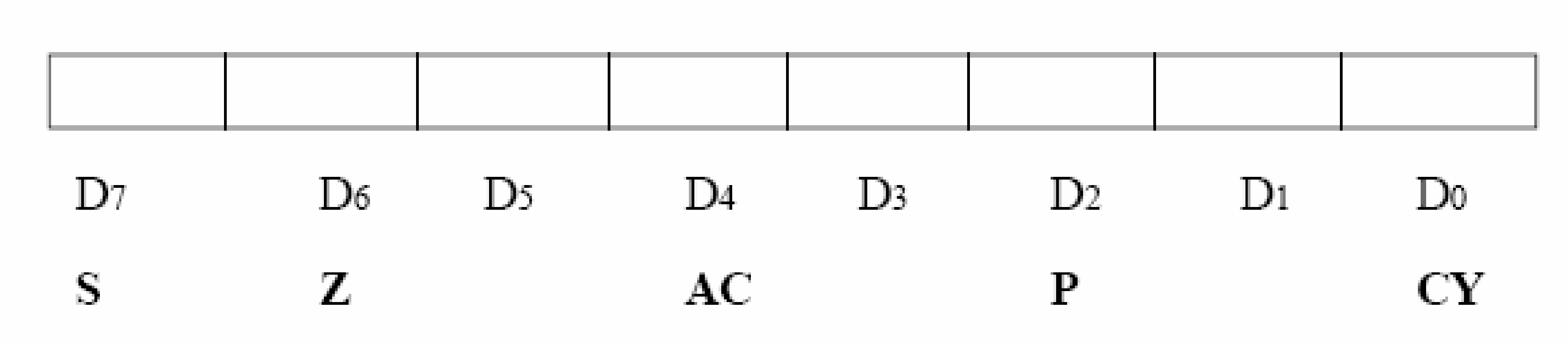
Flags

- If the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) – is set to one
- When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one

Flags

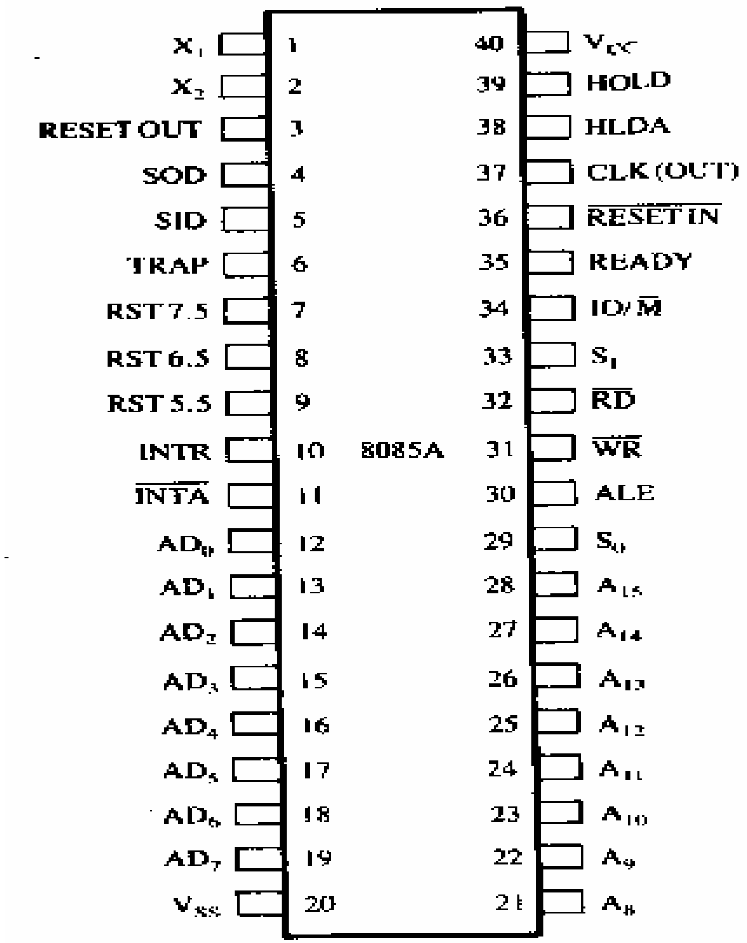
- These flags have critical importance in the decision-making process of the microprocessor
- The conditions (set or reset) of the flags are tested through the software instructions
- The thorough understanding of flag is essential in writing assembly language programs
- The combination of the flag register and the accumulator is called Program Status Word (PSW) and PSW is the 16-bit unit for stack operation

Flags



D7	D6	D5	D4	D3	D2	D1	D0
S	**Z**		**AC**		**P**		**CY**

Pin Diagram



8085 Pinout

Address & Data Bus

- Address Bus
- The 8085 has eight signal lines, A15-A8, which are unidirectional and used as the high order address bus
- Multiplexed Address/Data Bus
- The signal lines AD7-AD0 are bidirectional
- They serve a dual purpose

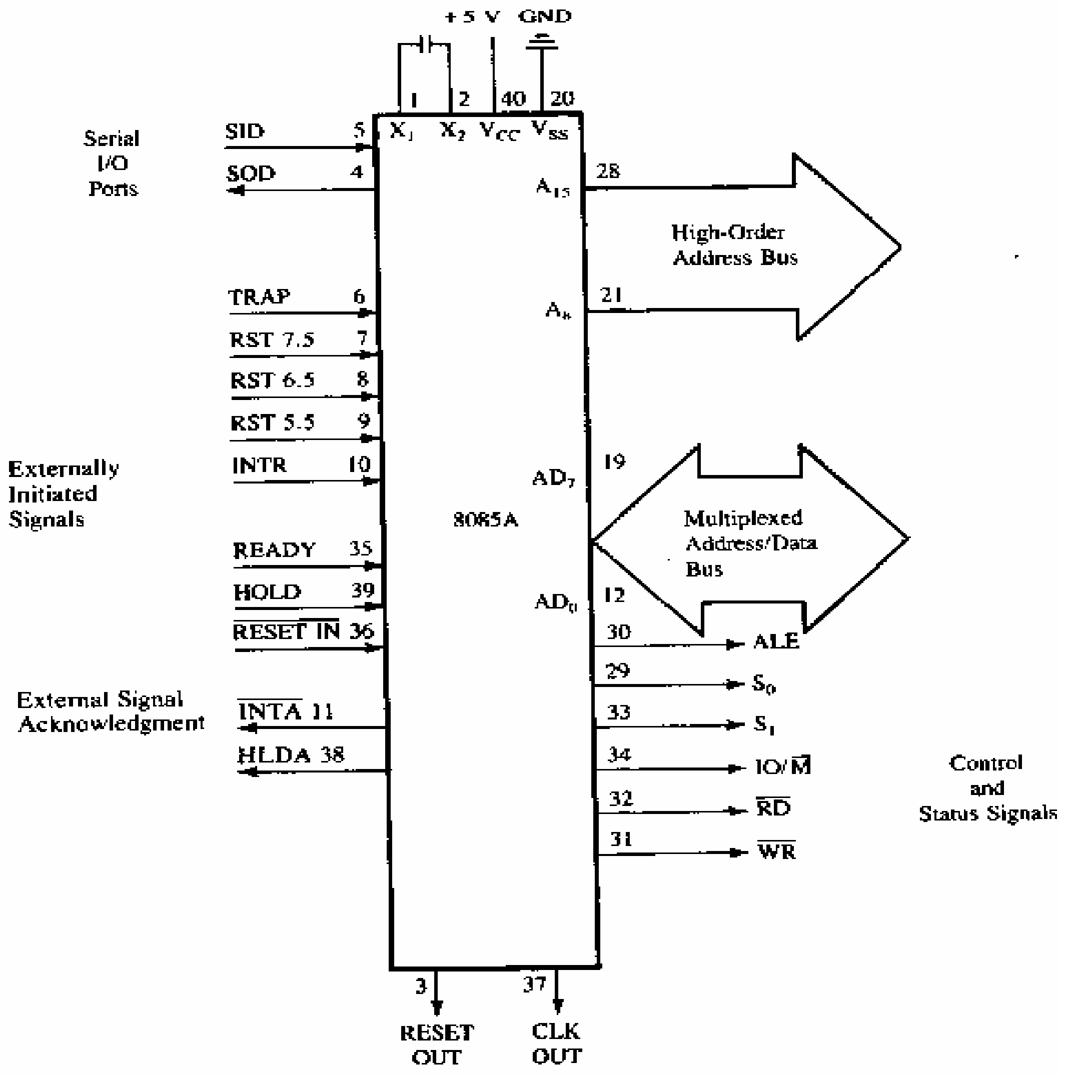
Address & Data Bus

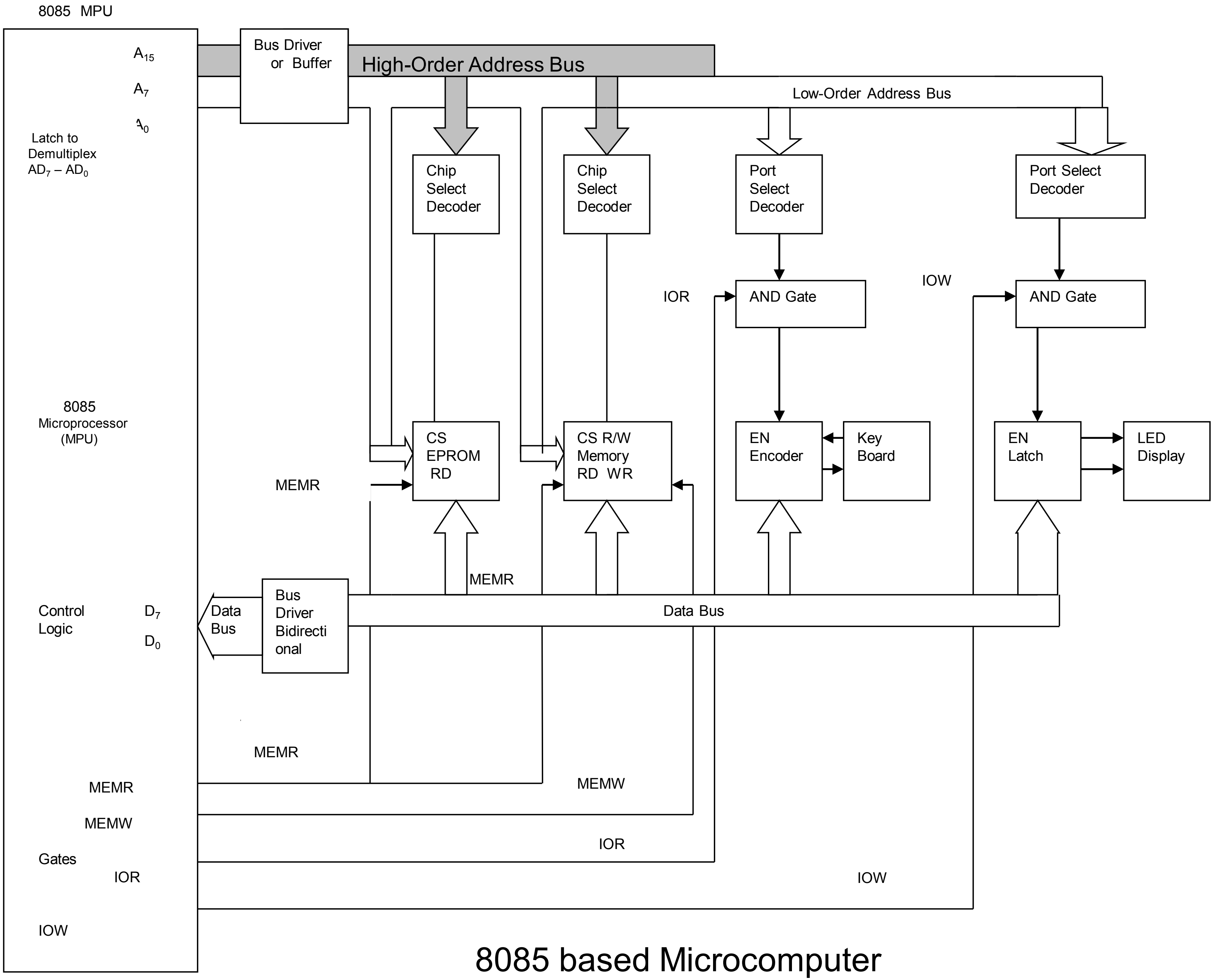
- They are used as the low-order address bus as well as the data bus
- In executing an instruction, during the earlier part of the cycle, these lines are used as the low-order address bus as well as the data bus
- During the later part of the cycle, these lines are used as the data bus
- However the low order address bus can be separated from these signals by using a latch

Control and Status Signals

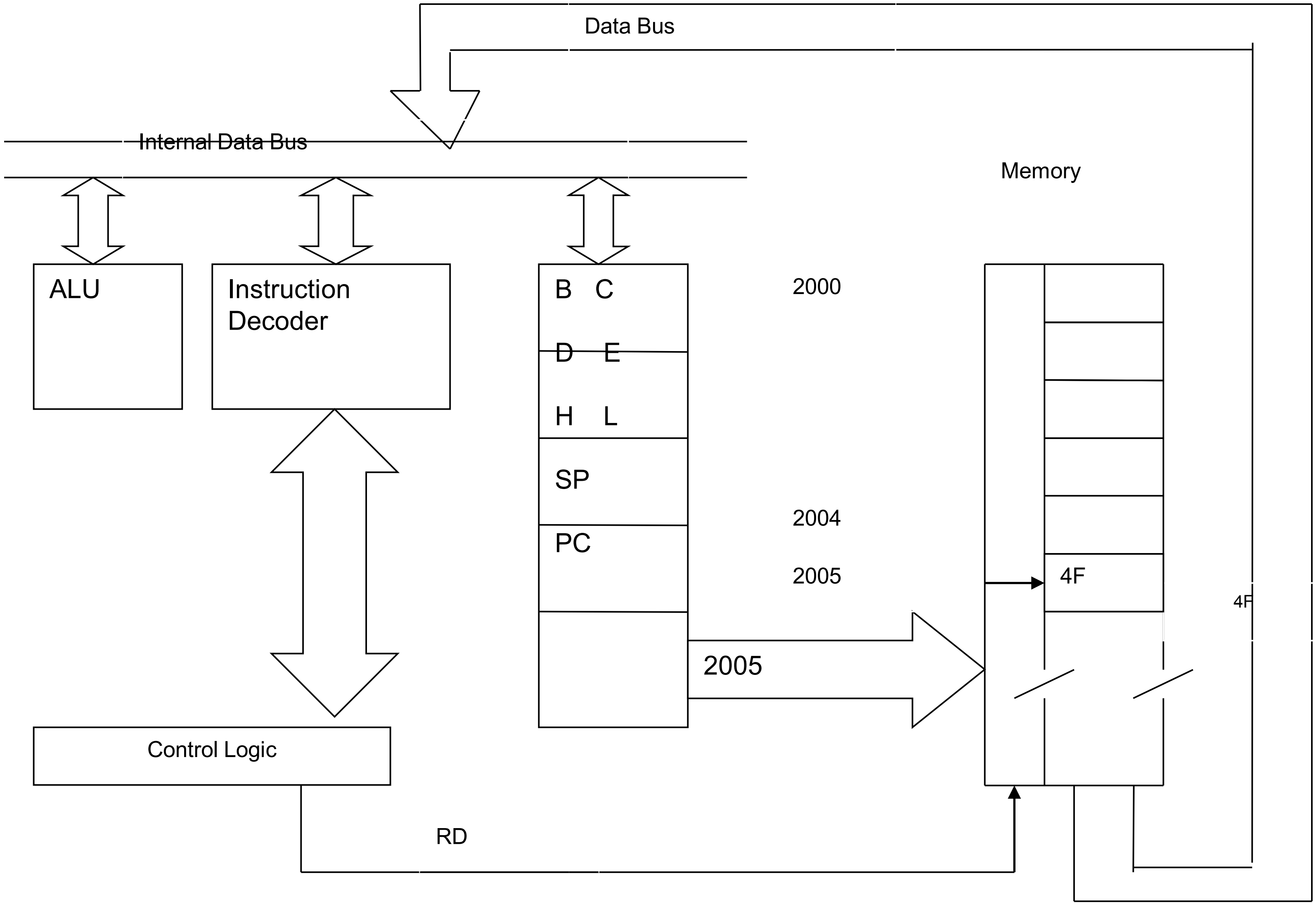
Machine Cycle	IO/M	S1	S0	Control signals
Opcode Fetch	0	1	1	RD=0
Memory Read	0	1	0	RD=0
Memory Write	0	0	1	WR=0
I/O Read	1	1	0	RD=0
I/O Write	1	0	1	WR=0
Interrupt Acknowledge	1	1	1	INTA=0
Halt	Z	0	0	RD, WR=z and INTA=1
Hold	Z	X	X	RD, WR=z and INTA=1
Reset	Z	X	X	RD, WR=z and INTA=1

Functional Description



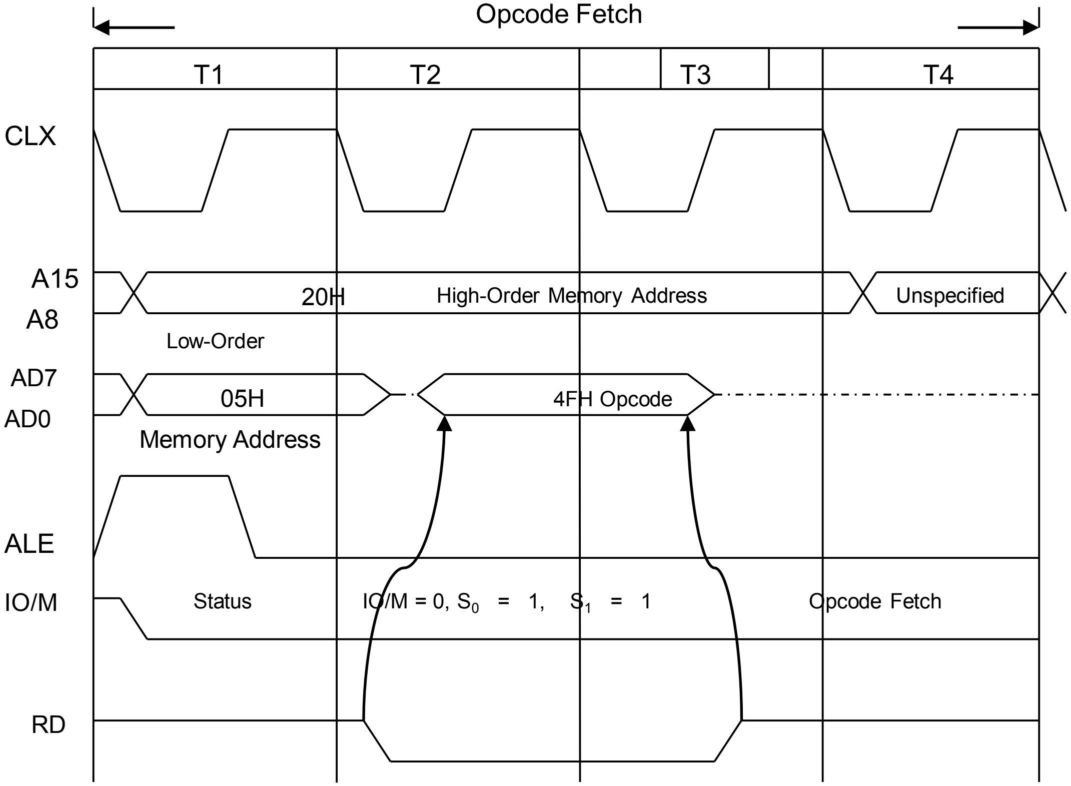


8085 based Microcomputer

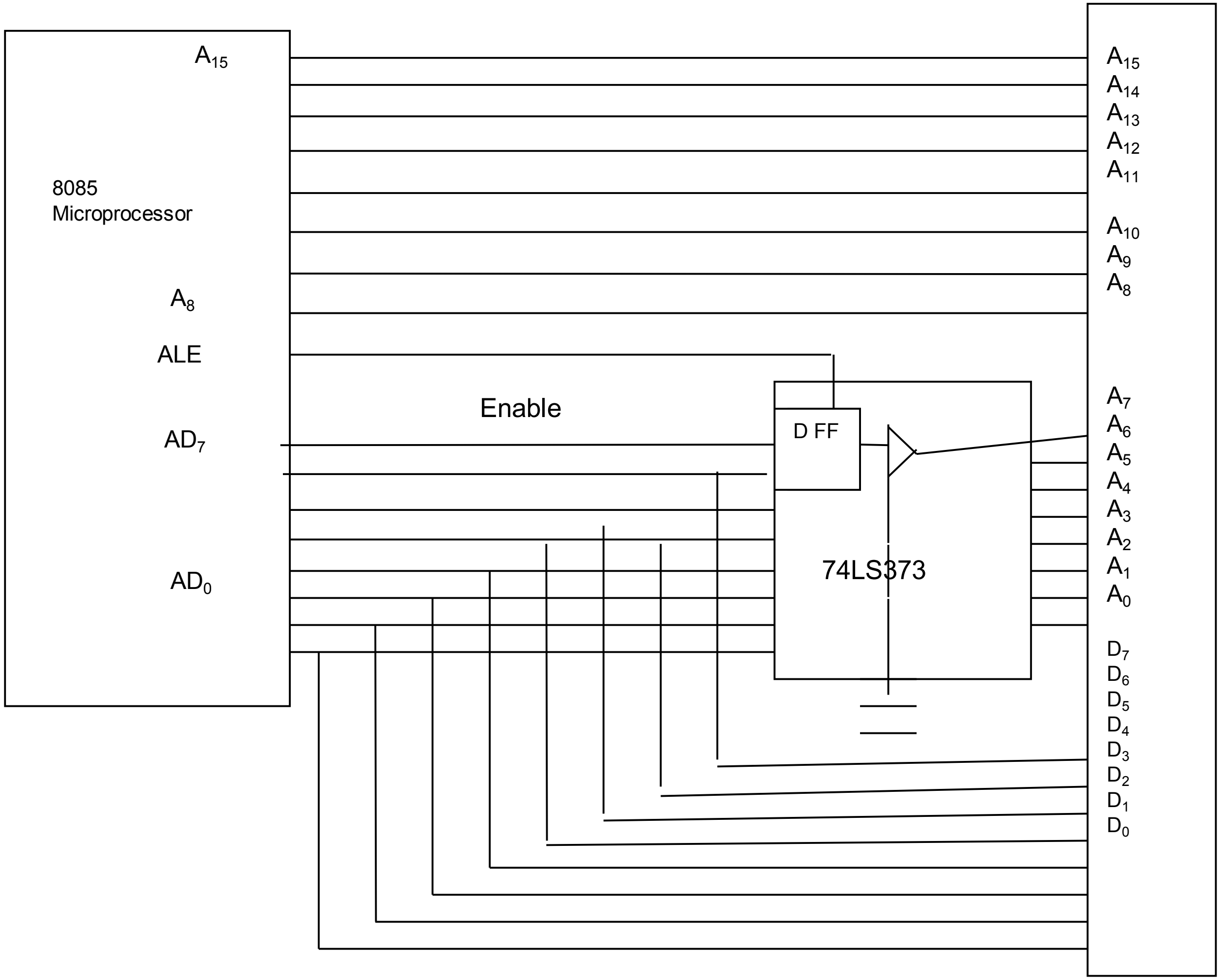

Data Bus
Internal Data Bus
Memory
ALU
Instruction Decoder
B C
D E
H L
SP
PC
2000
2004
2005
4F
4F
2005
Control Logic
RD

Data Flow

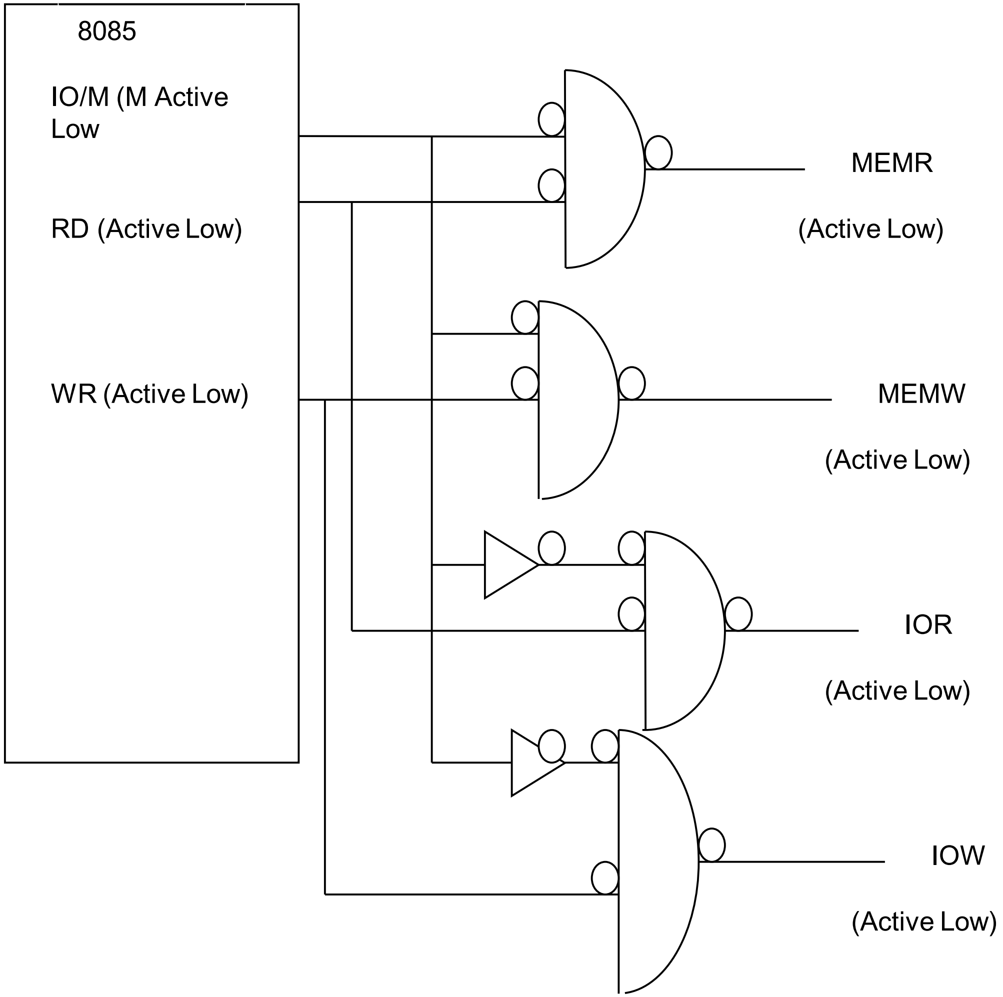
Timing Diagram of Read Cycle



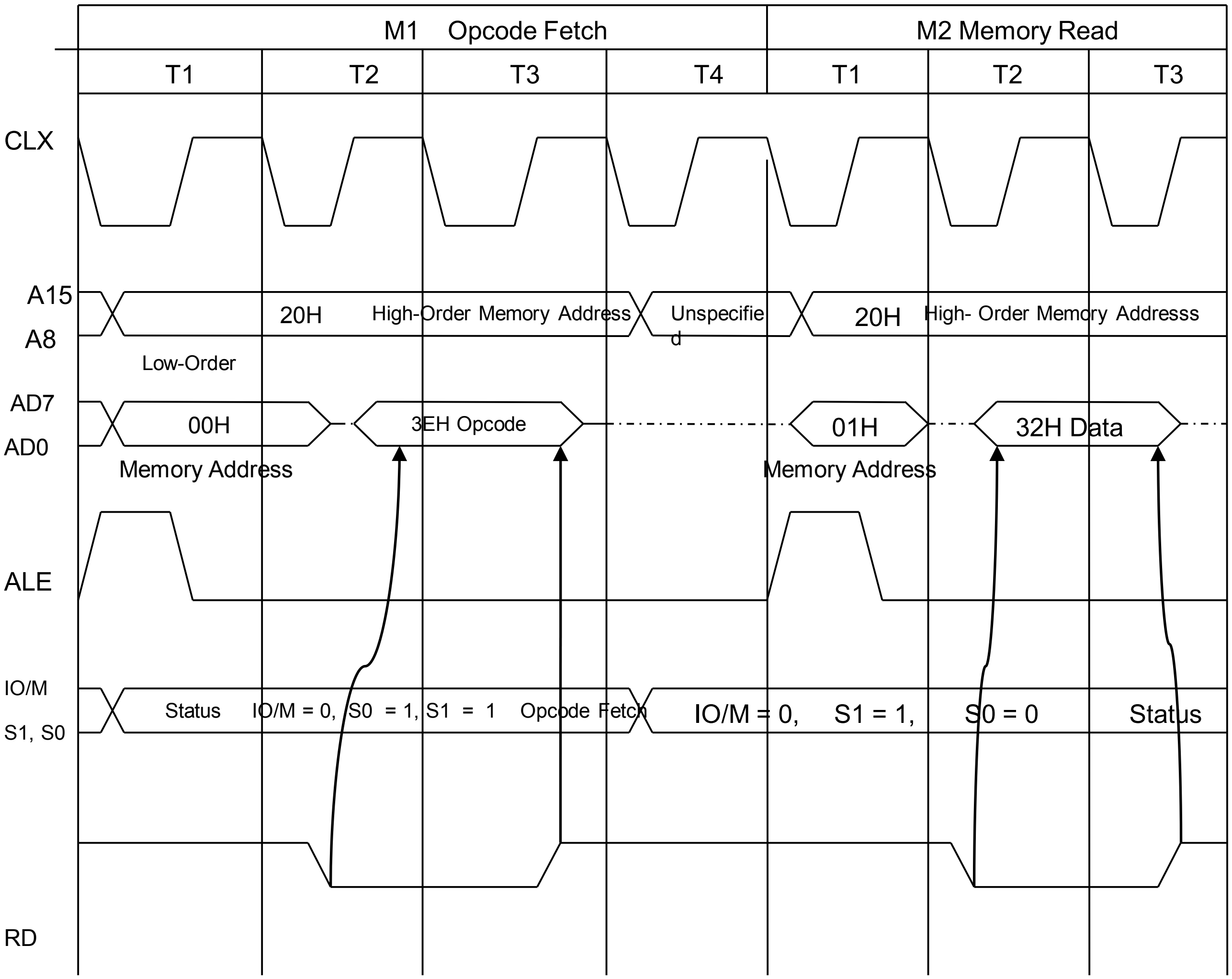
Demultiplexing the bus AD7-AD0



Schematic to generate Control Signals



Timing for Execution of the Instruction MVI A,32H



Addressing Modes

- Various ways of specifying the operands or various formats for specifying the operands is called addressing mode
- 8-bit or 16-bit data may be directly given in the instruction itself
- The address of the memory location, I/O port or I/O device, where data resides, may be given in the instruction itself
- In some instructions only one register is specified. The content of the specified register is one of the operands. It is understood that the other operand is in the accumulator.

Addressing Modes

- Some instructions specify one or two registers. The contents of the registers are the required data.
- In some instructions data is implied. The most instructions of this type operate on the content of the accumulator.

Addressing Modes

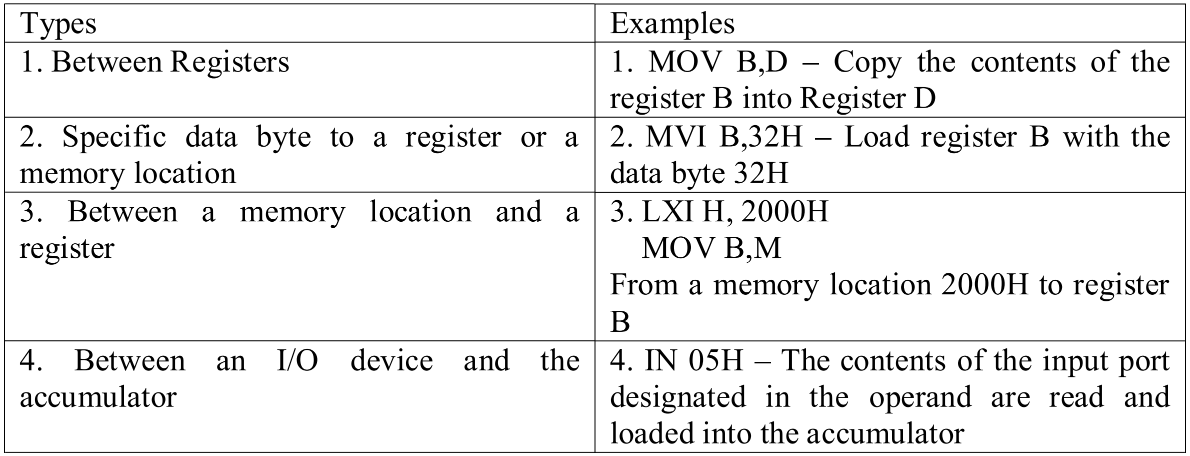
- Implicit addressing
  - CMA – Complement the contents of accumulator
- Immediate addressing
  - MVI R, 05H
  - ADI 06H
- Direct addressing – The address of the operand in the instruction - STA 2400H, IN 02H

Addressing Modes

- Register addressing
  - In register addressing mode the operands are in the general purpose registers
  - MOV A, B
  - ADD B
- Register indirect addressing
  - Memory location is specified by the contents of the registers
  - LDAX B, STAX D

Different Types of Instruction Sets

Data Transfer Instructions



Types	Examples
1. Between Registers	1. MOV B,D – Copy the contents of the register B into Register D
2. Specific data byte to a register or a memory location	2. MVI B,32H – Load register B with the data byte 32H
3. Between a memory location and a register	3. LXI H, 2000H MOV B,M From a memory location 2000H to register B
4. Between an I/O device and the accumulator	4. IN 05H – The contents of the input port designated in the operand are read and loaded into the accumulator

Arithmetic Instructions

- ADD B – [A] <----- [A]+[B]
- ADD M - [A] <----- [A]+[[HL]]
- DAD B – [HL] <----- [HL]+[BC]
- SUB C – [A] <----- [A]+[C]
- SUI 76H – [A] <---- [A]-76H
- SBB M – [A] <----- [A]-[[HL]]-[C]

Logical Instructions

- ANA C – [A] <----- [A] ^ [C]
- ANI 85H – [A] <----- [A] ^ 85H
- ORA M – [A] <----- [A] v [[HL]]
- XRA B – [A] <------ [A] XOR [B]

Rotate Instructions

- RLC
  - [An+1] <----- [An]
  - [A0] <------ [A7]
  - [CS] <----- [A7]
- RAR
  - [An] <------ [An+1]
  - [CS] <------ [A0]
  - [A7] <------ [CS]

Complement Instructions

- CMP R

- CPI data

Complement Instructions

- CMA – [A] <---- [A]’
- CMC – [CS] <----- [CS]’

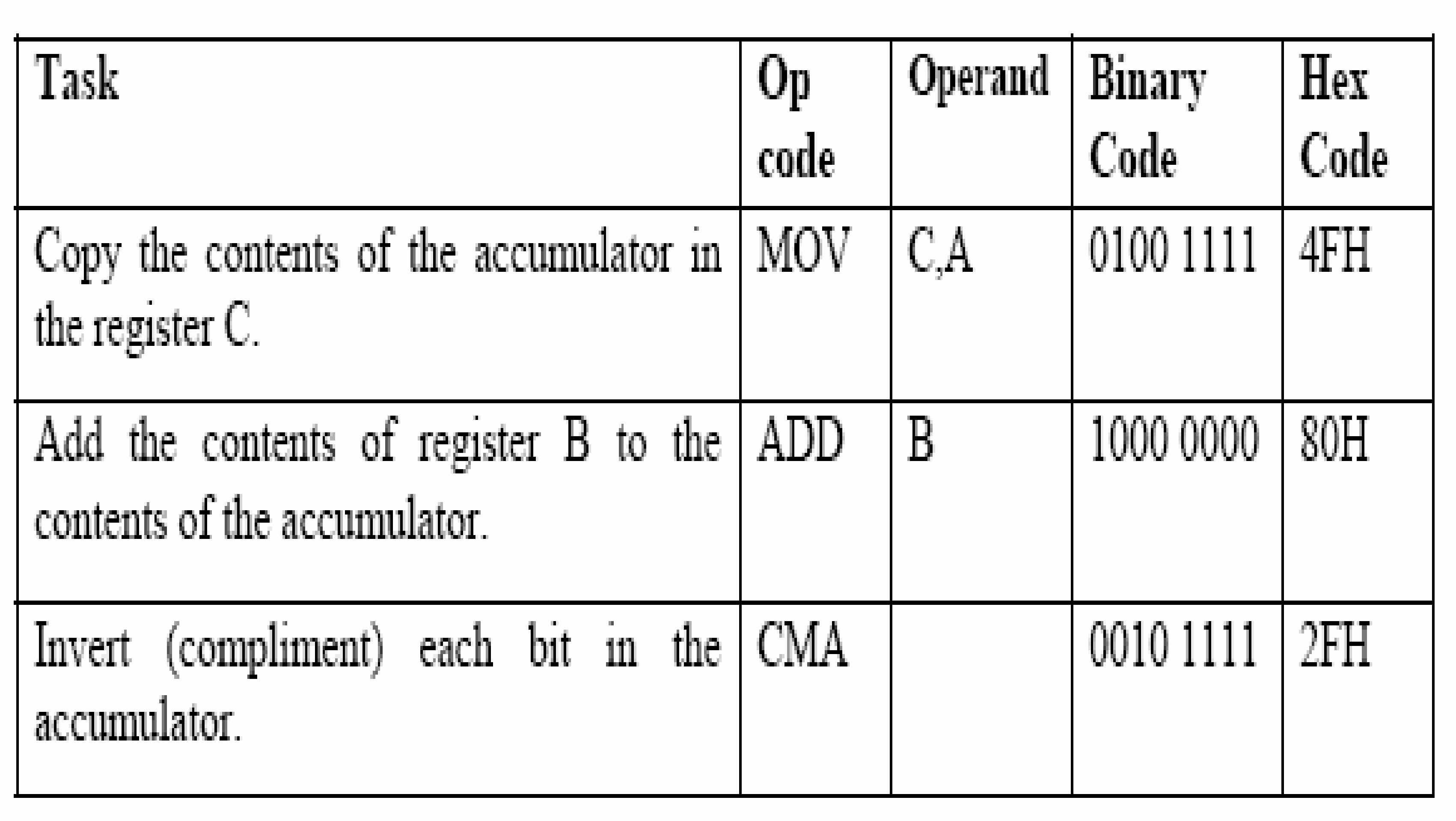
Transfer Instructions

- JMP 2050H – [PC] <----- 2050H
- JZ 3100H – [PC] <----- 3100H if Z=1, otherwise [PC] <----- [PC]+1
- JNC 4250H – [PC] <----- 4250H if C=0, otherwise [PC] <----- [PC]+1

CALL & RET

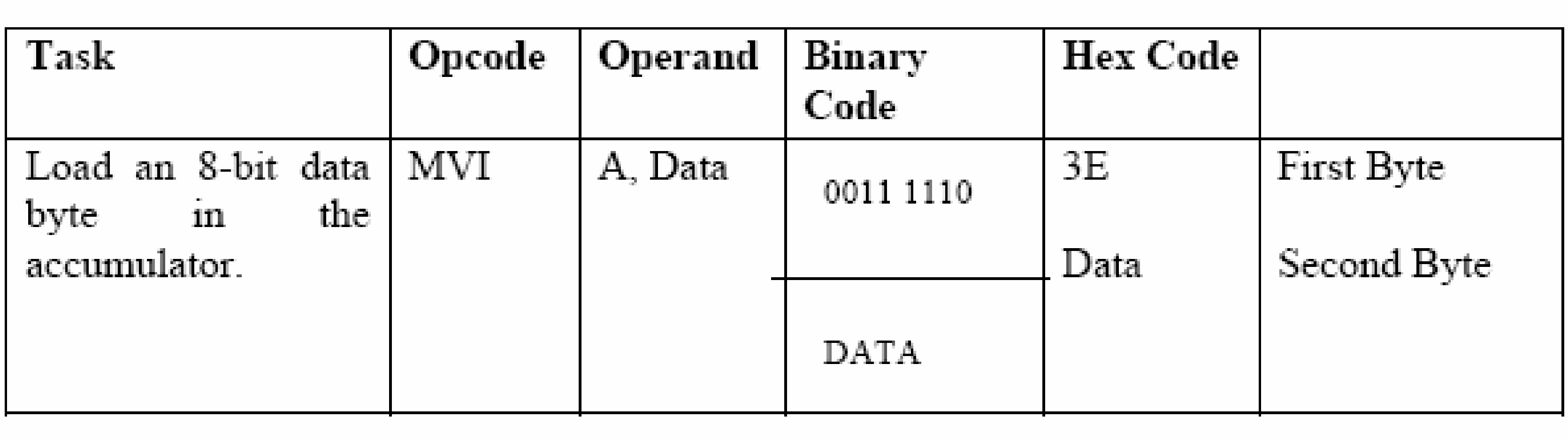
- CALL Addr
- [[SP]-1] <------- [PCH]
- [[SP]-1] <------- [PCL]
- [SP] <----- [SP]-2
- [PC] <----- Addr
- RET
- [PCL] <------ [[SP]]
- [PCH] <------ [[SP]+1]
- [SP] <------ [SP]+2

One Byte Instructions



Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

Two Byte Instructions



Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110 DATA	3E Data	First Byte Second Byte

3 Byte Instruction

Task	Opcode	operand	Binary Code	Hex Code	Bytes
Load an 8-bit data from memory location to accumulator	MVI	A,Address	0011 1110	3E	1st byte
			Address		2nd and 3rd bytes

Writing Assembly Language Program

- Define the problem clearly and make the problem statement.
- Analyze the problem thoroughly. In this step we divide the problem into smaller steps to examine the process of writing programs.
- Draw the flow chart. The steps listed in the problem analysis and the sequences are represented in a block diagram.
- Translate the blocks shown in the flowchart into 8085 operations and then subsequently into mnemonics.

Conversion and Execution

- Convert the mnemonics into Hex code; we need to look up the code in 8085 instruction set.
- Store the program in Read/Write memory of a single-board microcomputer. This may require the knowledge about memory addresses and the output port addresses.
- Finally execute the program.

INSTRUCTION SET OF 8085

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called ***Instruction Set***.
- 8085 has **246** instructions.
- Each instruction is represented by an 8-bit binary value.
- These 8-bits of binary value is called ***Op-Code*** or ***Instruction Byte***.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instructions

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination.
- While copying, the contents of source are not modified.

Data Transfer Instructions

Opcode	Operand	Description
MOV	Rd, Rs M, Rs Rd, M	Copy from source to destination.

- This instruction copies the contents of the source register into the destination register.

- The contents of the source register are not altered.

- If one of the operands is a memory location, its location is specified by the contents of the HL registers.

- **Example:** MOV B, C or MOV B, M

Data Transfer Instructions

Opcode	Operand	Description
MVI	Rd, Data M, Data	Move immediate 8-bit

- The 8-bit data is stored in the destination register or memory.
- If the operand is a memory location, its location is specified by the contents of the H-L registers.
- **Example:** MVI B, 57H or MVI M, 57H

Data Transfer Instructions

Opcode	Operand	Description
LDA	16-bit address	Load Accumulator

- The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.
- The contents of the source are not altered.
- **Example:** LDA 2034H

Data Transfer Instructions

Opcode	Operand	Description
LDAX	B/D Register Pair	Load accumulator indirect

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.
- **Example:** LDAX B

Data Transfer Instructions

Opcode	Operand	Description
LXI	Reg. pair, 16-bit data	Load register pair immediate

- This instruction loads 16-bit data in the register pair.
- **Example:** LXI H, 2034 H

Data Transfer Instructions

Opcode	Operand	Description
LHLD	16-bit address	Load H-L registers direct

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.
- **Example:** LHLD 2040 H

Data Transfer Instructions

Opcode	Operand	Description
STA	16-bit address	Store accumulator direct

- The contents of accumulator are copied into the memory location specified by the operand.
- **Example:** STA 2500 H

Data Transfer Instructions

Opcode	Operand	Description
STAX	Reg. pair	Store accumulator indirect

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.
- **Example:** STAX B

Data Transfer Instructions

Opcode	Operand	Description
SHLD	16-bit address	Store H-L registers direct

- The contents of register L are stored into memory location specified by the 16-bit address.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550 H

Data Transfer Instructions

Opcode	Operand	Description
XCHG	None	Exchange H-L with D-E

- The contents of register H are exchanged with the contents of register D.

- The contents of register L are exchanged with the contents of register E.

- **Example:** XCHG

Data Transfer Instructions

Opcode	Operand	Description
SPHL	None	Copy H-L pair to the Stack Pointer (SP)

- This instruction loads the contents of H-L pair into SP.
- **Example:** SPHL

Data Transfer Instructions

Opcode	Operand	Description
XTHL	None	Exchange H–L with top of stack

- The contents of L register are exchanged with the location pointed out by the contents of the SP.

- The contents of H register are exchanged with the next location (SP + 1).

- **Example:** XTHL

Data Transfer Instructions

Opcode	Operand	Description
PCHL	None	Load program counter with H-L contents

- The contents of registers H and L are copied into the program counter (PC).
- The contents of H are placed as the high-order byte and the contents of L as the low-order byte.
- **Example:** PCHL

Data Transfer Instructions

Opcode	Operand	Description
PUSH	Reg. pair	Push register pair onto stack

- The contents of register pair are copied onto stack.
- SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
- SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.
- **Example:** PUSH B

Data Transfer Instructions

Opcode	Operand	Description
POP	Reg. pair	Pop stack to register pair

- The contents of top of stack are copied into register pair.
- The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
- SP is incremented and the contents of location are copied to the high-order register (B, D, H, A).
- **Example:** POP H

Data Transfer Instructions

Opcode	Operand	Description
OUT	8-bit port address	Copy data from accumulator to a port with 8-bit address

- The contents of accumulator are copied into the I/O port.
- **Example:** OUT 78 H

Data Transfer Instructions

Opcode	Operand	Description
IN	8-bit port address	Copy data to accumulator from a port with 8-bit address

- The contents of I/O port are copied into accumulator.
- **Example:** IN 8C H

Arithmetic Instructions

- These instructions perform the operations like:
  - Addition
  - Subtract
  - Increment
  - Decrement

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.
- **Example:** The contents of register B cannot be added directly to the contents of register C.

Subtraction

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2’s complement form.
- If the result is negative, it is stored in 2’s complement form.
- No two other 8-bit 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.
- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Arithmetic Instructions

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- The contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADD B or ADD M

Arithmetic Instructions

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.
- **Example:** ADC B or ADC M

Arithmetic Instructions

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ADI 45 H

Arithmetic Instructions

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.
- **Example:** ACI 45 H

Arithmetic Instructions

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set.
- No other flags are changed.
- **Example:** DAD B

Arithmetic Instructions

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUB B or SUB M

Arithmetic Instructions

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBB B or SBB M

Arithmetic Instructions

Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SUI 45 H

Arithmetic Instructions

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.
- **Example:** SBI 45 H

Arithmetic Instructions

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** INR B or INR M

Arithmetic Instructions

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.
- **Example:** INX H

Arithmetic Instructions

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- **Example:** DCR B or DCR M

Arithmetic Instructions

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- **Example:** DCX H

Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.

- The logical operations are:
  - AND
  - OR
  - XOR
  - Rotate
  - Compare
  - Complement

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

  with the contents of accumulator.
- The result is stored in accumulator.

Rotate

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

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
  - Equality
  - Greater Than
  - Less Than

  with the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- The contents of the operand (register or memory) are compared with the contents of the accumulator.
- Both contents are preserved .
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- **Example:** CMP B or CMP M

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- **Example:** CPI 89H

Logical Instructions

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY is reset and AC is set.
- **Example:** ANA B or ANA M.

Logical Instructions

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY is reset, AC is set.
- **Example:** ANI 86H.

Logical Instructions

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORA B or ORA M.

Logical Instructions

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically ORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** ORI 86H.

Logical Instructions

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- CY and AC are reset.
- **Example:** XRA B or XRA M.

Logical Instructions

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- CY and AC are reset.
- **Example:** XRI 86H.

Logical Instructions

Opcode	Operand	Description
RLC	None	Rotate accumulator left

- Each binary bit of the accumulator is rotated left by one position.
- Bit D7 is placed in the position of D0 as well as in the Carry flag.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RLC.

Logical Instructions

Opcode	Operand	Description
RRC	None	Rotate accumulator right

- Each binary bit of the accumulator is rotated right by one position.
- Bit D0 is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RRC.

Logical Instructions

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

Logical Instructions

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RAR.

Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example:** CMA.

Logical Instructions

Opcode	Operand	Description
CMC	None	Complement carry

- The Carry flag is complemented.
- No other flags are affected.
- **Example:** CMC.

Logical Instructions

Opcode	Operand	Description
STC	None	Set carry

- The Carry flag is set to 1.
- No other flags are affected.
- **Example:** STC.

Branching Instructions

- The branching instruction alter the normal sequential flow.

- These instructions alter either unconditionally or conditionally.

Branching Instructions

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- **Example:** JMP 2034 H.

Branching Instructions

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0