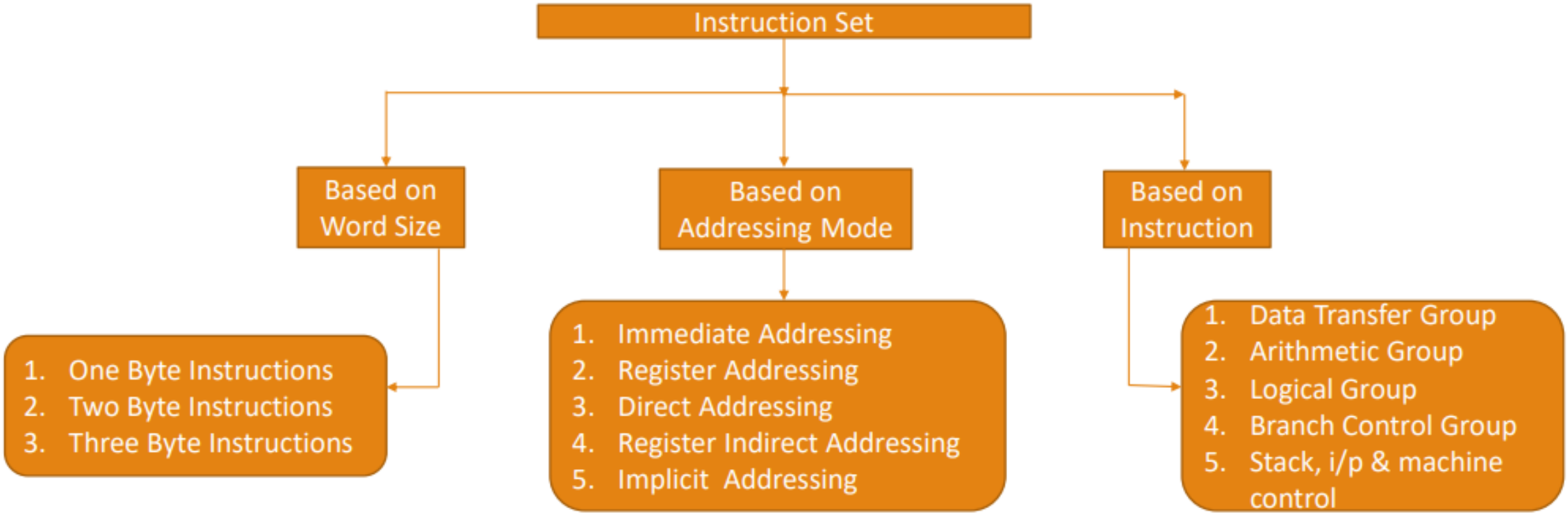
Instruction

- Instruction is a command or information given to the microprocessor to perform a given specific task on specified data.
- Each instruction has two parts:
  - one the task to be performed, and
  - the other is data to be operated.
- The code which specifies what operation to be performed is called operation code or opcode.
- The data on which operation is performed is called operand.
- For e.g. ADD B

  ADD is opcode and **B** is operand.
- The complete instruction is combination of opcode and operand.

Types of Instruction



Types of Instruction

- On the basis of length of word size
  1. One-byte instruction
  2. Two-byte instruction
  3. Three-byte instruction

- On the basis of function or operation mode
  1. Data transfer instructions
  2. Arithmetic instructions
  3. Logic and bit manipulation instructions
  4. Branch instructions
  5. Machine control instructions

On the basis of Word Size

One-Byte Instruction

- In 1-byte instruction, the opcode and the operand of an instruction are represented in one byte.
- A one-byte instruction includes opcode & operand in the same byte.
- Operand(s) are internal registers and are in the instruction in form of codes.
- If there is no numeral present in the instruction then that instruction will be of one-byte.
- E.g. MOV C

  ADD B

Two-Byte Instruction

- Two-byte instruction is the type of instruction in which
  - the first 8 bits indicates the opcode, and
  - the next 8 bits indicates the operand.
- This type of instructions need two bytes to store the binary codes.
- The mnemonic is always followed by 8-bit (byte) data.
- If an 8-bit numeral is present in the instruction then that instruction will be of two-byte.
- E.g. MVI H, 24H

  IN F1H

Three-Byte Instruction

- Three-byte instruction is the type of instruction in which
  - the first 8 bits indicates the opcode and
  - the next two bytes specify the 16-bit address.
- The low-order address is represented in second byte and the high-order address is represented in the third byte.
- This instruction will require three memory locations to store in the memory.
- The mnemonic is always followed by 16-bit (or adr).
- If a 16-bit numeral is present in the instruction then that instruction will be of three-byte.
- E.g. LXI H, 2400H
  LDA 2500H

On the basis of Function

Data Transfer Instruction

- These instructions move (or copy) data from source to destination.
  - Between register to register
  - Between register to memory
  - Between memory to register
  - Between I/O and accumulator
  - Load an 8-bit number in a register
  - Load an 16-bit number in a register pair
- Memory to memory transfer is not possible.
- After the data transfer,
  - the content of the source is not modified, and
  - the earlier content of the destination is altered.
- No flags are affected.
- E.g. MOV A , B LDA 1200H
  LXI H , 2800H LHLD 2500H

Arithmetic Instruction

- Arithmetic operations like addition, subtraction, increment and decrement are performed by this category of instructions.
- One of the operand is taken from the Accumulator and the other operand may be from registers or memory.
- The result of the arithmetic operations is stored in the Accumulator.
- All the flags are affected.
- E.g. ADD A INR D

  SUB B DCR E

Logical Instruction

- Logical functions like AND, OR and EX-OR are performed by this instructions.
- All logic functions are performed in relation with the contents of the Accumulator.
  - The logical operations cannot be performed directly with the content of two registers.
- After a logical operation has been performed, the result are placed in the accumulator replacing the original content of the accumulator.
- All the flags are affected.
- E.g. ANA B CMP B

  ORA B CMA

Branch Instruction

- Branch instructions change the sequence of the program execution unconditionally or conditionally.
- The condition of flags is used to take the decision for conditional branches.
- These instructions are the key to the flexibility and versatility of computer.
- No flags are affected.
- E.g. JMP 4000H

  JNZ 4000H

Stack, I/O & Machine Control Instruction

- The instructions dealing with interrupt handling and system operations are classified into this category.
- This type of instruction includes the instructions for input/output, stack and machine control.
- No flags are affected.
- E.g. PUSH B DI

  POP B HLT

Addressing Modes

Addressing Modes

- Addressing modes refers to the way in which the operand of an instruction is specified.
- The various addressing modes that are defined in a given instruction set architecture define
  - how machine language instructions in that architecture identify the operand(s) of each instruction.
- In any microprocessor instruction, the source and destination operands can be a:
  - Register
  - Memory Location
  - 8-bit number
- The method by which the address of source of data and the address of the destination of the result is given in the instruction are called the Addressing Modes.

Addressing Modes

- Addressing modes refers to the way in which the operand of an instruction is specified.
- It focuses on presenting the operand's address in the instructions.
- Types of Addressing Modes
    1. Immediate Addressing Mode
    2. Register Addressing Mode
    3. Direct Addressing Mode
    4. Register Indirect Addressing Mode
    5. Implied/Implicit Addressing Mode

Immediate Addressing Mode

- In this immediate addressing mode, the operand is specified within the instruction itself.
  - an immediate value is given to operate on.
- In 8085, the instructions having 'I' letter fall under this category.
  - 'I' indicates immediate addressing mode.
- Some Immediate Addressing modes mnemonics:

  LXI MVI ADI ACI SUI SBI CPI ANI XRI ORI

- E.g. MVI A , 01H (Move 01 in register A)

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

Direct Addressing Mode

- In this mode of addressing the address of the operand (data) is given in the instruction itslef.
- 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 instruction itself.
- This type of addressing is called absolute addressing.
- E.g. LDA 1020H (load content of memory location 1020 in A)

  SHLD 3050H (Load the content of L into memory location 3050

  Load the content of H into memory location 3051)

Register Direct Addressing Mode

- In this addressing mode, the data to be operated is in the general purpose register.
- The data to be operated is available inside the register(s) and register(s) is(are) operands.
- Therefore, the operation is performed within various registers of the microprocessor.
- E.g. ADD B (Add content of A and reg. B and store result in A)

  MOV A , B(Move content of register B to the accumulator)

Register Indirect Addressing Mode

- In this mode, the address of operand is specified by a register pair.
- The data to be operated is available inside a memory location and that memory location is indirectly specifies a register pair.
- So, in this type of addressing mode, it is the address of the address rather than the address itself.
- E.g. LDAX B (move content of the B-C register to the accumulator)

  MOV A , M (Move content of memory location pointed by

  register pair to the accumulator)

Implied/Implicit Addressing Mode

- The instruction of this mode do not have operands.
- In this mode, opcode specifies the address of the operands.
- If address of the 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.
- E.g. CMA (complement accumulator)

  RRC (rotate accumulator A right by one bit)

  RLC (rotate accumulator A left by one bit)

Operations Types in 8085

yeta dekhi pg 42 samma pardaina

- All of the operations of the microprocessor can be classified into three types:
  - Microprocessor Initiated Operations
  - Internal Operations
  - Externally Initiated Operations

Microprocessor Initiated Operation

- These are the operations that the microprocessor itself starts.
- There are basically four different microprocessor initiated operations:
  - Memory Read
  - Memory Write
  - I/O Read (Get input from the input device)
  - I/O Write (Send data to the output device)

Internal Data Operations

- The 8085 can perform a number of internal operations such as:
  - storing data,
  - arithmetic and logic operations,
  - testing for condition,
  - Sequencing the execution of instructions
  - Store/retrieve data from the stack during execution.
- To perform these operations, the microprocessor needs an internal registers like B, C, D, E, H, L, Accumulator, temporary registers, Flags, and others.

Externally Initiated Operations

- 8085 microprocessor support some Externally initiated operations, which are also known as Peripheral operations.
- Different external input/output devices or signals can initiate these type operations.
- External devices can initiate one of the 4 following operations:
  - Reset
  - Interrupt
  - Ready
  - Hold

Externally Initiated Operations (contd…)

- Reset
  - All operations are stopped and the program counter is reset to 0000.
  - When this RESET pin is activated by any external key, then all the internal operations are suspended for that time.
  - After that the execution of the program can begin at the zero memory address.
- Interrupt
  - 8085 MP chip have some pins for interrupt like TRAP, RST 5.5, RST 6.5 and RST 7.5.
  - The microprocessor's operations are interrupted and the microprocessor executes other emergency operations, what is called a "service routine".
  - This routine handles the interrupt, (perform the necessary operations).
  - Then the microprocessor returns to its previous operations and continues.

Externally Initiated Operations (contd…)

- Ready
  - The 8085 has a pin called READY.
  - This pin is used by external devices to stop the 8085 until they catch up.
  - The Input/Output devices that are connected to microprocessor are of different speed,
    - which is need to be synchronized with the speed of microprocessor.
  - This signal is used mainly to synchronize slower external devices with the microprocessor.

Externally Initiated Operations (contd…)

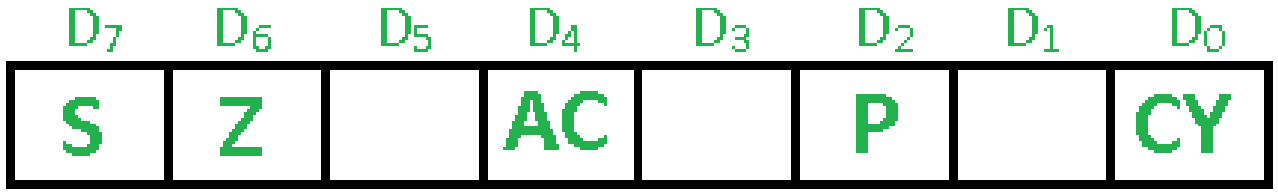
- Hold
  - The 8085 has a pin called HOLD.
  - This pin is used by external devices to gain control of the buses.
  - When the HOLD signal is activated by an external device, the 8085 stops executing instructions and stops using the buses.
  - This would allow external devices to control the information on the buses.
  - Eg: DMA
  - In this DMA, the external Input/Output devices are directly communicate with the memory without interfering the processor every time.

Data flow between memory and MPU

- First of all, the 16-bit address is placed on the address bus from the PC.
- Lets say, the address is 1020H where the data is placed.
- The higher order address i.e. 10H is placed on the address bus A8-A15 while the lower order address i.e. 20H is placed on the multiplexed address and data bus AD0-AD7.
- The lower order address continues to remain on this address bus so long as ALE (Address Latch Enable) remains high. Once ALE goes low, it carries data.
- The control unit sends the signal to indicate what type of the operations is to be performed.
- Since, the data is to be read from the memory therefore it is sends to enable the memory chip.
- The byte, from the memory location is then placed in the data bus, is placed on the data bus and sent to the instruction decoder.
- The instruction is decoded and accordingly the task is performed by the ALU.

Flag and Flag Register

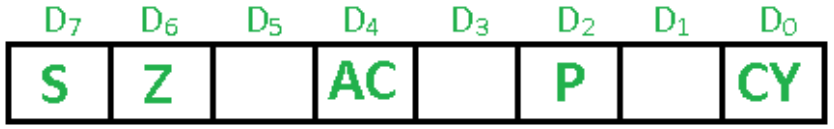
- The Flag register is a Special Purpose Register which shows the status of the task (microprocessor calculation).
- This is an 8-bit register but the only 5bit is used for the operation.
- The flag becomes set or reset after arithmetic and logical operation.
- The flag register has 5 flags which are
  1. Sign flag,
  2. Zero flag,
  3. Auxiliary carry flag,
  4. Parity flag, and
  5. the Carry flag.



Sign Flag

- Represented by the symbol S.
- After any type of arithmetic operation or logical operation,
  - if the value of D7 becomes 1 it basically shows that the number is negative and the sign flag is now set,
  - if the value of D7 becomes is 0, it basically shows that the number is positive and the sign flag is now reset.
- Eg:

```
MVI A 25H (load the value 25H in register A)
MVI B 05H (load the value 05H in register B)
SUB B (A = A – B)
```



- These set of instructions will basically reset the sign flag to 0 as 25 – 5 is always a positive number.

Zero Flag

- After any type of arithmetical or logical operation, if the output becomes 0 (00)H,
  - then zero flag is said to be set with value 1,
  - if not it becomes reset with the value 0.

- Eg:
```
MVI A 25H (load the value 25H in register A)
SUB A (A = A – A)
```
- These set of instructions will basically results 00H as 25 – 25 is always zero, so the zero flag is set to 1.

Carry Flag

- Carry is generated when performing n bit operations and the result is more than n bits,
  - then this flag becomes set i.e. 1,
  - otherwise it becomes reset i.e. 0.

- Eg:

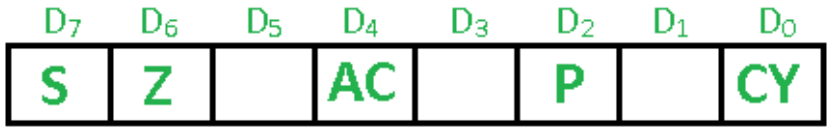
```
MVI A 25H (load the value 25H in register A)
MVI B 85H (load the value 05H in register B)
ADD B (A = A + B)
```

- These set of instructions will basically results answer with borrow.
- So, carry flag is set to 1.

Auxiliary Carry Flag

- Carry is generated when performing n bit operations and the result is more than n bits,
  - then this flag becomes set i.e. 1,
  - otherwise it becomes reset i.e. 0.
- This is the only flag register which is not accessible by the programmer
- Eg:

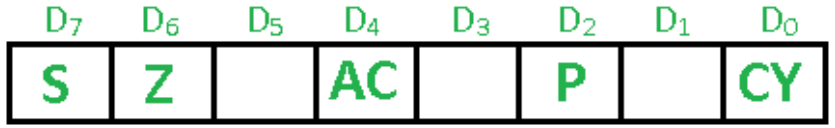
```
MVI A 2CH (load the value 25H in register A)
MVI B 85H (load the value 05H in register B)
ADD B (A = A + B)
```



- These set of instructions will basically results auxiliary during calculation as addition of lower order nibbles C and 5 will generate a carry.
- So, auxiliary carry flag is set to 1.

Parity Flag

- If after any arithmetic or logical operation the result has even parity, an even number of 1 bits,
  - the parity register becomes set i.e. 1,
  - otherwise it becomes reset i.e. 0.



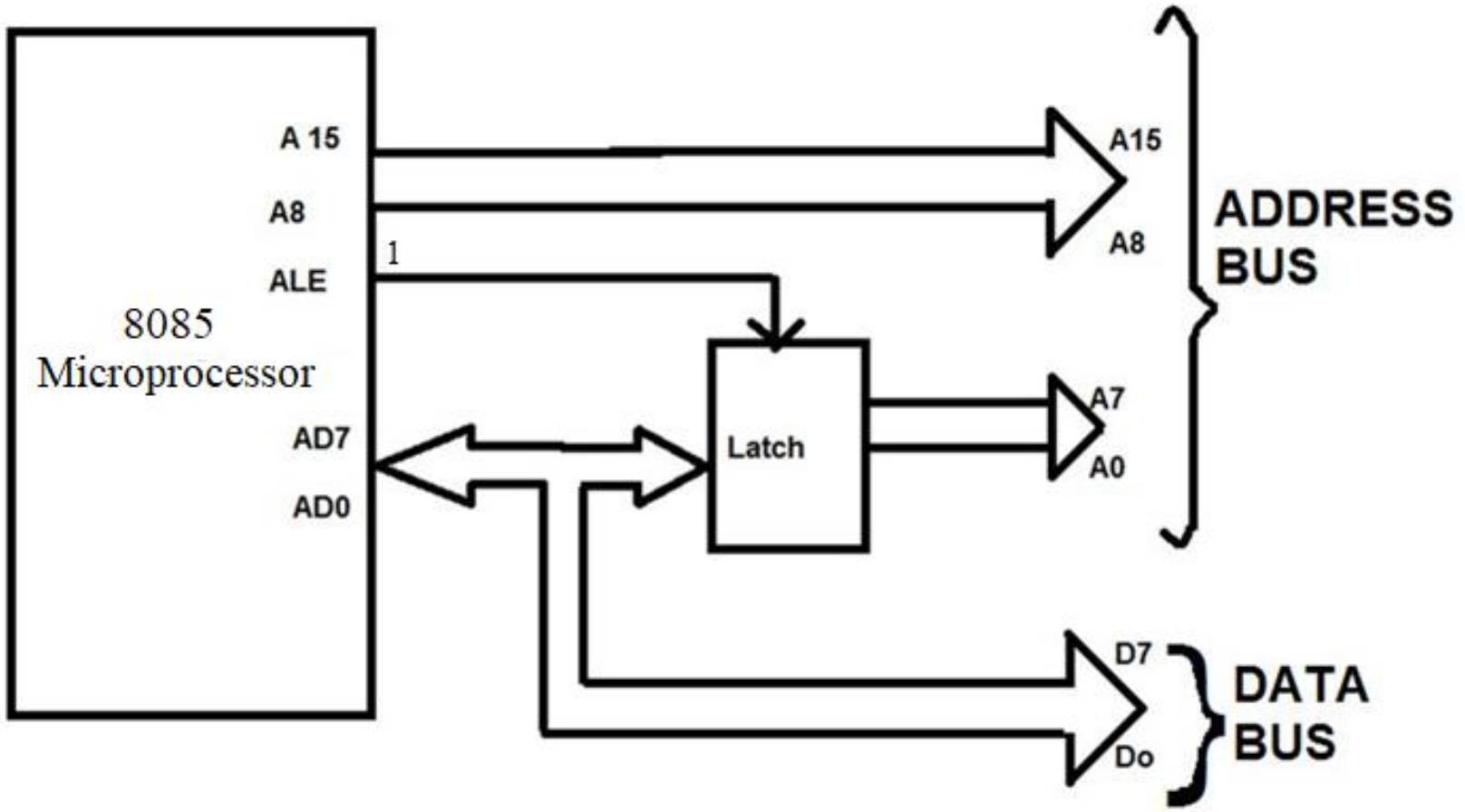
- Eg:

```
MVI A 01H (load the value 01H in register A)
MVI B 05H (load the value 05H in register B)
ADD B (A = A + B)
```

- This instruction will results 06H (i.e BCD -> 00000110), which contains even number of ones.
- So, parity flag is set to 1.

Multiplexing and Demultiplexing of Address-Data bus in 8085

Multiplexing and Demultiplexing of Address-Data bus in 8085



Control Bus and Control Signal

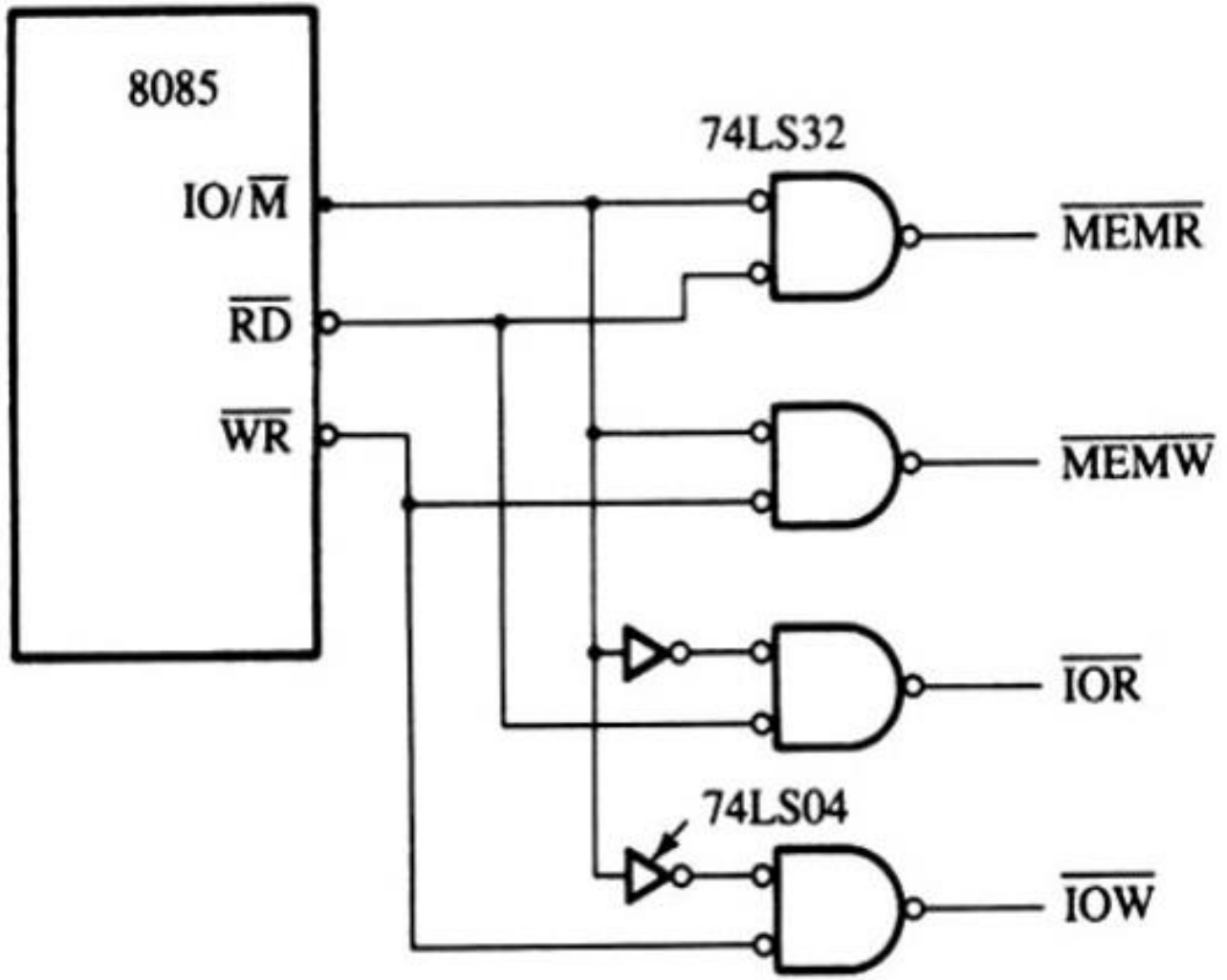
Control Bus in 8085

- This is a group of parallel lines used by the microprocessor to issue control signals such as IO/M, RD, WR.
- As you have known from the pin diagram of 8085, there are three different pins at which the microprocessor issues these control signals.
  1. Pin 30 for IO/M
  2. Pin 31 for WR
  3. Pin 32 for RD
- Control signals are the signals used to control the operations related to memory and other associated peripherals.
- How do we control *which operation* (read/write) is supposed to be done at *which location* (Memory/IO)?
  - The same data bus is used for *read* as well as *write* operations.
  - These two types of operations also need to be done at two possible locations (memory or I/O).
- The control signals issued by the microprocessor distinguish between these different types of operations.

Control Signals in 8085

- The three control signals are explained here in brief.
  - **IO/M (Control signal)**
    - This signal specifies whether the operation (read or write) is being performed on memory or an i/o device.
  - **RD (Control signal)**
    - Goes low for read operation and becomes high otherwise.
  - **WR (Control signal)**
    - Goes low for a write operation and becomes high otherwise.

Control Signals in 8085

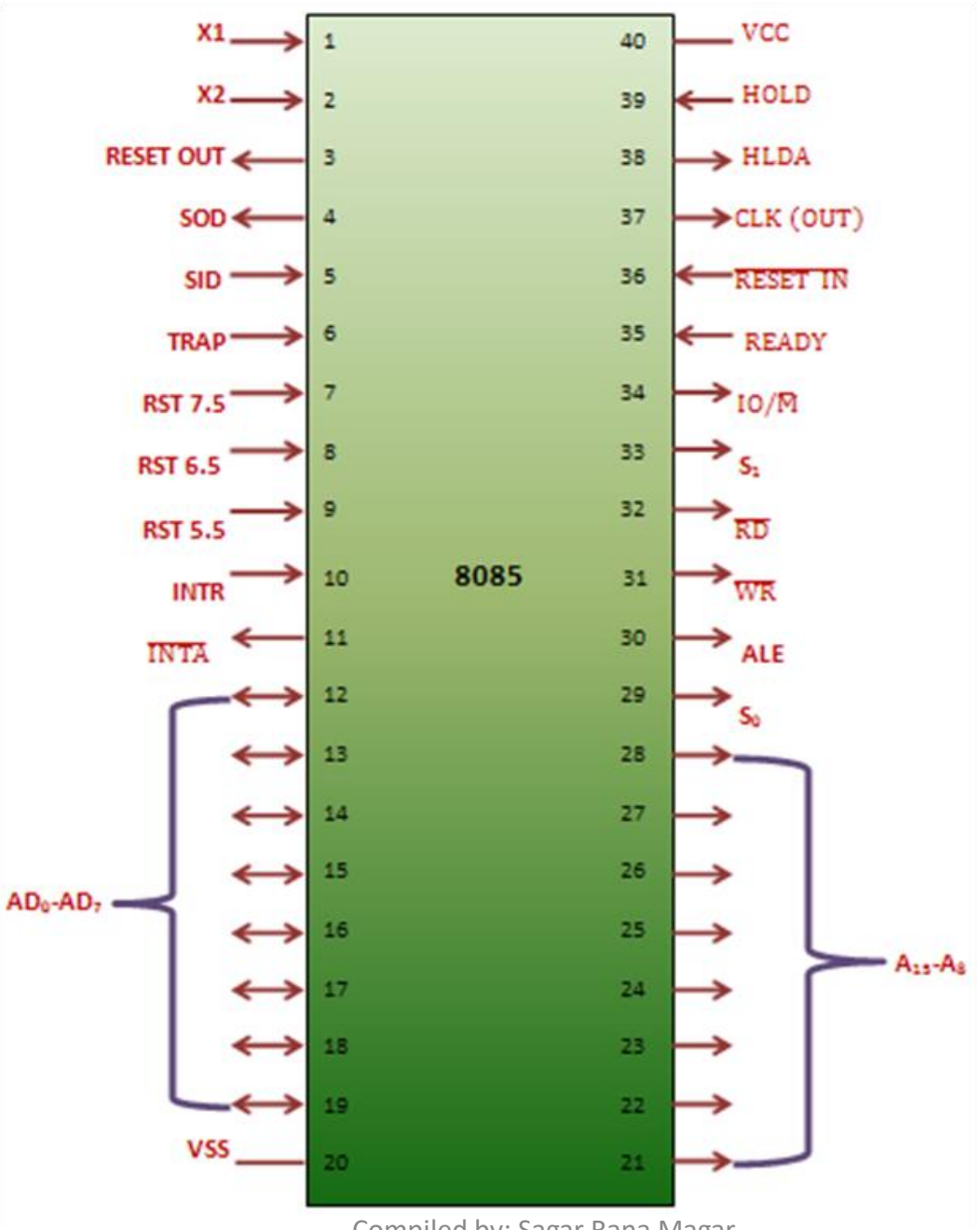


Truth Table of Control Signal

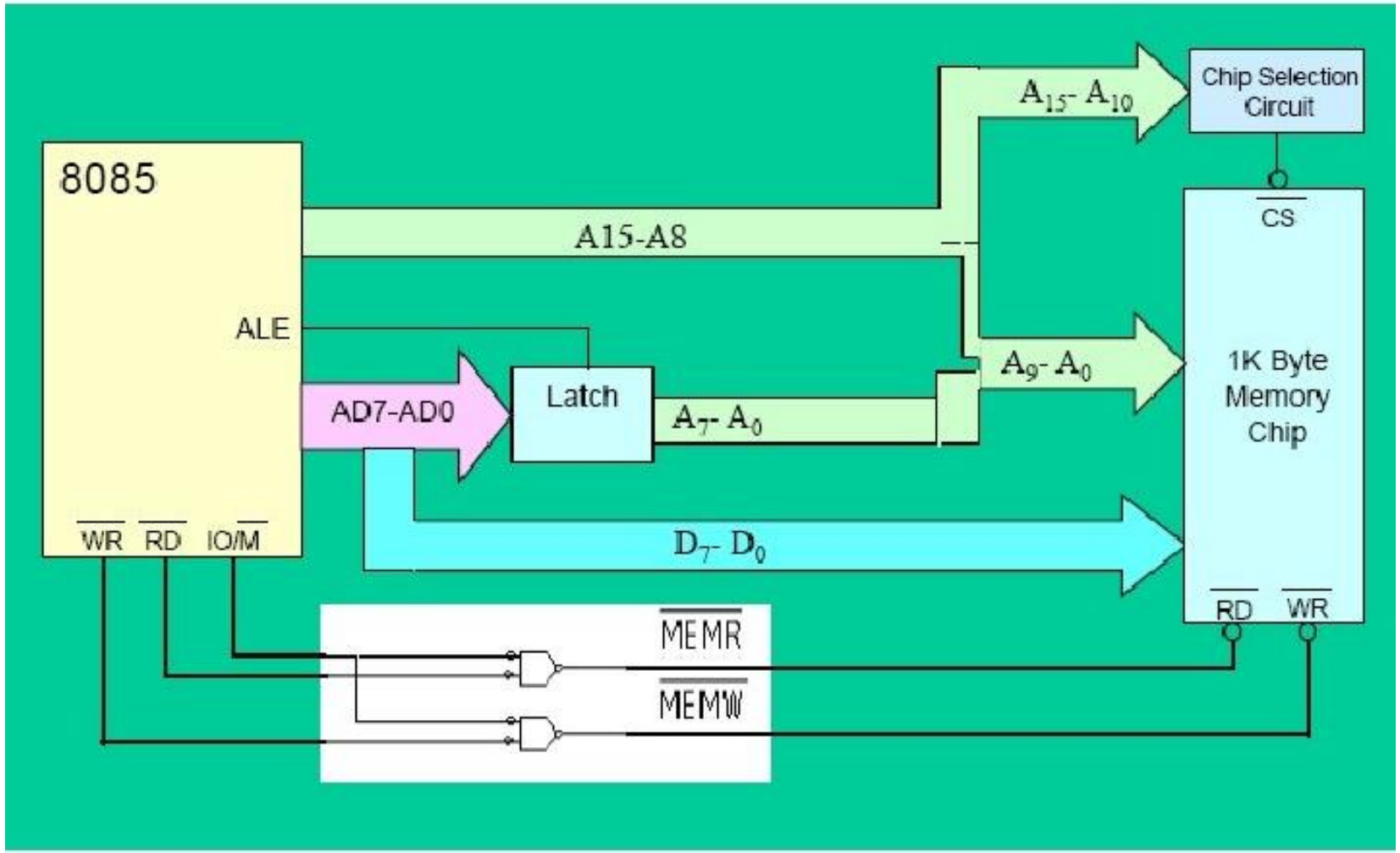
- The table explaining the control signals and status signals during various processes.

Operation	Status			Control signals
	IO/M	S1	S0	
Opcode Fetch	0	1	1	$\overline{RD}$= 0
Memory Read	0	1	0	$\overline{RD}$= 0
Memory Write	0	0	1	$\overline{WR}$ = 0
I/O Read	1	1	0	$\overline{RD}$= 0
I/O Write	1	0	1	$\overline{WR}$ = 0
Interrupt Acknowledge	1	1	1	$\overline{INTA}$ = 0

Memory Operation


X1
X2
RESET OUT
SOD
SID
TRAP
RST 7.5
RST 6.5
RST 5.5
INTR
INTA
AD0-AD7
VSS
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
8085
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
VCC
HOLD
HLDA
CLK (OUT)
RESET IN
READY
IO/M
S1
RD
WR
ALE
S0
A15-A8

Memory Read/Write



Machine Cycle

Important terms related to timing diagrams:

- **Instruction cycle:**
  - this term is defined as the number of steps required by the processor to complete the entire process
    - ie. Fetching and execution of one instruction.
  - The fetch and execute cycles are carried out in synchronization with the clock.
- **Machine cycle:**
  - It is the time required by the microprocessor to complete the operation of accessing the memory devices or I/O devices.
  - In machine cycle various operations like opcode fetch, memory read, memory write, I/O read, I/O write are performed.
- **T-state:**
  - Each clock cycle is called as T-states.