UNIT-II

Central processing unit

Processor Organization, Register organization, Instruction Cycle, The 8086 Processor Architecture, Register organization, physical memory organisation, General Bus operation, I/O addressing capability, special Processor activities, Minimum and maximum mode system and timings.

Processor Organization:-

→ To understand the organization of the processor, let us consider the requirements placed on the processor, the things that it must do:

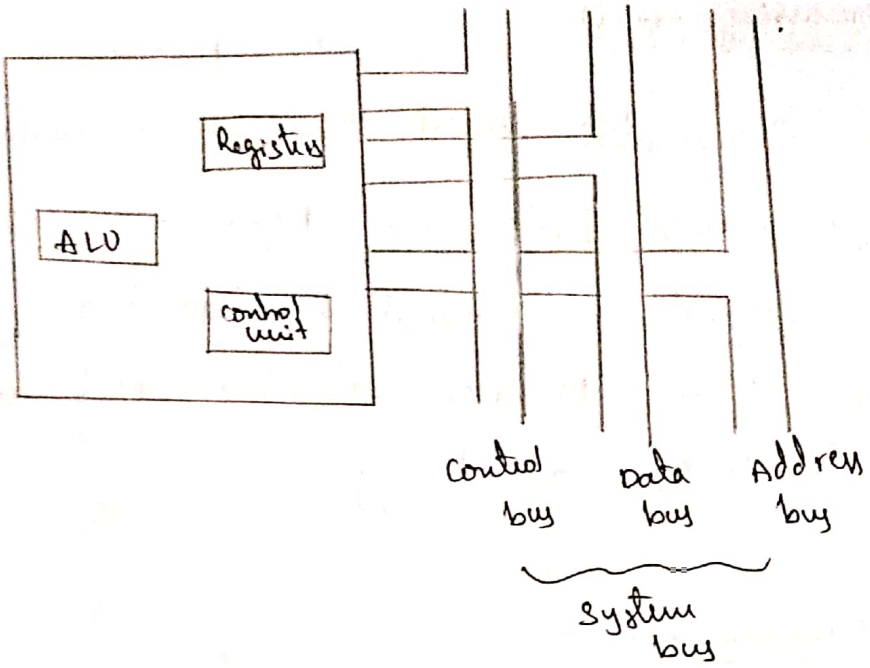
* fetch instruction
* Interpret instruction
* fetch data
* process data
* write data.

In other words, the processor needs a small internal memory.

The CPU with the system bus

Processor consist of

ALU, Control unit, Registers



→ ALU will perform arithmatic and logical operation (processing operation)

→ Control unit → control the movement of data or instruction and also controls the operations of ALU

→ Registers are nothing but internal memory.

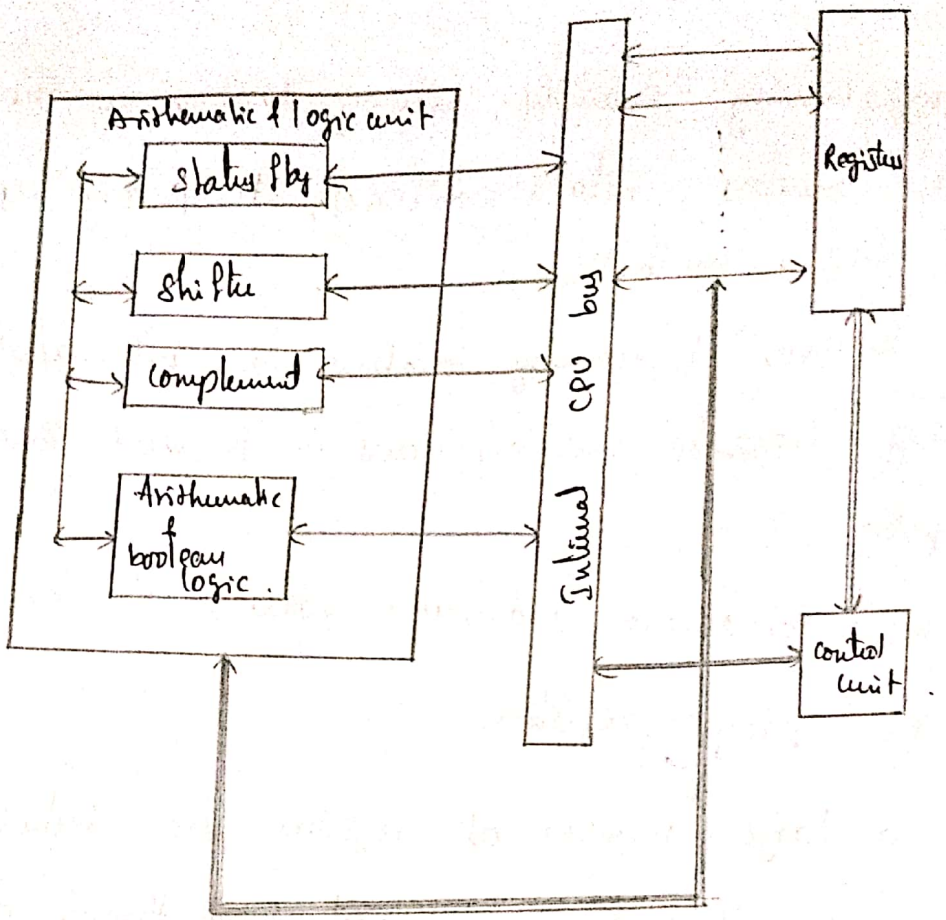
→ Here we have 3 buses

control bus carry control signals
data bus carry the data
address bus carry the address.

Internal structure of the CPU

1) Arithmatic and logic unit

Inside ALU it consist of status flag, shifter, complement & arithmatic and Boolean logic



Internal CPU bus will carry data from Register to ALU & ALC to Registers.

Similarity b/w computer and processor.

→ In both cases, there is a small collection of major elements connected by the data paths

→ computer
processor, I/O, Memory

→ processor
control unit, ALU, Registers.

Register Organization

In programming Example, memory locations are needed for storing pointers, counters, return addresses, temporary results, and partial Products during multiplication.

→ Having to refer to memory locations for such applications is time consuming because memory access is the most time consuming operation in computer.

→ It is more convenient and more efficient to store these intermediate values in processor registers.

→ when a large number of registers are included in the CPU, it is most efficient to connect them through a common bus system.

→ The registers communicate with each other not only for direct data transfers, but also while performing various microoperations.

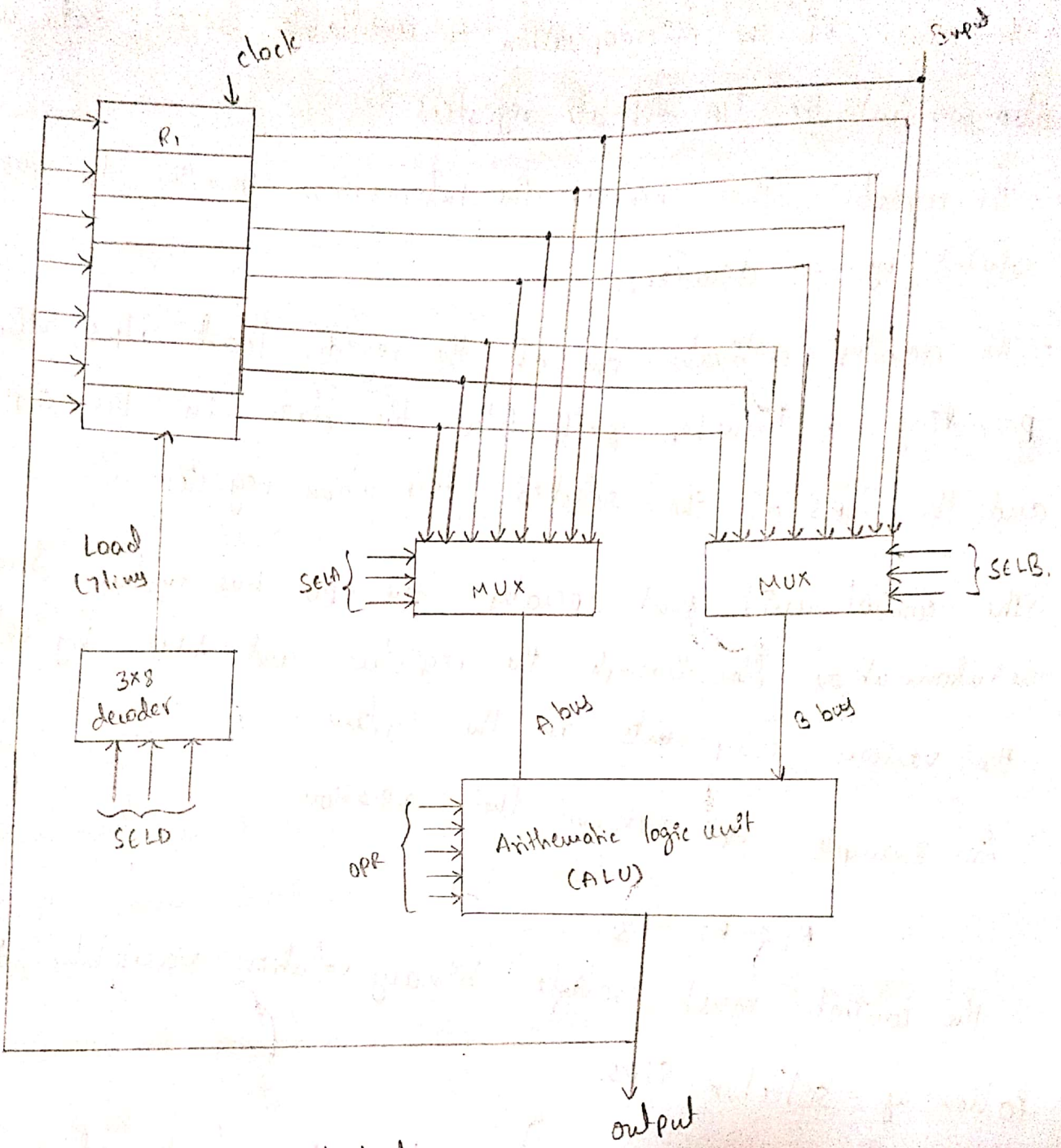
→ Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift microoperations in the processor.

→ A bus organization for ~~Gener~~ Seven CPU registers shown in figure.

→ The o/p of each register is connected to two multiplexers (MUX) to form the two buses A and B. The selection lines in each multiplexer select one register or the i/p data for particular Bus.

→ The A and B buses form the i/ps to a common arithmetic logic unit (ALU).

→ The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.



a. Block diagram.

3	3	3	5
SELA	SELB	SELD	OPR

b. control word.

Register set with common ALU.

→ The result of the microoperation is available for o/p data and also goes into the i/p of all registers.

→ The register that receives the information from the o/p bus is selected by a decoder.

→ The decoder activates one of the register load i/ps, thus providing a transfer path b/w the data in the o/p bus and the i/ps of the selected destination register.

→ The control unit that operates the CPU bus system directs the information flow through the registers and ALU by selecting the various components in the system.

For Example, to perform the operation

$R_1 \leftarrow R_2 + R_3$

the control must provide binary selection variable to the following selector i/ps.

1. MUX A selector (SELA): to place the content of R_2 into bus A.
2. MUX B selector (SELB): to place the content of R_3 into bus B.
3. ALU operation selector (OPR): to provide the arithmetic addition A+B.
4. Decoder destination selector (SELD): to transfer the content of the o/p bus into R1.

→ The four control selection variables are generated in the control unit and must be available at the beginning of a clock cycle.
→ The data from the two source registers propagate through the gates in the multiplexers and the ALU, to the o/p bus, and into the i/ps of the destination register, all during the clock cycle interval.
→ Then, when the next clock transition occurs, the binary information from the o/p bus is transferred into R1.
→ To achieve a fast response time, the ALU is constructed with high speed circuits.

Control word

→ There are 14 binary selection inputs in the unit, and their combined value specifies a control word.
→ The 14 bit control word is defined in figure.
→ It consist of four fields. Three fields contain three bits each, and one field has five bits.
→ The three bits of SELA select a source register for the A input of the ALU.
→ The three bits of SELB select a register for the B input of the ALU.
→ The three bits of SELD select a destination register using the decoder and its seven load o/p's.
→ The five bits of OPR select one of the operations in the ALU.
→ The 14-bit control word when applied to the selection i/ps specify a particular microoperation.

Binary code	SELA	SELB	SELD
000	Input	Input	None.
001	R_1	R_1	R_1
010	R_2	R_2	R_2
011	R_3	R_3	R_3
100	R_4	R_4	R_4
101	R_5	R_5	R_5
110	R_6	R_6	R_6
111	R_7	R_7	R_7

Encoding of Register Selection fields.

→ The 3-bit binary code listed in the first column of the table specifies the binary code for each of the three fields.

→ The register selected by fields SELA, SELB, and SELD is the one whose decimal number is equivalent to the binary in the code.

→ When SELA or SELB is 000, the corresponding multiplexer selects the external i/p data.

→ When SELD = 000, no destination register is selected but the contents of the o/p bus are available in the external o/p.

→ The ALU provides arithmetic and logic operations.

→ In addition the CPU must provide shift operations. The shifter may be placed in the i/p of the ALU to provide a preshift capability, or at the o/p of the ALU to provide postshifting capability.

→ In some cases, the shift operations are included with the ALU.

→ ~~The function table for this ALU is~~

→ The encoding of the ALU operations for the CPU is ~~to~~ listed below.

OPR select	operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B.	OR
01100	XOR A and B.	XOR
01110	complement A	COMA
10000	shift right A	SHRA
11000	shift left A.	SHLA.

→ The OPR field have five bits and each operation is designated with a symbolic name.

Examples of microoperations

→ A control word of 14 bits is needed to specify a microoperation in the CPU.

→ The control word for a given microoperation can be derived from the selection variables.

→ For example, the subtract microoperation given by the stmt

$R_1 \leftarrow R_2 - R_3$

Specifies R_2 for the A i/p of the ALU, R_3 for the B i/p of the ALU, R_1 for the destination register, and an ALU operation to subtract A-B.

→ Thus the control word is specified by the four fields and the corresponding binary value for each field is obtained from the encoding listed in above tables.

→ The binary control word for the subtract microoperation is 010 011 00101 and is obtained as follows

Field :	SELA	SELB	SELD	OPR
Symbol:	R_2	R_3	R_1	SUB
control word:	010	011	001	00101

→ The increment and transfer microoperations do not use the B i/p of the ALU.

→ For these cases, the B field is marked with a dash.

→ We assign 000 to any unused field when formulating the binary control word, although any other binary number may be used.

→ To place the content of a register into the o/p terminals we place the content of the register into the A i/p of the ALU, but none of the registers are selected to accept the data.

→ The ALU operation TSFA places the data from the register, through the ALU, into the o/p terminals.

The direct transfer from i/p to o/p is accomplished with a control word of all 0's (making the B field 000).

→ A register can be cleared to 0 with an Exclusive-OR operation

→ This is because $x \oplus x = 0$.

Microoperation	Symbolic Designation				Control word
	SELA	SELB	SELD	OPR	
$R_1 \leftarrow R_2 - R_3$	R_2	R_3	R_1	SUB	010 011 001 00101
$R_4 \leftarrow R_4 \vee R_5$	R_4	R_5	R_4	OR	100 101 100 01000
$R_6 \leftarrow R_6 + 1$	R_6	—	R_6	INCA	110 000 110 00001
$R_7 \leftarrow R_1$	R_1	—	R_7	TSFA	001 000 111 0000
o/p $\leftarrow R_2$	R_2	—	None	TSFA	010 000 000 00000
o/p $\leftarrow$ i/p	i/p	—	None	TSFA	000 000 000 00000
$R_4 \leftarrow$ shl R_4	R_4	—	R_4	SHLA	100 000 100 11000
$R_5 \leftarrow 0$	R_5	R_5	R_5	XOR	101 101 101 01100

→ It is apparent from these examples that many other microoperations can be generated in the CPU.

→ The most efficient way to generate control words with a large number of bits is to store them in a memory unit.

→ A memory unit that stores control words is referred to as a control memory.

→ By reading consecutive control words from memory, it is possible to initiate the desired sequence of microoperations for the CPU.

→ This type of control is referred to as microprogrammed control.

→ The binary control word for the CPO will come from the O/Ps of the control memory marked "micro-OPS".

Instruction cycle

— same as unit-I topic.

The 8086 Processor Architecture

The architecture of 8086 can be divided into two ~~parts~~ independent functional units.

1) Bus interface unit
2) Execution unit.

1) Bus interface unit

functions: 1) fetch the instruction or data from memory
2) write the data to memory.

Micro processor: It is also called CPU. It is main part of the computer which perform all operations.

→ Micro processor is a brain of micro computer.
→ Every thing is done by processor.
→ microprocessor, processor, CPU all are same.
→ Small processor (chip) is available in motherboard that is called up.
→ It is a single chip. which is capable of processing data.
→ It controls all components in computer (monitor, i/p device...)
→ It executes sequence of instructions.
→ Microprocessor fetch, decode and execute the instruction.
→ Internal architecture is complex.

Types of micro processor.

8080 — 1974
8085 — 1976
8086 — 1978
8088 — 1979
80286 — 1982
80386 — 1985
80486 — 1989
Pentium — 1993
Pentium - II — 1997
Pentium - II — 1999.
Pentium - IV — 200

Register Organization of 8086

→ The 8086 microprocessor has a total of 14 registers that are accessible to the programmer.

→ All these registers are 16 bit in size. The registers of 8086 are categorized into 4 different groups.

→ 8086 has set of registers known as general purpose and special purpose registers.

→ All of them are 16-bit registers

→ The general purpose registers, can be used as either 8 bit or 16 bit register.

→ They may be either used for holding data, variables and intermediate results temporarily or for other purpose like a counter or for storing offset address for some particular addressing mode etc.

→ The special purpose registers are used as segment registers, pointers, Index registers or as offset storage registers for particular address mode.

→ All these registers are 16 bit in size, The registers of 8086 are categorized into 4 different groups.

1. General Data Registers
2. Segment Registers
3. Pointers and Index Registers
4. Flag registers.

1. General Data Registers / General Purpose Registers

→ All general purpose registers of the 8086 microprocessor can be used for arithmetic and logic operations.

→ These all general registers can be used as either 8 bit or 16 bit registers.

→ The registers AX, BX, CX and DX are the general purpose 16 bit registers.

→ AX is used as 16-bit accumulator, with the lower 8 bits of AX designated as AL and higher 8 bits as AH.

→ In program we can use AX, AL, AH

→ AX (Accumulator register):-

→ This accumulator used in arithmetic, logic and data transfer operations, for manipulation and division operations, one of the numbers must be placed in AX or AL.

→ BX (Base Registers)

→ BX is a 16 bit register, but BL indicates the lower 8 bits of BX and BH indicates the higher 8 bits of BX.

→ The register BX is used as address register to form physical address in case of certain addressing modes (Eg: indexed, register indirect).

→ CX (Counter Register)

→ The register CX is used default counter in case of shift and loop instructions. Counter register can also be used as a counter in string manipulation and shift/rotate instruction.

→ DX (Data Register).

DX register is a general purpose register which may be used as an implicit operand or destination in case of a few few instructions.

→ Data register can also be used as a port number in I/O operations.

2. Segment Registers

The 8086 architecture uses the concept of segmented memory

→ 8086 can able to access a memory capacity of up to 1 megabyte.

→ This one megabyte of memory divided in to 16 logical segments. Each segment contains 64 kbyte of memory.

→ The segment register of 8086 are

1. code segment register
2. Data segment register
3. stack segment register
4. Extra segment register.

→ code segment Register (CS):—

code segment Register (CS) is a 16 bit register that is used for addressing memory location in the code segment of the memory (64 kb). where the executable program is stored. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

→ Stack Segment (SS) Register:—

Stack segment (SS) is a 16 bit register that used for addressing stack segment of the memory (64 kb) where stack

data is stored. SS register can be changed directly using POP instruction.

→ Data Segment (DS) Register

Data segment (DS) is a 16 bit register that points the data segment of the memory (64 kb) where the program data is stored. DS register can be changed directly using POP and LDS instructions.

→ Extra segment (ES) Register

Extra Segment (ES) is a 16-bit register that also points the data segment of the memory (64 kb) where the program data is stored. ES register can be changed directly using POP and LES instructions.

3. Pointer and Index Registers :—

(a) Index Registers :-

The index registers can be used for arithmetic operations but their use is usually concerned with the memory addressing modes of the 8086 microprocessor (indexed, base indexed and relative base indexed addressing modes).

→ The index registers are particularly useful for string manipulation.

→ There are two types of Index Registers

(i) Source Index :— SI is a 16-bit register. This register is used to to store the offset of source data in data segment. In other words the source Index Register is used to point the memory locations in the data segment.

(4) DI (Destination Index):

DI is a 16-bit registe. This is destination index registe performs the same function as SI.

→ There is a class of instructions called string operations that use DI to access the memory locations in DATA or EXTRA Segment.

(b) Pointer Registers:-

Pointer Registers contains the offset of data (variable, labels) and instructions from their base segments (default segments).

8086 microprocessor contain three pointer Registers

(i) stack pointer

(ii) Base pointer

(iii) Instruction pointer

(i) Stack pointer (SP):-

Stack pointer Registe points the program stack that means SP stores the base address of the stack segment.

(ii) Base Pointer (BP):-

Base pointer registe also points the same stack segment. Unlike SP, we can use BP to access data in the other segments also.

(iii) Instruction Pointer (IP):-

The instruction pointer is a registe that holds the address of the next instruction to be fetched from memory. It contains the offset of the next word of instruction code instead of its actual address

Flag registers (Status Register):-

The 8086 flag register contents indicate the results of computations in the ALU. It also contains some flag bits to control the CPU operations.

→ 8086 has a 16-bit flag register which is divided into two parts

(a) condition code or status flags

(b) machine control flags

→ The condition code flag register is the lower byte of the 16-bit flag register along with the overflow flag

→ The control flag register is the higher byte of the flag register of 8086.

→ It contains three flags. → direction flag (D), Interrupt flag (I) and trap flag (T).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	O	D	I	T	S	Z	X	Ac	X	P	X	Cy

O - Overflow flag

D - Direction flag

I - Interrupt flag

T - Trap flag

S - Sign flag

Z - Zero flag

Ac - Auxiliary carry flag

P - Parity flag

Cy - Carry flag

X - Not used.

1. SF (Sign flag) :- This flag represents sign of the result

0- Result is positive

1- Result is Negative.

2. ZF (Zero flag) :- ZF is set if the result of the computation or comparison performed by the previous instruction is zero. Otherwise ZF is reset.

3. PF (Parity flag) :-

This flag is set to 1, if the lower byte of the result contains even number of 1's.

0- odd parity

1- Even parity.

4. CF (Carry flag) :-

This flag is set, when there is a carry out of MSB in case of addition or borrow in case of subtraction

0 - No carry / Borrow

1 - Carry / Borrow.

5. AC (Auxiliary Carry flag)

This is set when there is a carry from the lowest nibble (i.e bit three during addition), or borrow for the lowest nibble (i.e bit three, during subtraction).

6. OF (Over flow flag):-

This flag is set, if an overflow occurs i.e if the result of a signed operation is large enough to accommodate in a destination register.

7. TF (Trap flag):-

If this flag is set, the processor enters the single step execution mode. when in the single-step mode, it executes an instruction and then jumps to a special service routine that may determine the effect of executing the instruction. This type of operation is very useful for debugging programs.

8. If (interrupt flag):-

If this flag is set, the maskable interrupts are recognized by the CPU, otherwise they are ignored.

9. Df (Direction flag):-

This is used by string manipulation instructions.

→ 0- The string is processed begging from the lowest address to the highest address (i.e auto incrementing mode).

→ 1- The string is processed ~~beg~~ from the highest address towards the lowest address (i.e auto decrementing mode).

AH	AL
BH	BL
CH	CL
DH	DL

General Data Registers.

CS
SS
DS
ES

Segment Registers

FLAGS/PSW

SP
BP
SI
DI
IP

Pointers and Index Registers

The 8086 processor Architecture

8086 processors support a 16-bit ALU, a set of 16-bit registers and provides segmented memory addressing capability, a rich instruction set, powerful interrupt structure, fetched instruction queue for overlapped fetching and execution etc.

The complete architecture of 8086 can be divided into two ~~parts~~ independent functional unit

a. Bus interface unit

b. Execution unit.

Bus Interface unit: The bus interface unit contains the circuit for physical address calculations and pre decoding instruction byte queue (6 byte long).

→ The bus interface unit makes the system's bus signals available for External interfacing of the devices.

→ In other words, this unit is responsible for establishing communication with external devices and peripherals including memory via the bus.

→ The 8086 addresses a segmented memory.

→ The complete physical address which is 20-bit long is generated using segment and offset registers, each 16-bit long.

→ for generating a physical address from content of these two registers, the content of a segment register also called as segment address is shifted left bit-wise four times and to this result, content of an offset register also called as offset address is added to produce a 20 bit physical address.

→ for example, if the segment address is 1005H and the offset is 5555H, then the physical address is calculated as below.

Segment address → 1005H

Offset address → 5555H

Segment address → 1005H → 0001 0000 0000 0101

shifted by 4 bit position → 0001 0000 0000 0101 0000

\+ offset address → 0101 0101 0101 0101

0001 0101 0101 1010 0101

1 5 5 A 5.

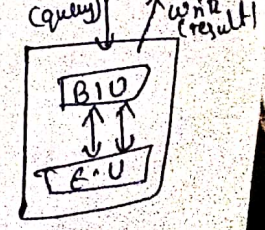
functions of bus Interface unit

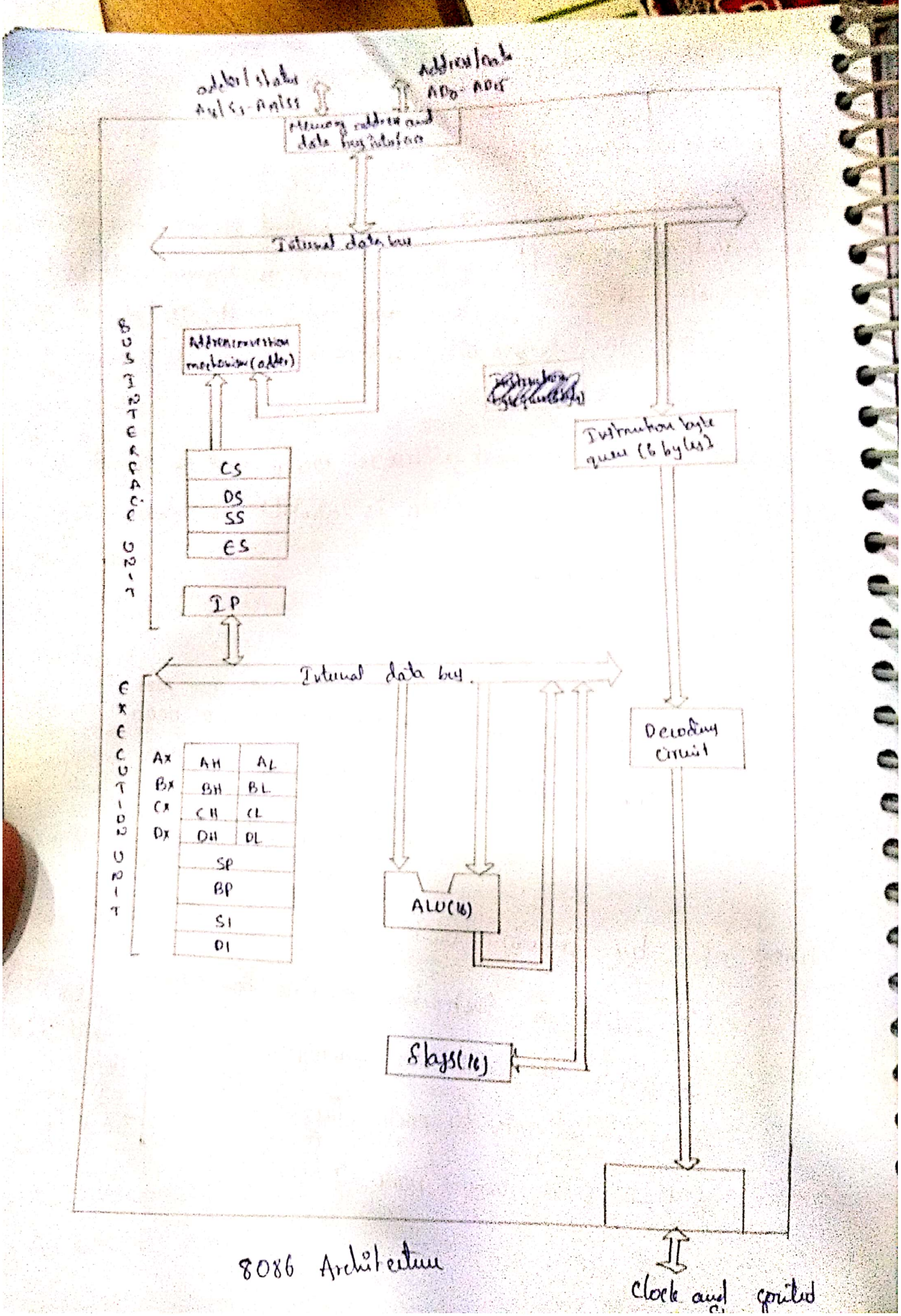
1.) fetch the instruction or data from memory.

2.) write the data to memory.

3.) write the data to port (I/O)

4.) Read data from port (I/O.





Bus Interface unit is divided into 3 parts to do it functions

1. Instruction pointer (IP)
2. Segment Register
3. Instruction queue.

1. Instruction pointer (address of the next instruction)

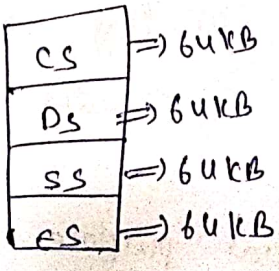
It is a 16-bit register that keeps the address of memory location of coming instruction to be executed.

2. Segment Register :-

The memory space 1MB of 8086 is segment into 4 blocks each block specified by register with max size 64 kB.

1. code segment
2. Data segment
3. stack segment
4. Extra segment.

} 1MB.



3. Instruction Queue :-

→ BIU performs it's operation in parallel with execution unit.
→ BIU fetches instruction bytes while execution unit is executing operations.
→ The prefetched instruction is saved in group of high speed register is known as instruction queue.

Execution Unit

functions :-

1) Execution unit to tell BIU where to fetch the instruction or data from.

2) TO decode the instruction

3) TO Execute the instruction.

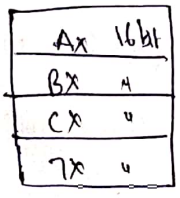
4) E.U contains the control circuitry to perform various internal operations.

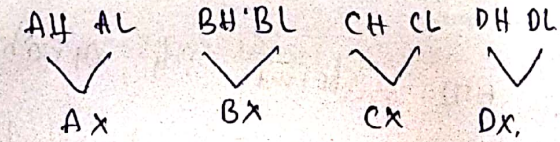
functional parts :-

1) General purpose Registers
2) pointer and inter Registers
3) ALU
4) flag registers
5) Timing & control unit

1) General purpose Registers :-

8086 up consist of 4 General purpose Registers with 16 bit





2) Pointer & Index Registers

8086 has two pointers & two indexed registers

1. Stack pointer (SP)
2. Base point (BP)

3. Source Index (SI)
4. Destination Index (DI)

3. Arithematic logic unit (ALU):-

→ It is a 16 bit Register
→ It peforms arithematic & logical operations. (16 as well as 8bit operations)

4. Flag register:- 8086 has 16 bit flag register.

7 flags are unused

9 flags are used.

Among 9 flags, we divided into two categories.

1. status flag ⇒ 6 status flags
2. control flag ⇒ 3 control flags.

6 status flags:- carry flag
Auxilary carry flag
zero flag
sign flag
parity flag
overflow flag.

3 control flags:- direction flag
Interrupt enable flag
Trap flag.

5. Timing & control unit:- The control unit of Execution unit directs all internal operations & also responsible for genration of control signal.

Physical Memory Organisation

→ In an 8086 based system, the 1M bytes memory is physically organised as an odd bank and an even bank each of 512 K bytes, addressed in parallel by the processor.

→ Byte data with an even address is transferred on D_7-D_0, while the byte data with an odd address is transferred on D_{15}-D_8 buslines.

→ The processor provides two enable signals, $\overline{BHE}$ and A_0 for selection of either even or odd or both the banks.

→ The instruction stream is fetched from memory as words and is addressed internally by the processor as necessary.

→ In other words, if the processor fetches a word from memory, there are different possibilities like,

→ Both the bytes may be data operands

→ Both the bytes may contain opcode bits

→ One of the bytes may be opcode while the other may be data.

→ All the above possibilities are taken care of by the internal decoder circuit of the microprocessor.

→ The opcodes and operands are identified by the internal decoder circuit which further derives the signals those act as input to the timing and control unit.

→ The timing and control unit then derives all the signals required for execution of the instruction.

→ While refering to word data, the BIU requires one or two memory cycles, depending upon whether the starting byte is located at an even or odd address.

→ It is always better to locate the word data at even address.

→ To read or write a complete word from/to memory. If it is located at an even address, only one read or write cycle is required.

→ If the word is located at an odd address, the first read or write cycle is required for accessing the lower byte while the second one is required for accessing the ~~lower byte while the second one is~~ upper byte.

→ Thus, two bus cycles are required, if a word is located at an odd address.

→ It should be kept in mind that while initializing the structure like stack they should be initialized at an even address for efficient operation.

→ 8086 is a 16 bit microprocessor and hence can access two bytes of data in one memory or I/O read or write operation.

→ But the commercially available memory chips are only byte size. i.e they can store only one byte in a memory location.

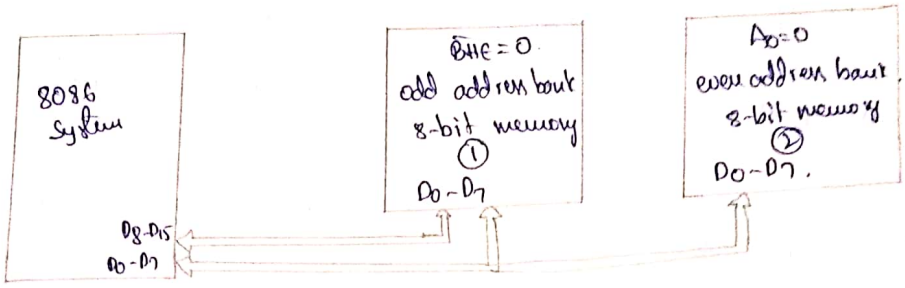
→ Obviously, to store 16-bit data, two successive memory locations are used and the lower byte of a 16-bit data can be stored in the first memory location while the second byte is stored in the next location.

→ In a sixteen bit read or write operation both of these bytes will be read or written in a single machine cycle.

→ Thus bits $D_0 - D_7$ of a 16-bit data will be transferred over D_0-D_7 (lower byte) of 16-bit data bus to/from 8-bit memory (1) and

bit D_8-D_{15} of the 16-bit data will be transferred over D_8-D_{15} (higher byte) of the 16-bit data bus of microprocessor, to/from 8-bit memory (1).

→ Thus to achieve 16-bit data transfer using 8-bit memories, in parallel, the map of the complete system byte memory addresses will obviously be divided into the two memory banks.



physical memory organisation.

→ The lower byte of 16 bit data is stored at the first address of the map 00000H and it is to be transferred over D_0-D_7 of the microprocessor bus so 00000H must be in 8-bit memory (2).

→ Higher byte of the 16-bit data is stored in the next address 00001H, it is to be transferred over D_8-D_{15} of the microprocessor bus so the address 00001H must be in 8-bit memory (1).

→ On similar lines, for the next 16-bit data stored in the memory, immediately after the previous one, the lower byte will be stored at the next address 00002H and it must be in 8-bit memory (2) while the higher byte will be stored at the next address 00003H that must be in 8-bit memory (1).

→ Thus, if it is imagined that the complete memory map of 8086 is filled with 16-bit data, all the lower bytes (D_0-D_7) will be stored in the 8-bit memory bank (2) and all the higher bytes (D_8-D_{15}) will be stored in the 8-bit memory bank (1)

→ The two signals A0 and $\overline{BHE}$ solve the problem of selection of appropriate memory banks.

$\overline{BHE}$	A0	Indication
0	0	whole word.
0	1	upper byte from or to add address
1	0	lower byte from or to even address.
1	1	none.

→ certain locations in memory are reserved for specific CPU operations. The locations from FFFF0H to FFFFFH are reserved for operations including jump to initialisation progamme and I/O-processor initialization.

→ The locations 00000H to 003FFH are reserved for interrupt vector cycle.

→ The interrupt structure provide space for a total of 256 interrupt vectors.

General Bus Operation

→ The 8086 has a combined

Memory segmentation

The memory in an 8086/8088 based system is organized as segmented memory.

→ In this, the complete physically available memory may be divided into a number of logical segments.

→ Each segment is four byte in size and is address by one of the segment registers.

→ The BIU contains four 16-bit special purpose registers called as segment Registers

1) code segment Register
2) Data segment Register
3) Extra segment Register
4) stack segment Registe.

→ The number of address lines in 8086 is 20,

→ 8086 BIU will send 20 bit address, so as to access one of the 1MB memory locations.

→ The four segment registers actually contain the upper 16 bits of the starting addresses of four memory segments of 64KB with which the 8086 is working

→ A segment is a logical unit of memory that may be upto 64 kilobyte long.

→ Each segment is made up of contiguous memory location.

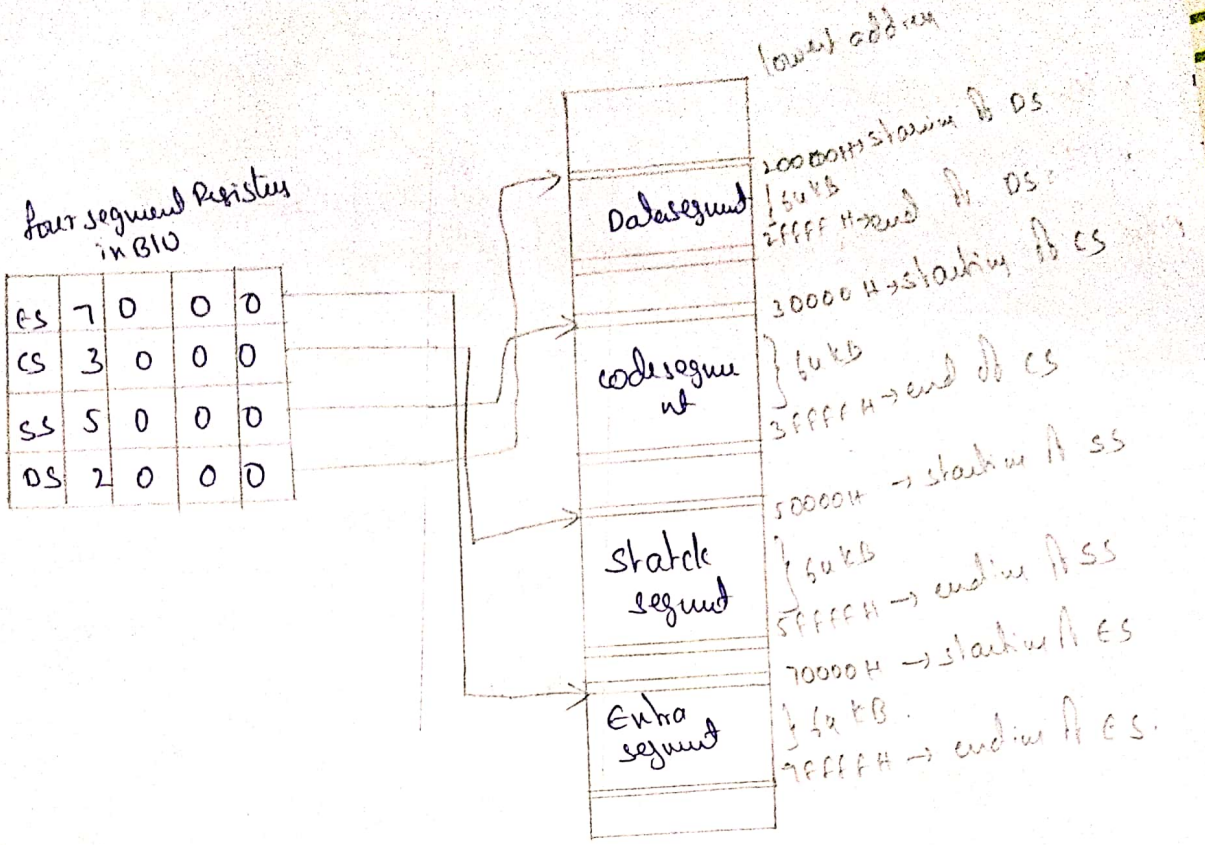
→ It is an independent, seperately addressable unit.

→ starting address will always be changed.

→ Note that the 8086 does not work the whole 1MB memory at any given time.

→ However, it work only with four 64 KB segments with in the whole 1MB memory.

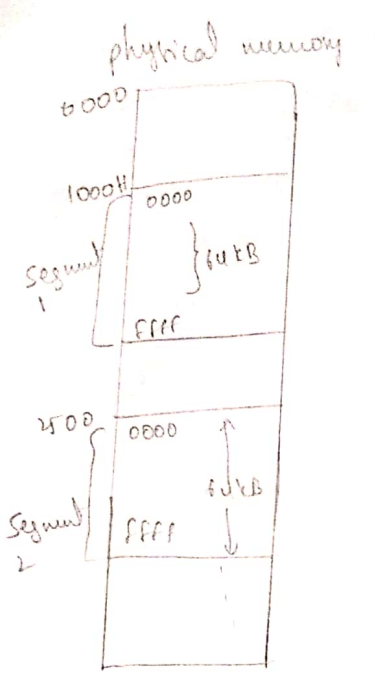
→ Below is the one way of positioning four 64 kilobyte segments within the 1M byte memory space of an 8086.



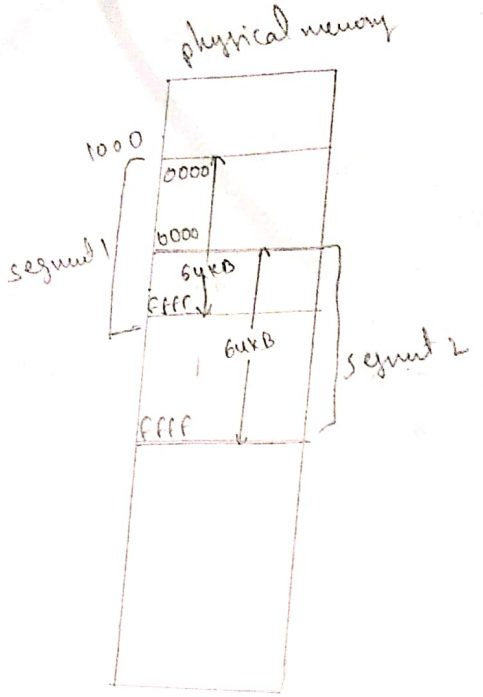
Types of segmentation

1) Overlapping segment:- A segment starts at a particular address and its maximum size can go up to 64 kilobytes. But if another segment starts along with this 64 kilobyte location of the first segment, then the two are said to be overlapping segment.

2) Non overlapped segment:- A segment starts a particular address and its maximum size can go up to 64 KB. But if another starts after this 64 KB location of the first segment, then the two segments are said to be non overlapped segments.



Non overlapped segments



overlapped segments.

Rules of segmentation:- segmentation process follows some rules as follows.

1) The starting address of a segment be should be such that it can be evenly divided by 16.

2) Minimum size of a segment can be 16 byte and maximum can be 64 KB.

Segment	offset register	function
CS	IP	address of next instruction
DS	BX, DI, SI	address of data
SS	BP, SP	Address in the stack.
ES	BX, DI, SI	Address of destination data.

Advantages of segmentation:

→ It provides a powerful memory management mechanism.

→ Data or stack related operations can be performed in different segments.

→ code related operation can be done in separate code segments.
→ It allows to process to easily share data
→ It allows to extend the address ability of the processor. i.e segmentation allows the use of 16 bit registers to give an addressing capability of 1Megabyte's. with out segmentation it would be require 20 bit registers.
→ It is possible to enhance the memory size of code data or stack segments beyond 64KB by alloting more than one segment for each data.

General Bus Operation

→ The 8086 has a combined address and data bus commonly referred to as a time multiplexed address and data bus.
→ The main reason behind multiplexing address and data over the same pins is the maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package.
→ The bus can be demultiplexed using a few latches and transreceivers, whenever required.
→ Basically, all the processor bus cycle consist of atleast four clock cycles. These are referred to as T_1, T_2, T_3, T_4. The address is transmitted by the processor during T_1.
→ It is present on the bus only for one cycle.
→ During T_2. i.e the next cycle, the bus is tristated for changing the direction of bus for the following data read cycle.
→ The data transfer takes place during T_3 and T_4.

→ In case, an addressed device is slow and shows 'NOT READY' status the wait states Tw are inserted b/w T_3 and T_4.

→ These clock states during wait period are called idle state (T_i), wait state (T_w) or inactive state.

→ The processor uses these cycles for internal housekeeping

→ The Address Latch enable (ALE) signal is emitted during T_1 by the processor (minimum mode) or the bus controller (maximum mode) depending upon the status of the MN/$\overline{MX}$ i/p.

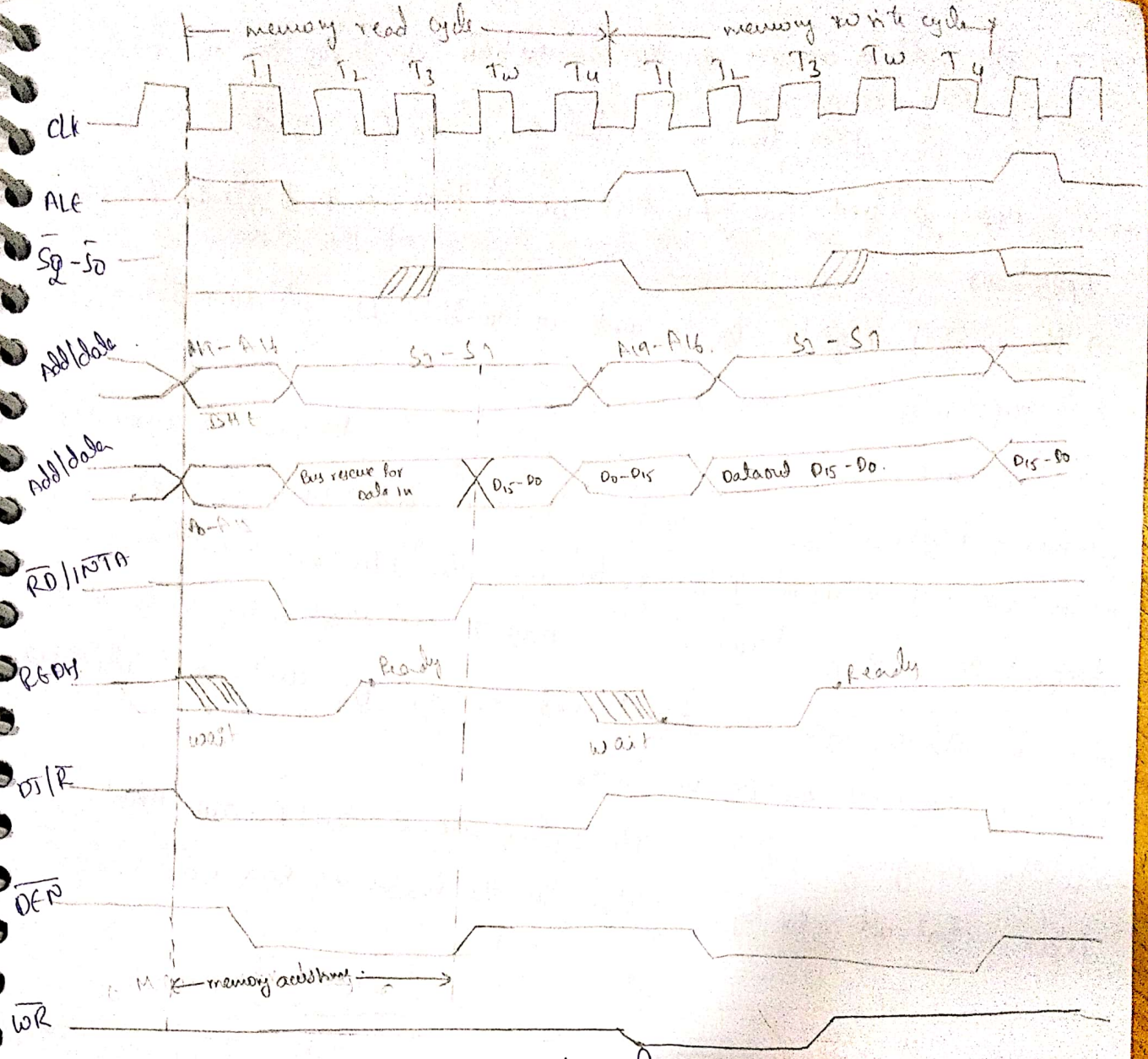
→ the negative edge of this ALE pulse is used to separate the address and the data or status information.

~~→ In maximum mode, the status lines $\overline{S_0}$, $\overline{S_1}$ and $\overline{S_2}$ are used to separate the address and the data or status information.~~

→ In maximum mode, the status lines $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$ are used to indicate the type of operation.

→ status bit $\overline{S_3}$ to $\overline{S_7}$ are multiplexed with higher order address bits and the BHE signals.

→ Address is valid during T, while the status bits S_3 to S_7 are valid during T_2 through T_4.



General bus operation of 8086.

I/O addressing capability

The 8086/8088 processor can address up to 64k I/O byte registers or 32k word registers. The limitation is that the address of an I/O device must not be greater than 16 bits in size, this means that a maximum number of 2^{16} = i.e 64k byte I/O devices may be accessed by the CPU.

→ The I/O address appears on the address lines A0 to A15 for one clock cycle (T_1). It may then be latched using the ALE signal.

→ The upper address lines (A16-A19) are at logic 0 level during the I/O operations.

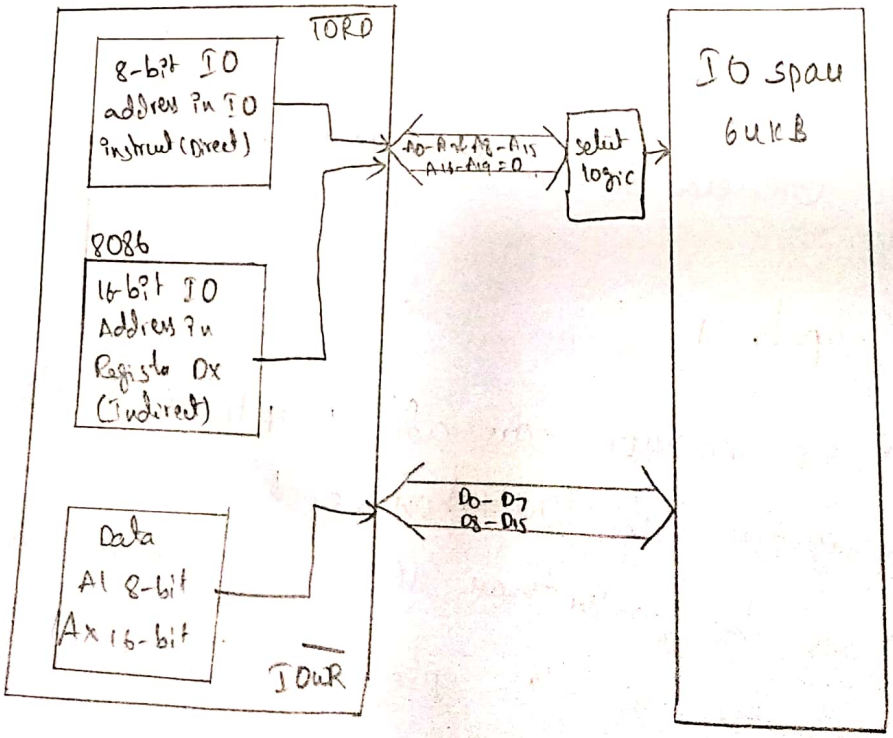
→ The 16-bit register DX is used as 16-bit I/O address pointer, with full capability to address up to 64K devices.

→ In this case, the I/O ports are addressed in the same manner as memory locations in the based addressing mode using BX.

→ In memory mapped I/O interfacing, the I/O device addresses are treated as memory locations in page 0. i.e segment address 0000H.

→ Even address byte are transferred on $D_0 - D_7$ and odd addressed byte are transferred on $D_8 - D_{15}$ lines.

→ While designing any 8-bit I/O system around 8086, care must be taken that all the byte registers in the should be even addressed.



8086 I/O Addressing

Minimum MODE 8086 system and timings

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by seperating its MN/$\overline{MX}$ pin to logic.

→ In this mode, all the control signals are given out by the microprocessor chip itself.

→ There is a single microprocessor in the minimum mode system.

→ The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices.

→ some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

M/$\overline{IO}$	$\overline{RD}$	$\overline{DEN}$	Transfer Type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	memory read
1	1	0	memory write

→ The latches are used for separating the valid address from the multiplexed address/data signals and controlled by the ALE signals generated by 8086

→ Transreceivers are the bidirectional buffers and some times they are called data amplifiers.

→ They are required to separate the valid data from the time multiplexed address/data signals.

→ They are controlled by two signals, namely $\overline{DEN}$ and DT/$\overline{R}$

→ The $\overline{DEN}$ signal indicates that the valid data is available on the data bus, while DT/$\overline{R}$ indicates the direction of data i.e from/to the processor.

→ The read cycle begins in T_1 with the assertion of the Address latch Enable (ALE) signal and M/$\overline{IO}$ signal.

→ During the negative going edge of this signal, the valid address is latched on the local bus.

→ The $\overline{BHE}$ and A0 signal address low, high or both bytes.

→ From T_1 to T_4, the M/$\overline{IO}$ signal indicates a memory or I/O operation.

→ At T_2, the address is removed from the local bus and is sent to the o/p.

→ The bus is then tristated. The Read ($\overline{RD}$) control signal is also activated in T_2.

⇒ A write cycle also begins with the assertion of ALE and the emission of the address.

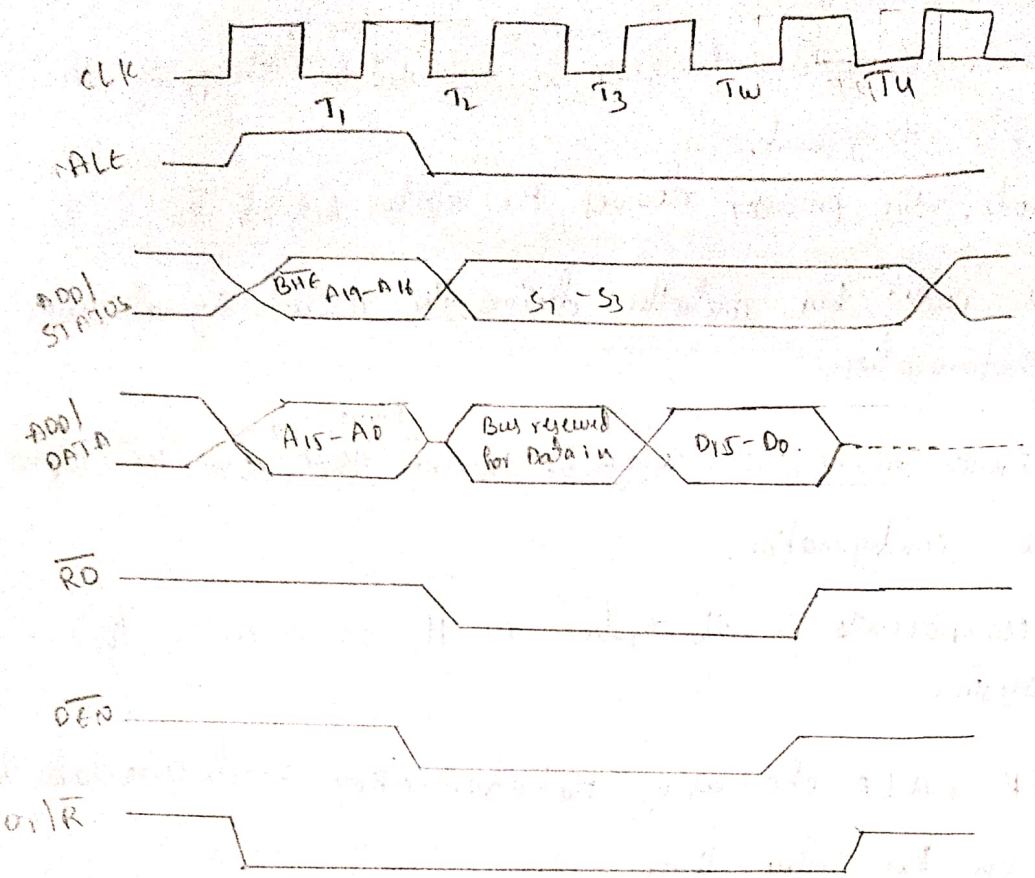
→ The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O operation.

→ In T_2, after sending the address in T_1, the processor sends the data to be written to the addressed location.

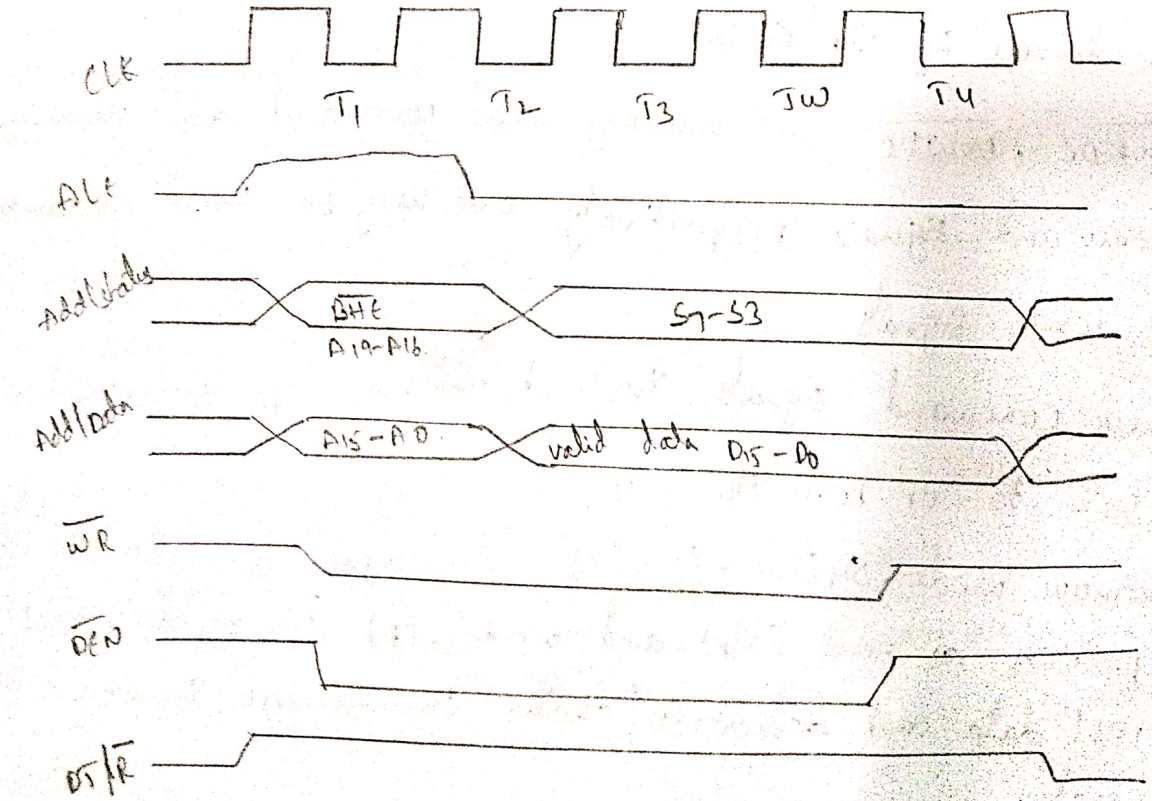
→ The data remains on the bus until the middle of T_4 state.

→ The $\overline{WR}$ becomes active at the beginning of T_2.

→ $\overline{BHE}$ and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.



Read cycle Timing Diagram for minimum mode.



Write cycle Timing Diagram for minimum mode operation.

Maximum mode 8086 System and Timings

In the maximum mode, the 8086 is operated by strapping the MN/$\overline{MX}$ pin to ground.

→ In these mode, the processor derives the status signals $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$.

→ Another chip called bus controller derives the control signals using this status information.

→ In the maximum mode, there may be more than one microprocessor in the system configuration.

→ The other components in the system are the same as in the minimum mode system.

→ $\overline{DEN}$, DT/$\overline{R}$, ALE etc using the information made available by the processor on the status lines.

→ The bus controller chip has i/p $\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$ and CLK, which are driven by the CPU.

→ The $\overline{MRDC}$, $\overline{MWTC}$ are memory read command and memory write command signals respectively and may be used as memory read and write signals.

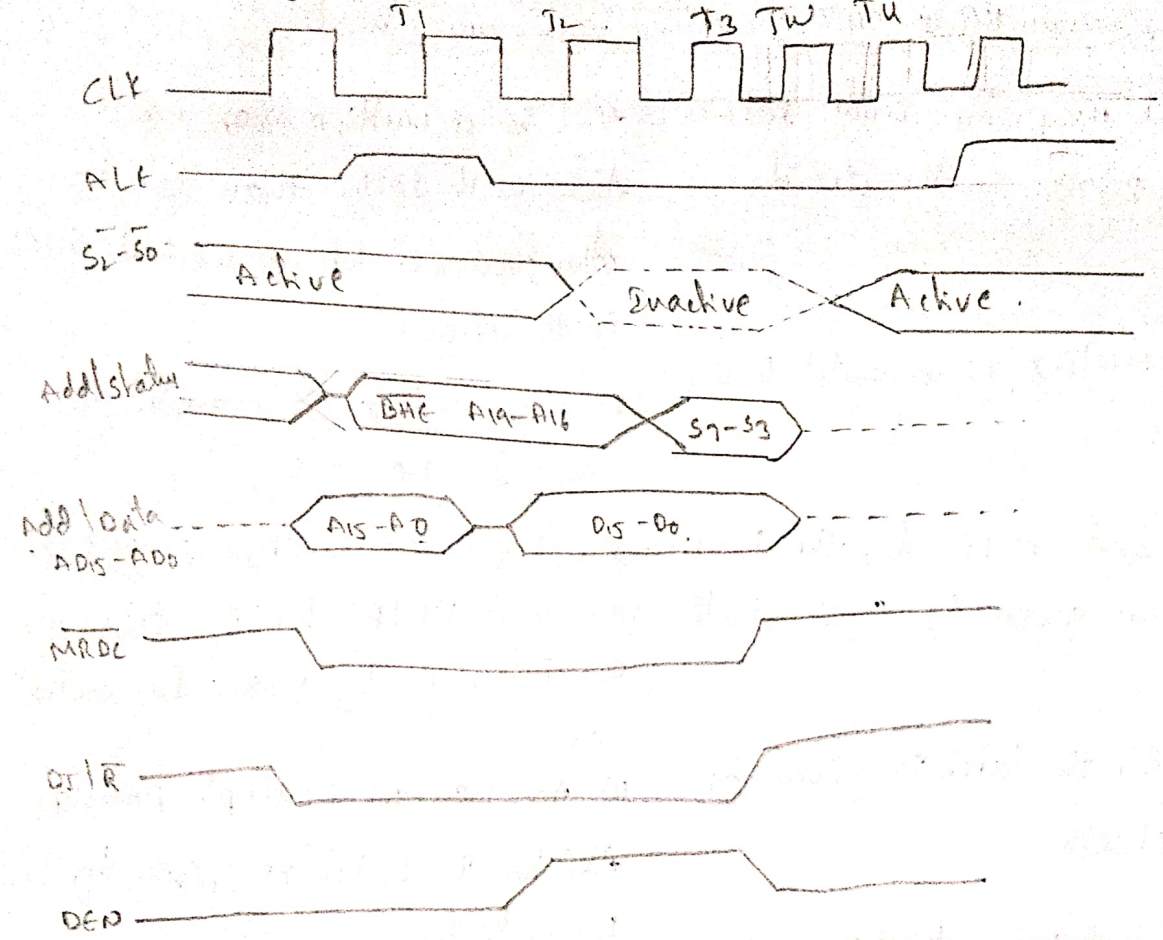
→ All these command signals instruct the memory to accept or send data to or from the bus.

→ The maximum mode system timing diagrams are also divided in two portions as read (i/p) and write (o/p) timing diagrams.

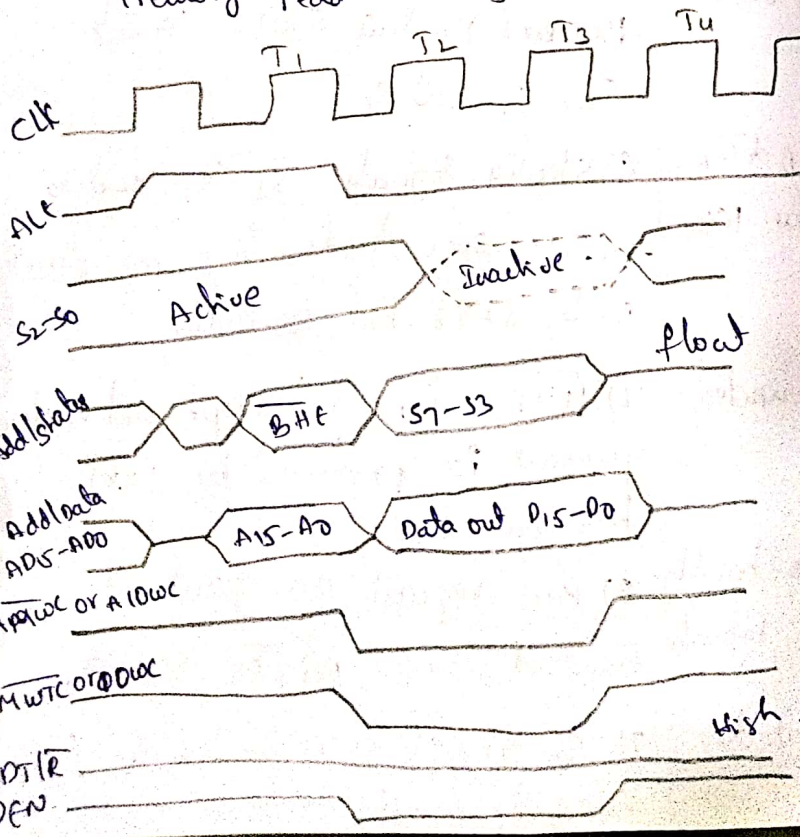
→ The address/data and address/status timings are similar to the minimum mode.

→ ALE is asserted in T_1, just like minimum mode.

→ The only difference lies in the status signals used and the available control and advanced command signals.



Memory read timing in maximum mode.



memory write timing in maximum mode.

Difference b/w minimum mode and maximum mode of 8086.

minimum mode	maximum mode.
1) It is a uniprocessor mode. 8086 is the only processor in the circuit.	1) It is a multiprocessor mode. Along with 8086, there can be other processors like 8087 and 8089 in the circuit.
2) Here MN/$\overline{MX}$ is connected to V_{cc} i.e = 1	2) Here MN/$\overline{MX}$ is connected to ground i.e = 1.
3) $\overline{DEN}$ and DT/$\overline{R}$ for the transreceivers are given by 8086 itself	3) As there are multiple processors, DEN and DT/$\overline{R}$ for the transreceivers is given by 8288 bus controller.
4) ALE for the latch is given by 8086 itself	4) As there are multiple processors, ALE for the Latch is given by 8288 bus controller
5) Direct control signals like M/$\overline{IO}$, $\overline{RD}$ and $\overline{WR}$ are produced by 8086 itself.	5) Instead of control signals, all processors produce status signals $\overline{S_2}$, $\overline{S_1}$ and $\overline{S_0}$.
6) Control signals M/$\overline{IO}$, $\overline{RD}$ & $\overline{WR}$ are decoded by a 3:8 decoder IC 74138.	6) status signals $\overline{S_2}$, $\overline{S_1}$, and $\overline{S_0}$ require special decoding are decoded by 8288 bus controller.
7) $\overline{INTA}$ for interrupt acknowledgement is produced by 8086	7) $\overline{INTA}$ for interrupt acknowledgement is produced by 8288 bus controller.
8) Bus request and grant is handled using HOLD and HLDA signals	8) Bus request and grant is handled using $\overline{RQ}$/$\overline{GT}$ signals.
9) This circuit is simpler but does not support multiprocessing	9) The circuit is more complex but supports multiprocessing.

Special Proassor Activities

Proussor Reset and initialization

→ when logic 1 is applied to the RESET pin of the microprocessor, it is reset.

→ It remains in this state till logic 0 is again applied to the RESET pin.

→ The 8086 terminates the ongoing operation on the positive edge of the reset signal.

→ when the negative edge is detected, the reset sequence starts and is continued for nearly 10 clock cycles.