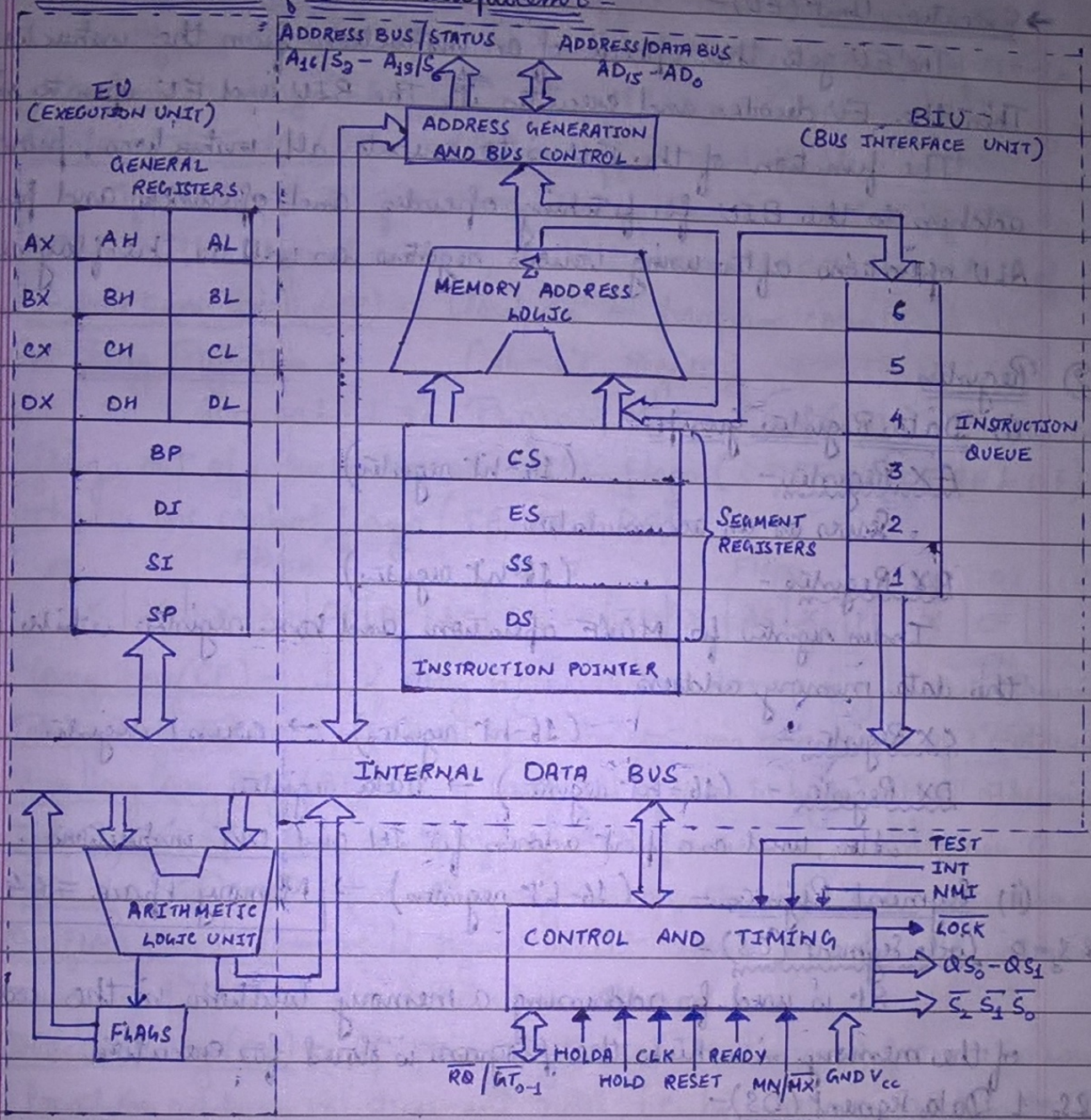
UNIT-3

8086 MICRO PROCESSOR

(20 bit address bus, 16 bit data bus)

① Architecture of 8086 Microprocessor -



The 8086 architecture has been implemented using two stage pipelining in instruction execution. The processor logic unit has been divided into Bus Interface Unit (BIU) and Execution Unit (EU).

→ Bus Interface Unit (BIU) -

It carries out all bus operations for the EU, and it is responsible for executing all external bus cycles.

Store fetched instruction codes in FIFO register which is called a queue.

BIU fills the queue when the queue becomes empty spaces of two bytes. This process is known as pipeline flush.

→ Execution Unit (EU) -

The EU gets the opcode of an instruction from the instruction queue. Then the EU decodes and executes it. The BIU and EU operate independently.

The function of the EU is to execute all instructions, provide address to the BIU for fetching opcodes and operands and perform ALU operations after using various registers as well as the flag register.

② Registers -

(i) Data Register Group -

AX Register - (16-bit register)

Serves as an accumulator.

BX Register - (16-bit register)

Index register for MOVE operation and base register while computing the data memory address.

CX Register - (16-bit register) → count register

DX Register - (16-bit register) → Data register

Also, used as a port address for IN and OUT instructions.

(ii) Segment Registers - (16-bit registers) → Memory space = 64KB

$S_4 = 0$ & $S_3 = 0$ Code Segment (CS) -

It is used for addressing a memory location in the code segment of the memory, in which the program is stored for execution.

$S_4 = 1$ & $S_3 = 1$ Data Segment (DS) -

It points to the data segment of the memory, where data is stored.

$S_4 = 0$ & $S_3 = 0$ Extra Segment (ES) - It contains another / extra data.

$S_4 = 0$ & $S_3 = 1$ Stack Segment (SS) -

It points to the stack segment of the memory, where stack data is stored.

(iii) Pointer and Index Registers - (16 bit registers)

Stack Pointer (SP) - Used to locate the stack-top address.

Base Pointer (BP) - Provide indirect access to data in a stack.

Source Index (SI) - to store offset and displacement.

If the content of SI is added with the content of DS to determine the physical address, it will be used as source address of data

Destination Index (DI) - to store offset and displacement.

If the content of DI is added with the content of ES to determine the physical address, it will be used as destination address of data

Instruction Pointer (IP) - Used as a program counter.

(iv) Flag Register - (16-bit Register)

Also called as Program status word (PSW). It has nine flags out of which six are status flags (CF, PF, AF, ZF, SF & OF) and three are control flags (TF, IF & DF).

15				Flags H				8	7			Flags L				0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF	

Carry Flag (CF) - 1, if carry is generated or borrow is generated otherwise 0.

Parity Flag (PF) - 1, if 8-bit operation contains even number of 1's otherwise 0

Auxiliary Carry Flag (AF) - 1, if carry out of lower to higher nibble otherwise 0

Zero Flag (ZF) - 1, if result of any ALU operation is zero otherwise 0.

Sign Flag (SF) - 0, if result of any ALU operation is positive number otherwise 1.

Overflow Flag (OF) - 1, if signed result cannot be expressed within the number of bits in the destination operand.

Direction Flag (DF) - 1, string bytes can be accessed from a memory ~~location~~ address in decrement order, i.e. high memory address to low memory address otherwise 0, then string bytes can be accessed from memory address in increasing order.

Interrupt Enable Flag (IF) - 1, then if maskable interrupt is enabled, otherwise 0, if maskable interrupt are disabled.

Trap Flag (TF) - 1, then a single step interrupt occurs after the next instruction executes and the program can be executed in single-step mode. The TF will be cleared by the single-step interrupt.

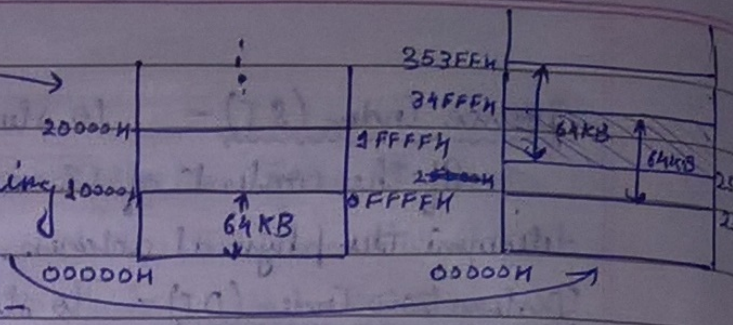
Physical Address = Segment Address $\times (10)_{16}$ + offset address

8086 Memory Address Range → 00000H – FFFFFH.

③ Memory Segmentation –

→ Segment without overlapping

→ Segment with overlapping



Advantages of segment memory –

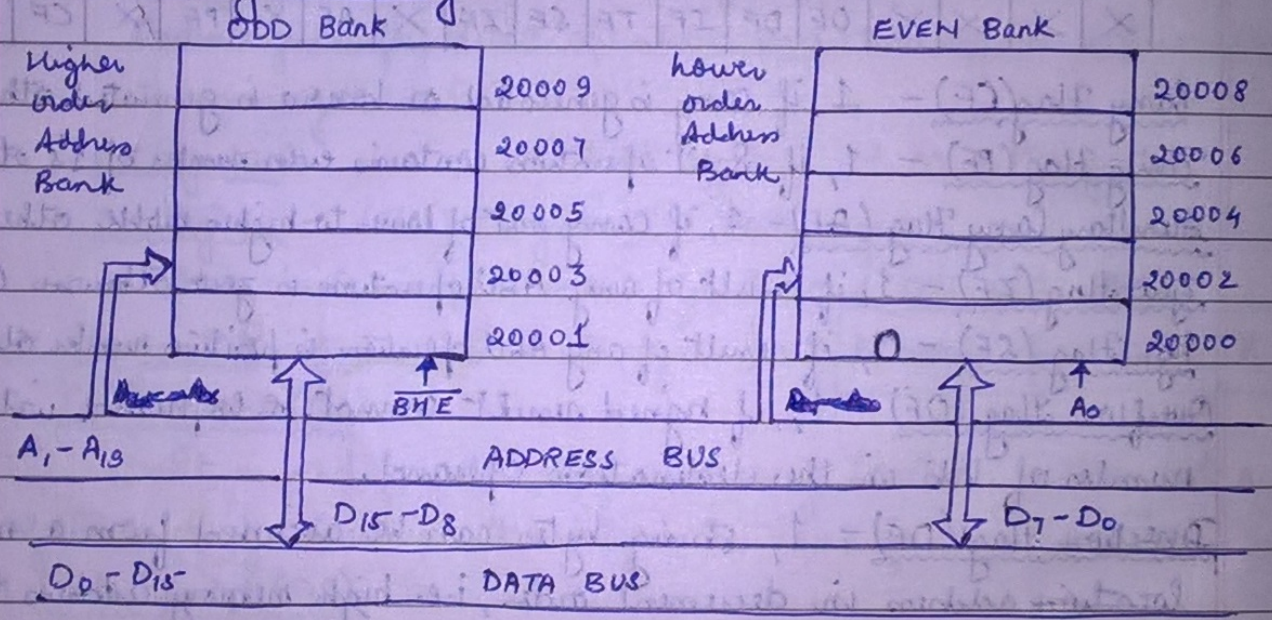
(i) Allow memory capacity to be 1MB even though the address associated with the individual instructions are 16 bits wide.

(ii) Allow the use of separate memory areas like for code, data, stack etc.

(iii) Multitasking becomes easy.

(iv) Programs are re-locatable so that programs can be run at any location in the memory.

④ 8086 Memory Addressing –



Odd-bank and Even-bank memory addressing

$\overline{BHE}$	A_0	PROCESSING
0	0	Both Bank active. 16 bit data transfer or word transfer on $AD_{15} - AD_0$
0	1	Only high bank active, one byte transfer on $AD_{15} - AD_8$
1	0	Only low bank active, one byte transfer on $AD_7 - AD_0$
1	1	No bank Active.

$D_T/\overline{R}$ → Data Transmit / Receive Receive → Data transfer from memory to processor

$\overline{DEN}$ → Data Enable → (1, enable 8286/8287 transceiver)

$\overline{BHE}/S_7$ → Bus High Enable / Status

Data can be accessed from memory in four different ways -

→ 8-bit data from even-address bank → $A_0 = 0$, $\overline{BHE} = 1$

→ 8-bit data from odd-address bank → $A_0 = 1$, $\overline{BHE} = 0$

→ 16-bit data starting from even-address bank → $A_0 = 0$, $\overline{BHE} = 0$

→ 16-bit data starting from odd-address bank → using two bus cycles.

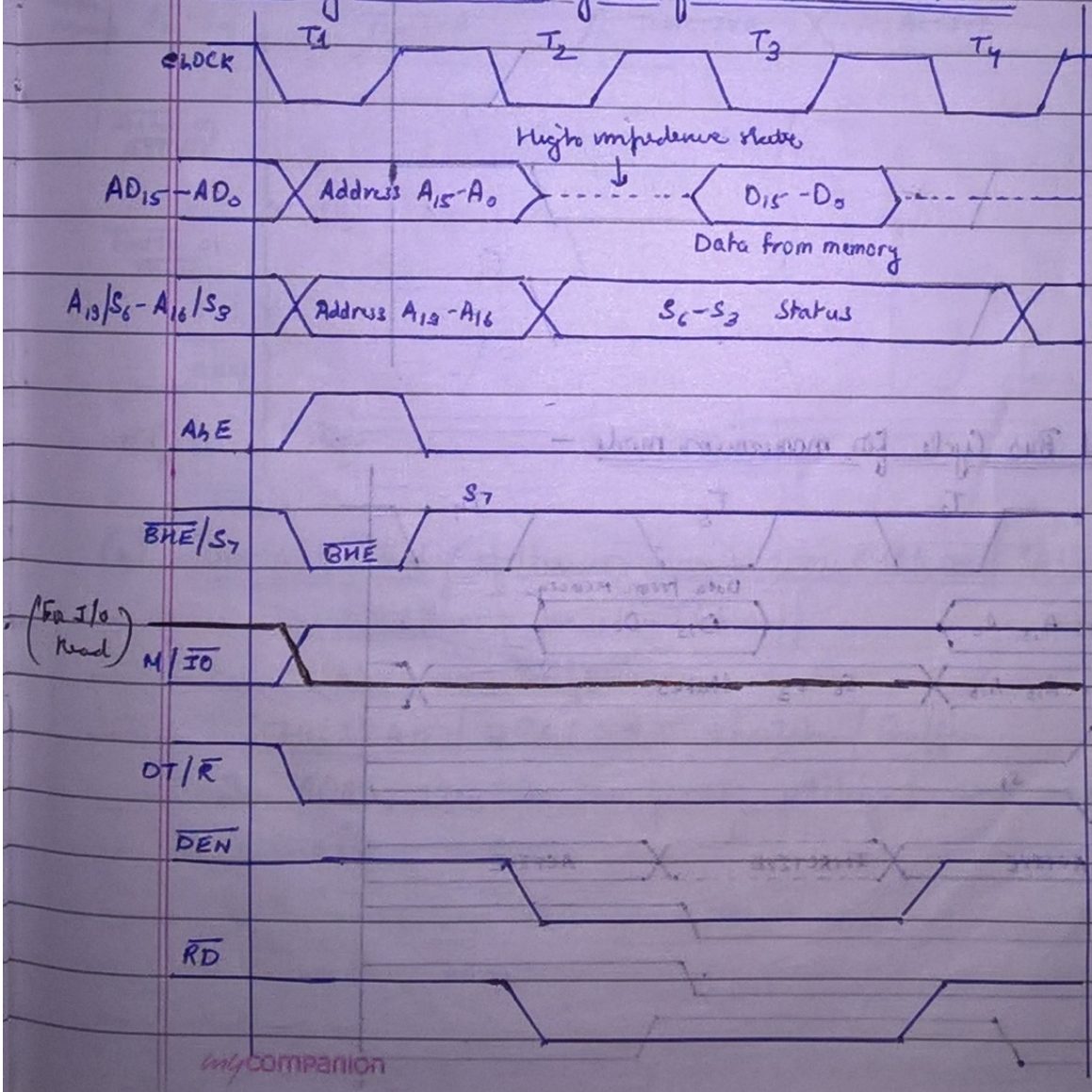
1st Cycle → $A_0 = 1$, $\overline{BHE} = 0$ & 2nd Cycle → $A_0 = 0$, $\overline{BHE} = 1$.

⑧ **Memory Read and Write Bus Cycle of 8086 -**

When any external memory or I/O devices are accessed, only four clock cycles are required to perform a read or write operation. These four clock cycles are grouped, which is called bus cycle.

(I/O Read)

→ **Memory Read Bus Cycles for minimum mode -**

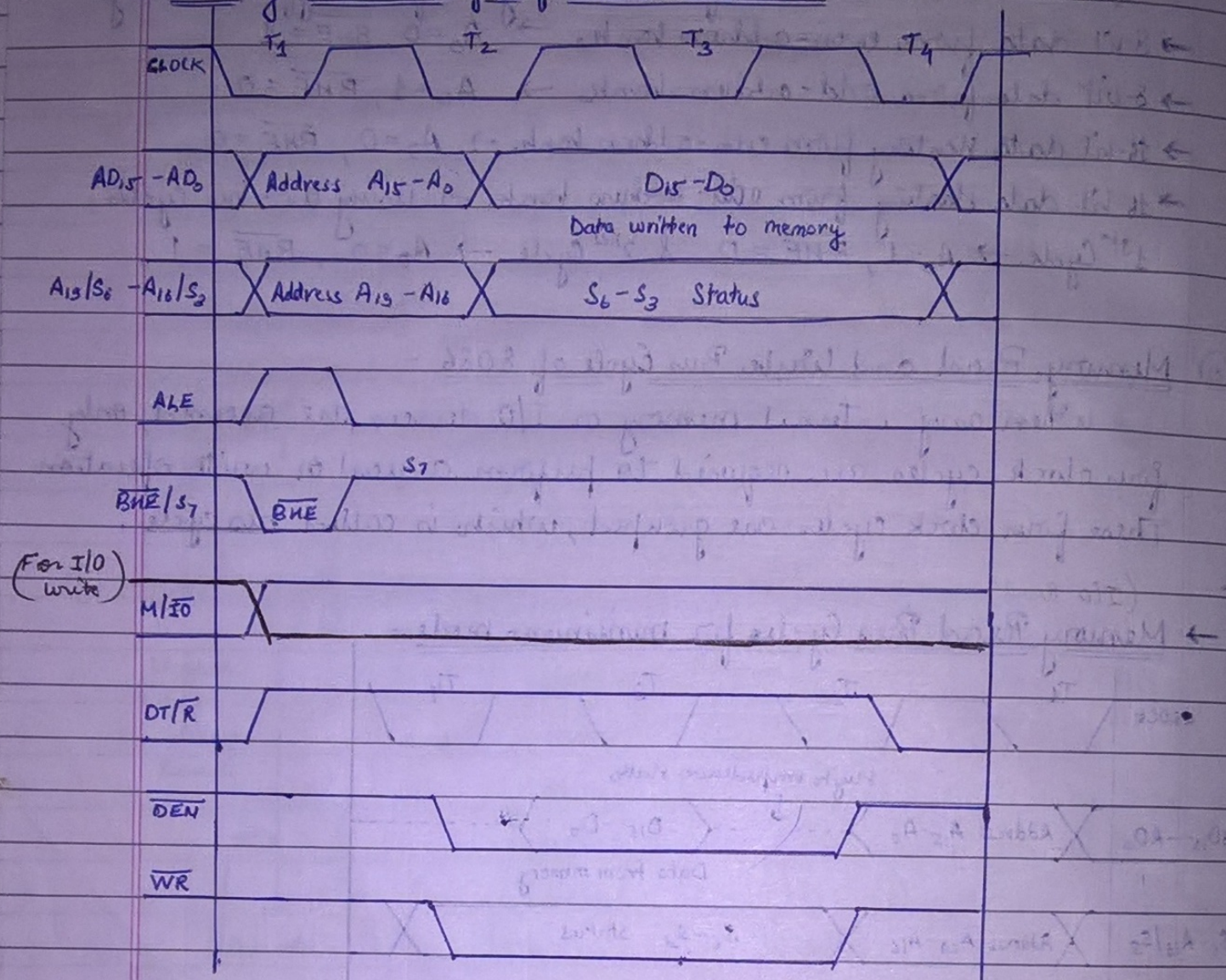


DT → Data Transmit (from processor to memory)

$\overline{MRDC}$ – Memory Read control signal

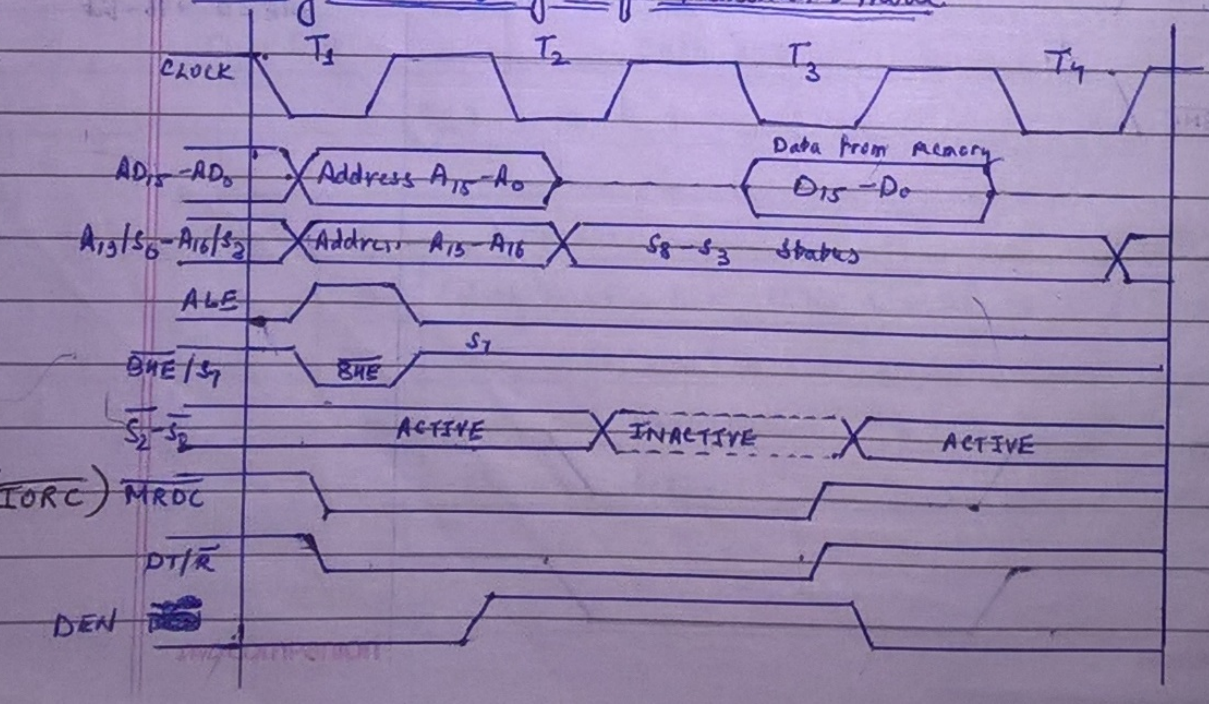
(I/O write)

→ Memory Write Bus Cycle for minimum mode –



(I/O Read)

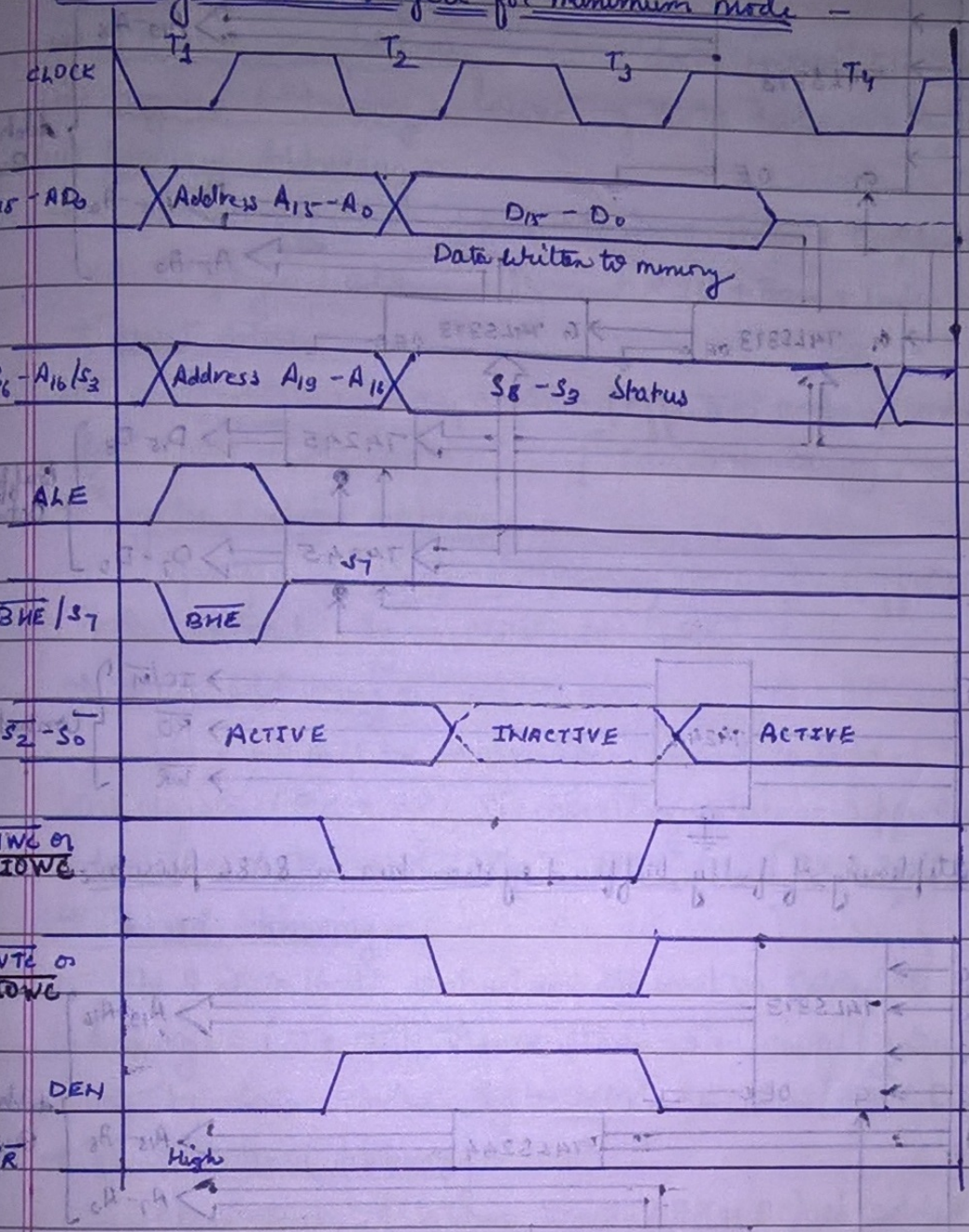
→ Memory Read Bus Cycle for maximum mode –



$\overline{MWTC}$ → Memory Write control signal

(I/O write)

→ Memory Write Bus Cycle for maximum mode –



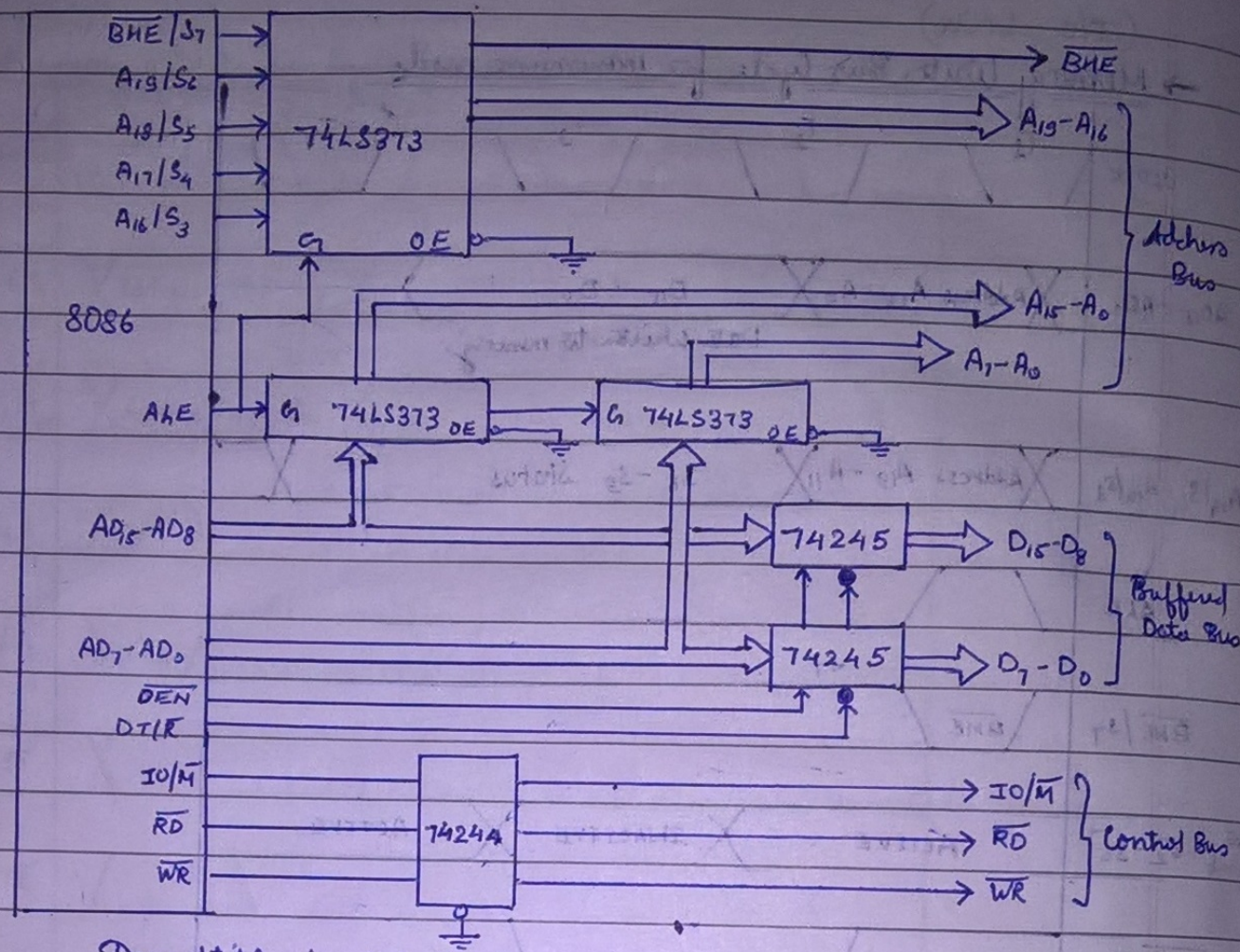
⑥ Demultiplexing of the system bus in 8086 and 8088 microprocessors –

74LS373 → latch IC's

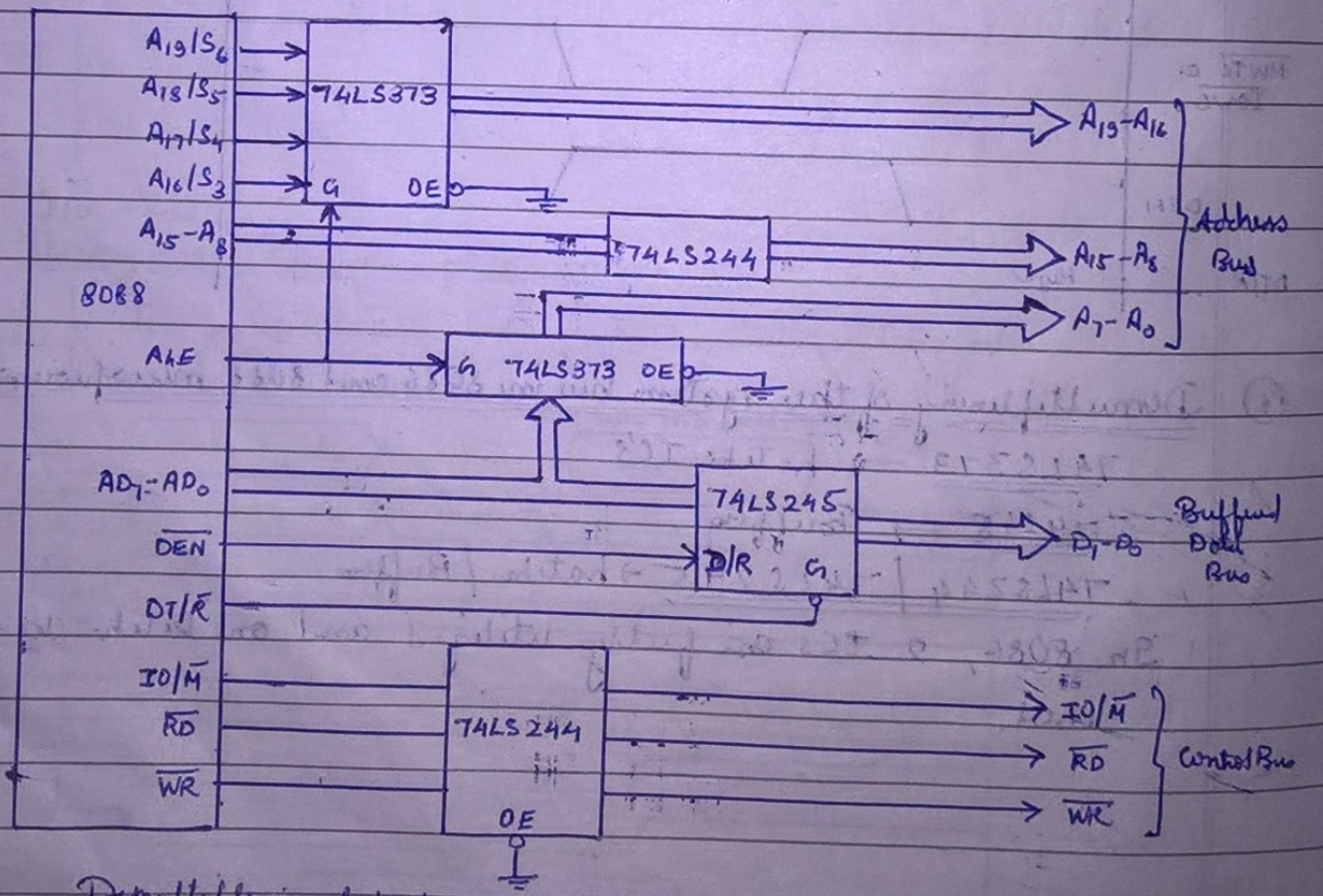
74245 → Buffers

74LS244 / 74LS245 → latches / Buffers

In 8086, 2 ICs are fully utilized and one latch is partially used.



Demultiplexing of fully buffered system bus in 8086 processor



Demultiplexing of fully buffered system bus in 8088 processor

⑦ Addressing modes of 8086 -

(i) Immediate Addressing - load data immediately. Eg - MOV BX, 7000H

(ii) Register Addressing - source register to destination register. Eg - MOV AL, BL

(iii) Memory Addressing -

16-bit Effective address = Base + Index + Displacement

20-bit physical address = Segment × 10 + Base + Index + Displacement.

→ Direct Addressing -

Specifies memory address (effective address) where data is stored.

Eg - MOV AX, [5000H]. [DS × 10 + 5000H]

→ Register Indirect Addressing -

Specifies a register containing an address (effective address), where data is stored. Eg - MOV AL, [BX].

→ Based Addressing -

The 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to the location where the data resides. Eg:- MOV AL, [BX + 8-bit DISP]

→ Indexed Addressing -

The 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to the location where the data resides. Eg - MOV AL, CS:[SI + DISP]

→ Based Indexed Addressing -

The contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to the location where the data resides. Eg, - MOV AL, [BX + DI]

→ Based Indexed with Displacement Addressing -

The 8-bit or 16-bit instruction operand is added to the content of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to the location where the data resides. Eg - MOV AL, [BX + DI + DISP]

→ String addressing mode -

DS:SI is used as a source of string and ES:DI is used to locate the destination address of the string. Eg - MOV SB (move string from source to destination)

Intersegment → different segment
Intrasegment → same segment

(iv) Branch addressing -

→ Intrasegment Direct -

EA = Contents of IP + 8- or 16-bit displacement

→ Intrasegment Indirect -

EA = Contents of a register or memory + 8- or 16-bit displacement
(Effective Branch address)

→ Intersegment Direct -

This replaces the contents of IP with a part of the instruction and the contents of CS with another part of the instruction.

→ Intersegment Indirect -

This mode replaces the contents of IP and CS with the contents of two consecutive words in memory that are referenced using any of the above data related addressing modes except the immediate and register modes.

(5) Instruction set of 8086 -

(i) Data Transfer Instructions - MOV, XCHG, LAHF (load the lower flags byte into AH), SAHF (saves AH into lower flags byte), IN, OUT, LEA (load effective address), LDS (load data segment), LES (load extra segment) XLAT (Translate byte in AL by table look-up), PUSH, POP, PUSHF (push flag word onto stack), POPF (Pop word at top of stack to flags register)

(ii) Arithmetic and logical Instructions - ADD, SUB, INC, DEC, CMP ADC (Add two operands with carry from previous add), SBB (subtract source and the carry flag bit from destination), NEG (changes the sign of an operand), MUL, IMUL (signed multiplication), DIV, IDIV (Signed division), DAA (Decimal adjustment after addition), DAS (Decimal adjustment after subtraction), AAA (ASCII adjust for addition), AAS (ASCII adjustment for subtraction), AAM (ASCII adjustment for multiplication) CBW (Convert from byte to word), CWD (Convert from word to double word) logical → NOT, AND, OR, XOR, TEST (Non-destructive logical AND) Result will not be available, but the flags are affected

(iii) Shift and Rotate Instructions -
SHL/SAL (Shift logical/Arithmetic left), SHR (Shift logical right)
$A_{15} \leftarrow A_{15}$ ← SAR (Shift arithmetic right), ROR (Rotate right without carry)
ROL (Rotate left without carry), RCR (Rotate right through carry)
RCL (Rotate left through carry).

(iv) Branch Instructions - JMP, JCXZ (Jump short if CX register is zero), Jcond (Jump on condition) [Eg - JO, JNO, JS, JNS, JP, JNP]
(overflow, sign, parity)

(v) Loop Instructions - LOOP (loop to short target), LOOPE/Z (loop to short target if Z bit set), LOOPNE/NZ (loop to short target if Z bit is clear)

→ CALL (Call a procedure), RET (Return from procedure), INT, INTO (interrupt on overflow), IRET (Return from interrupt service routine).

(vi) String instructions - MOVSW (Move string word), MOVSB (Move string byte), ~~STOS~~ STOSB (Store string byte), STOSW (Store string word), LODSB (load string byte), LODSW (load string word), CMPSB (compare string byte), CMPSW (compare string word), SCASB (Scan string byte), SCASW (Scan string word), REPEAT Instruction → { REP/REPE/REPZ (Repeat string instruction), REPNZ/REPNE (Repeat instruction while not zero)

(vii) Processor Control Instructions - CLC (Clear the carry flag), CMC (complement the carry flag), STC (set the carry flag), CLD (Clear direction flag), STD (Set direction flag), CLI (Clear interrupt flag), STI (Set interrupt flag), HLT, WAIT, LOCK (lock Bus), NOP (No operation), ESC (Escape).

~~(xii)~~ ~~(viii)~~ (viii) Flag manipulation instruction - CLD, STD, CLI, STI etc.

⑥ Assembly-language Programs of 8086 microprocessor: