

Digital Techniques and Microprocessors (313305)

Model Question Paper — Set B · Model Answers

Q.1

a) Applications of Gray code and ASCII code.

2 M

Gray code (only one bit changes between successive numbers):

1. Shaft / rotary position encoders (angular position sensing) — avoids false codes during transition.
2. Labelling the rows/columns of a K-map; error reduction in digital communication and A/D conversion.

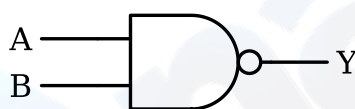
ASCII code (7-bit alphanumeric code):

1. Representing letters, digits and symbols inside computers, keyboards, printers and displays.
2. Data communication / text file exchange between computers and devices.

(½ M each)

b) NAND and EX-NOR gate.

2 M



NAND gate: $Y = (AB)'$

A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0



EX-NOR gate: $Y = A \odot B$

A	B	$Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

(Symbol ½ M + truth table ½ M for each gate)

c) Boolean laws.

2 M

1. **Commutative law:** $A + B = B + A$; $A \cdot B = B \cdot A$
2. **Associative law:** $A + (B + C) = (A + B) + C$; $A(BC) = (AB)C$
3. **Distributive law:** $A(B + C) = AB + AC$; $A + BC = (A + B)(A + C)$
4. **Complement law:** $A + \overline{A} = 1$; $A \cdot \overline{A} = 0$ (also identity $A + 0 = A$, $A \cdot 1 = A$; idempotent $A + A = A$; absorption $A + AB = A$)

(Any four, ½ M each)

d) DW and DUP directives.

2 M

DW (Define Word): reserves one or more words (2 bytes each) in memory and initialises them.

Example: NUM DW 1234H TABLE DW 0100H, 0200H, 0300H

DUP (Duplicate): used with DB/DW to reserve a number of locations with the same initial value.

Example: BUF DB 10 DUP(0) — 10 bytes, all 0; RES DW 5 DUP(?) — 5 uninitialised words.

(1 M each)

e) Assembler and linker.

2 M

Assembler (e.g. MASM / TASM): translates the assembly language source program (.ASM) into machine code and produces an object file (.OBJ); it also gives a list file and reports syntax errors.

Linker (e.g. LINK / TLINK): combines one or more object files and library routines into a single executable file (.EXE), resolves references between modules and assigns relocatable addresses.

(1 M each)

f) Use of SI, DI and IP.

2 M

1. **SI (Source Index):** holds the offset of source data in the data segment; used in indexed addressing and as source pointer (DS:SI) in string instructions.
2. **DI (Destination Index):** holds the offset of destination data; used in indexed addressing and as destination pointer (ES:DI) in string instructions.
3. **IP (Instruction Pointer):** holds the offset of the next instruction to be fetched from the code segment (CS:IP); it is updated automatically by BIU.

(SI, DI ½ M each, IP 1 M)

g) STC, CLC, CLD, STD.

2 M

Instruction	Operation	Use
STC	CF ← 1	set carry flag
CLC	CF ← 0	clear carry before ADC / RCL etc.
CLD	DF ← 0	string instructions auto-increment SI, DI
STD	DF ← 1	string instructions auto-decrement SI, DI

(½ M each)

Q.2

a) Conversions.

4 M

i) $(101101.101)_2 \rightarrow \text{decimal}$ (multiply each bit by its positional weight)

$$(101101.101)_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

$$= 32 + 0 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125$$

$$= (45.625)_{10}$$

ii) $(725)_8 \rightarrow \text{hexadecimal}$ (through binary)

Octal → Binary: write each octal digit as 3 bits.

Octal digit	7	2	5
Binary	111	010	101

$$(725)_8 = (111010101)_2$$

Binary → Hex: make groups of 4 bits from the right.

4-bit group	0001	1101	0101
Hex	1	D	5

$$\therefore (725)_8 = (1D5)_{16} \text{ (check: both = 469 in decimal)}$$

(2 M each)

b) Half subtractor design.

4 M

A half subtractor subtracts B from A (1-bit) and gives Difference (D) and Borrow (B_0).

A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

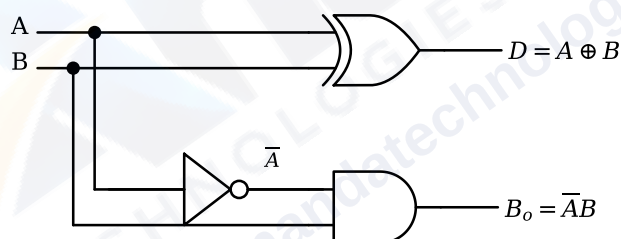
		B	
A	0	0	1
	1	1	0

$$D = A'B + AB'$$

		B	
A	0	0	1
	1	0	0

$$B_0 = A'B$$

$$D = \overline{A}B + A\overline{B} = A \oplus B \quad B_0 = \overline{A}B$$



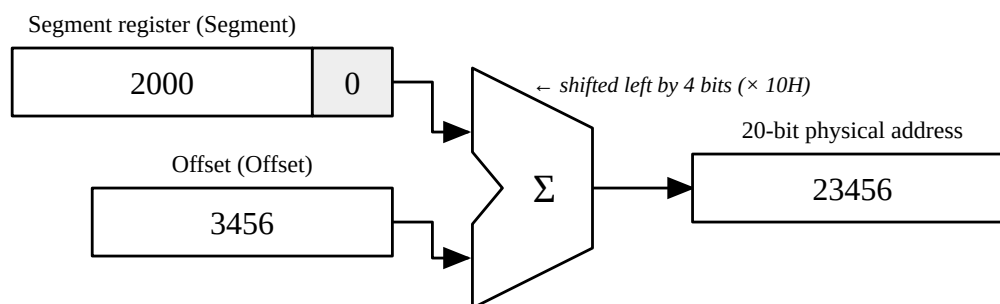
Half subtractor logic diagram

(Truth table 1 M, K-maps 1 M, equations & diagram 2 M)

c) Physical address generation.

4 M

- 8086 has 20 address lines, but all registers are 16-bit. The BIU therefore forms the 20-bit physical address (PA) from two 16-bit values: a **segment base** (in a segment register) and an **offset** (in IP, SP, BX, SI, DI, BP or given in instruction).
- The segment value is shifted left by 4 bits (multiplied by 10H) and the offset is added in the **address adder (Σ)** of the BIU: $PA = \text{Segment} \times 10H + \text{Offset}$.
- Default pairs: CS:IP (code), SS:SP / SS:BP (stack), DS:BX / SI / DI (data), ES:DI (string destination).



$$PA = \text{Segment} \times 10H + \text{Offset} = 20000H + 3456H = 23456H$$

Fig.: Physical address generation (Segment:Offset = 2000H:3456H)

i) 2000H : 3456H

Segment × 10H	=	20000H
+ Offset	=	03456H

PA	=	23456H

ii) 5A10H : 0FF0H

Segment × 10H	=	5A100H
+ Offset	=	00FF0H

PA	=	5B0F0H

(Explanation 1 M, diagram 1 M, calculations 1 M each)

d) Instructions for given operations.

4 M

Operation	Instructions	Check (simulated)
i) AL × 8 by shift	MOV CL, 03H SHL AL, CL	AL = 0BH → 58H (11 × 8 = 88 = 58H)
ii) Stack at 3000H, SP = 0FFE H	MOV AX, 3000H MOV SS, AX MOV SP, 0FFE H	SS = 3000H, SP = 0FFE H (segment register cannot be loaded with immediate data directly)
iii) Clear AX	XOR AX, AX	AX = 0000H, ZF = 1
iv) Rotate BX right through carry 2 times	MOV CL, 02H RCR BX, CL	BX = 1235H, CF = 0 → BX = 848DH, CF = 0

(1 M each)

Q.3

a) K-map with don't-care terms.

4 M

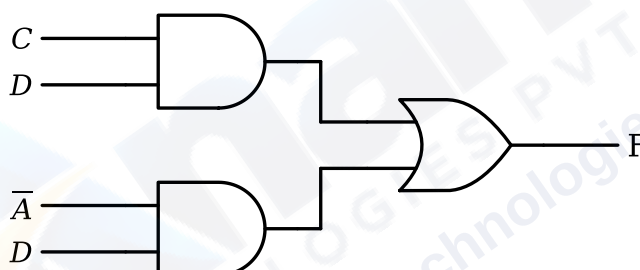
Don't-care cells (X) are taken as 1 only when they help to make a bigger group.

		CD			
AB		00	01	11	10
00		X	1	1	X
01		0	X	1	0
11		0	0	1	0
10		0	0	1	0

K-map for F (grouping 1s)

Group	Cells (minterms)	Term
Group 1	3, 7, 11, 15	CD
Group 2	1, 3, 5, 7	$\bar{A}D$

Minimal SOP: $F = CD + \bar{A}D$



Logic diagram (AND-OR)

(K-map 1 M, grouping 1 M, expression 1 M, diagram 1 M)

b) 1's and 2's complement subtraction.

4 M

$(10110101)_2 = 181$, $(01101110)_2 = 110$; expected answer $181 - 110 = 71$.

i) 1's complement method

1's complement of subtrahend $01101110 = \mathbf{10010001}$. Add it to the minuend:

```

10110101
+ 10010001
-----
101000110

```

A carry is generated $\rightarrow$ add it to the LSB (end-around carry): $01000110 + 1 = \mathbf{01000111}$. Result is positive: $+(1000111)_2 = +71$

ii) 2's complement method

2's complement of subtrahend $01101110 = 10010001 + 1 = \mathbf{10010010}$. Add it to the minuend:

```

  10110101
+ 10010010
-----
 101000111

```

Carry is generated → discard it. Result is positive: $(\mathbf{01000111})_2 = +71$

(2 M each)

c) Shift and rotate instructions.

4 M

Count is 1 or the value in CL (e.g. `MOV CL, 04H / ROL AX, CL`). Operand: 8/16-bit register or memory. Example: $AL = 96H = 1001\ 0110$, $CF = 0$ before each instruction.

Instruction	Operation	AL after	CF
<code>SHL AL, 1</code>	Shift logical left: each bit moves left, MSB → CF, 0 enters LSB (multiply by 2).	$0010\ 1100 = 2CH$	1
<code>SHR AL, 1</code>	Shift logical right: each bit moves right, LSB → CF, 0 enters MSB (unsigned divide by 2).	$0100\ 1011 = 4BH$	0
<code>SAR AL, 1</code>	Shift arithmetic right: like SHR but MSB (sign bit) is kept (signed divide by 2).	$1100\ 1011 = CBH$	0
<code>ROL AL, 1</code>	Rotate left: MSB goes to LSB and also to CF.	$0010\ 1101 = 2DH$	1
<code>ROR AL, 1</code>	Rotate right: LSB goes to MSB and also to CF.	$0100\ 1011 = 4BH$	0
<code>RCL AL, 1</code>	Rotate left through carry: MSB → CF, old CF → LSB (9-bit rotation).	$0010\ 1100 = 2CH$	1
<code>RCR AL, 1</code>	Rotate right through carry: LSB → CF, old CF → MSB.	$0100\ 1011 = 4BH$	0

(Any four with example, 1 M each)

d) Minimum mode configuration.

4 M

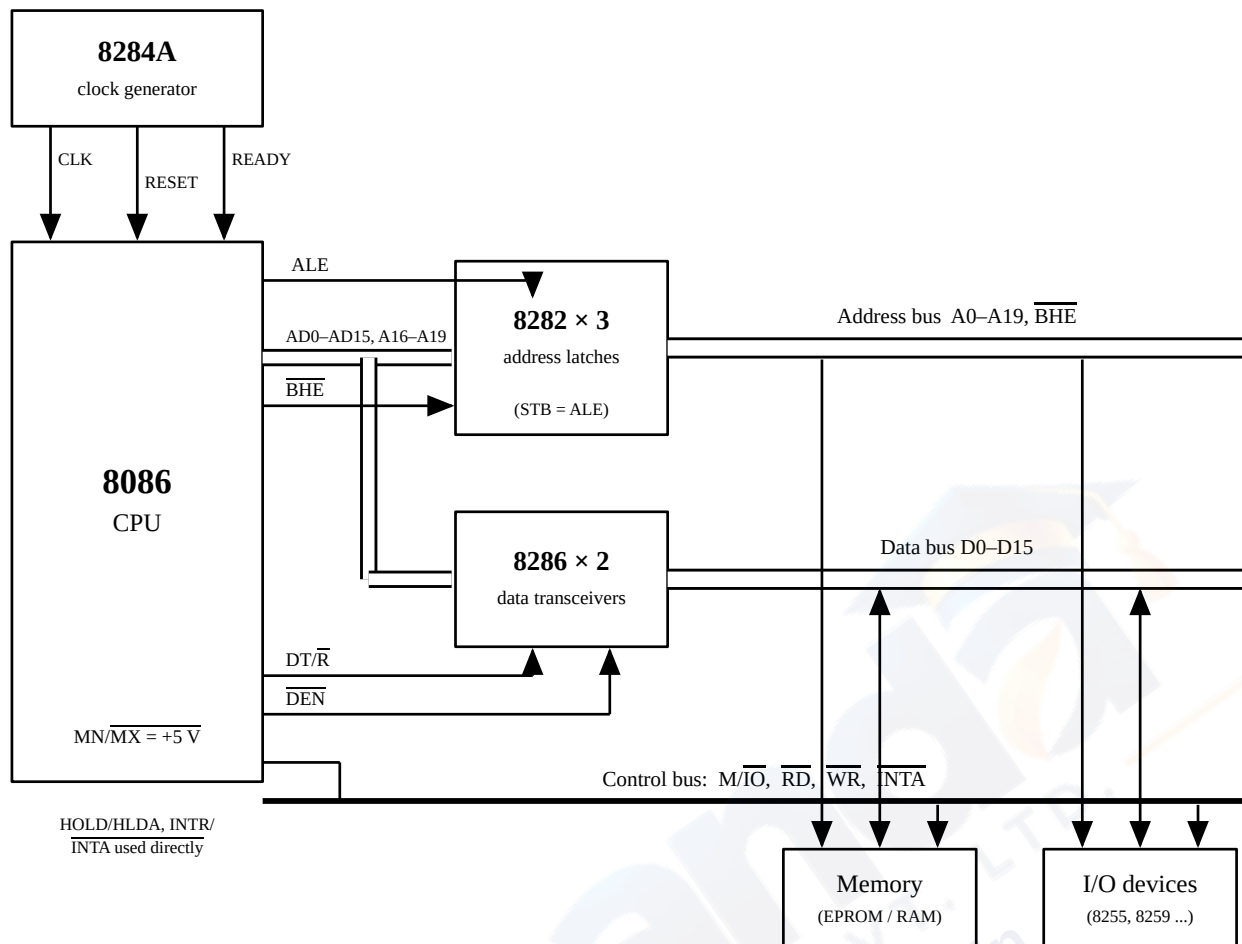


Fig.: Minimum mode configuration of 8086 ($MN/\overline{MX}$ pin = 1, single-processor system)

- $MN/\overline{MX}$ pin is connected to +5 V; the 8086 itself generates all bus control signals (single-processor system).
- **8284A clock generator** gives CLK, synchronised RESET and READY to the CPU.
- **8282 latches** (3 nos.) latch A0–A19 and $\overline{BHE}$ from the multiplexed bus using ALE as strobe → separate address bus.
- **8286 transceivers** (2 nos.) buffer the 16-bit data bus; $\overline{DT}/\overline{R}$ gives the direction and $\overline{DEN}$ enables them.
- Control signals $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$, $\overline{INTA}$, HOLD/HLDA are given directly by 8086 to memory and I/O.

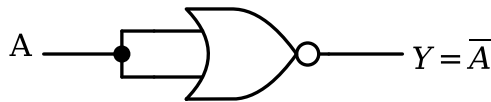
(Diagram 2 M, explanation 2 M)

Q.4

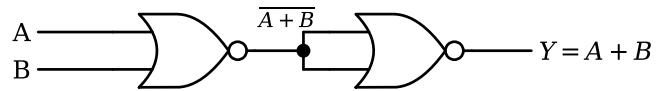
a) Basic gates using NOR gates only.

4 M

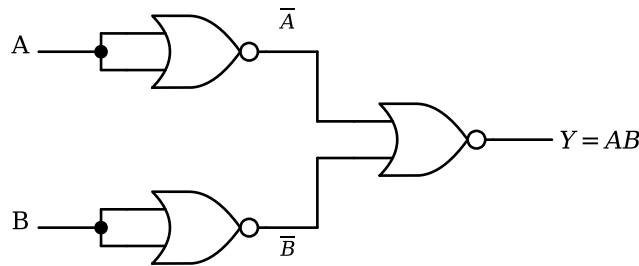
NOR is a universal gate; NOT, OR, AND and NAND can be built using NOR gates alone.



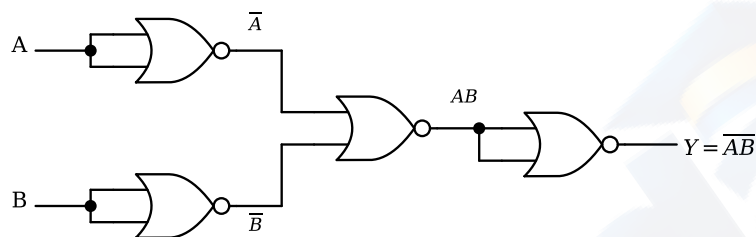
i) NOT: inputs tied together



ii) OR: NOR followed by NOR inverter



iii) AND: inverted inputs to NOR



iv) NAND: AND followed by NOR inverter

AND: $\overline{A + B} = A \cdot B$ (De Morgan) NAND: invert the AND output.

(1 M each)

b) Memory segmentation and advantages.

4 M

- The 1 MB memory of 8086 is divided into logical segments of up to 64 KB each, because the registers and offsets are only 16 bits.
- At any time four segments are active: code (CS), data (DS), stack (SS) and extra (ES). Each segment register holds the upper 16 bits of the segment's starting address.
- Address of any location = segment : offset; PA = segment $\times$ 10H + offset. Segments start on 16-byte (paragraph) boundaries and may overlap.

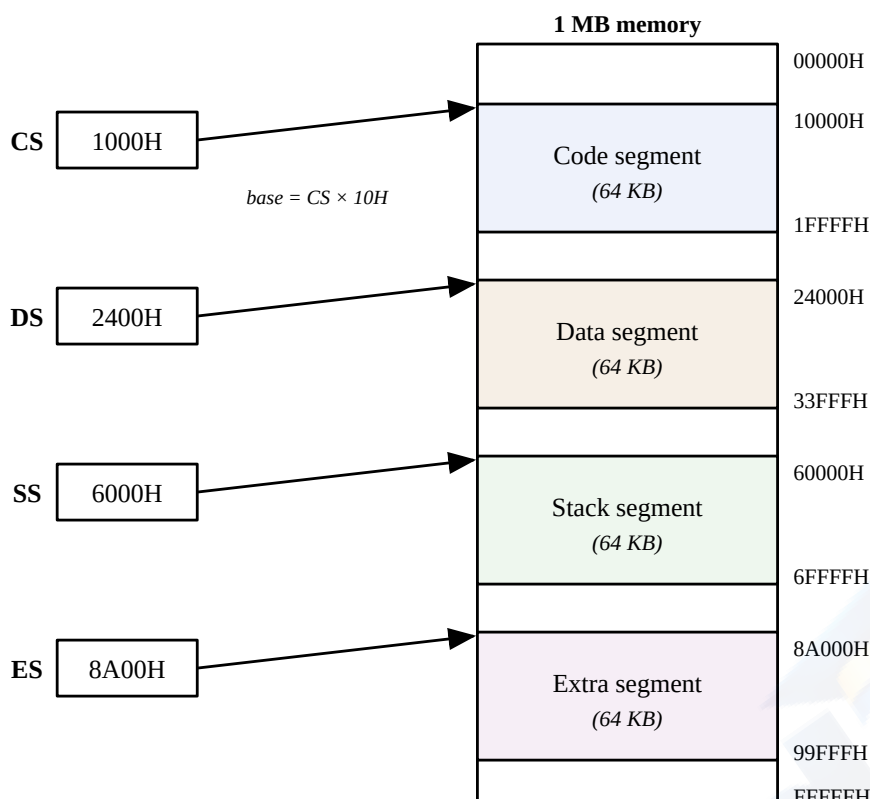


Fig.: Memory segmentation in 8086 (1 MB memory, four 64 KB segments)

Advantages:

1. Allows 1 MB memory to be addressed with 16-bit registers.
2. Separate areas for code, data and stack — easy management and a degree of protection.
3. Relocatability: a program can run at any place in memory by changing only segment registers.
4. Supports modular / multi-user programming; 64 KB per segment is enough for most programs.

(Explanation 2 M, diagram 1 M, advantages 1 M)

c) Pin functions.

4 M

1. **$\overline{RD}$ (Read, pin 32, output, active low):** indicates that the processor is reading data from memory or I/O (selected by $\overline{M/\overline{IO}}$).
2. **READY (pin 22, input, active high):** acknowledgement from slow memory / I/O that data transfer can be completed. If READY = 0, the CPU inserts wait states (T_w) between T_3 and T_4 .
3. **$\overline{M/\overline{IO}}$ (pin 28, output):** distinguishes memory and I/O operations: 1 → memory access, 0 → I/O access (minimum mode).
4. **HOLD (pin 31, input, active high):** request from another bus master (e.g. DMA controller) for the system bus. The CPU completes the current bus cycle, floats the buses and gives HLDA (hold acknowledge).

(1 M each)

d) Identify addressing modes.

4 M

	Instruction	Addressing mode
i	MOV AX, BX	Register addressing
ii	MOV CL, 25H	Immediate addressing
iii	MOV AL, [2000H]	Direct addressing
iv	MOV AX, [BX]	Register indirect addressing
v	MOV DX, [SI + 10H]	Indexed addressing (with displacement)
vi	MOV AX, [BX + DI + 05H]	Based indexed addressing with displacement
vii	ADD AL, [BP + 04H]	Based addressing (BP → stack segment)
viii	MOV [DI], AL	Register indirect addressing (destination)

(½ M each)

e) ALP — smallest number in an array.

4 M

```

DATA SEGMENT
    ARR DB 64H, 2FH, 0A9H, 47H, 1CH, 83H, 5EH, 0D2H, 39H, 71H
    SMALL DB ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
        MOV DS, AX
        LEA SI, ARR          ; SI -> first element
        MOV CX, 9            ; 9 comparisons
        MOV AL, [SI]         ; assume first is smallest
BACK:  INC SI
        CMP AL, [SI]
        JC  SKIP              ; AL < [SI] -> keep AL
        MOV AL, [SI]         ; else new smallest
SKIP:  LOOP BACK
        MOV SMALL, AL        ; store result
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START

```

Result (checked by running the program on an 8086 simulator):

Array: 64H, 2FH, A9H, 47H, 1CH, 83H, 5EH, D2H, 39H, 71H → **SMALL = 1CH**

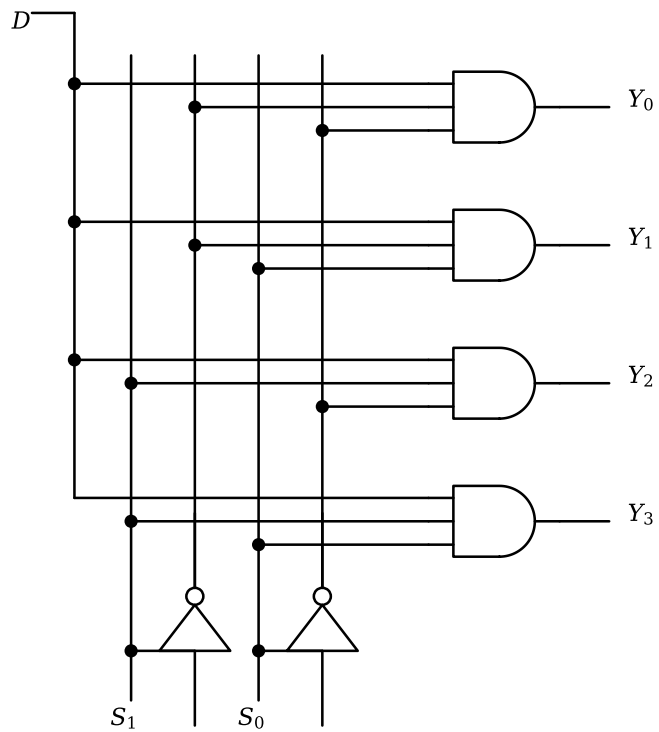
(Initialisation 1 M, comparison loop 2 M, result 1 M)

Q.5

a) 1 : 4 DEMUX and 1 : 16 DEMUX.

6 M

i) **1 : 4 demultiplexer:** sends the single data input D to one of four outputs Y_0 – Y_3 selected by S_1S_0 .

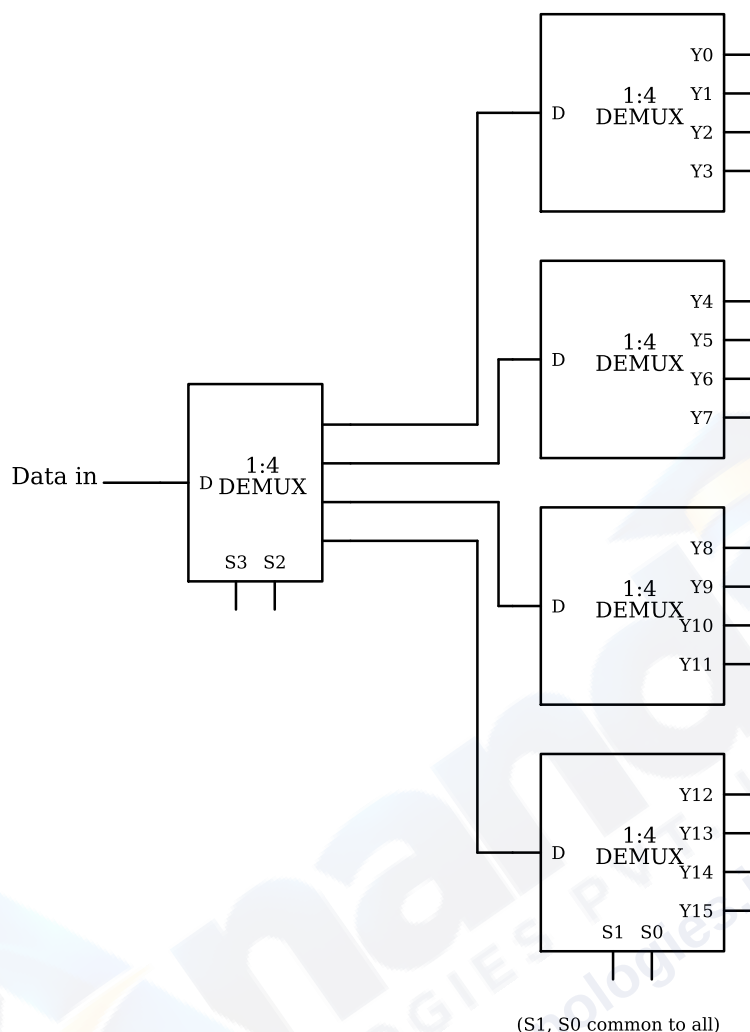


S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

1 : 4 DEMUX using logic gates

$$Y_0 = \bar{S}_1 \bar{S}_0 D, Y_1 = \bar{S}_1 S_0 D, Y_2 = S_1 \bar{S}_0 D, Y_3 = S_1 S_0 D$$

ii) 1 : 16 DEMUX using 1 : 4 DEMUX: 16 outputs need 4 select lines. The first 1 : 4 DEMUX (select S_3S_2) sends D to one of four second-level 1 : 4 DEMUX; each of these (select S_1S_0 , common) gives 4 outputs $\rightarrow Y_0-Y_{15}$. Total 5 DEMUX.



1 : 16 DEMUX using five 1 : 4 DEMUX

(i) diagram 2 M + table 1 M; ii) diagram 2 M + explanation 1 M)

b) Pin diagram of 8086 and pin functions.

6 M

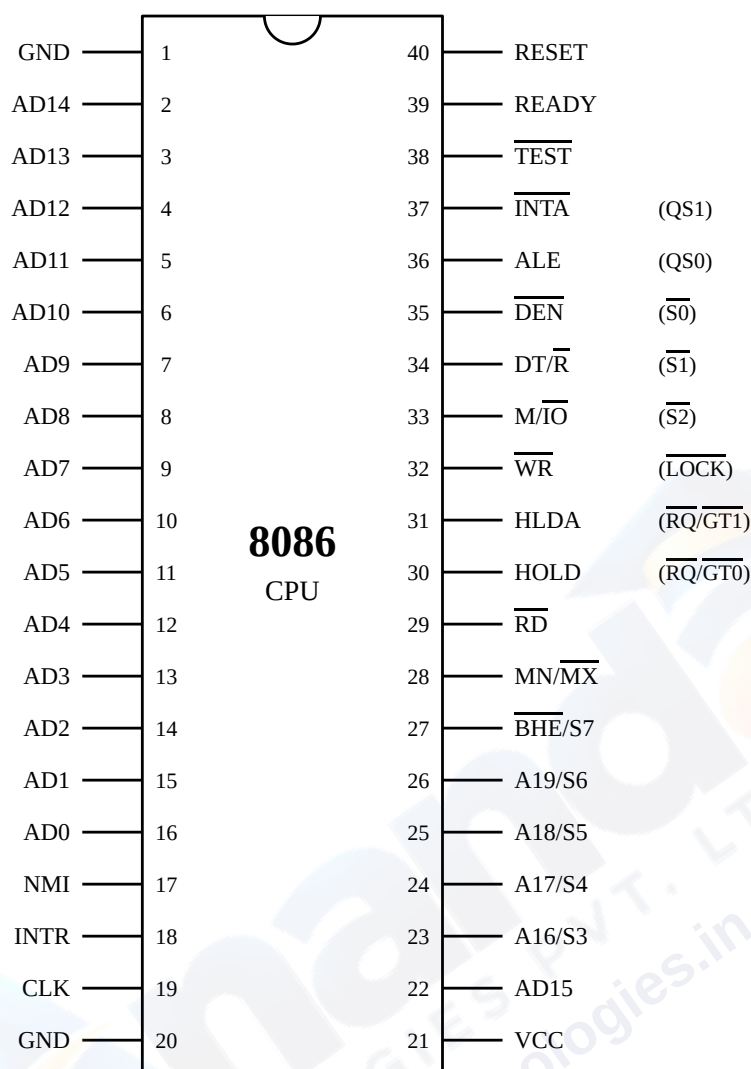


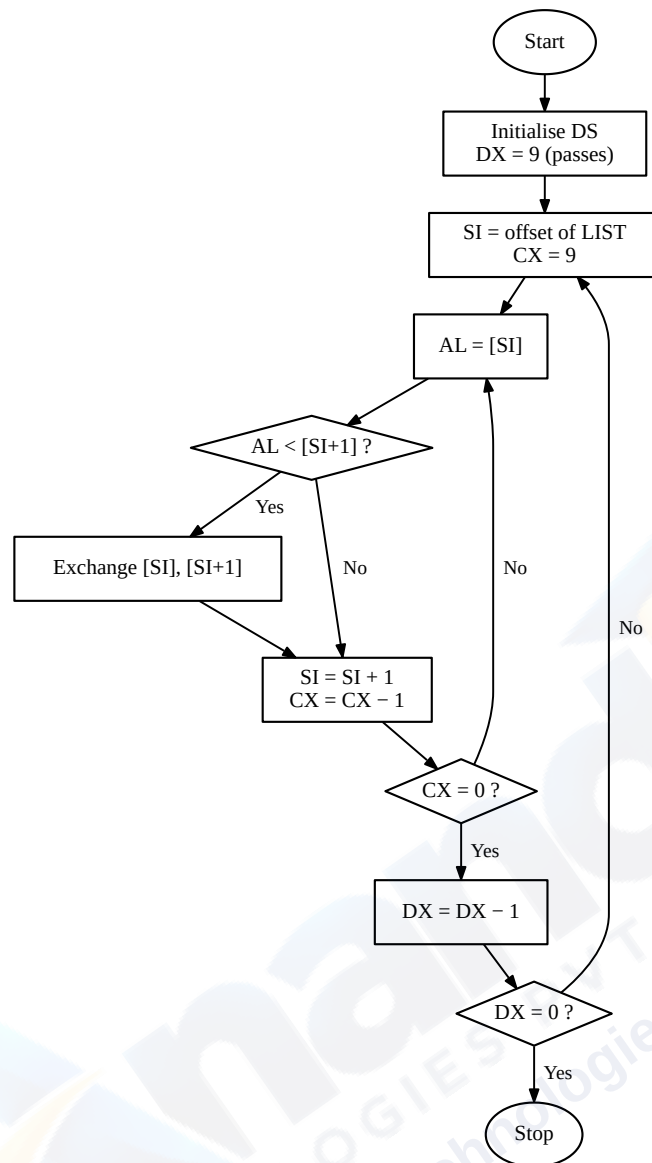
Fig.: Pin diagram of 8086 (maximum-mode functions in brackets)

1. **MN/MX (pin 33, input):** selects the mode — 1 (+5 V) = minimum mode, 0 (GND) = maximum mode.
2. **BHE/S7 (pin 34, output):** Bus High Enable — low when data is transferred on the upper byte D8–D15 (used with A0 to select odd/even bank). S7 is a status signal in later T-states.
3. **TEST (pin 23, input):** examined by the WAIT instruction; if TEST = 0 execution continues, otherwise the CPU waits (used with 8087 coprocessor).
4. **NMI (pin 17, input):** non-maskable interrupt, positive-edge triggered; cannot be disabled by software; vector type 2.
5. **INTR (pin 18, input):** maskable interrupt request, level triggered; recognised only if IF = 1; acknowledged through INTA.
6. **RESET (pin 21, input):** active high for at least 4 clock cycles; terminates present activity and restarts the CPU — CS = FFFFH, IP = 0000H, flags cleared, queue emptied (execution starts at FFFF0H).

(Pin diagram 3 M, six pins ½ M each)

c) ALP — descending order with flowchart.

6 M



Flowchart — descending order (bubble sort)

```

DATA SEGMENT
    LIST DB 36H, 0A1H, 05H, 7CH, 0E8H, 52H, 19H, 94H, 0C3H, 2BH
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START:  MOV AX, DATA
        MOV DS, AX
        MOV DX, 9                ; passes = N - 1
OUTER:  LEA SI, LIST
        MOV CX, 9                ; comparisons per pass
INNER:  MOV AL, [SI]
        CMP AL, [SI+1]
        JNC NEXT                ; AL >= next -> already descending
        XCHG AL, [SI+1]         ; swap
        MOV [SI], AL
NEXT:   INC SI
        LOOP INNER
        DEC DX
        JNZ OUTER
  
```

```

MOV AH, 4CH
INT 21H
CODE ENDS
END START

```

Result (checked by running the program on an 8086 simulator):

Before: 36H, A1H, 05H, 7CH, E8H, 52H, 19H, 94H, C3H, 2BH

After: E8H, C3H, A1H, 94H, 7CH, 52H, 36H, 2BH, 19H, 05H

For ascending order only JNC is changed to JC (or JBE).

(Flowchart 2 M, program 4 M)

Q.6

a) Combinational, sequential, latch, clock.

6 M

i) Combinational circuit: a circuit whose outputs at any instant depend only on the present inputs; it has no memory and no feedback. Built from logic gates. e.g. adders, MUX, DEMUX, decoders, encoders.

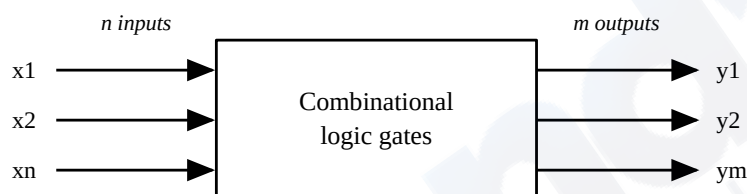


Fig.: Block diagram of combinational circuit

ii) Sequential circuit: a circuit whose outputs depend on the present inputs **and** the past outputs (present state). It has memory elements (flip-flops) and feedback, and is usually clocked. e.g. flip-flops, registers, counters.

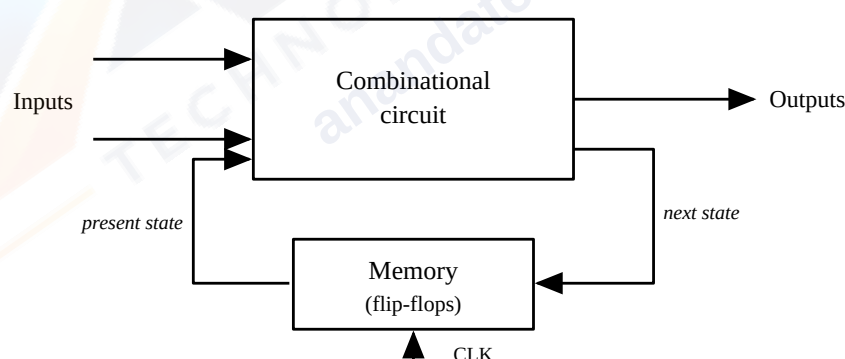
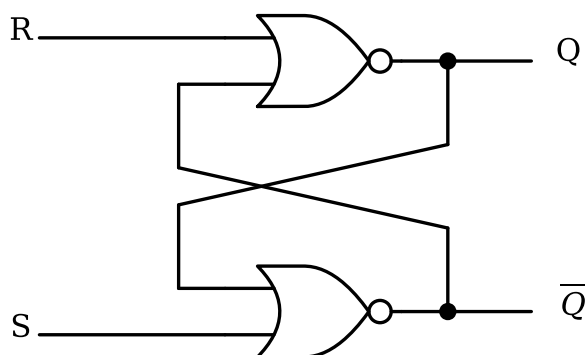


Fig.: Block diagram of sequential circuit

iii) Latch: a 1-bit memory element (bistable) whose output responds to inputs at any time (level sensitive, no clock edge). Two cross-coupled NOR (or NAND) gates form an SR latch: $S = 1$ sets $Q = 1$, $R = 1$ resets $Q = 0$, $S = R = 0$ holds the previous state.



SR latch using NOR gates

iv) Clock signal: a periodic rectangular (square) wave used to synchronise the operation of sequential circuits. Flip-flops change state at its rising (positive) or falling (negative) edge or during its level. Its frequency $f = 1/T$ decides the speed of operation.

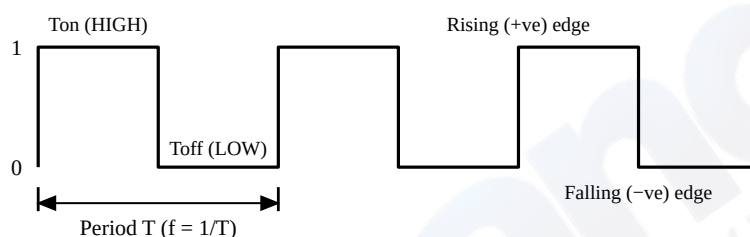


Fig.: Clock signal

(1½ M each)

b) De Morgan's theorems; simplification.

6 M

i) De Morgan's theorems

Theorem 1: $\overline{A + B} = \overline{A} \cdot \overline{B}$ **Theorem 2:** $\overline{A \cdot B} = \overline{A} + \overline{B}$

Proof of theorem 1:

A	B	A + B	$\overline{A + B}$ (LHS)	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$ (RHS)
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Proof of theorem 2:

A	B	$A \cdot B$	$\overline{A \cdot B}$ (LHS)	$\overline{A}$	$\overline{B}$	$\overline{A + B}$ (RHS)
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

In both tables LHS = RHS for every combination, hence the theorems are proved.

ii) $Y = (A + B)(A + C)$

$$\begin{aligned}
 Y &= (A + B)(A + C) \\
 &= AA + AC + AB + BC && \text{(distributive law)} \\
 &= A + AC + AB + BC && (AA = A) \\
 &= A(1 + C) + AB + BC \\
 &= A + AB + BC && (1 + C = 1) \\
 &= A(1 + B) + BC \\
 &= A + BC && (1 + B = 1)
 \end{aligned}$$

Verification by truth table:

A	B	C	$(A + B)(A + C)$	$A + BC$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

(i) 3 M, ii) simplification 2 M + verification 1 M)

c) ALP — block transfer.

6 M

i) Using string instruction (REP MOVSB)

```

DATA SEGMENT
    SRC DB 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, 0AAH
DATA ENDS
EXTRA SEGMENT
    DEST DB 10 DUP(0)
EXTRA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:EXTRA
START: MOV AX, DATA
        MOV DS, AX           ; DS -> source segment
        MOV AX, EXTRA
        MOV ES, AX           ; ES -> destination segment
        LEA SI, SRC           ; DS:SI -> source block
    
```

```
        LEA DI, DEST          ; ES:DI -> destination block
        MOV CX, 10           ; count = 10 bytes
        CLD                  ; auto-increment SI, DI
        REP MOVSB            ; move CX bytes
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

Result (checked by running the program on an 8086 simulator):

SRC = 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, AAH

DEST = 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, AAH

ii) Without string instruction

```
DATA SEGMENT
    SRC DB 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, 0AAH
    DEST DB 10 DUP(0)
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
        MOV DS, AX
        LEA SI, SRC          ; source pointer
        LEA DI, DEST         ; destination pointer
        MOV CX, 10           ; counter
BACK:  MOV AL, [SI]           ; read a byte
        MOV [DI], AL         ; write it
        INC SI
        INC DI
        LOOP BACK
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

Result (checked by running the program on an 8086 simulator):

DEST = 11H, 22H, 33H, 44H, 55H, 66H, 77H, 88H, 99H, AAH (same as SRC)

(3 M each program)

Note: This is a **model question paper prepared by Ananda Technologies** only for practice and exam preparation, based on the pattern of previous MSBTE question papers and the K-scheme syllabus. It is **not an official MSBTE paper**, and it does **not** mean that these questions will come in the exam. Answers follow standard textbook points; marks shown are indicative. Free for students — anandatechnologies.in