Reg No.:_______________ Name:__________________________

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Fifth Semester B.Tech Degree Regular and Supplementary Examination December 2022 (2019 Scheme)

Course Code: CST 307

Course Name: MICROPROCESSORS AND MICROCONTROLLERS

Max. Marks: 100 Duration: 3 Hours

PART A

(Answer all questions; each question carries 3 marks)

		Marks
1	What is pipelined architecture? How is it implemented in 8086?	3
2	Compare the architectural and signal difference between 8086 and 8088.	3
3	Write any three addressing mode of 8086 with example and write the effective address calculation in each.	3
4	Write the functions performed by PUSH and POP instructions in 8086 with appropriate diagram.	3
5	What is an interrupt vector table? Explain its structure in 8086.	3
6	Write notes on the following based on 8086: a. software interrupt b. hardware interrupt c. nested interrupt	3
7	Write the function of the following control signals in 8255. RD, WR, A_0, A_1, RESET, CS	3
8	Draw and explain the operational waveform of 8254 in MODE 0 operation.	3
9	Draw and explain the format of program status word in 8051.	3
10	Write an assembly language program for 8051 to compute *x* to the power *n* where both *x* and *n* are 8-bit numbers given by user and the result should not be more than 16 bits.	3

PART B

(Answer one full question from each module, each question carries 14 marks)

Module -1

11	Draw and discuss the internal block diagram of 8086.	14
12	With a neat sketch explain the read and write cycle timing diagram of 8086 in minimum mode.	14

Module -2

13		Write an assembly language program to find the largest and smallest number from an unordered array of 16-bit numbers. Assume the array contains 15 numbers and the starting location as 2500H. Draw the flowchart for the program.	14
14		Write an assembly language program to find the total number of even and odd numbers from an array of 16-bit numbers. Assume the array contains 20 numbers and the starting location as 5500H. Draw the flowchart for the program.	14

Module -3

15	a)	Explain the interrupt cycle of 8086.	8
	b)	Differentiate maskable and non-maskable interrupts in 8086.	6
16		Draw the architectural block diagram of 8259A and explain the function of each block.	14

Module -4

17		Explain the different modes of operation of 8255 in detail.	14
18		Draw and explain the internal architecture of 8257.	14

Module -5

19	a)	Explain the addressing modes of 8051 with example.	10
	b)	Write an assembly language program for 8051 to perform addition of two 2x2 matrices.	4
20	a)	Explain the interrupt and stack structure of 8051.	10
	b)	Write an assembly language program for 8051 to find the transpose of a 2x2 matrix.	4

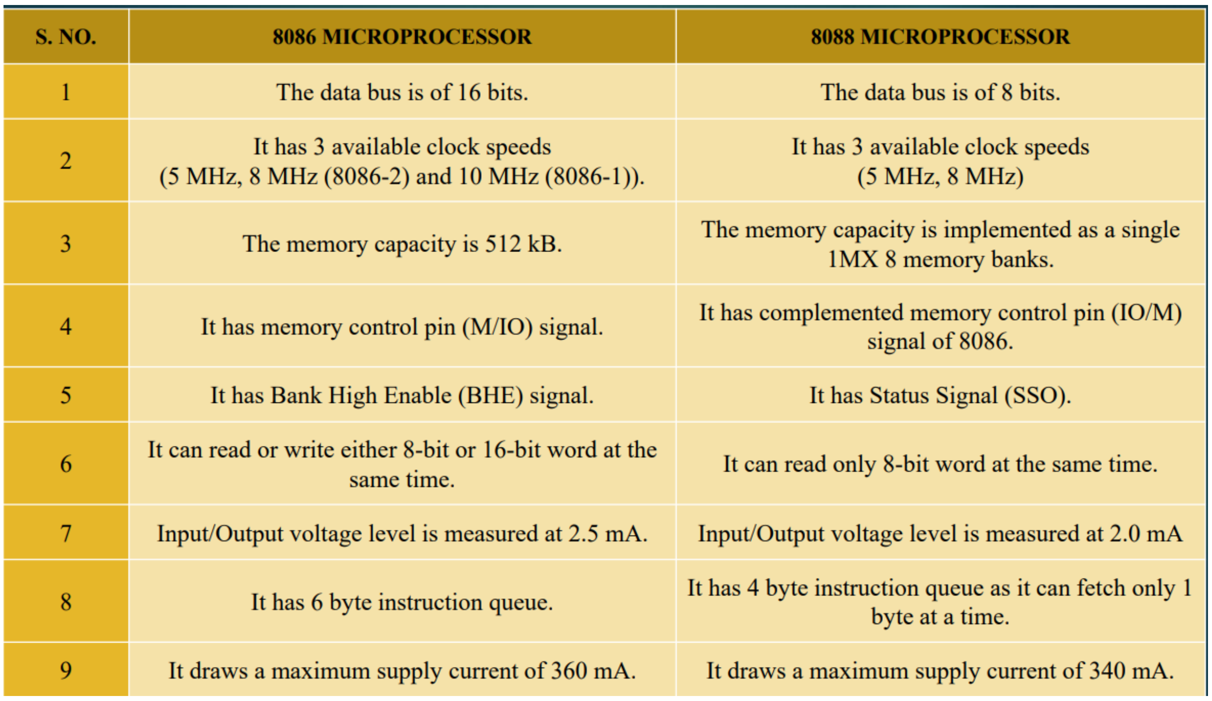
***

1.

To speed up the execution of program, the instruction fetching and execution of instructions are overlapped with each other. This process of fetching the next instruction when the present instruction is being executed is called as pipelining. In pipelining, when the nth instruction is executed, the (n+1) th instruction is fetched and thus the processing speed is increased. Pipelining has become possible due to the use of queue.

BIU (Bus Interfacing Unit) fills in the queue until the entire queue is full. When EU is busy in decoding and executing an instruction, the BIU fetches up to six instruction bytes for the next instructions. BIU restarts filling in the queue when at least two locations of queue are vacant. The execution unit (EU) is supposed to decode or execute an instruction. Decoding does not require the use of buses. These bytes are called as the pre-fetched bytes and they are stored in a first in first out (FIFO) register set, which is called as a queue.

2.



S. NO.	8086 MICROPROCESSOR	8088 MICROPROCESSOR
1	The data bus is of 16 bits.	The data bus is of 8 bits.
2	It has 3 available clock speeds (5 MHz, 8 MHz (8086-2) and 10 MHz (8086-1)).	It has 3 available clock speeds (5 MHz, 8 MHz)
3	The memory capacity is 512 kB.	The memory capacity is implemented as a single 1MX 8 memory banks.
4	It has memory control pin (M/IO) signal.	It has complemented memory control pin (IO/M) signal of 8086.
5	It has Bank High Enable (BHE) signal.	It has Status Signal (SSO).
6	It can read or write either 8-bit or 16-bit word at the same time.	It can read only 8-bit word at the same time.
7	Input/Output voltage level is measured at 2.5 mA.	Input/Output voltage level is measured at 2.0 mA
8	It has 6 byte instruction queue.	It has 4 byte instruction queue as it can fetch only 1 byte at a time.
9	It draws a maximum supply current of 360 mA.	It draws a maximum supply current of 340 mA.

3.

Direct

Here 16-bit memory address (offset) is directly specified in the instruction.

Eg: MOV AX, [5000H]

Here, data resides in a memory location in the data segment, whose effective address may be computed using 5000H as the offset address and content of DS as segment address. Thus the effective address is 10H*DS+5000H

Register Indirect

In this mode, the offset address of data is in either BX or SI or DI register. The default segment is either DS or ES.

Eg: MOV AX, [BX]

Here data is present in a memory location in DS whose offset address is in BX. The effective address of the data is given as 10H*DS+[BX]

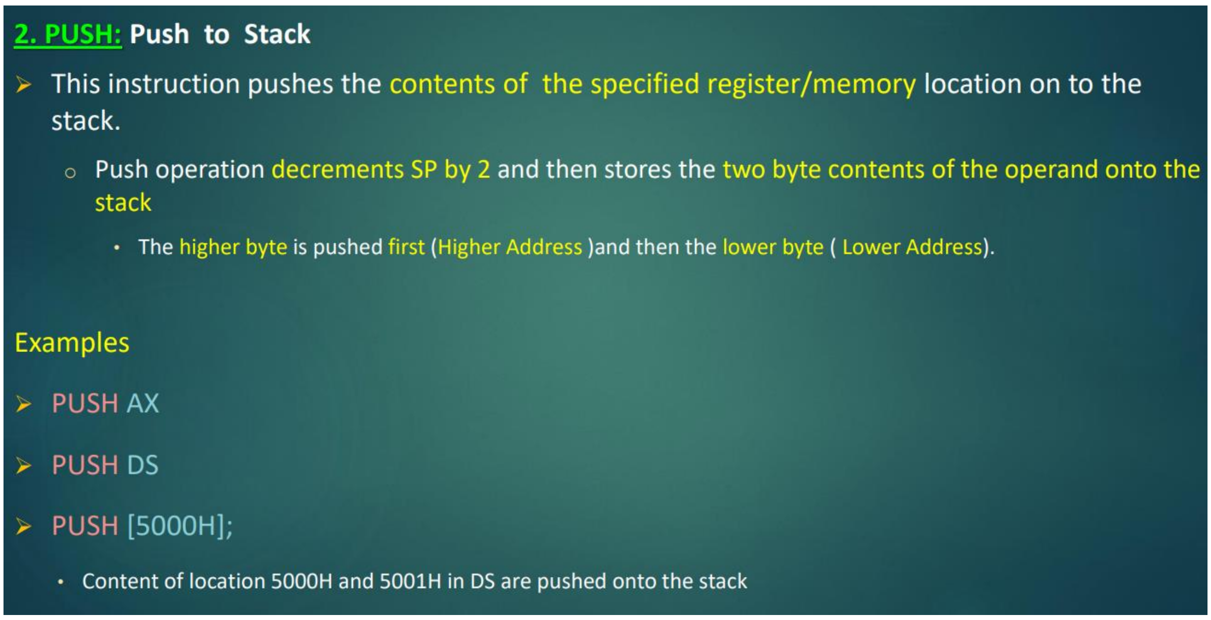
Indexed

In this addressing mode, offset of the operand is stored in one of the index registers. DS is the default segment for index registers SI and DI.

Eg: MOV AX, [SI]

Here data is available at an offset address stored in SI in DS. The effective address is computed as 10H*DS+[SI]

4.



2. PUSH: Push to Stack

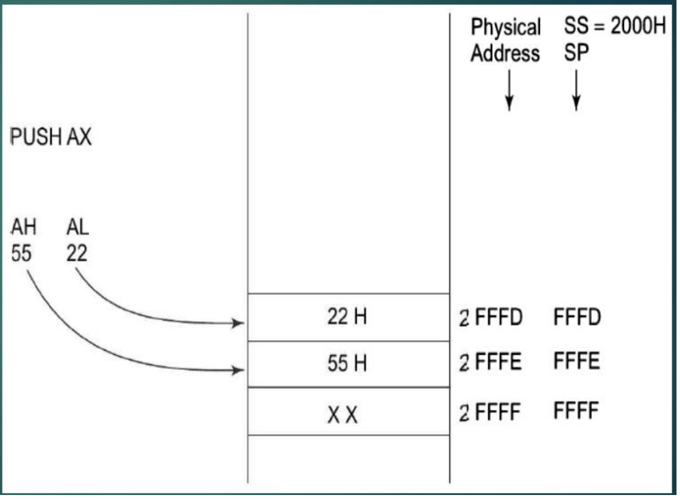
- This instruction pushes the contents of the specified register/memory location on to the stack.
  - Push operation decrements SP by 2 and then stores the two byte contents of the operand onto the stack
    - The higher byte is pushed first (Higher Address)and then the lower byte (Lower Address).

Examples

- PUSH AX
- PUSH DS
- PUSH [5000H];
  - Content of location 5000H and 5001H in DS are pushed onto the stack

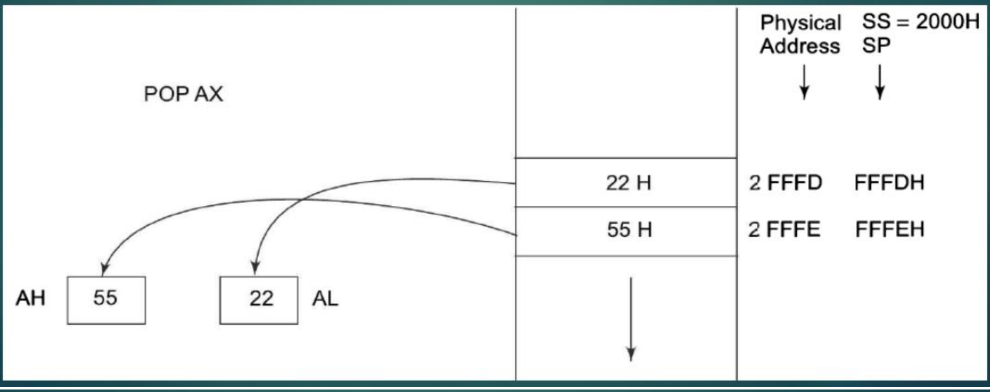
PUSH

- SS : SP points to the stack top and AH, AL contains data to be pushed.
- The sequence of operation as below:
  - Current stack top is already occupied so decrement SP by one then store AH into the address pointed to by SP.
  - Further decrement SP by one and store AL into the location pointed to by SP.
    - Thus SP is decremented by 2 and AH—AL contents are stored in stack memory
    - Contents of SP points to a new stack top.



POP

- 16-bit contents of current stack top are poped into the specified operand as follows.
  1. Contents of stack top memory location is stored in AL and SP is incremented by one
  2. Further contents of memory location pointed to by SP are copied to AH and SP is again incremented by 1
  3. Effectively SP is incremented by 2 and points to next stack top.



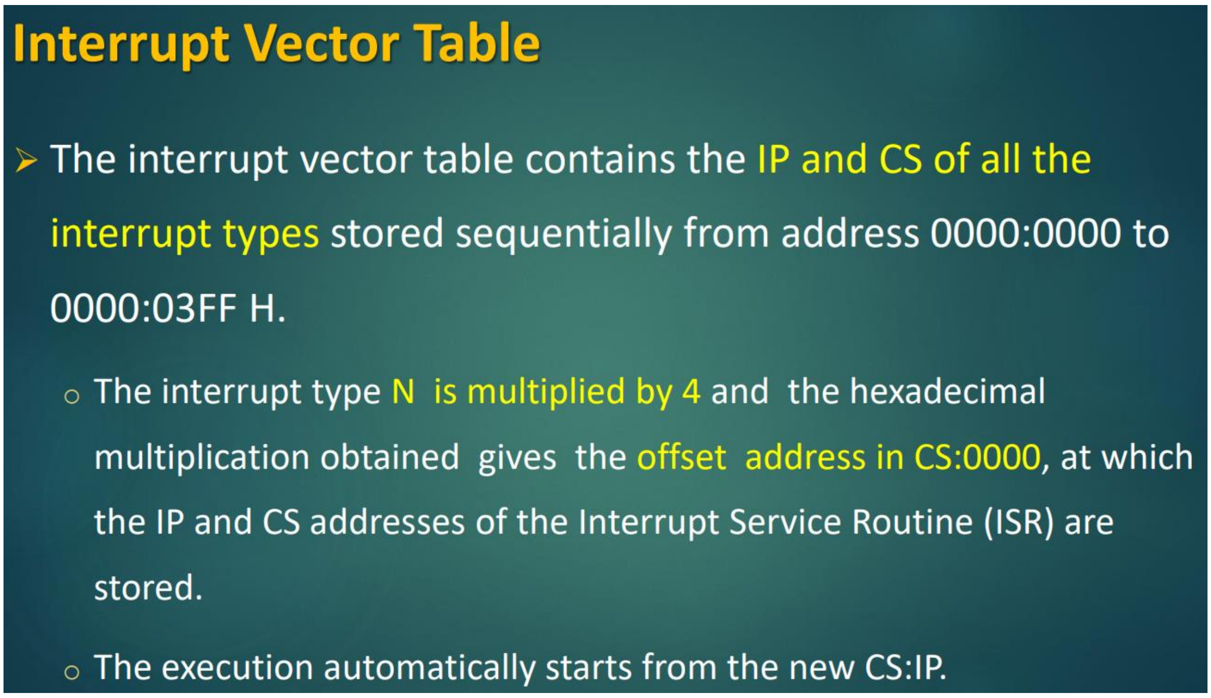
3. POP: Pop from Stack

- Loads the specified register/memory location with the contents of the memory location of which the address is formed using the current stack segment and stack pointer.
- The stack pointer is incremented by 2.
  - The POP instruction serves exactly opposite to the PUSH instruction.

Examples

- POP AX
- POP DS
- POP [5000H];

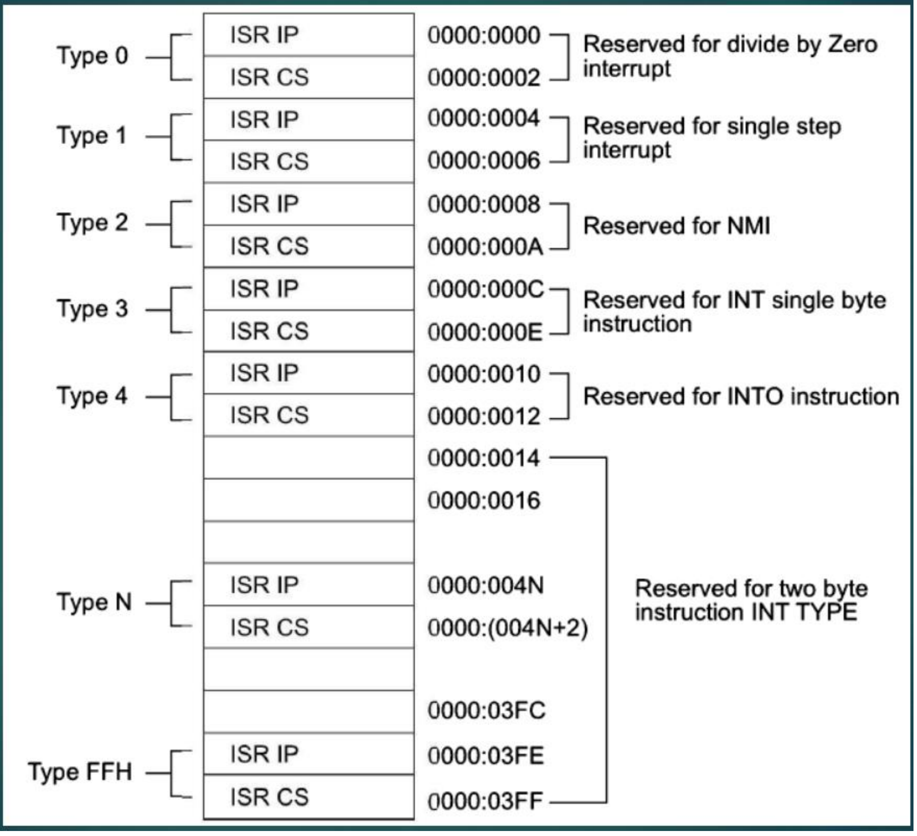
5.



Interrupt Vector Table

- The interrupt vector table contains the IP and CS of all the interrupt types stored sequentially from address 0000:0000 to 0000:03FF H.
  - The interrupt type N is multiplied by 4 and the hexadecimal multiplication obtained gives the offset address in CS:0000, at which the IP and CS addresses of the Interrupt Service Routine (ISR) are stored.
  - The execution automatically starts from the new CS:IP.

IVT



6.

Internal interrupt.

• Generated internally by the processor circuit, or by the execution of an interrupt instruction.

• Examples- Divide by zero interrupt, Overflow interrupt, Interrupts due to INT instructions, etc.

External interrupt

• An external device or a signal interrupts the processor from outside (Interrupt is generated outside the processor).

• Example- a keyboard interrupt

7.

RD# Input line driven by the microprocessor. A low to indicate read operation to 8255

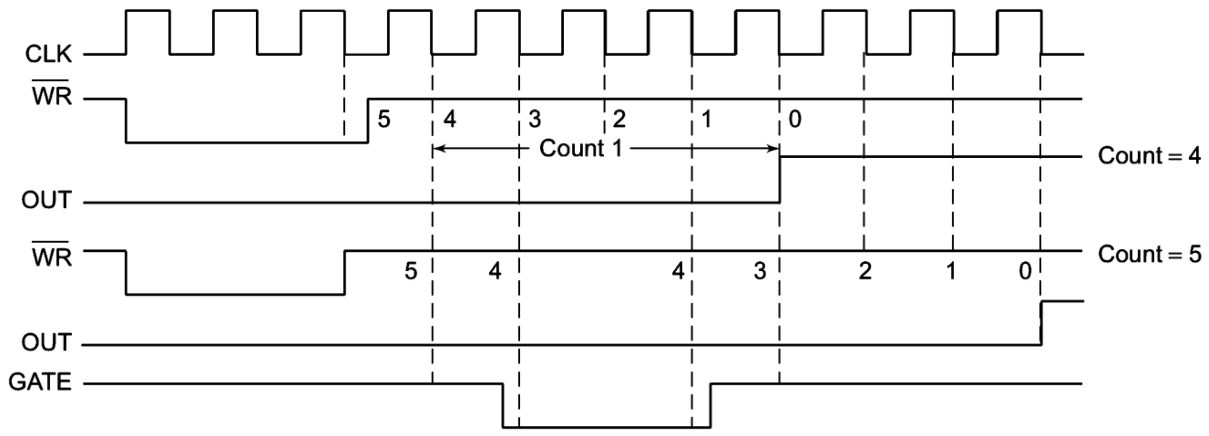
WR# Input line driven by the microprocessor. A low on this line indicates write operation

A1—A0 Used for addressing any one of the 4 registers -3 ports and a control word register . (A1 — A0) with RD# , WR# and CS# form the operations for 8255.

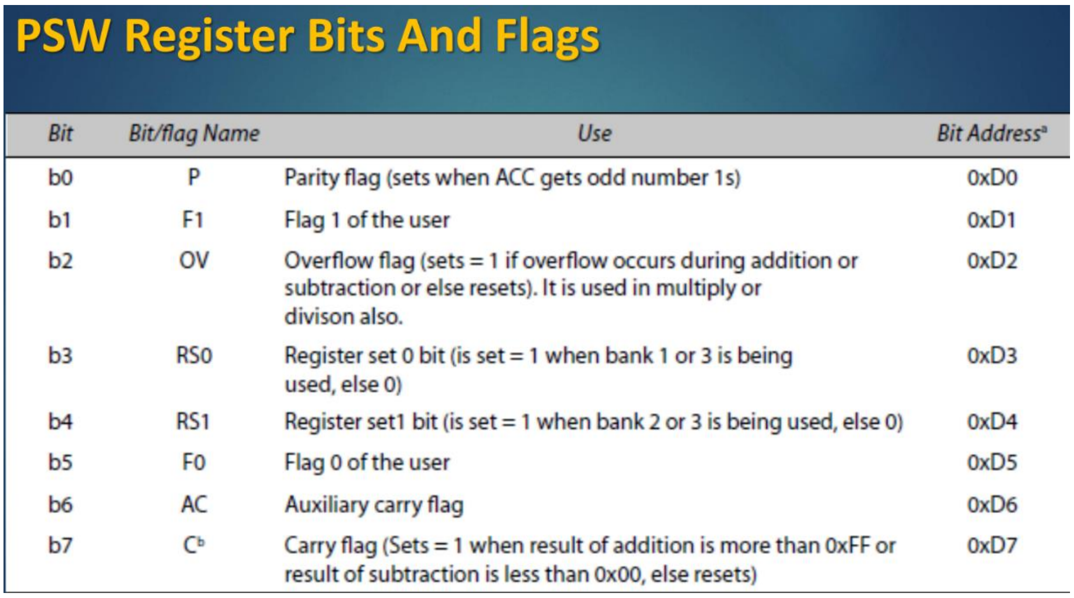
CS# This is a chip select line. If this line goes low, it enables 8255 to respond to RD# and WR# signals

RESET High on this line clears the control word register of 8255. All ports are set as input ports by default after reset

8.



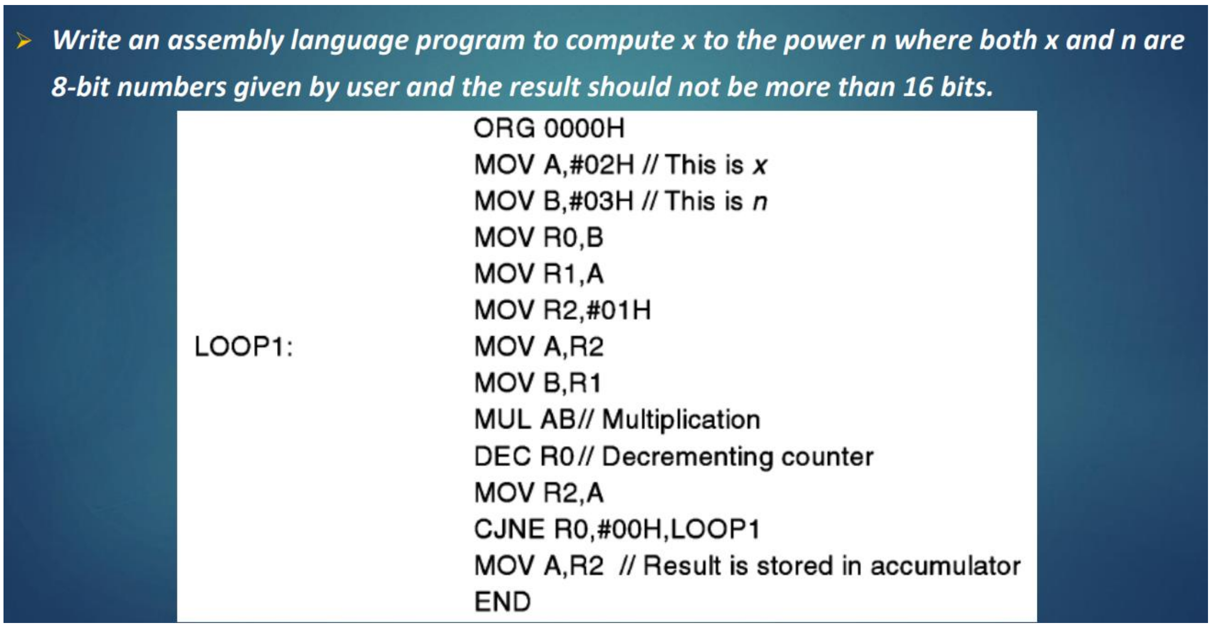
9.



PSW Register Bits And Flags

Bit	Bit/flag Name	Use	Bit Address[a]
b0	P	Parity flag (sets when ACC gets odd number 1s)	0xD0
b1	F1	Flag 1 of the user	0xD1
b2	OV	Overflow flag (sets = 1 if overflow occurs during addition or subtraction or else resets). It is used in multiply or divison also.	0xD2
b3	RS0	Register set 0 bit (is set = 1 when bank 1 or 3 is being used, else 0)	0xD3
b4	RS1	Register set1 bit (is set = 1 when bank 2 or 3 is being used, else 0)	0xD4
b5	F0	Flag 0 of the user	0xD5
b6	AC	Auxiliary carry flag	0xD6
b7	C[b]	Carry flag (Sets = 1 when result of addition is more than 0xFF or result of subtraction is less than 0x00, else resets)	0xD7

10.

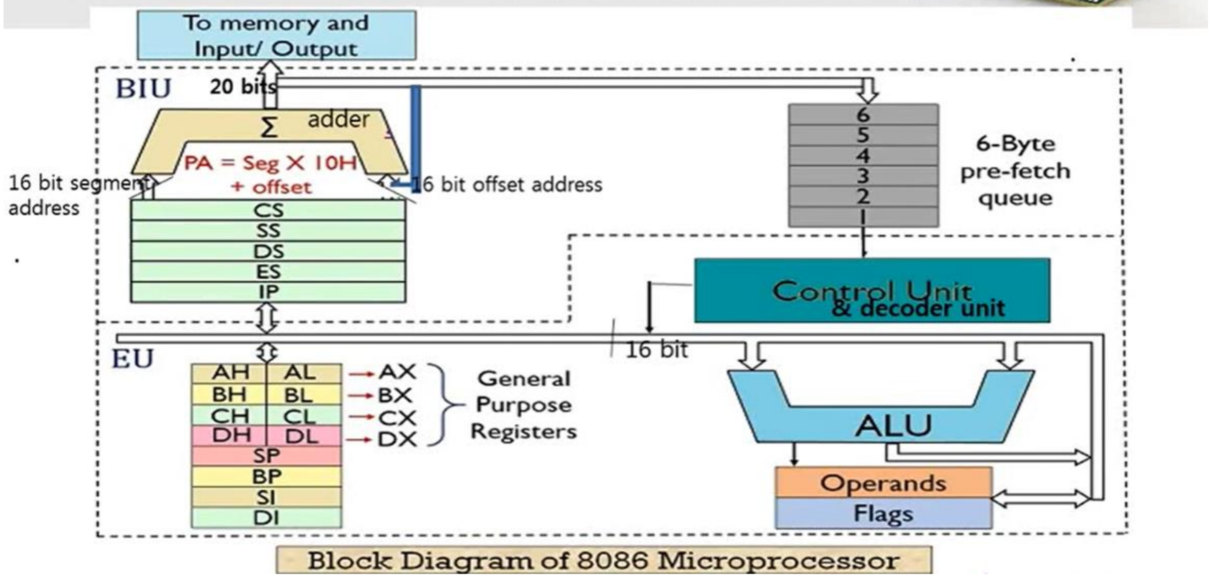


➢ ***Write an assembly language program to compute x to the power n where both x and n are 8-bit numbers given by user and the result should not be more than 16 bits.***

```
                ORG 0000H
                MOV A,#02H // This is x
                MOV B,#03H // This is n
                MOV R0,B
                MOV R1,A
                MOV R2,#01H
LOOP1:          MOV A,R2
                MOV B,R1
                MUL AB// Multiplication
                DEC R0// Decrementing counter
                MOV R2,A
                CJNE R0,#00H,LOOP1
                MOV A,R2  // Result is stored in accumulator
                END
```

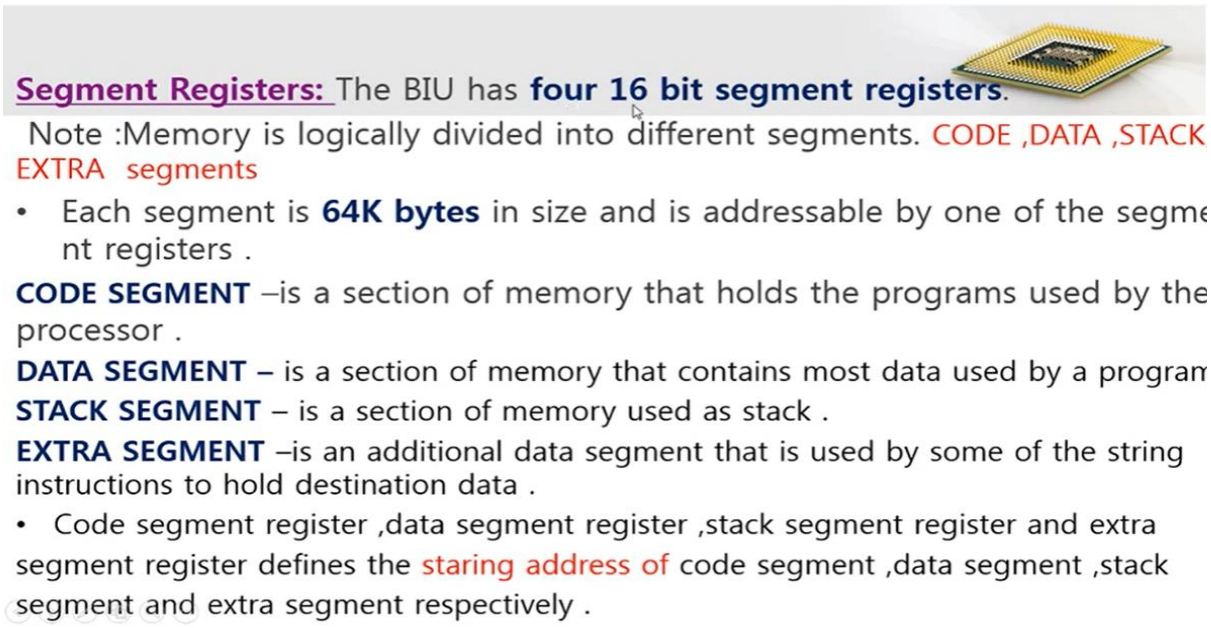
11.

8086 INTERNAL ARCHITECTURE



Block Diagram of 8086 Microprocessor

- 8086 CPU is divided into two independent functional parts ,the bus interface unit or BIU and execution unit or EU .
- Dividing the work between these two units speed up processing .
- The BIU handles all transfers of data and addresses on the buses for the execution unit .
- The execution unit of the 8086 tells the BIU where to fetch instructions or data from .



Segment Registers: The BIU has **four 16 bit segment registers**.

Note :Memory is logically divided into different segments. CODE ,DATA ,STACK EXTRA segments

- Each segment is **64K bytes** in size and is addressable by one of the segme nt registers .

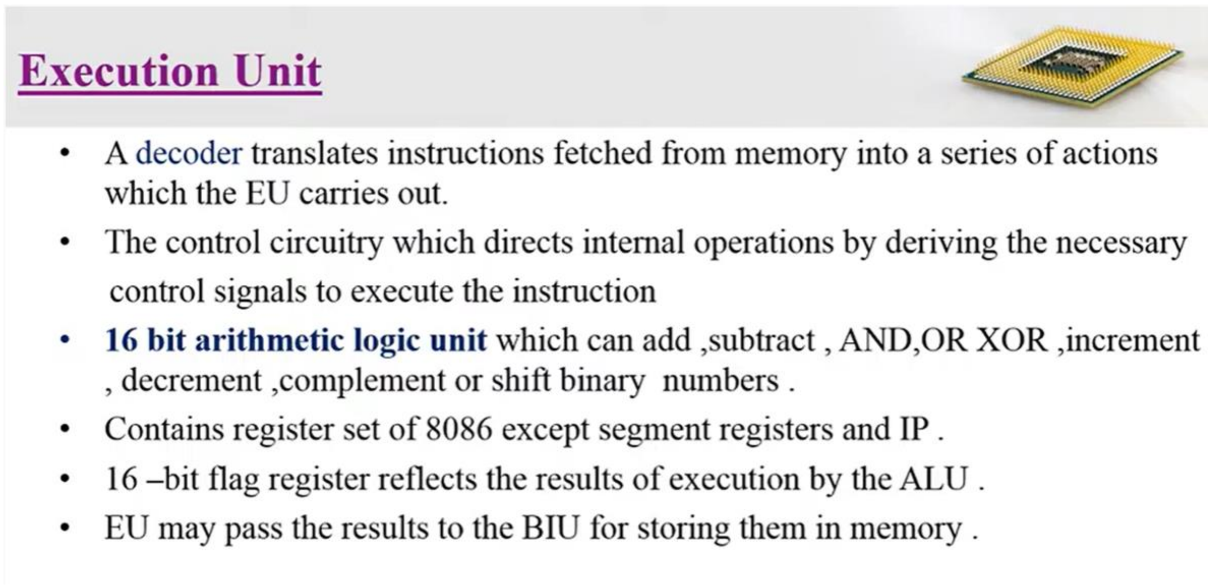
CODE SEGMENT –is a section of memory that holds the programs used by the processor .

DATA SEGMENT – is a section of memory that contains most data used by a progra

STACK SEGMENT – is a section of memory used as stack .

EXTRA SEGMENT –is an additional data segment that is used by some of the string instructions to hold destination data .

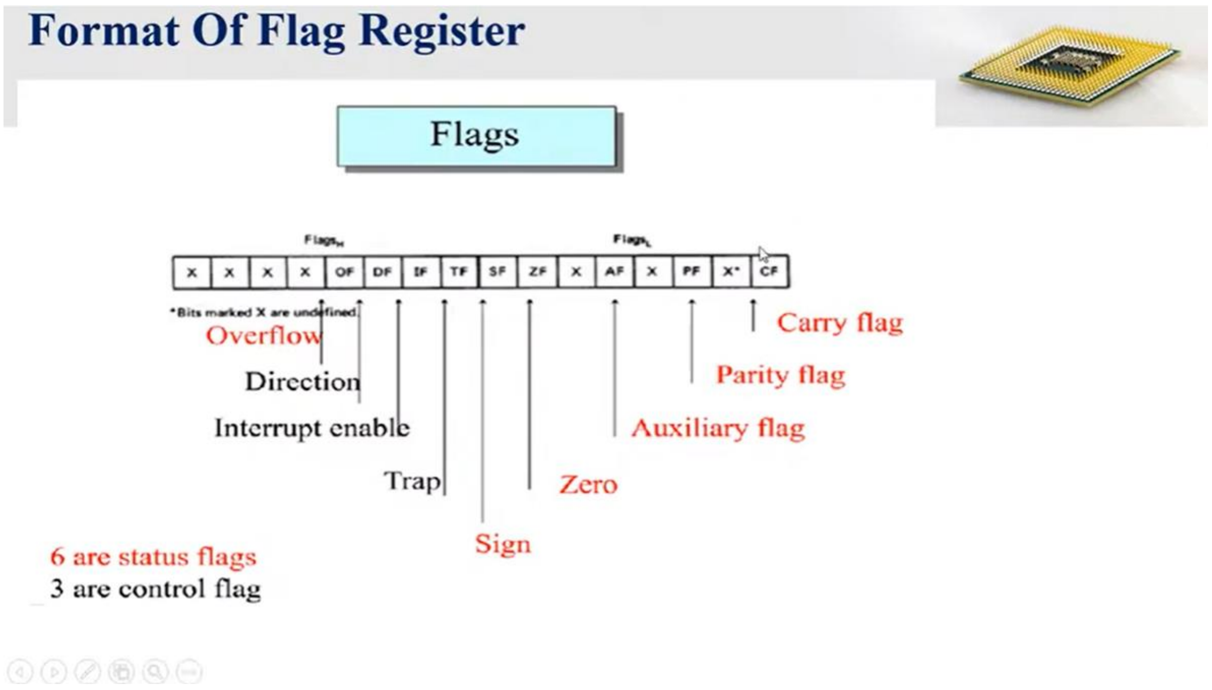
- Code segment register ,data segment register ,stack segment register and extra segment register defines the staring address of code segment ,data segment ,stack segment and extra segment respectively .



Execution Unit

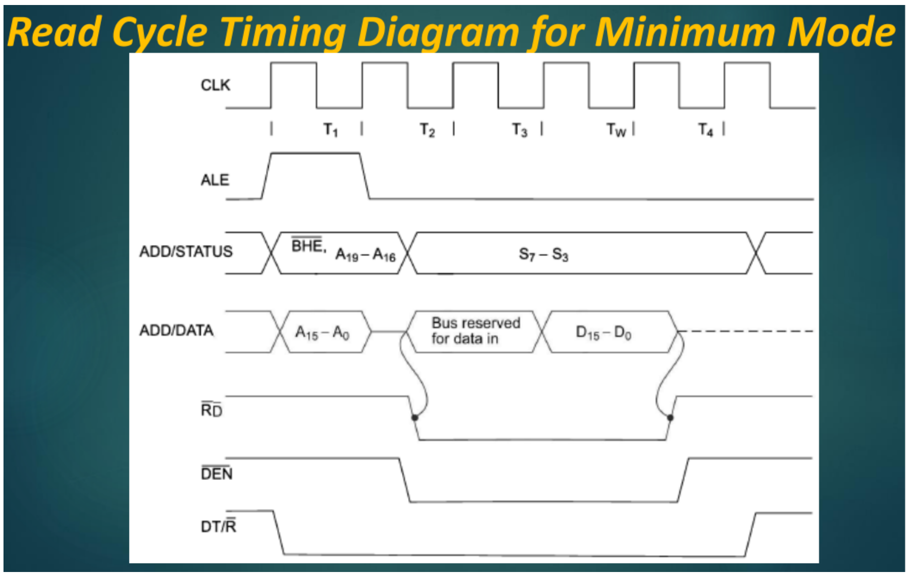
- A decoder translates instructions fetched from memory into a series of actions which the EU carries out.
- The control circuitry which directs internal operations by deriving the necessary control signals to execute the instruction
- **16 bit arithmetic logic unit** which can add ,subtract , AND,OR XOR ,increment , decrement ,complement or shift binary numbers .
- Contains register set of 8086 except segment registers and IP .
- 16 –bit flag register reflects the results of execution by the ALU .
- EU may pass the results to the BIU for storing them in memory .

Format Of Flag Register



12.

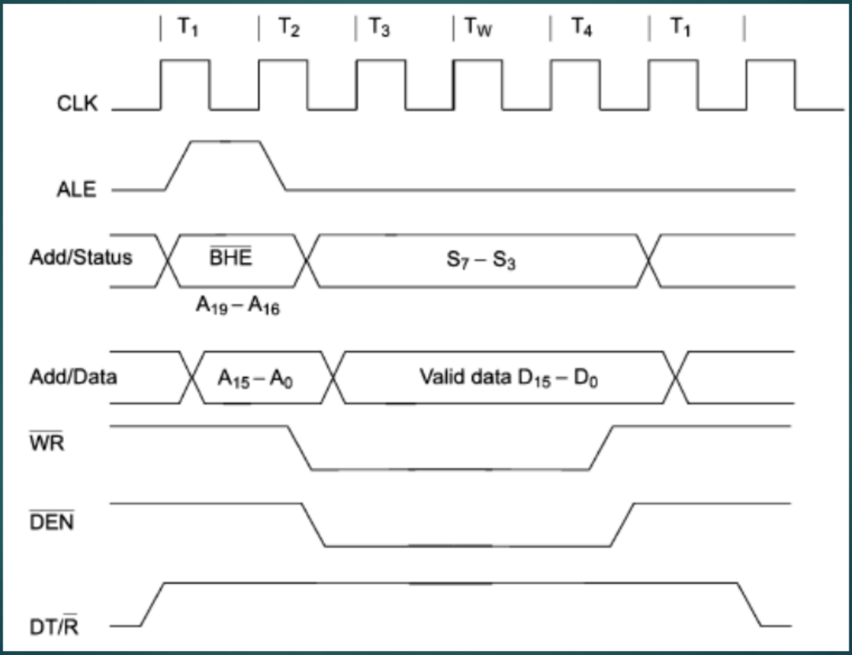
Read Cycle Timing Diagram for Minimum Mode

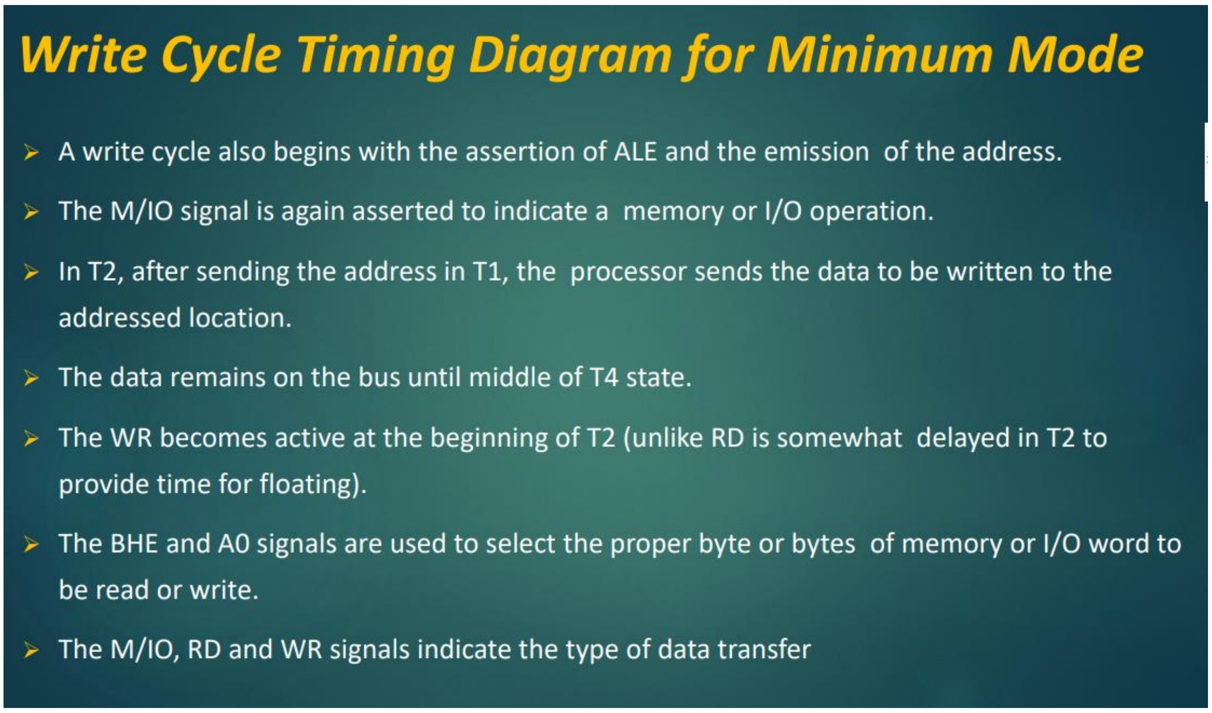


Read Cycle Timing Diagram for Minimum Mode

- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO# signal.
  - During the negative going edge of this signal, the valid address is latched on the local bus.
- From T1 to T4 , the M/IO signal indicates a memory or I/O operation.
- The BHE# and A0 signals address low, high or both bytes.
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tri-stated. The read (RD) control signal is also activated in T2.
- The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus.
- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tri-state its bus drivers.

Write Cycle Timing Diagram for Minimum Mode





Write Cycle Timing Diagram for Minimum Mode

- A write cycle also begins with the assertion of ALE and the emission of the address.
- The M/IO signal is again asserted to indicate a memory or I/O operation.
- In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.
- The data remains on the bus until middle of T4 state.
- The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).
- The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
- The M/IO, RD and WR signals indicate the type of data transfer

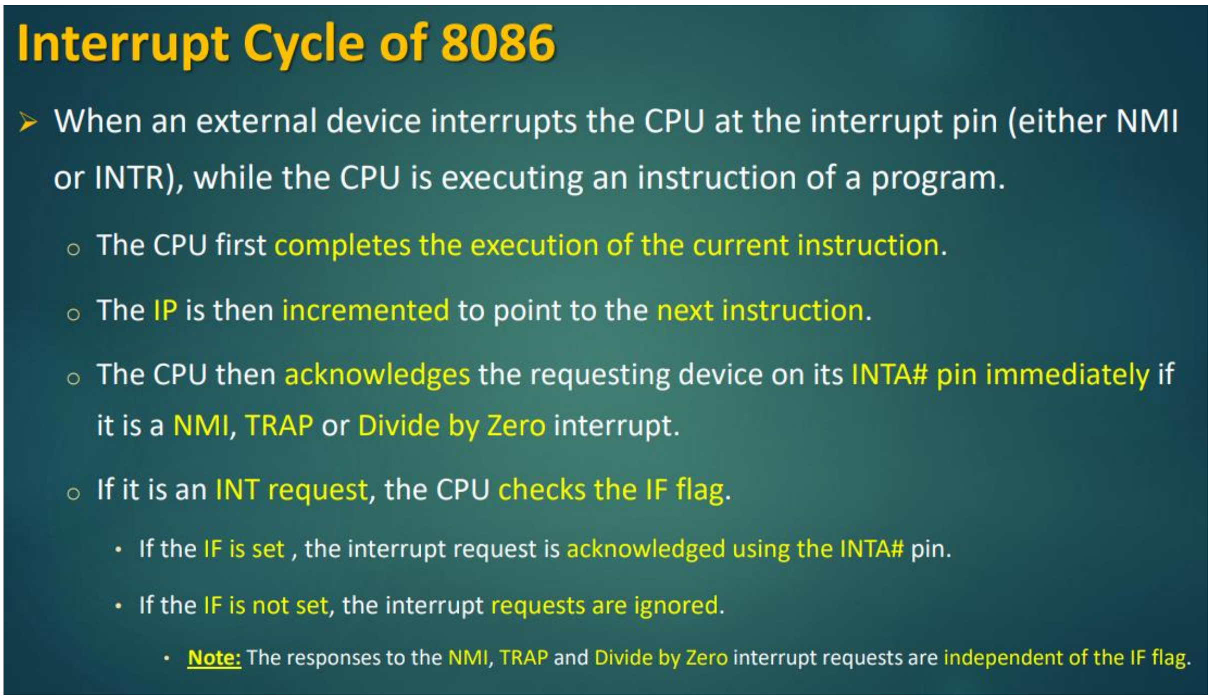
13.

```
MOV AX, 0 ; Initialize AX as the largest number
MOV BX, 65535 ; Initialize BX as the smallest number
MOV CX, 15 ; Set the counter to the number of elements
MOV SI, 2500H ; Set the starting location of the array

LOOP_START:
  MOV DX, [SI] ; Load the current element into DX
  CMP DX, AX ; Compare with the largest number
  JG UPDATE_MAX ; Jump if DX > AX
  CMP DX, BX ; Compare with the smallest number
  JL UPDATE_MIN ; Jump if DX < BX
  JMP NEXT_ELEMENT ; Jump to the next element

UPDATE_MAX:
  MOV AX, DX ; Update AX with the new largest number
  JMP NEXT_ELEMENT ; Jump to the next element

UPDATE_MIN:
  MOV BX, DX ; Update BX with the new smallest number

NEXT_ELEMENT:
  ADD SI, 2 ; Move to the next element (16-bit)
  LOOP LOOP_START ; Repeat until all elements are processed

; After the loop, the largest number will be in AX, and the
smallest number will be in BX
```

14.

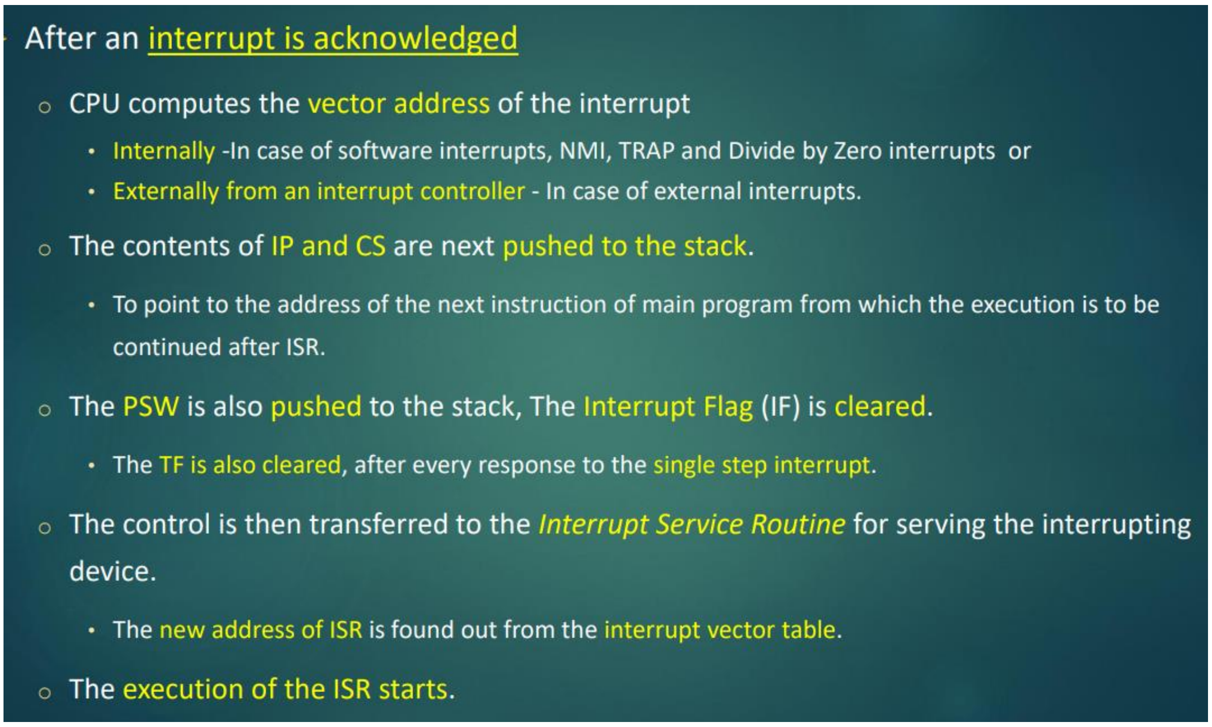
```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
LIST DW 2357H,0a579H,0c2322H,0c91eH,0c0000H,0957H
COUNT EQU 006h
DATA ENDS
CODE SEGMENT

START: XOR BX,BX
XOR DX,DX
MOV AX,DATA
MOV DS,AX
MOV CL,COUNT
MOV SI, OFFSET LIST
AGAIN: MOV AX,[SI]
RPR AX,01
JC ODD
INC BX
JMP NEXT
ODD: INC DX
NEXT: ADD SI,02
DEC CL
JNZ AGAIN
MOV AH,4CH
INC 21H
CODE ENDS
END START
```

15. a



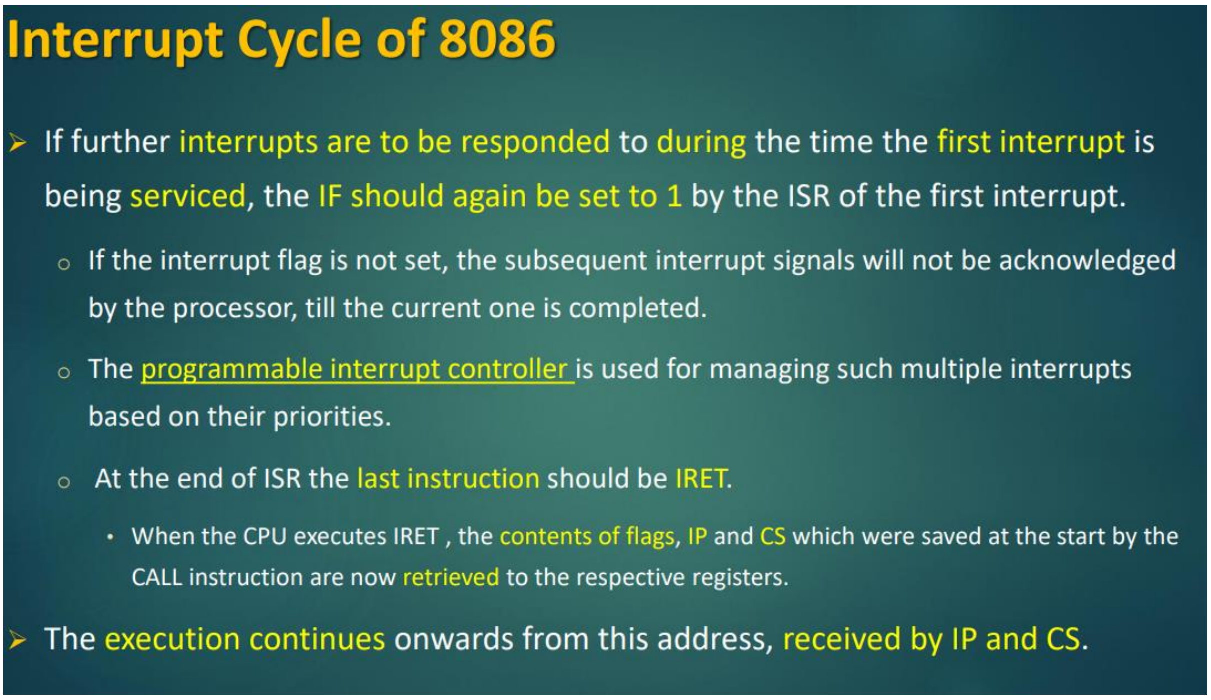
Interrupt Cycle of 8086

- When an external device interrupts the CPU at the interrupt pin (either NMI or INTR), while the CPU is executing an instruction of a program.
  - The CPU first completes the execution of the current instruction.
  - The IP is then incremented to point to the next instruction.
  - The CPU then acknowledges the requesting device on its INTA# pin immediately if it is a NMI, TRAP or Divide by Zero interrupt.
  - If it is an INT request, the CPU checks the IF flag.
    - If the IF is set , the interrupt request is acknowledged using the INTA# pin.
    - If the IF is not set, the interrupt requests are ignored.
      - **Note:** The responses to the NMI, TRAP and Divide by Zero interrupt requests are independent of the IF flag.



After an interrupt is acknowledged

- CPU computes the vector address of the interrupt
  - Internally -In case of software interrupts, NMI, TRAP and Divide by Zero interrupts or
  - Externally from an interrupt controller - In case of external interrupts.
- The contents of IP and CS are next pushed to the stack.
  - To point to the address of the next instruction of main program from which the execution is to be continued after ISR.
- The PSW is also pushed to the stack, The Interrupt Flag (IF) is cleared.
  - The TF is also cleared, after every response to the single step interrupt.
- The control is then transferred to the *Interrupt Service Routine* for serving the interrupting device.
  - The new address of ISR is found out from the interrupt vector table.
- The execution of the ISR starts.



Interrupt Cycle of 8086

- If further interrupts are to be responded to during the time the first interrupt is being serviced, the IF should again be set to 1 by the ISR of the first interrupt.
  - If the interrupt flag is not set, the subsequent interrupt signals will not be acknowledged by the processor, till the current one is completed.
  - The programmable interrupt controller is used for managing such multiple interrupts based on their priorities.
  - At the end of ISR the last instruction should be IRET.
    - When the CPU executes IRET , the contents of flags, IP and CS which were saved at the start by the CALL instruction are now retrieved to the respective registers.
- The execution continues onwards from this address, received by IP and CS.

15.b

Maskable Interrupts:

Definition: Maskable interrupts are interrupts that can be enabled or disabled by the programmer through the interrupt enable/disable flag (IF) in the FLAGS register.

Enable/Disable: The IF flag in the FLAGS register controls the maskable interrupts. If IF is set (1), the maskable interrupts are enabled, and the processor responds to them. If IF is cleared (0), the maskable interrupts are disabled, and the processor ignores them.

Priority: Maskable interrupts are typically assigned different priority levels, and the processor can respond to the interrupt with the highest priority first

Non-Maskable Interrupts (NMI):

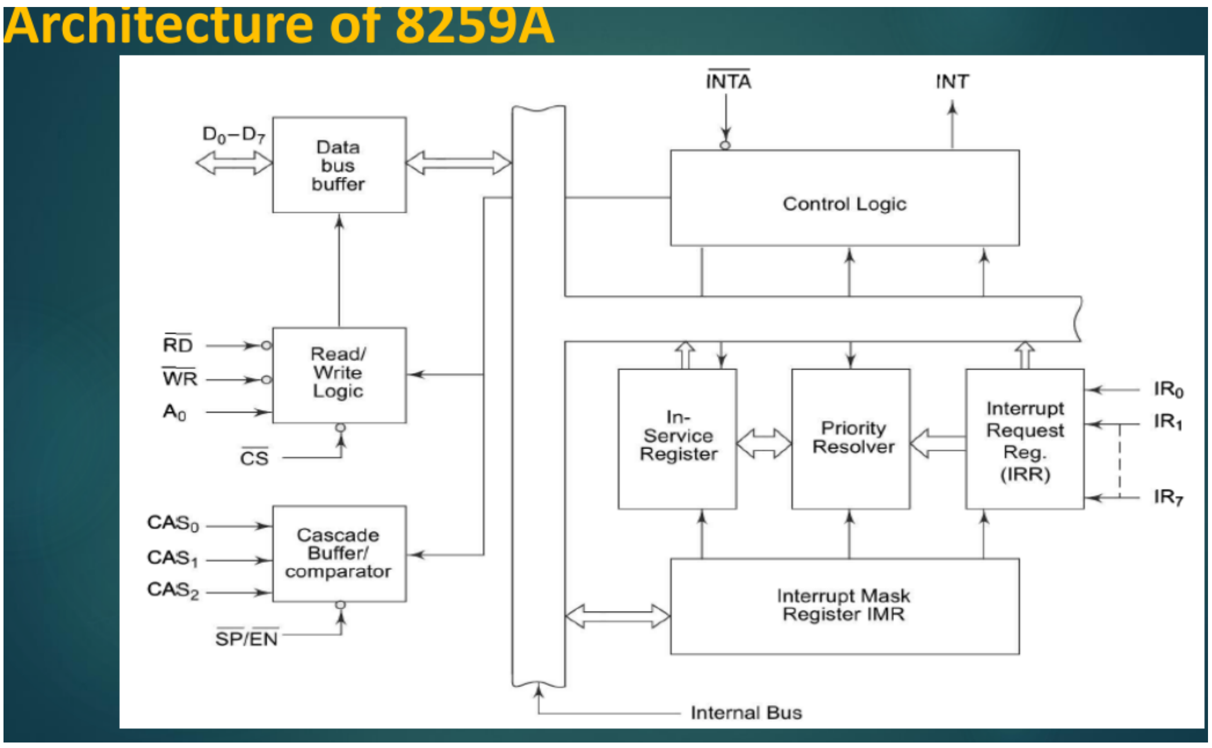
Definition: Non-maskable interrupts are interrupts that cannot be disabled by the programmer. They have a higher priority than maskable interrupts and are intended for critical events that require immediate attention.

Enable/Disable: NMIs are not affected by the IF flag; they can occur even when maskable interrupts are disabled. This makes NMIs suitable for handling emergency situations or critical errors.

Purpose: NMIs are often used for events that demand immediate attention, such as hardware faults, power failures, or other severe system issues.

Priority: NMIs generally take precedence over maskable interrupts. When an NMI occurs, the processor stops its current execution and jumps to the NMI handler

16



Architecture of 8259A

Interrupt Request Register (IRR)

o The interrupts at IRQ input lines are handled by IRR internally.

o IRR stores all the interrupt requests in its order to serve them one by one on the priority basis.

In-Service Register (ISR)

o Stores all the interrupt requests those are being served,

o ISR keeps a track of the requests being served.

Priority Resolver

o Determines the priorities of the interrupt requests appearing simultaneously.

o The highest priority is selected and stored into the corresponding bit of ISR

during INTA# pulse.

o The IR0 has the highest priority while the IR7 has the lowest one, normally in fixed priority mode.

o The priorities however may be altered by programming the 8259A in rotating priority mode

Interrupt Mask Register (IMR)

o Stores the bits required to mask the interrupt inputs.

o IMR operates on IRR at the direction of the Priority Resolves.

Interrupt Control Logic

o Manages the interrupt and the interrupt acknowledge signals to be sent to

the CPU for serving one of the eight interrupt requests.

o Also accepts the interrupt acknowledge (INTA) signal from CPU that causes

the 8259A to release vector address on to the data bus

17.

Mode 0 – Basic I/O Mode

Mode 1 – Strobed I/O Mode

Mode 2 – Strobed Bidirectional I/O **(refer module 4 ppt page number 50)**

18.

(Refer module 4 ppt page number 100)

19.a

Addressing Modes

An addressing mode is a method of specifying the data source or destination in an instruction.

The modes in the 8051 family instructions are as follows:

o Immediate

o Register

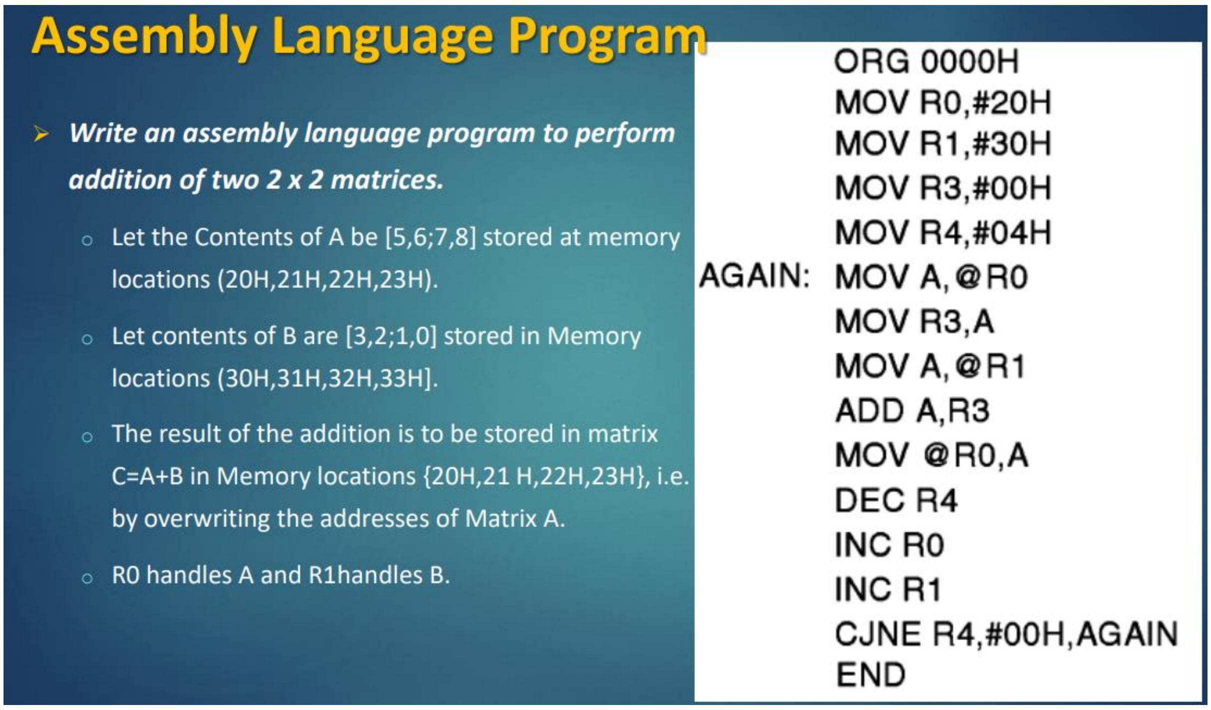
o Direct

o Indirect Register

o Indexed (Type of Indirect)

(Refer module 5.2.pdf page number 5)

19.b



Assembly Language Program

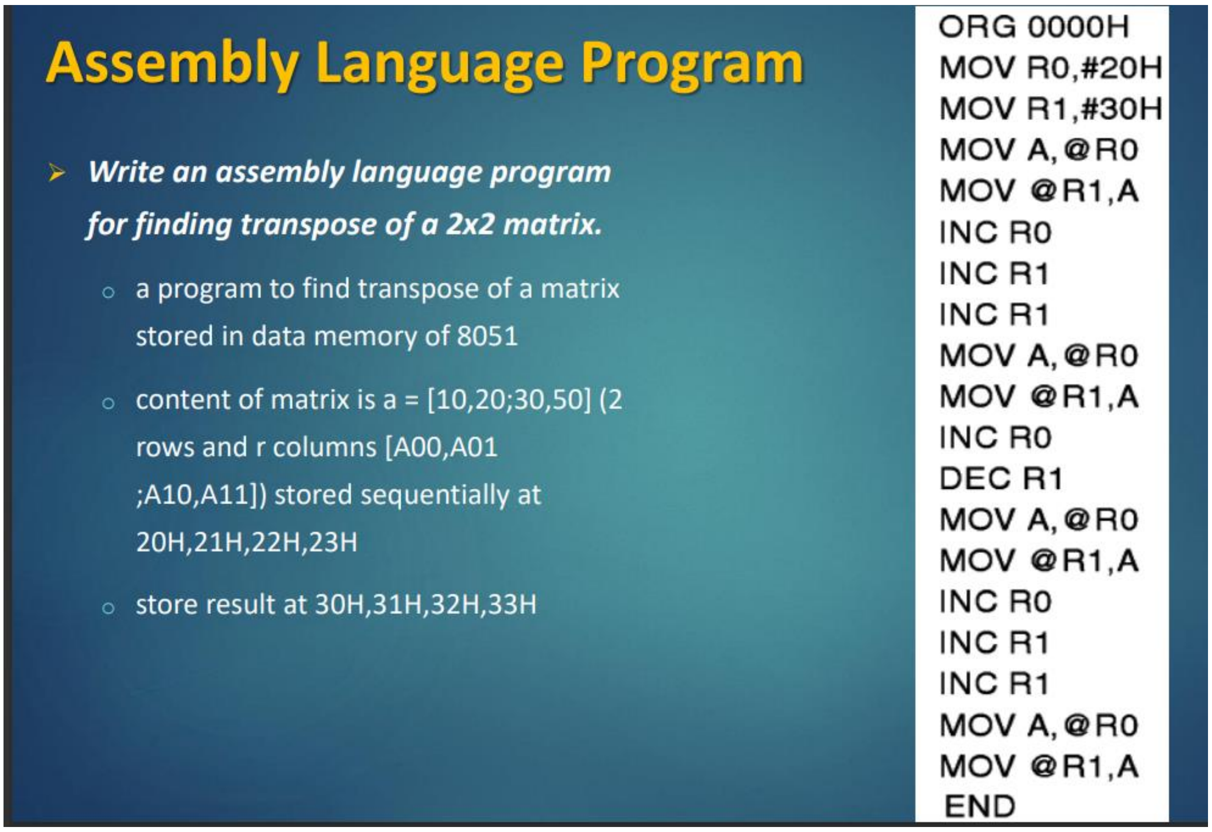
- ***Write an assembly language program to perform addition of two 2 x 2 matrices.***
  - Let the Contents of A be [5,6;7,8] stored at memory locations (20H,21H,22H,23H).
  - Let contents of B are [3,2;1,0] stored in Memory locations (30H,31H,32H,33H].
  - The result of the addition is to be stored in matrix C=A+B in Memory locations {20H,21 H,22H,23H}, i.e. by overwriting the addresses of Matrix A.
  - R0 handles A and R1handles B.

```
        ORG 0000H
        MOV R0,#20H
        MOV R1,#30H
        MOV R3,#00H
        MOV R4,#04H
AGAIN:  MOV A,@R0
        MOV R3,A
        MOV A,@R1
        ADD A,R3
        MOV @R0,A
        DEC R4
        INC R0
        INC R1
        CJNE R4,#00H,AGAIN
        END
```

20.a

Refer module 5.1.pdf page number 143

20.b



Assembly Language Program

- ***Write an assembly language program for finding transpose of a 2x2 matrix.***
  - a program to find transpose of a matrix stored in data memory of 8051
  - content of matrix is a = [10,20;30,50] (2 rows and r columns [A00,A01 ;A10,A11]) stored sequentially at 20H,21H,22H,23H
  - store result at 30H,31H,32H,33H

```
ORG 0000H
MOV R0,#20H
MOV R1,#30H
MOV A,@R0
MOV @R1,A
INC R0
INC R1
INC R1
MOV A,@R0
MOV @R1,A
INC R0
DEC R1
MOV A,@R0
MOV @R1,A
INC R0
INC R1
INC R1
MOV A,@R0
MOV @R1,A
 END
```