**Question 1**

; You may customize this and other start-up templates;

; The location of this template is c:\emu8086\inc\0\_com\_template.txt

org 100h

code segment

start:

MOV SI ,0A000H ; Address to source index register

MOV DI ,0C001H ; Destination address to destination index register

MOV CX, 8; Count register so as to iterate move operation 8 times

Loop1:

MOV AX , [SI]; Move contents of address pointed to by SI to accumulator

MOV [DI],AX ; Move accumulator contents to destination address

inc SI ; Increment SI contents to point to next memory location of source data

inc DI ; Increment DI contents to point to next memory location of destination data

loop Loop1

ends

end start; Set entry point and stop the assembler.

Ret

**Question 2**

**1. Data Transfer Instructions**

These instructions are used to transfer the data from the source operand to the destination operand. Instruction to transfer a word

MOV − Used to copy the byte or word from the provided source to the provided destination.

PPUSH − Used to put a word at the top of the stack.

POP − Used to get a word from the top of the stack to the provided location.

PUSHA − Used to put all the registers into the stack.

POPA − Used to get words from the stack to all registers.

XCHG − Used to exchange the data from two locations.

XLAT − Used to translate a byte in AL using a table in the memory. Instructions for input and output port transfer

IN − Used to read a byte or word from the provided port to the accumulator.

OUT − Used to send out a byte or word from the accumulator to the provided port. Instructions to transfer the address

LEA − Used to load the address of operand into the provided register.

LDS − Used to load DS register and other provided register from the memory

LES − Used to load ES register and other provided register from the memory. Instructions to transfer flag registers

LAHF − Used to load AH with the low byte of the flag register.

SAHF − Used to store AH register to low byte of the flag register.

PUSHF − Used to copy the flag register at the top of the stack.

POPF − Used to copy a word at the top of the stack to the flag register.

**2.Arithmetic Instructions**

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc. Instructions to perform addition.

ADD − Used to add the provided byte to byte/word to word.

ADC − Used to add with carry.

INC − Used to increment the provided byte/word by 1.

AAA − Used to adjust ASCII after addition.

DAA − Used to adjust the decimal after the addition/subtraction operation. Instructions to perform subtraction

SUB − Used to subtract the byte from byte/word from word.

SBB − Used to perform subtraction with borrow.

DEC − Used to decrement the provided byte/word by 1.

CMP − Used to compare 2 provided byte/word.

AAS − Used to adjust ASCII codes after subtraction.

DAS − Used to adjust decimal after subtraction. Instruction to perform multiplication

MUL − Used to multiply unsigned byte by byte/word by word.

IMUL − Used to multiply signed byte by byte/word by word.

AAM − Used to adjust ASCII codes after multiplication. Instructions to perform division

DIV − Used to divide the unsigned word by byte or unsigned double word by word.

IDIV − Used to divide the signed word by byte or signed double word by word.

AAD − Used to adjust ASCII codes after division.

CBW − Used to fill the upper byte of the word with the copies of sign bit of the lower byte.

CWD − Used to fill the upper word of the double word with the sign bit of the lower word.

**3.Bit Manipulation Instructions**

These instructions are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc. Instructions to perform logical operation

NOT − Used to invert each bit of a byte or word.

AND − Used for adding each bit in a byte/word with the corresponding bit in another byte/word.

OR − Used to multiply each bit in a byte/word with the corresponding bit in another byte/word.

XOR − Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word.

TEST − Used to add operands to update flags, without affecting operands. Instructions to perform shift operations

SHL/SAL − Used to shift bits of a byte/word towards left and put zero(S) in LSBs.

SHR − Used to shift bits of a byte/word towards the right and put zero(S) in MSBs.

SAR − Used to shift bits of a byte/word towards the right and copy the old MSB into the new MSB. Instructions to perform rotate operations

ROL − Used to rotate bits of byte/word towards the left, i.e. MSB to LSB and to Carry Flag [CF].

ROR − Used to rotate bits of byte/word towards the right, i.e. LSB to MSB and to Carry Flag [CF].

RCR − Used to rotate bits of byte/word towards the right, i.e. LSB to CF and CF to MSB.

RCL − Used to rotate bits of byte/word towards the left, i.e. MSB to CF and CF to LSB.

**4.String Instructions**

String is a group of bytes/words and their memory is always allocated in a sequential order.

REP − Used to repeat the given instruction till CX ≠ 0.

REPE/REPZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.

REPNE/REPNZ − Used to repeat the given instruction until CX = 0 or zero flag ZF = 1.

MOVS/MOVSB/MOVSW − Used to move the byte/word from one string to another.

COMS/COMPSB/COMPSW − Used to compare two string bytes/words.

INS/INSB/INSW − Used as an input string/byte/word from the I/O port to the provided memory location.

OUTS/OUTSB/OUTSW − Used as an output string/byte/word from the provided memory location to the I/O port.

SCAS/SCASB/SCASW − Used to scan a string and compare its byte with a byte in AL or string word with a word in AX.

LODS/LODSB/LODSW − Used to store the string byte into AL or string word into AX.

**5. Program Execution Transfer Instructions (Branch and Loop Instructions)**

These instructions are used to transfer/branch the instructions during an execution. Instructions to transfer the instruction during an execution without any condition:

CALL − Used to call a procedure and save their return address to the stack.

RET − Used to return from the procedure to the main program.

JMP − Used to jump to the provided address to proceed to the next instruction. Instructions to transfer the instruction during an execution with some conditions

JA/JNBE − Used to jump if above/not below/equal instruction satisfies.

JAE/JNB − Used to jump if above/not below instruction satisfies.

JBE/JNA − Used to jump if below/equal/ not above instruction satisfies.

JC − Used to jump if carry flag CF = 1

JE/JZ − Used to jump if equal/zero flag ZF = 1

JG/JNLE − Used to jump if greater/not less than/equal instruction satisfies.

JGE/JNL − Used to jump if greater than/equal/not less than instruction

JL/JNGE − Used to jump if less than/not greater than/equal instruction

JLE/JNG − Used to jump if less than/equal/if not greater than instruction

JNC − Used to jump if no carry flag (CF = 0)

JNE/JNZ − Used to jump if not equal/zero flag ZF = 0

JNO − Used to jump if no overflow flag OF = 0

JNP/JPO − Used to jump if not parity/parity odd PF = 0

JNS − Used to jump if not sign SF = 0

JO − Used to jump if overflow flag OF = 1

JP/JPE − Used to jump if parity/parity even PF = 1

JS − Used to jump if sign flag SF = 1

**5.Processor Control Instructions**

These instructions are used to control the processor action by setting/resetting the flag values.

STC − Used to set carry flag CF to 1

CLC − Used to clear/reset carry flag CF to 0

CMC − Used to put complement at the state of carry flag CF.

STD − Used to set the direction flag DF to 1

CLD − Used to clear/reset the direction flag DF to 0

STI − Used to set the interrupt enable flag to 1, i.e., enable INTR input.

CLI − Used to clear the interrupt enable flag to 0, i.e., disable INTR input.

**6.Iteration Control Instructions**

These instructions are used to execute the given instructions for number of times.

* LOOP − Used to loop a group of instructions until the condition satisfies, i.e., CX = 0
* LOOPE/LOOPZ − Used to loop a group of instructions till it satisfies ZF = 1 & CX = 0
* LOOPNE/LOOPNZ − Used to loop a group of instructions till it satisfies ZF = 0 & CX = 0
* JCXZ − Used to jump to the provided address if CX = 0

1. Interrupt Instructions

These instructions are used to call the interrupt during program execution.

* INT − Used to interrupt the program during execution and calling service specified.
* INTO − Used to interrupt the program during execution if OF = 1
* IRET − Used to return from interrupt service to the main program

**Question 3**

1. Binary Integers

The binary system is a kind of number system that uses a simplified notation to reduce modern counting to only the integers one and zero. A binary integer is one of these ones and zeros that make up a binary string, and is a whole number.

1. Packed decimal numbers

A packed decimal representation stores two decimal digits in one byte. Each byte has two nibbles, and each nibble is indicated by a hexadecimal digit. For example, the value 23 would be stored in two nibbles, using the hexadecimal digits 2 and 3 (the bit representation would be 0010 0011).

1. Real numbers

These are numbers that include both rational and irrational numbers. They can be both positive and negative.

4.Temporary real format

It was designed not to store data at higher precision as such, but rather primarily to allow for the computation of double results more reliably and accurately by minimising overflow and round off-errors in intermediate calculations.

**Question 5**

; You may customize this and other start-up templates;

; The location of this template is c:\emu8086\inc\0\_com\_template.txt

org 100h

A DW 01H

B DW 05H

MOV AX, A ;WE ARE INITIALISING THE AX REGISTER WITH THE VALUE OF A, BECOZ WE CANNOT ADD TWO VARIABLES AT THE SAME TIME BUT WE CAN ADD B WITH AX

LOOP1: ADD AX, B;LABEL LOOP1 IS ADDING CONTENTS OF AX AND B

INC A ;INCREMENTING A SO THAT A AT SOME POINT IS EAQUAL OR GREATER THAN B

MOV CX, A;YOU CANT COMPARE TWO VARIABLES THEREFORE WE ARE PUTTING VALUE OF A INTO REGISTER CX

CMP CX, B;COMPARE REGISTER CX AND B

JLE LOOP1;IF CX IS LESS THAN B IT WILL JUMP TO LABEL LOOP1

ret