
 Marwadi University Marwadi Chandarana Group 	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

Aim: To perform Data Transfer Operations.

Theory:

Data transfer instructions in the 8086 microprocessor are used to move data between memory locations, registers, and input/output (I/O) devices. These instructions are essential for manipulating data within a program, as well as for communicating with external devices.

Data transfer instructions are a fundamental part of programming in the 8086 microprocessor, and are used extensively in applications ranging from simple data manipulation to complex I/O device communication and string processing.

Data transfer instructions are the instructions which transfer data in the microprocessor. They are also called copy instructions.

Types of Data transfer instructions :

1. Move instructions:



These instructions are used to move data from one memory location to another or between a memory location and a register. They include the following instructions:

- **MOV:** Moves data from a source operand to a destination operand.
- **XCHG:** Swaps the contents of two operands.
- **XLAT:** Translates a byte in memory using a lookup table pointed to by the contents of the AL register.
- **LEA:** Loads a 16-bit offset address into a register.

2. Load instructions:

These instructions are used to load data from a memory location or I/O device into a register. They include the following instructions:

- **LDS:** Loads a 16-bit pointer value from a memory location into a register pair and loads the 8-bit value from the next memory location into another register.
- **LSS:** Loads a 16-bit pointer value from a memory location into a register pair and loads the 16-bit value from the next memory location into another register.
- **LXI:** Loads a 16-bit value into a register pair.
- **MOV with memory operand:** Loads data from a memory location into a register.

 Marwadi University Marwadi Chandarana Group 	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

3. Store instructions:

These instructions are used to store data from a register into a memory location or I/O device. They include the following instructions:

- MOV with memory operand: Stores data from a register into a memory location.
- STA: Stores the contents of the accumulator register (AL or AX) in memory.
- STAX: Stores the contents of a register pair (BC, DE, or HL) in memory using either the indirect addressing mode or the direct addressing mode.
- SHLD: Stores a 16-bit data word from registers H and L in memory using the direct addressing mode.
- PUSH: Stores the contents of a register onto the stack.

4. Input/Output instructions:



These instructions are used to communicate with external input/output (I/O) devices. They include the following instructions:

- IN: Reads a byte or word of data from an I/O port into a register.
- OUT: Writes a byte or word of data from a register to an I/O port.
- INS: Reads a block of data from an I/O port into a memory location.
- OUTS: Writes a block of data from a memory location to an I/O port.

5. String instructions:

These instructions are used for manipulating strings of data, such as moving, copying, or comparing strings. They operate on consecutive bytes or words in memory, and can be used for fast and efficient string processing. Some examples of string instructions include:

- MOVS: Moves a byte or word from a source location to a destination location, and updates the index registers to point to the next byte or word.
- CMPS: Compares a byte or word in memory to a byte or word in a register, and updates the index registers accordingly.
- LODS: Loads a byte or word from a memory location into a register, and updates the index registers to point to the next byte or word.
- STOS: Stores a byte or word from a register into a memory location, and updates the index registers to point to the next byte or word.



 Marwadi University Marwadi Chandarana Group 	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

Following is the table showing the list of data transfer instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
MOV	D, S	D = S	MOV AX, [SI]
PUSH	D	pushes D to the stack	PUSH DX
POP	D	pops the stack to D	POP AS
PUSHA	none	put all the registers into the stack	PUSHA
POPA	none	gets words from the stack to all registers	POPA
XCHG	D, S	exchanges contents of D and S	XCHG [2050], AX
IN	D, S	copies a byte or word from S to D	IN AX, DX
OUT	D, S	copies a byte or word from D to S	OUT 05, AL
XLAT	none	translates a byte in AL using a table in the memory	XLAT
LAHF	none	loads AH with the lower byte of the flag register	LAHF
SAHF	none	stores AH register to lower byte of the flag register	SAHF
PUSHF	none	copies the flag register at the top of the stack	PUSHF
POPF	none	copies a word at the top of the stack to the flag register	POPF

Here D stands for destination and S stands for source.
D and S can either be register, data or memory address.




 Marwadi University Marwadi Chandarana Group 	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

Code of program:

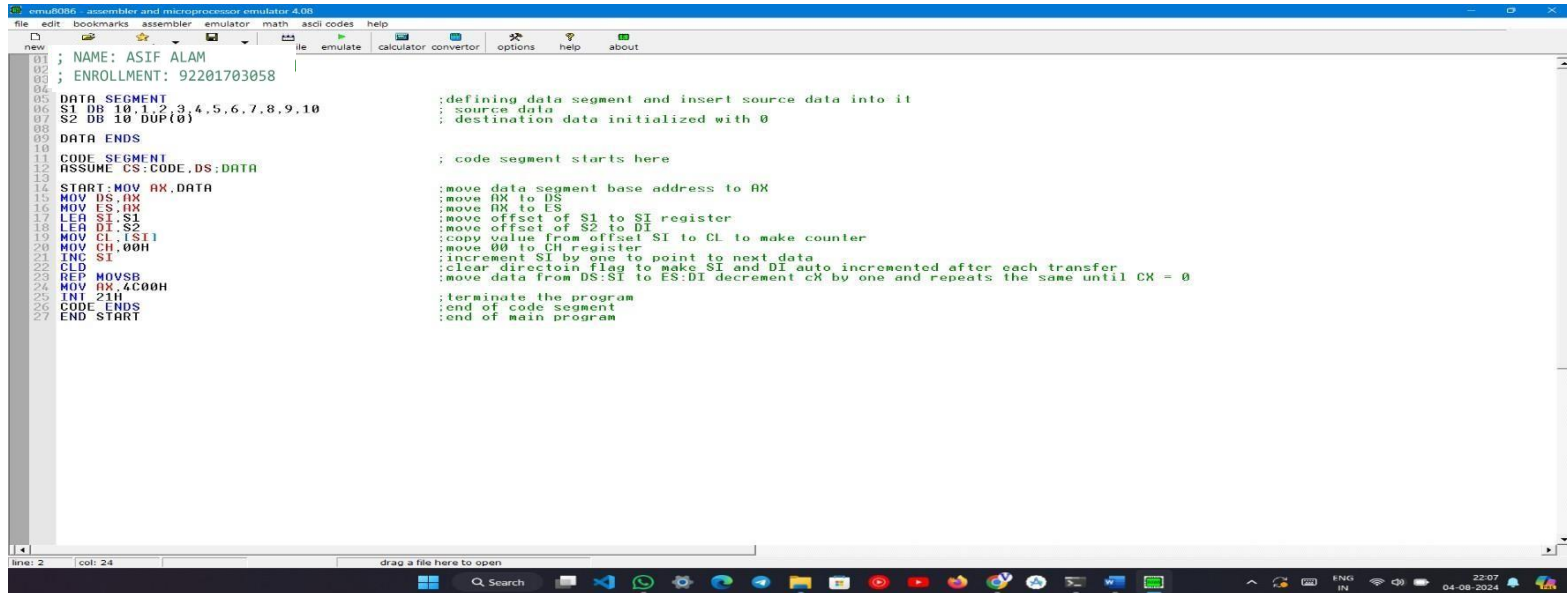
Steps to execute the program.

Step:1 – Start the emulator and write code in new file.



 Marwadi University Marwadi Chandarana Group	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

Step:2 –Write down the code and emulate the code.



```

; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

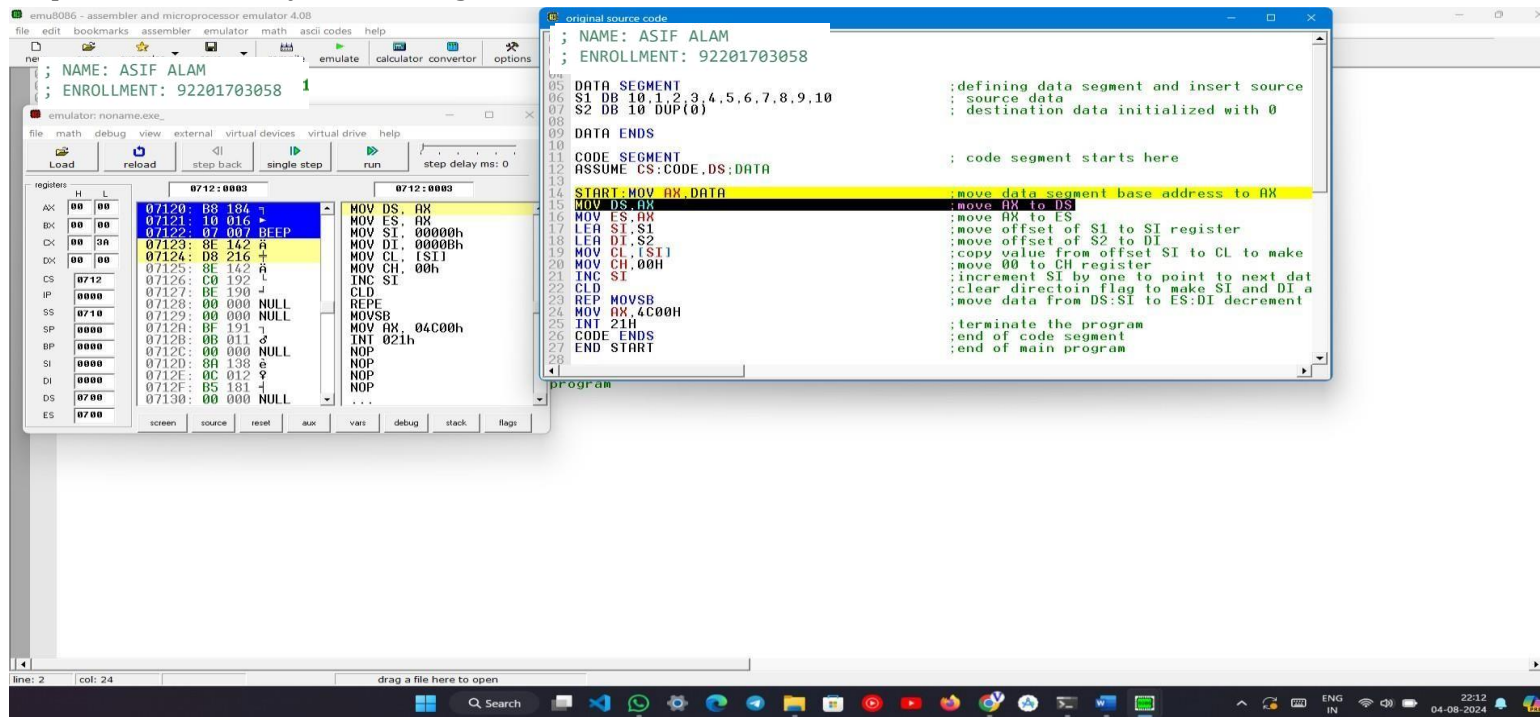
DATA SEGMENT
S1 DB 10,1,2,3,4,5,6,7,8,9,10
S2 DB 10 DUP(0)
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA

START: MOV AX, DATA
MOV DS, AX
MOV ES, AX
LEA SI, S1
LEA DI, S2
MOV CL, [SI]
MOV CH, 00H
INC SI
CLD
REP MOVSB
MOV AX, 4C00H
INT 21H
CODE ENDS
END START

```

Step:3 –Check the Physical and logical address.



The screenshot shows the emu8086 emulator interface with the 'original source code' window open. The 'original source code' window displays the assembly code, and the 'emulator: noname.exe' window shows the registers and memory addresses. The 'original source code' window highlights the following lines:

```

; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

DATA SEGMENT
S1 DB 10,1,2,3,4,5,6,7,8,9,10
S2 DB 10 DUP(0)
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA

START: MOV AX, DATA
MOV DS, AX
MOV ES, AX
LEA SI, S1
LEA DI, S2
MOV CL, [SI]
MOV CH, 00H
INC SI
CLD
REP MOVSB
MOV AX, 4C00H
INT 21H
CODE ENDS
END START

```

The 'emulator: noname.exe' window shows the registers and memory addresses. The 'original source code' window highlights the following lines:

```

; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

DATA SEGMENT
S1 DB 10,1,2,3,4,5,6,7,8,9,10
S2 DB 10 DUP(0)
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE,DS:DATA

START: MOV AX, DATA
MOV DS, AX
MOV ES, AX
LEA SI, S1
LEA DI, S2
MOV CL, [SI]
MOV CH, 00H
INC SI
CLD
REP MOVSB
MOV AX, 4C00H
INT 21H
CODE ENDS
END START

```

Subject: Fundamental of Processors (01CE0509)

Aim: To perform Data Transfer Operations

Experiment No: 03

Date:

Enrolment No: 92201703171

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options

```
01 ; NAME: ASIF ALAM
02 ; ENROLLMENT: 92201703058
03
```

emulator: noname.exe

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers		0712:0005		0712:0005	
	H	L			
AX	00	00	07120: B8 184	MOV ES, AX	
BX	00	00	07121: 10 016	MOV SI, 00000h	
CX	00	3A	07122: 07 007 BEEP	MOV DI, 0000Bh	
DX	00	00	07123: 8E 142	MOV CL, [SI]	
CS	0712		07124: D8 216	MOV CH, 00h	
IP	0000		07125: 8E 142	INC SI	
SS	0710		07126: C0 192	CLD	
SP	0000		07127: BE 190	REPE	
BP	0000		07128: 00 000 NULL	MOVSB	
SI	0000		07129: 00 000 NULL	MOV AX, 04C00h	
DI	0000		0712A: BF 191	INT 021h	
DS	0700		0712B: 0B 011	NOP	
ES	0700		0712C: 00 000 NULL	NOP	
			0712D: 8A 138	NOP	
			0712E: 0C 012	NOP	
			0712F: B5 181	NOP	
			07130: 00 000 NULL	...	

original source code

```
01 ; NAME: ASIF ALAM
02 ; ENROLLMENT: 92201703058
03
04
05 DATA SEGMENT                                ;defining data segment and insert source
06 S1 DB 10,1,2,3,4,5,6,7,8,9,10                ; source data
07 S2 DB 10 DUP(0)                                ; destination data initialized with 0
08
09 DATA ENDS
10
11 CODE SEGMENT                                ; code segment starts here
12 ASSUME CS:CODE,DS:DATA
13
14 START:MOV AX,DATA                            ;move data segment base address to AX
15 MOV DS,AX                                    ;move AX to DS
16 MOV ES,AX                                    ;move AX to ES
17 LEA SI,S1                                    ;move offset of S1 to SI register
18 LEA DI,S2                                    ;move offset of S2 to DI
19 MOV CL,[SI]                                  ;copy value from offset SI to CL to make
20 MOV CH,00h                                  ;move 00 to CH register
21 INC SI                                       ;increment SI by one to point to next dat
22 CLD                                       ;clear directoin flag to make SI and DI a
23 REP MOVSB                                  ;move data from DS:SI to ES:DI decrement
24 MOV AX,4C00h
25 INT 21h                                    ;terminate the program
26 CODE ENDS                                  ;end of code segment
27 END START                                  ;end of main program
28
```

program

line: 2 col: 24 drag a file here to open





Search



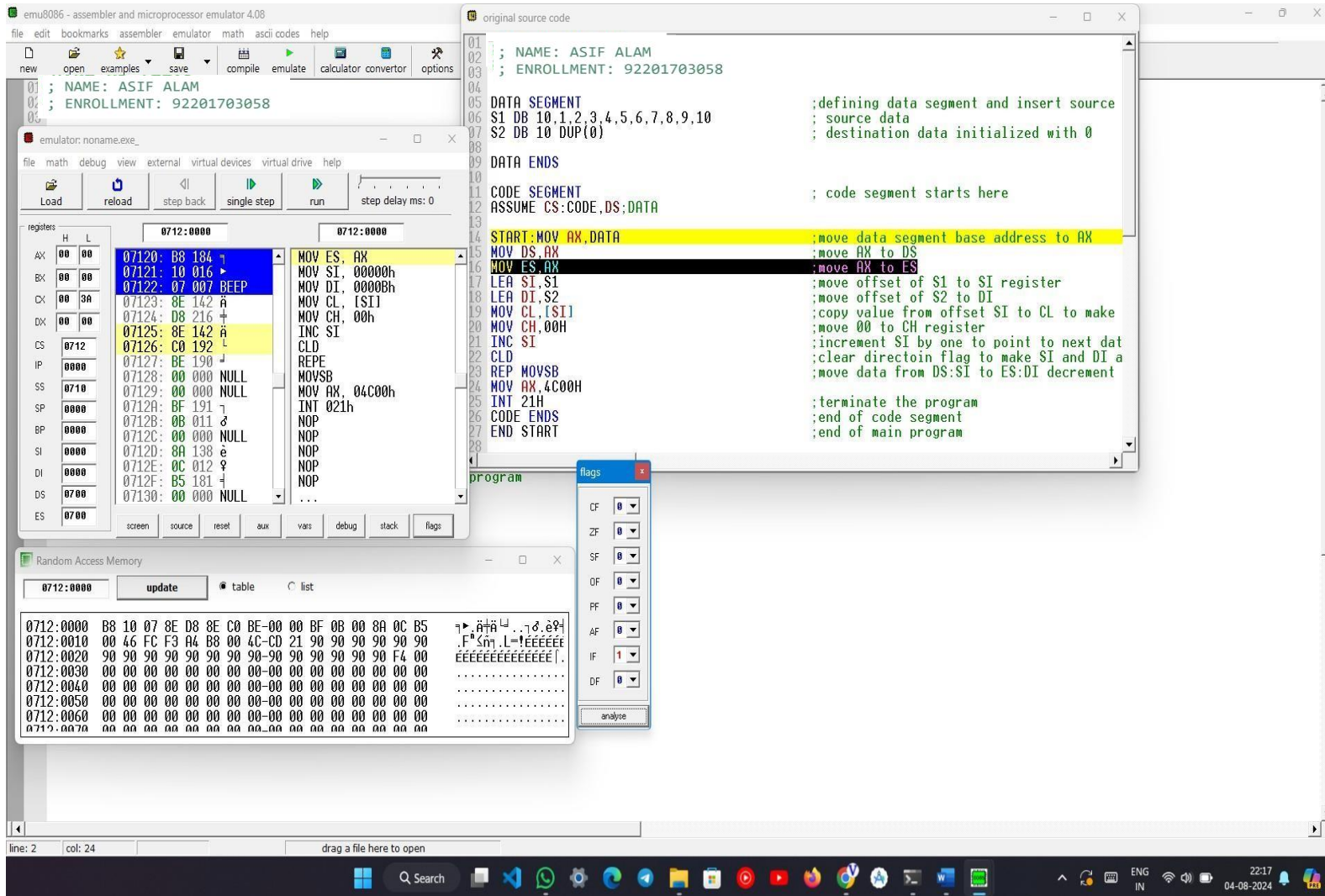
ENG
IN

22:14

04-08-2024

 Marwadi University Marwadi Chandarana Group 	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703058

Step:4 –Check the memory location and flag register.



The screenshot displays the emu8086 emulator interface with the following components:

- Assembly Code Window:** Shows the source code for a program named 'ASIF ALAM' with enrollment number '92201703058'. The code includes a data segment with a table of 10 bytes (10, 1, 2, 3, 4, 5, 6, 7, 8, 9) and a code segment that moves the data segment base to AX, then iterates through the data, moving each byte to the SI register and incrementing SI. The program ends with a 'Halt' instruction.
- Registers Window:** Shows the state of the 8086 registers. The AX register contains 0000h, SI contains 0000h, and DI contains 0000h. The CS register points to segment 0712.
- Random Access Memory Window:** Displays the memory contents starting at address 0712:0000. The first 10 bytes contain the values 10, 01, 02, 03, 04, 05, 06, 07, 08, 09, followed by 00.
- Flags Window:** Shows the status of the 8086 flags. The Zero Flag (ZF) is set to 1, indicating that the last operation resulted in a zero value.

Subject: Fundamental of Processors (01CE0509)

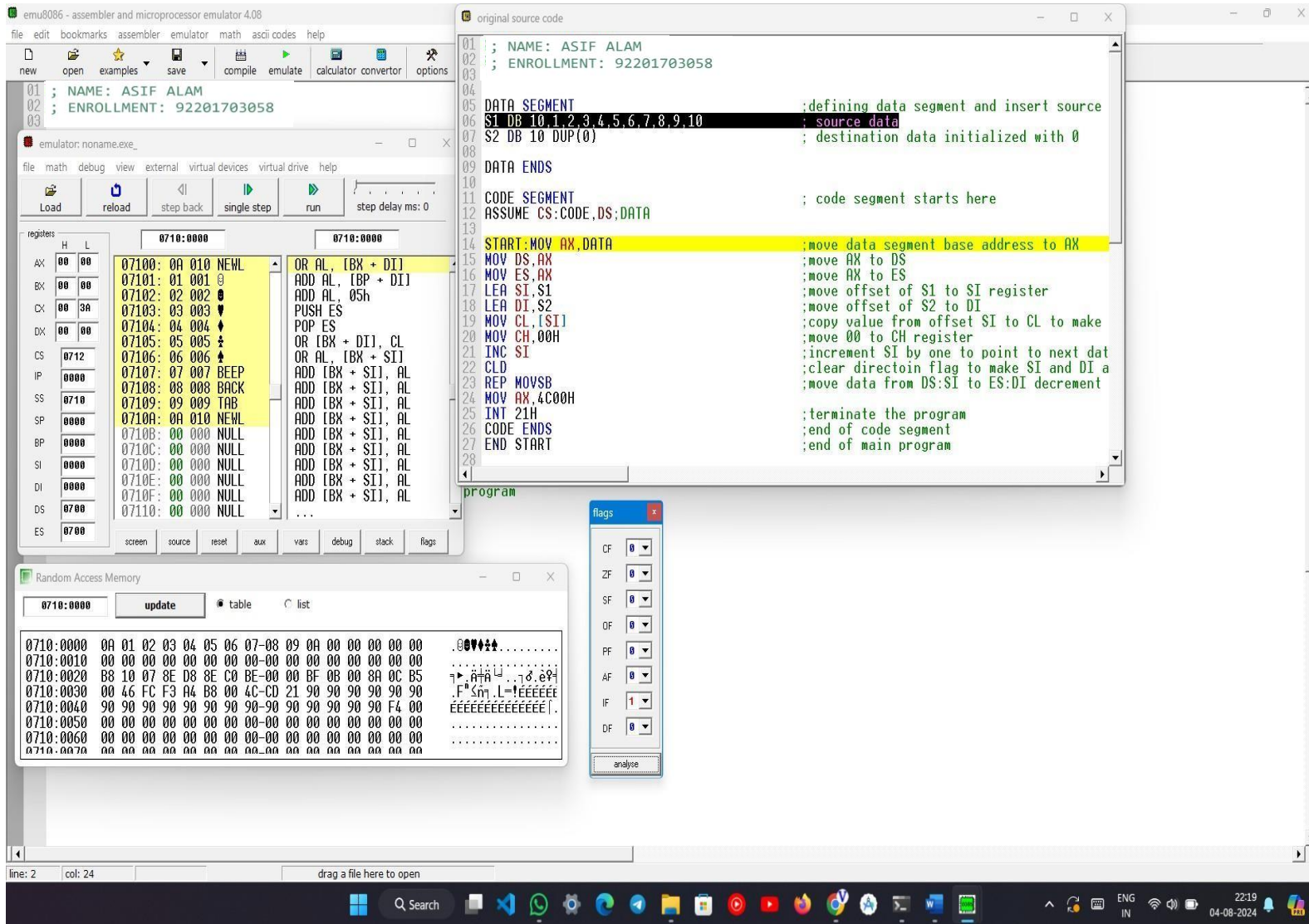
Aim: To perform Data Transfer Operations

Experiment No: 03

Date:

Enrolment No: 92201703171

Step:5 –Check the source array of data and update memory as per location.



The screenshot displays the emu8086 emulator interface. The main window shows the assembly code for a program named 'ASIF ALAM' with enrollment number '92201703058'. The code includes a data segment with source data and a code segment with instructions to move data from the source to the destination.

Assembly Code:

```

01 ; NAME: ASIF ALAM
02 ; ENROLLMENT: 92201703058
03
04
05 DATA SEGMENT ;defining data segment and insert source
06 S1 DB 10,1,2,3,4,5,6,7,8,9,10 ; source data
07 S2 DB 10 DUP(0) ; destination data initialized with 0
08
09 DATA ENDS
10
11 CODE SEGMENT ; code segment starts here
12 ASSUME CS:CODE,DS:DATA
13
14 START:MOV AX,DATA ;move data segment base address to AX
15 MOV DS,AX ;move AX to DS
16 MOV ES,AX ;move AX to ES
17 LEA SI,S1 ;move offset of S1 to SI register
18 LEA DI,S2 ;move offset of S2 to DI
19 MOV CL,[SI] ;copy value from offset SI to CL to make
20 MOV CH,00H ;move 00 to CH register
21 INC SI ;increment SI by one to point to next dat
22 CLD ;clear directoin flag to make SI and DI a
23 REP MOVSB ;move data from DS:SI to ES:DI decrem
24 MOV AX,4C00H
25 INT 21H ;terminate the program
26 CODE ENDS ;end of code segment
27 END START ;end of main program
28

```


The registers window shows the state of the 8086 registers. The DS register is 0710:0000, and the SI register is 0000. The memory window shows the data segment starting at 0710:0000, with the source data being copied to the destination.

Registers:

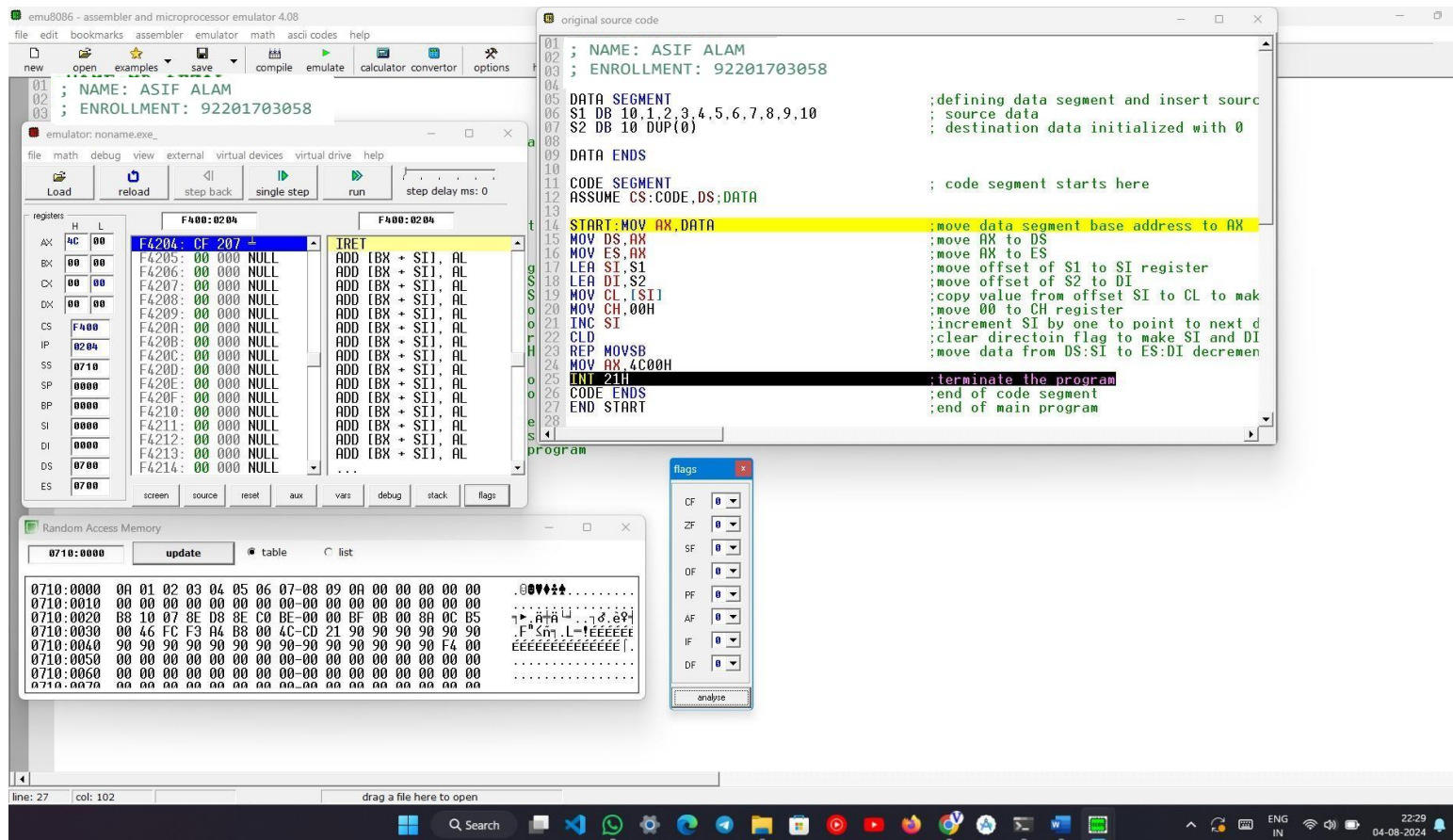
Register	Value
AX	0000
CX	0000
DX	0000
SI	0000
DI	0000
DS	0710:0000
ES	0710:0000

Memory:

Address	Value
0710:0000	0A 01 02 03 04 05 06 07-08 09 0A 00 00 00 00 00
0710:0010	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
0710:0020	B8 10 07 8E D8 8E C0 BE-00 00 BF 0B 00 8A 0C B5
0710:0030	00 46 FC F3 A4 B8 00 4C-CD 21 90 90 90 90 90
0710:0040	90 90 90 90 90 90 90 90-90 90 90 90 90 F4 00
0710:0050	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
0710:0060	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00
0710:0070	00 00 00 00 00 00 00 00-00 00 00 00 00 00 00

 Marwadi University Marwadi Chandarana Group	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim: To perform Data Transfer Operations	
Experiment No: 03	Date:	Enrolment No: 92201703171

Step:6 – Single step Debug (F8), Check the register CX and Flag DF.



The screenshot displays the emu8086 emulator interface. The main window shows the assembly code for a program named 'ASIF ALAM' with enrollment number '92201703058'. The code includes a data segment with source data (S1) and a code segment with instructions for moving data, incrementing registers, and terminating the program. The registers window shows the state of various registers, including AX, BX, CX, DX, SI, DI, and ES. The memory window shows the random access memory (RAM) starting at address 0710:0000. The flags window shows the status of various flags, including CF, ZF, SF, OF, PF, AF, IF, and DF.

Assembly Code:

```

01 ; NAME: ASIF ALAM
02 ; ENROLLMENT: 92201703058
03
04
05 DATA SEGMENT
06 S1 DB 10,1,2,3,4,5,6,7,8,9,10 ;defining data segment and insert source
07 S2 DB 10 DUP(0) ; source data
08 ; destination data initialized with 0
09
10 DATA ENDS
11
12 CODE SEGMENT
13 ASSUME CS:CODE,DS:DATA ; code segment starts here
14
15 START:MOV AX,DATA ;move data segment base address to AX
16 MOV DS,AX ;move AX to DS
17 MOV ES,AX ;move AX to ES
18 LEA SI,S1 ;move offset of S1 to SI register
19 LEA DI,S2 ;move offset of S2 to DI
20 MOV CL,[SI] ;copy value from offset SI to CL to make
21 INC SI ;increment SI by one to point to next d
22 MOV CH,00H ;move 00 to CH register
23 REP MOVSB ;clear directoin flag to make SI and DI
24 MOV AX,4C00H ;move data from DS:SI to ES:DI decremen
25 INT 21H ;terminate the program
26 CODE ENDS ;end of code segment
27 END START ;end of main program
28
29 program

```

Registers:

Register	Value
AX	0000
BX	0000
CX	0000
DX	0000
SI	0000
DI	0000
ES	0000

Flags:

Flag	Value
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	0
DF	0

Memory:

Address	Value
0710:0000	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0710:0010	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0710:0020	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
0710:0030	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
0710:0040	40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
0710:0050	50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
0710:0060	60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
0710:0070	70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F

Conclusion: