Activity No. 4

DATA TRANSFERS AND STRING INSTRUCTIONS

1. Objective:

To create programs using the data transfer and string instructions

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Describe the data transfer instructions

2.2 Describe the modes of addressing

2.3 Create programs using the different transfers instructions

2.4 Create programs using the string instructions

3. Discussion :

DATA TRANSFER INSTRUCTIONS

These instructions are used to move or transfer the values of the operators from a source to a destination. Table 3.1 lists the data transfer instructions and their description.

CODE	DESCRIPTION	CODE	DESCRIPTION
IN	Input byte or word from I/O port	**LSS**	Load pointer using SS
LAHF	Load AH from flags	**MOVSX**	Move with sign extended
LDS	Load pointer using data segment	**MOVZX**	Move with zero extended
LEA	Load effective address	**POPAD**	Pop all double (32-bit) register
LES	Load pointer using extra segment	**POPD**	Pop double register
MOV	Move to/from register/memory	**POPFD**	Pop double flag register
OUT	Output byet or word to I/O port	**PUSHAD**	Push all double registers
POP	Pop word off stack	**PUSHD**	Push double register
POPF	Pop flags off stack	**PUSHFD**	Push double flag register
PUSH	Push word onto stack	**BSWAP**	Byte swap
PUSHF	Push flags onto stack	**MOV**	Move to/from control register

SAHF	Store AH into flags	**POPA**	Pop all registers
XCHG	Exchange byte or word	**PUSHA**	Push all register
XLAT	Translate byte	**LFS**	Load pointer using FS
INS	Input string from port	**LGS**	Load pointer using GS
OUTS	Output string to port		

Table 3.1- The Data Transfer Instructions

Each instruction can be used with different modes of addressing. Addressing modes specifies the ways in which data is transferred. Table 3.1 shows the different addressing modes and their description.

Type	Description	Instruction	Address Generation

Register	Transfers a copy of a byte or word from the source register or contents of memory location to the destination register or memory location.	MOV AX, BX
		MOV AH, BH
		MOV CX, DX
			Source → Destination
			BX → **AX**

Immediate	Transfers the source, an immediate byte or word data into the destination register or memory location.	MOV CH, 3AH
		MOV AL, 22H
		MOV BX, 1234H
			Source → Destination
			3A → **CH**

Direct	Moves a byte or word between a memory location and a register.	MOV [1234H], AX MOV BX,[500H]	Source → Destination AX → 1234H DS X 10H + DISP 10000 +1234H = 11234H

Register Indirect	Transfer a byte or word between a register and a memory location addressed by an index or base register	MOV [BX], CL
		MOV CX, [BX]
		MOV AL, [BX]
			Source → Destination
			CL → **[BX]**
			DS X 10H + BX
			10000 +0300H = 10300H

Base-plus-index Addressing	Transfers a byte or word between a register and the memory location addressed by a register (BP or BX) plus an index register (DI or SI)	MOV [BX+SI],BP
		MOV BP, [BX+SI]
		MOV [BX+DI],CL
			Source → Destination
			BP → BX+SI
			DS X 10H + BX + SI
			10000+0300H+0200H= 10500H

Register Relative Addressing	Moves a byte or word between a register and the memory location addressed by an index or base register plus a displacement.	MOV CL, [BX+4] MOV AX, [BX+4]	Source → Destination BX+4 → CL DS X 10H + BX+4 10000+0300H+4 = 10304H
Base-Index-plus-Displacement Addressing	Transfer a byte or word between a register and the memory location addressed by an index register plus a displacement.	MOV ARRAY[BX+SI], DX	Source → Destination BX+SI+4 → CL DS X 10H + ARRAY+BX+SI 10000+1000H+0300H+0200H = 11500H

Figure 3.2- Addressing Modes and their description

STRING INSTRUCTION

CODE	DESCRIPTION	CODE	DESCRIPTION
CMPS	Compare byte or word string	REPE(REPZ)	Repeat while equal (zero)
LODS	Load byte or word string	REPNE(REPNZ)	Repeat while not equal (not zero)
MOVS	Move byte or word string	SCAS	Scan byte or word string
MOVSB (MOVSW)	Move byte string (word string)	STOS	Store byte or word string
REP	Repeat		

Figure 3.3- String Instructions

4. Resources:

PC
TASM

5. Procedure:

Using any text editor, write the following programs. Execute and show the output.
Use the space provide on the Data and Results.

1.

```
.MODEL SMALL
.STACK 64
.DATA
    MSG DB 'Hi! How are you?$'
.CODE
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 02H
    MOV DH, 12
    MOV DL, 18
    INT 10H

    MOV AH,09H
    MOV BX, 97H
    MOV CX, 10H
    INT 10H

    MOV AH, 09H
    LEA DX, MSG
    INT 21H
    MOV AH, 4CH
    INT 21H
    END
    2.
dosseg
     .model small
     .stack
     .data
         msg1 db 13,10,"Enter a character:$"
         msg2 db 13,10,"The character you entered is:$"
     .code
 main proc
     mov ax,@data
     mov ds,ax

     lea dx,msg1
     mov ah,09h
     int 21h

     mov ah,01h
     int 21h
     mov bl,al

     lea dx,msg2
     mov ah,09h
```

```
    int 21h
    mov dl,bl
    mov ah,02h
    int 21h

    mov ax,4c00h
    int 21h

main endp
     end
    3.

    dosseg

        .model small

        .stack

        .data

            msg1 db 13,10,"Enter a string with dollar symbol as a break:$"

            msg2 db 13,10,"Reverse of the string is:$"

            strg db 20 DUP(0)

            restr db 20 DUP(0)

        .code

     main proc

        mov ax,@data

        mov ds,ax

        mov es,ax

        mov di,00

        lea dx,msg1

        mov ah,09h

        int 21h

     read:mov ah,01h

        int 21h

        cmp al,24h
```

```
        je next
        inc di
        mov strg[di],al
        jmp read
    next: mov si,00
    start:cmp di,0
        je dmsg2
        mov al,strg[di]
        mov restr[si],al
        inc si
        dec di
        jmp start
    dmsg2:lea dx,msg2
        mov ah,09h
        int 21h
     dis:mov al,restr[di]
        cmp al,0
        je ou
        mov dl,al
        mov ah,02h
        int 21h
        inc di
        jmp dis
     ou: mov ax,4c00h
        int 21h
main endp
    end
```

Course: BET-CPET	**Activity No.: 4**
Group No.:	**Section: 2A**
Group Members: Joseph C. Arenas	**Date Performed: March 18, 2025**
	Date Submitted: March 19, 2025
	Instructor: Ma'am Delos Trinos

6. DATA AND RESULTS:

Problem 1:

a. V
b. V
c. I
d. I
e. I

Problem 2:

```
DOSSEG
.MODEL SMALL
.STACK 64H

.DATA
MSG DB 'Hi! How do you do?$'

.CODE
START:
   MOV AX, @DATA
   MOV DS, AX

   MOV AH, 02H
   MOV DH, 12
   MOV DL, 18
   INT 10H

   MOV AH, 09H
   MOV BX, 97H
   MOV CX, 12H
   INT 10H

   MOV AH, 09H
   LEA DX, MSG
   INT 21H

   MOV AH, 4CH
   INT 21H
   END START
```

Problem 3:

```
DOSSEG
.MODEL SMALL
.STACK 200H

.DATA
MSG DB 0DH, 0AH, "ENTER 10 CHARACTERS (Please end with dollar sign): $"
MSG1 DB 0DH, 0AH, "YOU HAVE ENTERED: $"
CHARACS DB 11 DUP (0)

.CODE
START:
   MOV AX, @DATA
   MOV DS, AX
   MOV DI, 00

   MOV AH, 09H
   LEA DX, MSG
   INT 21H

READ:
   MOV AH, 01H
   INT 21H

   CMP AL, 24H
   JE DISPLAY_MSG

   MOV CHARACS[DI], AL
   INC DI
   JMP READ

DISPLAY_MSG:
   MOV AH, 09H
   LEA DX, MSG1
   INT 21H

   MOV DI, 0
PRINT_CHAR:
   CMP DI, 10
   JGE END_TIME

   MOV AH, 02H
   MOV DL, CHARACS[DI]
   INT 21H

   INC DI
```

```
    JMP PRINT_CHAR

END_TIME:
    MOV AX, 4C00H
    INT 21H

END START
```

Problem 4:

```
DOSSEG
.MODEL SMALL
.STACK 100H

.DATA
MSG1 DB 0DH, 0AH, "Enter a string with dollar symbol as a break: $"
MSG2 DB 0DH, 0AH, "Modified string is: $"
BUFFER DB 100 DUP(0)

.CODE
START:
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 09H
    LEA DX, MSG1
    INT 21H

    MOV SI, 0

READ_STRING:
    MOV AH, 01H
    INT 21H

    CMP AL, 24H
    JE MODIFY_STRING

    MOV BUFFER[SI], AL
    INC SI
    JMP READ_STRING

MODIFY_STRING:
    MOV BUFFER[SI], '$'
    MOV SI, 0

    MOV AH, 09H
    LEA DX, MSG2
```

```
    INT 21H

CONVERT_CASE:
    MOV AL, BUFFER[SI]
    CMP AL, '$'
    JE DISPLAY_RESULT

    CMP AL, 'a'
    JB CHECK_UPPERCASE
    CMP AL, 'z'
    JA CHECK_UPPERCASE
    SUB AL, 32
    JMP STORE_CHAR

CHECK_UPPERCASE:
    CMP AL, 'A'
    JB STORE_CHAR
    CMP AL, 'Z'
    JA STORE_CHAR
    ADD AL, 32

STORE_CHAR:
    MOV BUFFER[SI], AL
    INC SI
    JMP CONVERT_CASE

DISPLAY_RESULT:
    MOV AH, 09H
    LEA DX, BUFFER
    INT 21H

    MOV AX, 4C00H
    INT 21H

END START
```

PROBLEMS:

1. Indicate wheter each of the following is valid (V-valid) or invalid(I-invalid)
   a. MOV DX, 7F65H
   b. MOV SI, -1
   c. MOV SS,DS
   d. MOV SI, CL
   e. MOV ECX, 6F23458H
2. Modify Program 1 to display the string “Hello!, How do you do?”
3. Develop and execute a program to read 10 characters from keyboard
   SampleOutput:

   Enter 10 characters:1234567890

4. Write a program which converts string lower case character to upper case characters and upper case character to lower case characters.
   SampleOutput:

   Enter a string with dolar symbol as a break:eNgliSH
   Modified string is:EnGLIsh

7. Conclusion:

In conclusion, the interrupt 21h function ah= 1 significantly helps developers to include a user input into their program. By utilizing this function, various tasks can be achieved by programmers in order to formulate programs for the end-user’s benefit.

8. Assessment (Rubric for Laboratory Performance):