Department of Computer Science Engineering Data Science

Academic Year: 2022-23 Semester: IV
Class / Branch: S.E.D.S. **Subject: Microprocessor Lab**

Experiment No. 6

1. **Aim:** Assembly program to perform overlapping block transfer.

2. **Software used:** tasm, tlink ,td, dosemu

3. **Theory :-**

String Instructions

The string instructions facilitate operations on sequences of bytes or words. None of them take an explicit operand; instead, they all work implicitly on the *source* and/or *destination* strings. The current element (byte or word) of the source string is at DS:SI, and the current element of the destination string is at ES:DI. Each instruction works on one element and then automatically adjusts SI and/or DI;if the Direction flag is clear, then the index is incremented, otherwise it is decremented (when working with overlapping strings it is sometimes necessary to work from back to front, but usuallyyou should leave the Direction flag clear and work on strings from front to back).

Operations that can be performed with string instructions:

- copy a string into another string
- search a string for a particular byte or word
- store characters in a string
- compare strings of characters alphanumerically

Moving (Copying) Strings

Instructions:

MOVSB - copies contents of BYTE given by **DS:SI** into **ES:DI**
MOVSW - copies contents of WORD given by **DS:SI** into **ES:DI**
MOVSD - copies contents of DOUBLE WORD given by **DS:SI** into **ES:DI**

Storing Strings

Instructions:

STOSB - copies contents of AL to BYTE address given by ES:DI.DI is incremented/decremented by 1.

STOSW - copies the contents of AX to the WORD address given by ES:DI.DIis incremented/decremented by 2.

STOSD - copies contents of EAX to the DOUBLE WORD address given by ES:DI.DI is incremented/decremented by 4.

Load String

Instructions:

LODSB - moves the BYTE at address DS:SI into AL. SI is incremented/decremented by 1.

LODSW - moves the WORD at address DS: SI into AX. SI is incremented/decremented by 2.

LODSD - moves the DOUBLE WORD at address DS:SI into EAX. SI is incremented/decremented by 4.

Compare String

Instructions:

CMPSB - compares BYTE at ES:DI with BYTE at DS:SI and sets flags.

CMPSW - compares WORD at ES:DI with WORD at DS:SI and sets flags.

CMPSD - compares DOUBLE WORD at ES:DI with WORD at DS:SI and sets flags.

REPE/REPNE/REPZ/REPNZ

To work on an entire string at a time, each string instruction can be accompanied by a repeat prefix, either REP or one of REPE and REPNE (or their synonyms REPZ and REPNZ).

REPE and **REPZ** are mnemonics for the same prefix; they stand for Repeat if Equal and Repeat if Zero respectively. REPE/REPZ causes the succeeding string instruction to be repeated as long as the compared bytes or words are equal (ZF = 1) **and** CX is not yet counted down to zero.

The **REPNE** and the **REPNZ** instructions stand for Repeat if Not Equal and Repeat if Not Zero respectively and cause the string instruction to be repeated until the compared bytes or words are equal (ZF = 1) **or** until CX = 0 (end of string.)

CX is decremented and the zero flag is tested after each operation.

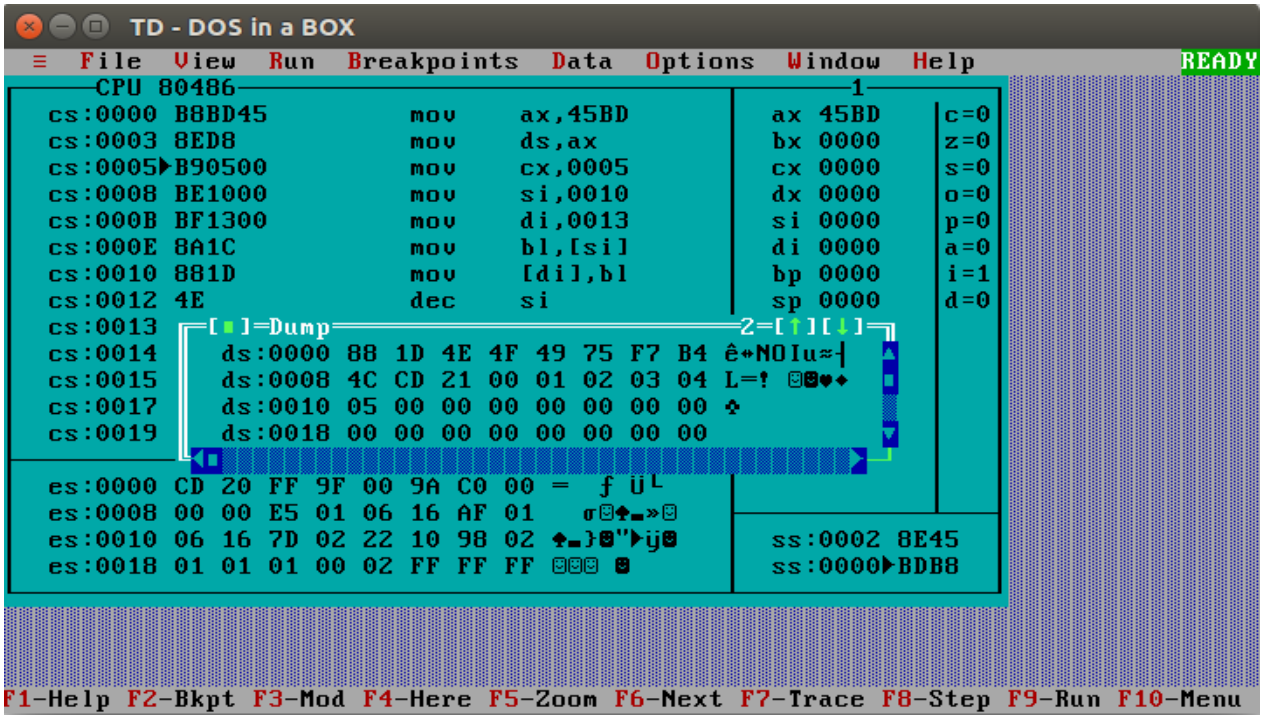
Usage REPE/REPE/REPZ/REPNZ <String Instruction>

4. Program

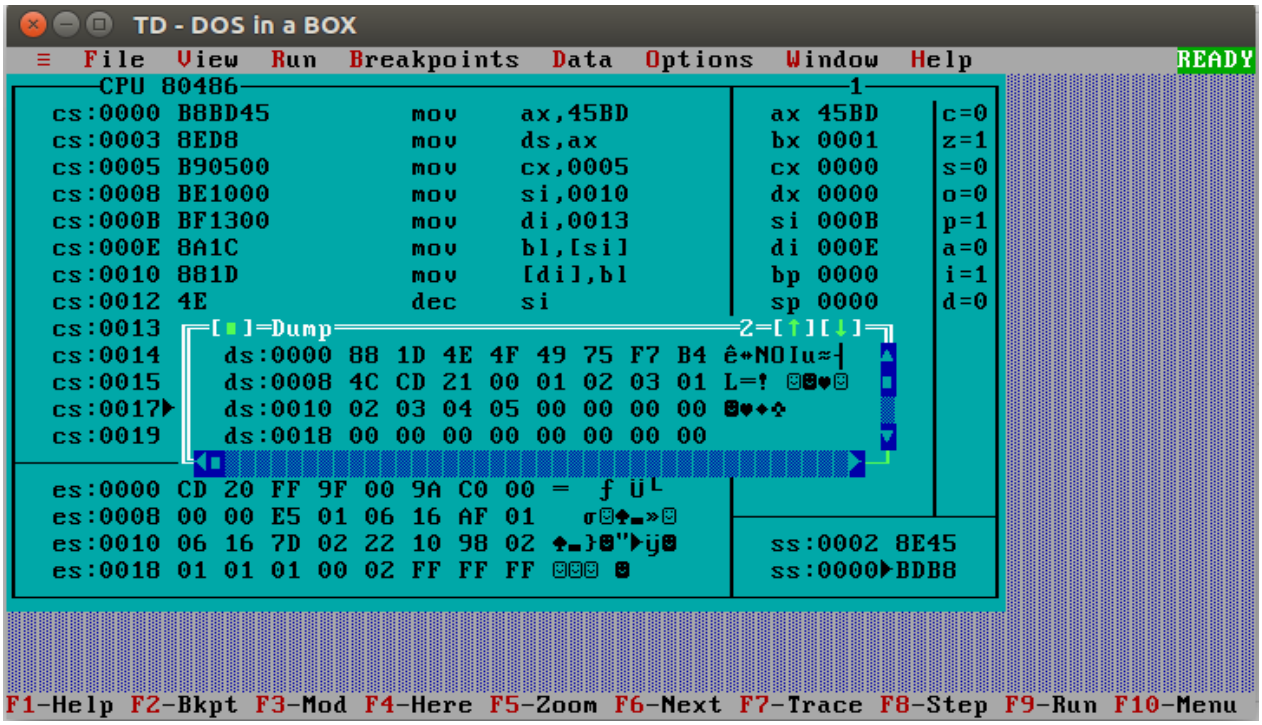
```
model small
.data

   x db 01h,02h,03h,04h,05h ; initialize data segments memory locations

.code

   start:
```

```
    mov ax,@data ; initialize ds to point to start of the memory
    mov ds,ax ; set aside for storing of data
    mov cx,05h ; load counter
    lea si,x+04 ; si pointer pointed to top of the memory block
    lea di,x+04+03 ; 03 is displacement of over lapping, di pointed to
                ;the top of the destination block
    up: mov bl,[si] ; move the si content to bl register
    mov [di],bl ; move the bl register to content of di
    dec si ; update si and di
    dec di
    dec cx ; decrement the counter till it becomes zero
    jnz up
    mov ah,4ch
    int 21h
end start
```

Output:



Screen-shot before transfer



Screen-shot after overlapping transfer

5. Conclusion :-