CSE 2006

MICROPROCESSOR AND INTERFACING

Task – 4

L11+L12 | SJT516

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105

ALPs to be completed

Problem 1: **Write a program to reverse the given string and store at the same location.**

Problem 2: **Write a program using the LOOP instruction with indirect addressing that copies a string from source to target, reversing the character order in the process.**

Problem 1

Aim: To write a program which reverses a given string and store it at the same location.

Algorithm:

```
1. Define the string to be reversed
2. Move the address of first element in SI
3. Move the addess of last element in DI
4. Loop:
        1. Move the element in SI to AL
        2. Move the element in DI to BL
        3. Exchange the two elements
        4. Move the element in BL, to the address in SI
        5. Move the element in AL, to the address in DI
        6. Increase SI to take the next element in the array
        7. Decrease DI to take the next element in the array
```

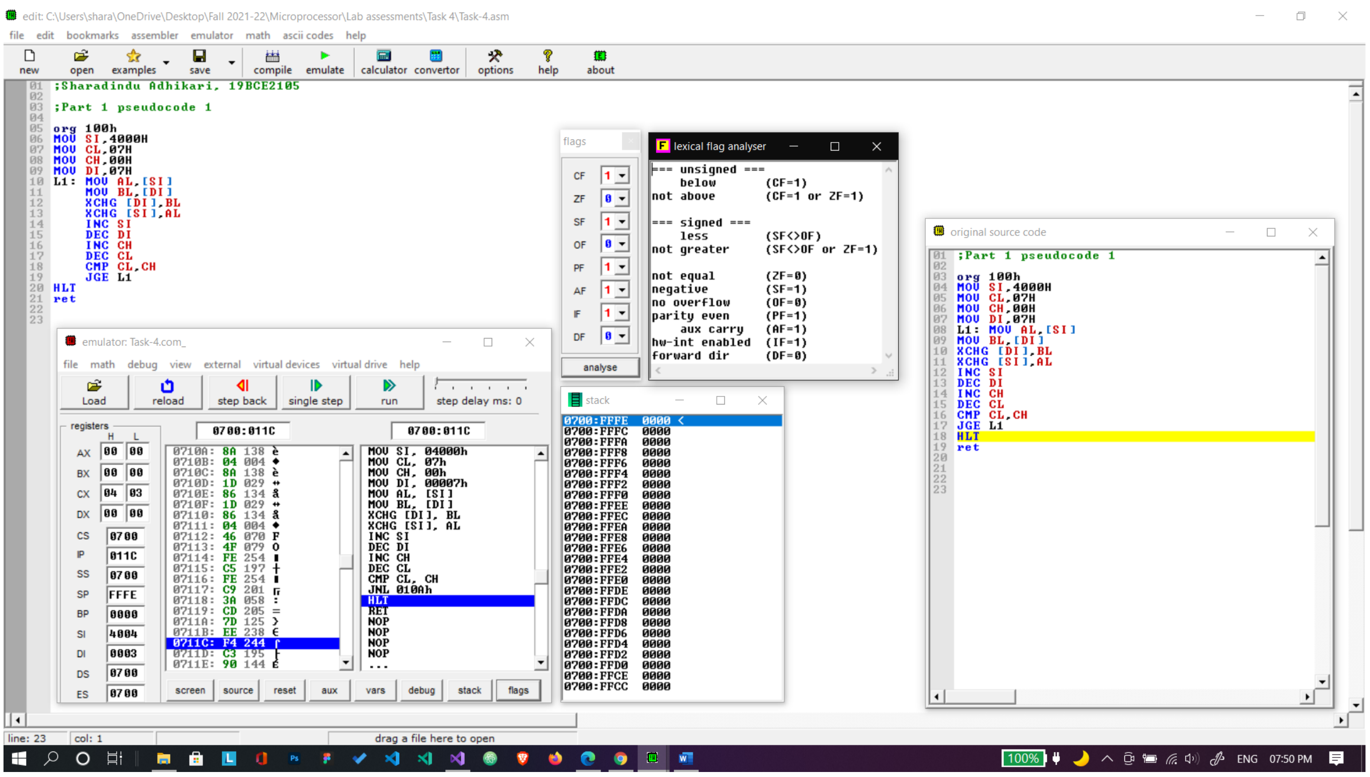
Codes and screenshots:

<u>P1 Code1:</u>

```
org 100h
MOV SI,4000H
MOV CL,07H
MOV CH,00H
MOV DI,07H
L1: MOV AL,[SI]
    MOV BL,[DI]
    XCHG [DI],BL
    XCHG [SI],AL
    INC SI
    DEC DI
```

sharadindu.adhikari2019@vitstudent.ac.in

```
    INC CH
    DEC CL
    CMP CL,CH
    JGE L1
HLT
ret
```



edit: C:\Users\shara\OneDrive\Desktop\Fall 2021-22\Microprocessor\Lab assessments\Task 4\Task-4.asm

```
;Sharadindu Adhikari, 19BCE2105

;Part 1 pseudocode 1

org 100h
MOV SI,4000H
MOV CL,07H
MOV CH,00H
MOV DI,07H
L1: MOV AL,[SI]
    MOV BL,[DI]
    XCHG [DI],BL
    XCHG [SI],AL
    INC SI
    DEC DI
    INC CH
    DEC CL
    CMP CL,CH
    JGE L1
HLT
ret
```

emulator: Task-4.com_

```
registers
     H   L
AX   00  00
BX   00  00
CX   04  03
DX   00  00
CS   0700
IP   011C
SS   0700
SP   FFFE
BP   0000
SI   4004
DI   0003
DS   0700
ES   0700
```

```
0700:011C
0710A: 8A 138 è
0710B: 04 004 ♦
0710C: 8A 138 è
0710D: 1D 029 ↔
0710E: 86 134 å
0710F: 1D 029 ↔
07110: 86 134 å
07111: 04 004 ♦
07112: 46 070 F
07113: 4F 079 O
07114: FE 254 ■
07115: C5 197 ┼
07116: FE 254 ■
07117: C9 201 ╔
07118: 3A 058 :
07119: CD 205 ═
0711A: 7D 125 }
0711B: EE 238 ε
0711C: F4 244 ⌠
0711D: C3 195 ├
0711E: 90 144 É
```

```
0700:011C
MOV SI, 04000h
MOV CL, 07h
MOV CH, 00h
MOV DI, 00007h
MOV AL, [SI]
MOV BL, [DI]
XCHG [DI], BL
XCHG [SI], AL
INC SI
DEC DI
INC CH
DEC CL
CMP CL, CH
JNL 010Ah
HLT
RET
NOP
NOP
NOP
NOP
...
```

```
flags
CF 1
ZF 0
SF 1
OF 0
PF 1
AF 1
IF 1
DF 0
```

lexical flag analyser

```
=== unsigned ===
    below           (CF=1)
not above           (CF=1 or ZF=1)

=== signed ===
    less            (SF<>OF)
not greater         (SF<>OF or ZF=1)

not equal           (ZF=0)
negative            (SF=1)
no overflow         (OF=0)
parity even         (PF=1)
    aux carry       (AF=1)
hw-int enabled      (IF=1)
forward dir         (DF=0)
```

stack

```
0700:FFFE  0000 <
0700:FFFC  0000
0700:FFFA  0000
0700:FFF8  0000
0700:FFF6  0000
0700:FFF4  0000
0700:FFF2  0000
0700:FFF0  0000
0700:FFEE  0000
0700:FFEC  0000
0700:FFEA  0000
0700:FFE8  0000
0700:FFE6  0000
0700:FFE4  0000
0700:FFE2  0000
0700:FFE0  0000
0700:FFDE  0000
0700:FFDC  0000
0700:FFDA  0000
0700:FFD8  0000
0700:FFD6  0000
0700:FFD4  0000
0700:FFD2  0000
0700:FFD0  0000
0700:FFCE  0000
0700:FFCC  0000
```

original source code

```
;Part 1 pseudocode 1

org 100h
MOV SI,4000H
MOV CL,07H
MOV CH,00H
MOV DI,07H
L1: MOV AL,[SI]
MOV BL,[DI]
XCHG [DI],BL
XCHG [SI],AL
INC SI
DEC DI
INC CH
DEC CL
CMP CL,CH
JGE L1
HLT
ret
```

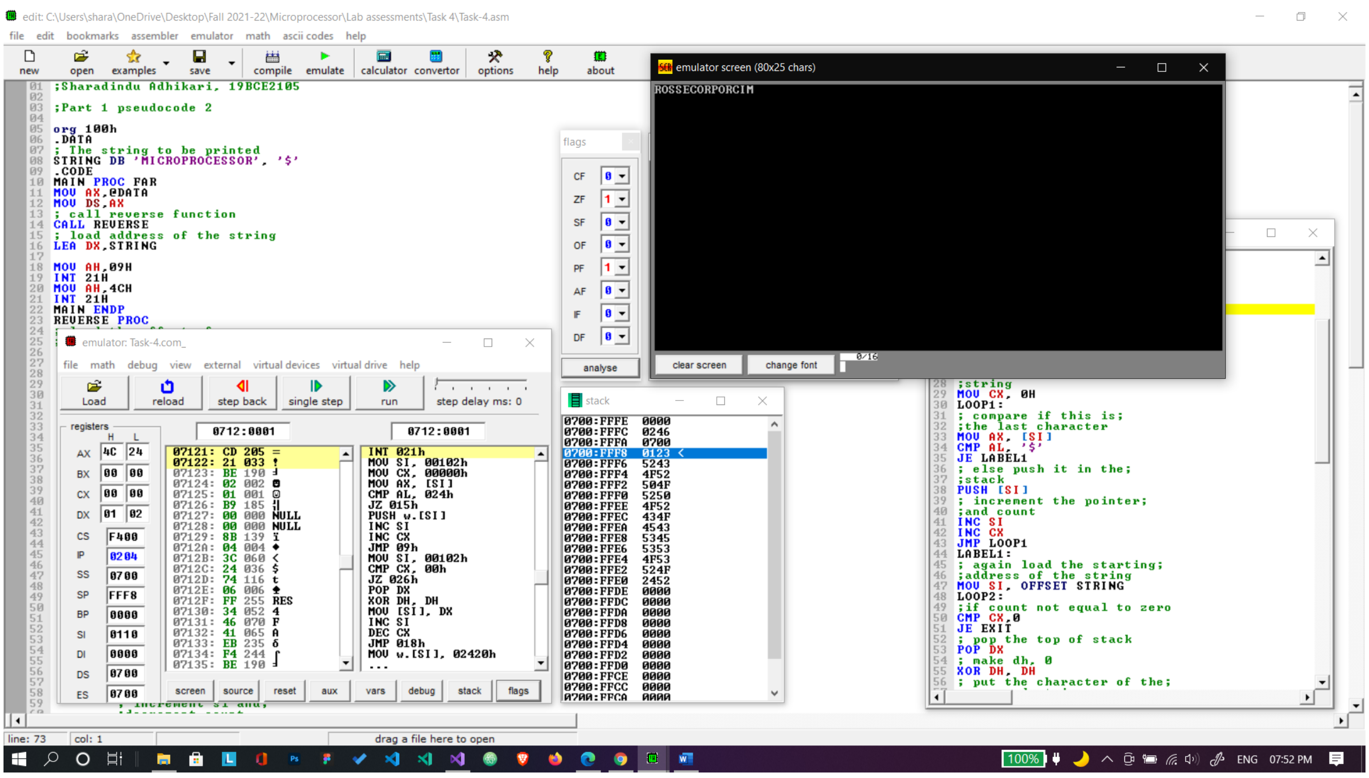
line: 23   col: 1   drag a file here to open

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
analyse
100%
ENG 07:50 PM

P1_code2:

```
org 100h
.DATA
; The string to be printed
STRING DB 'MICROPROCESSOR', '$'
.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
; call reverse function
CALL REVERSE
; load address of the string
LEA DX,STRING

MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
```

```
MAIN ENDP
REVERSE PROC
; load the offset of
; the string
    MOV SI, OFFSET STRING
    ; count of characters of the;
    ;string
    MOV CX, 0H

    LOOP1:
    ; compare if this is;
    ;the last character
    MOV AX, [SI]
    CMP AL, '$'
    JE LABEL1
    ; else push it in the;
    ;stack
    PUSH [SI]
    ; increment the pointer;
    ;and count
    INC SI
    INC CX
    JMP LOOP1

        LABEL1:
        ; again load the starting;
        ;address of the string
        MOV SI, OFFSET STRING
        LOOP2:
        ;if count not equal to zero
        CMP CX,0
        JE EXIT
        ; pop the top of stack
        POP DX
        ; make dh, 0
        XOR DH, DH
        ; put the character of the;
        ;reversed string

        MOV [SI], DX
        ; increment si and;
        ;decrement count
        INC SI
        DEC CX
        JMP LOOP2
        EXIT:
        ; add $ to the end of string
        MOV [SI],'$ '
        RET
        REVERSE ENDP
        END MAIN
ret
```



```
;Sharadindu Adhikari, 19BCE2105

;Part 1 pseudocode 2

org 100h
.DATA
; The string to be printed
STRING DB 'MICROPROCESSOR', '$'
.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
; call reverse function
CALL REVERSE
; load address of the string
LEA DX,STRING

MOV AH,09H
INT 21H
MOV AH,4CH
INT 21H
MAIN ENDP
REVERSE PROC
```

```
;string
MOV CX, 0H
LOOP1:
; compare if this is;
;the last character
MOV AX, [SI]
CMP AL, '$'
JE LABEL1
; else push it in the;
;stack
PUSH [SI]
; increment the pointer;
;and count
INC SI
INC CX
JMP LOOP1
LABEL1:
; again load the starting;
;address of the string
MOV SI, OFFSET STRING
LOOP2:
;if count not equal to zero
CMP CX,0
JE EXIT
; pop the top of stack
POP DX
; make dh, 0
XOR DH, DH
; put the character of the;
```

emulator: Task-4.com_

Registers: AX 4C 24, BX 00 00, CX 00 00, DX 01 02, CS F400, IP 0204, SS 0700, SP FFF8, BP 0000, SI 0110, DI 0000, DS 0700, ES 0700

Flags: CF 0, ZF 1, SF 0, OF 0, PF 1, AF 0, IF 0, DF 0

```
0712:0001                      0712:0001
07121: CD 205 =               INT 021h
07122: 21 033 !               MOV SI, 00102h
07123: BE 190 ╛               MOV CX, 00000h
07124: 02 002 ☻               MOV AX, [SI]
07125: 01 001 ☺               CMP AL, 024h
07126: B9 185 ╣               JZ 015h
07127: 00 000 NULL            PUSH w.[SI]
07128: 00 000 NULL            INC SI
07129: 8B 139 ï               INC CX
0712A: 04 004 ♦               JMP 09h
0712B: 3C 060 <               MOV SI, 00102h
0712C: 24 036 $               CMP CX, 00h
0712D: 74 116 t               JZ 026h
0712E: 06 006 ♠               POP DX
0712F: FF 255 RES             XOR DH, DH
07130: 34 052 4               MOV [SI], DX
07131: 46 070 F               INC SI
07132: 41 065 A               DEC CX
07133: EB 235 δ               JMP 018h
07134: F4 244 ⌠               MOV w.[SI], 02420h
07135: BE 190 ╛               ...
```

stack

```
0700:FFFE  0000
0700:FFFC  0246
0700:FFFA  0700
0700:FFF8  0123 <
0700:FFF6  5243
0700:FFF4  4F52
0700:FFF2  504F
0700:FFF0  5250
0700:FFEE  4F52
0700:FFEC  434F
0700:FFEA  4543
0700:FFE8  5345
0700:FFE6  5353
0700:FFE4  4F53
0700:FFE2  524F
0700:FFE0  2452
0700:FFDE  0000
0700:FFDC  0000
0700:FFDA  0000
0700:FFD8  0000
0700:FFD6  0000
0700:FFD4  0000
0700:FFD2  0000
0700:FFD0  0000
0700:FFCE  0000
0700:FFCC  0000
0700:FFCA  0000
```

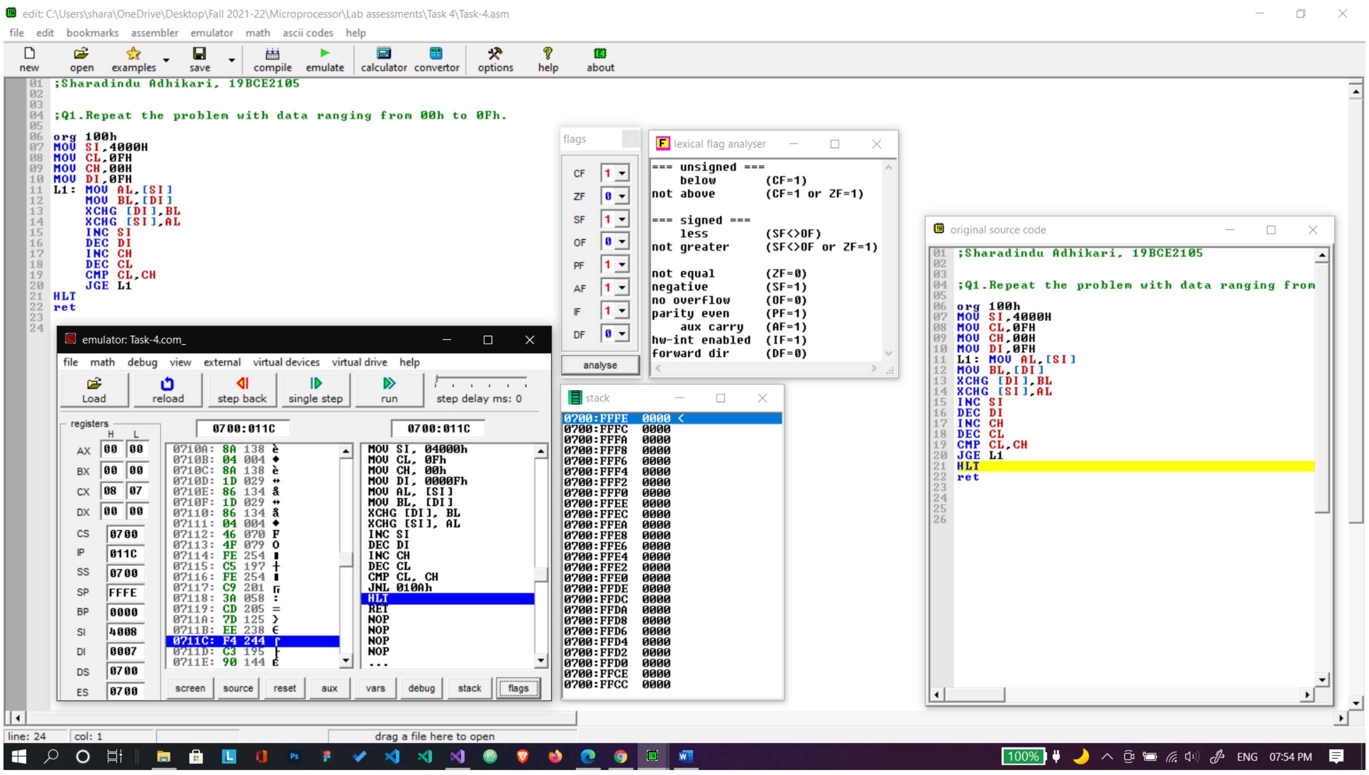
emulator screen (80x25 chars)

```
ROSSECORPORCIM
```

Review Questions:

1. **Repeat the problem with data ranging from 00h to 0Fh.**

```
org 100h
MOV SI,4000H
MOV CL,0FH
MOV CH,00H
MOV DI,0FH
L1: MOV AL,[SI]
    MOV BL,[DI]
    XCHG [DI],BL
    XCHG [SI],AL
    INC SI
    DEC DI
    INC CH
    DEC CL
    CMP CL,CH
    JGE L1
HLT
ret
```



edit: C:\Users\shara\OneDrive\Desktop\Fall 2021-22\Microprocessor\Lab assessments\Task 4\Task-4.asm

```
;Sharadindu Adhikari, 19BCE2105


;Q1.Repeat the problem with data ranging from 00h to 0Fh.

org 100h
MOV SI,4000H
MOV CL,0FH
MOV CH,00H
MOV DI,0FH
L1: MOV AL,[SI]
    MOV BL,[DI]
    XCHG [DI],BL
    XCHG [SI],AL
    INC SI
    DEC DI
    INC CH
    DEC CL
    CMP CL,CH
    JGE L1
HLT
ret
```

emulator: Task-4.com_

```
registers   H   L
AX         00  00
BX         00  00
CX         08  07
DX         00  00
CS         0700
IP         011C
SS         0700
SP         FFFE
BP         0000
SI         4008
DI         0007
DS         0700
ES         0700
```

```
0700:011C

0710A: 8A 138 è
0710B: 04 004 ♦
0710C: 8A 138 è
0710D: 1D 029 ↔
0710E: 86 134 å
0710F: 1D 029 ↔
07110: 86 134 å
07111: 04 004 ♦
07112: 46 070 F
07113: 4F 079 O
07114: FE 254 ■
07115: C5 197 ┼
07116: FE 254 ■
07117: C9 201 ╔
07118: 3A 058 :
07119: CD 205 =
0711A: 7D 125 }
0711B: EE 238 ε
0711C: F4 244 ⌠
0711D: C3 195 ├
0711E: 90 144 É
```

```
0700:011C

MOV SI, 04000h
MOV CL, 0Fh
MOV CH, 00h
MOV DI, 0000Fh
MOV AL, [SI]
MOV BL, [DI]
XCHG [DI], BL
XCHG [SI], AL
INC SI
DEC DI
INC CH
DEC CL
CMP CL, CH
JNL 010Ah
HLT
RET
NOP
NOP
NOP
NOP
...
```

```
flags
CF 1
ZF 0
SF 1
OF 0
PF 1
AF 1
IF 1
DF 0
```

lexical flag analyser

```
=== unsigned ===
    below          (CF=1)
not above          (CF=1 or ZF=1)

=== signed ===
    less           (SF<>OF)
not greater        (SF<>OF or ZF=1)

not equal          (ZF=0)
negative           (SF=1)
no overflow        (OF=0)
parity even        (PF=1)
    aux carry      (AF=1)
hw-int enabled     (IF=1)
forward dir        (DF=0)
```

stack

```
0700:FFFE  0000 <
0700:FFFC  0000
0700:FFFA  0000
0700:FFF8  0000
0700:FFF6  0000
0700:FFF4  0000
0700:FFF2  0000
0700:FFF0  0000
0700:FFEE  0000
0700:FFEC  0000
0700:FFEA  0000
0700:FFE8  0000
0700:FFE6  0000
0700:FFE4  0000
0700:FFE2  0000
0700:FFE0  0000
0700:FFDE  0000
0700:FFDC  0000
0700:FFDA  0000
0700:FFD8  0000
0700:FFD6  0000
0700:FFD4  0000
0700:FFD2  0000
0700:FFD0  0000
0700:FFCE  0000
0700:FFCC  0000
```

original source code

```
;Sharadindu Adhikari, 19BCE2105


;Q1.Repeat the problem with data ranging from

org 100h
MOV SI,4000H
MOV CL,0FH
MOV CH,00H
MOV DI,0FH
L1: MOV AL,[SI]
MOV BL,[DI]
XCHG [DI],BL
XCHG [SI],AL
INC SI
DEC DI
INC CH
DEC CL
CMP CL,CH
JGE L1
HLT
ret
```

line: 24 col: 1 drag a file here to open

file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
analyse
100%
ENG 07:54 PM

2. **What is the role of JGE instruction?**

   It is Used to jump if greater than/equal/not less than instruction satisfies.

3. **Which addressing mode is used here?**

   Immediate Addressing Mode is used in this ALP.

Problem 2

Aim: To write a program using the LOOP instruction with indirect addressing that copies a string from source to target, reversing the character order in the process.

Algorithm:

1. Get a string and remove the characters of the string.
2. Push the characters in a stack and maintain the count of the characters.
3. Load the starting address of the string.
4. POP the top character of the stack until count is not equal to zero.
5. Put the character and reduce the count and increase the address.
6. Continue until the count is greater than zero.
7. Load the effective address of the string in `dx` using `LEA` command.
8. Print the string by calling the interrupt with `9H` in `AH`.
9. The string must be terminated by '`$`' sign.

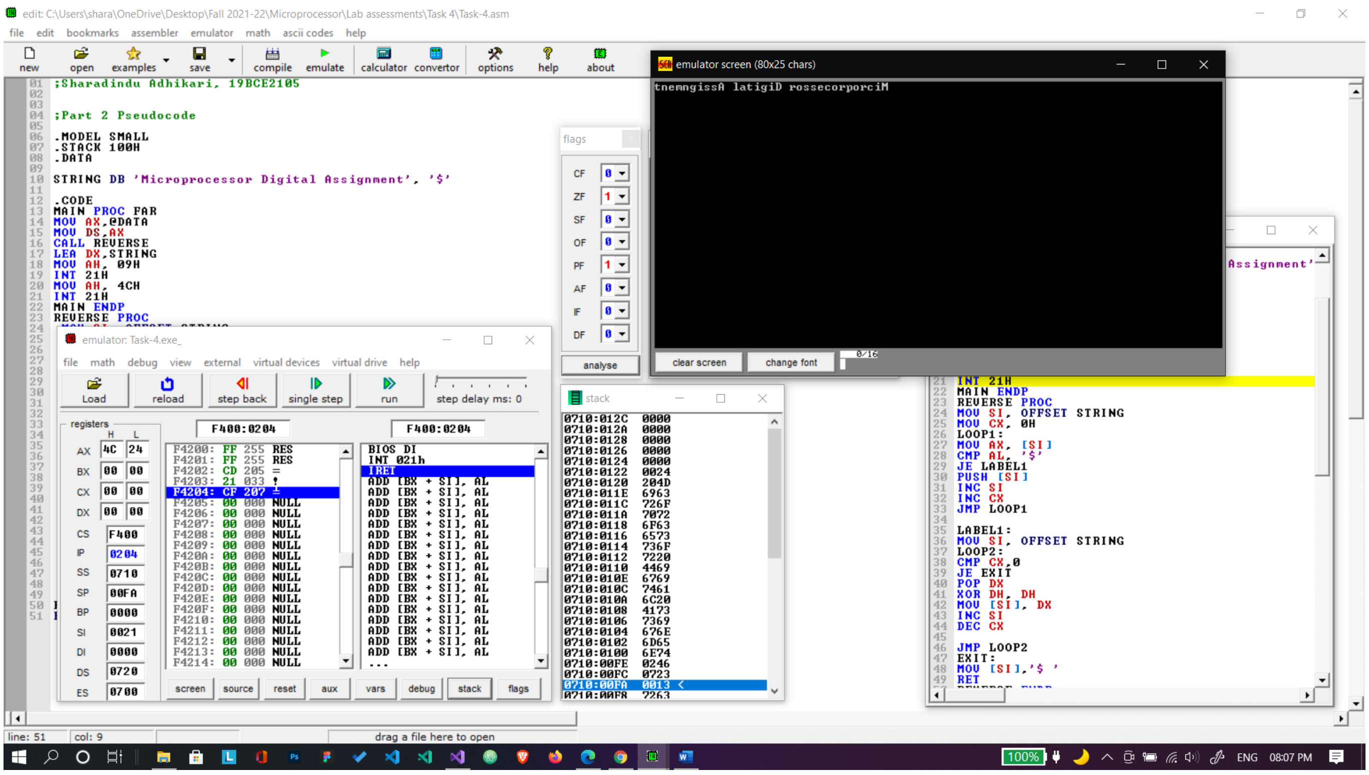
Code and input:

```
.MODEL SMALL
.STACK 100H
.DATA

STRING DB 'Microprocessor Digital Assignment', '$'

.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
CALL REVERSE
LEA DX,STRING
MOV AH, 09H
INT 21H
MOV AH, 4CH
INT 21H
MAIN ENDP
REVERSE PROC
 MOV SI, OFFSET STRING
 MOV CX, 0H
 LOOP1:
 MOV AX, [SI]
 CMP AL, '$'
 JE LABEL1
 PUSH [SI]
 INC SI
 INC CX
 JMP LOOP1

 LABEL1:
 MOV SI, OFFSET STRING
 LOOP2:
 CMP CX,0
 JE EXIT
 POP DX
 XOR DH, DH
 MOV [SI], DX
 INC SI
 DEC CX

 JMP LOOP2
 EXIT:
 MOV [SI],'$ '
 RET
REVERSE ENDP
END MAIN
```

Output:

```
tnemngissA latigiD rossecorporciM
```

Screenshot:



edit: C:\Users\shara\OneDrive\Desktop\Fall 2021-22\Microprocessor\Lab assessments\Task 4\Task-4.asm

```
;Sharadindu Adhikari, 19BCE2105


;Part 2 Pseudocode

.MODEL SMALL
.STACK 100H
.DATA

STRING DB 'Microprocessor Digital Assignment', '$'

.CODE
MAIN PROC FAR
MOV AX,@DATA
MOV DS,AX
CALL REVERSE
LEA DX,STRING
MOV AH, 09H
INT 21H
MOV AH, 4CH
INT 21H
MAIN ENDP
REVERSE PROC
MOV SI, OFFSET STRING
MOV CX, 0H
LOOP1:
MOV AX, [SI]
CMP AL, '$'
JE LABEL1
PUSH [SI]
INC SI
INC CX
JMP LOOP1

LABEL1:
MOV SI, OFFSET STRING
LOOP2:
CMP CX,0
JE EXIT
POP DX
XOR DH, DH
MOV [SI], DX
INC SI
DEC CX

JMP LOOP2
EXIT:
MOV [SI],'$ '
RET
```

emulator screen (80x25 chars)

```
tnemngissA latigiD rossecorporciM
```

emulator: Task-4.exe_

```
registers
     H  L
AX   4C 24
BX   00 00
CX   00 00
DX   00 00
CS   F400
IP   0204
SS   0710
SP   00FA
BP   0000
SI   0021
DI   0000
DS   0720
ES   0700
```

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

flags

```
CF 0
ZF 1
SF 0
OF 0
PF 1
AF 0
IF 0
DF 0
```

stack

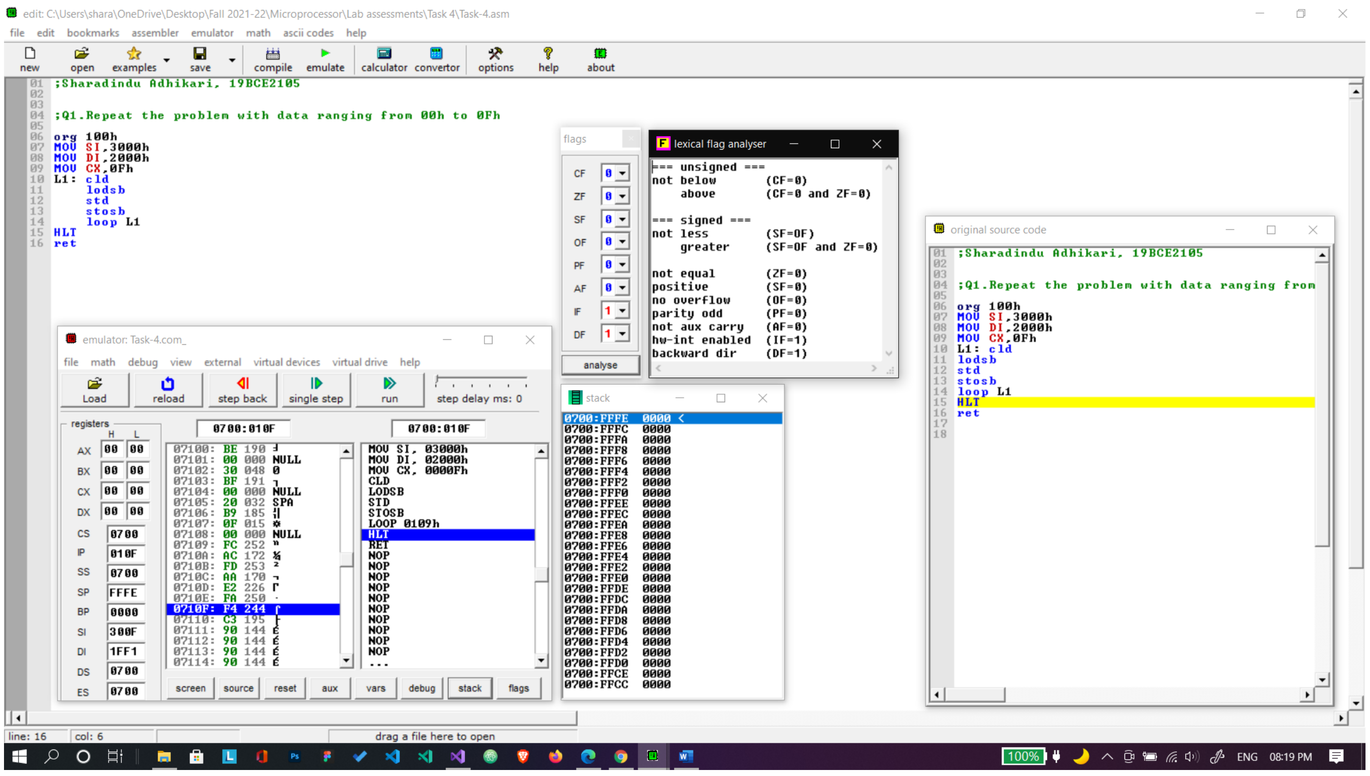
```
0710:012C  0000
0710:012A  0000
0710:0128  0000
0710:0126  0000
0710:0124  0000
0710:0122  0024
0710:0120  204D
0710:011E  6963
0710:011C  726F
0710:011A  7072
0710:0118  6F63
0710:0116  6573
0710:0114  736F
0710:0112  7220
0710:0110  4469
0710:010E  6769
0710:010C  7461
0710:010A  6C20
0710:0108  4173
0710:0106  7369
0710:0104  676E
0710:0102  6D65
0710:0100  6E74
0710:00FE  0246
0710:00FC  0723
0710:00FA  0013 <
0710:00F8  7263
```

Result: The input string has been reversed.

Review questions:

1. **Repeat the problem with data ranging from 00h to 0Fh.**

```
org 100h
MOV SI,3000h
MOV DI,2000h
MOV CX,0Fh
L1: cld
    lodsb
    std
    stosb
    loop L1
HLT
Ret
```


edit: C:\Users\shara\OneDrive\Desktop\Fall 2021-22\Microprocessor\Lab assessments\Task 4\Task-4.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

```
;Sharadindu Adhikari, 19BCE2105


;Q1.Repeat the problem with data ranging from 00h to 0Fh

org 100h
MOV SI,3000h
MOV DI,2000h
MOV CX,0Fh
L1: cld
    lodsb
    std
    stosb
    loop L1
HLT
ret
```

emulator: Task-4.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
screen source reset aux vars debug stack flags

```
     H  L
AX  00 00
BX  00 00
CX  00 00
DX  00 00
CS  0700
IP  010F
SS  0700
SP  FFFE
BP  0000
SI  300F
DI  1FF1
DS  0700
ES  0700
```

```
0700:010F
07100: BE 190 ╛
07101: 00 000 NULL
07102: 30 048 0
07103: BF 191 ┐
07104: 00 000 NULL
07105: 20 032 SPA
07106: B9 185 ╣
07107: 0F 015 ☼
07108: 00 000 NULL
07109: FC 252 ⁿ
0710A: AC 172 ¼
0710B: FD 253 ²
0710C: AA 170 ¬
0710D: E2 226 Γ
0710E: FA 250 ·
0710F: F4 244 ⌠
07110: C3 195 ├
07111: 90 144 É
07112: 90 144 É
07113: 90 144 É
07114: 90 144 É
```

```
0700:010F
MOV  SI, 03000h
MOV  DI, 02000h
MOV  CX, 0000Fh
CLD
LODSB
STD
STOSB
LOOP 0109h
HLT
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 1
```

```
=== unsigned ===
not below        (CF=0)
    above        (CF=0 and ZF=0)

=== signed ===
not less         (SF=OF)
    greater      (SF=OF and ZF=0)

not equal        (ZF=0)
positive         (SF=0)
no overflow      (OF=0)
parity odd       (PF=0)
not aux carry    (AF=0)
hw-int enabled   (IF=1)
backward dir     (DF=1)
```

```
0700:FFFE  0000 <
0700:FFFC  0000
0700:FFFA  0000
0700:FFF8  0000
0700:FFF6  0000
0700:FFF4  0000
0700:FFF2  0000
0700:FFF0  0000
0700:FFEE  0000
0700:FFEC  0000
0700:FFEA  0000
0700:FFE8  0000
0700:FFE6  0000
0700:FFE4  0000
0700:FFE2  0000
0700:FFE0  0000
0700:FFDE  0000
0700:FFDC  0000
0700:FFDA  0000
0700:FFD8  0000
0700:FFD6  0000
0700:FFD4  0000
0700:FFD2  0000
0700:FFD0  0000
0700:FFCE  0000
0700:FFCC  0000
```

```
;Sharadindu Adhikari, 19BCE2105


;Q1.Repeat the problem with data ranging from

org 100h
MOV SI,3000h
MOV DI,2000h
MOV CX,0Fh
L1: cld
lodsb
std
stosb
loop L1
HLT
ret
```

2. **What are the roles of CLD and STD instructions?**

CLD clears the direction flag. No other flags or registers are affected. After CLD is executed, string operations will increment the index registers (SI and/or DI) that they use.

STD is used to set the direction flag to a 1 so that SI and/or DI will automatically be decremented to point to the next string element when one of the string instructions executes.

If the direction flag is set SI/DI will be decremented by 1 for byte strings and 2 for word strings.