National University of Computer and Emerging Sciences 

**Laboratory Manual**

*for*

**Computer Organization and Assembly Language Programming**

| Course Instructor | Aleena Ahmad |
| --- | --- |
| Lab Instructor | Sana Ejaz |
| Semester | Fall 2024 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**OBJECTIVES:**

∙ Enhance Understanding of Advanced String and Screen Manipulation

∙ Practice String Search, Replace and Compression.

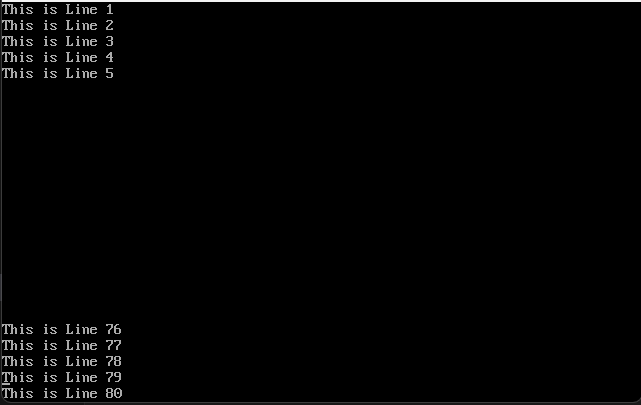
∙ Learn to Trim and Search in Strings.

**Instructions:**

| **1. Submit work in a single Word file with screenshots of meaningful results. 2. Make proper subroutines**  **3. Use Delay Function (if required)** |
| --- |

**Task 1: Write a program that scrolls the Screen 3 rows Up and 3 rows down in an infinite loop without losing any data of screen. (First call printstr to print 5,5 strings in top and bottom 5 rows of screen to test your ScrollUpAndDown functionality.)**

| [org 0x0100]  JMP start  sentences: db 'This is Line 1 ', 0  db 'This is Line 2 ', 0  db 'This is Line 3 ', 0  db 'This is Line 4 ', 0  db 'This is Line 5 ', 0  db 'This is Line 76', 0  db 'This is Line 77', 0  db 'This is Line 78', 0  db 'This is Line 79', 0  db 'This is Line 80', 0  clrscrn:  push es  push ax  push cx  push di  mov ax, 0xb800  mov es, ax  xor di, di  mov ax, 0x0720  mov cx, 2000    cld  rep stosw  pop di  pop cx  pop ax  pop es  ret    printstr: push bp  mov bp, sp  push es  pusha  push ds  pop es  mov di, [bp+4]  mov cx, 0xffff  xor al, al  repne scasb  mov ax, 0xffff  sub ax, cx  dec ax  jz exit  mov cx, ax  mov ax, 0xb800  mov es, ax  mov al, 80  mul byte [bp+8]  add ax, [bp+10]  shl ax, 1  mov di,ax  mov si, [bp+4]  mov ah, [bp+6]  cld  nextchar:  lodsb  stosw  loop nextchar  exit:  popa  pop es  pop bp  ret 8  scrollup:  push bp  mov bp,sp  push ax  push cx  push si  push di  push es  push ds  mov ax, 80 ; load chars per row in ax  mul byte [bp+4] ; calculate source position  mov si, ax ; load source position in si  push si    push ds  pop es  push 0xb800  pop ds  xor si, si  mov di, buffer  mov cx, ax  shl cx, 1  cld  rep movsw  pop si  push si  shl si, 1 ; convert to byte offset  mov cx, 2000 ; number of screen locations  sub cx, ax ; count of words to move  mov ax, 0xb800  mov es, ax ; point es to video base  mov ds, ax ; point ds to video base  xor di, di ; point di to top left column  cld ; set auto increment mode  rep movsw ; scroll up  mov ax, 0x0720  pop cx  rep stosw  pop ds  pop es  pop di  pop si  pop cx  pop ax  pop bp  ret 2  scrolldown:  push bp  mov bp,sp  push ax  push cx  push si  push di  push es  push ds  mov ax, 80 ; load chars per row in ax  mul byte [bp+4] ; calculate source position  push ax ; save position for later use  shl ax, 1 ; convert to byte offset  mov si, 3998 ; last location on the screen  sub si, ax ; load source position in si  mov cx, 2000 ; number of screen locations  shr ax, 1  sub cx, ax ; count of words to move  mov ax, 0xb800  mov es, ax ; point es to video base  mov ds, ax ; point ds to video base  mov di, 3998 ; point di to lower right column  std ; set auto decrement mode  rep movsw ; scroll up  cld  mov si, buffer  xor di, di  push cs  pop ds  pop cx ; count of positions to clear  rep movsw ; clear the scrolled space  pop ds  pop es  pop di  pop si  pop cx  pop ax  pop bp  ret 2  delay:  PUSH CX  MOV CX, 0xFFFF  delay\_loop:  LOOP delay\_loop  POP CX  RET  start:  CALL clrscrn  MOV CX, 0  MOV SI, sentences  printtop:  PUSH word 0  PUSH word CX  PUSH word 0x7  PUSH word SI  CALL printstr  ADD SI, 16  INC CX  CMP CX, 5  JNZ printtop  MOV CX, 20  printbottom:  PUSH word 0  PUSH word CX  PUSH word 0x7  PUSH word SI  CALL printstr  ADD SI, 16  INC CX  CMP CX, 25  JNZ printbottom  loplop:  PUSH word 3  CALL scrollup  CALL delay  CALL delay  CALL delay  CALL delay  CALL delay  PUSH word 3  CALL scrolldown  CALL delay  CALL delay  CALL delay  CALL delay  CALL delay  JMP loplop  buffer: times 5 \* 160 db 0 |
| --- |



**Task 2: [SCAS] Write a program that takes a c-string *myStr* and two characters *charToFind* and *charToReplace* from user and replaces all the occurrences of *charToFind* with *charToReplace* in *myStr*. Your program should create a space of 50 characters on heap in order to save *myStr*. Sample output:**

| [org 0x0100]  JMP start  InputString: db 'ddsdfhgrtsdfhjghjksdd', 0  CharToFind: db 'd'  CharToReplace: db '$'  ModifiedString: times 50 db 0  strrepl:  PUSH BP  MOV BP, SP    PUSH ES  PUSHA  PUSH DS  POP ES  XOR AX, AX  MOV CX, 0xFFFF  MOV DI, [BP + 8]  REPNE SCASB    MOV AX, 0xFFFF  SUB AX, CX  MOV CX, AX  PUSH CX  MOV SI, [BP + 8]  MOV DI, [BP + 10]  REP MOVSB  POP CX  MOV AL, [BP + 6]  MOV AH, [BP + 4]  MOV DI, [BP + 10]    next\_char:  REPNE SCASB    JCXZ end\_string    MOV [ES:DI - 1], AH  JMP next\_char  end\_string:  POPA  POP ES  POP BP  RET 8  clrscrn:  push es  push ax  push cx  push di  mov ax, 0xb800  mov es, ax  xor di, di  mov ax, 0x0720  mov cx, 2000    cld  rep stosw  pop di  pop cx  pop ax  pop es  ret    printstr: push bp  mov bp, sp  push es  pusha  push ds  pop es ; load ds in es  mov di, [bp+4] ; point di to string  mov cx, 0xffff ; load maximum number in cx  xor al, al ; load a zero in al  repne scasb ; find zero in the string  mov ax, 0xffff ; load maximum number in ax  sub ax, cx ; find change in cx  dec ax ; exclude null from length  jz exit ; no printing if string is empty  mov cx, ax ; load string length in cx  mov ax, 0xb800  mov es, ax ; point es to video base  mov al, 80 ; load al with columns per row  mul byte [bp+8] ; multiply with y position  add ax, [bp+10] ; add x position  shl ax, 1 ; turn into byte offset  mov di,ax ; point di to required location  mov si, [bp+4] ; point si to string  mov ah, [bp+6] ; load attribute in ah  cld ; auto increment mode  nextchar: lodsb ; load next char in al  stosw ; print char/attribute pair  loop nextchar ; repeat for the whole string  exit:  popa  pop es  pop bp  ret 8  start:  XOR AX, AX  PUSH word ModifiedString  PUSH word InputString  MOV AL, [CharToFind]  PUSH AX  MOV AL, [CharToReplace]  PUSH AX  CALL strrepl    CALL clrscrn    PUSH word 0  PUSH word 0  PUSH word 0x07  PUSH InputString  CALL printstr    PUSH word 0  PUSH word 1  PUSH word 0x07  PUSH ModifiedString  CALL printstr  MOV AX, 0  INT 0x16  MOV AX, 0x4C00  INT 0x21 |
| --- |



**Task 3:**

**A) TrimStart(char\* str)**

**Write a function that takes a string and removes all the space in start of the string. Sample Output:**

Page 2

| **Before TrimStart**  **str:** “ Hello How are you?”  **After TrimStart**  **str:** “Hello How are you?” |
| --- |

**B) Write a function that searches a substring from a string and highlights the found substring. If the string is not found it will not highlight anything.**

**Sample Run:**

| **String:** “I am a student of COAL”  **Substring:** “student”  **Printed String after Search:** “I am a student of COAL” |
| --- |

**C) String Compression**

**Write a function that compresses a string by removing consecutive occurrences of same character.**

**Sample Run:**

| **String Before Compression:**  **Str:** “ggggdddddddyyyyakxxxuww”  **String after Compression:**  **Str:** “gdyakxuw” |
| --- |

| [org 0x0100]  JMP start  trimString: db ' Hello How are you?', 0  highString: db 'I am a student of COAL', 0  subString: db 'student', 0  compString: db 'ggggdddddddyyyyakxxxuww', 0  TrimStart:  PUSH BP  MOV BP, SP    PUSH ES  PUSHA  PUSH DS  POP ES  XOR AX, AX  MOV CX, 0xFFFF  MOV DI, [BP + 4]  REPNE SCASB  MOV AX, 0xFFFF  SUB AX, CX  MOV CX, AX  DEC CX  JZ term\_trim  INC CX  MOV AL, 0x20  MOV DI, [BP + 4]  REPE SCASB  MOV SI, DI  DEC SI  MOV DI, [BP + 4]  REP MOVSB  MOV byte [ES:DI], 0    term\_trim:  POPA  POP ES  POP BP  RET 2  HighStr:  PUSH BP  MOV BP, SP  PUSH ES  PUSHA  PUSH DS  POP ES  XOR AX, AX  MOV CX, 0xFFFF  MOV DI, [BP + 4]  REPNE SCASB  MOV AX, 0xFFFF  SUB AX, CX  MOV CX, AX  DEC CX  JZ term\_high  INC CX  PUSH CX  XOR AX, AX  MOV CX, 0xFFFF  MOV DI, [BP + 6]  REPNE SCASB  MOV BX, 0xFFFF  SUB BX, CX  POP CX  SUB BX, CX  JL term\_high  PUSH word [BP + 12]  PUSH word [BP + 10]  PUSH word [BP + 8]  PUSH word [BP + 6]  CALL printstr  DEC BX  MOV SI, [BP + 6]  MOV DI, [BP + 4]  MOV AX, SI  PUSH SI  PUSH CX  rep\_compare:  REPE CMPSB  JCXZ found  POP CX  POP AX  PUSH SI  PUSH CX  MOV DI, [BP + 4]  DEC BX  JNZ rep\_compare  term\_high:  POP CX  POP SI  POPA  POP ES  POP BP    RET 4  found:  POP CX  POP SI  PUSH SI  PUSH CX  PUSH word 0xB800  POP ES  DEC CX  mov al, 80  mul byte [bp+10]  add ax, [bp+12]  sub si, [BP + 6]  add si, ax  shl si, 1    highlight:  MOV byte [ES:SI + 1], 0x40  ADD SI, 2  LOOP highlight  JMP term\_high  CompressStr:  PUSH BP  MOV BP, SP    PUSH ES  PUSHA  PUSH DS  POP ES  XOR AX, AX  MOV CX, 0xFFFF  MOV DI, [BP + 4]  REPNE SCASB  MOV AX, 0xFFFF  SUB AX, CX  MOV CX, AX  DEC CX  JZ term\_compr  MOV SI, [BP + 4]  MOV DI, [BP + 4]  compress\_str:  INC CX  MOV AL, [DI]  REPE SCASB  MOV [SI], AL  INC SI  DEC DI  CMP CX, 0  JNZ compress\_str  MOV byte [SI], 0  term\_compr:  POPA  POP ES  POP BP  RET 2  clrscrn:  push es  push ax  push cx  push di  mov ax, 0xb800  mov es, ax  xor di, di  mov ax, 0x0720  mov cx, 2000    cld  rep stosw  pop di  pop cx  pop ax  pop es  ret    printstr: push bp  mov bp, sp  push es  pusha  push ds  pop es ; load ds in es  mov di, [bp+4] ; point di to string  mov cx, 0xffff ; load maximum number in cx  xor al, al ; load a zero in al  repne scasb ; find zero in the string  mov ax, 0xffff ; load maximum number in ax  sub ax, cx ; find change in cx  dec ax ; exclude null from length  jz exit ; no printing if string is empty  mov cx, ax ; load string length in cx  mov ax, 0xb800  mov es, ax ; point es to video base  mov al, 80 ; load al with columns per row  mul byte [bp+8] ; multiply with y position  add ax, [bp+10] ; add x position  shl ax, 1 ; turn into byte offset  mov di,ax ; point di to required location  mov si, [bp+4] ; point si to string  mov ah, [bp+6] ; load attribute in ah  cld ; auto increment mode  nextchar: lodsb ; load next char in al  stosw ; print char/attribute pair  loop nextchar ; repeat for the whole string  exit:  popa  pop es  pop bp  ret 8  start:  CALL clrscrn    PUSH word 0  PUSH word 0  PUSH word 0x07  PUSH word trimString  CALL printstr  PUSH word trimString  CALL TrimStart  PUSH word 0  PUSH word 1  PUSH word 0x07  PUSH word trimString  CALL printstr  PUSH word 0  PUSH word 3  PUSH word 0x07  PUSH word highString  CALL printstr  PUSH word 0  PUSH word 4  PUSH word 0x07  PUSH word subString  CALL printstr  PUSH word 0  PUSH word 5  PUSH word 0x07  PUSH word highString  PUSH word subString  CALL HighStr  PUSH word 0  PUSH word 7  PUSH word 0x07  PUSH word compString  CALL printstr  PUSH word compString  CALL CompressStr  PUSH word 0  PUSH word 8  PUSH word 0x07  PUSH word compString  CALL printstr  MOV AX, 0  INT 0x16  MOV AX, 0x4C00  INT 0x21 |
| --- |

