Modern College of Engineering
Shivajinagar, Pune 5.

MP lab Assignment 8

Mahesh Jagtap
SE-A 21027 (S2)

Problem statement:

Write X86 Assembly language Program (ALP) to implement following OS Commands.
(i) COPY ii) TYPE using file operations.
User is supposed to provide command line arguments.

objective:

To learn about various aspects of GDTR, LDTR TR & MSW

Software/Hardware:

(1) OS : 64 bit open source linux.
(2) Assembler : NASM
(3) editor : geditor

Theory:

An OS is the software that controls 'O' Computers hardware & pheripheral devices & allows other programs to function. Early Computers did not had a hard disk drivers. but we hand wired to carry out specific computation.

MS-DOS kernel:
- Process Control
The kernel is a prospreitary program & provides a collection of hardcore Independent Services called system functions.

The functions include the following:
- file & record management.
- Memory management
- character, devices, input & output
- Spanning of the program.
- Access to the real time clock.

② The Command processor.

The command processor or shell is the users interface to the OS. It is responsible for passing (describing) & carrying out user commands, including the loading & execution of the other programming from a disk or other programming from a disk or other mass storage.

The default shell that is provided with MS-DOS is found in a file called COMMAND-Com It is aspected class of program running under the control of MS-DOS.

21027

User Commands:

The user commands that are accepted by COMMAND.COM fall into 3 categories:

Internal Commands
(a carried out by code embedded in COMMAND.COM)

External Commands
(names of program stored on disk file)

Batch files.
(batch of group of DOS commands)

i) TYPE Command:
Displays the contents of a text file in the terminal.

Algorithm for TYPE:

1) Start
2) Get the source file name from command line.
3) If the file is not present, display "error msg as file not found" & stop.
4) If present, open file in read mode.
5) Read the contents of file & display data on the screen.
6) stop.

ii) Copy :
It Copies one or more files to another location.

Algorithm :
1) start
2) Get source & destination file name from Command.
3) If file is not present, display error msg & stop
4) If present, open file in read mode.
5) Read name of destination file & open it in read mode.
6) Read the contents of file source & write it into destination file.
7) STOP.

Conclusion :

In this practical we have studied & Implemented COPY & TYPE OS commands.

```
%macro cmn 4            ;input/output
    mov rax,%1
    mov rdi,%2
    mov rsi,%3
    mov rdx,%4
    syscall
%endmacro
%macro exit 0
    mov rax,60
    mov rdi,0
    syscall
%endmacro

%macro fopen 1
    mov    rax,2        ;open
    mov    rdi,%1    ;filename
    mov    rsi,2        ;mode RW
    mov    rdx,0777o    ;File permissions
    syscall
%endmacro

%macro fread 3
    mov    rax,0        ;read
    mov    rdi,%1    ;filehandle
    mov    rsi,%2    ;buf
    mov    rdx,%3    ;buf_len
    syscall
%endmacro

%macro fwrite 3
    mov    rax,1        ;write/print
    mov    rdi,%1    ;filehandle
    mov    rsi,%2    ;buf
    mov    rdx,%3    ;buf_len
    syscall
%endmacro

%macro fclose 1
    mov    rax,3        ;close
    mov    rdi,%1    ;file handle
    syscall
%endmacro

section .data
    menu db 'MENU : ',0Ah
        db "1. TYPE",0Ah
        db "2. COPY",0Ah
        db "3. DELETE",0Ah
        db "4. Exit",0Ah
        db "Enter your choice : "
    menulen equ $-menu
    msg db "Command : "
    msglen equ $-msg
    cpysc db "File copied successfully !!",0Ah
    cpysclen equ $-cpysc
    delsc db 'File deleted successfully !!',0Ah
    delsclen equ $-delsc
    err db "Error ...",0Ah
    errlen equ $-err
    cpywr db 'Command does not exist',0Ah
    cpywrlen equ $-cpywr
    err_par db 'Insufficient parameter',0Ah
    err_parlen equ $-err_par


section .bss
    choice resb 2
    buffer resb 50
```

```
    name1 resb 15
    name2 resb 15
    cmdlen resb 1
    filehandle1 resq 1
    filehandle2 resq 1

    abuf_len        resq    1        ; actual buffer length
    dispnum resb 2

    buf resb    4096
    buf_len equ $-buf        ; buffer initial length

section .text
global _start
_start:

again:    cmn 1,1,menu,menulen
    cmn 0,0,choice,2

    mov al,byte[choice]
    cmp al,31h
    jbe op1
    cmp al,32h
    jbe op2
    cmp al,33h
    jbe op3

        exit
        ret

op1:
    call tproc
    jmp again

op2:
    call cpproc
    jmp again


op3:
    call delproc
    jmp again



;type command procedure
tproc:
    cmn 1,1,msg,msglen
    cmn 0,0,buffer,50
    mov byte[cmdlen],al
    dec byte[cmdlen]

    mov rsi,buffer
    mov al,[rsi]            ;search for correct type command
    cmp al,'t'
    jne skipt
    inc rsi
    dec byte[cmdlen]
    jz skipt
    mov al,[rsi]
    cmp al,'y'
    jne skipt
    inc rsi
    dec byte[cmdlen]
    jz skipt
    mov al,[rsi]
    cmp al,'p'
    jne skipt
    inc rsi
    dec byte[cmdlen]
```

```
    jz skipt
    mov al,[rsi]
    cmp al,'e'
    jne skipt
    inc rsi
    dec byte[cmdlen]
    jnz correctt
    cmn 1,1,err_par,err_parlen
    call exit

skipt:    cmn 1,1,cpywr,cpywrlen
    exit
correctt:
    mov rdi,name1             ;finding file name
    call find_name

    fopen name1             ; on succes returns handle
    cmp rax,-1H             ; on failure returns -1
    jle error
    mov [filehandle1],rax

    xor rax,rax
    fread [filehandle1],buf, buf_len
    mov [abuf_len],rax
    dec byte[abuf_len]

    cmn 1,1,buf,abuf_len        ;printing file content on screen

ret


;copy command procedure
cpproc:
    cmn 1,1,msg,msglen
    cmn 0,0,buffer,50         ;accept command
    mov byte[cmdlen],al
    dec byte[cmdlen]

    mov rsi,buffer
    mov al,[rsi]             ;search for copy
    cmp al,'c'
    jne skip
    inc rsi
    dec byte[cmdlen]
    jz skip
    mov al,[rsi]
    cmp al,'o'
    jne skip
    inc rsi
    dec byte[cmdlen]
    jz skip
    mov al,[rsi]
    cmp al,'p'
    jne skip
    inc rsi
    dec byte[cmdlen]
    jz skip
    mov al,[rsi]
    cmp al,'y'
    jne skip
    inc rsi
    dec byte[cmdlen]
    jnz correct
    cmn 1,1,err_par,err_parlen
    exit

skip:    cmn 1,1,cpywr,cpywrlen
    exit
correct:
    mov rdi,name1             ;finding first file name
```

```
    call find_name

    mov rdi,name2            ;finding second file name
    call find_name

skip3:    fopen name1            ; on succes returns handle
    cmp rax,-1H            ; on failure returns -1
    jle error
    mov [filehandle1],rax

    fopen name2            ; on succes returns handle
    cmp rax,-1H            ; on failure returns -1
    jle error
    mov [filehandle2],rax

    xor rax,rax
    fread [filehandle1],buf, buf_len
    mov [abuf_len],rax
    dec byte[abuf_len]

    fwrite [filehandle2],buf, [abuf_len]        ;write to file

    fclose    [filehandle1]
    fclose    [filehandle2]
    cmn 1,1,cpysc,cpysclen

    jmp again
error:
    cmn 1,1,err,errlen
    exit
ret



;delete command procedure
delproc:

    cmn 1,1,msg,msglen
    cmn 0,0,buffer,50        ;accept command
    mov byte[cmdlen],al
    dec byte[cmdlen]

    mov rsi,buffer
    mov al,[rsi]            ;search for copy
    cmp al,'d'
    jne skipr
    inc rsi
    dec byte[cmdlen]
    jz skipr
    mov al,[rsi]
    cmp al,'e'
    jne skipr
    inc rsi
    dec byte[cmdlen]
    jz skipr
    mov al,[rsi]
    cmp al,'l'
    jne skipr
    inc rsi
    dec byte[cmdlen]
    jnz correctr
    cmn 1,1,err_par,err_parlen
    exit

skipr:    cmn 1,1,cpywr,cpywrlen
    exit

correctr:
    mov rdi,name1            ;finding first file name
    call find_name
```

```
    mov rax,87              ;unlink system call
    mov rdi,name1
    syscall

    cmp rax,-1H              ; on failure returns -1
    jle errord
    cmn 1,1,delsc,delsclen
    jmp again

errord:
    cmn 1,1,err,errlen
    exit

ret


find_name:                  ;finding file name from command
    inc rsi
    dec byte[cmdlen]
cont1:    mov al,[rsi]
    mov [rdi],al
    inc rdi
    inc rsi
    mov al,[rsi]
    cmp al,20h              ;searching for space
    je skip2
    cmp al,0Ah              ;searching for enter key
    je skip2
    dec byte[cmdlen]
    jnz cont1
    cmn 1,1,err,errlen
    exit

skip2:
ret
```