NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(KARACHI CAMPUS)
FAST School of Computing
Fall 2023

PROJECT: Text File Analyzer

Members:

- Ayesha Ansari (22K-4453)
- Jahantaab Kulsoom (22k-4214)

Introduction:

In the realm of computer programming, assembly language serves as a low-level programming language that is closely tied to the architecture of a computer's central processing unit (CPU). The Text File Analyzer project aims to leverage assembly language to create a tool capable of analyzing and processing text files efficiently. Assembly language provides a direct interface with the hardware, offering a unique opportunity to optimize performance in resource-intensive tasks.

Problem Statement:

Text file analysis involves numerous operations such as counting characters, words, and lines, searching for specific patterns, and extracting relevant information. Traditional high-level languages may not provide the level of control and optimization required for these tasks. The challenge lies in developing a text file analyzer that is not only precise and fast but also resource-efficient.

Motivation:

The motivation behind this project stems from the need for a specialized tool to handle large text files with optimal performance. High-level languages may introduce unnecessary overhead, hindering the speed and efficiency required for processing extensive datasets. Assembly language, with its direct interaction with the hardware, offers a promising solution to overcome these challenges.

Proposed Solution:

The Text File Analyzer will be implemented in assembly language, specifically designed to harness the capabilities of the underlying hardware. The solution will include modules for character counting, word counting, line counting, pattern matching, and other text-related operations. The goal is to achieve a balance between precision and speed, catering to scenarios where quick and efficient text file analysis is paramount.

Methodology:

1. **Source Code:**

```
;Project Title :  Text File Analyzer
```

```
;Group Members: 1. Jahantaab Kulsoom
;               2. Ayesha Ansari
;Features:
;                  1. Create New File
;                 2. Get Word Count
;                 3. Word Search
;                 4. Delete Word
;                 5.Replace Word


INCLUDE Irvine32.inc
.data
buffer byte 500 dup(?)
filename byte "output.txt",0
filehandle handle ?
num dword ?
temp byte 30 dup (?)
tempstr byte 500 dup(?)
newstr byte 30 dup(?)
newfilemsg byte "Press 5 to create new file",0
wordcountmsg byte "Press 1 to get word count",0
searchwordmsg byte "Press 2 to search word ",0
deletewordmsg byte "Press 3 to delete word",0
replacewordmsg byte "Press 4 to replace word",0
exitmsg byte "Press 0 to exit",0
line byte "----------------------------------------",0
frequencymsg byte "Frequency: ",0
prompt byte "Enter: ",0
msg1 byte "Word count: ",0
getword byte "Enter word: ",0
getreplaceword byte "Enter new word: ",0
notfoundmsg byte "Word not found",0
.code
main proc
menu_loop:
        call crlf
        mov edx, offset line
        call writestring
        call crlf
        mov eax,black + (gray * 16) ; white on gray
        call SetTextColor
        mov edx, offset exitmsg
    call writestring
    call crlf
    mov edx, offset wordcountmsg
    call writestring
    call crlf
    mov edx, offset searchwordmsg
    call writestring
    call crlf
    mov edx, offset deletewordmsg
    call writestring
    call crlf
    mov edx, offset replacewordmsg
    call writestring
    call crlf
    mov edx, offset newfilemsg
```

```
    call writestring
    call crlf


    mov edx, offset prompt
    call writestring
    call readint

    cmp eax, 0 ; Assuming 0 is the exit option chosen by the user
    je exit_menu
    cmp eax, 1
    je wordcount
    cmp eax, 2
    je _searchword
    cmp eax, 3
    je _deleteword
    cmp eax, 4
    je _replaceword
    cmp eax, 5
    je _newfile
    jmp menu_loop

wordcount:
    call read_file
    call countwords
    jmp menu_loop

_searchword:
    call read_file
    call searchword
    jmp menu_loop

_deleteword:
    call read_file
    call deleteword
    call updatefile
    jmp menu_loop

_replaceword:
    call read_file
    call replaceword
    call updatefile
    jmp menu_loop

_newfile:
    call newfile
    jmp menu_loop

exit_menu:
    ; Clean-up code before exiting the program goes here
    ret

main endp
exit
updatefile proc
       mov edx,offset filename
       call createoutputfile
```

```
        mov filehandle,eax
        mov edx,offset buffer
        mov ecx,490
        call writetofile
        mov eax,filehandle
        call closefile
        ret
updatefile endp
replaceword proc
        push ebp
        mov ebp,esp
        sub esp,20
        mov edx,offset getreplaceword
        call writestring
        mov edx,offset newstr
        mov ecx,300
        call readstring
        mov [ebp-20],eax
        ;local start:dword,end:dword,base:dword,last:dword
        call searchword
        cmp eax,0
        je notfound
        mov ebx,offset buffer
        mov ecx,num
        L2:
                inc ebx
        loop L2
        mov [ebp-16],ebx
        mov ecx,eax
        mov [ebp-8],offset buffer
        L1:
                push ecx
                push [ebp-8]
                call findindex
                mov [ebp-4],esi
                mov [ebp-8],edi
                mov [ebp-12],offset buffer
                mov edi,offset tempstr
                cmp esi,offset buffer
                je middle
                dec esi
                mov eax,[ebp-4]
                sub eax,[ebp-12]
                mov[ebp-4],eax
                mov ecx,[ebp-4]
                mov esi,offset buffer
                rep movsb
                middle:
                mov esi,offset newstr
                mov ecx,[ebp-20]
                rep movsb
                mov eax,[ebp-8]
                cmp eax,[ebp-16]
                je done
                mov eax,[ebp-16]
                sub eax,[ebp-8]
                mov ecx,eax
```

```
                mov esi,[ebp-8]
                rep movsb
                mov edx,offset tempstr
                mov edx,lengthof tempstr
                mov num,edx
                mov esi,offset tempstr
                mov edi,offset buffer
                mov ecx,edx
                rep movsb
                pop ecx
        loop L1
        done:
        mov edx,offset buffer
        call writestring
        jmp quit
        notfound:
                mov edx,offset notfoundmsg
                call writestring
        quit:
                mov esp,ebp
                pop ebp
                ret
replaceword endp
deleteword proc
        push ebp
        mov ebp,esp
        sub esp,16
        ;local start:dword,end:dword,base:dword,last:dword
        call searchword
        cmp eax,0
        je notfound
        mov ebx,offset buffer
        mov ecx,num
        L2:
                inc ebx
        loop L2
        mov [ebp-16],ebx
        mov ecx,eax
        mov [ebp-8],offset buffer
        L1:
                push ecx
                push [ebp-8]
                call findindex
                mov [ebp-4],esi
                mov [ebp-8],edi
                mov [ebp-12],offset buffer
                mov edi,offset tempstr
                cmp esi,offset buffer
                je after
                dec esi
                mov eax,[ebp-4]
                sub eax,[ebp-12]
                mov[ebp-4],eax
                mov ecx,[ebp-4]
                mov esi,offset buffer
                rep movsb
                after:
```

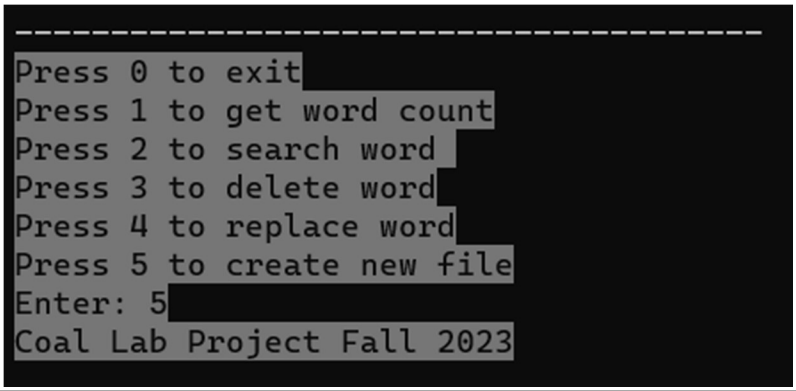
```
                mov eax,[ebp-8]
                cmp eax,[ebp-16]
                je done
                mov eax,[ebp-16]
                sub eax,[ebp-8]
                mov ecx,eax
                mov esi,[ebp-8]
                inc esi
                rep movsb
                mov edx,offset tempstr
                mov edx,lengthof tempstr
                mov num,edx
                mov esi,offset tempstr
                mov edi,offset buffer
                mov ecx,edx
                rep movsb
                pop ecx
        loop L1
        done:
        mov edx,offset buffer
        call writestring
        jmp quit
        notfound:
                mov edx,offset notfoundmsg
                call writestring
        quit:
                mov esp,ebp
                pop ebp
                ret
deleteword endp
findindex proc
        push ebp
        mov ebp,esp
        mov edi,offset buffer
        mov dword ptr[ebp-16],ecx
        mov ecx,num
        L1:
                inc edi
        loop L1
        sub esp,16
        mov [ebp-4],edi
        mov edi,[ebp+8]
        match_first_character:
                mov al,temp[0]
                mov ecx,[ebp-4]
                sub ecx,edi
                cld
                repne scasb
                jnz quit
                mov [ebp-12],edi
                dec dword ptr[ebp-12]
                call compare_substring
                jz found
                cmp byte ptr[edi],0
                jne match_first_character
                jmp quit
                found:
```

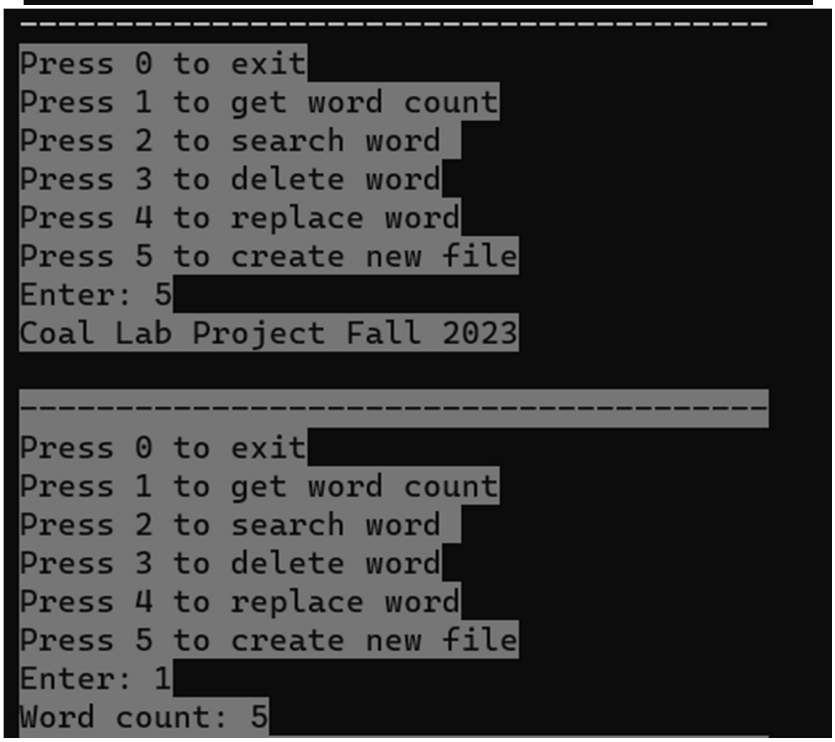
```
                        mov esi,[ebp-12]
                        mov edi,eax
        quit:
                mov ecx,[ebp-16]
                mov esp,ebp
                pop ebp
                ret 4
findindex endp
searchword proc
        push ebp
        mov ebp,esp
        mov edx,offset getword
        call writestring
        mov edx,offset temp
        mov ecx,20
        call readstring
        mov edi,offset buffer
        mov ecx,num
        L1:
                inc edi
        loop L1
        sub esp,12
        mov dword ptr[ebp-12],0
        mov [ebp-4],edi
        mov edi,offset buffer
        match_first_character:
                mov al,temp[0]
                mov ecx,[ebp-4]
                sub ecx,edi
                cld
                repne scasb
                jnz quit
                call compare_substring
                jz found
                cmp byte ptr[edi],0
                jne match_first_character
                jmp quit
                found:
                        add dword ptr[ebp-12],1
                        jmp match_first_character
        quit:
                mov edx,offset frequencymsg
                call writestring
                mov eax,[ebp-12]
                call writedec
                call crlf
                mov esp,ebp
                pop ebp
                ret
searchword endp
compare_substring proc
        push edi
        mov esi,offset temp
        inc esi
        L1:
                mov al,[esi]
                mov dl,[edi]
```

```
                cmp al,0
                jne L2
                cmp dl,32
                je L3
                cmp dl,0
                jmp L3
                L2:
                        inc esi
                        inc edi
                        cmp al,dl
                        je L1
                L3:
                        mov eax,edi
                        pop edi
                        ret
compare_substring endp
countwords proc
        push ebp
        mov ebp,esp
        sub esp,4
        mov dword ptr[ebp-4],1
        mov ecx,num
        mov esi,1
        mov eax,0
        L1:
                cmp buffer[esi],32
                jne L2
                add dword ptr [ebp-4],1
                L2:
                        add esi,1
        loop L1
        mov edx,offset msg1
        call writestring
        mov eax,[ebp-4]
        call writedec
        mov esp,ebp
        pop ebp
        ret
countwords endp
read_file proc
        mov edx,offset filename
        call openinputfile
        mov filehandle,eax
        mov edx,offset buffer
        mov ecx,500
        mov eax,filehandle
        call readfromfile
        jnc valid_file
        jmp quit
        valid_file:
                mov num,eax
                mov buffer[eax],0
                mov edx,offset buffer
                call writestring
                call crlf
                mov eax,filehandle
                call closefile
```

```
        quit:
                ret
read_file endp
newfile proc
        mov edx,offset filename
        call createoutputfile
        mov filehandle,eax
        mov ecx,500
        mov edx,offset buffer
        call readstring
        mov num,eax
        mov eax,filehandle
        mov ecx,num
        call writetofile
        call closefile
        ret
newfile endp
END main
```

2. **Results:**





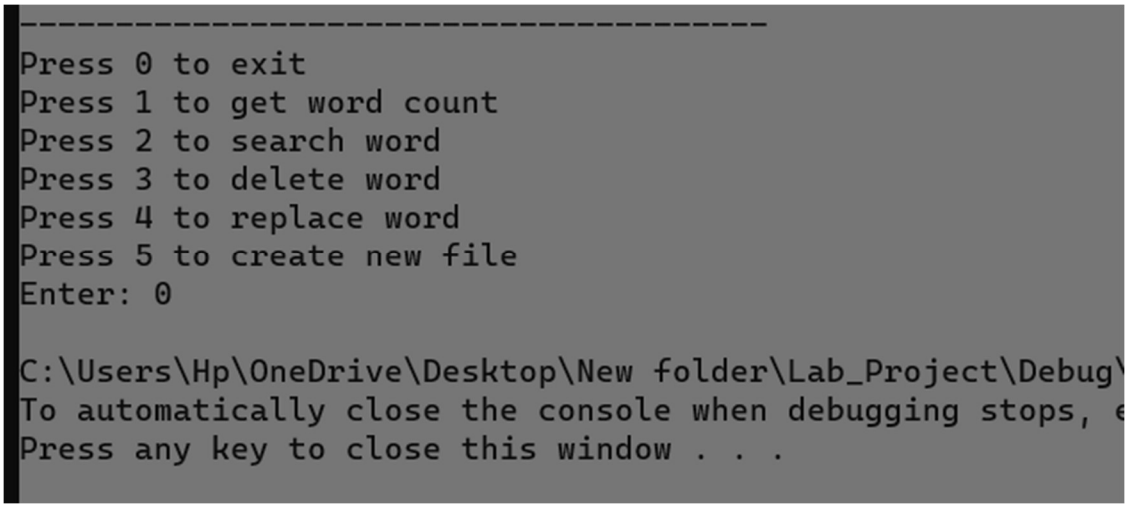
```
--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 5
Coal Lab Project Fall 2023

--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 1
Word count: 5
--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 2
Enter word: Lab
Frequency: 1
```

```
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 3
Coal Lab Project Fall 2023
Enter word: Lab
Frequency: 1
Coal Project Fall 2023
--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 4
Coal Project Fall 2023
Enter new word: Lab
Enter word: Coal
Frequency: 1
Lab Project Fall 2023
--------------------------------------
```

```
--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 5
New Coal Lab Project

--------------------------------------
Press 0 to exit
Press 1 to get word count
Press 2 to search word
Press 3 to delete word
Press 4 to replace word
Press 5 to create new file
Enter: 1
Word count: 4
--------------------------------------
```



Benefits:

Pros:

1. **Optimized Performance:** Assembly language allows for fine-grained control over hardware resources, leading to optimized performance in text file analysis.
2. **Low-Level Access:** Direct interaction with CPU instructions provides unparalleled control, enhancing the efficiency of the analyzer.
3. **Resource Efficiency:** The project aims to minimize resource consumption while maximizing output, making it suitable for large-scale text file processing.

Cons:

1. **Steep Learning Curve:** Assembly language may pose a learning curve for developers unfamiliar with low-level programming.
2. **Platform Dependency:** Code may need adjustments for different architectures, limiting portability.

Conclusion:

The Text File Analyzer project in assembly language addresses the critical need for a high-performance tool in the domain of text file analysis. By leveraging the advantages of assembly language, the project aims to provide a solution that balances efficiency, precision, and resource optimization. Despite potential challenges, the benefits of using assembly language for this project outweigh the drawbacks, making it a compelling approach for those seeking unparalleled performance in text file analysis.

References:

Assembly Language for X86 Processors by Kip R. Irvine