CS322 – Computer Arcitechture Lab 5 Report

Name : Chandrawanshi Mangesh Shivaji
Roll No. : 1801CS16
Date : 14/10/2020

Introduction : I have created an interactive program using 32-bit nasm assembly which will sort the given non-zero input integers in non-decreasing order using the **GNOME SORT ALGORITHM**.

GNOME SORT ALGORITHM ?

Input – Array- arr[], Total elements - n

Algorithm Steps

1. If you are at the start of the array then go to the right element (from arr[0] to arr[1]).

2. If the current array element is larger or equal to the previous array element then go one step right

```
if (arr[i] >= arr[i-1])
  i++;
```

3. If the current array element is smaller than the previous array element then swap these two elements and go one step backwards

```
if (arr[i] < arr[i-1])
{
   swap(arr[i], arr[i-1]);
   i--;
}
```

4. Repeat steps 2 and 3 till ‘i’ reaches the end of the array (i.e i = ‘n-1’)

5. If the end of the array is reached then stop and the array is sorted.

Output - Sorted Array of n elements

Corresponding C++ Code for gnome sort algorithm :

file1.cpp help lab5.asm

```cpp

// GNOME SORT in C++

#include <bits/stdc++.h>
using namespace std;

void GNOMESORT(int arr[],int n)
{
    int index = 0;

    while(index < n)
    {
        if(index == 0)
            index++;

        if(arr[index] >= arr[index-1])
        {
            index++;
        }
        else
        {
            int tmp = arr[index];
            arr[index] = arr[index-1];
            arr[index-1] = tmp;
            index--;
        }
    }
    return;
}

int main()
{
    cout << "Enter the size of the array : ";
    int n;
    cin >> n;

    cout << "Enter array elements : \n";
    int arr[n];
    for(int i=0;i<n;i++)
    {
        cin >> arr[i];
    }

    GNOMESORT(arr,n);

    cout << "Sorted array is : \n";

    for(int i=0;i<n;i++)
    {
        cout << arr[i] << " ";
    }

    cout << "\n";

    return 0;
}

```

32-bit Assembly Code for nasm :

1. Initialization of variables and including necessary files

```
;filename   lab5.asm
; Objective: To implement the GNOME sort algorithm for sorting given list of 32-bit integers (other than 0)
;           in non-decreasing order.
;     Input: Size of the array followed by list of integers present in the array
;     Output: Sorted list of given integers


MAX_SIZE   EQU   1000

%include "io.mac"

.DATA
prompt_msg   db  "Enter elements of the array ( Enter zero to terminate the input. ) : ",0
output_msg   db  "Sorted List of given integers : ",0
end_msg   db  "Terminate the program ? (Y/N) ",0

.UDATA
array       resw  MAX_SIZE     ; input array for integers

```

In this part, strings which are used for printing various messages during program execution are declared in data segment. ‘io.mac’ file is included to facilitate input and output of integers,strings and characters. It also contains some macro declarations in it. Max possible size for the array is also declared.

2. Prompt User for Input and Call Gnomesort Function

```

.CODE
    global    _start
_start:
        PutStr prompt_msg     ; request input numbers
        nwln
        mov     EBX,array       ; EBX := array pointer
        mov     ECX,MAX_SIZE     ; CX := array size
        sub     EDX,EDX           ; number count := 0

read_loop:
        GetLInt EAX             ; read input number
        cmp     EAX,0           ; if the number is zero
        je      stop_reading    ; no more numbers to read
        mov     [EBX],EAX       ; copy the number into array
        add     EBX,4           ; EBX points to the next element
        inc     EDX              ; increment number count
        loop    read_loop       ; reads a max. of MAX_SIZE numbers

stop_reading:
        cmp     EDX,0           ; Check if count of total input numbers is zero
        je    end_iteration     ; if yes, end this iteration

        push    EDX             ; push array size onto stack
        push    array           ; place array pointer on stack

        call GnomeSort
```

This part of the code first outputs a string msg prompting user for input of numbers in the array. It counts the size of array and stops taking input if 0 is given as input. It pushes size of array and array pointer on the stack and then calls the sorting function.

3.Gnomesort Function

```
64
65  .CODE
66  GnomeSort:
67      pushad                  ;store registers on stack for later restoration
68      mov EBP,ESP             ;move stack pointer to base pointer
69
70      ;EBP + 36   is the array
71      ;EBP + 40   is the number of integers in the array
72      mov ECX, [EBP+40]       ;setting ECX to the number of integers
73      mov ESI, [EBP+36]       ;ESI is the array
74      xor EAX, EAX            ;Setting EAX to 0, it'll be our iterator(i)
75
76      MainLoop:
77          cmp EAX, ECX        ;If 'i' >= the number of integers, exit the loop
78          jge EndLoop
79
80          cmp EAX, 0          ;If 'i' == 0, move to the next element
81          je IncreaseCounter
82
83          mov EBX, [ESI]      ;EBX = array[i]
84          mov EDX, [ESI-4]    ;EDX = array[i-1]
85          cmp EDX, EBX        ;If array[i-1] <= array[i], it means they are
86          jle IncreaseCounter ;sorted, so move to the next element
87
88          ;else, swap array[i-1] with array[i]
89          push DWORD [ESI]
90          push DWORD [ESI-4]
91          pop DWORD [ESI]
92          pop DWORD [ESI-4]
93
94          sub ESI, 4          ;Move to the previous element in the array
95          dec EAX             ;and decrease 'i'
96
97      BackToMainLoop:
98          jmp MainLoop        ;Loop back to the top
99
100     IncreaseCounter:
101         inc EAX             ;and increasing 'i'
102         add ESI, 4          ;Moving to the next element in the array
103         jmp BackToMainLoop
104
105     EndLoop:
106
107     popad       ; Restoring the registers
108     ret 6       ; return and clear parameters
```

This function sorts the input using the earlier mentioned algorithm. It preservers the registers also and returns back to _start.

4. Print the sorted array and Termination of program

```
46
47          PutStr  output_msg      ; display sorted input numbers
48          nwln
49          mov     EBX,array       ; store starting pointer of array in EBX register
50          mov     CX,DX           ; CX := number count
51  print_loop:
52          PutLInt  [EBX]          ; Print current element
53          nwln                    ; Print new line
54          add     EBX,4           ; Increment pointer to next element
55          loop    print_loop      ; Loop till CX is non zero
56
57  end_iteration:
58          PutStr  end_msg         ; ask user to terminate program
59          GetCh   AL              ; store input char into AL
60          cmp     AL,'N'          ; compare accordingly and decide
61          je    _start            ; to start again or end program
62  done:
63          .EXIT
64
```

This part prints the sorted array first and then asks user to terminate the program or not . If user wants to again sort some different list of integers, he can give input ‘N’.

Running Lab5.asm on Linux using nasm (instructions also inlcuded in help file)

```
mangesh2102000@Linux-Ubuntu:~$ cd Documents/
mangesh2102000@Linux-Ubuntu:~/Documents$ ls
1801CS16_Lab5_Report.odt  file  file1  file1.cpp  file.cpp  help  input.txt  io.mac  io.o  lab5  lab5.asm  lab5.o  output.txt
mangesh2102000@Linux-Ubuntu:~/Documents$ nasm -f elf lab5.asm
mangesh2102000@Linux-Ubuntu:~/Documents$ ld -m elf_i386 lab5.o io.o -o lab5
mangesh2102000@Linux-Ubuntu:~/Documents$ ./lab5
Enter elements of the array ( Enter zero to terminate the input. ) :
5
4
3
1
2
0
Sorted List of given integers :
1
2
3
4
5
Terminate the program ? (Y/N) N
Enter elements of the array ( Enter zero to terminate the input. ) :
-524
357
-987
36879
124
0
Sorted List of given integers :
-987
-524
124
357
36879
Terminate the program ? (Y/N) Y
mangesh2102000@Linux-Ubuntu:~/Documents$
```