Inline assembly C++ - Insertion Sort Implementation

Linda Gutierrez Montoya,

***Abstract*—This is the presentation of the integration of three technologies to create a program which pretends to make a implementation of insertion sort algorithm using assembler code.**

***Keywords*—C++, Assembler, Sort, Algorithm.**

I. Introduction

Given a file called aleatorio.txt with n random generate numbers in it, where 10 ¡ n ¡ 1000, take the numbers and throught the implementation of a sort algorithm using C++ inline assembler, return the n sort numbers in a new file called salida.txt

A. Specifications

This program was developed in Ubuntu 14.04 OS, C++ Compiler GCC and NASM 2.12.02 80x86 assembler

Constraints: n between 10 and 1000
Input: A number which mean a execution option
output: A file with sorted numbers
Files: makefile main.cpp proceso.asm

II. Solution description

Go to the directory where mentioned files are storaged, exec the command make in the terminal, and then type ./main. After that, the program will ask you for choose an option.

When the selection is done, the program starts its procedures... if option 1 was selected, the program take a file called aleatorio, in the current directory and starts reading it, line per line, to create an array of ints; if option 2 was selected the program creates the file aleatorio and writes random numbers on it, after that, it does the same thing as in option 1 and call the assembler function, proceso.

Proceso function is the implementation of insertion sort in assembler code.

This function has three sections for, while and while quit section. The first one moves i pointer in the array, the second one moves j pointer, and the last one is where are the items are swap, if it's the case.

When the algorithm has finished the array is already sorted an ready to be saved in salida.txt file, using saveFile function in main.cpp.

INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Fig. 1. Insertion sort

III. Conclusion

1) Assembler instructions change between Operative systems
2) Assembler language can be use to perform any kind of algorithm
3) Integration between languages allow us to optimize time and resources in the machine

References

[1] http://web.archive.org/web/20120822144129/
[2] https://www.youtube.com/watch?v=Jg9NBFwYJZM
[3] http://www.dreamincode.net/forums/topic
[4] http://www.alpcentauri.info/assm$_5$.htm