CC4301 Arquitectura de Computadores – Tarea 3 – Primavera 2020 – Profesor: Luis Mateu

La función *sort* está programada en assembler x86 en el archivo *sort-x86.s*. Esta función ordena lexicográficamente un arreglo de strings usando un algoritmo ridículamente ineficiente. El código equivalente en C está comentado, mostrando la ubicación de las variables en los registros.

El encabezado de la función es: *void sort(char *noms[ ], int n);*

En el arreglo de strings *noms* vienen *n* nombres de personas como *"Gonzalez DIEGO"*. Siempre viene primero el apellido y luego el nombre, separados por un solo espacio en blanco. El string no contiene otros espacios en blanco. Considere que los caracteres están en ASCII estándar de 7 bits. Está codificación no incluye letras acentuadas o la ñ.

El archivo *sort-x86-**im**.s* es una copia de *sort-x86.s*. Modifique la función *sort* en ***sort-x86-im.s*** de modo que se ordene el arreglo ignorando diferencias entre mayúsculas y minúsculas. La siguiente tabla muestra el ordenamiento lexicográfico versus el mismo ignorando diferencias entre mayúsculas y minúsculas:

Orden lexicográfico	*Ignorando diferencias entre mayúsculas/minúsculas*
FERNANDEZ maria	FERNANDEZ maria
Gonzalez DIEGO	fernandez monica
Gonzalez peDro	fernandez veri
PEREZ JOSE	gonzalez ana
fernandez monica	gonzalez Diego
fernandez veri	Gonzalez DIEGO
gonzalez Diego	Gonzalez peDro
gonzalez ana	jerez tatiana
jerez tatiana	perez alberto
pereZ juan	PEREZ JOSE
perez alberto	perez josefa
perez josefa	pereZ juan

Observe que en el orden lexicográfica las mayúsculas son menores que las minúsculas.

Instrucciones

Baje *t3.zip* de U-cursos y descomprímalo. Contiene el *Makefile* y los programas que necesita para hacer esta tarea. Compile y ejecute con:

```
$ make test-sort-x86
$ ./test-sort-x86
```

Se mostrará el contenido del arreglo ordenado *lexicográficamente*. Reprograme en assembler la función *sort* en el archivo ***sort-x86-im.s***, de manera que ordene lexicográficamente ignorando diferencias entre mayúsculas y minúsculas. Compile y ejecute con:

```
$ make run-sort-x86-im
```

Este comando invoca el script *verificar* para revisar que su ordenamiento es correcto.

Restricciones

Ud. solo puede modificar en *sort-x86-im.s* el código que compara los elementos consecutivos. Este código está delimitado por un par de comentarios en el archivo. No modifique nada más. Sin esta restricción la tarea sería trivial. Además para hacer la comparación *no puede* invocar otras funciones (como *strcmp* por ejemplo).

Ayuda

- El archivo *sort-c.c* es la versión en C de la función *sort*. Compile y ejecute esta versión con: *make test-sort-c; ./test-sort-c*
- Programe primero una versión en C de lo pedido en el archivo *test-sort-c-im.c*. Compile y ejecute con: *make run-sort-c-im*
- Antes de comparar un caracter *c* con otro, llévelo a minúsculas con:

```
c = 'A'<=c && c<='Z' ? 'a'+c-'A' : c.
```

- Puede obtener ideas de las instrucciones en assembler x86 que debe usar generando el archivo *sort-c-im.s* con: *make sort-c-im.s*. El código en assembler estará en el archivo *sort-c-im.s*. Asegúrese que su solución en C funcione correctamente antes de empezar a codificar su solución en assembler.
- Use *ddd* para *entender* y depurar su tarea. Seleccione el menú *View → Machine code window* para ver el assembler. Coloque breakpoints en lugares estratégicos con: *break .while_begin*. Lea la guía rápida para usar gdb en:

  http://www.dcc.uchile.cl/~lmateu/CC4301
- Debe compilar la tarea en 32 bits. Para ello debe instalar las bibliotecas de 32 bits con *sudo apt-get install gcc-multilib* en distribuciones derivadas de *Debian*, como por ejemplo *Ubuntu*.

Entrega

Entregue por medio de U-cursos el archivo ***sort-x86-im.s*** con su solución. Su tarea debe ordenar correctamente, si no su tarea será rechazada. Se descontará medio punto por día de atraso (excluyendo sábados, domingos, festivos o vacaciones).