CC3301 Programación de Software de Sistemas – Tarea 5 – Otoño 2022 – Profesor: Luis Mateu

La función *sort* está programada en C y en assembler Risc-V en los archivos *sort-c.c* y *sort-rv.s*. Esta función ordena lexicográficamente un arreglo de strings usando un algoritmo ridículamente ineficiente. En el archivo en assembler, 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 *frases[ ], int n);*

En el arreglo de strings *frases* vienen *n* frases como:

```
"defina universo y de dos ejemplos"
"soy inseguro, ¿o no?"
```

Los archivos *sort-c-esp.c* y *sort-rv-esp.s* son copias de *sort-c-esp.c* y *sort-rv-esp.s* respectivamente. Modifique las funciones *sort* en ***sort-c-esp.c*** y ***sort-rv-esp.s*** de modo que se ordene el arreglo ascendentemente por la cantidad de espacios en blanco encontrados en cada frase. En los 2 ejemplos, la primera frase contiene 5 espacios y la segunda 3 espacios, por lo que la segunda frase debe ir antes que la primera.

Instrucciones

Baje *t5.zip* de U-cursos y descomprímalo. Contiene el *Makefile* y los archivos que necesita para hacer esta tarea. Ejecute el comando *make* sin parámetros para recibir instrucciones sobre la ubicación de los archivos que debe modificar y cómo compilar, ejecutar y depurar.

Restricciones

Ud. solo puede modificar en *sort-c-esp.c* y *sort-rv-esp.s* el código que compara los elementos consecutivos. Está claramente delimitado en los archivos originales. No modifique nada más. Sin esta restricción la tarea sería trivial. Además para hacer la comparación *no puede* invocar otras funciones. En la versión en assembler, una vez hecha la comparación, la ejecución debe continuar en la etiqueta *.decision* y el resultado de la comparación debe quedar en el registro *t1*. Si $t1 \leq 0$, las frases en *p[0]* y *p[1]* están bien ordenada y por lo tanto no se intercambiarán.

Ayuda

- Programe primero la versión en C del nuevo *sort* en el archivo *sort-c-esp.c*. Solo cuando funcione, base su solución en assembler en su solución en C.
- Puede obtener ideas de las instrucciones en assembler RiscV que debe usar generando el archivo *sort-c-esp.s* con: *make sort-c-esp.**s***.
- Lance *ddd* para depurar su tarea con los comandos:

  make sort-c-esp.ddd

  make sort-rv-esp.ddd

  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/gdb.txt
- Revise si el toolchain para compilar y ejecutar programas para RiscV está instalado en */opt/riscv*. Si no encuentra ese directorio, descargue el archivo *riscv.tgz* de esta carpeta de google drive e instale el *toolchain* para RiscV con estos comandos:

  ```
  sudo bash
  cd /
  tar zxvf ...ruta.../riscv.tgz
  ```
- En la clase auxiliar del viernes 6 de mayo se estudió la solución de una tarea similar de un semestre pasado.

Entrega

Entregue por medio de U-cursos el archivo ***sort-esp.zip*** generado con el comando *make zip*. Este comando verifica que su tarea funcione correctamente y luego genera *sort-esp.zip* con *sort-c-esp.c*, *sort-rv-esp.c* y *resultados.txt* con la ejecución de la tarea. **Recuerde descargar el archivo que subió, descomprimirlo e inspeccionar el contenido para verificar que son los archivos correctos.** Su tarea debe ordenar correctamente, si no será rechazada. Se descontará medio punto por día de atraso (excluyendo sábados, domingos, festivos o vacaciones).