Universidad de Buenos Aires
Facultad de Ingeniería
Año 2018 - 2do Cuatrimestre

66.20 Organización de Computadoras - Práctica Jueves

TRABAJO PRÁCTICO 1: conjunto de instrucciones MIPS
FECHA DE ENTREGA: 11 de Octubre de 2018

INTEGRANTES:

A, Maximiliano - 11111
<a@gmail.com>
PEREZ, Flavio - 11111
<a@gmail.com>
YACOBUCCI, Maximiliano - 93321
<maxyacobucci@gmail.com>

1 Introducción

En este trabajo práctico se busca aprender, practicar y perfeccionar los conocimientos relacionados al conjunto de instrucciones MIPS y el concepto de ABI.

El programa será realizado en lenguaje C a excepción de una función (quicksort) que tendrá dos versiones: una MIPS32 y una en C. El mismo se ejecutará teniendo en cuenta los parámetros que se se especifiquen por consola.

2 Diseño e Implementación

2.1 Diseño

El programa consiste en una implementación del algoritmo Quicksort, recibiendo como argumento el archivo que contiene datos que se desean ordenar (se asume que cada cadena de caracteres a ordenar aparece de a una por línea), y devolviendo por stdout los valores ordenados. En caso de errores, se devuelven por stderr.

Asimismo, el programa contará con una sección de ayuda que mostrará las opciones disponibles. El usuario que lo utilice deberá ingresar por línea de comando una serie de parámetros describiendo el archivo que contiene los datos y el archivo de salida, pudiendo indicar si se quiere ordenar números.

Es importante destacar que el programa será portable, teniendo una versión del algoritmo quicksort en C y otra en MIPS32 para dar soporte generico a aquellos entornos que carezcan de una versión más especıfica .El programa será compilado y posteriormente enlazado usando herramientas de GNU disponibles en el sistema NetBSD, generando una aplicación ejecutable.

2.2 Implementación

Inicialmente el programa revisa los parámetros indicados por línea de consola. En caso de no haber errores en ellos se procede a correr el algoritmo correspondiente. Si se solicita ayuda se mostrará el cartel con los comandos disponibles. Si se quiere averiguar la versión del programa, el mismo cuenta con una opción para indicarla. Si en cambio se indicó un archivo para ser ordenado, procederá a leerlo.

Luego de leer el archivo se procede a llamar al algoritmo de quicksort. Como se comentó anteriormente, existen 2 versiones del mismo, una en lenguaje C y otra en MIPS32. Ambos contienen su propia función atoi. En el algoritmo se usan punteros a char siendo identificado el primero como izq y el último como der. En el caso del algoritmo en lenguaje Assembler se hizo uso de la ABI, almacenando por el callee a los argumentos correspondientes a los registros a0-a3.

Por último, se libera la memoria utilizada y se imprimen en pantalla por stdout los datos ordenados,

3 Comando para compilar el programa

En primer lugar para compilar el programa se debe abrir una consola, pararse en la carpeta correspondiente al programa y ejecutar la siguiente sentencia para compilarlo:

```
gcc -std=c99 -o qsort main.c quicksort.S
```

En caso que se quiera utilizar la versión del algoritmo de quicksort en C:

```
gcc -std=c99 -o qsort main.c quicksort.c
```

Luego se lo puede ejecutar en cualquiera de sus 3 usos:

1. "./qsort -h": muestra la ayuda que indica las diferentes formas de ejecutar el programa

2. "./qsort -V": muestra la versión del programa
3. "./qsort [options] archivo"; ejercuta el programa teniendo como entrada el archivo indicado

En este último caso de uso se pueden indicar las siguiente opciones:

1. -o, –output: Archivo de salida
2. -n, –numeric: Para ordenar los datos numéricamente en vez de alfabéticamente

4 Resultados obtenidos

A continuación se mencionarán algunas pruebas realizadas con sus correspondientes resultados. Recordar previamente compilar el programa tal como se explicó en la sección anterior.

- ./qsort -h

```
Usage:
./qsort -h
./qsort -V
./qsort [options] archivo
Options:
-h, –help Imprime ayuda.
-V, –version Version del programa.
-o, –output Archivo de salida.
-n, –numeric Ordenar los datos numericamente en vez de alfabeticamente.
Examples:
./qsort -n numeros.txt
```

- ./qsort -V

```
1.0
```

- ./qsort numeros.txt (conteniendo este archivo 10 numeros del 1 al 10 desordenados)

```
—1
—10
—2
—3
—4
—5
—6
—7
—8
—9
```

Como se puede observar, los números salieron ordenados alfabeticamente, estando el 10 luego del 1.

- ./qsort -n numeros.txt

```
—1
—2
—3
—4
—5
—6
—7
—8
—9
—10
```

En este caso los números se ordenaron numéricamente, estando el 10 luego del 9.

- ./qsort text1.txt

```
— aaaaaaa
—aaaaaaa
—aaaaaaa a
—aaaaaaa b
—aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
—hola
—que
—tal
```

Aquí se observa que el programa ordenó el archivo alfabéticamente, teniendo en cuenta también el texto siguiente al espacio en una línea. Lo mismo sucederá en el siguiente ejemplo.

- ./qsort zeta.txt

```
—zzzzzzzzzzz a
—zzzzzzzzzzz b
—zzzzzzzzzzz sabia que Asuntos Internos le tendia una trampa
```

- ./qsort archivoInexistente.txt

```
cannot open input file.
```

Al no existir el archivo indicado en la instrucción, el programa tira un error descripivo.

- ./qsort vacio.txt

```
NO ESTA FUNCIONANDO,
```

- ./qsort -n -o - numeros.txt archivo.txt

```
DEBERIA GUARDAR EN ARCHIVO.TXT?
```

5 Conclusiones

En este trabajo práctico hemos aprendido a combinar programación en C con MIPS32 logrando una portabilidad que puede ser importante en caso de ser necesario. Asimismo, pudimos observar cómo un algoritmo realizado en C, puede traducirse en Assembler, viendo detalladamente como surgen las distintas instrucciones. Relacionado a esto último, se hizo uso de la ABI, almacenando por el callee a los argumentos correspondientes a los registros a0-a3, comprendiendo la importancia de su uso.