66:20 Organización de computadoras
Trabajo práctico 1: Conjunto de instrucciones MIPS.

1. Objetivos

El objetivo de este trabajo es familiarizarse con la programación en assembler MIPS y con la ABI(Application Binary Interface).

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 7), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo[1], y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Descripción.

Al igual que el TP0, el presente trabajo práctico tiene como objetivo implementar el algoritmo de ordenamiento `mergesort`[3], esta vez en assembly de MIPS32. La implementación de este trabajo práctico debe tomar entrada exclusivamente desde *stdin*. La salida debe imprimirse por *stdout*, mientras que los errores deben imprimirse por *stderr*.

5. Implementación.

5.1. ABI

La implementación debe respetar la ABI usada por la cátedra[4], que difiere de la utilizada por el GCC.

[1] `http://groups.yahoo.com/group/orga6620`

5.2. Uso de memoria dinámica

Para hacer uso de memoria dinámica, se provee una implementación de *malloc* y *free* escrita en Assembly MIPS32. La función `realloc` debe ser implementada como parte del trabajo práctico.

5.3. Entrada / Salida

Para poder realizar el trabajo práctico, es necesario interactuar con el sistema operativo mediante las llamadas al sistema `SYS_read` y `SYS_write`. En la clase del 20/4 se explicaron los pasos necesarios:

- Cargar el número de llamada al sistema en el registro $v0.
- Cargar los argumentos en los registros correspondientes.
- Ejecutar la llamada mediante la instrucción `syscall`.
- Verificar la existencia de errores en el registro $a3.
- Obtener el valor retornado del registro $v0.

Se adjunta a este enunciado una versión minimalista del comando `echo` de UNIX a modo de ejemplo.

6. Ejemplos

```
$echo -n "9876543210" >digits.txt
$tp1 < digits.txt
0123456789$

$cat letters.txt
aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ$

$tp1 < letters.txt
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz$

$ head -c 64 /dev/urandom > random.txt
$ hexdump -C random.txt
00000000  5f 8e a1 e9 2d 25 49 85  77 04 05 19 16 ca 48 fc  |_...-%I.w.....H.|
00000010  5f d0 ea 84 a2 17 5a 86  e3 93 1c 79 19 bd 32 dc  |_.....Z....y..2.|
00000020  b9 2b 76 ed 75 49 95 37  43 c4 fe 22 d5 a5 bf 56  |.+v.uI.7C.."...V|
00000030  84 6a 06 0d 56 50 f7 b5  cb 41 7c fb 2d 33 49 b7  |.j..VP...A|.-3I.|
00000040
$tp1 < random.txt  > sorted.txt
$ hexdump -C sorted.txt
00000000  84 84 85 86 8e 93 95 a1  a2 a5 b5 b7 b9 bd bf c4  |................|
00000010  ca cb d0 d5 dc e3 e9 ea  ed f7 fb fc fe 04 05 06  |................|
00000020  0d 16 17 19 19 1c 22 25  2b 2d 2d 32 33 37 41 43  |......"%+--237AC|
00000030  48 49 49 49 50 56 56 5a  5f 5f 6a 75 76 77 79 7c  |HIIIPVVZ__juvwy||
00000040
```

7. Informe.

El informe debe incluir:

- Informe describiendo el desarrollo del trabajo práctico.
- Diagramas del stack de cada función involucrada.
- Comando(s) para compilar el programa.
- Corridas de prueba, con los comentarios pertinentes.
- CD conteniendo todo el material digital.
- Código fuente.
- Este enunciado.

8. Fechas de entrega.

- Primera entrega: Jueves 3 de Mayo.
- Revisión: Jueves 10 de Mayo.
- Vencimiento: Jueves 17 de Mayo.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project, `http://www.netbsd.org/`.

[3] Merge-sort `http://en.wikipedia.org/wiki/Merge_sort`

[4] `func_call_conv.pdf` en `http://groups.yahoo.com/group/orga6620/files/`.