Informe TP1

Miguel Alfaro, *Padrón Nro. 97.743*
miguel.alfaro.a95@gmail.com
Martín Errázquin, *Padrón Nro. 98.017*
martin.errazquin@hotmail.com
Agustina Mendez, *Padrón Nro. 98.338*
abmendez25@hotmail.com

1er. Cuatrimestre de 2017
66.20 Organización de Computadoras − Práctica
Martes
Facultad de Ingeniería, Universidad de Buenos Aires

Índice

1. Diseño

Se utilizó como base el diseño del Trabajo Práctico 0, realizándole algunos cambios a la sección principal (main) para adecuarlo a las nuevas exigencias. Se mantuvo el parser sin modificaciones, ya que el trabajo de leer e interpretar los argumentos sigue siendo del main, y por lo tanto sigue funcionando como lo hacía previamente. Para el caso de las llamadas a las funciones encode y decode, se debieron implementar los siguientes cambios:

- Las funciones reciben sólo dos file descriptors, uno de entrada y uno de salida. Para lograr esto se continúa leyendo los archivos de la misma forma que antes, pero al pasarlos a las funciones se obtienen sus file descriptors a través de la función de C *fileno(FILE*)*.
- Los errores deben ser informados desde el main. Para esto, las funciones devuelven un código de error (diferente dependiendo de qué error se trata) y este código se traduce a un mensaje de error en el main a través del arreglo *errmsg*, que luego se imprime por stderr.

Luego se cambiaron ambas funciones (encode y decode), pasando de realizarlas en C a realizarlas en assembly.

1.1. Encode

Siguiendo la abi y la convention calls de MIPS, se obtienen los argumentos que recibe la función, que en este caso son los file descriptors de los archivos. El file descriptor del archivo de entrada se encuentra en el registro a0 y el de salida en el a1. El paso siguiente a leerlos es escribirlos en el ABA de la funcion caller. Luego se crea el respectivo stack frame de la funcion donde se alojan los registros que se deben mantener entre llamadas.

En la sección de datos se tiene un buffer de 4 bytes, de los cuales se usaran 3 para guardar los bytes en base 256 y luego se reemplazaran los 4 para guardar el resultado del encode de los bytes leídos.

Con la SYSCALL read se leen 3 bytes del archivo de entrada, y se guardan el buffer. Esta syscall devuelve en el registro v0 la cantidad de bytes leidos, la cual es cero significa si se llego al final de archivo (EOF).

Se cargan los 4 bytes del buffer a un registro, y luego con los shifts necesarios se obtienen los indices correspondientes en la tabla de base64.

Se tienen en cuenta la cantidad de bytes leidos para el padding, y por último se carga el buffer con los caracteres en base 64 ya indexados en la tabla.

FInalmente se llama a la SYSCALL write pasándole el file descriptor de salida que se había recibido como argumento, y se escriben los 4 bytes en base64 en dicho archivo.

Se limpia el buffer y se vuelven a leer 3 bytes (en base 256)

Luego de escribir los bytes en base64 se tiene que volver a la funcion que nos llamo, en este caso main(), para lo que se restauran los registros fp, gp, ra, y se realiza un jump.

1.2. Decode

La lógica de decoding es similar a la del tp0, dados 4 bytes se chequea que correspondan efectivamente a base64 y si lo hacen se les suma o resta el offset

correspondiente para obtener el lugar correspondiente en la tabla; caso contrario se lanza un error de decoding. Nótese que si al intentar leer 4B del fd de entrada el resultado no es o bien 0 (EOF) o bien 4 (success) también se devuelve error de decoding. Suponiendo que se han aplicado los offset correspondientes se realizan los shifts necesarios para manipular los bits correspondientes a cada byte del resultado, obteniéndose los 3B de base 256, tras lo cual se imprimen en pantalla mediante la syscall SYS_write. Si se lee EOF o bien por entrada vacía o porque se terminó de decodear, la función termina restaurando los valores salvados al inicio de la misma, respetando la ABI.

2. Stack frame

Al codificar las funciones en assembly se tuvo en cuenta la convención de llamadas a funciónes ABI de MIPS. Tanto la función de encode como de decode son funciones leaf, es decir, que dentro de ellas no se llama a otras funciones.

Entonces, cumpliendo con la ABI, se deben guardar en la SRA (Saved registers area) tanto el gp como el fp. También se guardó el ra aunque en las funciones no se modifique.

También se debe cumplir que el tamaño del stack frame sea múltiplo de 8 bytes, por lo que el tamaño total es de 16 bytes. No se reserva espacio para el ABA (Argument building area) porque no se llama a ninguna otra función, y tampoco para el LTA (Local and temporary area) porque las variables locales y temporales no se modifican en la ejecución o, si se modifican, lo hace la propia función.

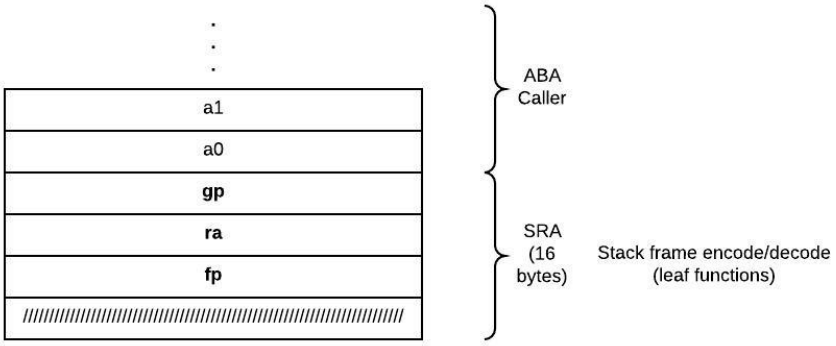

Figura 1: Stack frame encode/decode

3. Compilación

La compilación se debe realizar en el emulador NetBSD. Deben generarse los archivos objeto de cada módulo, haciendo: *gcc -c -ggdb -Wall tp1.c*, *gcc -c -ggdb -Wall parser/parser_tp0.c* y *gcc -c -ggdb -Wall base64.S*. Luego, unirlos para obtener el ejecutable final, haciendo : *gcc tp1.o parser_tp0.o base64.o -o tp1*. Alternativamente, se puede ejecutar el script *compilar.sh*

4. Tests

Para probar el correcto funcionamiento del trabajo práctico se realizaron diferentes pruebas de encode y decode

La primer entrada a probar fue "Man":

```
# echo -n Man > in.bin
# hexdump -C in.bin
00000000  4d 61 6e                                          |Man|
00000003
# ./tpexe -i in.bin -o out64.bin
# hexdump -C out64.bin
00000000  54 57 46 75                                       |TWFu|
#./tpexe -i out64.bin -o out256.bin -a decode
#hexdump -C out256.bin
00000000  4d 61 6e                                          |Man|
00000003
```

La sigueinte prueba fue decodear con padding:

```
# echo -n b > in.bin
# hexdump -C in.bin
00000000  62                                                |b|
00000001
# ./tpexe -i in.bin -o out64.bin
# hexdump -C out64.bin
00000000  59 67 3d 3d                                       |Yg==|
00000004
# ./tpexe -i out64.bin -o out256.bin -a decode
# hexdump -C out256.bin
00000000  62                                                |b|
```

La siguiente prueba fue una frase larga:

```
# hexdump -C in.bin
00000000  4d 61 6e 20 69 73 20 64  69 73 74 69 6e 67 75 69  |Man is distingui|
00000010  73 68 65 64 2c 20 6e 6f  74 20 6f 6e 6c 79 20 62  |shed, not only b|
00000020  79 20 68 69 73 20 72 65  61 73 6f 6e 2c 20 62 75  |y his reason, bu|
00000030  74 20 62 79 20 74 68 69  73 20 73 69 6e 67 75 6c  |t by this singul|
00000040  61 72 20 70 61 73 73 69  6f 6e 20 66 72 6f 6d 20  |ar passion from |
00000050  6f 74 68 65 72 20 61 6e  69 6d 61 6c 73 2c 20 77  |other animals, w|
00000060  68 69 63 68 20 69 73 20  61 20 6c 75 73 74 20 6f  |hich is a lust o|
00000070  66 20 74 68 65 20 6d 69  6e 64 2c 20 74 68 61 74  |f the mind, that|
00000080  20 62 79 20 61 20 70 65  72 73 65 76 65 72 61 6e  | by a perseveran|
00000090  63 65 20 6f 66 20 64 65  6c 69 67 68 74 20 69 6e  |ce of delight in|
000000a0  20 74 68 65 20 63 6f 6e  74 69 6e 75 65 64 20 61  | the continued a|
000000b0  6e 64 20 69 6e 64 65 66  61 74 69 67 61 62 6c 65  |nd indefatigable|
000000c0  20 67 65 6e 65 72 61 74  69 6f 6e 20 6f 66 20 6b  | generation of k|
000000d0  6e 6f 77 6c 65 64 67 65  2c 20 65 78 63 65 65 64  |nowledge, exceed|
000000e0  73 20 74 68 65 20 73 68  6f 72 74 20 76 65 68 65  |s the short vehe|
000000f0  6d 65 6e 63 65 20 6f 66  20 61 6e 79 20 63 61 72  |mence of any car|
00000100  6e 61 6c 20 70 6c 65 61  73 75 72 65 2e           |nal pleasure.|
0000010d
# ./tpexe -i in.bin -o out64.bin
# hexdump -C out64.bin
00000000  54 57 46 75 49 47 6c 7a  49 47 52 70 63 33 52 70  |TWFuIGlzIGRpc3Rp|
00000010  62 6d 64 31 61 58 4e 6f  5a 57 51 73 49 47 35 76  |bmd1aXNoZWQsIG5v|
00000020  64 43 42 76 62 6d 78 35  49 47 4a 35 49 47 68 70  |dCBvbmx5IGJ5IGhp|
00000030  63 79 42 79 5a 57 46 7a  62 32 34 73 49 47 4a 31  |cyByZWFzb24sIGJ1|
00000040  64 43 42 69 65 53 42 30  61 47 6c 7a 49 48 4e 70  |dCBieSB0aGlzIHNp|
00000050  62 6d 64 31 62 47 46 79  49 48 42 68 63 33 4e 70  |bmd1bGFyIHBhc3Np|
```

```
00000060  62 32 34 67 5a 6e 4a 76  62 53 42 76 64 47 68 6c  |b24gZnJvbSBvdGhl|
00000070  63 69 42 68 62 6d 6c 74  59 57 78 7a 4c 43 42 33  |ciBhbmltYWxzLCB3|
00000080  61 47 6c 6a 61 43 42 70  63 79 42 68 49 47 78 31  |aGljaCBpcyBhIGx1|
00000090  63 33 51 67 62 32 59 67  64 47 68 6c 49 47 31 70  |c3Qgb2YgdGhlIG1p|
000000a0  62 6d 51 73 49 48 52 6f  59 58 51 67 59 6e 6b 67  |bmQsIHRoYXQgYnkg|
000000b0  59 53 42 77 5a 58 4a 7a  5a 58 5a 6c 63 6d 46 75  |YSBwZXJzZXZlcmFu|
000000c0  59 32 55 67 62 32 59 67  5a 47 56 73 61 57 64 6f  |Y2Ugb2YgZGVsaWdo|
000000d0  64 43 42 70 62 69 42 30  61 47 55 67 59 32 39 75  |dCBpbiB0aGUgY29u|
000000e0  64 47 6c 75 64 57 56 6b  49 47 46 75 5a 43 42 70  |dGludWVkIGFuZCBp|
000000f0  62 6d 52 6c 5a 6d 46 30  61 57 64 68 59 6d 78 6c  |bmRlZmF0aWdhYmxl|
00000100  49 47 64 6c 62 6d 56 79  59 58 52 70 62 32 34 67  |IGdlbmVyYXRpb24g|
00000110  62 32 59 67 61 32 35 76  64 32 78 6c 5a 47 64 6c  |b2Yga25vd2xlZGdl|
00000120  4c 43 42 6c 65 47 4e 6c  5a 57 52 7a 49 48 52 6f  |LCBleGNlZWRzIHRo|
00000130  5a 53 42 7a 61 47 39 79  64 43 42 32 5a 57 68 6c  |ZSBzaG9ydCB2ZWhl|
00000140  62 57 56 75 59 32 55 67  62 32 59 67 59 57 35 35  |bWVuY2Ugb2YgYW55|
00000150  49 47 4e 68 63 6d 35 68  62 43 42 77 62 47 56 68  |IGNhcm5hbCBwbGVh|
00000160  63 33 56 79 5a 53 34 3d                           |c3VyZS4=|
# ./tpexe -i out64.bin -o out256.bin -a decode
# hexdump -C out256.bin
00000000  4d 61 6e 20 69 73 20 64  69 73 74 69 6e 67 75 69  |Man is distingui|
00000010  73 68 65 64 2c 20 6e 6f  74 20 6f 6e 6c 79 20 62  |shed, not only b|
00000020  79 20 68 69 73 20 72 65  61 73 6f 6e 2c 20 62 75  |y his reason, bu|
00000030  74 20 62 79 20 74 68 69  73 20 73 69 6e 67 75 6c  |t by this singul|
00000040  61 72 20 70 61 73 73 69  6f 6e 20 66 72 6f 6d 20  |ar passion from |
00000050  6f 74 68 65 72 20 61 6e  69 6d 61 6c 73 2c 20 77  |other animals, w|
00000060  68 69 63 68 20 69 73 20  61 20 6c 75 73 74 20 6f  |hich is a lust o|
00000070  66 20 74 68 65 20 6d 69  6e 64 2c 20 74 68 61 74  |f the mind, that|
00000080  20 62 79 20 61 20 70 65  72 73 65 76 65 72 61 6e  | by a perseveran|
00000090  63 65 20 6f 66 20 64 65  6c 69 67 68 74 20 69 6e  |ce of delight in|
000000a0  20 74 68 65 20 63 6f 6e  74 69 6e 75 65 64 20 61  | the continued a|
000000b0  6e 64 20 69 6e 64 65 66  61 74 69 67 61 62 6c 65  |nd indefatigable|
000000c0  20 67 65 6e 65 72 61 74  69 6f 6e 20 6f 66 20 6b  | generation of k|
000000d0  6e 6f 77 6c 65 64 67 65  2c 20 65 78 63 65 65 64  |nowledge, exceed|
000000e0  73 20 74 68 65 20 73 68  6f 72 74 20 76 65 68 65  |s the short vehe|
000000f0  6d 65 6e 63 65 20 6f 66  20 61 6e 79 20 63 61 72  |mence of any car|
00000100  6e 61 6c 20 70 6c 65 61  73 75 72 65 2e           |nal pleasure.|
```

5. Conclusiones

Este trabajo nos resultó interesante y como una buena experiencia a la hora de ver e identificar como las funciones en C funcionan a más bajo nivel. Tambien el manejo de SYSCALLS y stack frames (que es algo nuevo) fue muy útil, y aunque nos encontramos con dificultades en el proceso del trabajo, se pudieron resolver luego de leer la bibliografía recomendada de MIPS.