

66.20 Organización de Computadoras

Trabajo Práctico 0:

Infraestructura básica

Burdet Rodrigo, *Padrón Nro. 93440*
rodrigoburdet@gmail.com

Colangelo Federico, *Padrón Nro. 89869*
federico.colangelo@semperti.com

Manzano Matias, *Padrón Nro. 83425*
matsebman@gmail.com

2do. Cuatrimestre de 2014
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

21 de octubre de 2014

1. Objetivos

Familiarizarse con la programación en assembly y el concepto de ABI , implementando un programa (y su correspondiente documentación) que resuelva el problema piloto que presentaremos más abajo.

2. Resumen

En el presente trabajo, se implementó un algoritmo que permite graficar los conjuntos de Mandelbrot dados ciertos parámetros para permitir centrarnos en una región en particular de dicho conjunto. El programa fue realizado en 2 etapas. La primera en c, donde se crea el marco para el dibujo. La segunda parte en assembly MIPS, encargada de dibujar fractales a un archivo. El programa en c linkea contra el código assembly MIPS para lograr así más performance.

3. Desarrollo

3.1. Paso 1: Configuración de Entorno de Desarrollo

El primer paso fue configurar el entorno de desarrollo, de acuerdo a la guía facilitada por la cátedra. Trabajamos con distribuciones Linux y con el GxEmul proporcionado por la cátedra, emulando un sistema NetBSD.

3.2. Paso 2: Implementación del programa

El programa debe ejecutarse por línea de comando y la salida del mismo dependerá del valor de los argumentos con los que se lo haya invocado.

3.2.1. Ingreso de parámetros

El formato para invocar al programa es el siguiente:

```
./tp1-2014-2-bin [OPTIONS]
```

Los parámetros válidos que puede recibir el programa son los siguientes:

-o,	-output	(Parámetro obligatorio. Especifica archivo de salida, - para stdout)
-r,	-resolution	(Resolución de la imagen de salida).
-c,	-center	(Centro de la imagen).
-w,	-width	(Ancho del rectángulo a dibujar).
-H,	-height	(Alto del rectángulo a dibujar).
-v,	-version	(Muestra la versión).
-h,	-help	(Muestra la ayuda).

3.2.2. Interpretación de parámetros

Para parsear los parámetros se usó la librería de GNU getopt, en particular se usó getopt_long para permitir el pasaje de parámetros largos.

4. Compilación del programa

El proyecto puede ser corrido en cualquier ambiente pero se debe tener en consideración qué algoritmo se va a ejecutar ya que el código assembly está fuertemente ligado a un tipo de arquitectura. Es por esta razón que si queremos estamos sobre MIPS podremos linkear contra mips32plot.S pero eso no va a ser posible en x86 donde sólo podremos usar código c. Para compilar en MIPS se puede usar el comando make ya que el Makefile fue modificado para linkear el .S correspondiente. Para compilar en otro ambiente es necesario modificar el Makefile para que compile el código en c en vez del assembly.

5. Programa MIPS

El programa fue realizado siguiendo la ABI provista por la cátedra y generando un stack como se muestra a continuación

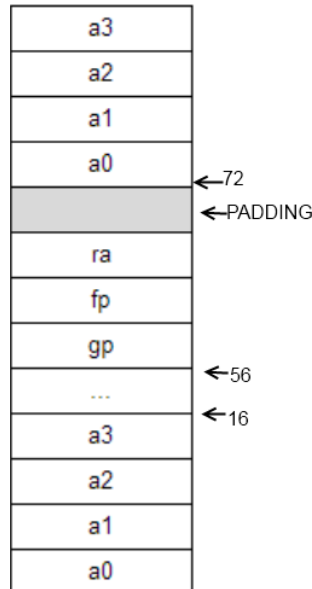


Figura 1: Stack frame generado

Se ve en la figura que LTA mide 40 bytes. El tamaño no fue arbitrario si no que se tuvieron en cuenta las variables locales que se tendrían que almacenar. Para debuggear se usaron llamadas de MIPS a C de la función printf en versiones para enteros y floats, las mismas se pueden ver al final del código adjunto como miprntf y printff respectivamente.

Nota: El programa en la versión actual contiene un bug que no permite la correcta impresión del brillo de los pixeles.

6. Corridas de prueba y Mediciones

7. Conclusiones

Como se enuncia en el objetivo de este trabajo práctico, aprendimos a instalar y manejar el GxEmul, a realizar transferencias de archivos en Linux, así como también compilar y ejecutar programas en el NetBSD. Por otro lado, aprendimos a manejar y escribir informes en \LaTeX . De este modo, estamos preparados para que en los próximos trabajos prácticos, nos aboquemos directamente al desarrollo de los mismos.

8. Código

8.1. mips32_plot.S

```
1 #include <mips/regdef.h>
2 #include <sys/syscall.h>
3
4 #defino las posiciones de los argumentos dentro del struct paramt_t
5 #define UL_re 0
6 #define UL_im 4
7 #define LR_re 8
8 #define LR_im 12
9 #define d_re 16
10 #define d_im 20
11 #define x_res 24
12 #define y_res 28
13 #define shades 32
14 #define FD 36
15
16 #define descriptor 14
17
18
19
20 #define FRAME_SIZE 72
21 #define GP_POS FRAME_SIZE-16
22 #define FP_POS FRAME_SIZE-12
23 #define RA_POS FRAME_SIZE-8
24 #define LTA_POS 16
25
26
27 #mips32_plot(param_t *);
28 .text
29 .align 2
30 .extern write_int
31 .globl mips32_plot
32 .ent mips32_plot
33 mips32_plot:
34 .frame $fp, FRAME_SIZE, ra
35 .set noreorder
36 .cpload t9
37 .set reorder
38 subu $sp, $sp, FRAME_SIZE
39 .cpstore GP_POS
40 sw $fp, FP_POS($sp)
41 sw ra, RA_POS($sp)
42 move $fp, $sp
43 sw a0, FRAME_SIZE($fp)
44 sw a1, FRAME_SIZE+4($fp)
45 sw a2, FRAME_SIZE+8($fp)
46 sw a3, FRAME_SIZE+12($fp)
47
48 #t0 = &parms
49 lw t0, FRAME_SIZE($fp)
50 lw t1, FD(t0)
51 #t1 = FD
52 lh t1, descriptor(t1) #me muevo al campo _file mediante un short
53
54 #imprimir cabecera
55 #imprimo P2
56 li v0, SYS_write
57 move a0, t1
58 la a1, msgP2
59 li a2, 3
60 syscall
61 #TODO: revisar error
```

```

62
63 #imprimo xres
64 lw t0,FRAME_SIZE($fp)
65 lw t1,FD(t0)
66 #t1 = FD
67 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
68 lw t0,x_res(t0)
69 move a0,t0
70 move a1,t1
71 jal write_int
72
73
74 #Imprimir fin de linea
75 lw t0,FRAME_SIZE($fp)
76 lw t1,FD(t0)
77 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
78
79 #t1 = FD
80 li v0,SYS_write
81 move a0,t1
82 la a1,endln
83 li a2,1
84 syscall
85
86
87 #imprimo yres
88 lw t0,FRAME_SIZE($fp)
89 lw t1,FD(t0)
90 #t1 = FD
91 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
92 lw t0,y_res(t0)
93 move a0,t0
94 move a1,t1
95 jal write_int
96
97
98 #Imprimir fin de linea
99 lw t0,FRAME_SIZE($fp)
100 lw t1,FD(t0)
101 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
102 #t1 = FD
103 li v0,SYS_write
104 move a0,t1
105 la a1,endln
106 li a2,1
107 syscall
108
109
110 #imprimo shades
111 lw t0,FRAME_SIZE($fp)
112 lw t1,FD(t0)
113 #t1 = FD
114 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
115 lw t0,shades(t0)
116 move a0,t0
117 move a1,t1
118 jal write_int
119
120 #Imprimir fin de linea
121 lw t0,FRAME_SIZE($fp)
122 lw t1,FD(t0)
123 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
124 #t1 = FD
125 li v0,SYS_write
126 move a0,t1
127 la a1,endln
128 li a2,1
129 syscall
130

```

```

131 #inicializo loop_y
132 li t0,0
133 sw t0,LTA_POS($fp) #guardo y=0 -> 0
134 lw t0,FRAME_SIZE($fp) #traigo parms
135 l.s $f4,UL_im(t0) #f4=UL_im
136 s.s $f4,LTA_POS+4($fp) #ci=UL_im -> 4
137 loop_y:
138
139 #carga y_res
140 lw t0,FRAME_SIZE($fp) #traigo parms
141 lw t0,y_res(t0) #t0 = parms->y_res
142 lw t1,LTA_POS($fp) #t1 = y
143 bge t1, t0, end # if (y > y_res) {end}
144
145 #inicializo loop_x
146 li t0,0
147 sw t0,LTA_POS+8($fp) #guardo x=0 -> 8
148 lw t0,FRAME_SIZE($fp) #traigo parms
149 l.s $f4,UL_re(t0) #f4 = UL_re
150 s.s $f4,LTA_POS+12($fp) #cr=UL_re -> 12
151
152 loop_x:
153 #carga x_res
154 lw t0,FRAME_SIZE($fp) #traigo parm
155 lw t0,x_res(t0) #t0 = parms->x_res
156 lw t1,LTA_POS+8($fp) #t1 = x
157 bge t1,t0,inc_y #if (x>x_rs){inc y}
158
159 l.s $f4,LTA_POS+4($fp) #f4 = ci
160 l.s $f6,LTA_POS+12($fp)
161 s.s $f4,LTA_POS+16($fp) # zi=ci -> 16
162 s.s $f6,LTA_POS+20($fp) # zr=cr -> 20
163
164 #inicializo loop_c
165 li t0,0
166 sw t0,LTA_POS+24($fp) # c -> 24
167
168 loop_c:
169
170 ##### ES UN ++c
171 lw t0,LTA_POS+24($fp)
172 addi t0,1
173 sw t0,LTA_POS+24($fp)
174
175 lw t1,FRAME_SIZE($fp) #traigo parms
176 lw t1,shades(t1) #t1 = parms->shades
177 bge t0,t1,write_out # if (c>parms->shades) {write_out}
178
179 l.s $f4, LTA_POS+20($fp) #f4 = zr
180 l.s $f6, LTA_POS+16($fp) #f6 = zi
181 mul.s $f4, $f4, $f4
182 mul.s $f6, $f6, $f6
183 add.s $f4, $f4, $f6
184 li.s $f6, 4.0
185 c.le.s $f6, $f4 # if 4<absz then write_out
186 bc1t write_out
187
188 l.s $f4, LTA_POS+20($fp) # f4 = zr
189 mul.s $f6, $f4, $f4
190 mul.s $f6, $f6, $f4 # f6 = zr ^ 3
191 l.s $f8, LTA_POS+16($fp)
192 mul.s $f8, $f8, $f8
193 mul.s $f8, $f4, $f8 # f8 = zi ^ 2 * zr
194 li.s $f10, 3.0
195 mul.s $f8, $f8, $f10
196 sub.s $f6, $f6, $f8
197 l.s $f8, LTA_POS+12($fp)
198 add.s $f6, $f6, $f8
199 s.s $f6, LTA_POS+28($fp)

```

```

200
201 l.s $f4,LTA_POS+20($fp) # f4 = zr
202 l.s $f6,LTA_POS+LTA_POS+16($fp) #f6 = zi
203 li.s $f8,3.0
204 mul.s $f4,$f4,$f4 #zr = zr^2
205 mul.s $f4,$f4,$f6 #f4 = zr^2*zi
206 mul.s $f4,$f4,$f8 #f4 = zr^2*zi*3
207
208 mul.s $f8,$f6,$f6 #f8 = zi*zi
209 mul.s $f6,$f8,$f6 #f6 = zi*zi*zi
210 sub.s $f4,$f4,$f6 #f4 = 3*zr^2 *zi - zi^3
211
212 l.s $f6,LTA_POS+4($fp)
213 add.s $f4, $f4,$f6
214 s.s $f4,LTA_POS+32($fp)
215
216 l.s $f4, LTA_POS+28($fp)
217 l.s $f6, LTA_POS+32($fp)
218 s.s $f4, LTA_POS+20($fp)
219 s.s $f6, LTA_POS+16($fp)
220
221 j loop_c
222
223
224 write_out:
225 lw t0, LTA_POS+24($fp)
226 lw t1, FRAME_SIZE($fp)
227 lw t1, FD(t1)
228 lh t1, descriptor(t1)
229 move a0, t0
230 move a1, t1
231 jal write_int
232
233 #Imprimir espacio
234 lw t0,FRAME_SIZE($fp)
235 lw t1,FD(t0)
236 lh t1,descriptor(t1) #me nuevo al campo _file mediante un short
237 li v0,SYS_write
238 move a0,t1
239 la a1,space
240 li a2,1
241 syscall
242 j inc_x
243
244 inc_x:
245 lw t0,LTA_POS+8($fp) #x
246 addi t0,1
247 sw t0,LTA_POS+8($fp)
248 lw t1,FRAME_SIZE($fp)
249 l.s $f4,d_re(t1) #f4 = parms->d_re
250 l.s $f6,LTA_POS+12($fp) #cr
251 add.s $f6,$f6,$f4
252 s.s $f6,LTA_POS+12($fp)
253 j loop_x
254
255 inc_y:
256 lw t0,LTA_POS($fp) #y
257 addi t0,1
258 sw t0,LTA_POS($fp)
259 lw t1,FRAME_SIZE($fp)
260 l.s $f4,d_im(t1) #f4 = parms->d_im
261 l.s $f6,LTA_POS+4($fp) #ci
262 sub.s $f6,$f6,$f4
263 s.s $f6,LTA_POS+4($fp)
264 j loop_y
265
266
267 #PRINTF FUNCIONANDO, imprime t0!!
268 miprintf:

```

```

269     la a0,print
270     move a1,t0
271     lw t9,%call16(sprintf) (gp)
272     jalr t9
273
274     #PRINTF DE FLOTS FUNCIONANDO, CARGAR FLOAT CON l.s en $f0
275 printf:
276     cvt.d.s $f0,$f0
277     la a0,print_float
278     mfc1 a2,$f0
279     mfc1 a3,$f1
280     lw t9,%call16(sprintf) (gp)
281     jalr t9
282
283 end:
284     lw ra,RA_POS($fp)
285     lw gp,GP_POS($fp)
286     lw $fp,FP_POS($fp)
287     addu sp,sp,FRAME_SIZE
288     jr ra
289
290     .end mips32_plot
291
292
293     .rdata
294 endl:
295     .asciiz "\n"
296 msgP2:
297     .asciiz "P2\n"
298 space:
299     .asciiz " "
300
301
302 print:
303     .asciiz "valor : %d \n"
304 print_float:
305     .asciiz "valor float : %f \n"

```