Homework 1: GCC calling convention

January 14, 2018

In this exercise, we will try to figure out the calling convention of gcc on a 64-bit platform. One way of doing this would be to call a routine with a different number of arguments and inspect the assembly generated by gcc. In Listing 1, main() routine makes calls to foo_8, foo_16, and foo_24 routines with eight arguments. Here, 8, 16, and 24 corresponds to the size of arguments (in bytes) which are passed to foo_* routines.

GCC compiles a C file to human readable assembly with -S option. For example, gcc -S hw1.c generates a file called hw1.s. The assembly output of Listing 1 is shown in Listing 2. Read the assembly output carefully and try to correlate it with the C source code.

Turn in:

- 1. Which line numbers in assembly code corresponds to argument passing. For every such line write the argument number and the target function.
- 2. Which line numbers in assembly code corresponds to argument receiving. For every such line write the argument number and the receiving function.
- 3. What do you think how arguments are passed in gcc? Do they differ, if the argument size is changed?
- 4. How does a function return a value to the caller? Write line number where the main routine receives the return value of foo_8.
- 5. Can you write foo_8 with less number of instructions and memory dereferences, without changing anything else? If yes, please write your optimized assembly. (Notice you can compile hw1.s using gcc hw1.s. Feel free to compile and run before submitting your changes.)
- 6. How is a function defined in GNU assembly? How do you declare global variables? Please look at the global_var declaration and foo_* definition in Listing 2 to figure it out. You may refer to https://sourceware.org/binutils/docs/as/ for assembler syntax.

```
1
2
  typedef unsigned long long arg8_t;
3
4 typedef struct {
    unsigned long long a, b, c;
6 | arg24_t;
8 typedef struct {
9
    unsigned long long a, b;
10 } arg16_t;
11
12
  unsigned long long global_var = 100;
13
14 unsigned long long foo_8(arg8_t p1, arg8_t p2, arg8_t p3, arg8_t
      p4, arg8_t p5, arg8_t p6, arg8_t p7, arg8_t p8)
15 {
16
    return p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + global_var;
17 }
18
19 unsigned long long foo_16(arg16_t p1, arg16_t p2, arg16_t p3,
      arg16_t p4, arg16_t p5, arg16_t p6, arg16_t p7, arg16_t p8)
20 {
    return p1.a + p2.b + p3.a + p4.b + p5.a + p6.b + p7.a + p8.b +
21
        global_var;
22 }
23
24
  unsigned long long foo_24(arg24_t p1, arg24_t p2, arg24_t p3,
      arg24_t p4, arg24_t p5, arg24_t p6, arg24_t p7, arg24_t p8)
25 {
    return p1.a + p2.b + p3.c + p4.a + p5.b + p6.c + p7.a + p8.b +
26
         global_var;
27 }
28
29 int main()
30 | {
31
    arg8_t arg8 = 10;
32
    arg16_t arg16 = {20, 30};
33
     arg24_t arg24 = {40, 50, 60};
34
    unsigned long long ret1, ret2, ret3;
35
36
    ret1 = foo_8(arg8, arg8, arg8, arg8, arg8, arg8, arg8, arg8);
37
    ret2 = foo_16(arg16, arg16, arg16, arg16, arg16, arg16,
        arg16);
38
    ret3 = foo_24(arg24, arg24, arg24, arg24, arg24, arg24, arg24,
        arg24);
39
40
    return (int)(ret1 + ret2 + ret3);
41 }
```

Listing 1: An example to understand the calling convention of gcc.

```
1    .file "hw1.c"
2    .globl global_var
3    .data
4    .align 8
5    .type global_var, @object
6    .size global_var, 8
```

```
7| global_var:
8
     .quad 100
9
     .text
10
     .globl foo_8
     .type foo_8, @function
11
12 foo_8:
13 .LFB0:
14
     .cfi_startproc
15
     pushq %rbp
16
     .cfi_def_cfa_offset 16
17
     .cfi_offset 6, -16
     movq %rsp, %rbp
18
     .cfi_def_cfa_register 6
19
     movq %rdi, -8(%rbp)
20
     movq %rsi, -16(%rbp)
movq %rdx, -24(%rbp)
movq %rcx, -32(%rbp)
21
22
     movq
23
24
     movq %r8, -40(%rbp)
25
     movq %r9, -48(%rbp)
     movq
           -16(%rbp), %rax
-8(%rbp), %rdx
26
27
     movq
     addq %rax, %rdx
28
29
     movq
            -24(%rbp), %rax
30
     addq %rax, %rdx
     movq
31
            -32(%rbp), %rax
32
     addq
           %rax, %rdx
33
            -40(%rbp), %rax
     movq
34
     addq %rax, %rdx
35
     movq
            -48(%rbp), %rax
36
            %rax, %rdx
     addq
37
     movq
            16(%rbp), %rax
     addq %rax, %rdx
38
39
     movq 24(%rbp), %rax
40
     addq %rax, %rdx
     movq global_var(%rip), %rax addq %rdx, %rax popq %rbp
41
42
43
44
     .cfi_def_cfa 7, 8
45
     ret
46
     .cfi_endproc
47
   .LFE0:
48
     .size foo_8, .-foo_8
49
     .globl foo_16
50
     .type foo_16, @function
51 foo_16:
52 .LFB1:
53
    .cfi_startproc
54
     pushq %rbp
55
     .cfi_def_cfa_offset 16
56
     .cfi_offset 6, -16
57
     movq %rsp, %rbp
58
     .cfi_def_cfa_register 6
     movq %rdi, %rax
movq %rsi, %r10
movq %rax, %rsi
59
60
61
     movq %rdx, %rdi
62
     movq %r10, %rdi
63
```

```
movq %rsi, -16(%rbp)
movq %rdi, -8(%rbp)
movq %rdx, -32(%rbp)
 64
 65
 66
      movq %rcx, -24(%rbp)
 67
     movq %r8, -48(%rbp)
movq %r9, -40(%rbp)
 68
 69
            -16(%rbp), %rdx
 70
      movq
 71
      movq
            -24(%rbp), %rax
 72
      addq %rax, %rdx
 73
      movq
            -48(%rbp), %rax
 74
            %rax, %rdx
      addq
 75
            24(%rbp), %rax
      {\tt movq}
 76
      addq
            %rax, %rdx
 77
      movq 32(%rbp), %rax
 78
      addq %rax, %rdx
 79
      movq
            56(%rbp), %rax
      addq
            %rax, %rdx
 80
 81
      movq 64(%rbp), %rax
 82
      addq %rax, %rdx
      movq
 83
            88(%rbp), %rax
 84
      addq
            %rax, %rdx
            global_var(%rip), %rax
 85
      movq
 86
      addq %rdx, %rax
 87
      popq %rbp
 88
      .cfi_def_cfa 7, 8
89
      ret
90
      .cfi_endproc
 91
    .LFE1:
92
      .size foo_16, .-foo_16
93
      .globl foo_24
94
      .type foo_24, @function
95 foo_24:
96 .LFB2:
97
     .cfi_startproc
98
      pushq %rbp
99
      .cfi_def_cfa_offset 16
      .cfi_offset 6, -16
100
101
      movq %rsp, %rbp
      .cfi_def_cfa_register 6
102
      movq 16(%rbp), %rdx
movq 48(%rbp), %rax
103
104
      addq %rax, %rdx
105
106
      movq 80(%rbp), %rax
      addq %rax, %rdx
107
108
      movq
            88(%rbp), %rax
      addq
109
            %rax, %rdx
110
      movq 120(%rbp), %rax
111
      addq %rax, %rdx
      movq
112
            152(%rbp), %rax
            %rax, %rdx
113
      addq
            160(%rbp), %rax
114
      movq
115
      addq %rax, %rdx
116
      movq
            192(%rbp), %rax
117
      addq
            %rax, %rdx
            global_var(%rip), %rax
118
      movq
      addq
            %rdx, %rax
119
      popq %rbp
120
```

```
121
      .cfi_def_cfa 7, 8
122
      ret
      .cfi_endproc
123
124
    .LFE2:
      .size foo_24, .-foo_24
125
126
      .globl main
127
      .type main, @function
128 main:
129 .LFB3:
130
      .cfi_startproc
131
      pushq %rbp
132
      \tt .cfi\_def\_cfa\_offset \ 16
133
      .cfi_offset 6, -16
134
      movq %rsp, %rbp
      .cfi_def_cfa_register 6
135
136
      pushq %rbx
137
      subq $280, %rsp
138
      .cfi_offset 3, -24
      movq $10, -96(%rbp)
movq $20, -64(%rbp)
movq $30, -56(%rbp)
139
140
141
             $40, -48(%rbp)
142
      movq
143
      movq
             $50, -40(\%rbp)
             $60, -32(%rbp)
144
      movq
      movq
             -96(%rbp), %r9
145
146
      movq
             -96(%rbp), %r8
147
             -96(%rbp), %rcx
      movq
148
      movq
             -96(%rbp), %rdx
149
      movq
             -96(%rbp), %rsi
             -96(%rbp), %rax
-96(%rbp), %rdi
150
      movq
151
      movq
152
      movq
             %rdi, 8(%rsp)
153
      movq
             -96(%rbp), %rdi
             %rdi, (%rsp)
%rax, %rdi
154
      movq
155
      movq
156
      call
             foo_8
             %rax, -88(%rbp)
157
      movq
             -64(%rbp), %rsi
-56(%rbp), %rdi
158
      movq
      movq
159
             -64(%rbp), %rcx
-56(%rbp), %rbx
160
      movq
161
      movq
162
      movq
             -64(%rbp), %r11
163
             -56(%rbp), %r10
      movq
             -64(%rbp), %rax
164
      movq
165
      movq
              -56(%rbp), %rdx
166
             %rax, 64(%rsp)
      movq
167
             %rdx, 72(%rsp)
      movq
168
             -64(%rbp), %rax
      movq
              -56(%rbp), %rdx
169
      movq
170
      movq
             %rax, 48(%rsp)
             %rdx, 56(%rsp)
171
      movq
172
              -64(%rbp), %rax
      movq
             -56(%rbp), %rdx
173
      movq
             %rax, 32(%rsp)
%rdx, 40(%rsp)
174
      movq
175
      movq
      movq
             -64(%rbp), %rax
176
177
      movq
             -56(%rbp), %rdx
```

```
%rax, 16(%rsp)
%rdx, 24(%rsp)
178
      movq
179
      movq
             -64(%rbp), %rax
180
      movq
181
      movq
             -56(%rbp), %rdx
             %rax, (%rsp)
%rdx, 8(%rsp)
%rsi, %r8
182
      movq
183
      movq
184
      movq
185
      movq
             %rdi, %r9
186
      movq
             %rcx, %rdx
             %rbx, %rcx
%r11, %rdi
%r10, %rsi
187
      movq
188
      movq
189
      {\tt movq}
190
      call
             foo_16
191
             %rax, -80(%rbp)
      movq
192
      movq
             -48(%rbp), %rax
193
      movq
             %rax, 168(%rsp)
194
      movq
             -40(%rbp), %rax
195
             %rax, 176(%rsp)
      movq
196
      movq
             -32(%rbp), %rax
197
      movq
             %rax, 184(%rsp)
198
      movq
             -48(%rbp), %rax
199
             %rax, 144(%rsp)
      movq
200
      movq
             -40(%rbp), %rax
201
      movq
             %rax, 152(%rsp)
202
      movq
             -32(%rbp), %rax
203
      movq
             %rax, 160(%rsp)
204
      movq
             -48(%rbp), %rax
205
             %rax, 120(%rsp)
      movq
206
             -40(%rbp), %rax
      movq
207
      movq
             %rax, 128(%rsp)
208
      movq
             -32(\%rbp), \%rax
209
      movq
             %rax, 136(%rsp)
210
      movq
             -48(%rbp), %rax
211
             %rax, 96(%rsp)
      movq
212
             -40(%rbp), %rax
      movq
213
      movq
             %rax, 104(%rsp)
214
      movq
             -32(\%rbp), \%rax
215
      movq
             %rax, 112(%rsp)
      movq
             -48(%rbp), %rax
216
217
      movq
             %rax, 72(%rsp)
218
             -40(%rbp), %rax
      movq
219
      movq
             %rax, 80(%rsp)
             -32(%rbp), %rax
220
      movq
             %rax, 88(%rsp)
221
      movq
222
      movq
             -48(%rbp), %rax
             %rax, 48(%rsp)
223
      movq
224
             -40(%rbp), %rax
      movq
225
             %rax, 56(%rsp)
      movq
226
      movq
             -32(\%rbp), \%rax
227
      movq
             %rax, 64(%rsp)
             -48(%rbp), %rax
228
      movq
229
             %rax, 24(%rsp)
      movq
230
             -40(%rbp), %rax
      movq
231
      movq
             %rax, 32(%rsp)
232
      movq
             -32(\%rbp), \%rax
      movq
233
             %rax, 40(%rsp)
234
      movq
             -48(%rbp), %rax
```

```
235
            %rax, (%rsp)
      movq
236
      movq
             -40(%rbp), %rax
237
      movq
            %rax, 8(%rsp)
             -32(%rbp), %rax
238
      movq
            %rax, 16(%rsp)
239
      {\tt movq}
240
      call
             foo_24
            %rax, -72(%rbp)
241
      movq
      {\tt movq}
242
             -88(%rbp), %rax
            %eax, %edx
243
      movl
244
             -80(%rbp), %rax
      {\tt movq}
            %eax, %edx
-72(%rbp), %rax
245
      addl
246
      {\tt movq}
247
      addl
            %edx, %eax
248
      addq $280, %rsp
249
            %rbx
      popq
250
      popq
            %rbp
251
      .cfi_def_cfa 7, 8
252
      ret
253
      .cfi_endproc
254
    .LFE3:
255
      .size main, .-main
      .ident "GCC: (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4"
256
257
      .section .note.GNU-stack,"",@progbits
```

Listing 2: Assembly output by gcc.

How to submit.

Please handle your hand-written answer sheets to the instructor before the lecture begins.