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.

```c

typedef unsigned long long arg8_t;

typedef struct {
  unsigned long long a, b, c;
} arg24_t;

typedef struct {
  unsigned long long a, b;
} arg16_t;

unsigned long long global_var = 100;

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)
{
  return p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + global_var;
}

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)
{
  return p1.a + p2.b + p3.a + p4.b + p5.a + p6.b + p7.a + p8.b +
      global_var;
}

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)
{
  return p1.a + p2.b + p3.c + p4.a + p5.b + p6.c + p7.a + p8.b +
      global_var;
}

int main()
{
  arg8_t arg8 = 10;
  arg16_t arg16 = {20, 30};
  arg24_t arg24 = {40, 50, 60};
  unsigned long long ret1, ret2, ret3;

  ret1 = foo_8(arg8, arg8, arg8, arg8, arg8, arg8, arg8, arg8);
  ret2 = foo_16(arg16, arg16, arg16, arg16, arg16, arg16, arg16,
      arg16);
  ret3 = foo_24(arg24, arg24, arg24, arg24, arg24, arg24, arg24,
      arg24);

  return (int)(ret1 + ret2 + ret3);
}
```

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

```
  .file "hw1.c"
  .globl  global_var
  .data
  .align 8
  .type global_var, @object
  .size global_var, 8
```

```
global_var:
  .quad 100
  .text
  .globl  foo_8
  .type foo_8, @function
foo_8:
.LFB0:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  movq  %rdi, -8(%rbp)
  movq  %rsi, -16(%rbp)
  movq  %rdx, -24(%rbp)
  movq  %rcx, -32(%rbp)
  movq  %r8, -40(%rbp)
  movq  %r9, -48(%rbp)
  movq  -16(%rbp), %rax
  movq  -8(%rbp), %rdx
  addq  %rax, %rdx
  movq  -24(%rbp), %rax
  addq  %rax, %rdx
  movq  -32(%rbp), %rax
  addq  %rax, %rdx
  movq  -40(%rbp), %rax
  addq  %rax, %rdx
  movq  -48(%rbp), %rax
  addq  %rax, %rdx
  movq  16(%rbp), %rax
  addq  %rax, %rdx
  movq  24(%rbp), %rax
  addq  %rax, %rdx
  movq  global_var(%rip), %rax
  addq  %rdx, %rax
  popq  %rbp
  .cfi_def_cfa 7, 8
  ret
  .cfi_endproc
.LFE0:
  .size foo_8, .-foo_8
  .globl  foo_16
  .type foo_16, @function
foo_16:
.LFB1:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  movq  %rdi, %rax
  movq  %rsi, %r10
  movq  %rax, %rsi
  movq  %rdx, %rdi
  movq  %r10, %rdi
```

```
  movq  %rsi, -16(%rbp)
  movq  %rdi, -8(%rbp)
  movq  %rdx, -32(%rbp)
  movq  %rcx, -24(%rbp)
  movq  %r8, -48(%rbp)
  movq  %r9, -40(%rbp)
  movq  -16(%rbp), %rdx
  movq  -24(%rbp), %rax
  addq  %rax, %rdx
  movq  -48(%rbp), %rax
  addq  %rax, %rdx
  movq  24(%rbp), %rax
  addq  %rax, %rdx
  movq  32(%rbp), %rax
  addq  %rax, %rdx
  movq  56(%rbp), %rax
  addq  %rax, %rdx
  movq  64(%rbp), %rax
  addq  %rax, %rdx
  movq  88(%rbp), %rax
  addq  %rax, %rdx
  movq  global_var(%rip), %rax
  addq  %rdx, %rax
  popq  %rbp
  .cfi_def_cfa 7, 8
  ret
  .cfi_endproc
.LFE1:
  .size foo_16, .-foo_16
  .globl  foo_24
  .type foo_24, @function
foo_24:
.LFB2:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  movq  16(%rbp), %rdx
  movq  48(%rbp), %rax
  addq  %rax, %rdx
  movq  80(%rbp), %rax
  addq  %rax, %rdx
  movq  88(%rbp), %rax
  addq  %rax, %rdx
  movq  120(%rbp), %rax
  addq  %rax, %rdx
  movq  152(%rbp), %rax
  addq  %rax, %rdx
  movq  160(%rbp), %rax
  addq  %rax, %rdx
  movq  192(%rbp), %rax
  addq  %rax, %rdx
  movq  global_var(%rip), %rax
  addq  %rdx, %rax
  popq  %rbp
```

```
  .cfi_def_cfa 7, 8
  ret
  .cfi_endproc
.LFE2:
  .size foo_24, .-foo_24
  .globl  main
  .type main, @function
main:
.LFB3:
  .cfi_startproc
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  pushq %rbx
  subq  $280, %rsp
  .cfi_offset 3, -24
  movq  $10, -96(%rbp)
  movq  $20, -64(%rbp)
  movq  $30, -56(%rbp)
  movq  $40, -48(%rbp)
  movq  $50, -40(%rbp)
  movq  $60, -32(%rbp)
  movq  -96(%rbp), %r9
  movq  -96(%rbp), %r8
  movq  -96(%rbp), %rcx
  movq  -96(%rbp), %rdx
  movq  -96(%rbp), %rsi
  movq  -96(%rbp), %rax
  movq  -96(%rbp), %rdi
  movq  %rdi, 8(%rsp)
  movq  -96(%rbp), %rdi
  movq  %rdi, (%rsp)
  movq  %rax, %rdi
  call  foo_8
  movq  %rax, -88(%rbp)
  movq  -64(%rbp), %rsi
  movq  -56(%rbp), %rdi
  movq  -64(%rbp), %rcx
  movq  -56(%rbp), %rbx
  movq  -64(%rbp), %r11
  movq  -56(%rbp), %r10
  movq  -64(%rbp), %rax
  movq  -56(%rbp), %rdx
  movq  %rax, 64(%rsp)
  movq  %rdx, 72(%rsp)
  movq  -64(%rbp), %rax
  movq  -56(%rbp), %rdx
  movq  %rax, 48(%rsp)
  movq  %rdx, 56(%rsp)
  movq  -64(%rbp), %rax
  movq  -56(%rbp), %rdx
  movq  %rax, 32(%rsp)
  movq  %rdx, 40(%rsp)
  movq  -64(%rbp), %rax
  movq  -56(%rbp), %rdx
```

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

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