Program Stack Layout on AMD64 and AArch64

Dr. Michael Marsh

November 8, 2024

1 Introduction

For several decades, the dominant personal computer architecture was based on Intel's x86 chips. In the 64-bit era, the most common of these is AMD64, also referred to as x86_64. Starting late 2020, Apple began introducing ARM-architecture chips in many of its popular personal computers. While the ARM architecture was not new, it had fallen out of favor for personal computers, and was mostly used in more resource-constrained environments (mobile devices, Raspberry Pis, etc.). The 64-bit ARM architecture used by Apple's chips is AArch64, also referred to as ARM64.

Because they are different architectures, AMD64 and AArch64 have different instruction sets, different registers, and different stack structures. Here we focus on the last two, particularly the stack layouts.

On AMD64, the instruction pointer is `rip`, and a copy of the caller's instruction pointer is saved in the callee's stack. The same is done for the caller's base pointer `rbp`.

On AArch64, the instruction pointer is `pc`, and the caller's instruction pointer is stored in the register `x30`. The caller's base pointer is stored in the register `x29`. As long as the callee does not call another function, these registers do not need to be stored on the stack. Once the callee calls another function, the caller's stored pointers are then saved on the stack.

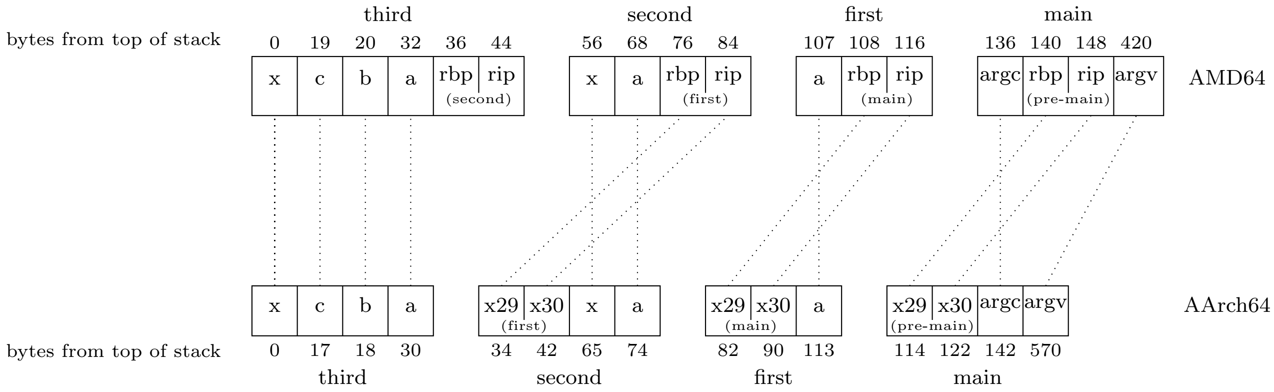
Thanks to Prof. Dave Levin for suggestions on improving this document.

2 Example Program

We will consider this simple program:

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>

uint64_t third(uint16_t x) {
  uint32_t a;
  uint64_t b;
  unsigned char c;

  a = x << 1;
  b = x << 10;
  c = x % 256;

  return a+b+c;
}

uint32_t second(uint8_t x) {
  uint64_t a;
  a = third(x << 4);
  return a >> 16;
}

void first() {
  uint8_t a;
  a = 0xf4; // 244
  printf("%x\n",second(a));
}

int main(int argc, char** argv) {
  first();
  return 0;
}
```

3 Comparing Stacks



If we look at the layouts above, we see that the stack layouts are similar, at least as far as the local variables and function arguments are concerned. They do not line up exactly, however, as shown in the number of bytes from the top of the stack at which the variable begins.

An interesting feature is that saved registers are stored at similar locations on the stack, but they are for *different* functions. For example, we see saved instruction pointers 44 bytes from the top of the stack on the AMD64 version of our program and 42 bytes from the top of the stack on the AArch64 version. However, the former is the instruction pointer for `second`, while the latter is the instruction pointer for `first`.

As mentioned in Section 1, this is because on AArch64 the saved registers `sp` and `pc` from the caller are stored in registers `x29` and `x30`, rather than on the stack. As long as a function does not call anything else, all of the information required on return is held in registers rather than on the stack. That is why we never see `second`'s saved registers on the stack. This makes certain security vulnerabilities, such as buffer overflows, harder to exploit.

Once a function (eg, `second`) calls another function (eg, `third`), the `x29` and `x30` registers will need to hold `second`'s frame pointer and program counter. That means the current values of `x29` and `x30` (`first`'s frame pointer and program counter) need to be written to the stack, so they can be restored later. From a buffer overflow perspective, if we were to overflow a buffer in `third` (which is not possible in our example), we would not be able to overwrite the next instruction to execute after `third` returns (as in AMD64). We *would*, however, be able to overwrite the next instruction to execute after `second` returns.

A gdb Session on AMD64

```
(gdb) where
#0  third (x=3904) at layout.c:12
#1  0x00000000004018a5 in second (x=244 ’\364’) at layout.c:20
#2  0x00000000004018ce in first () at layout.c:27
#3  0x0000000000401904 in main (argc=1, argv=0x7fffffffe5e8) at layout.c:31
(gdb) info frame
Stack level 0, frame at 0x7fffffffe478:
 rip = 0x40185c in third (layout.c:12); saved rip = 0x4018a5
 called by frame at 0x7fffffffe4a0
 source language c.
 Arglist at 0x7fffffffe468, args: x=3904
 Locals at 0x7fffffffe468, Previous frame’s sp is 0x7fffffffe478
 Saved registers:
  rbp at 0x7fffffffe468, rip at 0x7fffffffe470
(gdb) info locals
a = 7808
b = 12624466609913245184
c = 0 ’\000’
(gdb) x &a
0x7fffffffe464: 0x00001e80
(gdb) x &b
0x7fffffffe458: 0x92a8ae00
(gdb) x &c
0x7fffffffe457: 0xa8ae0000
(gdb) info args
x = 3904
(gdb) x &x
0x7fffffffe444: 0x00000f40

(gdb) up
#1  0x00000000004018a5 in second (x=244 ’\364’) at layout.c:20
20   a = third(x << 4);
(gdb) info frame
Stack level 1, frame at 0x7fffffffe4a0:
 rip = 0x4018a5 in second (layout.c:20); saved rip = 0x4018ce
 called by frame at 0x7fffffffe4c0, caller of frame at 0x7fffffffe478
 source language c.
 Arglist at 0x7fffffffe490, args: x=244 ’\364’
 Locals at 0x7fffffffe490, Previous frame’s sp is 0x7fffffffe4a0
 Saved registers:
  rbp at 0x7fffffffe490, rip at 0x7fffffffe498
(gdb) info locals
a = 4919712
(gdb) x &a
0x7fffffffe488: 0x004b11a0
(gdb) info args
x = 244 ’\364’
(gdb) x &x
0x7fffffffe47c: 0x00007ff4
```

```
(gdb) up
#2  0x00000000004018ce in first () at layout.c:27
27   printf("%x\n",second(a));
(gdb) info frame
Stack level 2, frame at 0x7fffffffe4c0:
 rip = 0x4018ce in first (layout.c:27); saved rip = 0x401904
 called by frame at 0x7fffffffe4e0, caller of frame at 0x7fffffffe4a0
 source language c.
 Arglist at 0x7fffffffe4b0, args:
 Locals at 0x7fffffffe4b0, Previous frame's sp is 0x7fffffffe4c0
 Saved registers:
  rbp at 0x7fffffffe4b0, rip at 0x7fffffffe4b8
(gdb) info locals
a = 244 '\364'
(gdb) x &a
0x7fffffffe4af: 0xffe4d0f4
(gdb) info args
No arguments.

(gdb) up
#3  0x0000000000401904 in main (argc=1, argv=0x7fffffffe5e8) at layout.c:31
31   first();
(gdb) info frame
Stack level 3, frame at 0x7fffffffe4e0:
 rip = 0x401904 in main (layout.c:31); saved rip = 0x401e38
 caller of frame at 0x7fffffffe4c0
 source language c.
 Arglist at 0x7fffffffe4d0, args: argc=1, argv=0x7fffffffe5e8
 Locals at 0x7fffffffe4d0, Previous frame's sp is 0x7fffffffe4e0
 Saved registers:
  rbp at 0x7fffffffe4d0, rip at 0x7fffffffe4d8
(gdb) info locals
No locals.
(gdb) info args
argc = 1
argv = 0x7fffffffe5e8
(gdb) x &argc
0x7fffffffe4cc: 0x00000001
(gdb) x argv
0x7fffffffe5e8: 0xffffe7d4
```

B gdb Session on AArch64

```
(gdb) where
#0  third (x=3904) at layout.c:11
#1  0x000000000040080c in second (x=244 ’\364’) at layout.c:20
#2  0x0000000000400838 in first () at layout.c:27
#3  0x0000000000400868 in main (argc=1, argv=0xfffffffff5e8) at layout.c:31
(gdb) info frame
Stack level 0, frame at 0xfffffffff3d0:
 pc = 0x4007ac in third (layout.c:11); saved pc = 0x40080c
 called by frame at 0xfffffffff400
 source language c.
 Arglist at 0xfffffffff3a0, args: x=3904
 Locals at 0xfffffffff3a0, Previous frame’s sp is 0xfffffffff3d0
(gdb) info locals
a = 0
b = 281474976707648
c = 0 ’\000’
(gdb) x &a
0xfffffffff3cc: 0x00000000
(gdb) x &b
0xfffffffff3c0: 0xfffff440
(gdb) x &c
0xfffffffff3bf: 0xfff44000
(gdb) info args
x = 3904
(gdb) x &x
0xfffffffff3ae: 0xf3c00f40

(gdb) up
#1  0x000000000040080c in second (x=244 ’\364’) at layout.c:20
20   a = third(x << 4);
(gdb) info frame
Stack level 1, frame at 0xfffffffff400:
 pc = 0x40080c in second (layout.c:20); saved pc = 0x400838
 called by frame at 0xfffffffff420, caller of frame at 0xfffffffff3d0
 source language c.
 Arglist at 0xfffffffff3d0, args: x=244 ’\364’
 Locals at 0xfffffffff3d0, Previous frame’s sp is 0xfffffffff400
 Saved registers:
  x29 at 0xfffffffff3d0, x30 at 0xfffffffff3d8
(gdb) info locals
a = 4790600
(gdb) x &a
0xfffffffff3f8: 0x00491948
(gdb) info args
x = 244 ’\364’
(gdb) x &x
0xfffffffff3ef: 0x497770f4
```

```
(gdb) up
#2  0x0000000000400838 in first () at layout.c:27
27   printf("%x\n",second(a));
(gdb) info frame
Stack level 2, frame at 0xfffffffff420:
 pc = 0x400838 in first (layout.c:27); saved pc = 0x400868
 called by frame at 0xfffffffff440, caller of frame at 0xfffffffff400
 source language c.
 Arglist at 0xfffffffff400, args:
 Locals at 0xfffffffff400, Previous frame's sp is 0xfffffffff420
 Saved registers:
  x29 at 0xfffffffff400, x30 at 0xfffffffff408
(gdb) info locals
a = 244 '\364'
(gdb) x &a
0xfffffffff41f: 0xfff440f4
(gdb) info args
No arguments.

(gdb) up
#3  0x0000000000400868 in main (argc=1, argv=0xfffffffff5e8) at layout.c:31
31   first();
(gdb) info frame
Stack level 3, frame at 0xfffffffff440:
 pc = 0x400868 in main (layout.c:31); saved pc = 0x400928
 caller of frame at 0xfffffffff420
 source language c.
 Arglist at 0xfffffffff420, args: argc=1, argv=0xfffffffff5e8
 Locals at 0xfffffffff420, Previous frame's sp is 0xfffffffff440
 Saved registers:
  x29 at 0xfffffffff420, x30 at 0xfffffffff428
(gdb) info locals
No locals.
(gdb) info args
argc = 1
argv = 0xfffffffff5e8
(gdb) x &argc
0xfffffffff43c: 0x00000001
(gdb) x argv
0xfffffffff5e8: 0xfffff7d5
```