Example of instruction-level program execution

This document illustrates how the stack memory and related registers evolve as a program is executed.

The example is based on the following program:

```
 1 #include <stdio.h>
 2
 3 void greet_visitor() {
 4     int gt_top = 0xAAAAAAAA;
 5     char name[10]; // BAD!!!
 6     printf("Hello.  What is your name?\n");
 7     gets(name);    // VERY, VERY, VERY BAD!!!
 8     printf("Hello, %s.", name);
 9     int gt_btm = 0xAABBBBAA;
10 }
11
12 void scare_visitor() {
13     int sv_top = 0xAACCCCAA;
14     char message[10] = "BRAH!!!\n"; // OKAY
15     printf(message);
16     int sv_btm = 0xAADDDDAA;
17 }
18
19 int main(int argc, char *argv[]) {
20     int mn_top = 0xAAEEEEAA;
21     greet_visitor();
22     int mn_btm = 0xAAFFFFAA;
23     return 0;
24 }
```

The scare_visitor() function is not called, and is thus not illustrated in this example. It is included only due to its context with the current ECE 264 homework assignment.

Legend

0xd8 0xe0	local variables or arguments
~~0x00 0x00~~	uninitialized
0x70 0x04	other or unknown purpose
0x38 0x00	address in text segment
0x00 0x00	placeholder for an address on the stack

Note: There are some details about the way main and the load system work that I don't know. Some aspects of main(..) don't fit other things I've read about this. I will post an update to this, if I learn more.

Author: Alexander J. Quinn
Date: 12/11/2015

Update 12/12/2015 to add labels for prologue and epilogue, and add characters in the name buffer.

In main(…), just before calling greet_visitor(…)

Code

```
19      int main(int argc, char *argv[]) {
   0x00000000004005d8 <+0>:     push   %rbp
   0x00000000004005d9 <+1>:     mov    %rsp,%rbp
   0x00000000004005dc <+4>:     sub    $0x20,%rsp
   0x00000000004005e0 <+8>:     mov    %edi,-0x14(%rbp)
   0x00000000004005e3 <+11>:    mov    %rsi,-0x20(%rbp)

20              int mn_top = 0xAAEEEEAA;
   0x00000000004005e7 <+15>:    movl   $0xaaeeeeaa,-0x8(%rbp)

21              greet_visitor();
   0x00000000004005ee <+22>:    mov    $0x0,%eax
=> 0x00000000004005f3 <+27>:    callq  0x400554 <greet_visitor>

22              int mn_btm = 0xAAFFFFAA;
   0x00000000004005f8 <+32>:    movl   $0xaaffffaa,-0x4(%rbp)

23              return 0;
   0x00000000004005ff <+39>:    mov    $0x0,%eax

24      }
   0x0000000000400604 <+44>:    leaveq
   0x0000000000400605 <+45>:    retq
```

prologue of main(…)

epilogue of main(…)

Registers

Base pointer: `$rbp = 0x7fffffffdff0`

Stack pointer: `$rsp = 0x7fffffffdfd0`

Stack

0x7fffffffdfd0:	0xd8	0xe0	0xff	0xff	0xff	0x7f	0x00	0x00
	stack pointer				*argv*			
0x7fffffffdfd8:	0x70	0x04	0x40	0x00	0x01	0x00	0x00	0x00
			???				*argc*	
0x7fffffffdfe0:	0xd0	0xe0	0xff	0xff	0xff	0x7f	0x00	0x00
					???			
0x7fffffffdfe8:	0xaa	0xee	0xee	0xaa	0x00	0x00	0x00	0x00
			mn_top		*mn_btm (uninitialized)*			
0x7fffffffdff0:	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	base pointer				*saved base pointer of _start(..).*			
0x7fffffffdff8:	0x5d	0xed	0x21	0x7d	0x38	0x00	0x00	0x00
					return address, to get back from main(…) to _start(…)			

main(…)

Execution

callq 0x400554 will push the return address onto the stack. In other words, 0x4005f8 will be stored at 0x7fffffffdfc8 and the stack pointer will be decremented by 8 bytes.

After calling greet_visitor(…), before prologue

Code

```
3       void greet_visitor() {
=> 0x0000000000400554 <+0>:     push   %rbp
   0x0000000000400555 <+1>:     mov    %rsp,%rbp
   0x0000000000400558 <+4>:     sub    $0x20,%rsp

4               int gt_top = 0xAAAAAAAA;
   0x000000000040055c <+8>:     movl   $0xaaaaaaaa,-0x8(%rbp)

5               char name[10]; // BAD!!!
6               printf("Hello.  What is your name?\n");
   0x0000000000400563 <+15>:    mov    $0x400708,%edi
   0x0000000000400568 <+20>:    callq  0x400438 <puts@plt>

7               gets(name);    // VERY, VERY, VERY BAD!!!
   0x000000000040056d <+25>:    lea    -0x20(%rbp),%rax
   0x0000000000400571 <+29>:    mov    %rax,%rdi
   0x0000000000400574 <+32>:    callq  0x400458 <gets@plt>

8               printf("Hello, %s.", name);
   0x0000000000400579 <+37>:    mov    $0x400723,%eax
   0x000000000040057e <+42>:    lea    -0x20(%rbp),%rdx
   0x0000000000400582 <+46>:    mov    %rdx,%rsi
   0x0000000000400585 <+49>:    mov    %rax,%rdi
   0x0000000000400588 <+52>:    mov    $0x0,%eax
   0x000000000040058d <+57>:    callq  0x400428 <printf@plt>

9               int gt_btm = 0xAABBBBAA;
   0x0000000000400592 <+62>:    movl   $0xaabbbbaa,-0x4(%rbp)

10      }
   0x0000000000400599 <+69>:    leaveq
   0x000000000040059a <+70>:    retq
```

prologue of greet_visitor(…)

epilogue of greet_visitor(…)

Stack

```
0x7fffffffdfc8: 0xf8   0x05   0x40   0x00   0x00   0x00   0x00   0x00
stack pointer   RETURN ADDRESS, next instruction to be executed after returning to main(..)
0x7fffffffdfd0: 0xd8   0xe0   0xff   0xff   0xff   0x7f   0x00   0x00
                                    argv
0x7fffffffdfd8: 0x70   0x04   0x40   0x00   0x01   0x00   0x00   0x00
                return address to _start() (?)              argc
0x7fffffffdfe0: 0xd0   0xe0   0xff   0xff   0xff   0x7f   0x00   0x00
                        saved base pointer of _start(…)
0x7fffffffdfe8: 0xaa   0xee   0xee   0xaa   0x00   0x00   0x00   0x00
                mn_top                      mn_btm (uninitialized)
0x7fffffffdff0: 0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00
base pointer    saved base pointer of _start(..).
0x7fffffffdff8: 0x5d   0xed   0x21   0x7d   0x38   0x00   0x00   0x00
                return address, to get back from main(…) to _start(…)
```

greet_visitor(…)

main(…)

Execution

push %rbp will push the value of the base pointer onto the stack. In other words, 0x7fffffffdff0 will be stored at 0x7fffffffdfc0 and the stack pointer decremented by 8 bytes, from 0x7fffffffdfc8 to 0x7fffffffdfc0.

mov %rsp $rbp will copy stack pointer into the base pointer, changing it from 0x7fffffffdff0 to 0x7fffffffdfc0.

sub $0x20,%rsp will decrement the stack pointer by 32 bytes, from 0x7fffffffdfc0 to 0xffffffffdfa0.

After calling greet_visitor(…), after prologue

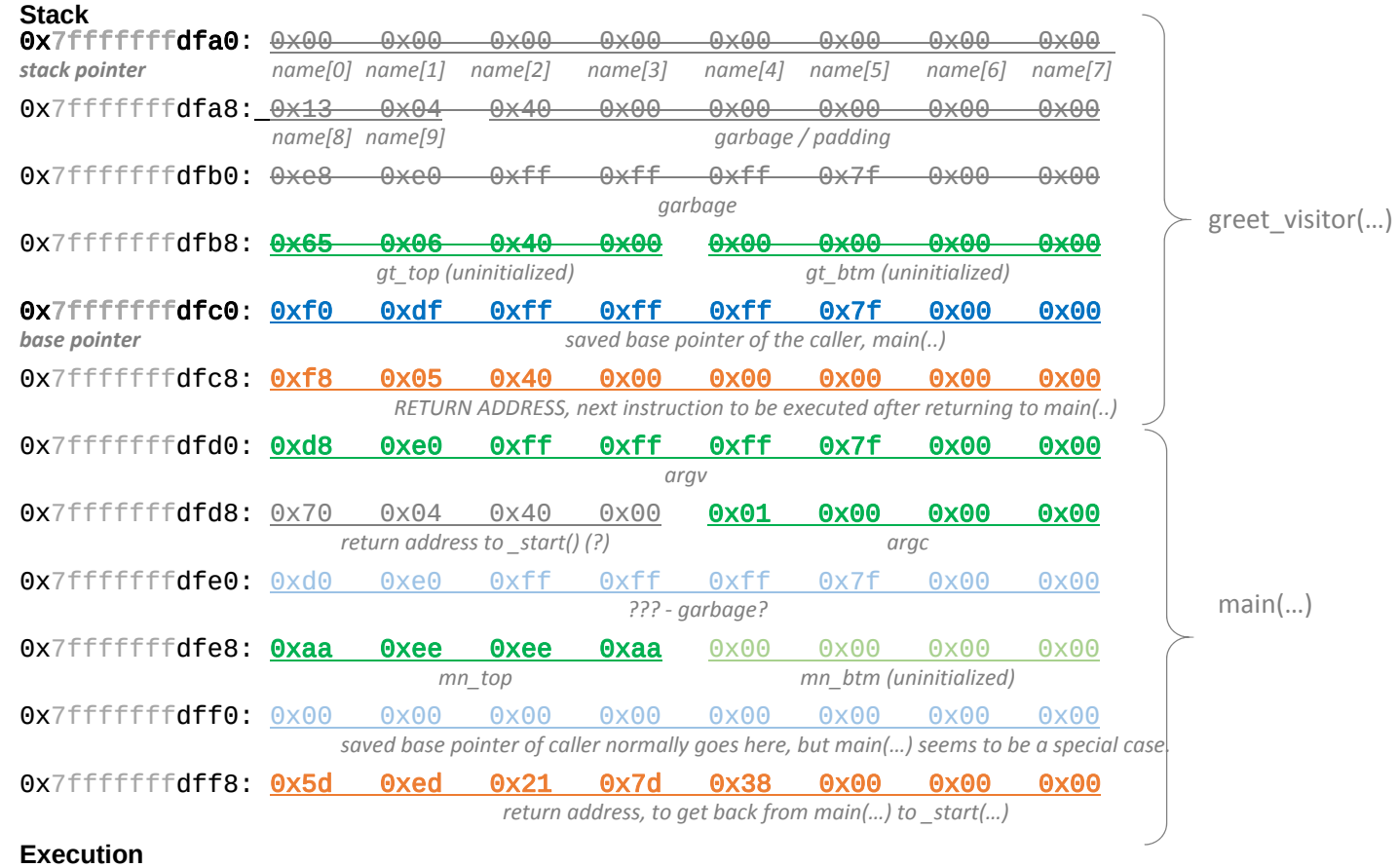
Code

```
3       void greet_visitor() {
   0x0000000000400554 <+0>:     push   %rbp
   0x0000000000400555 <+1>:     mov    %rsp,%rbp
   0x0000000000400558 <+4>:     sub    $0x20,%rsp

4               int gt_top = 0xAAAAAAAA;
=> 0x000000000040055c <+8>:     movl   $0xaaaaaaaa,-0x8(%rbp)

5               char name[10]; // BAD!!!
6               printf("Hello.  What is your name?\n");
   0x0000000000400563 <+15>:    mov    $0x400708,%edi
   0x0000000000400568 <+20>:    callq  0x400438 <puts@plt>

7               gets(name);     // VERY, VERY, VERY BAD!!!
   0x000000000040056d <+25>:    lea    -0x20(%rbp),%rax
   0x0000000000400571 <+29>:    mov    %rax,%rdi
   0x0000000000400574 <+32>:    callq  0x400458 <gets@plt>

8               printf("Hello, %s.", name);
   0x0000000000400579 <+37>:    mov    $0x400723,%eax
   0x000000000040057e <+42>:    lea    -0x20(%rbp),%rdx
   0x0000000000400582 <+46>:    mov    %rdx,%rsi
   0x0000000000400585 <+49>:    mov    %rax,%rdi
   0x0000000000400588 <+52>:    mov    $0x0,%eax
   0x000000000040058d <+57>:    callq  0x400428 <printf@plt>

9               int gt_btm = 0xAABBBBAA;
   0x0000000000400592 <+62>:    movl   $0xaabbbbaa,-0x4(%rbp)

10      }
   0x0000000000400599 <+69>:    leaveq
   0x000000000040059a <+70>:    retq
```

prologue of greet_visitor(…)

epilogue of greet_visitor(…)

Stack



Execution

We will now initialize the local variables and fill the name array.

Before returning from greet_visitor(…), before epilogue

Code

```
3       void greet_visitor() {
   0x0000000000400554 <+0>:     push   %rbp
   0x0000000000400555 <+1>:     mov    %rsp,%rbp
   0x0000000000400558 <+4>:     sub    $0x20,%rsp

4               int gt_top = 0xAAAAAAAA;
   0x000000000040055c <+8>:     movl   $0xaaaaaaaa,-0x8(%rbp)

5               char name[10]; // BAD!!!
6               printf("Hello.  What is your name?\n");
   0x0000000000400563 <+15>:    mov    $0x400708,%edi
   0x0000000000400568 <+20>:    callq  0x400438 <puts@plt>

7               gets(name);    // VERY, VERY, VERY BAD!!!
   0x000000000040056d <+25>:    lea    -0x20(%rbp),%rax
   0x0000000000400571 <+29>:    mov    %rax,%rdi
   0x0000000000400574 <+32>:    callq  0x400458 <gets@plt>

8               printf("Hello, %s.", name);
   0x0000000000400579 <+37>:    mov    $0x400723,%eax
   0x000000000040057e <+42>:    lea    -0x20(%rbp),%rdx
   0x0000000000400582 <+46>:    mov    %rdx,%rsi
   0x0000000000400585 <+49>:    mov    %rax,%rdi
   0x0000000000400588 <+52>:    mov    $0x0,%eax
   0x000000000040058d <+57>:    callq  0x400428 <printf@plt>

9               int gt_btm = 0xAABBBBAA;
   0x0000000000400592 <+62>:    movl   $0xaabbbbaa,-0x4(%rbp)

10      }
=> 0x0000000000400599 <+69>:    leaveq
   0x000000000040059a <+70>:    retq
```

prologue of greet_visitor(…)

epilogue of greet_visitor(…)

Stack

	'A'	'l'	'e'	'x'	'a'	'n'	'd'	'e'
0x7fffffffdfa0:	0x41	0x6c	0x65	0x78	0x61	0x6e	0x64	0x65
stack pointer	*name[0]*	*name[1]*	*name[2]*	*name[3]*	*name[4]*	*name[5]*	*name[6]*	*name[7]*
	'r'	'\0'						
0x7fffffffdfa8:	0x72	0x00	0x40	0x00	0x00	0x00	0x00	0x00
	name[8]	*name[9]*	*garbage / padding*					
0x7fffffffdfb0:	0xe8	0xe0	0xff	0xff	0xff	0x7f	0x00	0x00
	???							
0x7fffffffdfb8:	0xaa	0xaa	0xaa	0xaa	0xaa	0xbb	0xbb	0xaa
	gt_top				*gt_btm*			
0x7fffffffdfc0:	0xf0	0xdf	0xff	0xff	0xff	0x7f	0x00	0x00
base pointer	*saved base pointer of main(..)*							
0x7fffffffdfc8:	0xf8	0x05	0x40	0x00	0x00	0x00	0x00	0x00
	RETURN ADDRESS, next instruction to be executed after returning to main(..)							
0x7fffffffdfd0:	0xd8	0xe0	0xff	0xff	0xff	0x7f	0x00	0x00
	argv							
0x7fffffffdfd8:	0x70	0x04	0x40	0x00	0x01	0x00	0x00	0x00
	return address to _start() (?)				*argc*			
0x7fffffffdfe0:	0xd0	0xe0	0xff	0xff	0xff	0x7f	0x00	0x00
	??? - garbage?							
0x7fffffffdfe8:	0xaa	0xee	0xee	0xaa	0x00	0x00	0x00	0x00
	mn_top				*mn_btm (uninitialized)*			
0x7fffffffdff0:	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
	saved base pointer of caller normally goes here, but main(…) seems to be a special case.							
0x7fffffffdff8:	0x5d	0xed	0x21	0x7d	0x38	0x00	0x00	0x00
	return address, to get back from main(…) to _start(…)							

greet_visitor(…)

main(…)

Execution

leaveq will set the base pointer ($rbp) back to 0x7fffffffdff0 and the stack pointer ($rsp) back to 0x7fffffffdfc8. **retq** will pop the return address from the stack and set the instruction pointer ($rip) to it, so execution can continue back in main.

In main(…), just before returning

Code

```
19      int main(int argc, char *argv[]) {
   0x00000000004005d8 <+0>:     push   %rbp
   0x00000000004005d9 <+1>:     mov    %rsp,%rbp
   0x00000000004005dc <+4>:     sub    $0x20,%rsp
   0x00000000004005e0 <+8>:     mov    %edi,-0x14(%rbp)
   0x00000000004005e3 <+11>:    mov    %rsi,-0x20(%rbp)

20              int mn_top = 0xAAEEEEAA;
   0x00000000004005e7 <+15>:    movl   $0xaaeeeeaa,-0x8(%rbp)

21              greet_visitor();
   0x00000000004005ee <+22>:    mov    $0x0,%eax
   0x00000000004005f3 <+27>:    callq  0x400554 <greet_visitor>

22              int mn_btm = 0xAAFFFFAA;
   0x00000000004005f8 <+32>:    movl   $0xaaffffaa,-0x4(%rbp)

23              return 0;
   0x00000000004005ff <+39>:    mov    $0x0,%eax

24      }
=> 0x0000000000400604 <+44>:    leaveq
   0x0000000000400605 <+45>:    retq
```

prologue of main(…)

epilogue of main(…)

Stack

```
0x7fffffffdfd0: 0xd8   0xe0   0xff   0xff   0xff   0x7f   0x00   0x00
0x7fffffffdfd8: 0x70   0x04   0x40   0x00   0x01   0x00   0x00   0x00
0x7fffffffdfe0: 0xd0   0xe0   0xff   0xff   0xff   0x7f   0x00   0x00
0x7fffffffdfe8: 0xaa   0xee   0xee   0xaa   0x00   0x00   0x00   0x00
0x7fffffffdff0: 0x00   0x00   0x00   0x00   0x00   0x00   0x00   0x00
0x7fffffffdff8: 0x5d   0xed   0x21   0x7d   0x38   0x00   0x00   0x00
```

stack pointer — *argv*

return address to _start() (?) — *argc*

??? - garbage?

mn_top — *mn_btm (uninitialized)*

base pointer — *saved base pointer of caller normally goes here, but main(…) seems to be a special case.*

return address, to get back from main(…) to _start(…)

main(…)