

CSED211: Lab. 4

BombLab

조성준

allencho1222@postech.ac.kr

POSTECH

2023.10.02

What We Have Learned

- GDB
- Assembly Language
 - Opcode
 - Operand
 - Registers
 - Instructions

Table of Contents

- Assembly Language (Calling Convention)
- Memory Layout (Stack)

Assembly Language (Calling Convention)

- Calling convention answers
 - How does function **receive parameters** from the caller?
 - How does function **return a value**?

Assembly Language (Calling Convention)

- How does function **receive parameters**?
 - Some registers are **used to pass parameters**

Register	Purpose	Saved across calls
%rax	temp register; return value	No
%rbx	callee-saved	Yes
%rcx	used to pass 4th argument	No
%rdx	used to pass 3rd argument	No
%rsp	stack pointer	Yes
%rbp	callee-saved; base pointer	Yes
%rsi	used to pass 2nd argument	No
%rdi	used to pass 1st argument	No
%r8	used to pass 5th argument	No
%r9	used to pass 6th argument	No
%r10-r11	temporary	No
%r12-r15	callee-saved registers	Yes

```
int test(int a, int b, int c, int d) {  
    printf("%d %d %d %d\n", a, b, c, d);  
    return 3;  
}  
  
int main(void) {  
    int a = 0;  
    int b = 1;  
    int c = 2;  
    int d = 3;  
  
    test(a, b, c, d);  
    return 0;  
}
```

```
0x0000000000401161 <+0>:    push    %rbp  
0x0000000000401162 <+1>:    mov     %rsp,%rbp  
0x0000000000401165 <+4>:    sub     $0x10,%rsp  
0x0000000000401169 <+8>:    movl    $0x0,-0x4(%rbp)  
0x0000000000401170 <+15>:   movl    $0x1,-0x8(%rbp)  
0x0000000000401177 <+22>:   movl    $0x2,-0xc(%rbp)  
0x000000000040117c <+29>:   movl    $0x3,-0x10(%rbp)  
0x0000000000401185 <+36>:   mov     -0x10(%rbp),%ecx  
0x0000000000401188 <+39>:   mov     -0xc(%rbp),%edx  
0x000000000040118b <+42>:   mov     -0x8(%rbp),%esi  
0x000000000040118e <+45>:   mov     -0x4(%rbp),%eax  
0x0000000000401191 <+48>:   mov     %eax,%edi  
0x0000000000401193 <+50>:   call    0x401126 <test>  
0x0000000000401198 <+55>:   mov     $0x0,%eax  
0x000000000040119d <+60>:   leave  
0x000000000040119e <+61>:   ret
```

* <http://6.s081.scripts.mit.edu/sp18/x86-64-architecture-guide.html>

Assembly Language (Calling Convention)

- How does function **return a value**?
 - `rax` is **used to return a value**

Register	Purpose	Saved across calls
%rax	temp register; return value	No
%rbx	callee-saved	Yes
%rcx	used to pass 4th argument	No
%rdx	used to pass 3rd argument	No
%rsp	stack pointer	Yes
%rbp	callee-saved; base pointer	Yes
%rsi	used to pass 2nd argument	No
%rdi	used to pass 1st argument	No
%r8	used to pass 5th argument	No
%r9	used to pass 6th argument	No
%r10-r11	temporary	No
%r12-r15	callee-saved registers	Yes

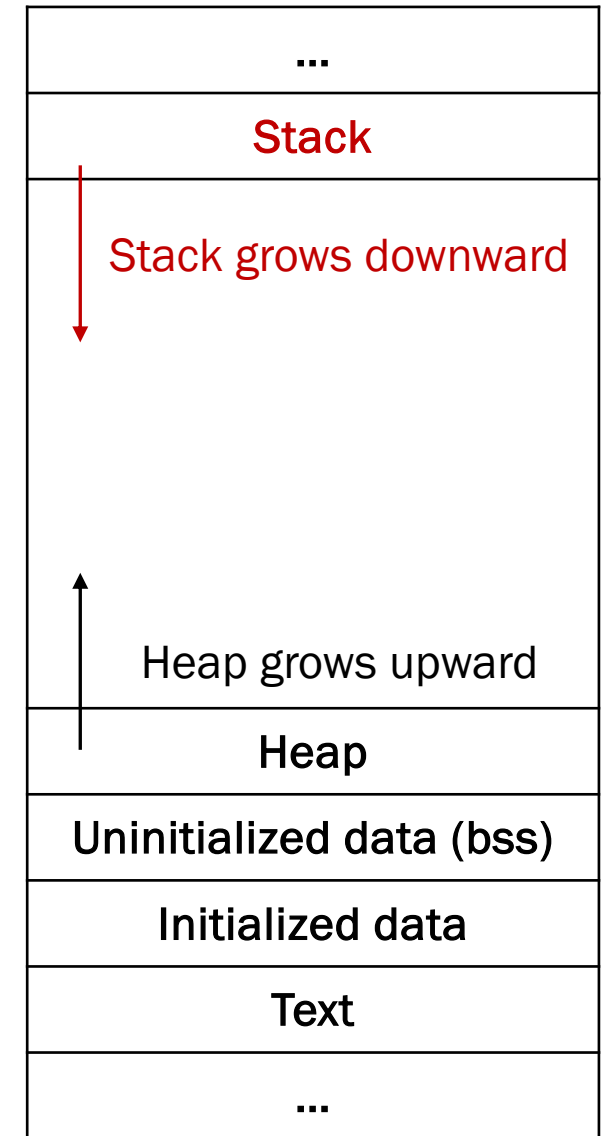
```
int test(int a, int b, int c, int d) {  
    printf("%d %d %d %d\n", a, b, c, d);  
    return 3;  
}  
  
int main(void) {  
    int a = 0;  
    int b = 1;  
    int c = 2;  
    int d = 3;  
  
    test(a, b, c, d);  
    return 0;  
}
```

```
0x0000000000401155 <+47>: call 0x401030 <printf@plt>  
0x000000000040115a <+52>: mov $0x3,%eax
```

Memory Layout (Stack)

- Each program (binary) has its own address space
 - Stack is a LIFO (Last-In-First-Out) data structure
 - Local variables are stored in stack
 - Stack can grow (downward) and shrink (upward) as a program is being executed

High address



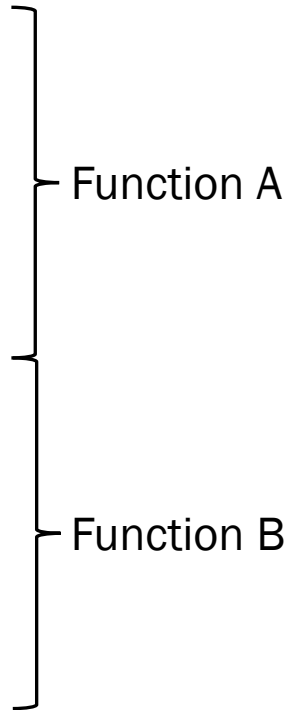
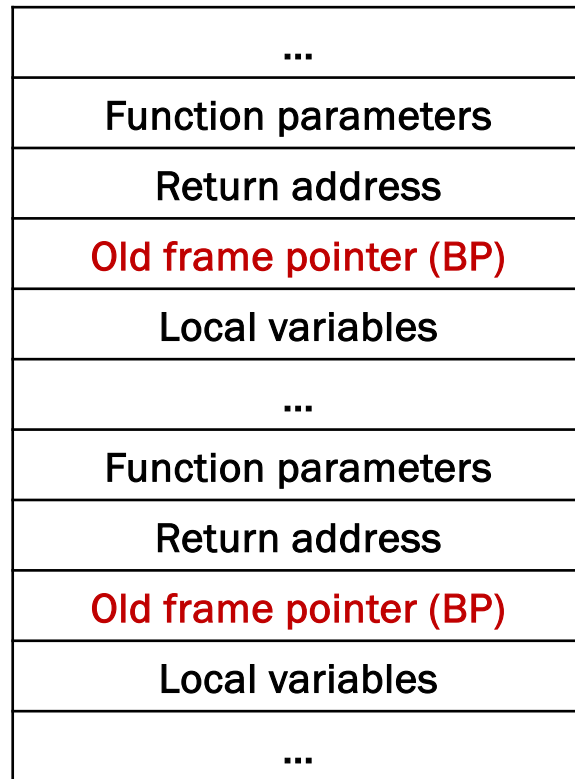
Low address

Memory Layout (Stack)

■ Stack frame

- Stack stores information **about subroutines (e.g., functions)**
- Stack is managed by
 - **Pushing (prologue)** and **popping (epilogue)** frame pointers (e.g., rbp, rsp)

High address



Function prologue:

- push rbp
- mov rsp, rbp
- sub N, rsp

Function epilogue:

- mov rbp, rsp
- pop rbp
- ret

Low address

Homework (Bomb Lab)

- Bomb server will close at
 - 10/16 (Wed) 23:59 (midnight)

Homework (Report)

- Deadline: 10/16 (Wed) 23:59 (midnight)
- You need to
 - Explain how you defuse bomb **in the report**
 - **Follow file name format, {student #}.pdf (No ZIP file)**