Assembly

Where we stand and where we go

Cauchy Pu

Outline

1. Overview
2. Why assembly
3. A simple world
4. Agreements(ABI)
5. Inline Assembly
6. How many architectures
7. Reverse Engineering
8. Questions
9. References
10. Thanks

Overview

Computers execute machine code, sequences of bytes encoding the low-level operations that manipulate data, manage memory, read and write data on storage devices, and communicate over networks.

Bits and Where

Information = Bits + Context

So we study what bits and in which we view them

Assembly!

Why assembly

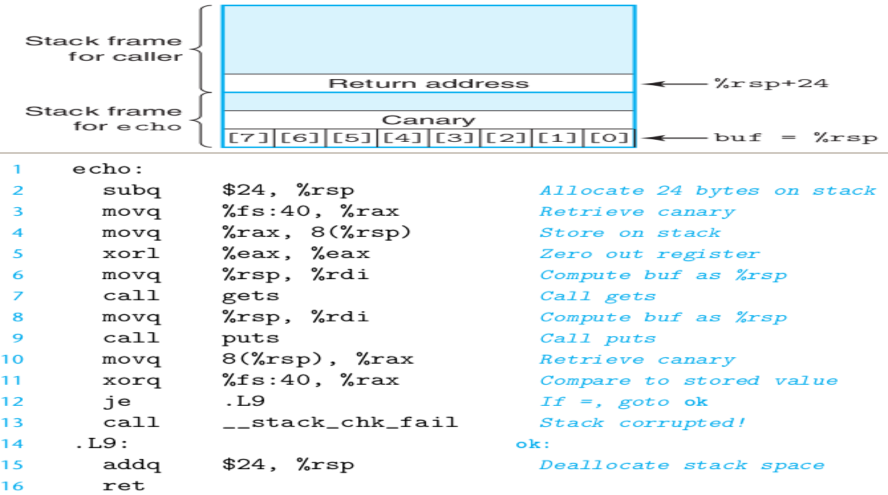
But, compilers do most of the work in generating assembly code, so why sould we spend our time?

1. Optimization. Sometimes we have to try the assembly code corresponding to various forms of upper language.
2. Some bugs more obvious in assembly. For exmaple, concurrent programs.
3. Security. Overwrite information is a common attack method.
4. Some code must be assembly, context switch.
5. When you encounter core dump, what the corresponding high level code?
6. And you told me, we are low-level engineer, right? So, here we go, assembly!
7. ......

Think like a computer

Why assembly

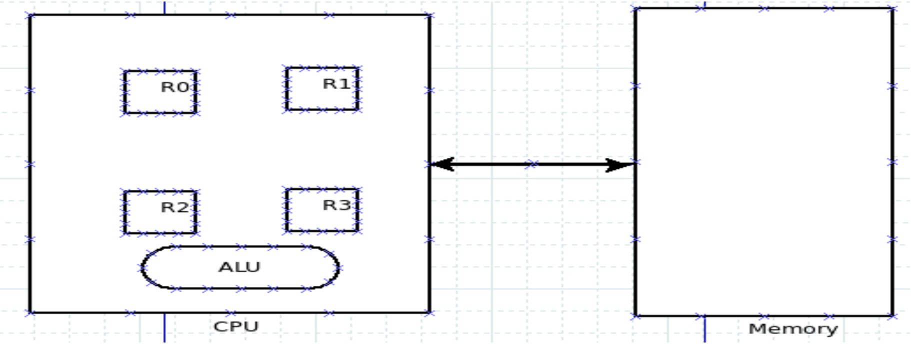
Example



```
    echo:
      subq     $24, %rsp                Allocate 24 bytes on stack
      movq     %fs:40, %rax             Retrieve canary
      movq     %rax, 8(%rsp)            Store on stack
      xorl     %eax, %eax               Zero out register
      movq     %rsp, %rdi               Compute buf as %rsp
      call     gets                     Call gets
      movq     %rsp, %rdi               Compute buf as %rsp
      call     puts                     Call puts
      movq     8(%rsp), %rax            Retrieve canary
      xorq     %fs:40, %rax             Compare to stored value
      je       .L9                      If =, goto ok
      call     __stack_chk_fail         Stack corrupted!
    .L9:                              ok:
      addq     $24, %rsp                Deallocate stack space
      ret
```

A simple world

In the world of assembly...



Simple, right?

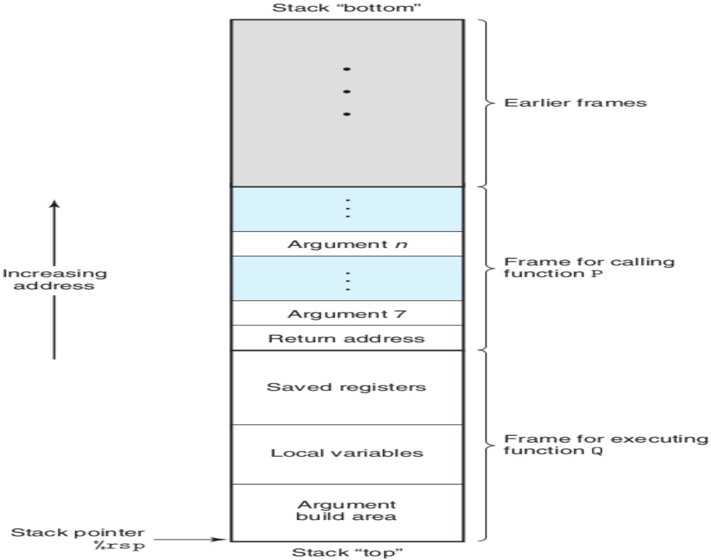
A simple world

The class of instructions:

1. assessing
    - mov
    - ldl
    - stl
2. arithmetic and logical operations
    - add
    - sub
3. control & procedures
    - jump
    - call
    - ret

A simple world

How stack operations...



A simple world

I want struct, array and many functions...

A simple world

Now some real feed...

Example

```
.section .data
data_items:
.long 3,67,34,222,45,75,54,34,44,33,22,11,66,0
.section .text
.global _start
_start:
movl $0, $edi
movl data_items(,%edi, 4), %eax
movl %eax, %ebx

start_loop:
cmpl $0, $eax
```

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax?

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax? the last item of data_items

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax? the last item of data_items
2. How to check result?

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax? the last item of data_items
2. How to check result? echo $?

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax? the last item of data_items
2. How to check result? echo $?
3. How to pass the result to next function?

A simple world

Example

```
je loop_exit
incl %edi
movl data_items(,$edi, 4), %eax
cmpl %ebx, %eax
jle start_loop
movl %eax, %ebx
jmp start_loop
loop_exit:
movl $1, %eax
int $0x80
```

1. Why cmpl $0, $eax? the last item of data_items
2. How to check result? echo $?
3. How to pass the result to next function? Euh...next section

A simple world

Some caveat

1. SP is a freak!
2. You only have a limited number of registers.
3. Frame, frame, it’s frame...
4. Alignment encounter with SP.
5. ......

Simple but frustrated...

Agreements

Inline Assembly

When you travel happily in the kernel source...

Damn!

```
__asm__ ("swp %0, %0, [%1]" : : "r"(val), "r"(addr));
```

This is inline assembly

How many architectures

Reverse Engineering

Some friends you need...

Questions

ANY
QUESTIONS?

References I

- Randal E. Bryant, David R. O’Hallaron
  Computer Systems, A prorgammer’s perspective.
  Pearson, 2018

- Jonathan Bartlett
  Programming from the Ground Up.

- John R. Levine
  Linkers & Loaders.

- Sandeep.S
  GCC-Inline-Assembly-HOWTO, 2003.
  https://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html

Thanks

Thank you ;)!

pqy7172@gmail.com