

TUTORIAL

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

C113 Architecture

Lecturer:

Dr. Jana Giceva

j.giceva@imperial.ac.uk

Head Teaching Assistant:

Izaak Coleman

ic711@imperial.ac.uk

Date: March 12, 2018

1 X86-64 Procedures and Complex Data Structures

1.1 Arrays in Assembly

Suppose the start address of a short array A and `size_t` index i are stored in registers `%rdi` and `%rsi`, respectively. For each of the following C expressions, give its type, a formula for its value, and an assembly code implementation. The result should be stored in register with name corresponding to the size of the element (e.g., `%rax` for 8 byte values). Note that each expression can be implemented with one single assembly instruction by choosing a suitable addressing mode.

Expression	Type	Formula	Assembly Code
$A+3$	short*	$x_A + 6$	<code>leaq 6(%rdi), %rax</code>
$A[5]$	short	$M[x_A + 10]$	<code>movw 10(%rdi), %ax</code>
$A[4*i+2]$	short	$M[x_A + 8*i + 4]$	<code>movw 4(%rdi,%rsi,8), %ax</code>
$A+2*i-7$	short*	$x_A + 4*i - 14$	<code>leaq -14(%rdi,%rsi,4), %rax</code>

1.2 Structs in Assembly

Take the struct below compiled on 32-bit Linux:

```
struct my_struct {
    short b;
    int x;
    short s;
    long z;
    char c[5];
    long long a;
    char q;
}
```

1. Please lay out the struct in memory below (each cell is 1 byte). Please shade in boxes used for padding.

```
+---+---+---+---+---+---+---+---+
| b | b | / | / | x | x | x | x |
+---+---+---+---+---+---+---+---+
| s | s | / | / | z | z | z | z |
+---+---+---+---+---+---+---+---+
| c | c | c | c | c | / | / | / |
+---+---+---+---+---+---+---+---+
| a | a | a | a | a | a | a | a |
+---+---+---+---+---+---+---+---+
| q | / | / | / |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

Given the following gdb interaction (where ms is a struct my_struct).

```
(gdb) x/40b      &ms
0xffffcde0:      0xbb  0x00  0x86  0x47  0xf9  0xd9  0x01  0x00
0xffffcde8:      0x6d  0x3b  0xff  0xff  0xbe  0xba  0xef  0xbe
0xffffcdf0:      0x68  0x6c  0x70  0x6d  0x65  0x00  0x00  0x00
0xffffcdf8:      0x1e  0xab  0xdf  0x1e  0xff  0xe1  0xaf  0xde
0xffffce00:      0x21  0x00  0x00  0x00  0xf4  0x7f  0x86  0x47
```

2. Label the fields above and fill in the values below:

- ms.b = 0x00bb
- ms.x = 0x0001d9f9
- ms.s = 0x3b6d
- ms.z = 0xbeefbabe
- ms.c = 0x68, 0x6c, 0x70, 0x6d, 0x65
- ms.a = 0xdeafe1ff1edfab1e
- ms.q = 0x21

3. Define a struct with the same elements that has a total size of less than 30 bytes. How many bytes is the struct that you just wrote down?

Answer:

There are many possible solutions. One option would be:

```
struct my_compressed_struct {
    long long a;
    long z;
    int x;
    short b;
    short s;
    char[5] c;
    char q;
}

+---+---+---+---+---+---+---+---+
| a | a | a | a | a | a | a | a |
+---+---+---+---+---+---+---+---+
| z | z | z | z | x | x | x | x |
+---+---+---+---+---+---+---+---+
| b | b | s | s | c | c | c | c |
+---+---+---+---+---+---+---+---+
| c | q | / | / |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

This struct has a size of 28 bytes.

1.3 Assembly to C/Java

Express the operations of the following assembly sequence as a C/Java program.

```
foo:
    testl    %edi, %edi
    js       .L3
    movl     $1, %eax
    cmpl     $1, %edi
    jg       .L9
    rep ret

.L9:
    pushq    %rbp
    pushq    %rbx
    subq     $8, %rsp
    movl     %edi, %ebx
    leal     -1(%rdi), %edi
    call     foo
    movl     %eax, %ebp
    leal     -2(%rbx), %edi
    call     foo
    addl     %ebp, %eax
    addq     $8, %rsp
    popq     %rbx
    popq     %rbp
    ret

.L3:
    movl     $-1, %eax
    ret
```

Give an example how the function `foo` can be called and provide type declarations for all parameters.

Answer:

```
int fib(int n)
{
    int return_value = 0;
    if (n < 0){
        return_value = -1;
    } else if (n <= 1){
        return_value = 1;
    } else {
        return_value = fib(n-1) + fib(n-2);
    }
    return return_value;
}
```