Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

SYSTEMS PROGRAMMING AND COMPUTER ARCHITECTURE
A little Quiz

a) The register rax currently has value 0. Which of the following statements are true?

(a) Executing movq (%rax), %rcx will cause a segmentation fault.

(b) Executing leaq (%rax), %rcx will cause a segmentation fault.

(c) Executing movq %rax, %rcx will cause a segmentation fault.

(d) Executing addq $8, %rsp will increase the stack allocation by 8 bytes.

Solution:

(a) true

(b) false

(c) false

(d) false

b) Which of the following lines of C produce the same outcome as `lea0xffffffff(%esi),%eax`? (32-bit Machine)

(a) `*(esi-1) = eax`

(b) `esi = eax + 0xffffffff`

(c) `eax = esi - 1`

(d) `eax = *(esi -1)`

Solution: c)

c) Which of the following statements are valid, which are not and why?

(a) `movl(, %eax, 4), %ebx`

(b) `movl 15, (%ebx)`

(c) `movl %eax, 655`

Solution:

(a) Valid: %ebx = 4*%eax

(b) Invalid: 15 is a memory address, not intermediate! mem $\longleftrightarrow$ mem transfers are not allowed

(c) Valid: store content of %eax to memory address 655

d) Which of the following values of `%eax` would cause the jump to be taken?

(a) 1

(b) 0

(c) Any value

(d) no value

Solution: a)

e) What does the `leave` instruction do? Write down an equivalent assembly. **Solution:**

```
mov %ebp,%esp
pop %ebp
```

f) Translate the following C Code to Assembly

```c
// input: int x (in %rdi)
// output int y (in %rax)
int func(int x) {
    int y = 0;
    if (x > 0) {
      y = 10;
    }
    y += 5;
    return y;
}
```

Using gcc 4.9.4 -O0:

```
func(int):
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -20(%rbp)
        movl    $0, -4(%rbp)
        cmpl    $0, -20(%rbp)
        jle     .L2
        movl    $10, -4(%rbp)
.L2:
        addl    $5, -4(%rbp)
        movl    -4(%rbp), %eax
        popq    %rbp
        ret
```

Using gcc 4.9.4 -O1:

```
func(int):
        testl   %edi, %edi
        movl    $10, %edx
        movl    $0, %eax
        cmovg   %edx, %eax
        addl    $5, %eax
        ret
```

g) Translate the following C Code to Assembly

```
// input: int x, int y (in %rdi, %rsi)
// output int z (in %rax)

int func(int x, int y) {
    int z = 0;
    while (z <= y) {
      z +=3*(x+1);
    }
    return z;
}
```

Using gcc 4.9.4 -O0:

```
func(int, int):
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -20(%rbp)
        movl    %esi, -24(%rbp)
        movl    $0, -4(%rbp)
        jmp     .L2
.L3:
        movl    -20(%rbp), %eax
        leal    1(%rax), %edx
        movl    %edx, %eax
        addl    %eax, %eax
        addl    %edx, %eax
        addl    %eax, -4(%rbp)
.L2:
        movl    -4(%rbp), %eax
        cmpl    -24(%rbp), %eax
        jle     .L3
        movl    -4(%rbp), %eax
        popq    %rbp
        ret
```

Using gcc 4.9.4 -O1:

```
func(int, int):
        testl   %esi, %esi
        js      .L4
        leal    3(%rdi,%rdi,2), %edx
        movl    $0, %eax
.L3:
        addl    %edx, %eax
        cmpl    %eax, %esi
        jge     .L3
        rep ret
.L4:
        movl    $0, %eax
        ret
```

h) Translate the following Assembly code to C

```
func(int, int):
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        movl    -4(%rbp), %eax
        cmpl    -8(%rbp), %eax
        jle     .L2
        movl    -4(%rbp), %eax
        jmp     .L3
.L2:
        movl    -8(%rbp), %eax
.L3:
        popq    %rbp
        ret
```

```
int func(int x, int y) {
  if (x > y) {
    return x;
  } else {
    return y;
  }
}
```