# Sample Questions

Ernesto Rivera-Alvarado and Saúl Guadamuz-Brenes

*Computer Science and Electronic Engineering*
*Costa Rica Institute of Technology*
ernestoriv7@yahoo.com - sguadamuz@itcr.ac.cr

## 1 Sample Questions

The following are sample questions of the exam that is going to be validated. For brevity, only one question per construct will be included. As stated before, the assembly language used for the questions is x86_64. The asterisk points to the correct answer. An important note is that the original questions will be validated in Spanish, so we provide the English translation here.

**Register operands and usage:**

To retrieve the lower 32 bits of the register RBX, the following register must be accessed:

a) EAX.
b) EBP.
c) EBX.*
d) R9D.
e) BX.

**Data transfer instructions:**

Identify the correct affirmation after executing the following assembly instruction `mov rax, rbx`:

a) The value of `rax` is copied into `rbx`.
b) The value of `rbx` is copied into `rax`*.
c) The values of `rax` and `rbx` are interchanged.
d) The values of `rax` and `rbx` are set to 0.
e) The values of `rax` and `rbx` are set to 1.

**Memory usage, addressing modes and operands:**

Identify the correct instruction that copies the value contained in the register `rax` into the memory address pointed by the register `rcx`.

a) `mov [rcx], rax` *
b) `mov rax, [rcx]`
c) `mov rcx, rax`
d) `mov rax, rcx`
e) `mov rax, rbx`

**Arithmetic and logic:**

Considering that the register `rax` contains the value `18`, and the register `rcx` is `33`, the value in `rax` after executing the instruction `xor rax, rcx` is:

a) 51. *
b) 0.
c) 12.
d) 48.
e) 33.

**Conditional/unconditional branch instructions:**

Consider the following assembly code and identify the actions performed by the instructions in lines 8 and 9.

```
1
2  .label:
3    mov rax, 43
4    mov rbx, 54
5    add rax,rbx
6    mov rbx, r14
7    sub rax, rbx
8    cmp rax, rbx
9    jle .label
10
11 .end:
12   mov rax, 60
13   mov rdi, 0
14   syscall
```

a) Jump to the address location `.label` if `rax` is less or equal than `rbx`. *
b) Code execution continues to line 10 if `rax` and `rbx` are equal.
c) Both instructions don't have an effect on code execution.
d) Jumps to the address location `.label` if `rbx` is less than `rax`.
e) Jumps to the address location `.label` if `rax` and `rbx` are different.

**Programming conventions:**

Consider a function that receives 2 parameters. The parameters must be stored in the following registers:

a) First parameter in `rdi`, second parameter in `rsi`. *
b) First parameter in `rdx`, second parameter in `rcx`.
c) First parameter in `rdi`, second parameter in `rcx`.
d) First parameter in `rdx`, second parameter in `rsi`.
e) First parameter in `rdx`, second parameter in `rdi`.

**Creation and usage of procedures and routines:**

The following assembly code is a routine named `function` that reads a value from memory and returns 1 if the value is equal to 100, and return 0 if it is different:

```
1  main:
2     call function
3
4  function:
5    mov rax, [memory_location]
6    mov cmp, 100
7    je .equal_100
8
9    .not_equal_100:
10     mov rax, 0
11      j .end
12
13   .equal_100:
14     mov rax, 1
15     j .end
16
17 .end:
18                    _________
```

The instruction that must in line 18 is:

a) `jmp main`
b) `ret`
c) `je main`
d) `ret main`
e) `call`

**Stack usage:**
Consider that the following starting values in the registers:

- `rax`: 32
- `rbx`: 76
- `rcx`: 43
- `rdx`: 03
- `rdi`: 23

After executing the following assembly code identify the option that has the correct value on the registers.

```
1
2  push rdi
3  push rdx
4  push rcx
5  push rbx
6  push rax
7
8  pop rdi
9  pop rdx
```

```
10  pop rcx
11  pop rbx
12  pop rax
```

| Register | i | ii | iii | iv | v |
|---|---|---|---|---|---|
| rax | 32 | 23 | 03 | 76 | 76 |
| rbx | 76 | 03 | 23 | 32 | 03 |
| rcx | 43 | 43 | 43 | 43 | 43 |
| rdx | 03 | 76 | 32 | 03 | 32 |
| rdi | 23 | 32 | 76 | 23 | 23 |

a) i
b) ii *
c) iii
d) iv
e) v

**Data types and interpretation:**

A string contained in address `sourcetext` needs to be copied to address `destinationtext`, using the following assembly code:

```
1
2  lea rax, sourcetext
3  lea rbx, destinationtext
4    .copyloop:
5
6      mov byte cl, [rax]
7      mov byte [rbx], cl
8
9      __________
10     je .end
11
12     inc rax
13     inc rbx
14     jmp .copyloop:
15
16 .end:
```

In order to copy the text string successfully, the missing instruction in line 8 is:

a) `cmp cl, 0`*
b) `inc cl`
c) `add rax, rbx`
d) `jmp .copyloop`
e) `cmp rax rbx`

**System call usage:**

Consider the following assembly language code and identify the **false** statement:

```
1
2  WRITE equ 1
3  STDOUT equ 1
4
5  section .data
6
7     message: db "text", 10, 0
8     message_length equ $-message-1
9
10 section .text
11
12    mov rax, WRITE
13    mov rdi, STDOUT
14    mov rsi, message
15    mov rdx, message_length
16    syscall
```

a) The syscall type is set in register `rax`.
b) The syscall has four arguments.*
c) The previous code writes the word "text" in the standard output.
d) The four register must be set up for the write syscall to work correctly.
e) `cmp rax rbx`

**Sample problem:**

Create a routine that receives an integer number by parameter and prints out its countdown down to zero. When finished, the routine returns the value 1.