CMPT 295: Assembly Language Programming SOLUTIONS

The following questions are sample questions based on using the x86-64 assembly language:

1. Write an assembly language program segment that will calculate 5 + 4 - 6 if the values 5, 4, and 6 are stored in locations 0x40000C, 0x4000E, and 0x40010 respectively. The result is left in a register.

```
        .text
        .globl  main
main    mov     $0x40000C, %rbx
        mov     0(%rbx), %ax
        add     2(%rbx), %ax
        sub     4(%rbx), %ax
```

2. Consider the following program:

```
        mov     $0x78, %rax
LOOP:   sub     $1, %rax
        cmp     $0, %rax
        jne     LOOP
        . . .
```

   (a) How many times is the **sub** instruction executed?

   ANSWER: $0x78 = 120

   (b) How many times does the program branch to LOOP?

   ANSWER: 120 − 1 = 119

   (c) What change is necessary to make the program branch 210 times?

   ANSWER: Change the 1st operand of the `mov` isntruction to 211 = $D3

3. Assume a subprogram is already defined for computing the square root of an integer value. This subprogram is named SQRT and requires a 4 byte integer as its argument. It returns the result as 8 bytes The first four bytes define the digits, as an unsigned integer, to the left of the decimal, the second four bytes define the digits to the right of the decimal point, again as an unsigned integer. For example the square root of 2 (i.e., 1.4142135) would be returned as 1 and 4142135. Write a calling program segment to call the subprogram, SQRT, and then check if the argument is a perfect square. Store the value 1 at a location labelled "PERFECT" if the number is a perfect square; else store 0 there.

   SOLUTION:

```
            .data
VALUE:      .long                   # 4-byte argument value
PERFECT:    .quad
            .text
            .globl  main
main:       mov     $VALUE, %rbx
            mov     0(%rbx), %edi
            call    SQRT
            cmp     $0,%eax
            je      PERFSQ
            mov     $0, 4(%rbx)
            ret
PERFSQ:     mov     $1, 4(%rbx)
            ret
```

4. Demonstrate your understanding of addressing modes by providing three different instructions that will place the value 0 into register `%rax`. You may assume that the value 0 is available in external memory location "ZERO" and also in register `%rdi`.

   ANSWERS:

   (a) `mov $0, %rax`

   (b) `mov %rdi, %rax`

   (c)
   ```
   mov $ZERO, %rbx
   mov 0(%rbx), %rax
   ```

5. Suppose the **neg** instruction does not exist. Write an assembly language subprogram, `NEG`, to compute the negation of a 64-bit number. The subprogram uses the call by value method to pass arguments and results.

   SOLUTION:

```
        .text
        .globl      NEG
NEG:    cmp $0, %rdi
        jnl DONE
        mov         $0, %rcx
        sub         %rdi, %rcx
        mov         %rcx, %rdi
DONE:   mov         %rdi, %rax   ret
```

6. A subprogram requires the use of four registers `r12, r13, r14`, and `r15`. As well, it will need to allocate three quad values to store temporary results obtained during execution. Construct a stack frame diagram showing how stack memory should be organized for proper execution

   SOLUTION:

frame pointer →	local var 3
	local var 2
	local var 1
	r15
	r14
	r13
	r12
stack pointer →	return address

Also check out the following problems from the textbook: 3.1, 3.2, 3.8, 3.9, 3.18, 3.34, 3.35.