Homework 7

Problem 7.1
Solution:

my_function:

```
.       addi $sp,$sp , -12        # adjust stack for three items
.       sw $ra , 8($sp)       # save the return address
.       sw $a0 , 4($sp)       # save the argument x
.       sw $a1 , 0($sp)       # save the argument y
.       add $t0, $a0, $a1       # storing x + y
.       sub $t1, $a0, $a1       # storing x - y
.       slti $t2, $a0, 10       # checking else: x < 10
.       bne $t2, $a0, IF       # if x > 10 and ! =, go to IF
.       add $v0, $t0, $zero       # if x <= 10 then return x + y
.       addi $sp, $sp, 4       # adjusting stack to delete 1 item
.       jr $ra              # jump back to calling routine
IF:

.       add $v0, $t1, $zero       # if x > 10, return x - y
.       addi $sp, $sp, 12       # adjusting stack to delete 3 items
.       jr $ra             # jump back to calling routine
```

Problem 7.2
Solution:

In this, letâs assume that the answer of the multiplication will be 32 bits only.

is_more_than_fifty:

```
.       addi $sp ,$sp , -12             # adjust stack for three items
.       sw $ra , 8($sp)             # save the return address
.       sw $a0 , 4($sp)             # save the argument a
.       sw $a1 , 0($sp)             # save the argument b
.       jal prod                 # jump  link the product function
.       slti $t2, $t1, 50            # checking if prod < 50
.       bne $t2, $t1, IF             # if prod ! = 50  prod > 50
.       addi $v0, $zero, 0              # return value 0
.       addi $sp, $sp, 4              # adjusting stack to delete 1 item
.       jr $ra                       # jump back to calling routine
IF:
.       addi $v0, $zero, 1             # if prod > 50, return 1
.       addi $sp, $sp, 12             # adjusting stack to delete 3 items
.       jr $ra
prod:
.       mult $a0, $a1              # multiplying a * b
.       mfhi $t0              # loads upper 32 bits from product
.       mflo $t1              # loads lower 32 bits from product
.       jr $ra             # return (Copy $ra to PC)
```

Problem 7.3
Solution:

```
int i = 0;
while(A[i] != -1){
.          i++;
}
```

sll is for shifting 2 bits. By this, we multiply the value by 4 in order to load the array A[i]. After loading the array, we run a while loop. We exit the loop when the value in A[i] == -1. Within the loop, we increment the value each time.

Problem 7.4

Solution:

For each line of the MIPS code, we check the corresponding values from the MIPS reference sheet. For example, first line is sll $t1, $s3, 2. sll has an opcode and function code of 0. This remains same when we convert it into binary too. Its rd value is $t1, which coressponds to 9 and 01001 in binary. Similarly, its rs value is $s3, which is equal to 19 in decimal and 10011 in binary.

Load Word (lw) is an I type instruction so instead of its function code, we write a 16 bit address. Beq is also an I type instruction.

Program Counter	Machine Code	Binary Format
60000	0 0 19 9 2 0	000000 00000 10011 01001 00010 000000
60004	0 9 22 9 0 32	000000 01001 10110 01001 00000 100000
60008	35 9 8 0	100011 01001 01000 0000000000000000
60012	4 8 21 2	000100 01000 10101 0000000000000010
60016	8 19 19 1	001000 10011 10011 0000000000000001
60020	2 15000	000010 00000000000011101010011000
60024		

For jump: destination = PC[0 : 3] + word address (+=concatenate)

Problem 7.5

Solution:

(a) First let's convert these into binary numbers. To do so, we evaluate each digit and write its corressponding binary value. Each 0 in the hexa value gives 0000 in terms of binary. In case of letters, we first convert them into decimal and then to binary. For instance, C in hexa gives 12 in decimal. $12_{10} = 1100_2$: We complete 32 bits by adding any 0s in the start if necessary.

0x0C000000$_{16} = 00001100000000000000000000000000_2$

Now, $2^{26} + 2^{27} = 201326592_{10}$

$0xC4630000_{16} = 11000100011000110000000000000000_2$.

Since the first bit is 1, we know that it is a negative number.In order to convert it to a positive number, we'll have to invert all bits and then add a 1.

Before Inverting Bits:1100 0100 0110 0011 0000 0000 0000 0000$_2$

After Inverting Bits: 0011 1011 1001 1100 1111 1111 1111 1111
After Adding 1: 0011 1011 1001 1101 0000 0000 0000 0000

0011 1011 1001 1101 0000 0000 0000 0000$_2 = 1000144896_{10}$

Final Answer: -1000144896_{10}

(b) 0x0C000000$_{16}$
$= 00001100000000000000000000000000_2$
$= 201326592_{10}$

0C4630000$_{16} = 11000100011000110000000000000000_2$
$= 2^{16} + 2^{17} + 2^{21} + 2^{22} + 2^{26} + 2^{30} + 2^{31}$
$= 3294822400_{10}$

(c) (i) SRA (Shift Right Arithmetic) according to this link: https://www.slideshare.net/tagbagtroj/mips-opcodes

On some sites, Jump and Link (JAL) ((Source: CS UCR) also has the same opcode but since it's an I type instruction, I would believe that the first one is correct.
(ii) Invalid opcode.