6. This exercise uses MIPS assembly instructions. The relevant entries from the Appendix B of your book are given for the instructions used in this exercise.

Given below is an assembly program to perform a certain operation. Go through the program step by step to answer the following questions.

A MIPS Assembly Program

```
begin:  addi $t1, $0, 0
        addi $t2, $0, 1

loop:   slt $t3, $t5, $t2
        bne $t3, $0, output
        add $t1, $t1, $t2
        addi $t2, $t2, 2
        j loop

output: add $t6, $t1, $0
```

a) What does the above MIPS assembly program do? What is the value stored in output register $t6 at the end of program execution if the input register $t5 contains the decimal value 10? (3 points)

b) Modify the program to load input from memory address 0x00000010 and store the output in memory address 0x00000020 instead of the registers $t5 and $t6. (2 points)

c) For reusability of code, we rewrite the assembly program given in (a) using subroutines (procedures). The functionality of the code remains the same. Complete the modified assembly code below by filling in the empty blocks. (2 points)

Assembly program using subroutines

```
begin     : add $a0, $t1, 10 # $t1 is the input reg
            jal function
            add $t6, $v0, $0
halt      : j  halt

function  : addi $t1, $0, 0
            addi $t2, $0, 1

   loop   : slt $t3, $a0, $t2
            bne $t3, $0, exit_func
            add $t1, $t1, $t2
            addi $t2, $t2, 2
            j loop

exit_func : add $v0, $t1, $0
            jr $ra
```

d) What is the value stored in register $t1 at the end of program execution for the code given in (c)? (1 point)

e) As you can observe that the subroutine *function* overwrites register $t1, suggest modifications to the code to preserve $t1's contents. (2 points)

Relevant entries from Appendix B

`[reg]:`	contents of register
`SignImm:`	sign-extended immediate `={{16{imm[15]}},imm}`
`[Address]:`	contents of memory location `Address`
`BTA:`	branch target address `= PC + 4 + SignImm <<2)`
`JTA:`	jump target address `= {(PC+4)[31:28], addr, 2'b0}`

Name	Description	Operation
`j`	Jump	`$ra = PC +4, PC =JTA`
`jal`	Jump and link	`$ra = PC +4, PC =JTA`
`beq`	Branch if equal	`If ([rs] == [rt]) PC = BTA`
`addi`	Add immediate	`[rt] = [rs] + SignImm`
`lw`	Load word	`[rt] = [Address]`
`sw`	Store word	`[Address] = [rt]`
`jr`	Jump register	`PC = [rs]`
`and`	And	`[rd] = [rs] & [rt]`
`xor`	Xor	`[rd] = [rs] ^ [rt]`