

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	
<b>begin:</b>	<b>addi</b> \$t1, \$0, 0 <b>addi</b> \$t2, \$0, 1
<b>loop:</b>	<b>slt</b> \$t3, \$t5, \$t2 <b>bne</b> \$t3, \$0, output <b>add</b> \$t1, \$t1, \$t2 <b>addi</b> \$t2, \$t2, 2 <b>j</b> loop
<b>output:</b>	<b>add</b> \$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	
<b>begin</b>	: <b>add</b> <span style="border: 1px solid black; padding: 0 2px;">\$a0</span> , \$t1, 10 # <i>\$t1 is the input reg</i> <span style="border: 1px solid black; padding: 0 2px;">jal</span> function <b>add</b> \$t6, \$v0, \$0
<b>halt</b>	: <b>j</b> halt
<b>function</b>	: <b>addi</b> \$t1, \$0, 0 <b>addi</b> \$t2, \$0, 1
<b>loop</b>	: <b>slt</b> \$t3, \$a0, \$t2 <b>bne</b> \$t3, \$0, exit_func <b>add</b> \$t1, \$t1, \$t2 <b>addi</b> \$t2, \$t2, 2 <b>j</b> loop
<b>exit_func</b>	: <b>add</b> <span style="border: 1px solid black; padding: 0 2px;">\$v0</span> , \$t1, \$0 <span style="border: 1px solid black; padding: 0 2px;">jr</span> \$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 =  $PC + 4 + SignImm \ll 2$   
`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	$[rd] = [rs] \wedge [rt]$