CMPE 361

LABORATORY REPORT 2

Task 1:

Write a MIPS assembly code for the C code given.

C Code:

int a=12;

int b=23;

if (a>b) c=a*2;

else d=a+b;

Answer:

```
.data
   a: .word 12   # Variable a with initial value 12
   b: .word 23   # Variable b with initial value 23
   c: .word 0    # Variable c, initialized to 0
   d: .word 0    # Variable d, initialized to 0

.text
   lw $t0, a     # Load the value of a into register $t0
   lw $t1, b     # Load the value of b into register $t1

   # Compare a and b
   bgt $t0, $t1, a_greater_than_b  # Branch to a_greater_than_b if a > b

   # Code for the else part
   add $t2, $t0, $t1  # Add a and b, store result in $t2
   sw $t2, d          # Store the result in variable d
   j end_if           # Jump to the end of the if-else block

a_greater_than_b:
   # Code for if part
   sll $t2, $t0, 1    # Multiply a by 2, store result in $t2
   sw $t2, c          # Store the result in variable c

end_if:
   # Exit the program
   li $v0, 10   # Load the exit syscall code into register $v0
   syscall      # Execute the syscall
```

Task 2a:

Write the associated MIPS assembly code for the C code given (20 Points).

C Code: int a=128;

int b=0;

while (a!=1){

a=a/2;

b=b+1;}

Answer:

```
.data
a: .word 128   # Initialize a with 128
b: .word 0     # Initialize b with 0

.text
main:
    lw $t0, a    # Load the value of a into register $t0
    lw $t1, b    # Load the value of b into register $t1

loop:
    beq $t0, 1, endloop # If a is 1, exit the loop
    sra $t0, $t0, 1     # Shift right arithmetic by 1 (equivalent to division by 2)
    addi $t1, $t1, 1    # Increment b
    j loop              # Jump back to the start of the loop

endloop:
    sw $t1, b   # Store the final value of b back into memory
    jr $ra      # Return from main
```

Task 2b:

Explain what each line of your MIPS assembly code does, write modified registers, and values the modified registers get using a table given below. Note that you can extend the rows of the table as much as required (20 Points).

Answer:

Here is the explanation of each line:

a. .data: Directive to declare initialized data or constants.
b. a: .word 128: Declares an integer variable a initialized to 128.
c. b: .word 0: Declares an integer variable b initialized to 0.
d. .text: Directive indicating the start of the code section.
e. .globl main: Declares main as a global function that can be called from outside.

f. main:: Label for the start of the main function.
g. lw $t0, a: Loads the word (integer) from the memory address of a into register $t0.
h. lw $t1, b: Loads the word from the memory address of b into register $t1.
i. loop:: Label for the start of the loop.
j. beq $t0, 1, end: Branch if equal. If the value in $t0 (a) is equal to 1, jump to the end label.
k. sra $t0, $t0, 1: Performs an arithmetic right shift on $t0 by 1 bit, effectively dividing a by 2.
l. addi $t1, $t1, 1: Adds immediate (1) to $t1, incrementing b.
m. j loop: Unconditional jump back to the loop label.
n. end:: Label marking the end of the loop.
o. sw $t1, b: Stores the word in $t1 back to the memory address of b.
p. jr $ra: Jump to the return address ($ra), effectively returning from the main function.

This table down below shows the modified registers and their values after each instruction in the loop. The value of $t0 (representing a) is halved each time the loop iterates, while $t1 (representing b) is incremented by 1 on each iteration. The loop will be stopped working when a becomes 1.

MIPS Assembly Code Line	Explanation of the Line	Modified Register	Values of the Modified Registers
lw $t0, a	Load value of 'a' into register '$t0'	$t0	Initially 128
lw $t1, b	Load value of 'b' into register '$t1'	$t1	Initially 0
beq $t0, 1, endloop	If '$t0' (a) is equal to 1, jump to 'endloop'	-	-
sra $t0, $t0, 1	Arithmetic right shift of '$t0' by 1 (a = a/2)	$t0	Halved each iteration
addi $t1, $t1, 1	Increment '$t1' by 1 (b = b + 1)	$t1	Incremented by 1 each iteration
j loop	Jump back to start of the loop	-	-
sw $t1, b	Store the final value of '$t1' and (b) back into the memory	-	-
jr $ra	Return from main function	-	-

Task 3:

Write the associated MIPS assembly code for the C code given.

C Code:

int array [1000];

int a;

for (a=0; a<=1000; a=a+2)

array[a]=array[a]*2;

Answer:

```
.data
array: .space 4000  # Allocate space for 1000 integers (4 bytes each)

.text
.globl main
main:
    la $s0, array   # Load the base address of array into $s0
    li $s1, 0       # Initialize loop variable a to 0

loop:
    bgt $s1, 999, endloop  # If a > 999, exit the loop

    sll $t0, $s1, 2       # Multiply a by 4 to get the byte offset (as each int is 4 bytes)
    add $t0, $s0, $t0     # Add the offset to the base address to get the address of array[a]

    lw $t1, 0($t0)        # Load the value at array[a] into $t1
    sll $t1, $t1, 1       # Multiply the value by 2
    sw $t1, 0($t0)        # Store the doubled value back into array[a]

    addi $s1, $s1, 2      # Increment a by 2
    j loop                # Jump back to the start of the loop

endloop:
    # Your code to exit or return from main
    li $v0, 10            # Exit syscall
    syscall
```