MELİSA SUBAŞI ***ID:22829169256***

TASK 1:

```
    add $s0, $0, 12     # $s0 = a
    add $s1, $0, 23     # $s1 = b
     blt $s0, $s1, else   # Branch to else if a is less than b
    add $s2, $s0, $s0    # Multiply a by 2 and store the result in $s2
    sw $s2, 0x10000000($0)        # Store the result in 'c'
    j done
else:
    # If a <= b, execute the following:
    add $s3, $s0, $s1    # Add a and b and store the result in $s3
    sw $s3, 0x10000004($0)        # Store the result in 'd'
done:
```

TASK 2:

a)

```
add $s0, $0, 128
add $s1, $0, 0
add $s2, $0, 2
add $s4, $0, 1
while_loop:
  beq $s0, $s4, end_while
  div $s0, $s2
  mflo $s0
  add $s1, $s1, 1
  j       while_loop
end_while:
lui  $s0, 0x23B8        # $s0 = 0x23B80000
ori  $s0, $s0, 0xF000   # $s0 = 0x23B8F000
```

b)

Line	Explanation	Modified Register	Values of Modified Register
1	Adds the immediate value 128 to the value in register $0 and stores the result in register $s0.	$s0	128
2	Adds the immediate value 0 to the value in register $0 and stores the result in register $s1.	$s1	0
3	Adds the immediate value 2 to the value in register $0 and	$s2	2

	stores the result in register $s2.		
4	Adds the immediate value 1 to the value in register $0 and stores the result in register $s4.	$s4	1
5	Marks the beginning of a loop.	-	-
6	Branches to the label "end_while" if the values in registers $s0 and $s4 are equal.	-	-
7	Divides the value in register $s0 by the value in register $s2. The quotient is stored in register $s0, and the remainder is stored in register $s2.	$s0, $s2	Quotient in $s0, Remainder in $s2
8	Moves the contents of the special register LO (where the quotient is stored after division) to register $s0.	$s0	Quotient from the division
9	Adds 1 to the value in register $s1.	$s1	$s1 + 1
10	Jumps (unconditionally) to the label "while_loop", restarting the loop.	-	-
11	Marks the end of the loop.	-	-
12	Loads the immediate value 0x23B80000 into the upper 16 bits of register $s0.	$s0	0x23B80000
13	Bitwise ORs the value in register $s0 with the immediate value 0xF000, updating the lower 16 bits of $s0.	$s0	0X23B8F000

TASK 3:

```
       addi $s1, $0, 0
       addi $t2, $0, 1000
loop:
       sle  $t0, $s1, 1000
       beq  $t0, $0, done
       sll  $t0, $s1, 2
       add  $t0, $t0, $s0
       lw   $t1, 0x10000000($0)
       sll  $t1, $t1, 1
       sw   $t1, 0x10000004($0)
       addi $s1, $s1, 2
       j    loop
done:
```

Explanations for each line:

1-Adds the immediate value 0 to the value in register $0 and stores the result in register $s1.

2- Adds the immediate value 1000 to the value in register $0 and stores the result in register $t2.

3-Marks the beginning of a loop.

4- Sets register $t0 to 1 if the value in register $s1 is less than or equal to 1000, otherwise sets it to 0.

5- Branches to the label "done" if the value in register $t0 is equal to 0.

6- Shifts the value in register $s1 left by 2 bits and stores the result in register $t0.

7- Adds the values in registers $t0 and $s0 and stores the result in register $t0.

8- Loads a 32-bit word from the memory address 0x10000000 + the value in register $0 and stores it in register $t1.

9- Shifts the value in register $t1 left by 1 bit.

10- Stores the value in register $t1 into the memory address 0x10000004 + the value in register $0.

11- Adds the immediate value 2 to the value in register $s1.

12- Unconditionally jumps to the label "loop", restarting the loop.

13-Marks the end of the loop.