5. The questions 5, 6 and 7 are based on the MIPS assembly code given below.

```
        addi $t0, $0, 8
        xor  $s0, $s0, $s0
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        add  $t3, $t1, $s0
        add  $s0, $t2, $t3
        addi $t0, $t0, -1
        j    loop
done:
```

(a) (2 points) Briefly explain what the above MIPS assembly code does.

Solution:
The program will execute a loop 8 times. In each iteration of the loop, the content of the address `0x0000 0004` and the content of the address `0x0000 0024` will be added together and added to the register `$s0` which was initialized to the value 0 (A xor A is 0).

(b) (1 point) Assuming the data at memory location `0x0000 0004` is decimal 16 and at memory location `0x0000 0024` is decimal 32, what will be the content of the register `$s0` in **hexadecimal** when the program execution jumps to `done:`?

Solution: The program calculates $8 \times (mem(0x00000004) + mem(0x00000024))$. This equals to $8 \times (32 + 16) == 384$. In hexadecimal this will be $0x0180$. It is actually very easy to do the calculation if you do it in binary.

(c) (2 points) Briefly explain the three different MIPS instruction types (R, I, J) and show **one** instruction of each type from the example code above.

Solution:

R-type Uses up to three registers, two for source and one for destination.
Example: `xor $s0, $s0, $s0`.

I-type Uses a 16-bit constant as part of the instruction.
Example: `addi $t0, $0, 8`.

J-type Uses a 26-bit constant that can be used to calculate the address of the next instruction, allowing the program execution to jump to a new location.
Example: `j loop`.