

5. In this section, you will be given a task and two code snippets in MIPS assembly language. You will have to decide which of the code snippets can be used for the task.

For all the questions assume the following initial values:

Registers:

Register	Value
\$s0	0x0000 00FF
\$s1	0x0000 0004
\$s2	0x0000 0008
\$s3	0x0000 000C

Memory:

Address	Value
0x0000 00000	0x0000 FF00
0x0000 00004	0x0000 00FF
0x0000 00008	0xFFFF FFF7
0x0000 0000C	0x1234 5678

- (a) (3 points) Set the content of the register \$t1 to 0x0000 1234

(A)

```
lw  $t1, 0xC($0)
srl $t1, $t1, 16
```

(B)

```
xor $t1, $t1, $t1
ori $t1, 0x1234
```

☐ none☐ A☐ B☒ **Both A and B**

- (b) (3 points) Starting from the address 0x0000 4000 write all zeroes to 1024 consecutive memory locations (until 0x0000 5000)

(A)

```
addi $s0, $s0, 0x1000
LOOP: sw  $0, 0x4000($s0)
addi $s0, $s0, -1
bne  $s0, $0, LOOP
```

(B)

```
addi $s0, $s0, 0x4000
addi $s1, $s0, 0x1000
addi $s2, $0, 1
LOOP: sw  $0, $s0
sub  $s1, $s1, $s2
bne  $s0, $s1, LOOP
```

☐ none☒ **A**☐ B☐ Both A and B

B is incorrect since the assignment is on \$s0 which is constant at 0x4000. If that line were to read:

```
LOOP: sw $0, $s1
```

it would be correct.

- (c) (3 points) Add all the numbers from 0 to 255

(A)

```

        lw    $s1, $s0
        xor   $s0, $s0, $s0
LOOP:   add   $s0, $s0, $s1
        addi  $s1, $s1, -1
        bne   $s1, $0, LOOP

```

(B)

```

        addi  $s1, $0, 255
        lw    $s0, $0
LOOP:   addi  $s1, $s1, -1
        beq   $s1, $0, DONE
        jmp   LOOP
DONE:

```

☐ none☒ A☐ B☐ Both A and B

B is incorrect since there is no addition of numbers anywhere. The loop is correct though

- (d) (3 points) Jump to subroutine STOP if only the 4th bit from the right (representing
- 2^3
-) of the data written at address 0x0000 0020 is 1. Otherwise continue with the program at CONT.

(A)

```

        lw    $s0, 0x20($0)
        srl   $s0, $s0, 3
        addi  $s1, $0, 1
        beq   $s0, $s1, CONT
        jmp   STOP
CONT:   ...
STOP:   ...

```

(B)

```

        addi  $s0, $0, 0x20
        lw    $s1, $s0
        lw    $s2, 0x8($0)
        and   $s3, $s1, $s2
        bne   $s3, $0, CONT
        jal   STOP
CONT:   ...
STOP:   ...

```

☐ none☐ A☒ B☐ Both A and B

A looks ok, but it would also work if other bits (higher than 4) are one as well, the jump is not to a subroutine, and the condition is inverse

- (e) (3 points) Save the two registers \$s0 and \$s1 to the stack

(A)

```

        jal   SAVE
        ...
SAVE:   sw    $sp, $s0
        sw    $sp, $s1
        jr    $ra

```

(B)

```

        addi  $sp, $sp, -8
        sw    $s0, 8($sp)
        sw    $s1, 4($sp)

```

☐ none☐ A☒ B☐ Both A and B

A is incorrect since both values overwrite the last value in the stack.