1 Activity 1 - Sight Reading MIPS

We’re going to be learning MIPS assembly! We’re going to start by reading a little bit of it - because I bet you can figure out a lot about MIPS just by making educated guesses.

Some basic hints about MIPS:

• Almost all MIPS operations take 32-bit inputs and produce 32-bit outputs. (There are exceptions, but they are very rare!)

• Names that start with dollar signs, like $s0, are registers - basically, 32-bit variables inside the CPU.

  The $sX and $tX registers ($s0, $t0, etc.) are general purpose registers, and can be used by the program for any purpose. The rest of the registers have special meanings, which we’ll describe as we go.

• If a MIPS instruction writes to a register, the register it writes to is (almost always) the one on the left.

Do your best to figure out the following code snippets. If you can’t figure it out, don’t worry - but figure out as much as you can!

Snippet 1

```
add $s0, $s1,$s2
```

Snippet 2

```
addi $s0, $s1,123
```

Snippet 3

```
add $t3, $s7,$zero
```

Snippet 4

```
addi $t3, $t3,-1
```
Snippet 5

```assembly
add $t0, $s0,$s1
add $t1, $s2,$s3
sub $s4, $t0,$t1
```

Snippet 6

```assembly
andi $t0, $s0,0x1
```

Snippet 7

```assembly
sub $s3, $zero,$s3
```

Snippet 8

```
# HINT: L is for "load"
la $s0, my_variable
lw $s1, 0($s0)
```

Snippet 9

```
# HINT: SW breaks one of the rules I told you about
la $s0, counter
lw $s1, 0($s0)
addi $s1, $s1,1
sw $s1, 0($s0)
```

Snippet 10

```
addi $v0, $zero,1      # 1 = print_int
add $a0, $s0,$zero    # parameter
syscall
```
Bonus Snippet 1

```c
# if (x == y)
#   printf("%d", x);
#
#   -- Can you determine what each instruction does???
```

```c
bne $s0,$s1, AFTER
addi $v0, $zero,1
add $a0, $s0,$zero
syscall
```

```
AFTER:
```

Bonus Snippet 2

```c
# if (foo < bar)
#   foo++;
# else
#   bar++;
#
#   -- Can you determine what each instruction does???
```

```c
slt $t0, $s0,$s1
beq $t0,$zero, FALSE
addi $s0, $s0,1
j AFTER
```

```
FALSE:
  addi $s1, $s1,1
AFTER:
```

Bonus Snippet 3

```c
addi $v0, $zero,4     # what do you think 4 means???
la $a0, HELLO_WORLD_MSG
syscall
```

```
# further down in the file, somewhere...
.data
HELLO_WORLD_MSG:
  .string "Hello world!\n"
```