MIPS

MIPS is a load and store architecture that allows all operands to be register except for load ans store operations.

Register Name	Usage
$v0	Argument (read/write and type: int, float, string, etc.)
$a0	Argument (address or value)
$t0 - $t9	Temporary registers

MIPS is byte-addressed.

Register size: 32 bits.

The above registers hold integer values only.

MIPS program: data

Two parts:

Part A: Variables

```
.data
    Label: .type initialization
```

Example:

```
.data
    x: .word 3 # word is integer
    y: .word
    t: .asciiz "Salut"    #asciiz is string
```

MIPS program: instructions

Part B: Instructions

.text

....

Three types of instructions: R-type, I-type and J-type.

Type 1: R-type (Register type)

Opcode	Operands

Operands: 3, 2 or 1 register(s).

R-type

3 registers:

```
add $t2, $t1, $t0
div   $t2, $t1, $t0
```

2 registers:

```
move $t0, $t1
div $t1, $t0   # quotient is in the register LO and remainder is in the reigter HI
```

1 register:

```
mflo $t2  # move the quotient from lo to t2
Mfhi $t3  # move the remainder from hi to t3
```

I-type

Type 2: I-type (Immediate type)

- Load/Store

Opcode	Register	Variable
lw	$t0	x
sw	$t0	x
la	$t0	x

lw: load word

sw: store word

la: load address

I-type

- Immediate value

Opcode	Operands

Operands:

2 registers, 1 immediate value

addi $t0,$t0,1 # incrementation

1 register, 1 immediate value

li $t0, 1 # initialization

To be continued …

Print data

Print an integer

v0 ← 1

a0 ← value

syscall

Print a string

v0 ← 4

a0 ← @text

syscall

Exercise

Write a MIPS program to divide two integers a=7 and b=2, then display the quotient and the remainder.