

MIPS (Microprocessor without Interlocked Pipelined Stages)

MARS (MIPS Assembler and Runtime Simulator)

Have a look at the register panel on the right

MIPS Program

- 2 sections:
 - .data
 - .text

Data

- name: datatype value
- Eg:
 - msg: .asciiz "Hello \n"
 - letter: .byte 'a'
 - number: .word 97
 - real: .float 2.2
 - real: .double 3.512
 - inStr: .space 20

Get Input & Print Output

Instructions

syscall

Prerequisites

content in register \$v0:

- 1 - print integer
- 2 - print float
- 3 - print double
- 4 - print string
- 5 - get integer
- 6 - get float
- 7 - get double
- 8 - get string
- 11 - print character
- 12 - get character

li \$v0, <number>

arguments

Integer / character / address:
\$a0 - \$a3
Float / double: \$f12

Load from Memory

string : la
character : lb
integer : lw
float : lwc1
double : ldc1

Available in Register

integer / character /
address : mov
float : mov.s
double : mov.d

return value

Integer / character :
\$v0-\$v1
Float / double: \$f0

Output

String

.data

msg: **.asciiz** "Hello\n"

.text

li \$v0, **4**

la \$a0, msg

syscall

Character

.data

letter: **.byte** 'm'

.text

li \$v0, **4**

la \$a0, letter

syscall

Integer

.data

num: **.word** 97

.text

li \$v0, **1**

lw \$a0, num

syscall

Float

.data

flNum: **.float** 2.2

.text

li \$v0, **2**

lwc1 \$f12, flNum

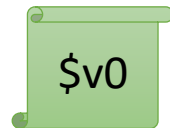
syscall

Input

Integer

.text

```
li $v0, 5  
syscall
```



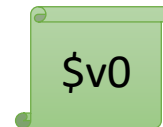
```
move $t0, $v0
```

```
li $v0, 1  
move $a0, $t0  
syscall
```

Integer

.text

```
li $v0, 5  
syscall
```



```
add $t0, $zero, $v0
```

```
li $v0, 1  
add $a0, $zero, $t0  
syscall
```


Input (contd.)

Float

.data

zeroFl: .float 0.0

.text

li \$v0, 6
syscall



lwc1 \$f4, zeroFl

li \$v0, 2
add.s \$f12, \$f0, \$f4
syscall

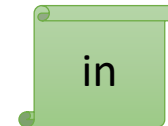
String

.data

in: .space 20

.text

li \$v0, 8
la \$a0, in
li \$a1, 20
syscall



Assignment

- Input a character
- Input a double
- Output a double
- Identify the Instruction encoding / format for each instruction
 - Highlight what each field indicates
 - Identify the pseudo instructions
- Store from register to memory