FREE eBook

LEARNING mips

Free unaffiliated eBook created from **Stack Overflow contributors.**



Table of Contents

About	. 1
Chapter 1: Getting started with mips	. 2
Remarks	.2
Examples	.2
Installation or Setup	.2
QtSpim for windows	2
MARS MIPS Simulator	2
Credits	13



You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mips

It is an unofficial and free mips ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official mips.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with mips

Remarks

This section provides an overview of what mips is, and why a developer might want to use it.

It should also mention any large subjects within mips, and link out to the related topics. Since the Documentation for mips is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting mips set up or installed.

QtSpim for windows

- 1. download QtSpim from here 32.6 MB
- 2. install it easy installation
- 3. make your first assembly file (.s) or use the sample *C:\Program Files* (*x86*)\QtSpim\helloworld.s
- 4. run the program from the desktop shortcut or C:\Program Files (x86)\QtSpim\QtSpim.exe

there are two windows for the program the main one labeled QtSpim here you see the program you are executing (labeled text), the memory(labeled data), the values of the registers (labeled FP Regs for floating point and Int Regs for integer) and the control for the simulator

the other window labeled console is where you will see the output and enter the input of your program if there are any

- 5. load the file using File -> Load File
- 6. you can use click run (f5) to see the end result or go step by step (p10) to see state of the register and memory while the program executing to debug

MARS MIPS Simulator

MARS MIPS simulator is an assembly language editor, assembler, simulator & debugger for the MIPS processor, developed by Pete Sanderson and Kenneth Vollmar at Missouri State University (src).

You get the MARS for free here. As for installing the 4.5 version, you might need the suitable Java SDK for your system from here

Before assembling, the environment of this simulator can be simplisticly split to three segments: the *editor* at the upper left where all of the code is being written, the compiler/output right beneath the editor and the *list of registers* that represent the "CPU" for our program.

<u>F</u> ile	<u>E</u> dit <u>R</u> un <u>S</u> ettings <u>T</u> ools <u>H</u> elp
Edi	it Execute
lab	b2ask1.asm
1	.text
	.globl main
	main:
4 5	li \$v0, 11
6	la \$a0, 'a'
7	syscall
8 9	li \$v0, 10
10	syscall
11	
•	
	: 1 Column: 1 🗹 Show Line Numbers
	Irs Messages Run I/O
ma	a
	u l
0	Clear

After assembling (by simply pressing F3) the environment changes, with two new segments

getting the position of the editor: the text segment where

i) each line of assembly code gets cleared of "pseudoinstructions" (we'll talk about those in a sec) at the "basic" column and

ii) the machine code for each instruction at the "code" column,

and the *data segment* where we can have a look at a representation of the memory of a processor with little-endian order.

	tings <u>T</u> ools <u>H</u>		G 🔒 🗞	» O O1		
dit Execute						•
Text Segment						
3kpt Address	Code	Basic		Sol	urce	1000000000
0x00400000		addiu \$2,\$0,0x0000	0000b 5: li	\$v0, 11	uice	
0x00400000		addiu \$4,\$0,0x0000		\$a0, 'a'		
0x00400008			7: syscal			
0x0040000c		addiu \$2,\$0,0x0000				
0x00400010			10: syscal			
						•
] Data Segment						
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+1
0x10010000	0x0000000		0x00000000	0x00000000	0x00000000	0x0000
0x10010020	0x0000000		0x00000000	0x00000000	0x00000000	0x0000
0x10010020	0x0000000		0x00000000	0x00000000	0x00000000	0x0000
	0x0000000		0x0000000	0x00000000	0x00000000	0x0000
0x10010060	0x0000000	-			0x00000000	
0x10010060 0x10010080	0x0000000	0000000x0 00	0x00000000	0x00000000		0x0000
0x10010060 0x10010080 0x100100a0	0x0000000 0x0000000	0000000 x 0000000	0x0000000	0x0000000	0x0000000	0x0000
0x10010060 0x10010080 0x100100a0 0x100100c0	0x000000 0x000000 0x000000	00000000x00000000000000000000000000000	0x00000000 0x00000000	0x00000000 0x00000000	0x00000000 0x0000000	0x0000 0x0000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0	0x0000000 0x0000000 0x0000000 0x0000000	00 0x000000 00 0x000000 00 0x0000000	0x0000000 0x0000000 0x0000000	0x0000000 0x00000000 0x00000000	0x0000000 0x0000000 0x0000000	0x0000 0x0000 0x0000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000	0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120	0x000000 0x000000 0x0000000 0x0000000 0x000000	00 00000x0 00 00 0000x0 00 00 0000x0 00 00 0000x0 00 00 00000x0 00	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120 0x10010140	0x000000 0x000000 0x000000 0x000000 0x000000	00 0x000000 00 0x000000 00 0x000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120 0x10010140 0x10010160	0x000000 0x000000 0x000000 0x000000 0x000000	00 0x0000000 0x0000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120 0x10010140 0x10010160	0x000000 0x000000 0x000000 0x000000 0x000000	00 0x0000000 0x0000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000 0x00000000 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000 0x0000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	0x000000 0x000000 0x000000 0x000000 0x000000	00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000 00 0x0000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010140 0x10010160 0x10010180 0x10010180	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010140 0x10010160 0x10010180 0x10010180	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100e0 0x10010100 0x10010140 0x10010160 0x10010180 0x10010180	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x10010000 0x10010000 0x10010100 0x10010120 0x10010140 0x10010180 0x10010180 Fars Messages Go: run	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x00000000 0x00000000 0x00000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180 Mars Messages Go: run Go: exe	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180 Mars Messages Go: run Go: exe	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x000 0x000 0x000 0x000 0x000 0x000 0x000
0x10010060 0x10010080 0x10010000 0x10010000 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180 F Mars Messages Go: run Go: exe Assembl	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
0x10010060 0x10010080 0x10010000 0x10010000 0x10010100 0x10010120 0x10010140 0x10010160 0x10010180 F Mars Messages Go: run Go: exe Assembl	0x0000000 0x0000000 0x0000000 0x0000000 0x000000	00 0x00000000 00 0x00000000 00 0x00000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x00000000 0x00000000 0x00000000 0x000000	0x0000 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000

After assembling, we can execute our code either all at once (F5) or step by step (F7), as well as rewinding the execution several steps backwards to the back (F8).

ile <u>E</u> dit <u>R</u> un <u>S</u> etti	tings <u>T</u> ools <u>H</u> elp	ρ				
	A 20101	5 0 🇞 🕻	1 🗊 🕙	× •		
Edit Execute						
Text Segment						■ • •
Bkpt Address	Code	Basic		Sou	Irce	
0x00400000	0x2402000b ad	diu \$2,\$0,0x000000)b 5: li	\$v0, 11		•
0x00400004		diu \$4,\$0,0x000006		\$a0, 'a'		
0x00400008			7: syscall			
0x0040000c		diu \$2,\$0,0x000000				
0x00400010			10: syscall			
•						•
Data Segment						
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000
0x10010080	0x00000000	0x0000000	0x00000000	0x00000000	0x00000000	0x0000000
0x100100a0	0x00000000	0x0000000	0x00000000	0x00000000	0x00000000	0x000000
0x100100c0	0x00000000	0x0000000	0x00000000	0x00000000	0x00000000	0x000000
0x100100e0	0x00000000	0x0000000	0x00000000	0x00000000	0x00000000	0x000000
0x10010100	0x00000000	0x0000000	0x00000000	0x0000000	0x00000000	0x000000
0x10010120	0x00000000	0x0000000	0x00000000	0x0000000	0x0000000	0x000000
0x10010140	0x00000000	0x0000000	0x00000000	0x0000000	0x00000000	0x000000
0x10010160	0x00000000	0x0000000	0x00000000	0x0000000	0x00000000	0x000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x000000
		♠ 0x10010000	0 (.data) 🔻	✓ Hexadecimal A	ddresses 🖌 Hexa	adecimal Value
lars Messages 🛛 R	Run I/O					
a	gram is finishe	ad running				

Now, let's see the example code from above and explain each line:

```
.text
.globl main
                #main function
main:
    $v0, 11
              #11=system code for printing a character, v0=register that gets the system
li
code for printing as value
   $a0, 'a' #'a'=our example character, $a0=register that accepts the character for
la
printing
         #Call to the System to execute our instructions and print the character at
syscall
the a0 register
li $v0, 10
               #11=system code for terminating, $v0=register that gets the system code for
terminating (optional, but desirable)
             #Call to the System to terminate the execution
syscall
```

MARS accepts and exports files with the .asm filetype

But the code above prints just a character, what about the good ol' "Hello World"? What about, dunno, adding a number or something? Well, we can change what we had a bit for just that:

```
#data section
.data
str: .asciiz "Hello world\n"
number: .word 256
.text
                   #code section
.globl main
main:
       $v0, 4
                            #system call for printing strings
li
       $a0, str
                             #loading our string from data section to the $a0 register
la
syscall
    $t0, number #loading our number from data section to the $t0 register
la
lw
       $s1, 0($t0)
                         #loading our number as a word to another register, $s1
addi
       $t2, $s1, 8
                          #adding our number ($s1) with 8 and leaving the sum to register
$t2
       $t2, 0($t0) #storing the sum of register $t2 as a word at the first place of
SW
$±0
1 i
      $v0, 10
                             # system call for terminating the execution
syscall
```

Before illustrating the results through MARS, a little more explanation about these commands is needed:

- **System calls** are a set of services provided from the operating system. To use a system call, a *call code* is needed to be put to \$v0 register for the needed operation. If a system call has arguments, those are put at the \$a0-\$a2 registers. Here are all the system calls.
- 1i (load immediate) is a pseudo-instruction (we'll talk about that later) that instantly loads a register with a value. 1a (load address) is also a pseudo-instruction that loads an address to a register. With 1i \$v0, 4 the \$v0 register has now 4 as value, while 1a \$a0, str loads the

string of str to the $\ensuremath{\texttt{sa0}}$ register.

- A *word* is (as much as we are talking about MIPS) a 32 bits sequence, with bit 31 being the Most Significant Bit and bit 0 being the Least Significant Bit.
- lw (load word) transfers from the memory to a register, while sw (store word) transfers from a register to the memory. With the lw \$\$1, 0(\$t0) command, we loaded to \$\$1 register the value that was at the LSB of the \$t0 register (thats what the 0 symbolizes here, the offset of the word), aka 256. \$t0 here has the address, while \$\$1 has the value. sw \$t2, 0(\$t0) does just the opposite job.
- MARS uses the Little Endian, meaning that the LSB of a word is stored to the smallest byte address of the memory.
- MIPS uses **byte addresses**, so an address is apart of its previous and next by 4.

By assembling the code from before, we can further understand how memory and registers exchange, disabling "Hexadecimal Values" from the Data Segment:

\$

٩

¢

			2	2 0101	
Edit	Execu	Ite			

Bkpt	Address	Code	Basic				Source			
	0x00400000	0x24020004	addiu \$2,\$0,4	8:	li	\$v0,	4	#sys		
	0x00400004	0x3c011001	lui \$1,4097	9:	la	\$a0,	str	#loa		m
	0x00400008	0x34240000	ori \$4,\$1,0							Ľ
	0x0040000c	0x000000c	syscall	10:	syscall					
	0x00400010	0x3c011001	lui \$1,4097	12:	la	\$t0,	number	#loa		st
	0x00400014	0x34280010	ori \$8,\$1,16							nι
	0x00400018	0x8d110000	lw \$17,0(\$8)	13:	lw	\$s1,	0(\$t0)	#loa		
	0x0040001c	0x222a0008	addi \$10,\$17,8	15:	addi	\$t2,	\$s1, 8	#add		
	0x00400020	0xad0a0000	sw \$10,0(\$8)	17:	SW	\$t2,	0(\$t0)	#sav		
	0x00400024	0x2402000a	addiu \$2,\$0,10	19:	li	\$v0,	10	# sy		
	0x00400028	0x000000c	syscall	20:	syscall					

•

🔲 Data Segment	t						
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Val
0x10010000	1819043144	1870078063	174353522	0	264	0)
0x10010020	0	0	0	0	0	0)
0x10010040	0	0	0	0	0	0)
0x10010060	0	0	0	0	0	0)
0x10010080	0	0	0	0	0	0)
0x100100a0	0	0	0	0	0	0)
0x100100c0	0	0	0	0	0	0)
0x100100e0	0	0	0	0	0	0)
0x10010100	0	0	0	0	0	0)
0x10010120	0	0	0	0	0	0)
0x10010140	0	0	0	0	0	0)
0x10010160	0	0	0	0	0	0)
0x10010180	0	0	0	0	0	0)
•							
		♦ 0x10010	000 (.data) 🔻	Hexadecimal	Addrossos 🗌 🛛	lexadecimal Valu	
			000 (.uata)	Mexadecimal	Auuresses n		25
Mars Messages	Run I/O						
	o world cogram is finis	bed running					

or enabling "ASCII" from the Data Segment:

📑 🗎 🏖		۵ 👌 👌	b []	2 🕅		1		0
dit Execute								
Text Segment							• • D	
3kpt Address	Code	Basic			Source			
0x004000	00 0x24020004	addiu \$2,\$0,4	8: 1	li \$v0,	4		#sys 4	·III-
0x004000	04 0x3c011001	lui \$1,4097	9: 1	La \$aO,	str		#loa	
0x004000	08 0x34240000	ori \$4,\$1,0						ma
	0c 0x000000c	-	10: s	syscall				
	10 0x3c011001		12: 1	la \$t0,	number		#loa	str
	14 0x34280010							nu
	18 0x8d110000		13: 1		0(\$t0)		#loa	
		addi \$10,\$17,8	15: a		\$s1, 8		#add	
	20 0xad0a0000 24 0x2402000a	addiu \$2,\$0,10	17: s 19: 1		0(\$t0) 10		#sav	
	28 0x0000000c			syscall	10		# sy	
							•	
] Data Segment	:							
Data Segment	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Valu	e (+10)	Value (+14) .)
			Value (+8) \n d 1 :			e (+10) \0 . \b		-
Address 0x10010000 0x10010020	Value(+0) 1 1 e H \0 \0 \0 \0 \0	0 W 0	\n d l :	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \	\0 . \b \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0
Address 0x10010000 0x10010020 0x10010040	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0	0 W 0 0/ 0/ 0/ 0/ 0/ 0/ 0/ 0/	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \	\0 . \b \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	0 W 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	\0 . \b \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \0	\0 . \b \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	w o 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \0 \	\0 . \b \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100a0	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0 \0	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l x \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \0 \ \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \	\0 . \b \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0	\0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l x \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \0 \ \0 \0 \0 \ \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \	\0 . \b \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100a0	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \	W O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l x \0	r \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 . \b \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x100100c0	Value (+0) 1 1 e H \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l x \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 . \b \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010100	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l x \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\0 . \b \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140	Value (+0) 1 1 e H \0 \	0 W 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0		Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	Value (+0) 1 1 e H \0 \	0 W 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0		Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	0 . \b 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	0 . \b 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \
Address 0x10010000 0x10010020 0x10010040 0x10010060 0x10010080 0x100100a0 0x100100c0 0x100100c0 0x10010100 0x10010120 0x10010140 0x10010160	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	0 . \b 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \
Address 0x10010000 0x10010020 0x10010040 0x10010080 0x10010080 0x10010080 0x10010000 0x10010000 0x10010100 0x10010120 0x10010140 0x10010140 0x10010180 0x10010180	Value (+0) 1 1 e H \0	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	0 . \b 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \
Address 0x10010000 0x10010020 0x10010040 0x10010080 0x10010080 0x10010080 0x10010000 0x10010000 0x10010100 0x10010120 0x10010140 0x10010140 0x10010180 0x10010180	Value (+0) 1 1 e H \0 \	0 W 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	\n d l 1 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	r \0 \0 \0 0 \0 \0 \0	\0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \ \0 \0 \	0 . \b 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0	Value (+14 \0 \0 \0 \0 \0 \0	\0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \

Start it like this

\$ java -jar Mars4_5.jar

Create this file and save it.

```
.text
main:
    li $s0,0x30
loop:
    move $a0,$s0  # copy from s0 to a0
    li $v0,11  # syscall with v0 = 11 will print out
    syscall  # one byte from a0 to the Run I/O window
    addi $s0,$s0,3  # what happens if the constant is changed?
    li $t0,0x5d
    bne $s0,$t0,loop
    nop  # delay slot filler (just in case)
stop: j stop  # loop forever here
    nop  # delay slot filler (just in case)
```

Press F3 to assembly it and then press run. Now you are started compiling and executing MIPS code.

Read Getting started with mips online: https://riptutorial.com/mips/topic/7049/getting-started-withmips

Credits

S. No	Chapters	Contributors
1	Getting started with mips	Community, Coursal, Dac Saunders, robert