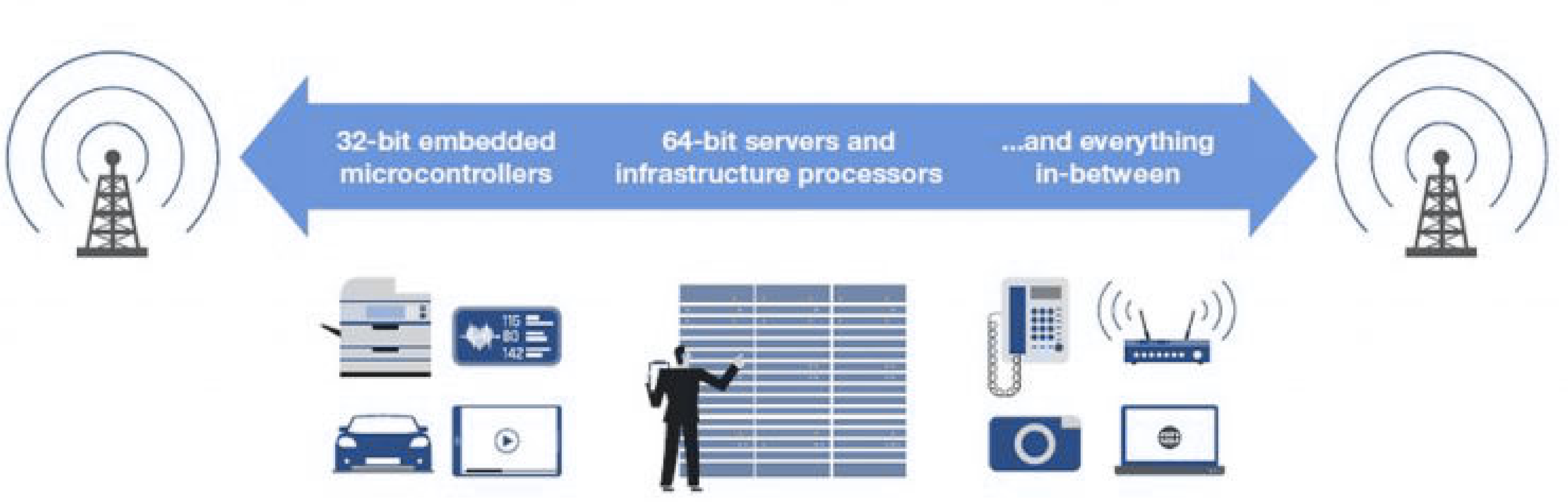
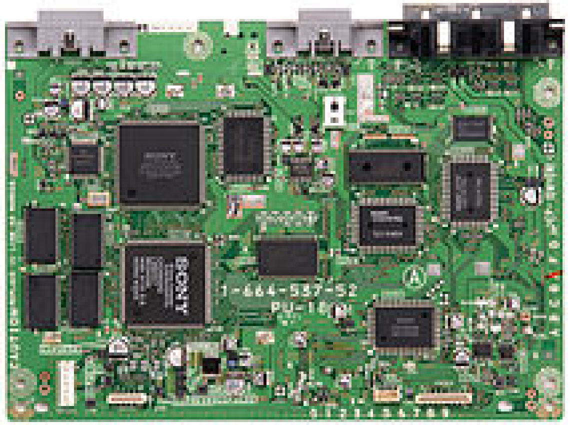
Processador MIPS

Processador RISC

MIPS Atual


32-bit embedded microcontrollers
64-bit servers and infrastructure processors
...and everything in-between
116
80
142

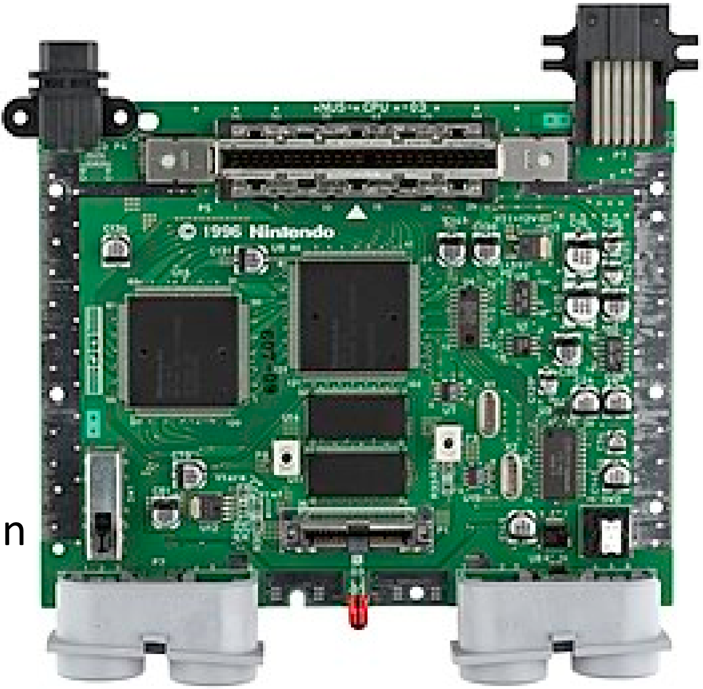
PSX



LSI CoreWare CW33300-based core

MIPS R3000A-compatible 32-bit RISC CPU MIPS R3051 with 5 KB L1 cache, running at 33.8688 MHz.
The microprocessor was manufactured by LSI Logic Corp. with technology licensed from SGI.
Features:
  Initial feature size (process node) was 0.5 micron (500 nm).
  850k – 1M transistors[citation needed]
  Operating performance: 30 MIPS
  Bus bandwidth 132 MB/s[5]
  One arithmetic/logic unit (ALU)
  One shifter
CPU cache RAM:
  4 KB instruction cache
  1 KB non-associative SRAM data cache

Nintendo 64



CPU**: 64-bit NEC VR4300 (MIPS R4300i)** with 24 KB L1 cache, running at 93.75 MHz.

Performance: 125 MIPS (million instructions per second), 93.75 MFLOPS (million floating-point operations per second).[1]

GPU: 64-bit Reality Coprocessor, running at 62.5 MHz and over half a billion arithmetic operations per second, capable of dual-issuing scalar and vector operations under the right circumstances.[2] It is a microcode-reprogrammable T&L GPU,[3] composed of two integrated processors: the Reality Signal Processor (RSP) and the Reality Display Processor (RDP).[4]

Curiosidade – MIPS The Rabbit

Super Mario 64

MIPS, the rabbit, can be first seen in Super Mario 64, after the player collected fifteen Power Stars. He is found in the basement of the Mushroom Castle - when Mario approaches MIPS, he runs away. After Mario catches MIPS, the rabbit rewards him a Power Star. MIPS appears in the basement a second time after Mario has collected fifty Power Stars, and Mario can catch MIPS again for another Power Star. MIPS will not reappear for the rest of the game after that.

In the remake Super Mario 64 DS, MIPS does not make a reappearance, instead being replaced by the rabbits scattered throughout the castle for each character to find. They are modeled after MIPS, but they do not give up Power Stars. Instead, they give up keys to unlock minigames in the Rec Room. Two of Mario's, one of Wario's, and one of Yoshi's rabbits can be found in the same location MIPS was in the original game. The rabbits are also internally named "MIP" with their key known as a "MIP Key," suggesting a connection.

Mario Party 3

MIPS also makes a cameo appearance in Mario Party 3 in Woody Woods, where several can be seen gathering near the item shop. MIPS appears in three colors: orange, yellow, and pink, which are colors matching the rabbits that replace MIPS in Super Mario 64 DS (the only absent color is green).

MIPS in Mario no Bōken Land

MIPS makes a cameo in the Super Mario 64 storyline of Mario no Bōken Land. The above text is from the Super Mario Wiki and is available under a Creative Commons license.

Attribution must be provided through a list of authors or a link back to the original article. Source: https://www.mariowiki.com/MIPS

Playstation 2

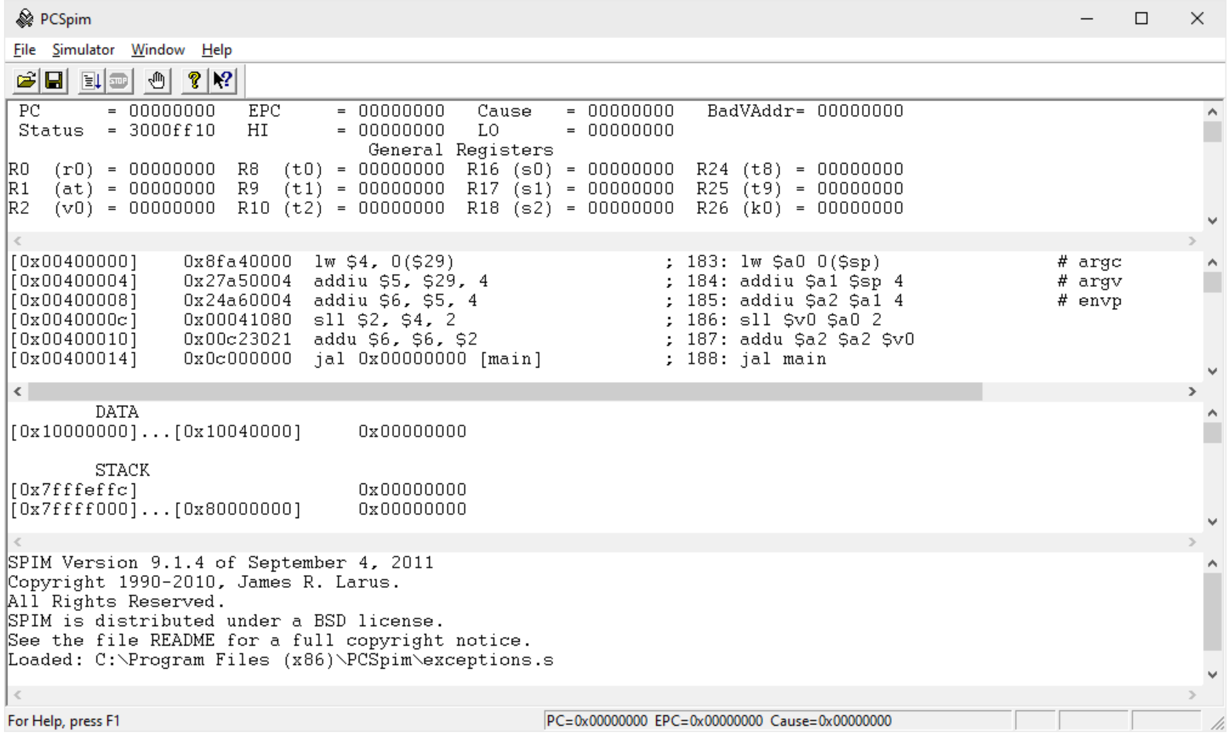
CPU: **MIPS III R5900-based "Emotion Engine", clocked at 294.912 MHz** (299 MHz on newer versions), with 128-bit SIMD capabilities[4][5]
250-nm CMOS manufacturing (ending with 65-nm CMOS), 13.5 million transistors, 225 mm² die size,[6] 15 W dissipation (combined EE+GS in SCPH-7500x and later SCPH-7000x): 86 mm², 53.5 million transistors)[7] (combined EE+GS+RDRAM+DRAM in SCPH-7900x ended with 65 nm CMOS design)[8]

CPU core: MIPS R5900 (COP0), 64-bit, little endian (mipsel). CPU is a superscalar, in-order execution 2-issue design with 6-stage long integer pipelines, 32 32-bit GPR registers, 32 128-bit SIMD linear scalar registers, two 64-bit integer ALUs, 128-bit load-store unit (LSU) and a branch execution unit (BXU).

Instruction set: MIPS III, MIPS IV subset with Sony's proprietary 107 vector SIMD multimedia instructions (MMI). The custom instruction set was implemented by grouping the two 64-bit integer ALUs.
32-bit FPU coprocessor (COP1) with 6-stage long pipeline (floating point multiply accumulator × 1, floating point divider × 1). FPU is not IEEE compliant.

Simuladores para Assembly MIPS

- PCSPIM:
  - http://www.leandrocolevati.com.br/downloadmateriais?idFile=0ByaHylR4Cic0SG11WG1PMHRWRlk&arquivo=SPIMWin7.zip



```
 PC       = 00000000   EPC      = 00000000   Cause    = 00000000   BadVAddr= 00000000
 Status   = 3000ff10   HI       = 00000000   LO       = 00000000
                                 General Registers
R0  (r0) = 00000000  R8  (t0) = 00000000  R16 (s0) = 00000000  R24 (t8) = 00000000
R1  (at) = 00000000  R9  (t1) = 00000000  R17 (s1) = 00000000  R25 (t9) = 00000000
R2  (v0) = 00000000  R10 (t2) = 00000000  R18 (s2) = 00000000  R26 (k0) = 00000000
```

```
[0x00400000]    0x8fa40000  lw $4, 0($29)                ; 183: lw $a0 0($sp)          # argc
[0x00400004]    0x27a50004  addiu $5, $29, 4             ; 184: addiu $a1 $sp 4        # argv
[0x00400008]    0x24a60004  addiu $6, $5, 4              ; 185: addiu $a2 $a1 4        # envp
[0x0040000c]    0x00041080  sll $2, $4, 2                ; 186: sll $v0 $a0 2
[0x00400010]    0x00c23021  addu $6, $6, $2              ; 187: addu $a2 $a2 $v0
[0x00400014]    0x0c000000  jal 0x00000000 [main]        ; 188: jal main
```

```
        DATA
[0x10000000]...[0x10040000]     0x00000000

        STACK
[0x7fffeffc]                    0x00000000
[0x7ffff000]...[0x80000000]     0x00000000
```

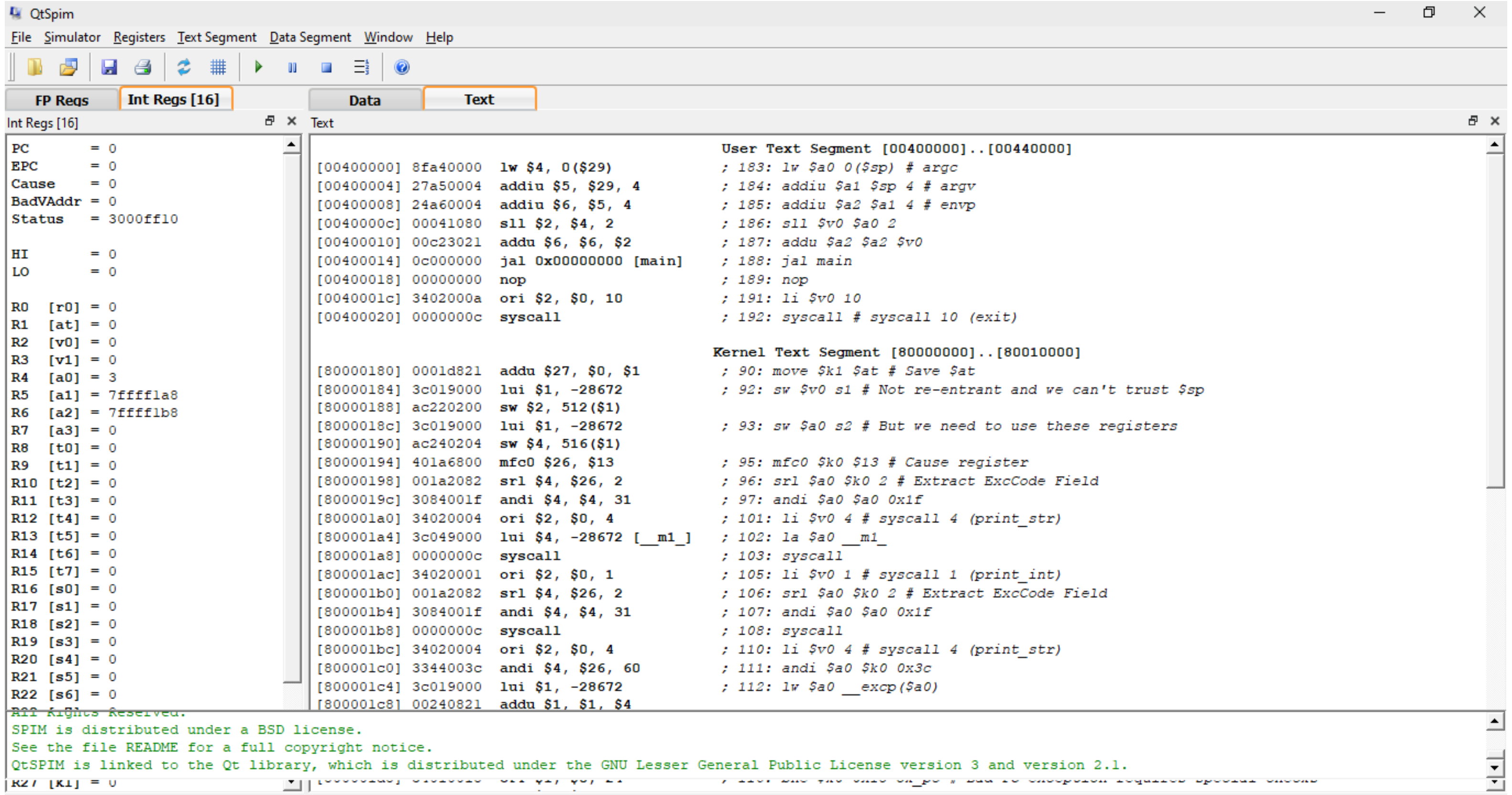
```
SPIM Version 9.1.4 of September 4, 2011
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
Loaded: C:\Program Files (x86)\PCSpim\exceptions.s
```

For Help, press F1

PC=0x00000000 EPC=0x00000000 Cause=0x00000000

Simuladores para Assembly MIPS

- QtSPIM:
  - https://sourceforge.net/projects/spimsimulator/files/



QtSpim
File Simulator Registers Text Segment Data Segment Window Help
FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text

```
PC       = 0
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 7ffff1a8
R6  [a2] = 7ffff1b8
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R27 [k1] = 0
```

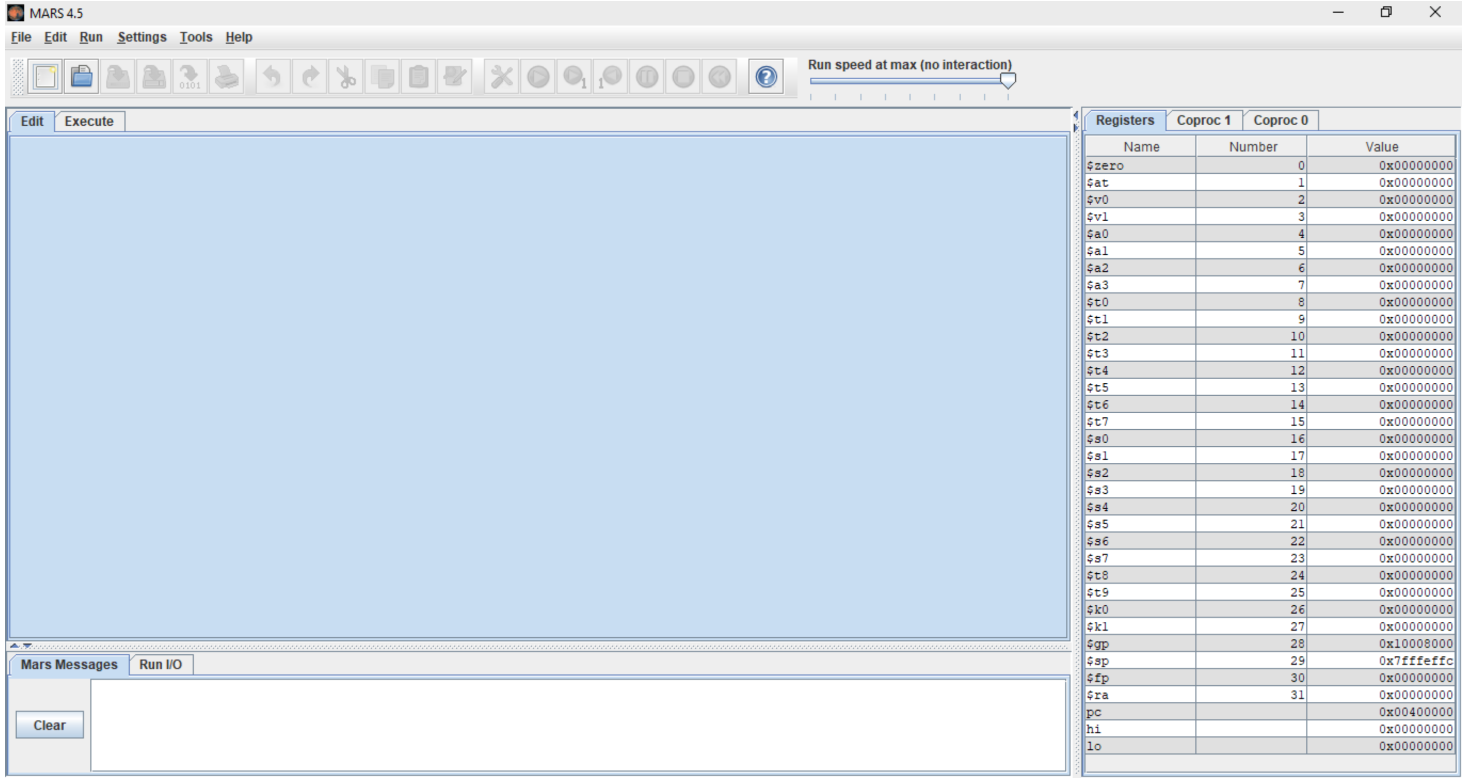
```
                                                User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)             ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $v0
[00400014] 0c000000  jal 0x00000000 [main]     ; 188: jal main
[00400018] 00000000  nop                       ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10            ; 191: li $v0 10
[00400020] 0000000c  syscall                   ; 192: syscall # syscall 10 (exit)

                                               Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31           ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                   ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4             ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60          ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672            ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
```

```
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.
```

Simuladores para Assembly MIPS

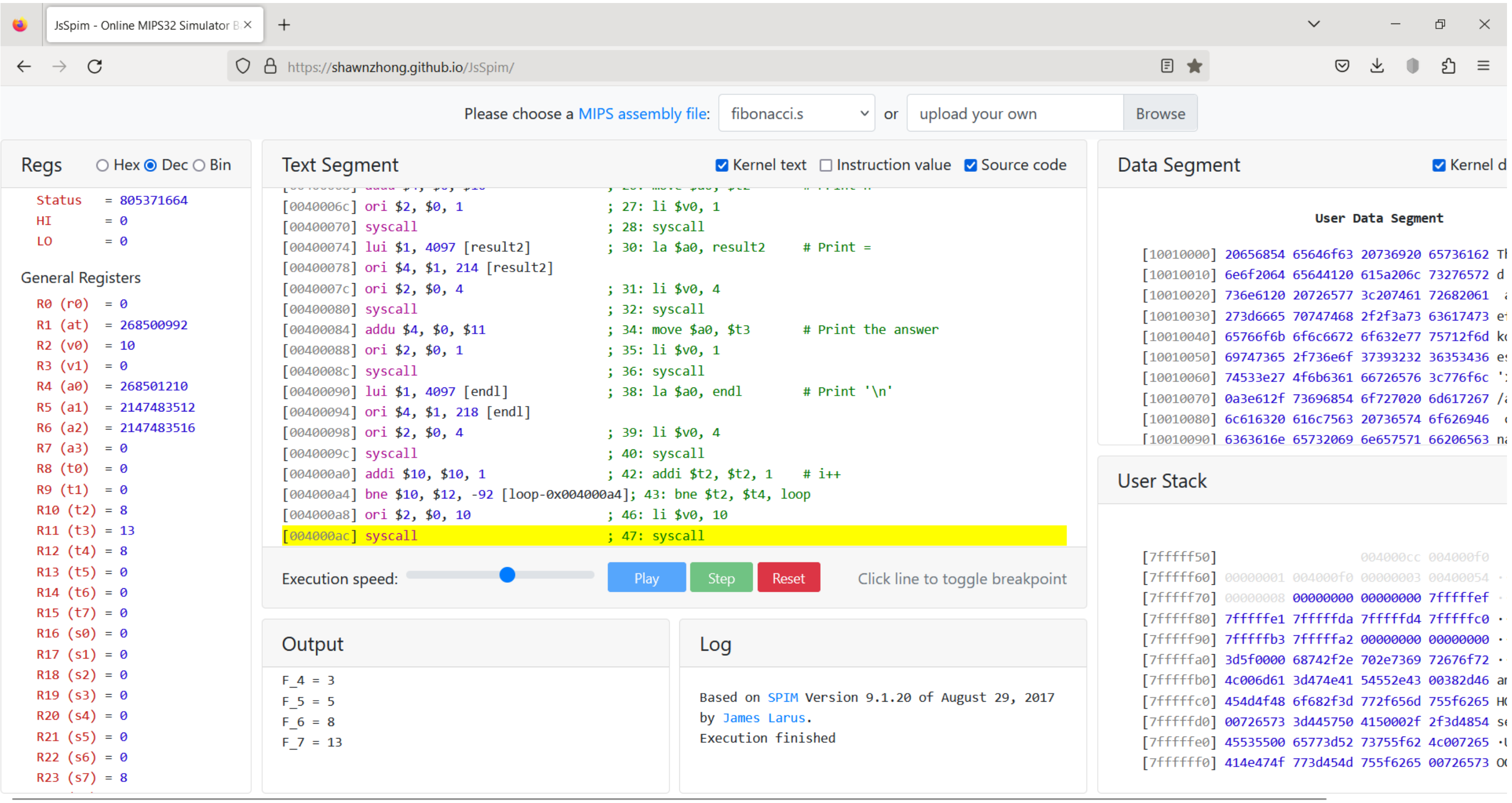
- Mars:
  - http://www.leandrocolevati.com.br/downloadmateriais?idFile=1XMma1gX-OAwFxxIMDtggZwUIkiK8Caz-&arquivo=SPIMMars4_5.zip



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Simuladores para Assembly MIPS

- JsSpim:
  - https://shawnzhong.github.io/JsSpim/



JsSpim - Online MIPS32 Simulator
https://shawnzhong.github.io/JsSpim/

Please choose a MIPS assembly file: fibonacci.s or upload your own Browse

Regs ○ Hex ● Dec ○ Bin

```
Status    = 805371664
HI        = 0
LO        = 0
```

General Registers

```
R0 (r0)  = 0
R1 (at)  = 268500992
R2 (v0)  = 10
R3 (v1)  = 0
R4 (a0)  = 268501210
R5 (a1)  = 2147483512
R6 (a2)  = 2147483516
R7 (a3)  = 0
R8 (t0)  = 0
R9 (t1)  = 0
R10 (t2) = 8
R11 (t3) = 13
R12 (t4) = 8
R13 (t5) = 0
R14 (t6) = 0
R15 (t7) = 0
R16 (s0) = 0
R17 (s1) = 0
R18 (s2) = 0
R19 (s3) = 0
R20 (s4) = 0
R21 (s5) = 0
R22 (s6) = 0
R23 (s7) = 8
```

Text Segment ☑ Kernel text ☐ Instruction value ☑ Source code

```
[0040006c] ori $2, $0, 1                 ; 27: li $v0, 1
[00400070] syscall                       ; 28: syscall
[00400074] lui $1, 4097 [result2]        ; 30: la $a0, result2     # Print =
[00400078] ori $4, $1, 214 [result2]
[0040007c] ori $2, $0, 4                 ; 31: li $v0, 4
[00400080] syscall                       ; 32: syscall
[00400084] addu $4, $0, $11              ; 34: move $a0, $t3       # Print the answer
[00400088] ori $2, $0, 1                 ; 35: li $v0, 1
[0040008c] syscall                       ; 36: syscall
[00400090] lui $1, 4097 [endl]           ; 38: la $a0, endl        # Print '\n'
[00400094] ori $4, $1, 218 [endl]
[00400098] ori $2, $0, 4                 ; 39: li $v0, 4
[0040009c] syscall                       ; 40: syscall
[004000a0] addi $10, $10, 1              ; 42: addi $t2, $t2, 1    # i++
[004000a4] bne $10, $12, -92 [loop-0x004000a4]; 43: bne $t2, $t4, loop
[004000a8] ori $2, $0, 10                ; 46: li $v0, 10
[004000ac] syscall                       ; 47: syscall
```

Execution speed: Play Step Reset Click line to toggle breakpoint

Data Segment ☑ Kernel d

```
                  User Data Segment

[10010000] 20656854 65646f63 20736920 65736162 Th
[10010010] 6e6f2064 65644120 615a206c 73276572 d
[10010020] 736e6120 20726577 3c207461 72682061  a
[10010030] 273d6665 70747468 2f2f3a73 63617473 ef
[10010040] 65766f6b 6f6c6672 6f632e77 75712f6d ko
[10010050] 69747365 2f736e6f 37393232 36353436 es
[10010060] 74533e27 4f6b6361 66726576 3c776f6c ':
[10010070] 0a3e612f 73696854 6f727020 6d617267 /a
[10010080] 6c616320 616c7563 20736574 6f626946  c
[10010090] 6363616e 65732069 6e657571 66206563 na
```

User Stack

```
[7ffff50]                    004000cc 004000f0
[7ffff60] 00000001 004000f0 00000003 00400054 ·
[7ffff70] 00000008 00000000 00000000 7ffffef ·
[7ffff80] 7ffffe1 7ffffda 7ffffd4 7ffffc0 ·
[7ffff90] 7ffffb3 7ffffa2 00000000 00000000 ·
[7ffffa0] 3d5f0000 68742f2e 702e7369 72676f72 ·
[7ffffb0] 4c006d61 3d474e41 54552e43 00382d46 ar
[7ffffc0] 454d4f48 6f682f3d 772f656d 755f6265 HO
[7ffffd0] 00726573 3d445750 4150002f 2f3d4854 se
[7ffffe0] 45535500 65773d52 73755f62 4c007265 ·U
[7fffff0] 414e474f 773d454d 755f6265 00726573 OG
```

Output

```
F_4 = 3
F_5 = 5
F_6 = 8
F_7 = 13
```

Log

```
Based on SPIM Version 9.1.20 of August 29, 2017
by James Larus.
Execution finished
```