

PC SPIM

Visão Preliminar

PC SPIM

The screenshot displays the PCSpim simulator window. At the top, the menu bar includes File, Simulator, Window, and Help. Below the menu is a toolbar with icons for file operations and simulation control. The main window is divided into several sections:

- Registers:** A table showing the current values of 32 general-purpose registers (R0-R31). The registers are organized into columns. The first column shows registers R0-R7, the second R8-R15, the third R16-R23, and the fourth R24-R31. Each register entry includes its name (e.g., R0, R1), its alias (e.g., r0, at), and its current value in hexadecimal.
- Instructions:** A list of instructions being executed, each with its address, opcode, and assembly code. The instructions are color-coded (blue for instructions, green for comments). The instructions shown include `lw $4, 0($29)`, `addiu $5, $29, 4`, `addiu $6, $5, 4`, `sll $2, $4, 2`, `addu $6, $6, $2`, `jal 0x00400024 [main]`, `nop`, `ori $2, $0, 10`, `syscall`, and `ori $8, $0, 5`.
- DATA:** A section showing memory contents at various addresses. The data is organized into columns, with the first column showing addresses and the subsequent columns showing the data values in hexadecimal.
- Status Bar:** At the bottom, it displays the current PC (Program Counter) value, EPC (Exception Program Counter), and Cause.

Arrows from the text labels point to the corresponding sections in the simulator window:

- Registradores:** Points to the General Registers section.
- Registradores de uso geral (Inteiros e Caracteres):** Points to the General Registers section.
- Set de Instruções:** Points to the Instructions section.

PC SPIM

Serão livres para utilização:

- Registradores de t0 a t9
- Registradores de s0 a s7

Serão reservados:

- v0, para códigos de leitura e escrita
- a0, para sequência de caracteres oriundas do .data

PC SPIM

The screenshot shows the PC SPIM simulator window. At the top is a menu bar with 'File', 'Simulator', 'Window', and 'Help'. Below it is a toolbar with icons for file operations and simulation control. The main window is divided into three panes. The top pane, titled 'Single Floating Point Registers', displays a grid of 32 registers (FP0 to FP31), all containing the value 0.000000. The middle pane shows assembly code with addresses from 0x00400000 to 0x00400024. The bottom pane shows a memory dump from 0x10000000 to 0x10010060. A status bar at the bottom indicates the current PC, EPC, and Cause values.

```
File Simulator Window Help
[Icons]

Single Floating Point Registers
FP0 = 0.000000  FP8 = 0.000000  FP16 = 0.000000  FP24 = 0.000000
FP1 = 0.000000  FP9 = 0.000000  FP17 = 0.000000  FP25 = 0.000000
FP2 = 0.000000  FP10 = 0.000000  FP18 = 0.000000  FP26 = 0.000000
FP3 = 0.000000  FP11 = 0.000000  FP19 = 0.000000  FP27 = 0.000000
FP4 = 0.000000  FP12 = 0.000000  FP20 = 0.000000  FP28 = 0.000000
FP5 = 0.000000  FP13 = 0.000000  FP21 = 0.000000  FP29 = 0.000000
FP6 = 0.000000  FP14 = 0.000000  FP22 = 0.000000  FP30 = 0.000000
FP7 = 0.000000  FP15 = 0.000000  FP23 = 0.000000  FP31 = 0.000000

[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] 0x0c100009 jal 0x00400024 [main] ; 188: jal main
[0x00400018] 0x00000000 nop ; 189: nop
[0x0040001c] 0x3402000a ori $2, $0, 10 ; 191: li $v0 10
[0x00400020] 0x0000000c syscall ; 192: syscall # syscall 10 (exit)
[0x00400024] 0x34080005 ori $8, $0, 5 ; 8: li $t0, 5

DATA
[0x10000000]...[0x10010000] 0x00000000
[0x10010000] 0x72204f0a 0x6c757365 0x6f646174 0x20616420
[0x10010010] 0x616d6f73 0x0a00203a 0x6572204f 0x746c7573
[0x10010020] 0x206f6461 0x6d206164 0x69746c75 0x63696c70
[0x10010030] 0xc3a7c361 0x203a6fa3 0x6e490a00 0x61726973
[0x10010040] 0x206d7520 0x65746e69 0x3a6f7269 0x490a0020
[0x10010050] 0x7269736e 0x6d752061 0x61657220 0x00203a6c
[0x10010060]...[0x10040000] 0x00000000

Memory and registers cleared and the simulator reinitialized.

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
C:\Users\PC\Desktop\exemplo1.s successfully loaded

For Help, press F1 PC=0x00400000 EPC=0x00000000 Cause=0x00000000
```

Registradores

Registradores de ponto flutuante de precisão Single (reais)

PC SPIM

Serão livres para utilização:

- Registradores de f1 a f11
- Registradores de f13 a f31

Serão reservados:

- f12, para escrita de pontos flutuantes
- f0, para leitura de pontos flutuantes

PC SPIM

Primeiros comandos:

- Atribuição de inteiros:

li pos, valor

Ex.: li \$t0, 3

- Atribuição de reais:

li.s pos, valor

Ex.: li \$f1, 5.3

- Operações Fundamentais Inteiros:

Soma: add

Subtração: sub

Multiplicação: mul

Divisão: div

Resto: rem

- Exemplo: Operação de \$t4 e \$t5
com resultado em \$t0:

add \$t0, \$t4, \$t5

sub \$t0, \$t4, \$t5

mul \$t0, \$t4, \$t5

div \$t0, \$t4, \$t5

rem \$t0, \$t4, \$t5

- Operações Fundamentais Reais:

Soma: add.s

Subtração: sub.s

Multiplicação: mul.s

Divisão: div.s

- Exemplo: Operação de \$f4 e \$f5
com resultado em \$f1:

add.s \$f1, \$f4, \$f5

sub.s \$f1, \$f4, \$f5

mul.s \$f1, \$f4, \$f5

div.s \$f1, \$f4, \$f5

PC SPIM

Códigos para \$v0:

1: Escrita de Inteiros

2: Escrita de Reais (Single)

4: Escrita de Caracteres

5: Leitura de Inteiros

6: Leitura de Reais (Single)

- Escrita de caracteres do .data:

```
li $v0, 4  
la $a0, texto1  
syscall
```

- Escrita de Inteiros:

```
li $v0, 1  
add $a0, $t2, $zero  
syscall
```

*Passando para a0 o valor do registrador que será impresso

- Escrita de Reais:

```
li $v0, 2  
mov.s $f12, $f3  
syscall
```

*Passando para f0 o valor do registrador que será impresso

- Leitura de Inteiros:

```
li $v0, 5  
syscall  
add $t3, $v0, $zero
```

* Passando para \$t3 o valor lido em a0

- Leitura de Reais:

```
li $v0, 6  
syscall  
mov.s $f31, $f0
```

* Passando para \$f31 o valor lido em \$f0

PC SPIM

Para criar estruturas:

- Jump:
 j label

- Branches:

Branch se igual	beq \$s1, \$s2, L	if(\$s1 == \$s2) go to L
Branch se maior ou igual	bge \$rs,\$rt,Label	if(\$rs>=\$rt) PC=Label
Branch se maior	bgt \$rs,\$rt,Label	if(\$rs>\$rt) PC=Label
Branch se módulo maior	bgtu \$rs,\$rt,Label	if(\$rs >= rt) PC=Label
Branch se maior que zero	bgtz \$rs,\$rt,Label	if(\$rs>0) PC=Label
Branch se menor ou igual	ble \$rs,\$rt,Label	if(\$rs<=\$rt) PC=Label
Branch se menor	blt \$rs,\$rt,Label	if(\$rs<\$rt) PC=Label
Branch se diferente	bne \$s1, \$s2, L	if(\$s1 != \$s2) go to L