MIPS

Instrucciones
LW SW

MIPS. INSTRUCCIONES DE CARGA/ ALMACENAMIENTO

Son instrucciones que leen y escriben en memoria utilizando registros.

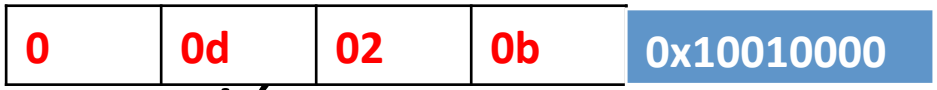
- lw $t0, dir: carga en el registro t0 el contenido de la palabra de memoria cuya dirección es dir
  - t0←m[dir]
- sw $t0, dir: almacena en m[dir] el contenido de t0
  - m[dir]←t0

MIPS. INSTRUCCIONES DE CARGA

```
        .data
dir:    .byte 0x10010000
```

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0d	02	0b

MIPS. INSTRUCCIONES DE CARGA



- Valor tras ejecutar la instrucción

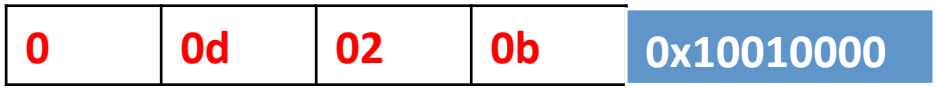
  lw $t0, dir

  $t0= 0x000b020d

dir es una etiqueta que indica la dirección 0x10010000

Es la dirección del byte 0x0d (primera posición de la palabra de memoria)

MIPS. INSTRUCCIONES DE CARGA



- La instrucción

  lb $t0, dir + 2

  $t0= 0x0000000b

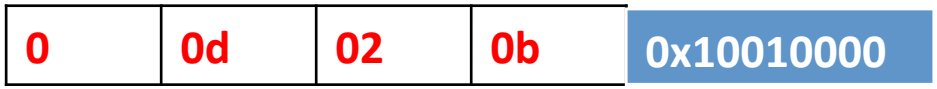
- La instrucción

  la $t0, dir

carga en t0 la dirección de memoria donde se almacena dir:

$t0= 0x10010000

MIPS. INSTRUCCIONES DE CARGA



Instrucción	Memoria	Resultado
lw $t0, dir	dir:.byte 0xd, 2, 11	$t0= 0x000b020d
lb $t0, dir + 2		$t0= 0x0000000b
la $t0, dir		$t0= 0x10010000

MIPS. INSTRUCCIONES DE ALMACENAMIENTO

Son instrucciones que leen y escriben en memoria utilizando registros.

- sw $t0, dir: almacena el contenido de la palabra de memoria cuya dirección es dir en el registro t0.

MIPS. INSTRUCCIONES DE CARGA/ ALMACENAMIENTO

Datos en registro y guardados en memoria:

```
.data
dir:        .byte 1,2,3,4
```

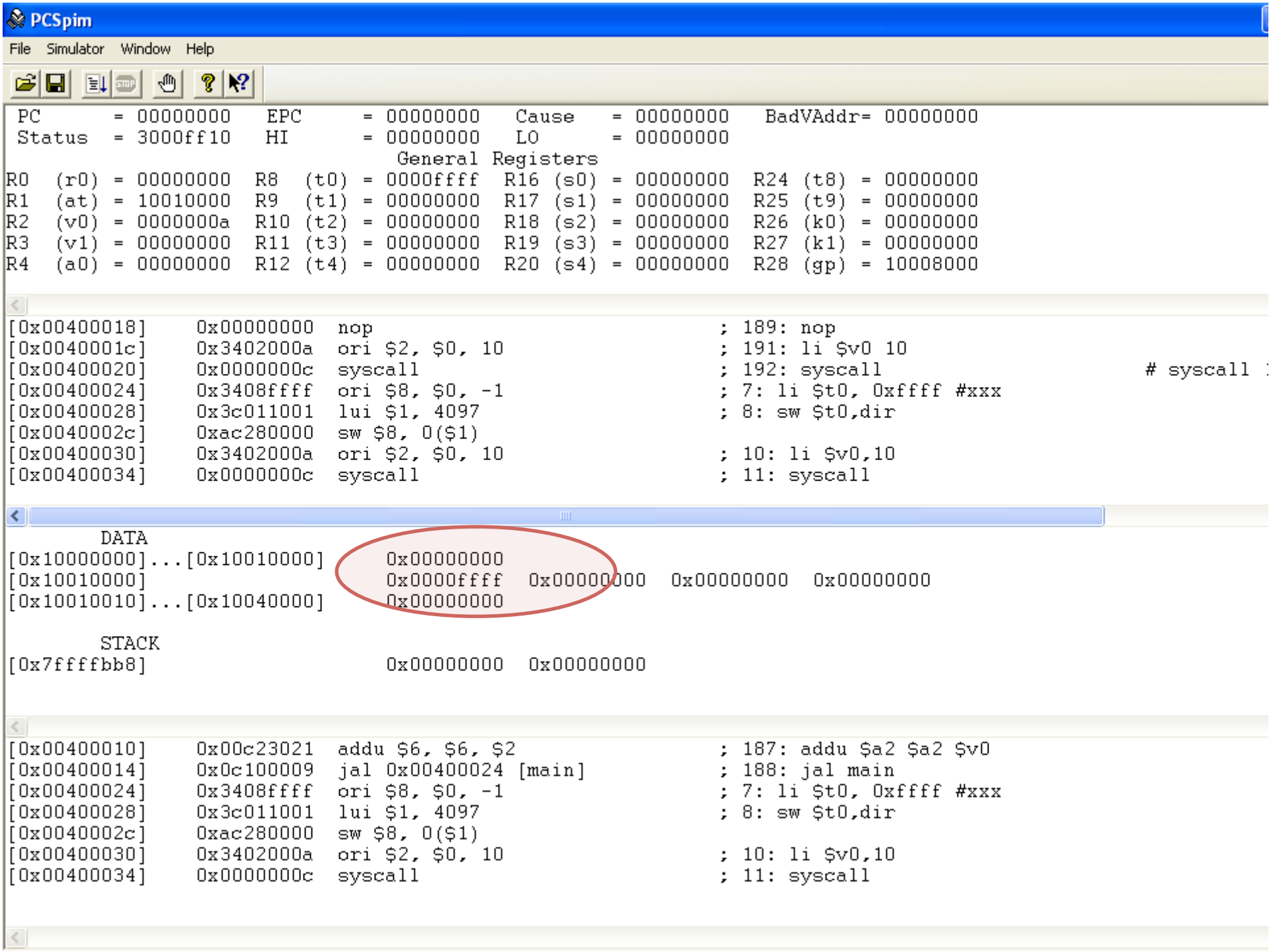
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
04	03	02	01

← **0x10010000**

MIPS. INSTRUCCIONES DE CARGA/ ALMACENAMIENTO

```
.text
main:
    li $t0, 0xffff
    sw $t0,dir

    li $v0,10       # Fin ejecución
    syscall
```


PCSpim
File Simulator Window Help
PC = 00000000 EPC = 00000000 Cause = 00000000 BadVAddr= 00000000
Status = 3000ff10 HI = 00000000 LO = 00000000
General Registers
R0 (r0) = 00000000 R8 (t0) = 0000ffff R16 (s0) = 00000000 R24 (t8) = 00000000
R1 (at) = 10010000 R9 (t1) = 00000000 R17 (s1) = 00000000 R25 (t9) = 00000000
R2 (v0) = 0000000a R10 (t2) = 00000000 R18 (s2) = 00000000 R26 (k0) = 00000000
R3 (v1) = 00000000 R11 (t3) = 00000000 R19 (s3) = 00000000 R27 (k1) = 00000000
R4 (a0) = 00000000 R12 (t4) = 00000000 R20 (s4) = 00000000 R28 (gp) = 10008000
[0x00400018] 0x00000000 nop ; 189: nop
[0x0040001c] 0x3402000a ori $2, $0, 10 ; 191: li $v0 10
[0x00400020] 0x0000000c syscall ; 192: syscall # syscall
[0x00400024] 0x3408ffff ori $8, $0, -1 ; 7: li $t0, 0xffff #xxx
[0x00400028] 0x3c011001 lui $1, 4097 ; 8: sw $t0,dir
[0x0040002c] 0xac280000 sw $8, 0($1)
[0x00400030] 0x3402000a ori $2, $0, 10 ; 10: li $v0,10
[0x00400034] 0x0000000c syscall ; 11: syscall
DATA
[0x10000000]...[0x10010000] 0x00000000
[0x10010000] 0x0000ffff 0x00000000 0x00000000 0x00000000
[0x10010010]...[0x10040000] 0x00000000
STACK
[0x7fffbb8] 0x00000000 0x00000000
[0x00400010] 0x00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
[0x00400014] 0x0c100009 jal 0x00400024 [main] ; 188: jal main
[0x00400024] 0x3408ffff ori $8, $0, -1 ; 7: li $t0, 0xffff #xxx
[0x00400028] 0x3c011001 lui $1, 4097 ; 8: sw $t0,dir
[0x0040002c] 0xac280000 sw $8, 0($1)
[0x00400030] 0x3402000a ori $2, $0, 10 ; 10: li $v0,10
[0x00400034] 0x0000000c syscall ; 11: syscall

sw $t0, dir

Partiendo de esta situación de la memoria (ver tabla) supongamos que el contenido del registro $t0 es 0x010f123a

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	02	0d

- Ejecutamos sw $t0, dir+8

sw $t0, dir

- Ejecutamos sw $t0, dir+8

La situación de la memoria después de la ejecución de esta instrucción es:

00	00	00	00
01	0f	12	3a
00	00	00	00
00	0b	02	0d

sb $t0, dir+1

$t0 = 0x010f123a

Antes:

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	02	0d

Después:

00	00	00	00
00	00	00	00
00	00	00	00
00	0b	3a	0d