Estructura de computadores- Práctica 2

Laura Álvarez Flórez 100363965
Grupo 82
Laura Yunta García 100363785
Grupo 82

Grado en ingeniería informática

Contenido

1. Ejercicio 1.

1.

<table>
<tr><th>Instrucción</th><th>Ejemplo</th><th>Estado modificado</th><th></th></tr>
<tr><td rowspan="3">add R1 R2 R3</td><td>li $t0 2
li $t1 1
add $t2 $t0 $t1</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R2</td><td>= 0</td><td>0x2</td></tr><tr><td>cpu</td><td>R3</td><td>= 0</td><td>0x1</td></tr><tr><td>cpu</td><td>R5</td><td>= 0</td><td>0x3</td></tr></table></td><td rowspan="3">Sí</td></tr>
<tr><td>li $t0 2
li $t1 -7
add $t2 $t0 $t1</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x2</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xfffffff9</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xfffffffb</td></tr><tr><td>cpu</td><td>SR</td><td>= 0</td><td>0x20000000</td></tr></table></td></tr>
<tr><td>li $t0 -2
li $t1 -10
add $t2 $t0 $t1</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xfffffff6</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xfffffff4</td></tr><tr><td>cpu</td><td>SR</td><td>= 0</td><td>0xa0000000</td></tr></table></td></tr>
</table>

2.

<table>
<tr><td>sub R1 R2 R3</td><td>li $t0 2
li $t1 1
sub $t2 $t0 $t1</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x2</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x1</td></tr></table></td><td>Si</td></tr>
</table>

<table>
<tr>
<td></td>
<td>li $t0 -2
li $t1 1
sub $t2 $t0 $t1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xfffffffd</td></tr>
<tr><td>cpu</td><td>SR</td><td>= 0</td><td>0x20000000</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td></td>
<td>li $t0 -2
li $t1 -1
sub $t2 $t0 $t1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>clipboard state</th><th>selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xffffffff</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xffffffff</td></tr>
<tr><td>cpu</td><td>SR</td><td>= 0</td><td>0xa0000000</td></tr>
</table>
</td>
<td></td>
</tr>
</table>

3.

<table>
<tr>
<td rowspan="3">mul R1 R2 R3</td>
<td>li $t0 2
li $t1 1
mul $t2 $t0 $t1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x2</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x2</td></tr>
</table>
</td>
<td rowspan="3">Si</td>
</tr>
<tr>
<td>li $t0 -2
li $t1 1
mul $t2 $t0 $t1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xfffffffe</td></tr>
<tr><td>cpu</td><td>SR</td><td>= 0</td><td>0x20000000</td></tr>
</table>
</td>
</tr>
<tr>
<td>li $t0 -2
li $t1 -1
mul $t2 $t0 $t1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xffffffff</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x2</td></tr>
</table>
</td>
</tr>
</table>

4.

<table>
<tr><td rowspan="2">lb R1 (R2)</td><td>li $t0 2
li $t1 0x0004
sw $t0 ($t1)
lb $t2 ($t1)</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8010</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x2</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x2</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0x2</td></tr></table></td><td rowspan="2">Si</td></tr>
<tr><td>li $t0 -2
li $t1 0x0004
sw $t0 ($t1)
lb $t2 ($t1)</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8010</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xfffffffe</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xfe</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0xfffffffe</td></tr></table></td></tr>
</table>

5.

<table>
<tr><td rowspan="2">li R inm</td><td>li $t0 100</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8004</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x64</td></tr></table></td><td rowspan="2">Sí</td></tr>
<tr><td>li $t0 -100</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8004</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xffffff9c</td></tr></table></td></tr>
</table>

6.

<table>
<tr><td>lb R1 (R2)</td><td>li $t0 0x0004
li $t1 1
sw $t1 ($t0)
lbu $t2 ($t0)</td><td><table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8010</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x1</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0x1</td></tr></table></td><td>Sí</td></tr>
</table>

	li $t0 0x0004 li $t1 -1 sw $t1 ($t0) lbu $t2 ($t0)	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8010</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xffffffff</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0xff</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0xffffffff</td></tr></table>	

7.

lbu R1 (R2)	li $t0 0x0004 li $t1 1 sb $t1 ($t0)	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0x1</td></tr></table>	Sí
	li $t0 0x0004 li $t1 -1 sb $t1 ($t0)	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x800c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x4</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0xffffffff</td></tr><tr><td>memory</td><td>0x4</td><td>= 0</td><td>0xff</td></tr></table>	

8.

Sb R1 (R2)	b 0x0004	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x4</td></tr></table>	Sí
	B 0xffff	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0xffff</td></tr><tr><td>memory</td><td>0x8000</td><td>= 0x30008000</td><td>0x30007ffb</td></tr></table>	

9.

	li $t0 0 li $t1 3 li $t2 1 bucle: beq $t0 $t1 fin add $t0 $t0 $t2 b bucle fin:	<table><tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr><tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x801c</td></tr><tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x3</td></tr><tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x3</td></tr><tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x1</td></tr></table>	Sí

<table>
<tr>
<td>beq R1 R2 addr</td>
<td>li $t3 1</td>
<td>
<table>
<tr><td>cpu</td><td>R11</td><td>= 0</td><td>0x1</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td></td>
<td>li $t0 1
li $t1 1
li $t2 1

bucle:
beq $t0 $t1 fin
add $t0 $t0 $t2
b bucle

fin:
li $t3 1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x801c</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R11</td><td>= 0</td><td>0x1</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td></td>
<td>li $t0 3
li $t1 1
li $t2 1

bucle:
beq $t0 $t1 fin
add $t0 $t0 $t2
b bucle

fin:
li $t3 1</td>
<td>
<table>
<tr><th>Type</th><th>Identification</th><th>Clipboard state</th><th>Selected state</th></tr>
<tr><td>cpu</td><td>PC</td><td>= 0x8000</td><td>0x8014</td></tr>
<tr><td>cpu</td><td>R8</td><td>= 0</td><td>0xc7</td></tr>
<tr><td>cpu</td><td>R9</td><td>= 0</td><td>0x1</td></tr>
<tr><td>cpu</td><td>R10</td><td>= 0</td><td>0x1</td></tr>
</table>
#El programa comienza un bucle infinito donde el valor contenido en t0 se incrementa en cada instrucción hasta que toda la memoria es utilizada.
</td>
<td></td>
</tr>
</table>

- En este primer ejercicio hemos analizado una a una las instrucciones propuestas, las comprobaciones para detectar si el funcionamiento es el correcto se basan en el uso de diferentes valores (tanto positivos como negativos) para comprobar que las instrucciones se ajustaban de forma correcta a todos los casos posibles.
- El micro código no ha sido modificado durante la realización de este ejercicio.

2. Ejercicio 2.

strlen_1:

Ejemplo utilizado: cadena “hola”

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8000	0x8034
cpu	R2	= 0	0x4
cpu	R4	= 0	0x1000
cpu	R10	= 0	0x1
cpu	R11	= 0	0x4

	cpu	R12	= 0	0x1004

Como podemos ver en R11 (lugar donde tenemos estipulado que almacene el número de caracteres) obtenemos un 4 que es el número esperado en la ejecución

CLK ticks	215

Ejemplo utilizado: “ensamblador”

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8000	0x8034
cpu	R2	= 0	0xb
cpu	R4	= 0	0x1000
cpu	R10	= 0	0x1
cpu	R11	= 0	0xb
cpu	R12	= 0	0x100b

Como podemos ver en R11 (lugar donde tenemos estipulado que almacene el número de caracteres) obtenemos una b, lo que corresponde a un 11 en decimal, que es el número esperado en la ejecución.

CLK ticks	481

skipspaces_1

Ejemplo utilizado: " hola"

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8000	0x8030
cpu	R2	= 0	0x1002
cpu	R4	= 0	0x1000
cpu	R8	= 0	0x68
cpu	R9	= 0	0x20
cpu	R10	= 0	0x1
cpu	R12	= 0	0x1002

Analizando la tabla anterior, se puede observar que el resultado en el registro 2, el correspondiente a $v0 es 1002, la dirección de inicio de la cadena. Siendo el comienzo la dirección 1000 y conteniendo esta dos espacios, el resultado concuerda con lo esperado.

CLK ticks	129

Ahora añadiremos un espacio más a la cadena, siendo el input “ hola” obtenemos los siguientes resultados:

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8004	0x8030
cpu	R2	= 0	0x1003
cpu	R8	= 0	0x68

cpu	R9	= 0	0x20
cpu	R10	= 0	0x1
cpu	R12	= 0	0x1003

La dirección se ha incrementado en 1, lo que nos indica que el programa ha detectado el nuevo espacio y lo ha saltado correctamente.

CLK ticks	161

- Comparando los datos de los ciclos de reloj del ejercicio 2 y el 3 obtenemos que en el ejercicio 2 para realizar la misma operación se utilizan un número muy superior de ciclos al de las instrucciones del ejercicio 3. Por tanto deducimos que las instrucciones que hemos microprogramado en el ejercicio 3 son mucho más eficaces que las programadas en el ejercicio 2.

3. Ejercicio 3.

- **strlen_2 reg**

Esta función lee cada uno de los bytes y los analiza independientemente, el algoritmo sabe que la cadena ha finalizado una vez encuentra un numero negativo, ya que el valor más pequeño de la tabla ascii es el 0, el mismo que indica que la cadena ha finalizado (por convenio).

C0: R2→RT2 #La dirección es almacenada
C1: RT2→MAR #Se guarda la misma para su posterior lectura

BUCLE:
C2: MP→MBR #Se realiza la lectura de la direccion puntual
C3: MBR→RT1 #Se almacena el contenido de la misma
C4: RT1-1 #Se resta uno a la misma y la ALU analiza el resultado
dándose dos casos posibles:

#Si el resultado >0
C5: RT2→RT1 #Se mueve el contenido
C6: RT1+1→RT2 #Incrementamos en 1 la dir.

C7: RT2→MAR #Colocamos la dirección para su posterior lectura

C8: B BUCLE #Repetimos el proceso

#Si el resultado es <0

C5': RT2→RT1

C6': RT1-RT2→R1 #Restamos la dirección actual menos la inicial para calcular cuantos char hemos leido

- **skipasciicode_2 $r1 $r2 n**

Con esta función se leen de nuevo uno a uno os caracteres de la cadena, dado un char obtenido por valor, cuando se detecta que son iguales (la resta de ambos es 0) se continua a la siguiente dirección mientras que si son diferentes se desvia el flujo hacia el final del programa donde se devuelve la dirección en la que nos encontramos en ese instante.

C0: R2→RT1 #Almacenamos la dirección
C1: RT1→MAR #Colocamos la misma para su posterior lectura
C2: Val→k1 #Guardamos el valor en un registro auxiliar k1

Bucle:
C3:MAR→MP #Realizamos la lectura del carácter
MP→MBR
C4:MBR→RT2 #Lo almacenamos en r2
C5:k1-RT2 #Restamos el valor y el carácter dado y procedemos al análisis

#Si son iguales
C6:RT1+1→RT1 #Aumentamos uno el valor donde esta la dir
C7:RT1→MAR #Lo colocamos para la posterior lectura
B bucle #Repetimos el proceso

#Si son distintos
Fin:
C6':RT1→R1 #Devolvemos la dirección encontrada

Microinstrucción
strlen_2 reg reg

Ejemplo a comprobar: "hola"

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8000	0x800c
cpu	R9	= 0	0x4
cpu	R10	= 0	0x1000

CLK ticks	60

Se cumple los requisitos esperados ya que obtenemos que nos cuenta todos los caracteres de la cadena y viene reflejado en el registro R9, en este caso 4.

Ejemplo a comprobar:

Type	Identification	Values in the clipboard state	Values in the selected state
cpu	PC	= 0x8000	0x800c
cpu	R9	= 0	0xb
cpu	R10	= 0	0x1000

“ensamblador”	CLK ticks 116 Se cumple los requisitos esperados ya que obtenemos que nos cuenta todos los caracteres de la cadena y viene reflejado en el registro R9 en este caso 11 (Equivalencia decimal a b).

Microinstrucción:
skipasciicode_2 $r1 $r2 n

Ejemplo: Cadena “llaura” value:”l”	Type / Identification / Values in the clipboard state / Values in the selected state cpu PC = 0x8000 0x800c cpu R9 = 0 0x1002 cpu R10 = 0 0x1000 cpu R27 = 0 0x6c CLK ticks 38 En esta instrucción el resultado que obtenemos en R9 es 1002, ya que se detecta una primera ‘l’, se salta y posteriormente encontramos una segunda ‘l’ que también ignora. Posicionados en la dirección 1002 reconoce un char diferente y devuelve el resultado de esa dirección.
Cadena: “ hola” Value: “ “	Type / Identification / Values in the clipboard state / Values in the selected state cpu PC = 0x8000 0x800c cpu R9 = 0 0x1002 cpu R10 = 0 0x1000 cpu R27 = 0 0x20 cpu SR = 0 0x60000000 CLK ticks 38

4. Ejercicio 4.

	Palabra	Ensamblador	Microprogramado
Strlen	‘hola’	215	60
	‘ensamblador’	481	116
SkipSpace	‘ hola’	129	38

- Gracias a estas comparaciones podemos ver que son más eficientes las microprogramaciones de las instrucciones que las programaciones en ensamblador debido a que se han obtenido unos valores de tiempo de ejecución mucho menores en los primeros que en los segundos, por tanto se recomienda el uso del microprogramado para futuros programas y proyectos empresariales ya que asegura un tiempo para el trabajo y el proceso mucho menor. Ya que aunque en estas prácticas con un número pequeño de datos y de pruebas no sea muy notable la diferencia, en el ámbito laboral (con un alto nivel de datos utilizados) sí que puede llegar a se un inconveniente el mal uso de uno u otro. Por tanto siempre recomendaremos el uso del microprogramado.

  Para llegar a esta conclusión hemos analizado las dos funciones Strlen realizadas, primero con la palabra hola y finalmente con la palabra ensamblador, como se infiere de la tabla en ambos casos el resultado es proporcional. Cuanto mayor es la palabra mayor es el resultado de ambas opciones siendo siempre inferior el tiempo en el método de la instrucción micro programada.
  Para el caso de SkipSpace hemos usado la función de ejercicio dos correspondiente a la misma y además para aprovechar la microinstrucción SkipAsciiCode, hemos analizado el caso ayudándonos de esta introduciendo la misma cadena y un espacio ‘ ‘ como valor.
  Llegamos a la misma conclusión que con la primera función.

  Para finalizar, siendo la voluntad de la empresa un mínimo de una reducción al 20% de los ciclos para optar por la opción microprogramada, se puede observar que en todos los casos los ciclos

reducen su número en más de un 50%. Dado que hemos superado el porcentaje con creces se recomienda a la institución el uso de la microprogramación incluso siendo esta más costosa.

5. Conclusiones y problemas encontrados.

- Es importante destacar que las instrucciones de Strlen en los ejercicios tanto en el 2 como en el tres solo paran cuando la cadena ha terminado, es decir cuando llega a un 0 por tanto los espacios entre palabras también los tiene en cuenta como carácteres.

  Según llevábamos acabo el desarrollo de las microinstrucciones hemos llegado a la conclusión de que aun siendo una mejor opción como hemos determinado en el punto número 4, sí que es cierto que es una programación más lenta y más compleja, la cual dificulta el manejo de excepciones en comparación con la programación con instrucciones en ensamblador.

  Otra dificultad a comentar, surgió durante el desarrollo de las instrucciones del punto 3, ya que inicialmente realizábamos el bucle de manera incorrecta y el programa WepSIM nunca salía de este, por lo que se colapsaba constantemente la página. Finalmente resolvimos este incidente.