Tecnológico de Monterrey

School of Engineering and Information Technology
Department of Electrical and Computer Engineering
Computer Architecture

LAB 2. C to MIPS

Héctor Javier Pequeño Chairez **A01246364**
Gabriela Jazmín Álvarez Espinoza **A00825719**

Translate the following C program to MIPS:

Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays **R** and **E** are in registers $s6 and $s7, respectively.

Code in C: f = g + E[R[4]-R[3]]

$s0 -> f

$s1 -> g : 0x41

$s2 -> h

$s3 -> i

$s4 -> j

$s6 -> Base Address array_R : 0x10010010

array_R[4] = 0x4*4 = 0x10 = 16

array_R[3] = 0x4*3 = 0x0C = 12

$s7 -> Base Address array_E : 0x10010040

Execution Code: (Código Utilizado en el simulador)

```
lw $t0, 16($s6)          # $t0 = R[4]

lw $t1, 12($s6)          # $t1 = R[3]

sub $t0, $t0, $t1        # $t0 = $t0 - $t1 = R[4] - R[3]

sll $t0, $t0, 2          # 2^2 = 4, entonces multiplicamos por 4, para tener correctamente los bytes. Entonces tenemos que $t0 = $t0*4

add $t0, $t0, $s7        # $t0 = dir de E[R[4] + R[3]]

lw $t1 0($t0)            # Cargamos $t1 =  E[R[4] + R[3]]

add $s0, $s1, $t1        # f = g + E[R[4]-R[3]]
```

In your simulator, enter the values for the registers and memory locations shown in the following table:

	0X2D	
	0X20	
	0X1F	
	0X34	
0X10010040	0X21	E
	.	
	.	
	.	
	.	
	.	
	0X3C	
	0X38	
	0X5F	
	0X1A	
0X10010010	0X18	R
MEMORY		

$s0	
$s1	0x41
$s2	
$s3	
$s4	
$s5	
$s6	0x10010010
$s7	0x10010040
REGISTERS	

Note: Remember that each word in MIPS contains 32 bits.

Memoria antes de ejecución

ata Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000018	0x0000001a	0x0000005f	0x00000038
0x10010020	0x0000003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000021	0x00000034	0x0000001f	0x00000020	0x0000002d	0x00000000	0x00000000	0x00000000

Registros antes de ejecución

Registers | Coproc 1 | Coproc 0

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	0x00000041
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x10010010
$s7	23	0x10010040
$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

Memoria despues de ejecución

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000018	0x0000001a	0x0000005f	0x00000038
0x10010020	0x0000003c	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000021	0x00000034	0x0000001f	0x00000020	0x0000002d	0x00000000	0x00000000	0x00000000

Registros despues de ejecución

Registers | Coproc 1 | Coproc 0

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	0x10010050
$t1	9	0x0000002d
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0000006e
$s1	17	0x00000041
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x10010010
$s7	23	0x10010040
$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		0x0040001c
hi		0x00000000
lo		0x00000000

Ahora podemos observar como $s0, correspondiente a F, tiene el valor final guardado, el cual es 0X6E.

Laboratorio2.asm

```
1  lw $t0, 16($s6)        # $t0 = R[4]
2  lw $t1, 12($s6)        # $t1 = R[3]
3  sub $t0, $t0, $t1      # $t0 = $t0 - $t1 = R[4] - R[3]
4  sll $t0, $t0, 2        # 2^2 = 4, entonces multiplicamos por 4, para tener correctamente los bytes. Entonces tenemos que $t0 = $t0*4
5  add $t0, $t0, $s7      # $t0 = dir de E[R[4] + R[3]]
6  lw $t1 0($t0)          # Cargamos $t1 =  E[R[4] + R[3]]
7  add $s0, $s1, $t1      # f = g + E[R[4]-R[3]]
```