

Estructura de Computadores

Tema 3. Traducción de Programas

Ejercicio

- ▶ Una función tiene las siguientes variables locales guardadas en \$s0, \$s1 y \$s2:
 - ▶ `int *a, b, c;`
- ▶ Traduce la siguiente sentencia en C a ensamblador MIPS
 - ▶ `*(a+1) = b & 1 || c >> 1;`

<input type="text"/>	<code>\$t1, \$s1, 1</code>
<input type="text"/>	<code>\$t0, \$zero, \$t1</code>
<input type="text"/>	<code>\$t0, \$zero, fi</code>
<input type="text"/>	<code>\$t2, \$s2, 1</code>
<input type="text"/>	<code>\$t0, \$zero, \$t2</code>

<code>fi:</code>	<code>sw</code>	<input type="text"/>
------------------	-----------------	----------------------

Ejercicio

```

func:          ble    $a0, $a2, etiq1
              ble    $a1, $a2, etiq2
              etiq1: beq    $a2, $zero, etiq3
              etiq2: li    $v0, 0
                      b     etiq4
              etiq3: li    $v0, 1
              etiq4:
                      jr    $ra

```

```

int func(int x, int y, int z)
{

```

```

    int res;

```

```

    if ( ((  ) && (  )) || (  ) )

```

```

    {

```

```

        res = 0;

```

```

    } else

```

```

    {

```

```

        res = 1;

```

```

    }

```

```

    return res;

```

```

}

```

Ejercicio

a está en \$t0 y b en \$t1

```
if ((a>b || b<0) && a>0)
    a=0;
else
    a=1;
```

	<input type="text"/>	\$t0, \$t1, <input type="text"/>
etiq1:	<input type="text"/>	\$t1, \$zero, <input type="text"/>
etiq2:	<input type="text"/>	\$t0, \$zero, <input type="text"/>
etiq3:	move	\$t0, \$zero
etiq4:	b	<input type="text"/>
etiq5:	li	\$t0, 1
etiq6:		

Ejercicio

- Escribe el valor final en hexadecimal del registro \$t0.

```
li    $t0, 0x0020A040
sw    $t0, 0($sp)
lb    $t0, 1($sp)
srl   $t0, $t0, 4
ori   $t0, $t0, 0x0099
```

Ejercicio

- Escribe el valor final en hexadecimal del registro \$t0.

```
addiu $t0, $zero, -1  
sltu  $t0, $zero, $t0  
addiu $t0, $t0, -1
```

Ejercicio

- Traduce la siguiente subrutina en C a ensamblador MIPS

```
int fib(int n) {  
    int tmp;  
    if (n<2)  
        tmp = n;  
    else  
        tmp = fib(n-1) + fib(n-2);  
    return tmp;  
}
```


- Dadas las siguientes declaraciones en C:

```
char a;  
int b;  
long long int c;  
main() {  
    char *p;           /* punter guardat en $t0 */  
    int *q;             /* punter guardat en $t1 */  
    long long int *h;   /* punter guardat en $t2 */  
}
```

- Traduce a MIPS las siguientes sentencias en C:

1. $q = q + 1$;
2. $a = *p$;
3. $h = \&c$;
4. $b = *(q + b)$;
5. $p[*q + 10] = a$;
6. $h = \&h[*p]$;

Ejercicio

- ▶ Dado el código en C que se muestra:
 - ▶ Dibuja el bloque de activación de la subrutina f
 - ▶ Traduce a MIPS la sentencia: $y = g(w, \&y, v[i]) + x;$

```
char g(int *a, char *b, int c);  
char f(int w[], char *p, int i)  
{  
    char x, y, z;  
    int v[10];  
    x = g(v, p, i);  
    y = g(w, &y, v[i]) + x;  
    z = g(v, &y, i);  
    return y + z;  
}
```

Ejercicio

- Traduce la siguiente subrutina en C a ensamblador MIPS

```
short s3(unsigned long long *p1) {  
    short v1;  
    if (*p1 != 0)  
        v1 = 1 + s3(p1+1);  
    else  
        v1 = 0;  
    return v1;  
}
```