Tema 4. El lenguaje de la máquina

Ejercicios

1 Sentencias de asignación

Se tiene la siguiente sentencia en C++.

```
a = b;
```

Sabiendo que **a** se almacena en R0 y **b** en R1, traduce la anterior sentencia al lenguaje del CT.

```
mov r0, r1
```

2 Sentencias de asignación

Se tiene la siguiente sentencia en C++.

```
a = 45;
```

Sabiendo que **a** se almacena en R0, traduce la anterior sentencia al lenguaje del CT.

```
movl r1, 45
movh r1, 0
mov r0, r1
```

3 Sentencias de asignación

Se tiene la siguiente sentencia en C++.

```
a = b;
```

Sabiendo que **a** se almacena en R0 y **b** en la dirección de memoria 2010h, traduce la anterior sentencia al lenguaje del CT.

```
movl r1, 10h
movh r1, 20h
mov r0, [r1]
```

4 Sentencias de asignación

Se tiene la siguiente sentencia en C++.

```
a = b;
```

Sabiendo que **a** se almacena en la dirección de memoria 3456h y **b** en R3, traduce la anterior sentencia al lenguaje del CT.

```
movl r0, 56h
movh r0, 34h
mov [r0], r3
```

5 Sentencias de asignación

Se tiene la siguiente sentencia en C++.

```
a = b;
```

Sabiendo que **a** se almacena en la dirección de memoria 3456h y **b** en la dirección de memoria AB12h, traduce la anterior sentencia al lenguaje del CT.

```
movl r0, 12h
movh r0, ABh
mov r1, [r0]
movl r0, 56h
movh r0, 34h
mov [r0], r1
```

6 Sentencias aritméticas

Se tiene la siguiente sentencia en C++.

```
a = b + 5;
```

Sabiendo que **a** se almacena en R0 y **b** en R1, traduce la anterior sentencia al lenguaje del CT.

```
movl r2, 5
movh r2, 0
add r0, r1, r2
```

7 Sentencias aritméticas

Se tiene la siguiente sentencia en C++.

```
a = b - c;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en la posición de memoria 4567h, traduce la anterior sentencia al lenguaje del CT.

```
movl r2, 67h
movh r2, 45h
mov r3, [r2]
sub r0, r1, r3
```

8 Sentencias aritméticas

Se tiene la siguiente sentencia en C++.

```
a = (b + c) – (a + 1024);
```

Sabiendo que **a** se almacena en la dirección de memoria 1000h, **b** en R1 y **c** en R2, traduce la anterior sentencia al lenguaje del CT.

```
movl r3, 00h
movh r3, 10h
mov r4, [r3]
movl r5, 00h
movh r5, 04h
add r4, r4, r5
add r6, r1, r2
sub r7, r6, r4
mov [r3], r7
```

9 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if (a == b)
   c = c + 1;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT.

```
   cmp r0, r1
   brz consecuente
   jmp siguiente
consecuente:
   movl r3, 1
   movh r3, 0
   add r2, r2, r3
siguiente:
```

10 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if (a != b)
   c = c - a;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT.

```
   cmp r0, r1
   brnz consecuente
   jmp siguiente
consecuente:
   sub r2, r2, r0
siguiente:
```

11 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if (a < b)
   c = 0;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT. Las variables **a**, **b** y **c** almacenan números naturales.

```
   cmp r0, r1
   brc consecuente
   jmp siguiente
consecuente:
   xor r2, r2, r2
siguiente:
```

12 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if (a <= b)
   c = 27;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT. Las variables **a**, **b** y **c** almacenan números naturales.

```
   cmp r0, r1
   brc consecuente
   brz consecuente
   jmp siguiente
consecuente:
   movl r2, 27
   movh r2, 0
siguiente:
```

13 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if ((a == b) && (a != c))
   a = 0;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT. Las variables **a**, **b** y **c** almacenan números naturales.

```
   cmp r0, r1
   brz comprueba_a_dis_c
   jmp siguiente
comprueba_a_dis_c:
   cmp r0, r2
   brnz consecuente
   jmp siguiente
consecuente:
   xor r0, r0, r0
siguiente:
```

14 Sentencias condicionales

Se tienen las siguientes sentencias en C++.

```
if ((a == b) || (a != c))
   a = 0;
```

Sabiendo que **a** se almacena en R0, **b** en R1 y **c** en R2, realiza la traducción al lenguaje del CT. Las variables **a**, **b** y **c** almacenan números naturales.

```
   cmp r0, r1
   brz consecuente
   cmp r0, r2
   brnz consecuente
   jmp siguiente
consecuente:
   xor r0, r0, r0
siguiente:
```

15 Bucles

Se tienen las siguientes sentencias en C++.

```
for (i = 0; i < 100; i++)
   a = a - i;
```

Sabiendo que **a** se almacena en R5, **i** en R0 y es natural, realiza la traducción al lenguaje del CT.

```
   xor r0, r0, r0 ; i = 0
   movl r1, 100
   movh r1, 0

inicio_for:
   cmp r0, r1
   brnc fin_for

   sub r5, r5, r0

   inc r0
   jmp inicio_for
fin_for:
```

16 Bucles

Se tienen las siguientes sentencias en C++.

```
i = 0;
while (i < 100)
{
   a = a - i;
   i++;
}
```

Sabiendo que **a** se almacena en R5, e **i** en R0 y es natural, realiza la traducción al lenguaje del CT.

```
   xor r0, r0, r0 ; i = 0
   movl r1, 100
   movh r1, 0
inicio_while:
   cmp r0, r1
   brnc fin_while

   sub r5, r5, r0

   inc r0
   jmp inicio_while
fin_ while:
```

17 Bucles

Se tienen las siguientes sentencias en C++.

```
i = 0;
do
{
   a = a - i;
   i++;
} while (i < 100);
```

Sabiendo que **a** se almacena en R5, e **i** en R0 y es natural, realiza la traducción al lenguaje del CT.

```
   xor r0, r0, r0 ; i = 0
   movl r1, 100
   movh r1, 0
inicio_do_while:
   sub r5, r5, r0

   inc r0
   cmp r0, r1
   brc inicio_do_while
```

18 Procedimientos

Se tiene la siguiente sentencia en C++.

```
x = Min (a, b);
```

Sabiendo que **a** se almacena en R4, **b** se almacena en R5, **x** en R6, el paso de parámetros se realiza a través de los registros R1 y R2 para el primer y segundo parámetro respectivamente y que el valor de retorno se almacena en R0, realiza la traducción al lenguaje del CT.

```
mov r1, r4
mov r2, r5
call Min
mov r6, r0
```

19 Procedimientos

Se tiene la siguiente sentencia en C++.

```
x = Min (a, b);
```

Sabiendo que **a** se almacena en R4, **b** se almacena en R5, **x** en R6, el paso de parámetros se realiza a través de la pila de derecha a izquierda por valor y que el valor de retorno se almacena en R0, realiza la traducción al lenguaje del CT.

```
push r5
push r4
call Min
inc r7
inc r7
mov r6, r0
```

20 Procedimientos

Se tienen las siguientes sentencias en C++.

```
int Min (unsigned int a,
         unsigned int b)
{
   if (a < b)
      return a;
   else
      return b;
}
```

Sabiendo que **a** y **b** se han pasado como parámetros a través de los registros R1 y R2 respectivamente y que el valor de retorno se almacena en R0, realiza la traducción al lenguaje del CT.

```
Min:
   cmp r1, r2
   brc consecuente
   mov r0, r2
   jmp siguiente
consecuente:
   mov r0, r1
siguiente:
   ret
```

21 Procedimientos

Se tienen las siguientes sentencias en C++.

```
int Min (unsigned int a,
         unsigned int b)
{
   if (a < b)
      return a;
   else
      return b;
}
```

Sabiendo que **a** y **b** se han pasado como parámetros a través de la pila de derecha a izquierda y que el valor de retorno se almacena en R0, realiza la traducción al lenguaje del CT.

```
Min:
   push r6
   mov r6, r7
   push r1
   push r2
   inc r6
   inc r6
   mov r1, [r6]
   inc r6
   mov r2, [r6]

   cmp r1, r2
   brc consecuente
   mov r0, r2
   jmp siguiente
consecuente:
   mov r0, r1
siguiente:

   pop r2
   pop r1
   pop r6
   ret
```

22 Procedimientos

Se tiene la siguiente sentencia en C++.

```
Min (x, a, b);
```

Sabiendo que **a** se almacena en R4, **b** se almacena en R5, **x** en la dirección de memoria 1000h, el paso de parámetros se realiza a través de la pila de derecha a izquierda, **x** se pasa por dirección y los otros parámetros por valor y que el valor de retorno se almacena en R0, realiza la traducción al lenguaje del CT.

```
push r5
push r4
movl r0, 0
movh r0, 10h
push r0
call Min
inc r7
inc r7
inc r7
```

23 Procedimientos

Se tienen las siguientes sentencias en C++.

```
void Min (unsigned int& x,
   unsigned int a, unsigned int b)
{
   if (a < b)
      x = a;
   else
      x = b;
}
```

Sabiendo que **x**, **a** y **b** se han pasado como parámetros a través de la pila de derecha a izquierda, realiza la traducción al lenguaje del CT.

```
Min:
   push r6
   mov r6, r7

   push r0
   push r1
   push r2

   inc r6
   inc r6
   mov r0, [r6]
   inc r6
   mov r1, [r6]
   inc r6
   mov r2, [r6]

   cmp r1, r2
   brc consecuente
   mov [r0], r2
   jmp siguiente
consecuente:
   mov [r0], r1
siguiente:

   pop r2
   pop r1
   pop r0
   pop r6
   ret
```

24 Procedimientos

Se tienen las siguientes sentencias en C++.

```
int SumaAcumulada(int Vector[],
   unsigned int NumElems)
{
   int Suma;
   unsigned int i;

   Suma = 0;
   for (i = 0; i < NumElems; i++)
      Suma = Suma + Vector[i];
   return Suma;
}
```

Sabiendo que **Vector** y **NumElems** se han pasado como parámetros a través de la pila de derecha a izquierda, que el valor de retorno se almacena en R0, que la variable local **Suma** se almacena en la pila y variable local **i** en R3, realiza la traducción al lenguaje del CT.

```
SumaAcumulada:
   push r6
   mov r6, r7 ; Prólogo

   dec r7

   push r1 ; Salvaguarda de registros
   push r2
   push r3
   push r4
   push r5
   inc r6
   inc r6
   mov r1, [r6]
   inc r6
   mov r2, [r6]

   movl r4, 4
   movh r4, 0
   sub r6, r6, r4 ; r6 = &Suma

   ; Cuerpo del procedimiento
   xor r4, r4, r4
   mov [r6], r4 ; Suma = 0
   ; Comienzo del bucle
   xor r3, r3, r3 ; i = 0
inicio_for:
   cmp r3, r2
   brnc fin_for

   ; Cuerpo del for
   add r5, r1, r3 ; r5 = &(Vector[i])
   mov r5, [r5]   ; r5 = Vector[i]

   mov r4, [r6]
   add r4, r4, r5
   mov [r6], r4

   inc r3 ; i++
   jmp inicio_for
fin_for:

   mov r0, [r6]

   pop r5
   pop r4
   pop r3
   pop r2
   pop r1
   inc r7
   pop r6
   ret ; Epílogo
```