Cada respuesta incorrecta, ilegible o vacía no suma ni resta.

1❑ (1 punto) El siguiente fragmento de código C++ debe traducirse al ensamblador del CT. Indica las instrucciones necesarias teniendo en cuenta que el vector `array` se almacena a partir de la dirección de memoria 9528h.

```
array[9] = -8;
```

2❑ (1 punto) El siguiente fragmento de código C++ debe traducirse al ensamblador del CT. Indica las instrucciones necesarias teniendo en cuenta que las variables `e`, `b` y `a` son naturales y se almacenan en los registros `r0`, `r4` y `r1`, respectivamente.

```
if ((b + a) != e)
  e = a;
else
  e = e ^ b; // e = e XOR b
```

3❑ (1 punto) Se ha escrito el siguiente fragmento de código en C++. Sabiendo que la variable `m` se almacena en `r4` y que la variable `s` se almacena en `r5`, traduce el código al lenguaje del CT.

```
unsigned int m = 100;
unsigned int s = 0;
while (m == s)
 {
   s = s + 2;
 };
```

4❑ (1 punto) Se ha escrito el siguiente fragmento de código en C++.

```
z = Result(x, y);
```

Tradúcelo al lenguaje del CT teniendo en cuenta que la variable `z` se almacena en memoria en la posición `5A51h`; la variable `x` se almacena en el registro `r3`; la variable `y` se almacena en el registro `r5`; el paso de parámetros se realiza a través de la pila; el orden de paso de parámetros es de derecha a izquierda y el procedimiento devuelve el resultado en `R0`.

A

5 ❑ **(2 puntos)** Se tiene el siguiente procedimiento en C++:

```
unsigned int ProcB(unsigned int n1, unsigned& int n2)
{
  if (n1 == n2)
    return (n1 + n1);
  else
    return (n1 & n2);
}
```

Sabiendo que el paso de parámetros se realiza a través de la pila de derecha a izquierda, que uno de los parámetros se pasa por valor y otro por referencia y que el procedimiento devuelve su valor en `r0`, traduce el procedimiento anterior al lenguaje del CT.

6 ❑ Se tiene siguiente fragmento en el lenguaje del CT:

```
   xor r6, r3, r0
loop:
   cmp r1, r2
   brc next
   xor r0, r4, r6
   not r6
   inc r2
   jmp loop; --INSTR1--
next:
   dec r2
   pop r5; --INSTR2--
```

El CT acaba de terminar la ejecución de una instrucción (no perteneciente al código anterior) y va a comenzar a ejecutar el código anterior. Se conoce el valor de los registros del CT:

`R0=0006h` `R1=0A39h` `R2=0A39h` `R3=0105h` `R4=FAA0h`
`R5=0CBFh` `R6=0CBFh` `R7=AB4Bh` `PC=F178h` `IR=3600h`

Recuerda que el número que va a continuación de `1239h` es `123Ah`, y el número que va antes de `1210h` es `120Fh`.

a — **(0,5 puntos)** ¿Cuál es el mnemónico de la instrucción que se ha terminado de ejecutar?

b — **(0,5 puntos)** ¿Cuál sería la codificación de la instrucción marcada como **–INSTR1–** si se sustituyese por la instrucción `JMP there` y la etiqueta *there* marcase la posición de memoria `F178h`? (Responder en hexadecimal)

c — **(0,5 puntos)** Cuando se ejecute la instrucción marcada como **–INSTR2–**, ¿a qué dirección de memoria se accede después del paso 4 de su ejecución y antes de su finalización? (Responder en hexadecimal)

d — **(1 punto)** Suponiendo que el CT emplea un reloj de 0.4 kHz ¿cuánto tiempo tarda en ejecutarse el fragmento de programa anterior? (Responder en ms)

7 ❑ **(0,5 puntos)** Se ha añadido al Computador Teórico la siguiente instrucción:

Código instrucción	Mnemónico	Operación
011100 Rd Ri Rs2 0	XOR Rd, [Ri], Rs2	Rd ← [Ri] XOR Rs2

¿Cuál sería el código que le corresponde a la instrucción `XOR R4, [R3], R4`? (Contesta en hexadecimal).

8 ❑ **(1 punto)** Se desea añadir una nueva instrucción aritmética para el CT con el mnemónico `CMP R0, Inm8`. Esta instrucción resta el registro R0 menos la constante `Inm8` (extendiendo su signo para transformarla a 16 bits) y actualiza los bits correspondientes del registro de estado. Indicar la secuencia de señales de control para los pasos 4 y siguientes necesarias para su ejecución en el menor número de pasos posibles.

Paso	Señales