Alumno: Suelo Enrique Técnicas Digitales II 04/07/19

Examen Final

1 1) Indique la diferencia entre ejecución normal y ejecución condicional de las instrucciones en el ARM. Ejemplifique. (1P)

1 2) ¿Cuáles son las banderas en el ARM? ¿Para qué sirven? (1P)

0 3) La unidad de corrimiento en el ARM se conecta al operando 2 y posee las siguientes funciones: desplazamiento lógico a la izquierda y a la derecha, desplazamiento aritmético a la derecha y rotación a la derecha. ¿Se puede afectar al operando dos con desplazamiento aritmético a la izquierda y rotación a la izquierda? Justifique la respuesta. (1P)

4) Indicar el tiempo de conversión en ciclos de reloj Tclk de un A/D de n bits: (1P)

0,5 4.1- tipo "flash".

4.2- tipo "aproximaciones sucesivas".

4.3- tipo "doble rampa".

1 5) Determine el error de cuantificación de un conversor A/D de 8bits cuyo rango de entradas analógicas a convertir es 1V. Si se desea disminuir a la mitad ese error, ¿se debería duplicar el número de bits del conversor para lograrlo? Justifique su respuesta. (1P)

1 6) Muestre dos diferentes formas de limpiar todos los bits del registro R12 con ceros. No puede usar otro registro que no sea R12 ni tampoco un operando inmediato. (1P)

1 7) Grafique una trama completa de la comunicación serie RS-232. Indique claramente cuales son cada uno de los bits que la componen. ¿Cómo se determina el bit de paridad y para qué sirve? (1P)

1 8) Considere el siguiente fragmento de código de alto nivel. Suponga que las variables enteras (con signo) g y h están en los registros R0 y R1, respectivamente. Escriba un fragmento de código en lenguaje Assembly del ARM con ejecución condicional disponible para todas las instrucciones. Use la menor cantidad de instrucciones posible. (1P)

```
if (g >= h)
  g = g + h;
else
  g = g – h;
```

0 9) Implemente una secuencia de instrucciones que active los bits de la dirección de memoria 0x8000. El orden de las posiciones de bit a activar es 1, 3, 5, 7, 6, 4, 2 y 0 respectivamente. Entre la activación de un bit y otro debe llamar a una subrutina denominada "delay" a la que se le pasa como argumento el tiempo de 500 mseg. (2P)

Suelto Enrique

FECHA

1) La diferencia que existe entre ambas es que la ejecución condicional se ejecuta si se cumple cierta condición inicial, indicadas por las banderas

```
cmp r0,#0
add r0,r0,#1
```
} Ejecución normal ✓

```
cmp r0,#0
addeq r0,r0,#1
```
} Ejecución condicional

2) Banderas

Z → Indica si el resultado es igual a cero, entonces la bandera se activa

rdo = 0 ∴ Z = 1
rdo ≠ 1 ∴ Z = 0

C (carry) → Se activa por suma o comparación. +/−
→ Se activa por desplazamiento, toma el valor del bit desplazado

V → Overflow : Se activa si una suma o resta produce overflow

N → Se activa si el resultado es un número negativo

rdo = (−) número ∴ N = 1
rdo = (+) número ∴ N = 0

4) A) Flash. $t_{conv} = n \cdot t_{clk}$

B) Aprox sucesiva $t_{conv} = (n+1)\, t_{clk}$

C) Doble rampa $t_{conv} = 2^{(n+1)} \cdot t_{clk}$

3) No se puede aplicar porque no permite asl (desplazamiento aritmético a la izquierda) y rol (rotación a la izquierda).

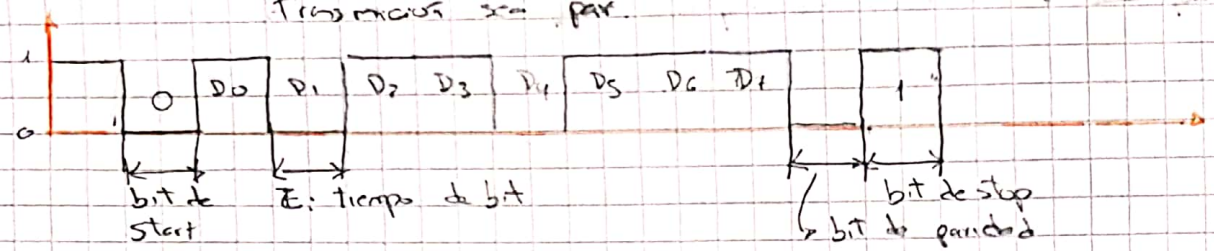
5) Para disminuir a la mitad el error de cuantificación se debe incrementar 1 bit

$$E_{ADC} = \left(\frac{V_{lsb}}{2}\right)\frac{1}{2} = \left(\frac{V_{ref}}{2^n}\right)\frac{1}{2} = \left(\frac{V_{ref}}{2^{n+1}}\right)\frac{1}{2} = \frac{V_{ref}}{2^{n+2}} \rightarrow \boxed{E_{ADC} = \frac{V_{ref}}{2^{n+2}}}$$

✓

NOTA

7) Bit de paridad: indica si el nº de bits con el valor de "1" en el conjunto de bits de datos es par o impar
Sirve para detectar errores y no los corrige
Si la transmisión es par pero el conjunto de bits es impar, el bit de paridad toma el valor de '1' para lograr que la transmisión sea par.



6)
```
ldr   r2, [r0]
ldr   r3, [r1]

cmp   r2, r3
addge r2, r2, r3
sublt r2, r2, r3
b salir

salir: b salir
```

r0 → 9
r1 → h

8) 1) BIC r12, r12, r12

2) EOR r12, r12, r12

BIC

$r_{12} = 0011$
$\overline{r_{12}} = \text{cmc}\ 1100$
0000

EOR

$r_{12} = 0011$
$r_{12} = 0011$
0000

```
mov r12, r12, lsl #32
```

9)
```
      mov r0, #0x8000
      mov r1, #1
      mov r2, #1
      mov r3, #0
impar: cmp r2, #7
      bhi par
      strb r1, [r0, r2]
      add r2, r2, #2
      bl delay
par:  cmp r3, #6
      bhi salir
      strb r1, [r0, r3]
      add r3, r3, #2
      bl delay
delay: b impar
salir: b salir
```

NOTA