Tarea H 1 - IE-623

Device	Flash	RAM	EEPROM
D256	256 K	12 K	4 K

Moisés Campos Zepeda - B13400
II - 2020

1) MCU MC9S12D6 256

Tabla para INITREG

inicio	fin		REG14	REG13	REG12	REG11				Total INITREG
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
$0000	$07FF		0	0	0	0				$00
$0800	$0FFF		0	0	0	1				$08
$1000	$17FF		0	0	1	0				$10
$1800	$1FFF		0	0	1	1				$18
$2000	$27FF		0	1	0	0				$20
$2800	$2FFF		0	1	0	1				$28
$3000	$37FF		0	1	1	0				$30
$3800	$3FFF	X	0	1	1	1	X	X	X	$38
$4000	$47FF		1	0	0	0				$40
$4800	$4FFF		1	0	0	1				$48
$5000	$57FF		1	0	1	0				$50
$5800	$5FFF		1	0	1	1				$58
$6000	$67FF		1	1	0	0				$60
$6800	$6FFF		1	1	0	1				$68
$7000	$77FF		1	1	1	0				$70
$7800	$7FFF		1	1	1	1				$78

Tabla para INITRM

inicio	fin	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		RAM15	RAM14	RAM13	RAM12	RAM11			RAMHAL
$0000	$2FFF	0	0						0
$1000	$3FFF	0	0						1
$4000	$6FFF	0	1						0
$5000	$7FFF	0	1	0 (en 256)	0 (en 256)	0 (en 256)	X	X	1
$8000	$AFFF	1	0						0
$9000	$BFFF	1	0						1
$C000	$EFFF	1	1						0
$D000	$FFFF	1	1						1

Valor INITRM

$0 0
$0 1
$9 0
$9 1
$8 0
$8 1
$C 0
$C 1

Tabla para INITEE

inicio	fin	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		EE15	EE14	EE13	EE12	EE11			EEON
$0000	$0FFF	0	0	0	0				
$1000	$1FFF	0	0	0	1				
$2000	$2FFF	0	0	1	0				
$3000	$3FFF	0	0	1	1				
$4000	$4FFF	0	1	0	0				
$5000	$5FFF	0	1	0	1				
$6000	$6FFF	0	1	1	0				
$7000	$7FFF	0	1	1	1	0	X	X	1
$8000	$8FFF	1	0	0	0	0 (en 256)			
$9000	$9FFF	1	0	0	1				
$A000	$AFFF	1	0	1	0				
$B000	$BFFF	1	0	1	1				
$C000	$CFFF	1	1	0	0				
$D000	$DFFF	1	1	0	1				
$E000	$EFFF	1	1	1	0				
$F000	$FFFF	1	1	1	1				

Valor INITEE

$0 1
$1 1
$2 1
$3 1
$4 1
$5 1
$6 1
$7 1
$D 1
$9 1
$A 1
$B 1
$C 1
$E 1
$F 1

2)

Ciclos			Direccionamiento	
2	Ldd	#$FE3D	; IMM	D = $FE3D , A = $FE , B = $3D
2	Ldx	#$1030	; IMM	X = $1030
1	Ldab	#$10	; IMM	B = $10 , D = $FF10
3	Std	b,X	; IDX	B+X = $1040 M = (M:$1040)
4	Bset	b,X,$55	; IDX	
4	Bclr	a,X,$37	; IDX	

a. Si en $1040 se almacena A y en $1041 se almacena B

* la posición de memoria en b+X: $1040 con valor: $FE
* la posición de memoria en a+X: $112E con valor: $E5

$$\$FE \mid \$55 = \begin{array}{r} \%1010\ 1110 \\ +\%0101\ 0101 \\ \hline 1111\ 1111 \end{array} = \$FF$$

$$\$E5 \cdot (\overline{\$37}) = \begin{array}{r} \%1110\ 0101 \\ \cdot\%1100\ 1000 \\ \hline 1100\ 0000 \end{array} = \$C0$$

Al modificarse los valores en $112E (A+X) con el Bclr y $1040 (B+X) con Bset se obtienen los valores:

$FF del Bset en la posición de memoria $1040
$C0 del Bclr en la posición de memoria $112E

b. Del set de instrucciones, anotando en la columna Ciclos, se tienen los ciclos de maquina que dura cada instrucción. Sumando todos los elementos:

Ciclos de máquina = 16

c. Para determinar cuanto tiempo toma este programa:

$$t_{prog} = \frac{\#\ Ciclos\ de\ máq}{f_{sysclk}} = \frac{16}{48 \times 10^6} = [333.3\ ns]$$

3/8

3) Código para programa en ensamblador 9S12.

a)

```
        Lda   $20,Y     ; Cargo el contenido de Y+$20 en a
        Ldb   #0        ; El registro b cuenta ceros, b=0
        Ldx   #16       ; El registro x almacena el tamaño del
                        ; word
LOOP:
        Bita  $1        ; A . ($1)
        Bne   ROTAR     ; si no hay cero salta a ROTAR
        Incb            ; B+1
ROTAR:
        Dex             ; x-1
        Beq   FIN_LOOP  ; x=0 => FIN_LOOP
        Cpx   #8        ; si(x == 8) salta a CAMBIO_BYTE
        Beq   CAMBIO_BYTE
        Jmp   LOOP      ; regresa a LOOP
CAMBIO_BYTE:
        Ldaa  $21,Y     ; carga segunda mitad de word
        JMp   LOOP
FIN_LOOP:
        stab  $EE10,Y   ; Guarda el # de ceros en Y+EE10
FIN:
        Jmp   FIN       ; Finaliza
```

4/8

b)

```
Ldaa   $2087    ; Carga contenido en posición a A
Eora   $FF      ; Toggle general
Anda   $55      ; Mascara elimina impares.
Staa   $2087    ; Guarda a en $2087
```

4)

Para N < 200 valores de un byte con signo en la dirección de memoria DATOS en RAM

```
        Ldx   #DATOS      ; x = DATOS, apunta a DATOS
        Ldy   #MAYORES    ; y apunta a dir de MAYORES
        clrb              ; b = 0
LOOP:
        Ldaa  1,X+        ; carga dato en a, x+1
        cmpa  #$CE        ; A > -50
        Ble   NO_COPIAR   ; salta el guardado
        Staa  1,Y+        ; Guarda A en dir MAYORES
NO_COPIAR:
        Incb              ; b+1
        Cmpb  #$C8        ; b ≥ 200 ?
        bcs   LOOP        ; sino regrese a LOOP
FIN:
        Jmp   FIN
```

5)

```
        Ldx     # DATOS        ; se cargan las direcciones de inicio
        Ldy     # MAYORES
        Clrb                   ; b = 0
        Stab    $1300          ; Vaciamos contenido en $1300
        Stab    $1301          ; y en $1301
LOOP:
        Incb
        Stab    $1300          ; Almacenamos numero de datos leidos
        Ldaa    b,X            ; Cargamos DATOS + b
        cmpa    #$CE           ; ¿Es mayor a -50?
        Ble     SIG_DATO       ; sino se continua al siguiente dato
        Idab    $1301          ; cargamos numero de datos guardados
        Incb                   ; en MAYORES e incrementamos b+1
        Staa    b,Y            ; Almacenamos dato en MAYORES
        Stab    $1301          ; Almacenamos numero de datos guardados
SIG_DATO:
        Ldab    $1300
        cmpb    #$C8           ; ¿Es b ≥ 200?
        Bcs     LOOP           ; sino continua en LOOP
FIN:
        Jmp     FIN            ; Fin del programa
        end
```

6/8

6)

```
        Ldy   #ORDEN          ; se carga dir. destino de datos
        clra                  ; se asegura q.e A=0 y B=0
        clr b
INICIO:
        ldx   #DATOS_NODOS+1  ; SE aumenta 1 para ver peridad
        ldaa  b,x             ; de ultimo byte
        anda  #$01
        beq   PAR             ; Verifica peridad para mandar a
        jmp   IMPAR           ; subrutina correspondiente
PAR:
        ldx   #DATOS_NODOS    ; se mueve el numero de modelo a
        movw  b,x,2,y+        ; ORDEN
        ldx   #OFFSET_SERIE   ; se mueve el num. de serie a ORDEN
        movw  b,x,2,y+
        ldx   #OFFSET_BR      ; se mueve el baud rate a ORDEN
        movw  b,x,2,y+
        ldx   #OFFSET_THP     ; se mueve el throughput a ORDEN
        movw  b,x,2,y+
        jmp   AVANZAR
IMPAR:
        ldx   #OFFSET_SERIE   ; num. de serie a ORDEN
        movw  b,x,2,y+
        ldx   #OFFSET_BR      ; num. de baud rate a ORDEN
        movw  b,x,2,y+
        ldx   #OFFSET_THP     ; throughput a ORDEN
        movw  b,x,2,y+
        ldx   #DATOS_NODOS    ; num. de modelo a ORDEN
        movw  b,x,2,y+
```

```
AVANZAR:  addb   #2          ; Se verifica si b < N
          cmpb   #N
          bcs    INICIO

FIN:      jmp    FIN         ; Fin del programa.
```

Nota: Se aclaran las siguientes posiciones de inicio:

```
OFFSET_SERIE:  equ  2 * N + DATOS_NODOS
OFFSET_BR:     equ  4 * N + DATOS_NODOS
OFFSET_THP:    equ  6 * N + DATOS_NODOS
```