Ejercicio 1 (18 pts.)

a)

A[15..0]: T1+T2+TW -$t6_{(max)}$ - $tdeco_{(max)}$ - $tor_{(max)}$ - $toe\_buf_{(max)}$ > t15
MREQ\: T1L+T2+TW -$t8_{(max)}$ - $tor_{(max)}$ - $toe\_buf_{(max)}$ > t15
RD\: T1L+T2+TW -$t13_{(max)}$ - $tor_{(max)}$ - $toe\_buf_{(max)}$ > t15
M1\: T1+T2+TW -$t19_{(max)}$ - $tor_{(max)}$ - $toe\_buf_{(max)}$ > t15

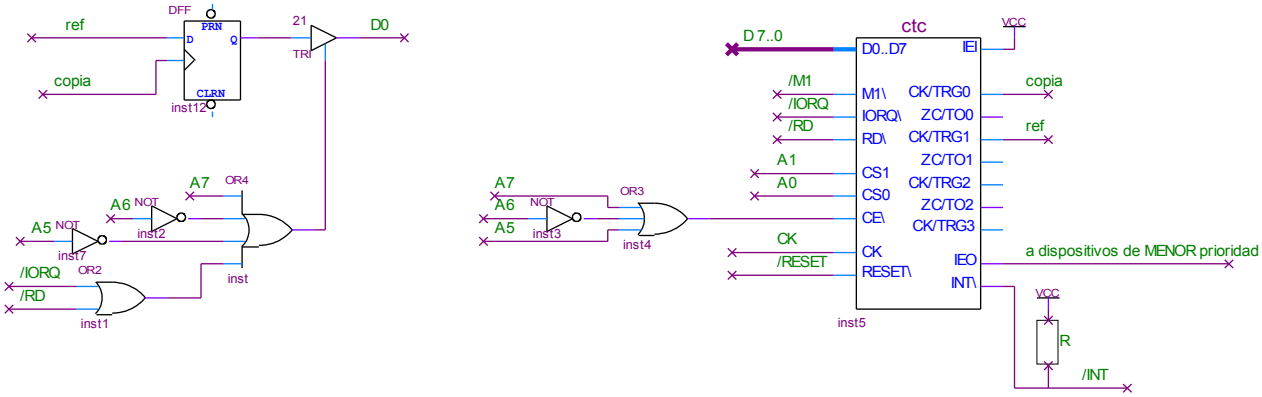
b)

A[15..0]: 50ns + 50ns + 50ns - 57ns - 30ns - 10ns - $toe\_buf_{(max)}$ > 12ns
MREQ\: 25ns + 50ns + 50ns - 40ns - 10ns - $toe\_buf_{(max)}$ > 12ns
RD\: 25ns + 50ns + 50ns - 40ns - 10ns - $toe\_buf_{(max)}$ > 12ns
M1\: 50ns + 50ns + 50ns - 45ns - 10ns - $toe\_buf_{(max)}$ > 12ns

Requerimiento para el buffer: $toe\_buf_{(max)}$<= 41 ns

Ejercicio 2 (18 pts.)

a) Hardware



b) Inicialización

```
; contador, interrumpe cada flanco
        ctrl_ch0 EQU 11X1X111B
        cte_ch0  EQU 1
; timer, pre=256, arranca con trigger
        ctrl_ch1 EQU 00111111B
; cte = 0, retardo = -cuenta
        cte_ch1  EQU 0
        vector   EQU 0x080x80
        dir_ch0  EQU 0x40
        dir_ch1  EQU 0x41
        read_ref EQU 0x60

org 0x0000
        LD SP, 0
        LD AI, tabla/256
        LD I, A
        LD A, ctrl_ch0
        OUT (dir_ch0), A
        LD A, cte_ch0
        OUT (dir_ch0), A
        LD A, ctrl_ch1
        OUT (dir_ch1), A
        LD A, cte_ch0
        OUT (dir_ch1), A
        LD A, vector
        OUT (dir_ch0), A
        ...
```

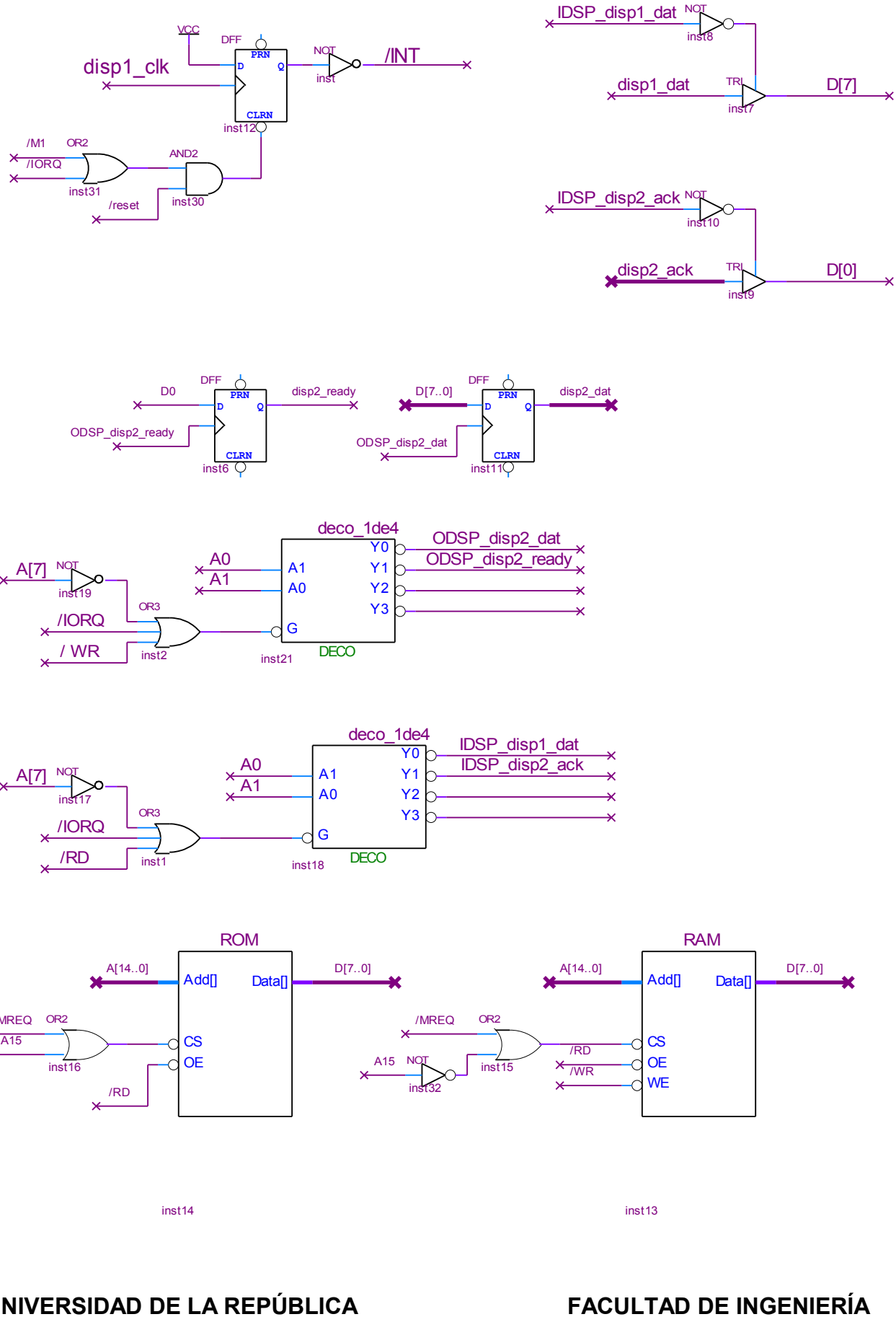
```
org (algun lugar de ROM)
tabla:  DW
        DW
        DW
        DW
        DW rut_int_ch0
        DW
        DW
        DW
        DW
```

Parte c)

```
org (algun lugar de ROM)
        EI
        PUSH AF
        IN A, (dir_ch0)
        NEG
        LD (RETARDO), A
        LD A, ctrl_ch1
        OUT (dir_ch1), A
        LD A, cte_ch1
        OUT (dir_ch1), A
        IN A, (read_ref)
        LD (VALOR_REF), A
        POP AF
        RETI
```

Ejercicio 3 (64 pts.)

Parte a) Hardware



Parte b)

```
org 0x0000

        LD SP, 0x0000
        IM 1
        LD A, 0x00
        LD (hay_palabra),A
        LD (nro_bit), A
        OUT (disp2_ready), A    ;desactivo disp2_ready
        EI
        JP loop_ppal
        ...

org 0x8000
hay_palabra:    DB
palabra:        DB
palabra_aux:    DB              ;variable para ir armado la palabra
ult_lectura:    DB              ;variable para guardar el último bit leido
nro_bit:        DB              ;variable que lleva el nro de bit leido
```

Parte c)

```
disp1_dat       EQU 0x80        ;puerto de lectura D[7]
disp2_dat       EQU 0x80        ;puerto de escritura D[7..0]
disp2_ready     EQU 0x81        ;puerto de escritura D[0]
disp2_ack       EQU 0x81        ;puerto de lectura D[0]

org 0x0038
rut_int:
        PUSH AF
        PUSH BC
        LD A, (nro_bit)
        CP 0                    ;veo si voy a leer el primer bit (bit_7)
        JP NZ, bit_6a0
bit_7:                          ;estoy en el MSB que se transmite primero
        IN A, (disp1_dat)       ;leo dato
        AND 0x80                ;limpio
        LD (ult_lectura), A     ;guardo para comparar luego
        LD A, 0x00
        LD (palabra_aux), A     ;el MSB es siempre 0, lo dejo seteado
        LD A, (nro_bit)
        INC A                   ;incremento nro de bit
        LD (nro_bit), A
        JP fin_int
bit_6a0:
        LD A, (ult_lectura)
        LD C, A                 ;C <- ultima lectura
        IN A, (disp1_dat)       ;leo dato
        AND 0x80                ;limpio
        LD (ult_lectura), A     ;actualizo ult_lectura
        XOR C                   ;comparo con ultima lectura y proceso
        CPL                     ;xor da 1 para bits distintos y 0 para iguales
        AND 0x80                ;complemento bit 7 y reseteo el resto
        LD C, A                 ;C <- bit a añadir en palabra
        LD A, (palabra_aux)
        RLC A                   ;roto palabra a la izquierda
        OR C                    ;agrego bit recien procesado
        LD (palabra_aux), A     ;actualizo palabra
        LD A, (nro_bit)
        INC A                   ;incremento nro_bit
        AND 0x07                ;si nro_bit = 8 -> nro_bit = 0
        LD (nro_bit), A         ;actualizo nro_bit
        JP NZ, fin_int          ;LD (nn), A no afecta bandera Z ->
```

```
        LD A, (palabra_aux)     ;-> verifico resultado de instruccion AND
        RLC A                   ;octava rotación
        LD (palabra), A         ;guardo palabra completa
        LD A, 0xFF
        LD (hay_palabra), A     ;seteo hay_palabra
fin_int:
        POP BC
        POP AF
        EI
        RET
```

Parte d)

```
org algun_lugar_de_ROM
put_palabra:
        PUSH AF
        OUT (disp2_dat), A      ;pongo palabra en bus de datos de disp2
        LD A, 0xFF
        OUT (disp2_ready), A    ;activo disp2_ready
espero_ack:
        IN A, (disp2_ack)
        BIT 0, A
        JP Z, espero_ack        ;espero a que se active disp2_ack
        LD A, 0x00
        OUT (disp2_ready), A    ;desactivo disp2_ready
        POP AF
        RET
```