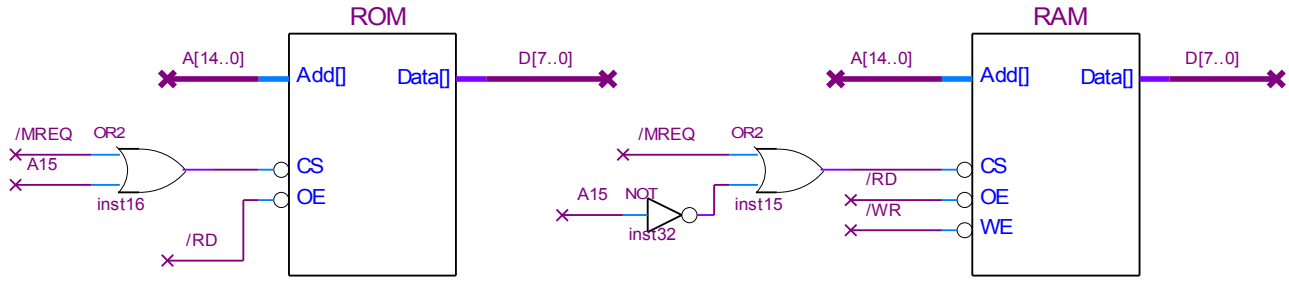
SOLUCIÓN PROBLEMA 1

a) Hardware

-- Memorias:



-- Puertos, decodificación e Interrupciones.

Puertos salida:

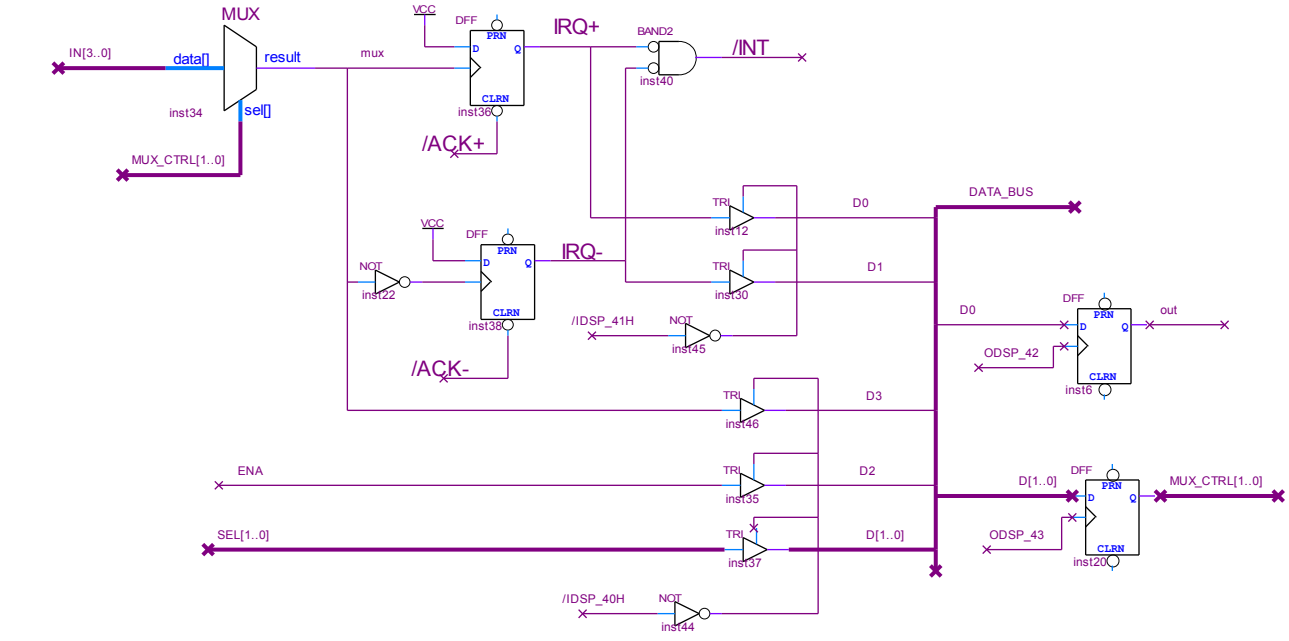
- mux_ctrl[2]
- out

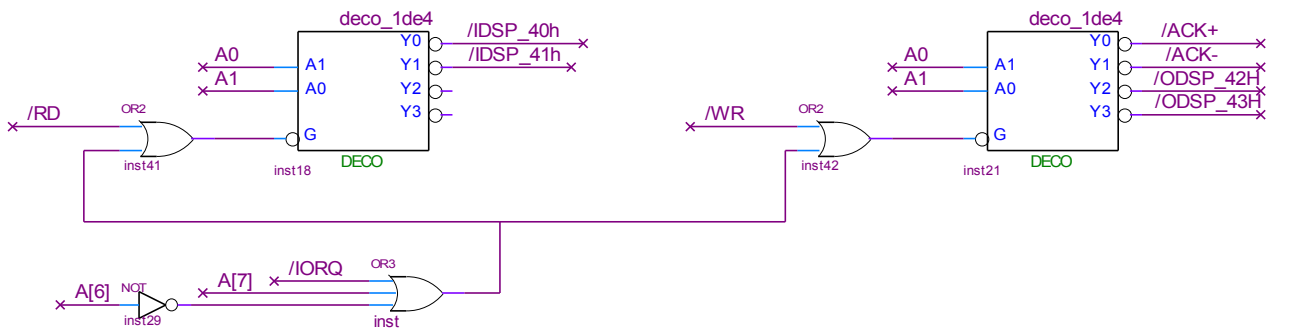
Puertos Entrada:

- port_in[4] = mux, ena, sel[2]
- ffint[2]

Pulsos (mapeados en espacio de salida):

- ack+
- ack-





b) Rutina atención interrupciones

```
; isr:
;      preservar registros
;      si irq+ pendiente entonces
;            isr_subida()
;            borro ff subida
;      else
;            isr_bajada()
;            borro ff bajada
;      finsi
;      restaurar registros
;      ei, ret
;

       org 38h
isr:
       push af
       in a, (irqs)
       bit 0, a
       jr z, else
       call isr_subida
       out (ack_subida), a
       jr finsi
else:
       call isr_bajada
       out (ack_bajada), a
finsi:
       pop af
       ei
       ret

; isr_subida:
;      si (ena=1 and not cambiando) entonces
out <- 1
;
isr_subida:
       in a, (port_in)
       bit bit_ena, a
       jr z, fin1
       ld a, (CAMBIANDO)
       or a
       jr nz fin1
       ld a, 1
       out (port_out), a
       ld (var_out), a
fin1:
       ret

; isr_bajada:
;      out <- 0
isr_bajada:
       ld a, 0
       out (port_out), a
       ld (var_out), a
       ret
```

c) Subrutina manejo_mux

```
; manejo_mux(){
;    si sel != sel_ant or ena != ena_ant {
;       sel_ant = sel
;       ena_ant = ena
;       mientras (out == 1) {}  // espero
;       cambiando = TRUE
;       pmux_ctrl = sel
;       mientras (mux = 1) {}  // espero
;       cambiando = FALSE
;    }
; }
manejo_mux:
       push hl
       push af
       push bc
       ld hl, sel_ena_anterior
       in a, (port_in)
       and mask_sel_ena
       cp (hl)
       jr z, fin_si_sel
             ;            sel_anterior = sel
       ld (hl), a
mientras_out:
       ld a, (var_out)
       bit bit_mux, a
       jr nz, mientras_out
       ld b, a
       ld a, 0xFF
       ld (CAMBIANDO), a   ; cambiando = TRUE
       ld a, b
       out (mux_ctrl), a   ; pmux_ctrl = sel
mientras_mux:
       in a, (port_in)
       bit bit_mux, a
       jr nz, mientras_mux
       xor a
       ld (CAMBIANDO), a
fin_si_sel:
       pop bc
       pop hl
       pop af
       ret
```

d) Inicialización, definición de constantes, reserva de memoria

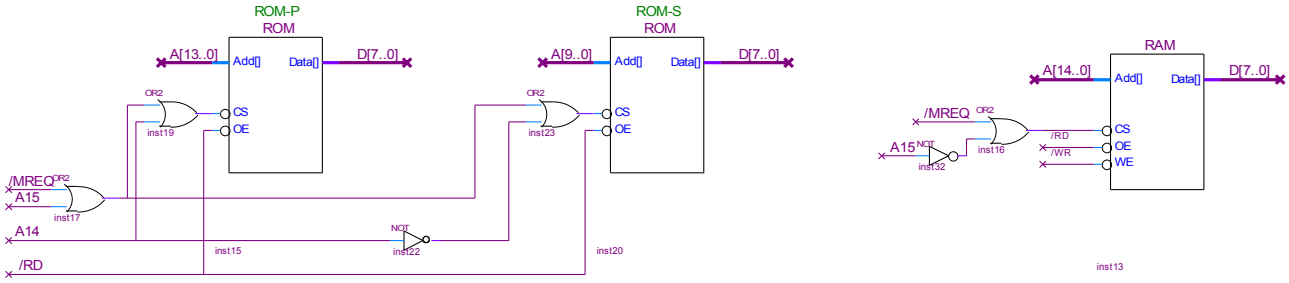
```
; sp, modo 1
; port_out, var_out
; sel_ant=sel, ena_ant=ena
; mux_ctrl =sel
; esperar mux=0
; inicializo_otros()
; habilitar interrupciones
; jp loop principal

       org 0
init:
       ld sp, 0
       im 1
       ld a, 0
       out (port_out), a
       ld (var_out), a
       in a, (port_in)
       and mask_sel_ena
       ld (sel_ena_anterior), a
       out (mux_ctrl), a
espero_inicial:
       in a, (port_in)
       bit bit_mux, a
       jr nz, espero_inicial
       call inicializo_otros
       ei

loop:
    call atiendo_otros
    call manejo_mux
    jp loop
```

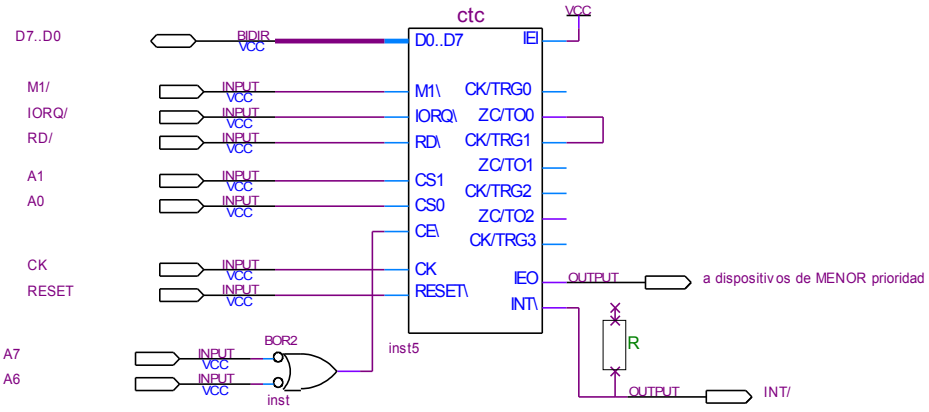
```
; constantes
irqs          equ    0x41
port_in       equ    0x40
bit_mux       equ    3
bit_ena       equ    2
mask_sel_ena  equ    00000111B

ack_subida    equ    0x40
ack_bajada    equ    0x41
port_out      equ 0x42
mux_ctrl      equ    0x43

; Reserva de memoria
      org 8000h
CAMBIANDO:          DB
sel_ena_anterior:   DB
var_out:            DB
```

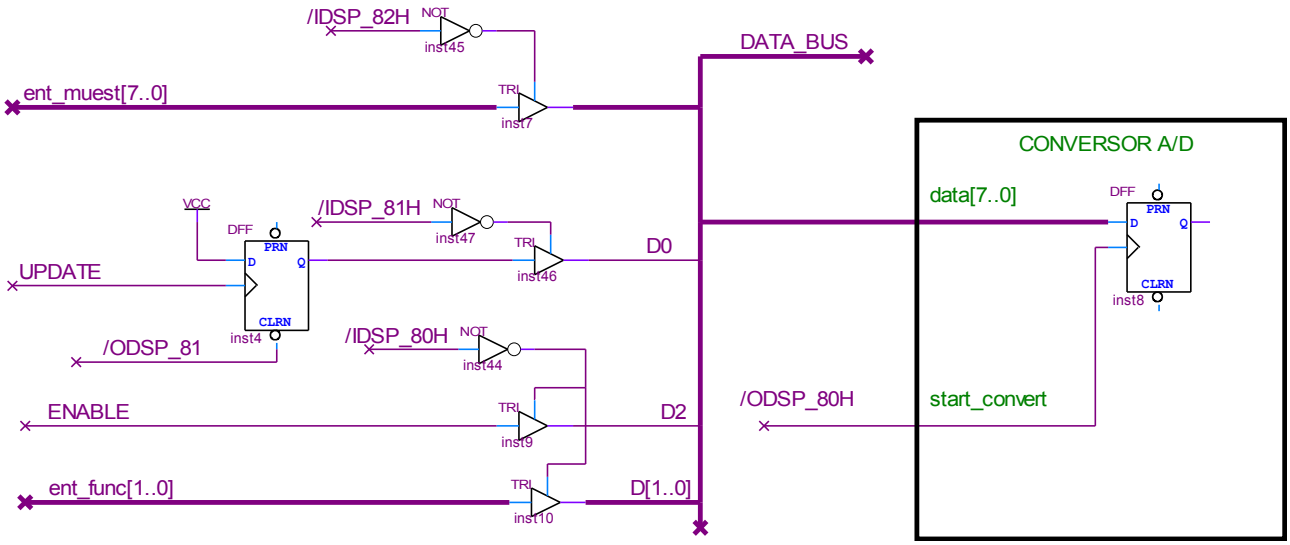
SOLUCIÓN PROBLEMA 2

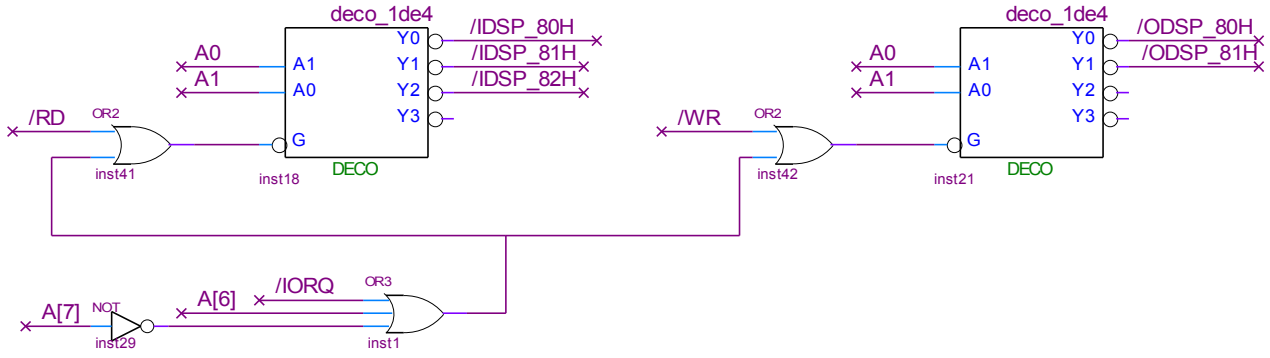
a) Hardware.

Memorias









Aclaración hecha en el salón: cuando ena=0 lo que debe valer 0 es la salida del conversor A/D, la señal data[] puede variar siempre y cuando no haya un flanco en start_convert. En particular en esta solución data[] es directamente el bus de datos por lo que cambia todo el tiempo.

b) Inicialización y programa principal

```
org 0
init:
       ld sp, 0
       im 2
       ld a, int_table/256
       ld i, a
       call inicializo_timer
       call inicializo_otros
       ld a, 0
       out(START_CONVERT), a
       out(UPDATE_CLR), a
       ld (ENABLED), a
       ei

main_loop:
       call atiendo_otros
       in a, (UPDATE)
       or 00h
       jr z, main_loop

       ;; hay update pendiente
       in a, (ENT_FUNC)
       ld b, a
       and ENABLE_MASK
       ld (ENABLED), a
       in a, (ENT_MUEST)
       ld (MUEST_MSEC), a
       ld a, b
       and FUNC_MASK

       ;; preparar puntero a tabla con func
       ld hl, tab_cuad
       add h
       ld h,a
       ld (TAB_FUNC_PTR), hl

       ;; borrar flag de update y
       ;; reconfigurar contador
       out(UPDATE_CLR), a
       call inicializo_counter
       jp main_loop

inicializo_timer:
       ld a, TIMER_VEC
       out(CTC0), a
       ld a, TIMER_CTL
       out(CTC0), a
       ld a, TIMER_CONST
       out(CTC0), a
       ret

inicializo_counter:
       ld a, COUNTER_CTL
       out(CTC1), a
       ld a, (MUEST_MSEC)
       out(CTC1), a
       ret
```

c) Rutina interrupciones

```
isr_counter:
       ei
       push af
       push hl
       ld a, (ENABLED)
       or 00h
       jr z, dac_write

       ld hl, (TAB_FUNC_PTR)
       ld a, (hl)
       inc l
       ld (TAB_FUNC_PTR), hl

dac_write:
       out(START_CONVERT), a
       pop hl
       pop af
       reti


;; tabla interrupciones, reserva de memoria
y constantes

;; tabla ints
org 1000h
int_table:
       dw isr_reservado0
       dw isr_reservado1
       dw isr_reservado2
       dw isr_reservado3
       dw
       dw isr_counter

;; reserva memoria
org 8000h
ENABLED: DB
MUEST_MSEC: DB
TAB_FUNC_PTR: DW

;; constantes
; in ports
ENT_FUNC  equ 80h
UPDATE    equ 81h
ENT_MUEST equ 82h

; out ports
START_CONVERT equ 80h
UPDATE_CLEAR  equ 81h

; CTC
CTC0 equ C0h
CTC1 equ C1h
TIMER_CTL   equ 00110111b ;di,timer,256,tcf
TIMER_VEC   equ 08h
TIMER_CONST equ 250d

COUNTER_CTL equ 11010111b ;ei,cntr,trg,tcf


;; otros
ENABLE_MASK equ 00000100b
FUNC_MASK   equ 00000011b
```