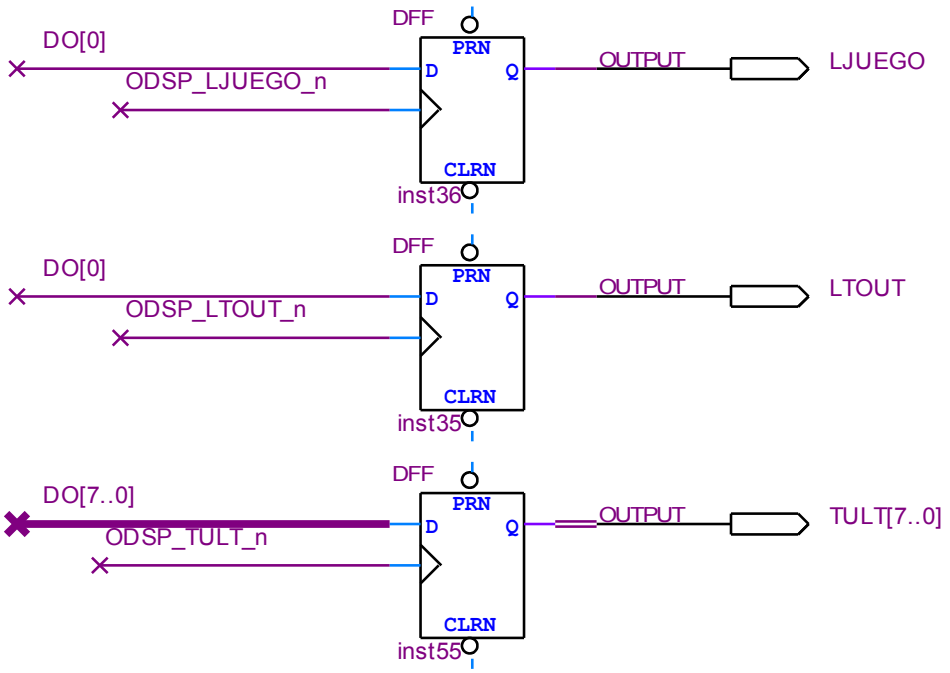
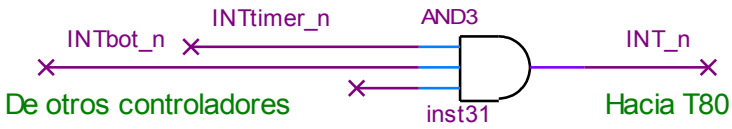
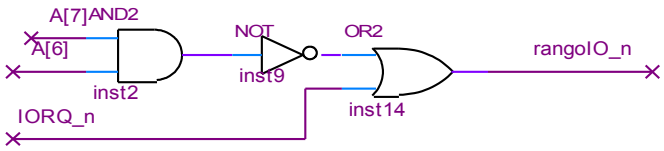
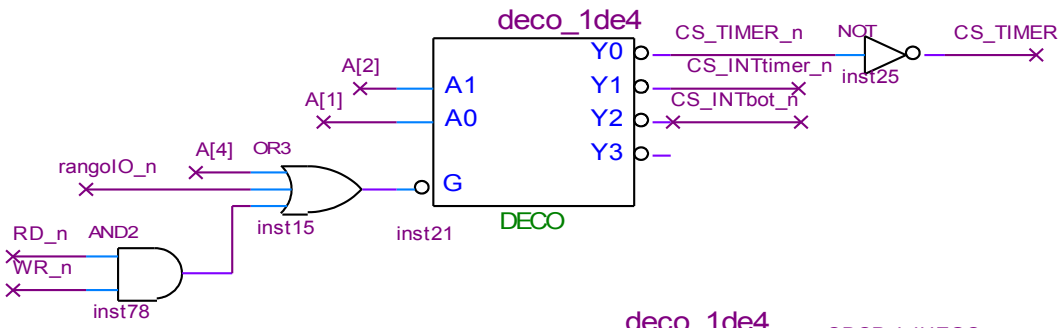
PROBLEMA 1 – Parte a) Caso T80 y bloques lab

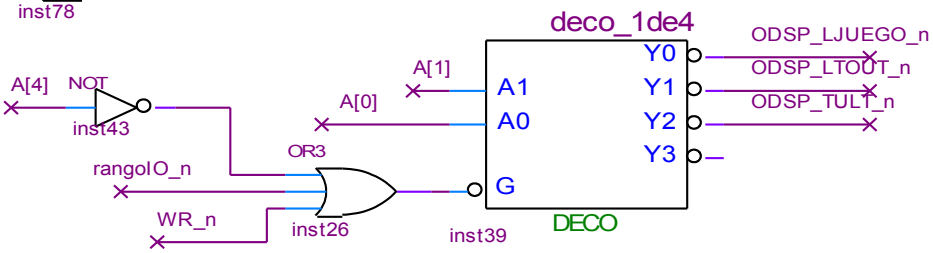
- Bloque Timer
- Controlador1 con bloque Timer
- Controlador0 con BOTON
- TULT[8], LJUEGO, LTOUT puertos out

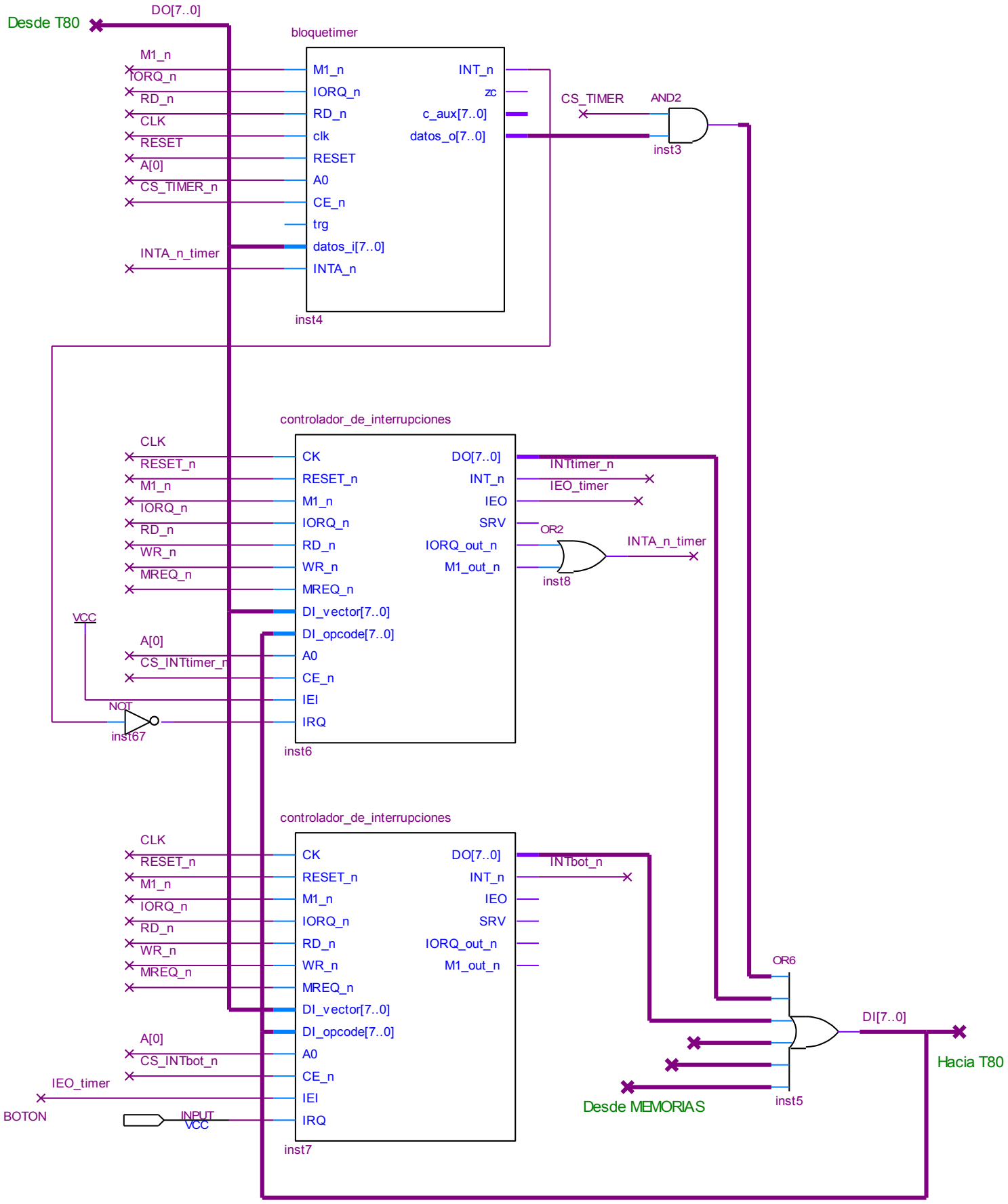




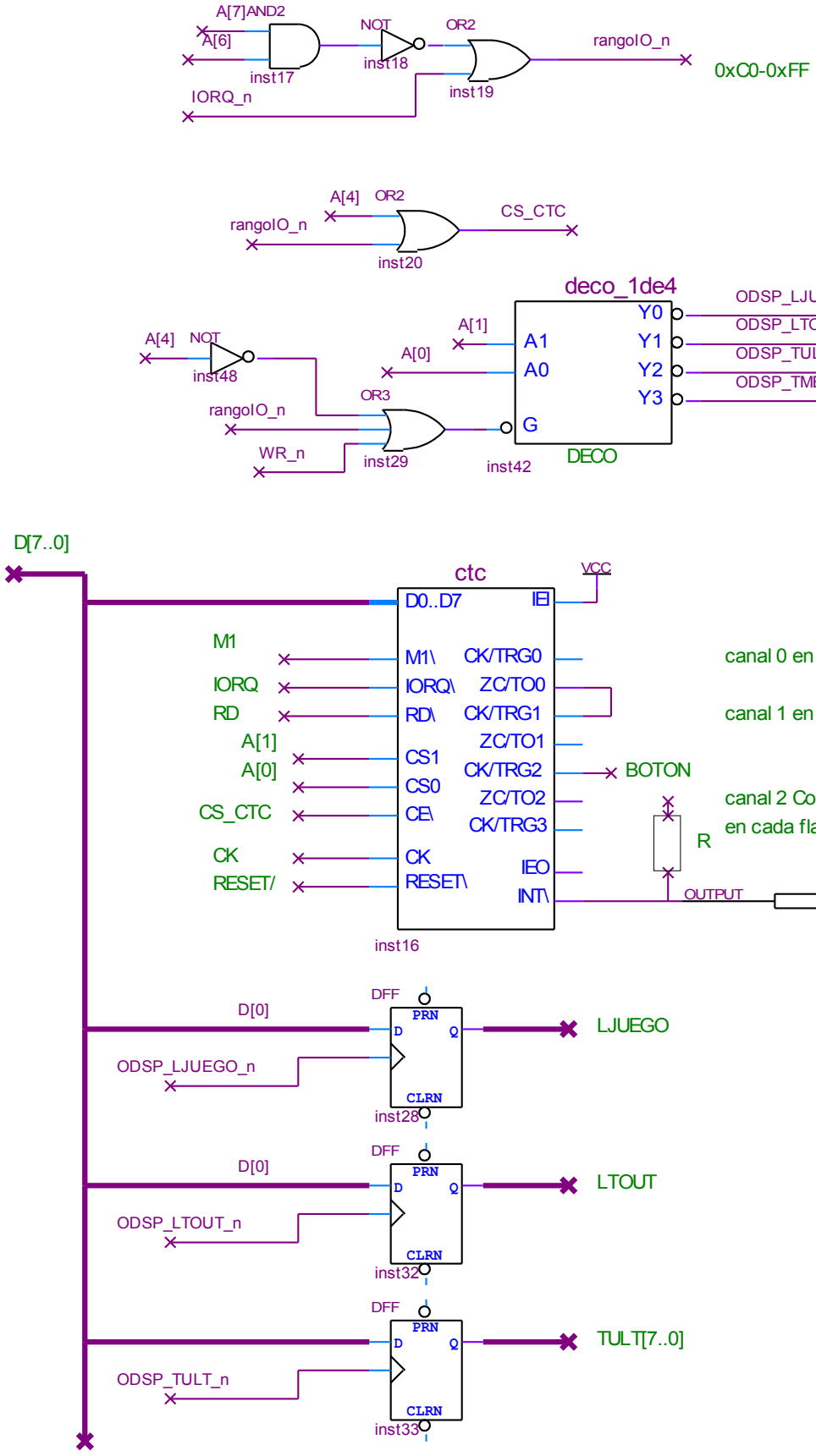







Desde T80
DO[7..0]
bloquetimer
M1_n
IORQ_n
RD_n
CLK
RESET
A[0]
CS_TIMER_n
INTA_n_timer
M1_n
IORQ_n
RD_n
clk
RESET
A0
CE_n
trg
datos_i[7..0]
INTA_n
INT_n
zc
c_aux[7..0]
datos_o[7..0]
inst4
CS_TIMER
AND2
inst3
controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
DI_vector[7..0]
DI_opcode[7..0]
A0
CE_n
IEI
IRQ
DO[7..0]
INT_n
IEO
SRV
IORQ_out_n
M1_out_n
INTtimer_n
IEO_timer
OR2
INTA_n_timer
inst8
inst6
VCC
A[0]
CS_INTtimer_n
NOT
inst67
controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
DI_vector[7..0]
DI_opcode[7..0]
A0
CE_n
IEI
IRQ
DO[7..0]
INT_n
IEO
SRV
IORQ_out_n
M1_out_n
INTbot_n
A[0]
CS_INTbot_n
IEO_timer
BOTON
INPUT
VCC
inst7
OR6
DI[7..0]
Hacia T80
Desde MEMORIAS
inst5

Parte a) Caso buses multiplexados y CTC



```
b) ---- isr de boton y timer ----
isr_boton:
;si ESTADO=Jugando entonces
;    detener Timer
;    ESTADO = Inactivo
;    LJUEGO = 0
;    tjugada = TOUT - timer  ;; timer
decreciente desde TOUT hasta 0
;    TULT = tjugada
;else
;    Estado = Jugando
;    LJUEGO = 1
;    LTOUT = 0
;    TULT = 0
;    Arrancar Timer
;fin_si estado

    ei
    push af
    ld a, (ESTADO)
    cp EJUGANDO
    jr nz, else
        ;; si ESTADO == Jugando
        ;;        detener Timer
    call stop_timer

        ;;     ESTADO = Inactivo
        ;;     LJUEGO = 0
    ld a, EINACTIVO
    ld (ESTADO), a
    ld a, 0
    out (LJUEGO), a

        ;; TULT = TOUT - timer
        ;; decreciente desde TOUT
    in a, (CUENTA_TIMER)
    neg
    add TOUT
    out (TULT), a
    jr finsi
else:
        ;;     Estado = Jugando
    ld a, EJUGANDO
    ld (ESTADO), a
        ;; LJUEGO = 1
        ;; LTOUT = 0
        ;; TULT = 0
    ld a, 0xFF
    out (LJUEGO), a
    ld a, 0
    out (LTOUT), a
    out (TULT), a
        ;; Arrancar Timer
    call start_timer
finsi:
    pop af
    reti

isr_timer:
;    detener Timer
;    LJUEGO = 0
;    LTOUT = 1
;    ESTADO = Inactivo
;
    ei
    push af
    call stop_timer
    ld a, 0
    out (LJUEGO), a
    ld a, 0xff
    out (LTOUT), a
    ld a, EINACTIVO
    ld (ESTADO), a
    pop af
    reti

start_timer:
    ld a, TIMER_ON_CTE
    out (BASE_TIMER+0), a
    ret

stop_timer:
    ld a, TIMER_OFF_CTE
    out (BASE_TIMER+0), a
    ret

c) --- Init ----------------

org 0
        ;; stack, modo2, tabla int
    ld sp, 0
    im 2
    ld hl, BASETABLA
    ld a, h
    ld i, a
        ;; ESTADO = Inactivo
        ;; TULT = 0
        ;; LJUEGO = 0
        ;; LTOUT = 0
    ld a, EINACTIVO
    ld (ESTADO), a
    ld a, 0
    out (TULT), a
    out (LJUEGO), a
    out (LTOUT), a
        ;; inic timer y ctrl int.
    call init_perif
        ;; inic otros dispositivos
    call init_otros
    ei
    jp ppal

init_perif:
        ;; vectores interr
    ld a, TIMER_VEC
    out (BASE_CTRL_TIMER+0), a
    ld a, TOUT_VEC
    out (BASE_CTRL_BOTON+0), a
        ;; borro eventuales peticiones
pendientes
    out (BASE_CTRL_TIMER+1), a
    out (BASE_CTRL_TOUT+1), a
        ;; cte recarga timer
    ld a, TOUT
    out (BASE_TIMER+1), a
    ret
```

```
org 100
        ;; tabla interrupciones
basetabla:
    DW isr_otro0
    DW isr_otro1
    DW isr_otro2
    DW vacio
    DW isr_timer  ;; vect = 8
    DW isr_boton   ;; vect = 10

org 0x8000
        ;; variables
ESTADO: DB


        ;; definicion de ctes
EJUGANDO          EQU  0xFF
EINACTIVO         EQU  0x0

LJUEGO            EQU  0xF0
LTOUT             EQU  0xF1
TULT              EQU  0xF2

DELTA             EQU 100
TOUT              EQU 200

BASE_TIMER        EQU  0xC0
BASE_CTRL_TIMER   EQU  0xC2
BASE_CTRL_BOTON   EQU  0xC4

TIMER_VEC         EQU   8
BOTON_VEC         EQU  10

            ; ei, reset, auto, pre=15
TIMER_ON_CTE      EQU 10101111b

               ; di, resto idem
TIMER_OFF_CTE     EQU 00101111b
```

-- Cambios en solución con CTC

Se utilizan tres canales del CTC:

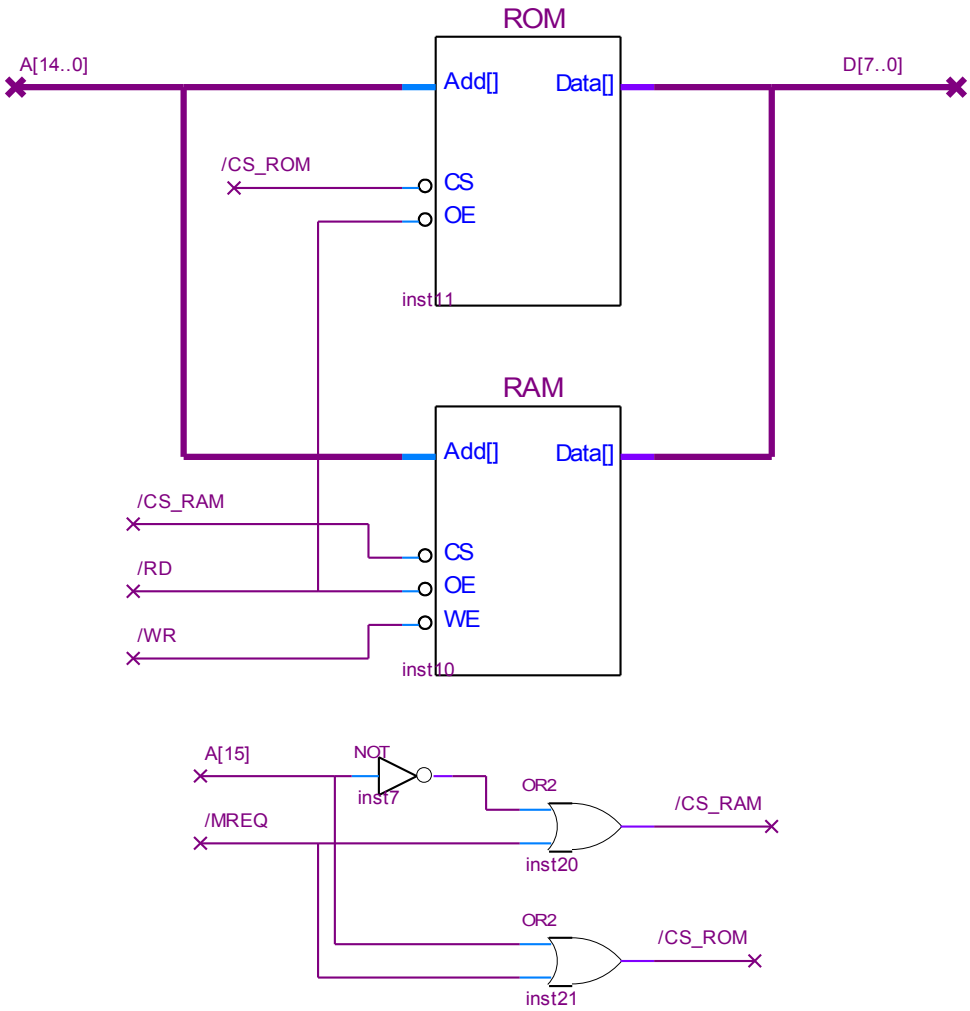
- canal 0: timer, pre=256, cte=128 para llegar a período de 100ms
- canal 1: counter, cte=TOUT para interrumpir por timeout
- canal 2: counter, cte=1 para generar interrupcion con cada flanco de BOTON

```
BASE_CTC          EQU  0xC0
CTC_VEC           EQU  8
CTC_CTE0          EQU  128
   ; di, timer, 256, x, auto, tc, reset
CTC_CW0           EQU  00100111

CTC_CTE1          EQU  TOUT
   ; ei, cnt, x, x, x, tc, reset
CTC_CW1           EQU  11000111
  ; idem con di
CTC_CW1_OFF       EQU  01000111

CTC_CTE2          EQU  1
  ; ei, cnt, x, edge-, x, tc, reset
CTC_CW2           EQU  11000111

Las nuevas versiones de las subrutinas
son:

init_perif:
        ;; vector
    ld a, CTC_VEC
    out (BASE_CTC+0), a
        ;; canal 2 para boton
    ld a, CTC_CW2
    out (BASE_CTC+2), a
    ret

start_timer:
        ;; rearranco canales 0 y 1
    ld a, CTC_CW0
    out (BASE_CTC+0), a
    ld a, CTC_CTE0
    out (BASE_CTC+0), a
    ld a, CTC_CW1
    out (BASE_CTC+1), a
    ld a, CTC_CTE1
    out (BASE_CTC+1), a
    ret

stop_timer:
        ;; deshabilito generación de
interrupciones
    ld a, CTC_CW1_OFF
    out (BASE_CTC+1), a
    ld a, CTC_CTE1
    out (BASE_CTC+1), a
    ret

;;; con los canales elegidos debo
modificar la tabla de interrupciones.
basetabla:
    DW isr_otro0
    DW isr_otro1
    DW isr_otro2
    DW vacio
    DW vacio    ;; vect = 8
                ;; canal 0 no interrumpe
    DW isr_timer
    DW isr_boton
```

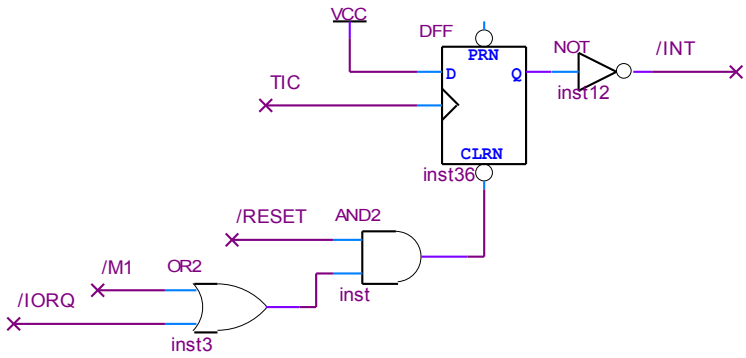
PROBLEMA 2 – Solución

a) Hardware

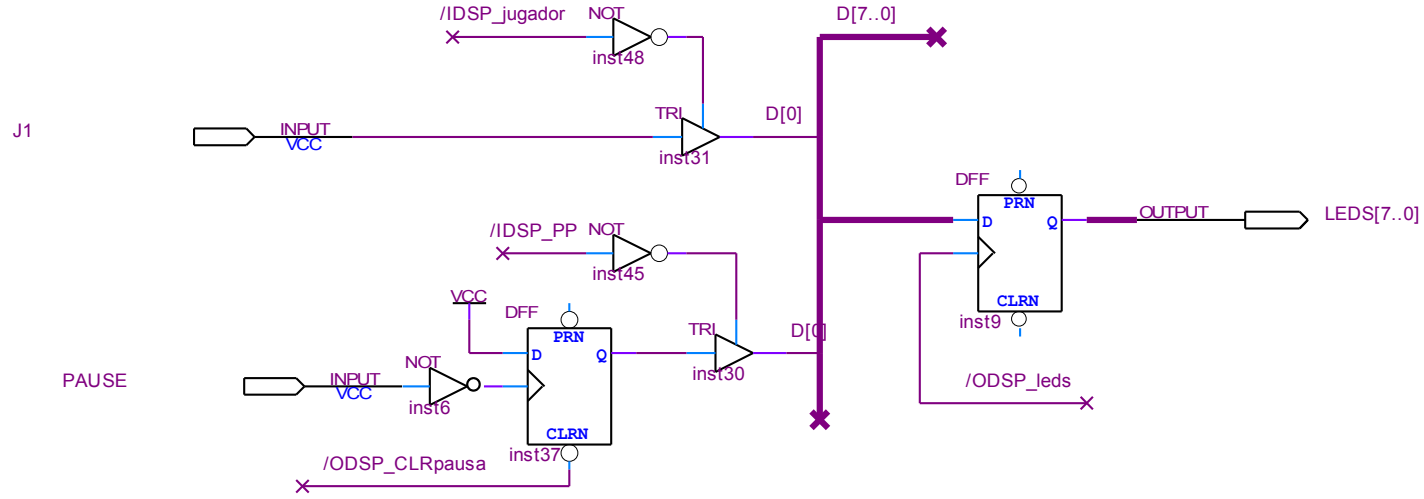
Memoria



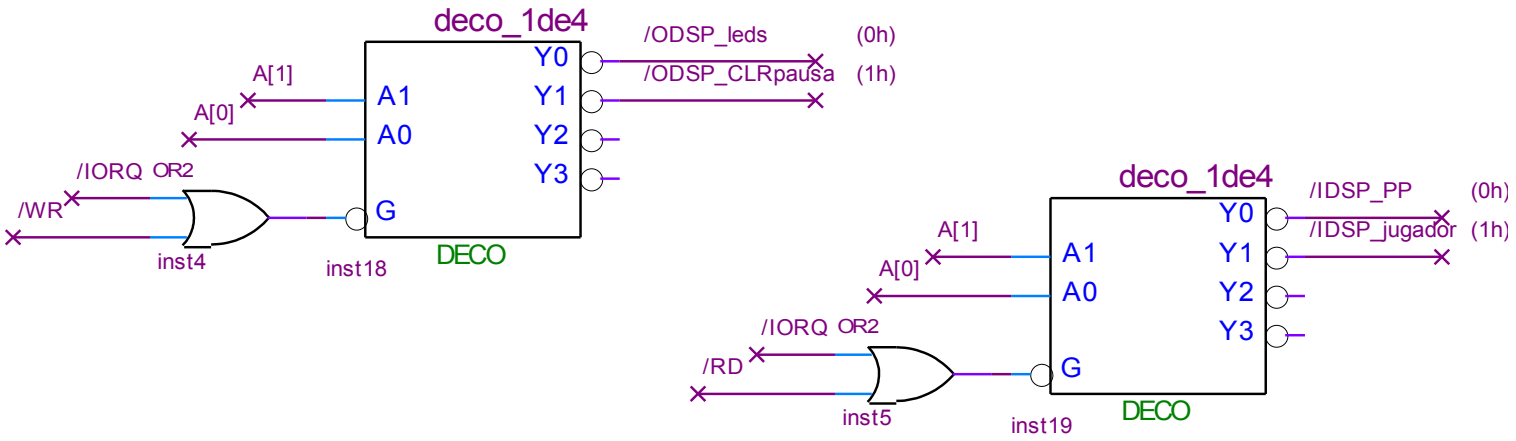
Interrupción



Puertos



Decodificación puertos



d) Inicialización y variables

```
;----- variables
org 8000h
   PAUSA    db  ; FFh = pausa
   FIN      db  ; FFh = fin
   SENTIDO  db  ; 00h der, FFh izq
   POSICION db  ; donde esta la pelota

;----- constantes
   PAUSADO       equ FFh
   FINALIZADO    equ FFh
   DERECHA       equ 00
   SENTIDO_INI   equ DERECHA
   POSICION_INI  equ 0001 0000b
   P_EN_FRONTON  equ 0000 0001b
   P_EN_JUGADOR  equ 1000 0000b

;----- puertos
   LEDS          equ 00h
   PAUSE_PLAY    equ 01h ; en bit 0
   CL_PAUSE_PLAY equ 00h
   JUGADOR       equ 01h ; en bit 0

;----- inicialización
; modo 1 y stack
; variables: fin, pausa, sentido
; HW: borrar flag PAUSE, leds

org 0000h
   im 1
   ld SP, 0000h

   ld A, FINALIZADO
   cpl A
   ld (FIN), A
   ld A, PAUSADO
   ld (PAUSA),A
   ld A, SENTIDO_INI
   ld (SENTIDO),A
   ld A, POSICION_INI
   ld (POSICION),A
   out (LEDS), A
   out (CL_PAUSE_PLAY), A

   EI
   jp PPAL
```

c) Programa principal

```
org 1000h
PPAL:
   ld A, FINALIZADO
   cp (FIN)
   jp Z, END

   in A, (PAUSE_PLAY)
   bit 0, A
   jp Z, TOGGLE_PAUSE_PLAY
   jp PPAL

TOGGLE_PAUSE_PLAY:
   ld A, (PAUSA)
   cpl A,
   ld (PAUSA), A
   out (CL_PAUSE_PLAY), A
   jp PPAL

END: di
END_LOOP:
  jp END_LOOP
```

b) Rutina atención interrupción

```
; si pausa retorno
;
; si pelota yendo a la derecha
;   si pelota en posicion 0
;     cambio sentido
;   actualizo posicion y retorno
; sino
;   si pelota en posicion 7
;     si J1 apretado
;       cambio sentido
;       actualizo posicion y retorno
;     sino
;       fin de juego
;   sino  "hacia izq y no en led 7"
;     si J1 apretado
;       fin de juego
;     sino
;       actualizo posicion y retorno
;
; fin de juego:
;     indico fin en leds
;     seteo variable FIN
;     retorno
;
; actualizo posicion y retorno:
;   si SENTIDO = derecha
;     roto POSICION a la derecha
;   sino
;     roto POSICION a la izquierda
;   actualizo leds
;   retorno

org 38h
   push AF
   push IX

   ld A, (PAUSA)
   cp PAUSADO
   jp Z, retorno

   ld A, (SENTIDO)
   cp DERECHA
   jp NZ, hacia_izquierda
hacia_derecha:
   ld A, (POSICION)
   cp P_EN_FRONTON
   jp Z, cambio_sentido
   jp actualizo
hacia_izquierda:
   ld A, (POSICION)
   cp P_EN_JUGADOR
   jp nz, pelota_lejos_jugador

pelota_en_jugador:
   in A, (JUGADOR)
   bit 0, A
   jp Z, cambio_sentido ; le pego
   jp perdio
pelota_lejos_jugador:
   in A, (JUGADOR)
   bit 0, A
   jp Z, perdio ; le pego_antes
   jp actualizo

cambio_sentido:
   ld A, (SENTIDO)
   cpl
   ld (SENTIDO), A

actualizo:
   ld IX, POSICION
   ld A, (SENTIDO)
   cp DERECHA
   jp nz, hacia_jugador
hacia_fronton:
   rr (IX)
   jp actualizo_leds
hacia_jugador
   rl (IX)
actualizo_leds:
   ld A, (POSICION)
   out (LEDS), A
   jp retorno

perdio:
   ld A, FINALIZADO
   ld (FIN), A
   out (LEDS), A

retorno:
   pop IX
   pop AF
   ei
   ret
```