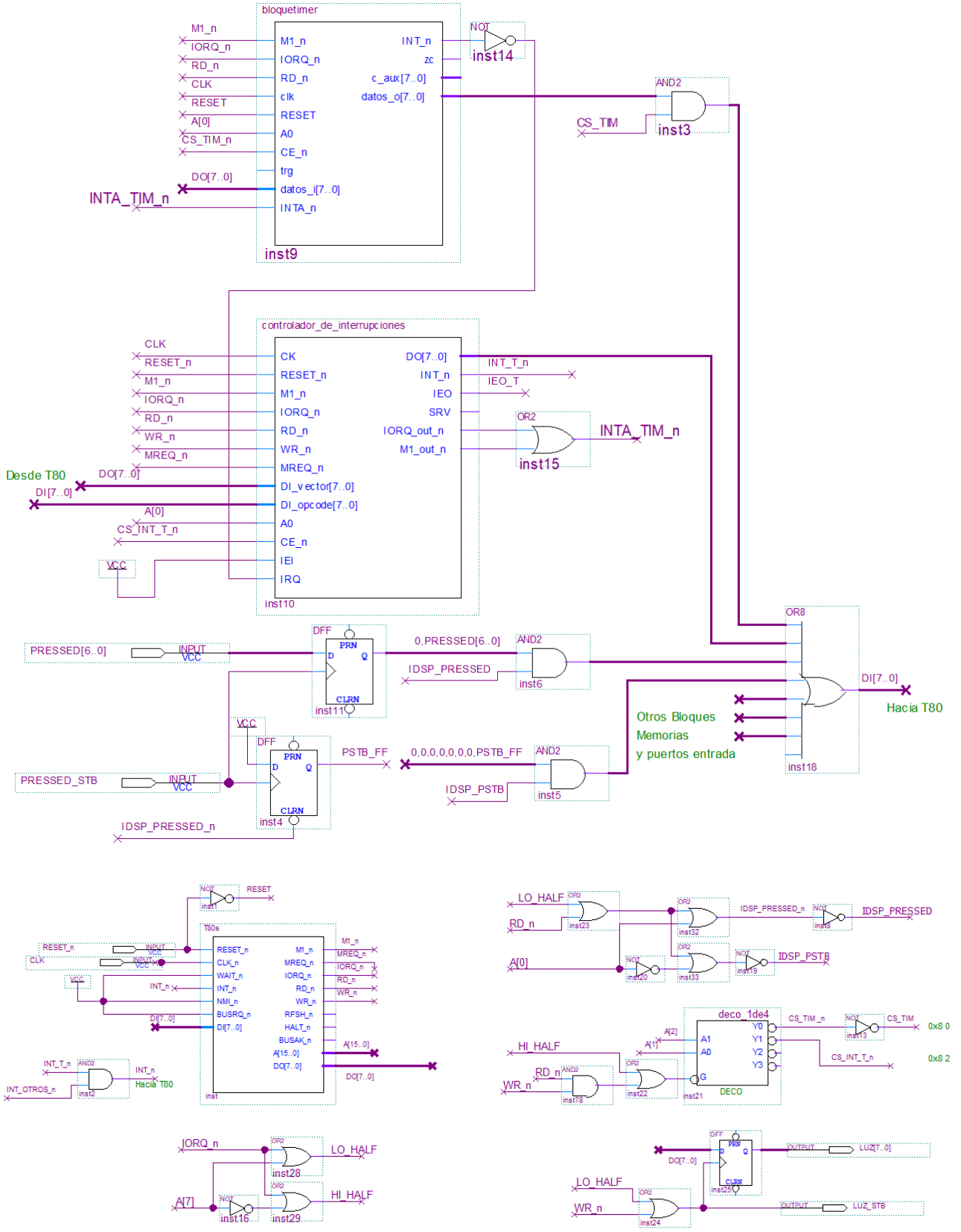
PROBLEMA 1

a) hardware



d) Inicialización, directivas, reservas

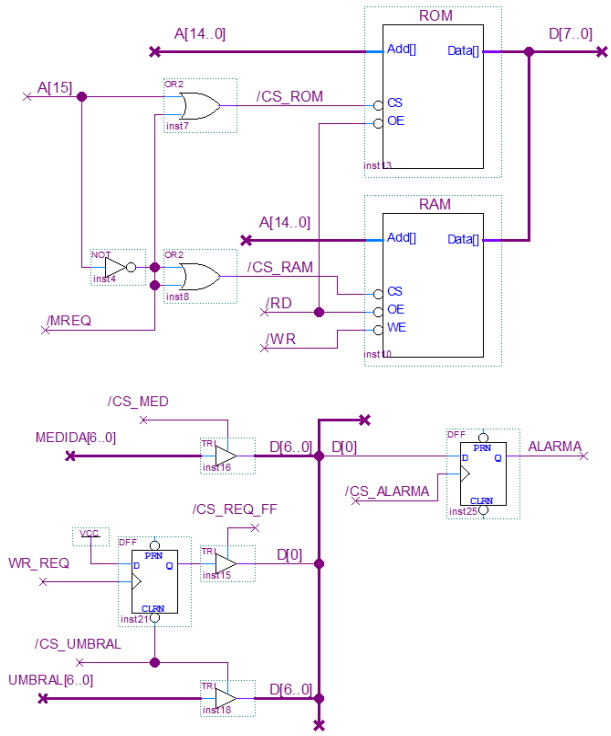
```
PRESSED         EQU 0x00 ;; in
PRESSED_STB     EQU 0x01 ;; in
LUZ             EQU 0x00 ;; out
BASE_TIMER      EQU 0x80 ;; inout
P_TIMER_PCTE    EQU BASE_TIMER
P_TIMER_PCTL    EQU BASE_TIMER+1
BASE_CINT       EQU 0x82 ;; inout
P_CI_VECTOR     EQU BASE_CINT
P_CI_STATE      EQU BASE_CINT+1

; ie, no trigger, sw reset, pre=2^13
TIMER_CTL       EQU 10101101B

TIMER_CTE       EQU 250

STATE_DEMO      equ 0
STATE_ENJUEGO   equ 1
STATE_FIN       equ 2

;; BORRO FF PRESSED_STB
;; AL LEER PRESSED

.org 0x0000
ld SP, 0
im 2
ld HL, tab_int
ld A, H
ld I, A

ld A, 0 ; // vector 0
out (P_CI_VECTOR), A
out (P_CI_STATE), A

ld A, STATE_DEMO
ld (estado), A

ld A, 0
ld (LUZ), A

;; timer
ld A, TIMER_CTE
out (P_TIMER_PCTE), A
ld A, TIMER_CTL
out (P_TIMER_PCTL), A
ei
jp main

.org 0x0100
tab_int:
dw rutint_timer

.org 0x8000
estado: db 0;
luz_encendida: db 0;
```

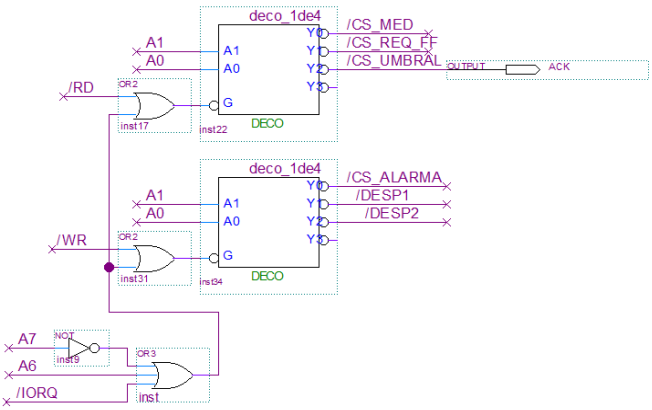
b) rutina atención a interrupción

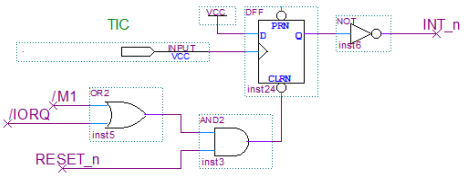
```
; rutint_timer:
; if (estado == demo) {
;   off(luz_encendida)
;   luz_encendida=(luz_encendida+1)% 128
;   on(luz_encendida)
; }
; else { ;; timeout mientras jugando,fin
del juego
;   ;; encender todas las luces al mismo
tiempo
;   for (i=0, 127) {
;       on(i)
;   }
;   estado = STATE_FIN
; }

rutint_timer:
ei
push AF
push BC
ld A, (estado)
cp STATE_DEMO
jr nz, pierde

demo:   ;; demo
    ld A, (luz_encendida)
    out (LUZ), A ;; off(luz_encendida)
    inc A
    and 0x7F; mod 128
    ld (luz_encendida), A
    or 0x80; prendo MSB para enc. luz
    out (LUZ), A
    jp fin:

pierde: ;; else
    ld B, 128
on_loop:
    ld A, B
    or 0x80 ; prendo MSB
    out (LUZ), A
    djnz on_loop
    ld A, STATE_FIN
    ld (estado), A
    di   ; no mas interrupciones
fin:
pop BC
pop AF

reti
```

d) loop ppal

```
; while (!pressed_stb)
; {
; }
; luz_pressed = IN(pressed)
; if (estado == enjuego ) {
;    if (luz_pressed == luz_encendida) {
;       ;; presiono luz correcta
;       ;; juego continua
          ; apago actual
;       off (luz_encendida)
;       luz_encendida = rand()
           ; enciendo siguiente
;       on (luz_encendida)
;       reset_timer()
          ; reseteamos timer de nuevo
;    }
;    else {
;       ;; presiono luz equivocada
;       ;; juego termina
;       estado = STATE_FIN;
;    }
; }
; else if (estado == demo) {
;       off (luz_encendida)
          ; apago actual
;       luz_encendida = rand()
;       on (luz_encendida)
           ; enciendo siguiente
;       reset_timer()
          ; reseteamos timer de nuevo
;       estado = STATE_ENJUEGO
; }
; else { ;; en fin de juego, nada que
hacer
; }
;}

main:
poll_pressed: ; espera por pressed
    in A, (PRESSED_STB)
    and 0xFF
    jr z, poll_pressed

pressed:
    in A, (PRESSED)
    ld B, A ; B = luz_presionada
    di ;protejo datos compartidos c/ isr

    ld A, (estado)
    cp STATE_ENJUEGO
    jr z, en_juego
    cp STATE_DEMO
    jr z, en_demo
fin:
    jp fin_loop

en_juego:
    ld A, (luz_encendida)
    cp B ; (luz_encendida ==
luz_presionada) ?
    jr nz, en_juego_perdio
en_juego_continua:
    out (LUZ), A ; apago actual
    call rand ; A = RAND()
    ld (luz_encendida), A
    or 0x80 ; prendo MSB para encender
luz
    out (LUZ), A ; enciendo siguiente
    out (P_TIMER_PCTL), TIMER_CTL ;;
reset timer
    jp fin_loop

en_juego_perdio:
    ld A, STATE_FIN
    ld (estado), A
    jp fin_loop

en_demo:
    ld A, (luz_encendida)
    out (LUZ), A ; apago actual
    call rand ; A = RAND()
    ld (luz_encendida), A
    or 0x80 ; prendo MSB para encender
luz
    out (LUZ), A ; enciendo siguiente
    out (P_TIMER_PCTL), TIMER_CTL ;;
reset timer
    ld A, STATE_ENJUEGO
    ld (estado), A
    jp fin_loop

fin_loop:
    ei   ; rehabilito ints
jp main
```







c) Inicialización y programa ppal

```
PI_MEDIDA     EQU 0x80
PI_REQ_FF     EQU 0x81
PI_UMBRAL     EQU 0x82
PO_ALARMA     EQU 0x80
PO_DESP1      EQU 0x81
PO_DESP1      EQU 0x82
;; ACK CON IDSP DE LECTURA DE UMBRAL

UMBRAL_INICIAL EQU 0x40
MAX_VECES      EQU 4

org 0x8000
umbral : DB
veces_bajo_umbral : DB
ultimo_despejado : DB

org 0x000
ld SP, 0x0000
im1
;inicializo variables
ld A, UMBRAL_INICIAL
ld (umbral), A
ld A, 0x00
ld (veces_bajo_umbral), A
ld (ultimo_despejado),A

;inicializo puertos
in A, (PI_UMBRAL) ; borra FF
ld A, 0
out (PO_ALARMA),A

ei
jp main

; forever{
;     si ( REQ_FF ) {
;        borro REQ_FF
;        actualizo umbral
;     }
;     otras_tareas
; }
```

```
main:
in A, (PI_REQ_FF)
bit 0, A
jp Z, otros

actualizo_umbral:
in A, (PI_UMBRAL) ; borra FF y da ACK
ld (umbral), A

otros:
call otras_tareas
jp main
```

b) Atención interrupción

```
;si (carga 2 no despejada aún){
;    si (medida < umbral){
;       veces_bajo_umbral++
;       si veces_bajo_umbral == MAX_VECES {
;          /// despejar cargas
;          veces_bajo_umbral = 0
;          alarma = 1
;          ultimo_despejado++
;          despejar carga(ultimo_despejado)
;       }
;    }
;    si no{
;       veces_bajo_umbral = 0
;    }
;}

org 0x0038
rutint:
  push AF
  push BC

;¿ya se despejo carga 2?
  ld A,(ultimo_despejado)
  cp 2
  jp fin
```

```
;¿medida debajo de umbral?
  in A, (PI_MEDIDA)
  ld B, A
  ld A, (umbral)
  cp B; umbral - medida
  jp M, sobre_umbral


;¿se llego a máximo debajo umbral?
  ld A, (veces_bajo_umbral)
  inc A
  ld (veces_bajo_umbral), A
  cp MAX_VECES
  jp NZ, fin

despejar_cargas:
  ld A, 0
  ld (veces_bajo_umbral), A
  ld A, 0xFF
  out (PO_ALARMA)
  ld A, (ultimo_despejado)
  inc A
  ld (ultimo_despejado), A
  cp 2
  jr z, despejar2
despejar1:
  out (PO_DESP1), a
  jp fin
despejar2:
  out (PO_DESP2), a
  jp fin

sobre_umbral:
  ld A,0
  ld (veces_bajo_umbral), 0
  jp fin


fin:
  pop BC
  pop AF
  ei
  ret
```