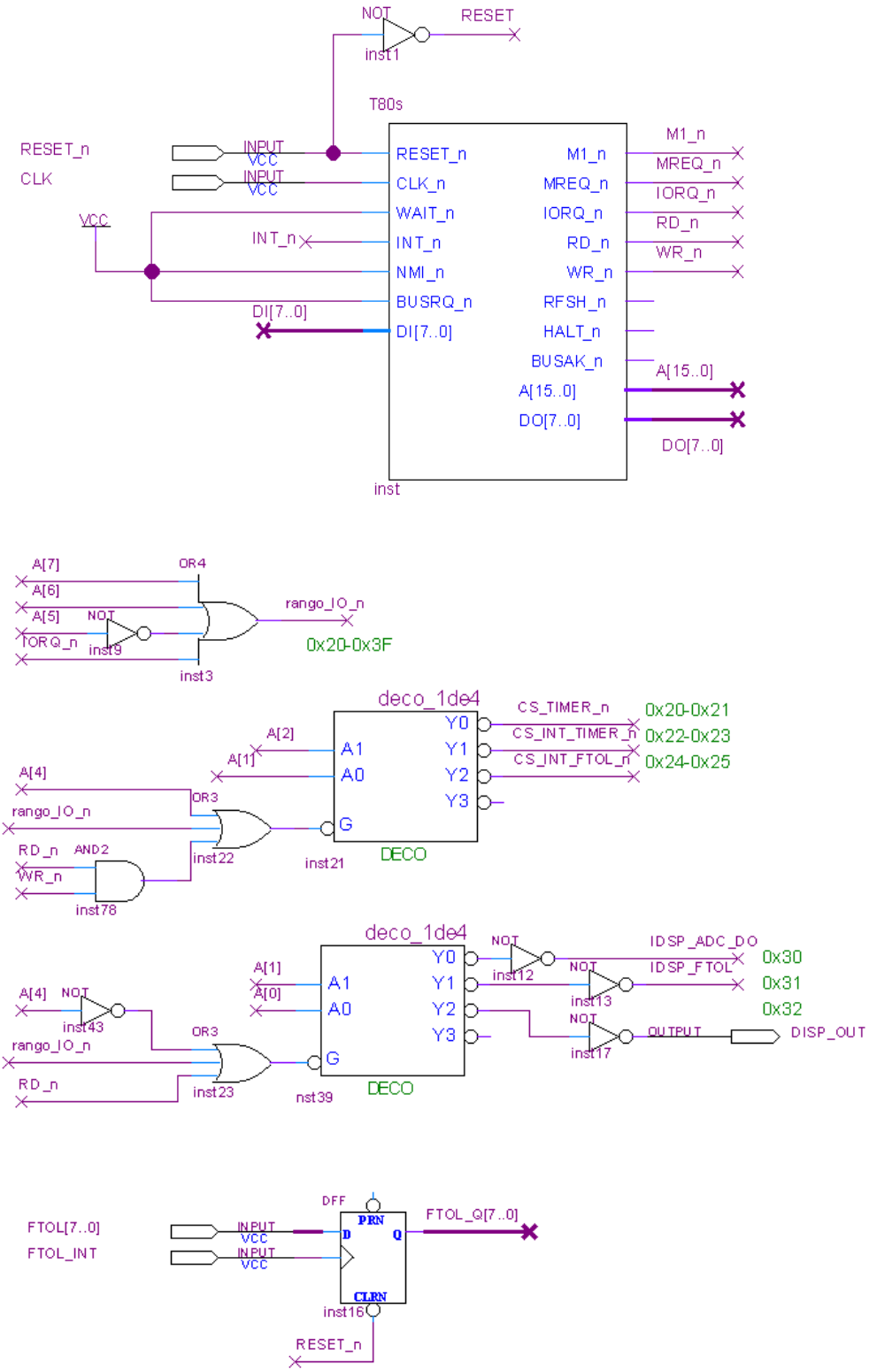
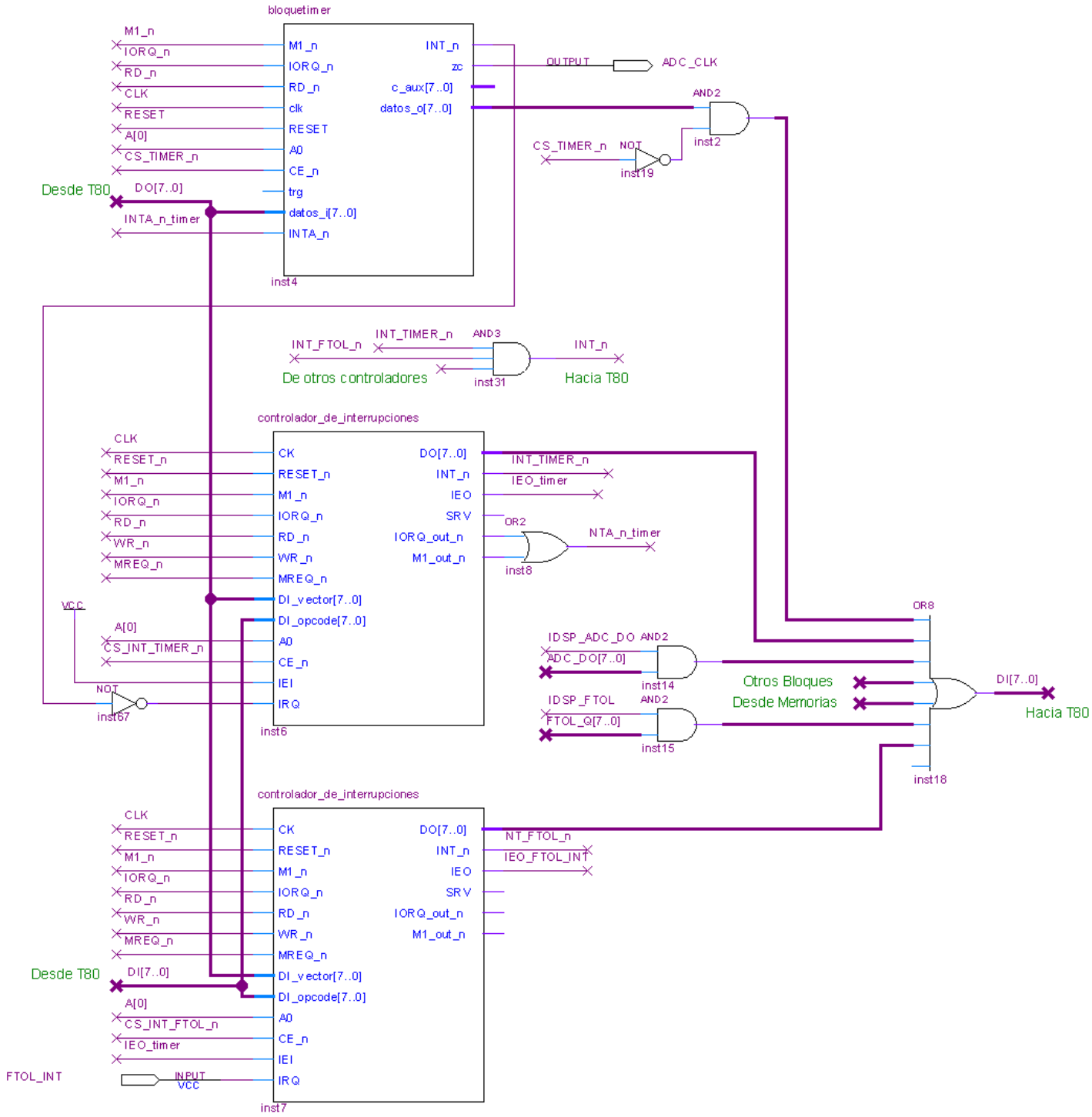
PROBLEMA 1 – Parte a)




bloquetimer
M1_n
IORQ_n
RD_n
CLK
RESET
A[0]
CS_TIMER_n
Desde T80
DO[7..0]
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
OUTPUT
ADC_CLK
AND2
inst2
CS_TIMER_n
NOT
inst19
INT_FTOL_n
INT_TIMER_n
AND3
De otros controladores
inst31
INT_n
Hacia T80
controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
VCC
A[0]
CS_INT_TIMER_n
NOT
inst67
CK
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
inst6
INT_TIMER_n
IEO_timer
OR2
NTA_n_timer
inst8
IDSP_ADC_DO
ADC_DO[7..0]
AND2
inst14
IDSP_FTOL
FTOL_Q[7..0]
AND2
inst15
OR8
Otros Bloques
Desde Memorias
DI[7..0]
Hacia T80
inst18
controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
Desde T80
DI[7..0]
A[0]
CS_INT_FTOL_n
IEO_timer
FTOL_INT
INPUT
VCC
CK
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
inst7
NT_FTOL_n
IEO_FTOL_INT

Parte b) -- isr de ftol_int y timer --

```
+-------+ 0xFFFF
|       |
| vacío |
|       |
|       |
+-------+ 0x8000 (comienza vacío)
|T.Int  | <- a partir de 0x7000
|Ocupado| <- 0x4000 a 0x4FFF
+-------+ 0x4000 (comienzo RAM)
|       |
|p.ppal | <- 0x0000 a 0x0FFF
+-------+ 0x0000 (comienzo ROM)

org 0x1000
isr_ftol_int:

;resetear bandera de cruce por -TOL
;si FTOL[7] = 0
;     configurar timer período 1ms
;sino
;     configurar timer período 10ms
;fin_si FTOL[7] = 0
;TOL = FTOL and 0x0F
;TOL_neg = neg(tol)

      ei
      push af
      push bc
      ld a,0x00
      ld (cruce_TOL_neg),a
      in a, (FTOL)
      ld b,a
      bit 7,a
      jp z, conf_1ms

conf_10ms:
      ld a,cuenta_10ms
      jr conf_timer

conf_1ms:
      ld a,cuenta_1ms

conf_timer:
      out (TIMER),a
      ld a, cw_timer
      out (TIMER+1),a ;restart cuenta
      ld a,b
      and 0x0f
      ld (TOL),a
      neg
      ld (TOL_neg),a
      pop bc
      pop af
      reti
```

```
isr_timer:

; si ya se cruzó -TOL
;     lee valor de ADC
;     si valor leído > TOL
;       pulso en DISP_OUT
;       set   bandera DISP_FLAG
;       reset bandera de cruce por -TOL
;       fin
;     sino
;       fin
; sino
;     lee valor de ADC
;     si valor leído < -TOL
;       set bandera de cruce por -TOL
;       fin
;     sino
;       fin

      ei
      push af
      push bc
      ld a, (cruce_TOL_neg)
      cp 0x00
      jp z, esperando_TOL_neg

esperando_TOL:
      in a, (ADC_DO)
      ld b, a
      ld a, (TOL)
      cp b  ;TOL - valor leído
      jp P, fin

disparo_detectado:
      ld a, 0xFF
      ld (DISP_FLAG),a
      in a, (DISP_OUT),a ;pulso con deco
      ld a, 0x00
      ld (cruce_TOL_neg),a
      jp fin

esperando_TOL_neg:
      in a, (ADC_DO)
      ld b, a
      ld a, (TOL_neg)
      cp b  ;TOL_neg - valor leído
      jp P, fin

seteo_TOL_neg:
      ld a, 0xFF
      ld (cruce_TOL_neg),a

fin:
      pop bc
      pop af
      reti
```

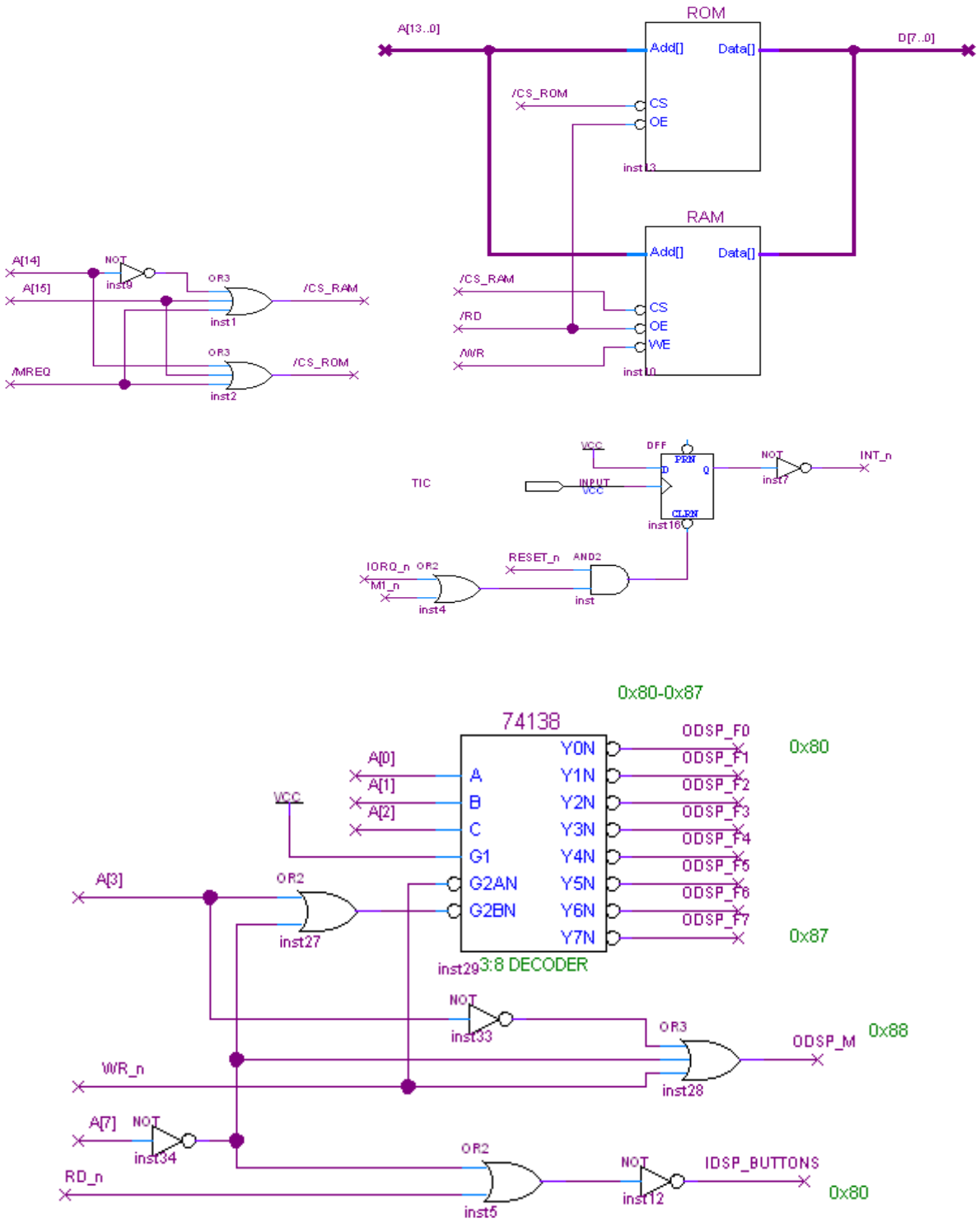
Parte c) ----- INIT_DET_DISP -----

```
org 0x1500
INIT_DET_DISP:
; cargo tabla int
; inicializo variables:
;   - cruce_TOL_neg
;   - TOL y TOL_neg
; configuro cont. Int timer
; configuro cont. Int Ftol


push af
push ix
push hl

ld ix, INICIO_T_INT
ld hl, isr_timer
ld (ix + vi_timer), L
ld (ix + vi_timer + 1), H
ld hl, isr_ftol_int
ld (ix + vi_ftol), L
ld (ix + vi_ftol + 1), H

ld a, 0x00
ld (cruce_TOL_neg), a
ld a, TOL_INI
ld (TOL), a
neg
ld (TOL_neg), a

ld a, vi_timer
out (INT_TIMER), a ;conf. vector
out (INT_TIMER+1), a ;borra I pendiente
ld a, vi_ftol
out (INT_FTOL), a ;conf. vector
out (INT_FTOL+1), a ;borra I pendiente

ld a, palabra_1ms
out (TIMER),a
ld a, cw_timer
out (TIMER+1),a

pop hl
pop ix
pop af
ret
```

```
org 0x5000
      ;; variables
cruce_TOL_neg     : DB
TOL               : DB
TOL_neg           : DB
DISP_FLAG         : DB

      ;; definicion de ctes
TOL_INI           EQU   0x0F ;val. inic.
TIMER             EQU   0x20
INT_TIMER         EQU   0x22
INT_FTOL          EQU   0x24
ADC_DO            EQU   0x30
FTOL              EQU   0x31
DISP_OUT          EQU   0x32

INICIO_T_INT      EQU   0x7000

vi_timer          EQU   0x08
vi_ftol           EQU   0x0A
;CW:
;  - int. Habilitada (1)
;  - trigger no importa (0)
;  - sw reset (1)
;  - arranque auto(0)
;  - prescaler 2^12 = 4096 = 1100b
cw_timer          EQU   10101100

; 1ms = 1x10^-3 s
; => cte * (fclk/prescaler)^-1 = 1x10^-3
; => cte = 40960*10^3 / 4096 * 1x10^-3
; => cte = 10
cuenta_1ms        EQU 9  ; 10-1
cuenta_10ms       EQU 99 ; 100-1
```

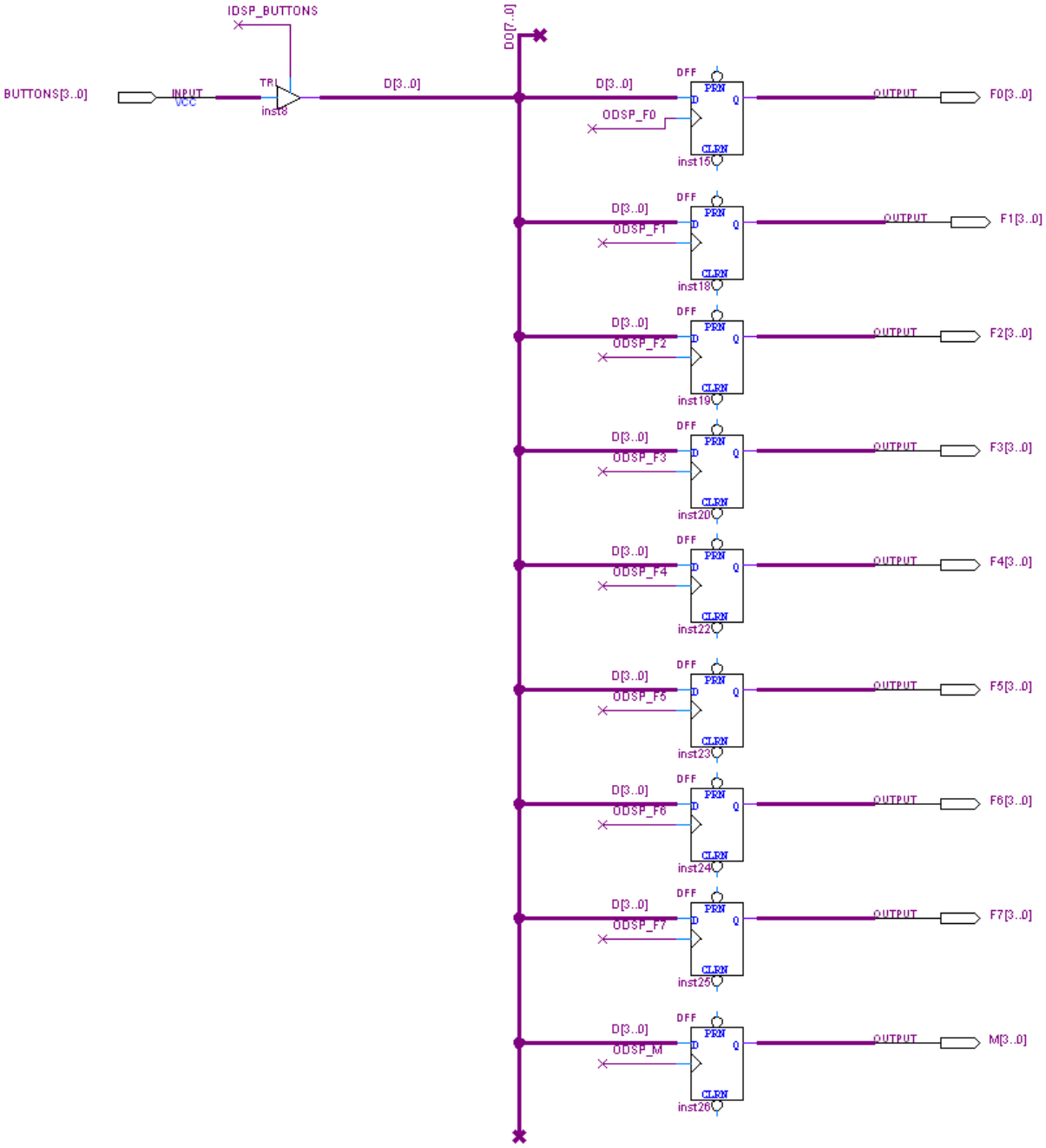
PROBLEMA 2 – Solución

a) Hardware




```
A7 A6 A5 A4 A3 A2 A1 A0
 1  x  x  x  0  0  0  0 - fila0 (F0)
 1  x  x  x  0  0  0  1 - fila1
 1  x  x  x  0  0  1  0 - fila2
...
 1  x  x  x  0  1  1  1 - fila7 (F7)
 1  x  x  x  1  x  x  x - rock_meter (M)
```

1 x x x x x x x – Buttons (128 fantasmas)



b) Software

```
;; CONSTANTES
; Puertos de entrada
BUTTONS     EQU 80
;Puertos de salida
F0    EQU 80
F1    EQU 81
F2    EQU 82
F3    EQU 83
F4    EQU 84
F5    EQU 85
F6    EQU 86
F7    EQU 87
M     EQU 88

0RG 0x0000
 LD SP, 0x8000  ; 16Krom + 16Kram
 IM 1
 LD A, 0
 LD B, 8
 LD C, F0
 LD HL, fila0
init:
 LD (HL), A
 OUT ( C ), A
 INC HL
 DJNZ init
 LD (toggle), A
 LD (game_over), A
 LD A, 5
 LD (rock_meter), A
 OUT (M), A
 EI
 JP prog_ppal

; ;;; at. interrupcion
; si (game_over != true)
;   toggle = ! toggle
;   si fila0 != buttons {
;     ROCK_METER--
;     si ROCK_METER==0 {
;       game_over=true
;     }
;   }else{
;     ROCK_METER++
;     si ROCK_METER= 10 {
;       game_over = true
;     }
;   }
;   si (game_over != true)
;     ;; desplazo
;     for (i=0, i<7, i++){
;       fila[i] = fila[i+1]
;     }
;     si toggle == true {
;       fila7 = random()
;     }else{
;       fila7 = 0000
;     }
;   }
; }
; retorno

ORG 0x38
rutint_tic:
  PUSH AF
  PUSH BC
  PUSH IX
  LD A, (game_over)
  CP 0xFF
  JP Z, fin
;;; si (game_over != true)
  LD A, (toggle)  ; cambio toggle
  CPL
  LD (toggle), A

  IN A, (BUTTONS)
  AND 0x0F
  LD B, A
  LD A, (fila0)
  CP B
  JP Z, sumar
;;;   si fila0 != buttons {
;;;     ROCK_METER--
  LD A, (rock_meter)
  DEC A
  ld (rock_meter), a
  OUT (M), A
  CP 0
  JP NZ, mover
;;;     si ROCK_METER==0 {
;;;       game_over=true
  LD A, 0xFF
  LD (game_over), A
  JP fin
sumar:
;;;   }else{
;;;     ROCK_METER++
  LD A, (rock_meter)
  INC A
  LD (rock_meter), A
  OUT (M), A
  CP 10
  JP NZ, mover
;;;     si ROCK_METER= 10 {
;;;       game_over = true
  LD A, 0xFF
  LD (game_over), A
  jp fin
mover:
;;;      si (game_over != true)
;;;         for (i=0, i<7, i++){
;;;            fila[i] = fila[i+1]
  LD C, F0
  LD B, 7
  LD IX, fila0
iterar:
  INC IX
  LD A, (IX)      ; en A F[i+1]
  LD (IX-1), A    ; F[i]=F[i+1]
  OUT (C), A      ; escribo puerto i
  INC IX
  INC C
  DJNZ iterar
```

```
  LD A, (toggle)
  AND 0xFF
  JP Z, ceros
;;;     si toggle == true {
;;;       fila7 = random()
  CALL RANDOM
  AND 0x0F
  JP seguir
ceros:
;;;     }else{
;;;       fila7 = 0000
  LD A, 0
seguir:
  LD (fila7), A
  OUT (F7), A
fin:
  POP IX
  POP BC
  POP AF
  EI
  RETI

; Variables
ORG 0x4000

fila0: db
fila1: db
fila2: db
fila3: db
fila4: db
fila5: db
fila6: db
fila7: db
toggle: db
game_over: db
rock_meter: db

.end
```