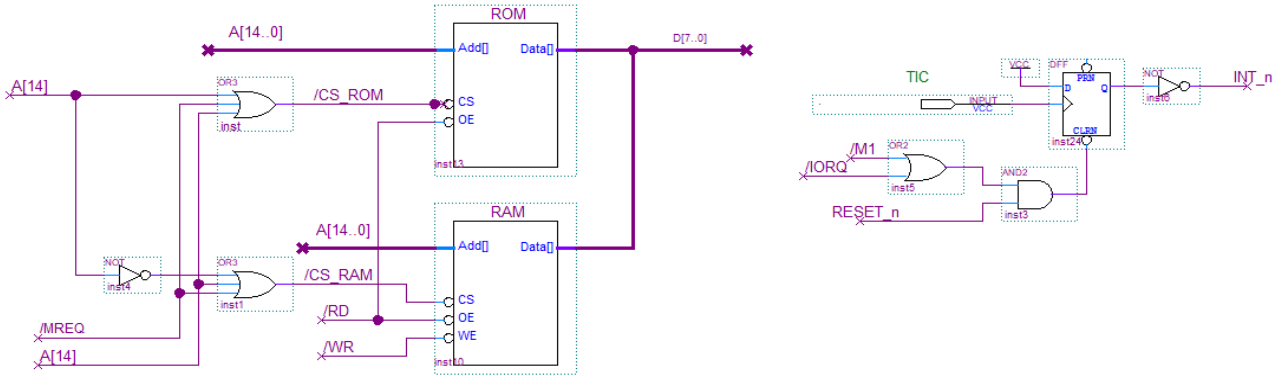
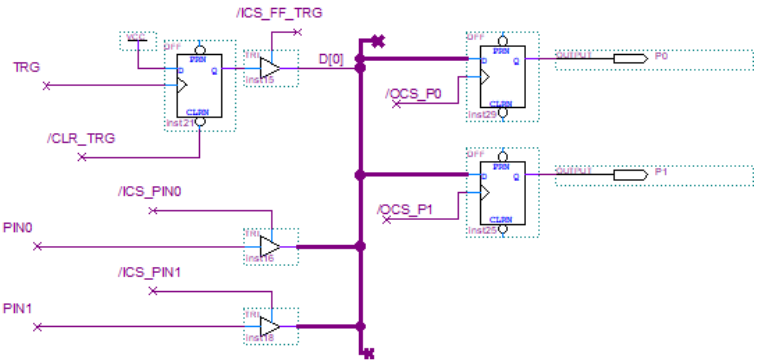
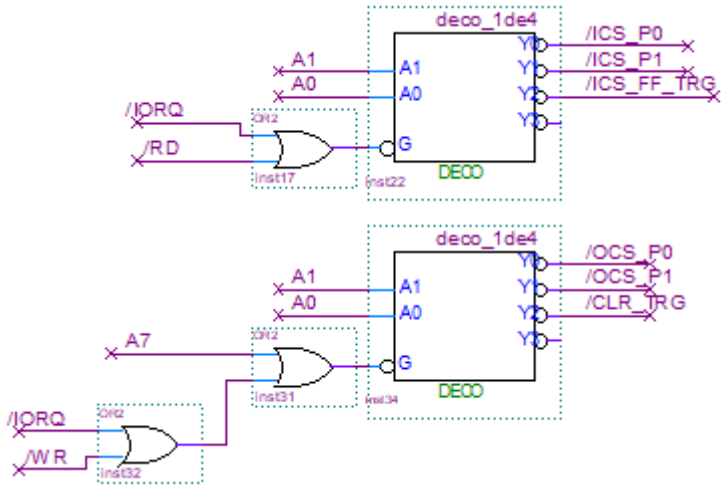
PROBLEMA 1 - a) Hardware





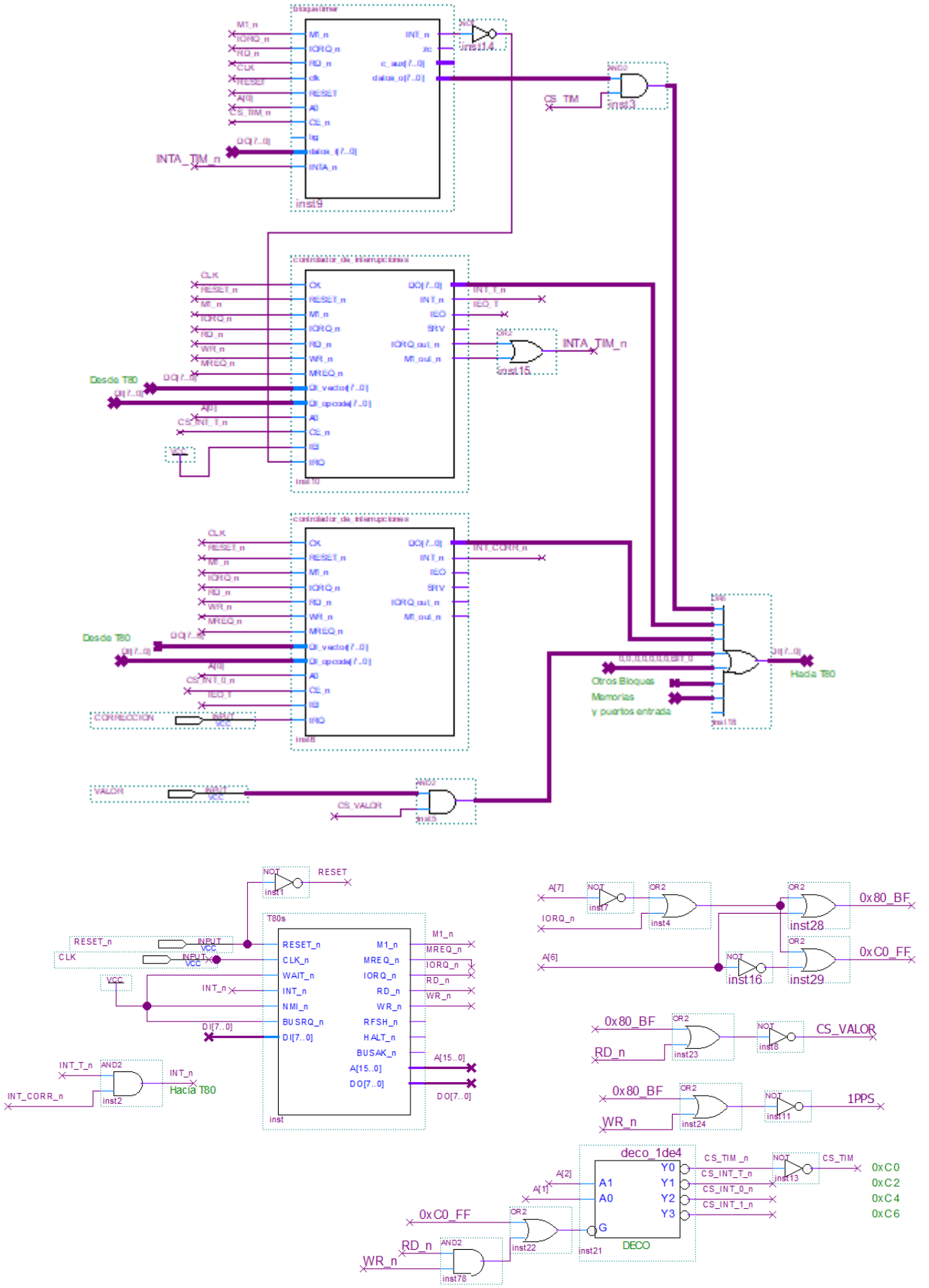
0xFFFF 0x8000	Reservado 32KB
0x7FFF 0x4000	RAM 16KB
0x3FFF 0x0000	ROM 16KB




```
b) subrutina chk_trg
;; si flanco trg{
;;     borro FF_trg
;;     si test_on{
;;        disable interrupts
;;        test_on = false
;;     }else{
;;        puntero=tabla_sub
;;        enable interrupts
;;        test_on = false
;;     }
;; }
chk_trg:
   in a, (ff_trg)
   bit 0, a
   jr z, finsi_trg
;; si flanco trg{
;;     borro FF_trg
   out (clr_trg), a
   ld hl, test_on
   ld a, 0xFF
   cp (hl)
   jr nz, else_test_on
;;     si test_on{
;;        disable interrupts
;;        test_on = false
   di
   ld a, 0
   ld (hl), a
   jr finsi_trg
else_test_on:
;;     }else{
;;        puntero=tabla_sub
;;        enable interrupts
;;        test_on = false
```

```
   ld (hl), a
   ld hl, tabla_sub
   ld (puntero), hl
   ei
finsi_trg:
   ret

;; c) Rutina servicio interrupcion

;; dir = mem[puntero]
;; callsub_hl(dir)
;; si puntero == fin_tabla_sub-2{
;;   puntero = tabla_sub
;; }else{
;;   incremento puntero
;; }

   org 0x0038
isr:
   push af
   push hl
   push ix
   push bc
   ld ix, puntero
   ld h, (ix+0)
   ld l, (ix+1)
   push ix
   call callsub_hl
   pop bc     ;; bc = puntero

   ld hl, fin_tabla_sub-2
   scf
   ccf   ;; carry = 0
   sbc hl, bc
   ld a, l
   or h
   jr nz, else_fintabla
;; si puntero == fin_tabla_sub-2{
;;   puntero = tabla_sub
   ld ix, tabla_sub
   ld l, (ix+0)
   ld h, (ix+1)
   jr finsi_fintabla
else_fintabla:
;; }else{
;;   incremento puntero
   inc ix
   inc ix
finsi_fintabla:
   ld (puntero), ix
   pop bc
   pop ix
   pop hl
   pop af
   ret

d) inicialización y reserva mem
      org 0
   im 1
   ld sp, 0x8000  ;; tope ram + 1
     ;; variables
       ;;   test_on = false
   ld a, 0
   ld (test_on), a
     ;; puertos
        ;;   ff_trg
   out (ff_trg), a
     ;; principal
   call init_principal
     ;; no habilito int hasta flanco trg
   jp main

;; rutina serv. interrupcion
   org 0x0038
isr:
   ;; ver parte c)
   …

;; programa principal

main:
   call chk_trg
   call principal
   jr main

;; subrutina chk_trg
chk_trg:
   ver parte b)

;; tablas, subrutinas
    ;; tabla_sub con tres subrutinas
tabla_sub:
   dw sub_0
   dw sub_1
   dw sub_2
fin_tabla_sub:

callsub_hl:
   jp (hl)

;; subrutinas de test
sub_0:
   ...
init_principal:
   …

     org 0x4000
puntero: dw 0
test_on: db 0

ff_trg  equ  2
clr_trg equ  2
```

PROBLEMA 2 - a) Hardware



b)

```
CI_TIMER_VI_AD    equ 0xC2
CI_TIMER_RESET_AD equ 0xC3
CI_CORR_VI_AD     equ 0xC4
CI_CORR_RESET_AD  equ 0xC5

CI_TIMER_VI       equ 0
CI_CORRECCION_VI  equ 2

TIMER_CTE_AD      equ 0xC0
TIMER_CW_AD       equ 0xC1
DEFAULT_TIMER_CTE equ 80
TIMER_CW          equ 0x1010 1111

VALOR_AD          equ 0x80
1PPS_AD           equ 0x80

org 0300h ; en ROM
Tabla_int:
  DW rutint_timer
  DW rutint_correccion

org 8000h // RAM
  NUEVA_CTE_TIMER  db
  ACTUAL_CTE_TIMER  db

org 0000h
  ld SP, 0000h
  ld A,Tabla_int/256
  ld I,A
  IM2

  call init_ci_timer
  call init_ci_correccion

  ld A, DEFAULT_TIMER_CTE
  ld (NUEVA_CTE_TIMER), A
  ld (ACTUAL_CTE_TIMER), A
  call reset_timer

  ei
  jp main

init_ci_timer:
 ld A, CI_TIMER_VI
 out (CI_TIMER_VI_AD), A
 ; borro peticiones pendientes
 out (CI_TIMER_RESET_AD), A
 ret

init_ci_correccion:
 ld A, CI_CORRECCION_VI
 out (CI_CORR_VI_AD), A
 ; borro peticiones pendientes
 out (CI_CORR_RESET_AD), A
 ret

reset_timer:
 ld A,(ACTUAL_CTE_TIMER)
 out (TIMER_CTE_AD),A
 ld A,(TIMER_CW)
 out (TIMER_CW_AD),A
 ret
```

c)

```
org algun_lugar_rom
rutint_timer:
   ; Da pulso en 1PPS
   ; si la constante actual es igual a
la nueva
   ;   retorna sin hacer nada. El timer
continua con el valor
   ; sino
   ;   si actual - nueva > 0
   ;      actual = actual -1
   ;   sino
   ;      actual = actual + 1
   ;   reinicio timer
   ; retorno

   ei
   push AF
   push BC

   out (1PPS_AD),A

   ld A,(NUEVA_CTE_TIMER)
   lb B,A
   ld A,(ACTUAL_CTE_TIMER)
   cp B   ; A-B = actual - nueva. Si > 0
resto 1, sino sumo 1
   jp Z, no_corregir

corregir:
   jp P, resto
sumo:
   inc A
   ld (ACTUAL_CTE_TIMER), A
   jp reseteo
resto:
   dec A
   ld (ACTUAL_CTE_TIMER), A

reseteo:
   call reset_timer

no_corregir:
   pop BC
   pop AF
   reti


rutint_correccion:
   ; leo valor a corregir
   ; nueva = nueva + valor a corregir
   ;   (esta en comp a 2 y no es
   ;   necesario chequear OverFlow)
   ei
   push AF
   in A, (VALOR_AD)
   ld B, A
   ld A, (NUEVA_CTE_TIMER)
   add A,B
   ld (NUEVA_CTE_TIMER), A
   pop AF
   reti
```