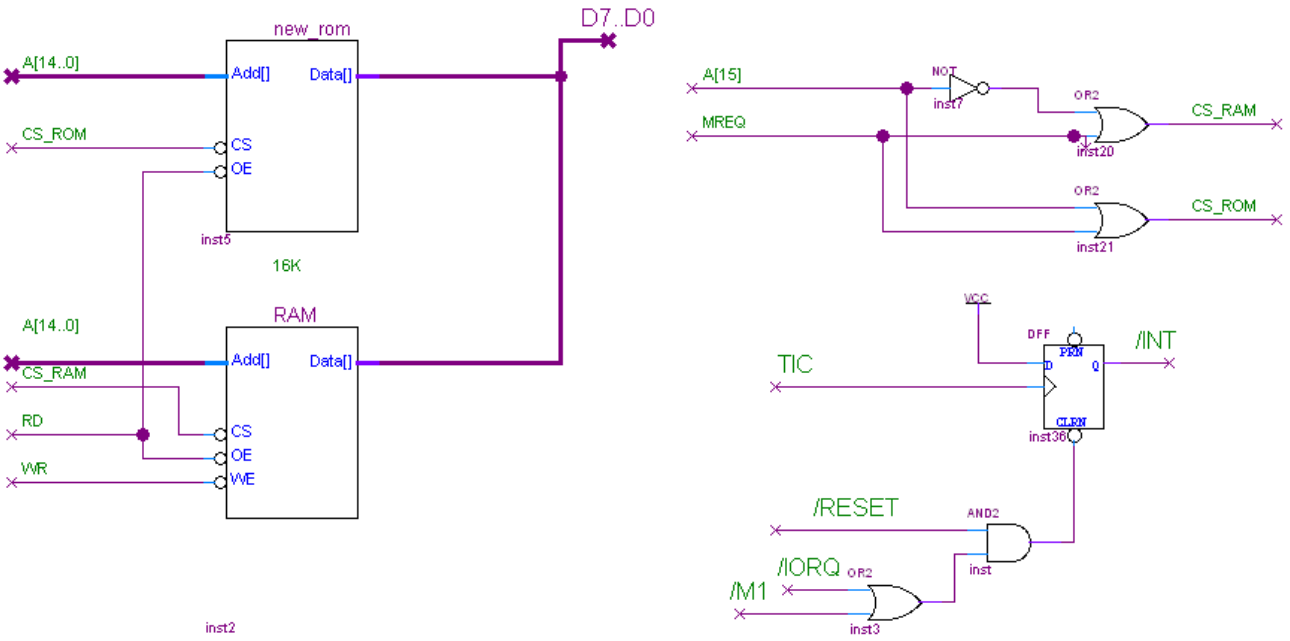
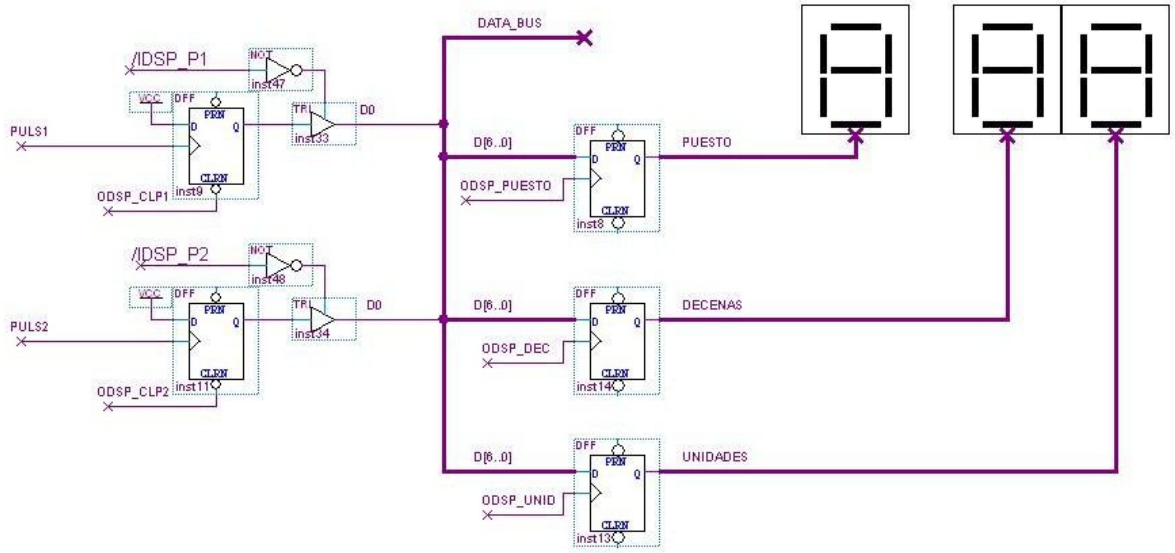
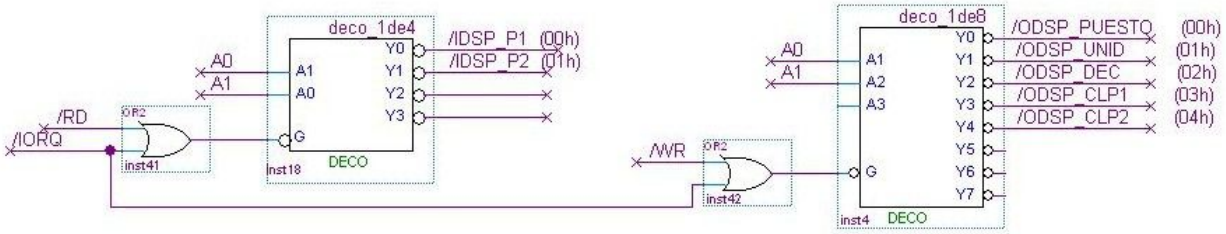
PROBLEMA 1 – Solución

a)
32K RAM (8000h a FFFFh)
32K ROM (0000h a 7FFFh)







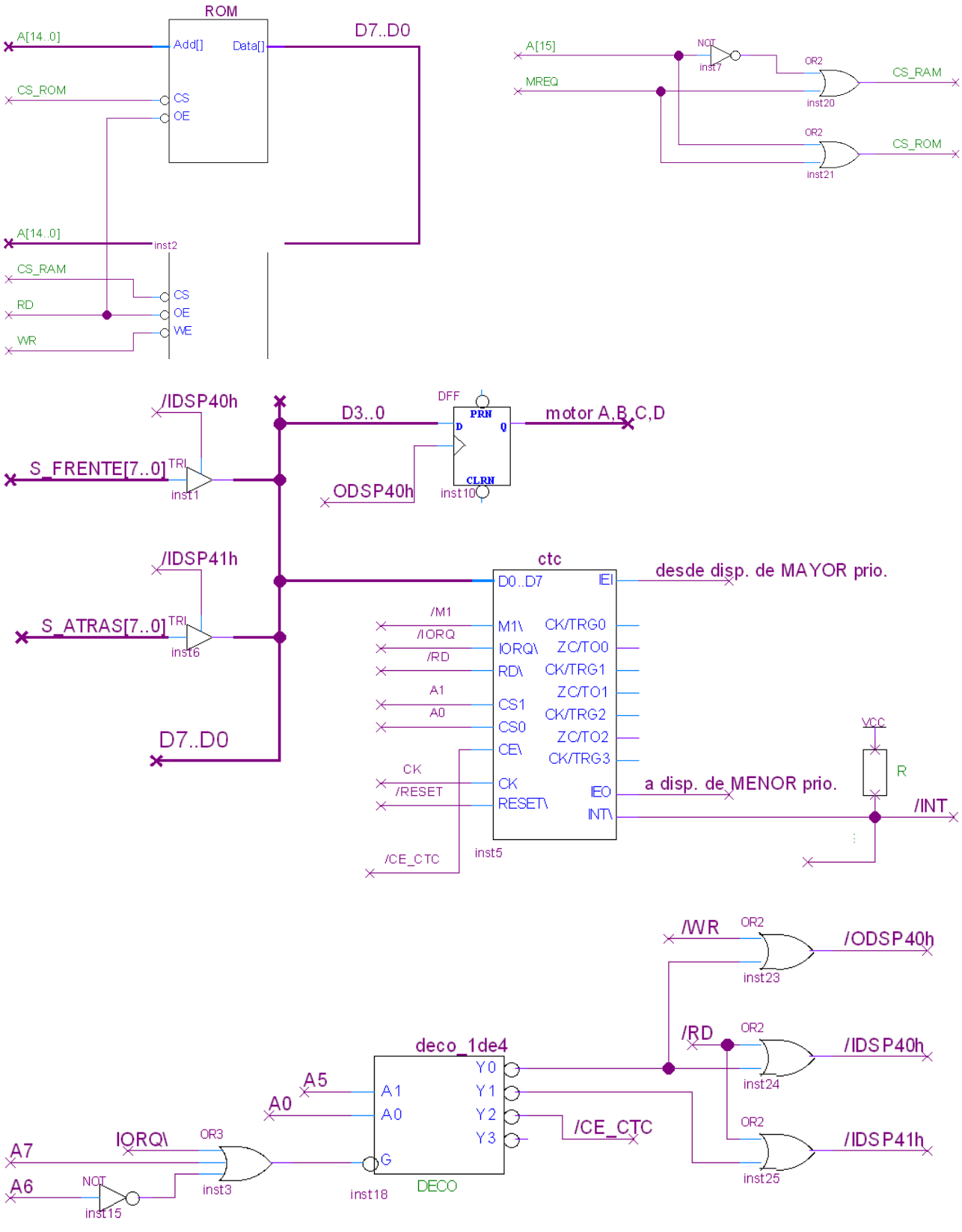
parte b)

```
; INICIALIZACIÓN
                org 0000h
                ld SP,0           ; Inicializo stack
                im 1              ; Interrupciones modo 1
                ld a,0
                ld (Tiempo),a     ; Inicializo tiempo en 0
                ld (NUM),a        ; Inicializo num en 0
                call bin2bcd
                ld a,c
                out(puesto), a
                out(Unidades), a
                out(Decenas), a
                ei                ; Habilito interrupciones
                jp PPAL

; DEF. CONSTANTES
                Puesto1     equ   01h
                Puesto2     equ   02h
                Pulsador1   equ   00h
                Pulsador2   equ   01h
                Puesto      equ   00h
                UNIDADES    equ   01h
                DECENAS     equ   02h
                ClrPuls1    equ   03h
                ClrPuls2    equ   04h
; RESERVA MEM
                org 8000h
                NUM:        DB
                Tiempo:     DB


PPAL:
PREG_POR_1:     in a,(Pulsador1)  ; Adquiero el pulsador 1
                bit 0,a           ; Me fijo si fue presionado
                jr z,PREG_POR_2   ; Si no se presionó, pregunto por el 2
                out (ClrPuls1),a  ; Borro el FF del pulsador 1.
                 ld A, (Tiempo)
                 cp 00h           ; Me fijo si estamos en los 5 segundos
                jp Z, LLAMO_POR_1
ENCOLA_1:       ld a, Puesto1     ; Cargo el puesto en el acumulador
                call put_fifo     ; Encolo puesto 1
                jr PREG_POR_2
LLAMO_POR_1:    ld a, Puesto1     ; Cargo el puesto en el acumulador
                call ACTUALIZA

PREG_POR_2:
                in a,(Pulsador2)  ; Adquiero el pulsador 1
                bit 0,a           ; Me fijo si fue presionado
                jr z,PREG_POR_1   ; Si no se presionó, pregunto por el 2
                out (ClrPuls2),a  ; Borro el FF del pulsador 1.
                 ld A, (Tiempo)
                 cp 00h           ; Me fijo si estamos en los 5 segundos
                jp Z, LLAMO_POR_2
ENCOLA_2:       ld a, Puesto2     ; Cargo el puesto en el acumulador
                call put_fifo     ; Encolo puesto 2
                jr PREG_POR_1
LLAMO_POR_2:    ld a, Puesto2     ; Cargo el puesto en el acumulador
                call ACTUALIZA

                jr PPAL



org 0200h
```

```
ACTUALIZA:                      ; Subrutina actualiza, necesita el puesto
                                ; en el acumulador
                push bc
                call bin27seg   ; Paso el puesto a 7seg, supongo que deja
                                ; el acumulador sin modificar
                 ld a, c
                out (Puesto),a  ; Actualizo el display del puesto
                ld a, (NUM)     ; Cargo el número de cliente
                cp 100          ; Me fijo si es 100
                jr nz, NO_ES_100 ; Si no es 100, sigo
                ld a, 0         ; Si es 100, lo paso a 0
NO_ES_100:
                call bin2_7seg  ; Lo paso a 7seg, supongo que deja el
                                ; acumulador sin modificar
                inc a           ; Incremento el número de cliente
                ld (NUM),a      ; Guardo el número de cliente
                 ld a, b
                out (Decenas),a ; Actualizo decenas
                ld a, c
                out (Unidades),a ; Actualizo unidades

                ld a,5          ; Inicio cuenta de tiempo
                ld (Tiempo),a
                pop bc
                ret


Parte c)

                org 38h
ISR_TIME:
                push af         ; Preservo estado
                ld a,(Tiempo)
                cp 0
                jp Z, FIN

                dec a           ; Decremento el tiempo
                ld (Tiempo), a  ; Guardo el tiempo decrementado
                cp 0            ; ACTUALIZO EN LA 5 INTERRUPCIÓN
                jr nz, FIN      ; Si no pasaron 5 interrupciones desde que
                                ; actualicé, terminé
                call get_fifo   ; Si pasaron, me fijo si hay algo en la
                                ; cola para actualizar
                cp ffh
                jr z, FIN_2     ; Si no hay nada pongo la variable tiempo
                                ; en 0 y salgo.
                call actualiza  ; Si hay algo actualizo y pongo en 0 tiempo
                jr FIN
FIN_2:
                ld a,0
                ld (Tiempo),a
FIN:
                pop af
                ei
                ret
```

PROBLEMA 2 – Solución

a) Hardware
- memoria: 32K + 32K
- CTC para generar la interrupción. Un solo canal modo timer, pre = 256, cte calculada según |error|.
- sensores (dos puertos de entrada de 8 bits siempre listos)
- motor (puerto salida de 4 bits)



b) Prog. Principal

```
-- variables globales:
modo: -1 retroceso, 0 detener, 1 avance
deltat: valor del próximo período del
timer.

forever{
    leo sensores
    calculo diferencia
    modo = +1
    si (diferencia < 0 ) {
        modo = -1
        diferencia = - diferencia
    }
    si (diferencia < umbral){
        modo = 0
    }
    deltat = 255 - diferencia

    call atiendo_otros
}
```

```
forever:
    ld c, 1
    in a, (s_atras)
    ld b, a
    in a, (s_frente)
    sub b
    ld b, a
    jp P, finsi1
        ; si (diferencia < 0 ) {
        ;     modo = -1
        ;     diferencia = - diferencia
    neg
    ld b, a
    ld a, -1
    ld c, a
finsi1:
    ld a, b
    sub UMBRAL
    jp P, finsi2
        ; si (diferencia < umbral){
        ;     modo = 0
    ld a, 0
    ld c, a
finsi2:
    ld a, c
    ld (modo), a
        ; deltat = 255 - diferencia
    ld a, 255
    sub b
    ld (deltat), a
    call atiendo_otros
    jp forever
```

c) Rutina atención interrupción

```
; preservar estado
; actual = (actual + modo + 4) mod 4
; puerto = tabla[actual]
 // si modo = 0 mantiene valor y queda
detenido
; posición = posición + modo
  // ext. signo al llevar modo a 16 bits
; reprogramar CTC con deltat
; restaurar estado

    org 0x0200
rutint:
    ei
    push af
    push bc
    push hl
        ; inc actual según modo
        ; mód 4
    ld a, ( actual )
    add 4
    ld b, a
    ld a, ( modo )
    add b
    and 00000011B
    ld ( actual ), a
        ; puerto = tabla[actual]
    ld c, a
    ld b, 0  ; bc = actual
    ld hl, tabla
    add hl, bc ; tabla + actual
    ld a, ( hl )
    out (puerto), a

        ; posición = posición + modo
    ld hl, (posicion)
    ld a, (modo)
    cp -1
    jr nz sigo1
    dec hl
sigo1:
    cp 1
    jr nz sigo2
    inc hl
sigo2:
    ld (posicion), hl

        ; reprogramar ctc
        ; valor inic = deltat
    ld a, CW_CTC
    out (CTC0), a
    ld a, (deltat)
    out (CTC0), a

    pop hl
    pop bc
    pop af
    reti
```

e) Inicialización

```
    ld sp, 0    ; stack
        ; interrupciones
    im 2
    ld a, tabint_hi
    ld i, a


        ; CTC, canal 0 en modo timer
    ld a, vector  ; vector
    out (CTC0), a
    ld a, CW_CTC  ; canal 0
    out (CTC0), a
    ld a, 0       ; cte inicial = 256
    out (CTC0), a

        ; variables y puertos
    ld a, 0
    ld (modo), a
    ld (deltat), a
        ; actual = 0
        ; puerto con tabla(actual)
    ld (actual), a
    ld a, (tabla)
    out (puerto), a
        ; posición = 0
    ld hl, 0
    ld (posicion), hl
    ei

        ; jp a prog. principal
    jr forever
```

```
; tabla secuencia motor
    org 0x0400  ; algun lugar de rom
tabla:
    db 00001100B
    db 00000110B
    db 00000011B
    db 00001001B

; tabla interrupciones
    org tabla_hi * 256  ; 0x1000
tabla_int:
    DW xxxx  ; isr de otros
dispositivos
    DW yyyy
    DW
    DW
    DW rutint ; canal 0

; variables
    org 0x8000
posicion: DW
modo:     DB
actual:   DB
deltat:   DB


tabla_hi EQU 0x10
s_frente EQU 0x40
s_atras  EQU 0x41
p_motor  EQU 0x40
CTC0     EQU 0x60
    ; offset 8, vector 4 en adelante
vector   EQU 0x08
    ; control word
      ; 1 ei
      ; 0 modo timer
      ; 1 pre=256
      ; x edge don't care
      ; 0 trigger auto
      ; 1 sigue time constant
      ; 1 soft reset
      ; 1 control word
CW_CTC   EQU 10100111B
```