

- | | |
|--|--|
| <ul style="list-style-type: none">• Mínimo para aprobar: un problema• Nombre y CI en cada hoja• Numere las hojas, indique el total en la primera | <ul style="list-style-type: none">• Utilice sólo un lado de las hojas• Incluya un solo problema por hoja• Sea prolijo |
|--|--|

Si usa las directivas `.text` y `.data` indicar la dirección de comienzo de cada sección.

PROBLEMA 1

Se desea agregar un sensor de temperatura a un sistema existente basado en el procesador T80. Para esto se deberá agregar hardware y también software: rutinas de atención a interrupciones y subrutina **iniciar_medida** que será invocada desde el programa principal ya existente.

Cuando se invoque la subrutina **iniciar_medida**, se espera que el sistema lea la entrada **sensor_conectado** y, en caso que valga 1, dé un flanco de subida en la entrada ***start*** del sensor para iniciar una medida.

Cuando el sensor recibe el flanco en **start**, genera un pulso a 0 en su salida **temp**. La duración del pulso medida en intervalos de **delta** = 64us es el valor de la temperatura. El resto del tiempo la salida **temp** vale 1.

Para medir la duración del pulso se utilizará un bloque timer que debe configurarse para arrancar con el flanco de bajada del pulso en **temp**. Se utilizarán interrupciones para detectar las dos posibles condiciones de finalización de medida:

- la finalización del pulso (flanco de subida en **temp**), en cuyo caso debe escribirse el valor de temperatura en la salida **T[8]** y un 1 lógico en la salida **OK**.

- la expiración del timer, que indicará que el pulso tiene una duración mayor que $60 \times \text{delta}$. En este caso debe escribirse un 1 lógico en la salida **OVF**.

En ambos casos los valores de **OK** y **OVF** deben mantenerse hasta que se invoque nuevamente **iniciar_medida**, momento en que deben volver a 0 sin importar el estado de **sensor_conectado**.

Se trabajará en modo 2 de interrupciones. Las interrupciones de timer y fin de pulso (en ese orden) serán las de más prioridad. Otros dispositivos ocupan las primeras tres posiciones de la tabla de interrupciones.

Se pide:

- El hardware de entrada/salida necesario para el sensado de temperatura, incluyendo el timer, los controladores de interrupciones, los puertos y su decodificación. Podrá usarse el intervalo 0x40 a 0x7F del espacio de E/S.
- Subrutina ***iniciar_medida*** que debe preservar el contenido de todos los registros.
- Subrutinas de atención a interrupciones de timer y fin de pulso.
- Inicialización de todo el sistema y salto a ***programa_ppal***. La subrutina ***init_otros*** se usará para inicializar todo lo que no está relacionado con la medida de temperatura. Tanto dicha subrutina como el bucle de programa principal están previamente implementados y cargados en ROM. El sistema cuenta con 32K de ROM y 32K de RAM. La tabla de interrupciones debe ubicarse en la dirección 0x9000 de RAM y se debe reservar espacio acorde a la cantidad de dispositivos que interrumpen mediante directivas de reserva de memoria.

Nota: $T_{clk} = 250\text{ns}$ y $250\text{ns} * 2^8 = 64\mu\text{s}$



PROBLEMA 2

Se desea diseñar un sistema basado en el microprocesador Z80 que controle la interfaz de una fuente de tensión variable. El objetivo es poder modificar el valor de tensión determinado por la salida **VOUT**.

El sistema consta de una perilla para modificar la tensión y de un display para visualizar el valor de tensión a cargar. Mientras se esta modificando el valor de tensión este se visualiza en el display, pero solamente se actualiza el valor de **VOUT** al presionar un botón. El botón genera un pulso corto en la entrada **BOTON**. Para manejar el display se cuenta con la salida **VDISP[8]**.

El giro de la perilla incrementa o decrementa la tensión en 1 según gire en sentido horario o antihorario, respectivamente. Para detectar el giro de la perilla esta tiene dos sensores que manejan las señales **S1** y **S2**. Un flanco de subida de señal **S1** significa que hay un giro, pero la determinación del sentido de giro se hace con **S2**. Si **S2=0** en el flanco de **S1** entonces el giro es en sentido horario, en cambio, si **S2=1** en el flanco de **S1** entonces el giro es antihorario. La señal **S1** debe interrumpir al sistema, el cual deberá funcionar en **modo 1**, y la rutina de interrupción se debe encargar de incrementar o decrementar el valor de tensión a mostrar en el display.

Además, el sistema cuenta con una salida **LED** que se enciende mientras se está modificando la tensión deseada, es decir, se enciende con una interrupción de **S1** y se apaga al recibir un pulso en **BOTON**.

El valor de tensión siempre debe mantenerse en el rango **[0, MAX_V[8]]**. El valor **MAX_V[8]** puede ser modificado en cualquier momento al recibir un flanco en la entrada **NuevoMAX**. El valor de **MAX_V[8]** es válido solamente en el flanco de subida de **NuevoMAX**. Si se recibe un nuevo valor de **MAX_V[8]** y este es inferior al valor actual de tensión, se debe ajustar el valor de tensión para que concida con **MAX_V[8]**. Se garantiza que mientras se está modificando la tensión deseada no llegará un flanco en **NuevoMAX** para modificar el valor de **MAX_V**.

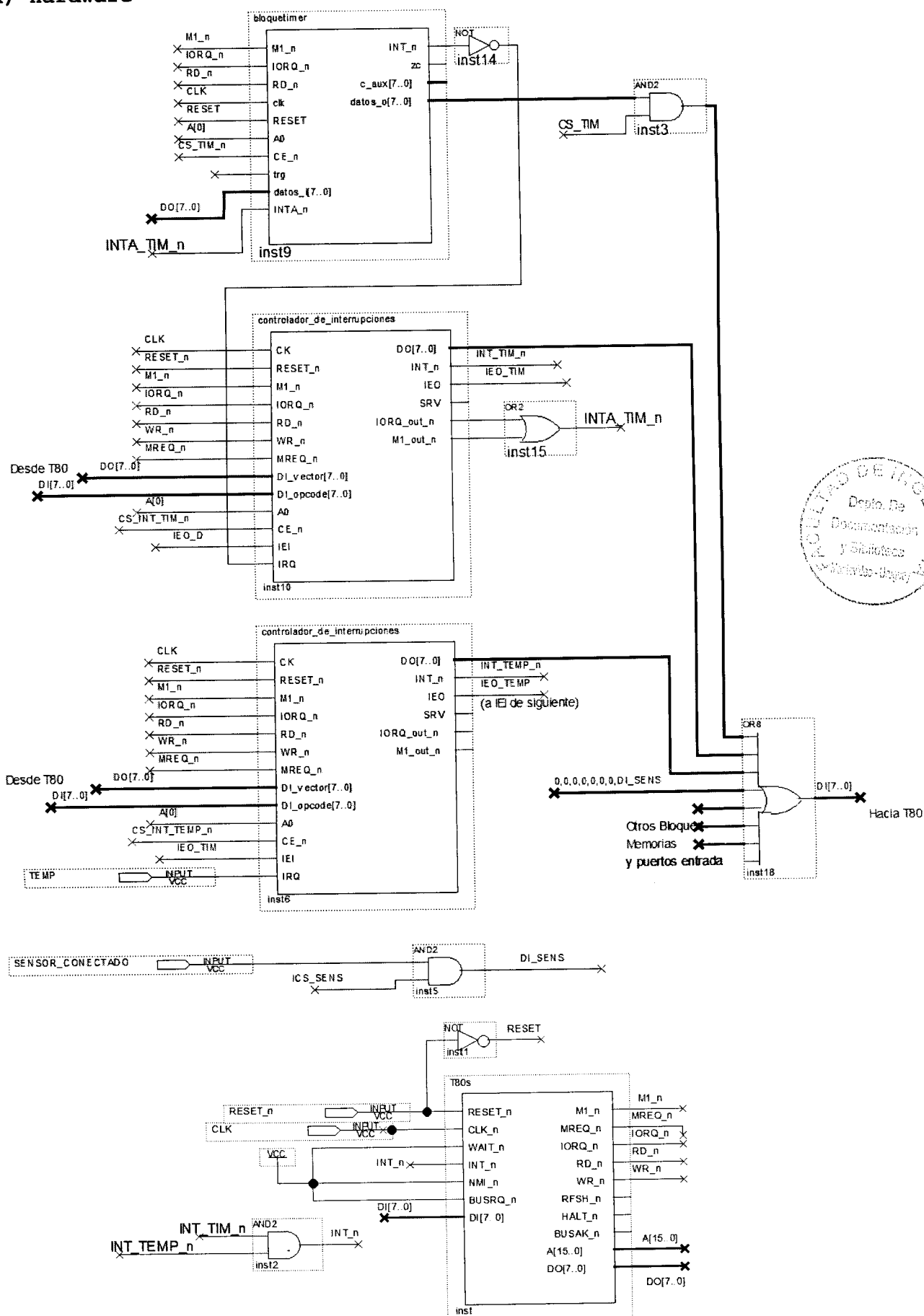
El programa principal deberá quedarse en loop monitoreando el estado de **BOTON** y de **NuevoMAX** para actualizar los valores de **VOUT** y **MAX_V** como se mencionó anteriormente. Además debe llamar a la subrutina **atiendo_otros** que realiza otras tareas. Esas tres tareas deben atenderse en forma cíclica, evitando que la espera por un evento bloquee la atención de los demás.

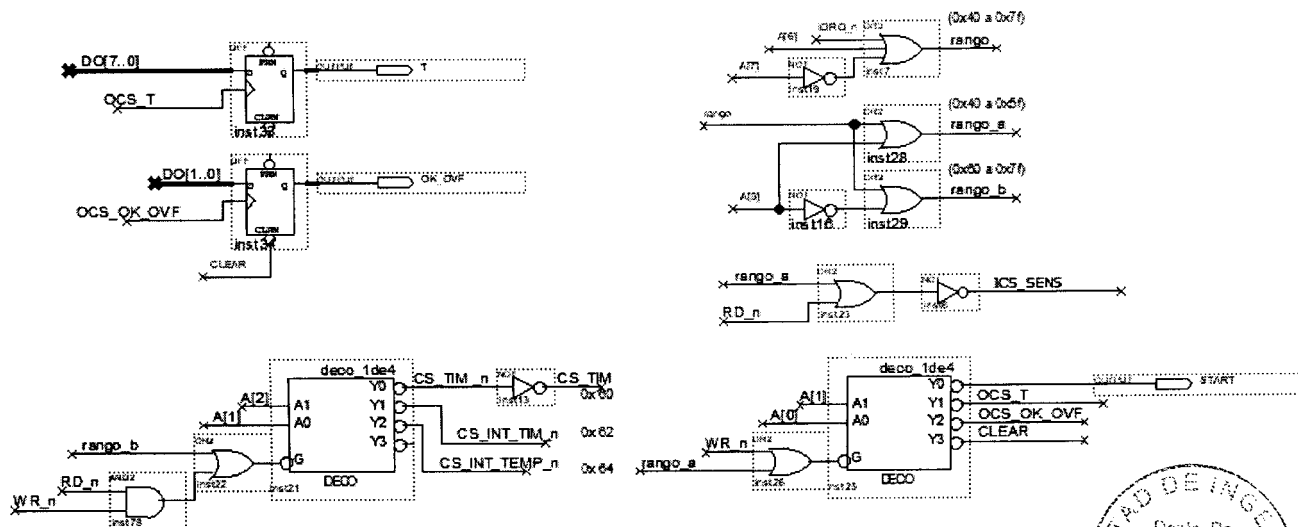
Se pide:

- Todo el hardware incluyendo memorias 32kB de ROM y 32kB de RAM
- Inicialización, programa principal y reserva de memoria.
- Rutina de atención a la interrupción.



a) Hardware

 $\frac{3}{8}$



```
b)
org 0x3000
iniciar_medida:
    ; preservar registros
    ; borrar OK y OVF
    ; borrar flag de TIMEOUT
    ; si sensor_conectado = 1
    ; dar pulso en START
    ; restaurar registros
    ; retornar
push AF
    out (CLEAR), A
    ld A, 0
    ld (TIMEOUT), A
    in A, (SENSOR_CONECTADO)
    bit 0, A
    jp Z, fin_iniciar_medida
    out (START), A
fin_iniciar_medida:
    pop AF
    ret
```

```
c)
org 0x4000
rutint_temp:
    ; habilitar interrupciones
    ; preservar registros
    ; si NO hubo timeout
    ; leer timer
    ; reprogramar timer (EI,
sw_reset, arranque con flanco bajada
trg, pre=8)
    ; temperatura = 60 - valor
leido
    ; escribir temperatura en
puerto T
    ; OK=1
    ; restaurar registros
    ; retono de interrupción
ei
push AF
push BC

ld A, (TIMEOUT)
cp 0
jp NZ, fin_rutint_temp
in A, (TIMER_A1)
ld B, A
ld A, CW_TIMER
;; reprogramo timer para prox.
medida
;; y evito que gener int por
timeout
    out (TIMER_A1), A
    ld A, MAX_TEMP-1
    sub B ; MAX_TEMP-1 - valor timer
    out (T), A
    ld A, OK
    out (OK_OVF), A
```

```
pop BC
pop AF
reti

rutint_timer:
    ; habilitar interrupciones
    ; preservar registros
    ; setear flag de timeout
    ; reprogramar timer (EI,
sw_reset, arranque con flanco bajada
trg, pre=8)
    ; OVF=1
    ; restaurar registros
    ; retono de interrupción
ei
push AF

ld A, 0xFF
ld (TIMEOUT), A
ld A, CW_TIMER
;; reprogramo timer para prox.
medida
;; y evito nuevas int por
timeout
    out (TIMER_A1), A
    ld A, OVF
    out (OK_OVF), A

pop AF
reti

d)
org 0x0000
; SP= 0x0000
; I=0x90
; cargar direcciones rutinas en
tabla de int
; programar VI=6 en CI_1
; borrar int pendientes CI_1
; programar VI=8 en CI_2
; borrar int pendientes CI_2
; cargar constante MAX_TEMP-1 en
timer
; programar timer (EI, sw_reset,
arranque con flanco bajada trg,
pre=8)
; borrar puertos OK y OVF y escribir
0 en puerto T
; llamar a init_otros
; habilitar interrupciones
; saltar a programa_ppal

ld SP, 0x0000
ld A, tabla_int /256
ld I, A
ld HL, (tabla_int + VI_TIMER)
ld (HL), rutint_timer
ld HL, (tabla_int + VI_TEMP)
```



FACULTAD DE INGENIERIA
UNIVERSIDAD DE LA REPUBLICA

```
ld (HL), rutint_temp
ld A, VI_TIMER
out (CI_1_A0), A
out (CI_1_A1), A ; borro
peticiones
ld A, VI_TEMP
out (CI_2_A0), A
out (CI_2_A1), A ; borro
peticiones
ld A, MAX_TEMP-1
out (TIMER_A0), A
ld A, CW_TIMER
out (TIMER_A1), A
out (CLEAR), A
call init_otros
ei
jp programa_ppal
```

```
;puertos
CLEAR equ 0x43
SENSOR_CONECTADO equ 0x40
START equ 0x40
OK_OVF equ 0x42
T equ 0x41
TIMER_A0 equ 0x60
TIMER_A1 equ 0x61
CI_1_A0 equ 0x62
```

INTRODUCCION A LOS MICROPROCESADORES
Diciembre 2022 – pag. 4/6

```
CI_1_A1 equ 0x63
CI_2_A0 equ 0x64
CI_2_A1 equ 0x65
```

```
;constantes
MAX_TEMP equ 60
OK equ 0x01
OVF equ 0x02
```

```
CW_TIMER equ 1011 1000 ; EI, trigger
bajada, reset sw, arranca por
trigger, pre=8
VI_TIMER equ 6
VI_TEMP equ 8
```

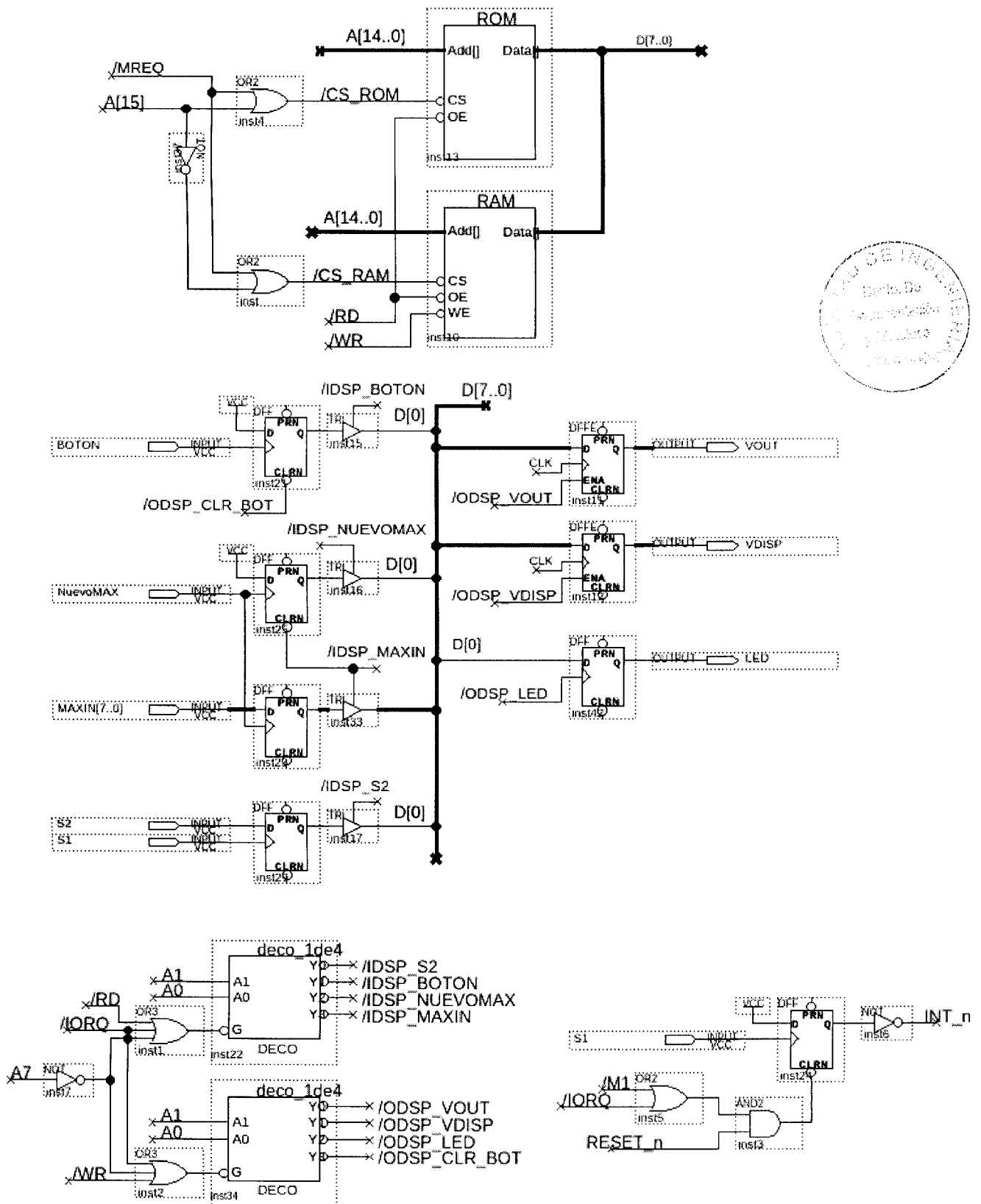
```
org 0x8000
TIMEOUT DB
```

```
org 0x9000
tabla_int:
DW ;
DW ;
DW ;
DW ; timer
DW ; temp
```



SOLUCIÓN PROBLEMA 2

a) Hardware



b) Inicialización, programa principal y reserva de memoria

```

; puertos E/S
S2          equ 0x80
BOTON       equ 0x81
NUEVOMAX    equ 0x82
MAXIN       equ 0x83
VOUT        equ 0x80
VDISP       equ 0x81
LED         equ 0x82
CLR_BOT     equ 0x83
;constantes
MIN_V       equ 0

.data ; en comienzo RAM 0x8000
vaux:       db
max_v:      db

.text ; inicio de ROM 0x0000
im 1
ld sp, 0
ld a, 0
ld (vaux), a
out (VOUT), a
out (LED), a
ei

main:
; forever{
;   atiando otros
;   si Boton{
;     clr FF Boton
;     VOUT = vaux
;   }

;   LED = 0
;   si NuevoMax{
;     max_v = MAXIN
;     si vaux>max_v {
;       vaux = max_v
;     }
;   }

  call atiando_otros
  in a, (BOTON)
  bit 0, a
  jr z, miro_vmax
actualizo_vout:
  out (CLR_BOT), a
  ld a, (vaux)
  out (VOUT), a
  ld a, 0
  out (LED), a
miro_vmax:
  in a, (NUEVOMAX)
  bit 0, a
  jr z, main
actualizo_vmax:
  ld a, (vaux)
  ld b, a
  in a, (MAXIN)
  ld (max_v), a
  cp b          ;MAXIN-vaux
  jp P, main    ;no es mayor, sigo
  ld (vaux), a  ;es mayor, saturo
  jr main

```



c) Rutina de atención a la interrupción -----

```

; isr_modificar_volt:
; preservar registros
; flag = 1
; LED = 1
; si S2=0 entonces
;   vaux = min(MAX_V, vaux + 1)
; else
;   vaux = max(MIN_V, vaux - 1)
; end si
; restaurar

      org 0x0038
isr_modificar_volt:
  push af
  push bc
  ld a, 1
  out (LED), a
  in a, (S2)
  bit 0, a
  jr nz, antihorario
horario:
  ld a, (max_v)

  ld b, a
  ld a, (vaux)
  inc a
  cp b
  jp M, fin_si ;MAX>vaux
satura_max:
  ld a, b      ; saturo en MAX_V
  jr fin_si
antihorario:
  ld a, (vaux)
  dec a
  ld b, MIN_V
  cp b
  jp P, fin_si ;MIN<=vaux
satura_min:
  ld a, b      ; saturo en MIN_V
fin_si:
  ld (vaux), a
  out (VDISP), a
  pop bc
  pop af
  ret

```