Universidad de la República
Facultad de Ingeniería
Instituto de Ingeniería Eléctrica

Introducción a los Microprocesadores

Laboratorio 2022

Práctica 3

Grupo:	***Mie 15:00 - 2***
Cédula de Identidad	Nombre
5.405.795-0	Joel Carabajal
5.569.102-0	Henola Cócaro
4.911.609-4	Carlos Gruss

Índice

1. Mapa entrada/salida

Dirección	Entrada	Salida
0x00	-	-
...	-	-
0x80	SW[7:0]	HEX0[7:0]
0x81	-	HEX1[7:0]
0x82	-	HEX2[7:0]
0x83	-	HEX3[7:0]
0x84	-	LEDS[7:0]
0x85	PSDATA	PSDATA
0x86	PSCLK	PSCLK
0x87	-	CL_PSCLK
0x9X	TIMER	TIMER
0xAX	CONTADOR	CONTADOR
0xBX	C. INT. 1	C. INT. 1
0xCX	C. INT. 2	C. INT. 2
...	-	-
0xFF	-	-

Cuadro 1: Mapa de entrada y salida con puertos diseñados.

2. Pseudocódigo

2.1. Parte b

2.1.1. rutint_counter

```
1 function rutint_counter:
2     Preservar registros
3     int_counter++
4     A ← int_counter
5     call pbcda7seg
6     HEX0 ← C
7     HEX1 ← B
8     Restaurar registros
9     return
```

2.1.2. rutint_counter_test

```
1  Inicializar sistema
2  Habilitar interrupciones
3  while(true)
4      A ← SW
5      LEDS ← A
```

2.2. Parte d simple

2.2.1. rutint_leds

```
1  function rutint_leds:
2      Habilitar interrupciones
3      Preservar registros
4      int_cpl ← not inc_cpl
5      HEX3 ← int_cpl
6      Restaurar registros
7      return
```

2.2.2. rutint_leds_test

```
1  Inicializar sistema
2  Habilitar interrupciones
3  while(true)
4      A ← SW
5      LEDS ← A
```

2.3. Parte d completa

2.3.1. rutint_timer

```
1  function rutint_timer:
2      Habilitar interrupciones
3      Preservar registros
4      if(pause != 0xFF)
5          IX ← reloj
6          call decreloj
7          call despreloj
8      Restaurar registros
9      return
```

2.3.2. rutint_timer_test

```
1  Inicializar sistema
2  Habilitar interrupciones
3  while(true)
4      if (SW[0] == 0)
5          pause ← 0x00
6      else
7          pause ← 0xFF
```

2.4. Parte e

2.4.1. get_ps2_nb (suministrada)

```
1  function get_ps2_nb
2      Preservar registros
3      if (not rx_done == 0)
4          rx_done ← 0
5          A ← rx_byte
6      Restaurar registros
7      return
```

2.4.2. ps2clk_isr

```
1  function ps2clk_isr :
2      Habilitar interrupciones
3      Preservar registros
4      if (num_bit == 0 or num_bit == 9)
5          num_bit++
6      elsif (num_bit == 10)
7          num_bit ← 0
8      else
9          num_bit++
10         rx_byte ← rx_byte | (entradaDATA & 0x01)
11         rx_byte ← rx_byte >> 1 ;Rota circular
12         rx_done ← (rx_done << 1) | 0x01
13     Restaurar registros
14     return
```

2.4.3. get_ps2_nb_test

```
1  Inicializar Stack, hardware y variables
2  Habilitar interrupciones en modo 2
3  while(true)
4      call get_ps2_nb
5      if(Z == 1)
6          datos_ps2[2] ← datos_ps2[1]
7          datos_ps2[1] ← datos_ps2[0]
8          datos_ps2[0] ← A
9
10         call pbcda7seg
11         HEX0 ← C
12         HEX1 ← B
13
14         A ← datos_ps2[1]
15         call pbcda7seg
16         HEX2 ← C
17         HEX3 ← B
18
19         A ← datos_ps2[2]
20         LEDS ← A
21
22         rx_byte ← 0
```

2.5. Parte f

2.5.1. get_tecla_nb

```
1  function get_tecla_nb:
2      call get_ps2_nb
3      if(Z==1)
4          Preservar registros
5          rx_byte ← 0
6          B ← A
7          if(A == 0xF0)
8              llego_f0 ← 1
9              Z ← 0
10         elsif(llego_f0 == 1)
11             llego_f0 ← 0
12             A ← B
13             Z ← 1
14         else
15             Z ← 0
16         Restaurar registros
17     return
```

2.5.2. get_tecla_nb_test

Idéntico a get_ps2_nb_test pero llamando a get_tecla_nb.

2.6. Parte g

2.6.1. temporizador_kbd

```
1  Inicializar sistema
2  pause ← 0xff ;Valor para true
3  state ← STATE_INITIAL
4  while(true)
5      if (state == STATE_INITIAL)
6          call s_initial
7      elsif (state == STATE_COUNTING)
8          call s_counting
9      elsif (state == STATE_FINISHED)
10         call s_finished
11     elsif (state == STATE_PAUSED)
12         call s_paused
13     elsif (state == STATE_EDIT)
14         call s_edit
15
16 function s_initial :
17     call get_tecla_nb
18     if (Z == 1)
19         if (A == SCODE_SPACE)
20             state ← STATE_COUNTING
21             pause ← 0x00
22         elsif (A == SCODE_ENTER)
23             state ← STATE_EDIT
24             time_s ← 0
25             time_c ← 0
26             call despreloj
27             LEDS ← 0xF0
28     return
29
30 function s_counting:
31     call get_tecla_nb
32     if (Z == 1)
33         if (A == SCODE_ENTER)
34             state ← STATE_INITIAL
35             time_s ← DEFAULT_SECS
36             time_c ← 0
37             pause ← 0xFF
38             IX ← &time_s
39             call despreloj
40             LEDS ← 0
41         elsif (A == SCODE_SPACE)
```

```
42                  state ← STATE_PAUSED
43                  pause ← 0x00
44          elsif (time_s == 0 and time_c == 0)
45              state ← STATE_FINISHED
46              pause ← 0xFF
47              LEDS ← 0x0F
48          return
49
50      function s_finished:
51          call get_tecla_nb
52          if (Z == 1)
53              if (A == SCODE_ENTER)
54                  state ← STATE_INITIAL
55                  time_s ← DEFAULT_SECS
56                  time_c ← 0
57                  IX ← &time_s
58                  call despreloj
59                  pause ← 0xFF
60                  LEDS ← 0
61          return
62
63      function s_paused:
64          call get_tecla_nb
65          if (Z == 1)
66              if (A == SCODE_SPACE)
67                  state ← STATE_COUNTING
68                  pause ← 0
69          return
70
71      function s_edit:
72          call get_tecla_nb
73          if (Z == 1)
74              if (A == SCODE_ENTER)
75                  LEDS ← 0
76                  state ← STATE_INITIAL
77              else
78                  call scodeadigito
79                  if (A != 100)
80                      B ← A
81                      A ← time_s
82                      call binapbcd
83                      A ← A & 0x0F
84                      call por_10 ;A←A×10
85                      time_s ← A + B
86                      IX ← &time_s
87                      call despreloj
88          return
```

2.6.2. por_10

```
function por_10:
    Preservar registros
    A ← A×2
    B ← A
    A ← A×4
    A ← A + B
    Restaurar registros
    return
```

3. Código assembler

Se cuentan para los siguientes programas con los siguientes símbolos definidos en el archivo "subrutinas_g.s":

```
HEX0 equ 0x80
HEX1 equ 0x81
HEX2 equ 0x82
HEX3 equ 0x83
SW equ 0x80
BTN equ 0x81
LEDS equ 0x84
; Puertos agregados en Practica 2:
clearFF equ 0x87
salidaDATA equ 0x85
salidaCLK equ 0x86
entradaDATA equ 0x85
entradaCLK equ 0x86
; Puertos agregados en Practica 3:
CSTIMER_CTE equ 0x90
CSTIMER_CONTROL equ 0x91
CSCOUNTER_CTE equ 0xa0
CSCOUNTER_CONTROL equ 0xa1
CSINT1_CLR equ 0xb1
CSINT1_RDSTATE equ 0xb1
CSINT1_VINT equ 0xb0
CSINT2_CLR equ 0xc1
CSINT2_RDSTATE equ 0xc1
CSINT2_VINT equ 0xc0
; Scodes comandos:
SCODE_ENTER equ 0x5a
SCODE_SPACE equ 0x29
; Codificacion de estados:
STATE_INITIAL equ 0x00
STATE_COUNTING equ 0x01
STATE_FINISHED equ 0x02
STATE_PAUSED equ 0x03
```

```
33 STATE_EDIT equ 0x04
34 ; Constantes:
35 DEFAULT_SECS equ 24d
```

También se tiene la siguiente tabla de interrupciones:

```
1 .org 0x200
2 tabla_int:
3     dw rutint_counter
4     dw rutint_leds
5     dw rutint_timer
6     dw ps2clk_isr
```

3.1. Parte b

3.1.1. rutint_counter

```
1  rutint_counter:
2      ei
3      push af
4      push bc
5      ld a, (int_count)
6      inc a
7      ld (int_count), a
8      call pbcda7seg
9      ld a, c
10     out (HEX0), a
11     ld a, b
12     out (HEX1), a
13     pop bc
14     pop af
15     reti
```

3.1.2. rutint_counter_test

```
1  ld sp, 0x0000
2  ; seleccionar interrupciones modo 2
3  im 2
4  ; inicializar tabla de interrupciones
5  ld hl, tabla_int
6  ld a, h
7  ld i, a
8  ; se inicializan en nivel alto los puertos salidaCLK y salidaDATA:
9  ld a, 1d
10 out (salidaCLK), a
11 out (salidaDATA), a
```

```
12 ; incializar variable contador y mostrarlo en displays
13 ld a, 0d
14 ld (int_count), a
15 call pbcda7seg
16 ld a, c
17 out (HEX0), a
18 ld a, b
19 out (HEX1), a
20 ; inicializar controlador de interrupciones 1
21 ld a, 0d
22 out (CSINT1_VINT), a
23 out (CSINT1_CLR), a
24 ; habilitar interrupciones
25 ei
26
27 loop_rutint_counter_test:
28     in a, (SW)
29     out (LEDG), a
30     jr loop_rutint_counter_test
```

3.2. Parte d simple

3.2.1. rutint_leds

```
1 rutint_leds:
2     ei
3     push af
4     ld a, (int_cpl)
5     cpl
6     ld (int_cpl), a
7     out (HEX3), a
8     pop af
9     reti
```

3.2.2. rutint_leds_test

Se omiten las partes de la inicialización que coinciden con rutint_counter_test.

```
1 ld sp, 0x0000
2 ; seleccionar interrupciones modo 2
3 im 2
4 ; inicializar tabla de interrupciones
5 ...
6 ; se inicializan en nivel alto los puertos salidaCLK y salidaDATA:
7 ...
8 ; incializar variable contador y mostrarlo en displays
9 ...
```

```
; apagar HEX2
ld a, 0xff
out (HEX2), a
; inicializar controlador de interrupciones 1:
...
; inicializar controlador de interrupciones 2:
ld a, 2d
out (CSINT2_VINT), a
out (CSINT2_CLR), a
; inicializar contador
ld a, 1d
out (CSCOUNTER_CTE), a
ld a, 11000000B
out (CSCOUNTER_CONTROL), a
; incializar timer
ld a, 75d
out (CSTIMER_CTE), a
ld a, 11001111B
out (CSTIMER_CONTROL), a
; habilitar interrupciones
ei

loop_rutint_leds_test:
       in a, (SW)
       out (LEDG), a
       jr loop_rutint_leds_test
```

3.3. Parte d completa

3.3.1. rutint_timer

```
rutint_timer:
       ei
       push af
       push ix
       ld a, (pause)
       cp 0xff
       jr z, fin_rutint_timer
       ld ix, time_s
       call decreloj
       call despreloj
       fin_rutint_timer:
       pop ix
       pop af
       reti
```

3.3.2. rutint_timer_test

Se omiten las partes de la inicialización que coinciden con las anteriores.

```
ld sp, 0x0000
; seleccionar interrupciones modo 2
im 2
; inicializar tabla de interrupciones
...
; se inicializan en nivel alto los puertos salidaCLK y salidaDATA:
...
; incializar variable contador y mostrarlo en displays
...
; apagar HEX2
...
; inicializar variable reloj
ld a, 90d
ld (reloj), a
ld (reloj+1), a
ld a, 0d
ld (reloj+2), a
; inicializar controlador de interrupciones 1 y 2
ld a, 0d
out (CSINT1_VINT), a
out (CSINT1_CLR), a
ld a, 4d
out (CSINT2_VINT), a
out (CSINT2_CLR), a
; inicializar contador
ld a, 1d
out (CSCOUNTER_CTE), a
ld a, 11000000B
out (CSCOUNTER_CONTROL), a
; incializar timer
ld a, 75d
out (CSTIMER_CTE), a
ld a, 11001111B
out (CSTIMER_CONTROL), a
; habilitar interrupciones
ei

loop_rutint_timer_test:
    in a, (SW)
    bit 0, a
    jr z, no_pausado
    pausado:
        ld a, 0xff
        ld (pausa), a
        jr loop_rutint_timer_test
    no_pausado:
```

```
47          ld a, 0x00
48          ld (pausa), a
49      jr loop_rutint_timer_test
```

3.4. Parte e

3.4.1. get_ps2_nb (suministrada)

```
1   get_ps2_nb:
2       push bc ; preserva registros
3       ld a,(rx_done) ; rx_done tiene en 0 los bits correspondientes
4       cpl            ; a los que no se han leido y 1 los que si de rx_byte
5       or 0
6       jr nz, not_yet
7       ready:
8           ld a, 0
9           ld (rx_done), a ; cuando el dato esta completo: rx_done <- 0
10          ld a, (rx_byte) ; y se carga en a el byte completo
11      not_yet:
12          pop bc ; restaura registros
13          ret
```

3.4.2. ps2clk_isr

```
1   ps2clk_isr:
2       ei
3       push af
4       push bc
5       ld a, (num_bit)
6       cp 0d
7       jr nz, num_bit_not_cero ; if num_bit == 0 llego bit de start
8           inc a
9           ld (num_bit), a
10          pop bc
11          pop af
12          reti
13      num_bit_not_cero:
14      cp 9d
15      jr nz, num_bit_not_nueve ; if num_bit == 9 llego bit de paridad
16          inc a
17          ld (num_bit), a
18          pop bc
19          pop af
20          reti
21      num_bit_not_nueve:
22      cp 10d
```

```
23      jr nz, num_bit_not_diez ; if num_bit == 10 llego bit de stop
24          ld a, 0d
25          ld (num_bit), a
26          pop bc
27          pop af
28          reti
29      num_bit_not_diez: ; else, el dato entrante es un bit de dato
30          inc a
31          ld (num_bit), a ; num_bit += 1
32          ; recibir y enmascarar dato serie:
33          in a, (entradaDATA)
34          and 0x01
35          ld b, a
36          ; rotamos los datos ya recibidos a la derecha:
37          ; (se pone el ultimo dato recibido en el LSB)
38          ld a, (rx_byte)
39          or b
40          rrc a
41          ; el resultado se guarda a memoria:
42          ld (rx_byte), a
43          ; se agrega un 1 a rx_done desde la derecha:
44          ld a, (rx_done)
45          rlc a
46          or 1d
47          ld (rx_done), a
48          ; restaurar registros y retornar:
49          pop bc
50          pop af
51          reti
```

3.4.3. get_ps2_nb_test

```
1  ld sp, 0x0000
2  ; seleccionar interrupciones modo 2
3  im 2
4  ; inicializar tabla de interrupciones
5  ...
6  ; se inicializan en nivel alto los puertos salidaCLK y salidaDATA:
7  ...
8  ; incializar variable contador y mostrarlo en displays
9  ...
10 ; apagar HEX2
11 ...
12 ; inicializar variable reloj
13 ...
14 ; incializar variable datos_ps2
15 ld a, 0d
16 ld (datos_ps2), a
```

```
17 ld (datos_ps2+1), a
18 ld (datos_ps2+2), a
19 ld ix, datos_ps2
20 ; inicializar variables de recepcion
21 ld a, 0d
22 ld (rx_done), a
23 ld (rx_byte), a
24 ld (num_bit), a
25 ; inicializar controlador de interrupciones 1 y 2
26 ld a, 6d
27 out (CSINT1_VINT), a
28 out (CSINT1_CLR), a
29 ld a, 4d
30 out (CSINT2_VINT), a
31 out (CSINT2_CLR), a
32 ; inicializar contador
33 ld a, 1d
34 out (CSCOUNTER_CTE), a
35 ld a, 01000000B ; se desactiva esta interrupcion
36 out (CSCOUNTER_CONTROL), a
37 ; incializar timer
38 ld a, 75d
39 out (CSTIMER_CTE), a
40 ld a, 11001111B
41 out (CSTIMER_CONTROL), a
42 ; habilitar interrupciones
43 ei
44
45 loop_get_ps2_test:
46    call get_ps2_nb
47    jr nz, loop_get_ps2_test
48
49        ; rotacion circular de datos:
50        ; (ix+2) <-- (ix+1)
51        ld e, (ix+1)
52        ld (ix+2), e
53        ; (ix+1) <-- (ix)
54        ld e, (ix)
55        ld (ix+1), e
56        ; (ix) <-- a (ultimo dato)
57        ld (ix), a
58
59        ; mostrar ultimo dato
60        call pbcda7seg
61        ld a, c
62        out (HEX0), a
63        ld a, b
64        out (HEX1), a
65        ; mostrar penultimo dato
66        ld a, (ix+1)
```

```
67          call pbcda7seg
68          ld a, c
69          out (HEX2), a
70          ld a, b
71          out (HEX3), a
72          ; mostramos antepenultimo dato
73          ld a, (ix+2)
74          out (LEDG), a
75
76          ld a, 0d
77          ld (rx_byte), a
78
79      jr loop_get_ps2_test
```

3.5. Parte f

3.5.1. get_tecla_nb

```
1   get_tecla_nb:
2       call get_ps2_nb
3       jr z, llego_dato
4       ret
5       llego_dato:
6           push bc
7           ld b, a
8           ld a, 0d
9           ld (rx_byte), a
10          ld a, b
11          cp 0xf0
12          jr nz, not_f0
13                  ld a, 1d
14                  ld (llego_f0), a
15                  or 1d
16                  pop bc
17                  ret
18          not_f0:
19                  ld a, (llego_f0)
20                  cp 1d
21                  jr nz, tecla_presionada
22                          ld a, 0d
23                          ld (llego_f0), a
24                          ld a, b
25                          cp a
26                          pop bc
27                          ret
28                  tecla_presionada:
29                          or 1d
30                          pop bc
```

```
31                          ret
```

3.5.2. get_tecla_nb_test

```
1  ld sp, 0x0000
2  ; seleccionar interrupciones modo 2
3  im 2
4  ; inicializar tabla de interrupciones
5  ...
6  ; se inicializan en nivel alto los puertos salidaCLK y salidaDATA:
7  ...
8  ; incializar variable contador y mostrarlo en displays
9  ...
10 ; apagar HEX2
11 ...
12 ; inicializar variable reloj
13 ...
14 ; incializar variable datos_ps2
15 ...
16 ; inicializar variables de recepcion
17 ...
18 ; inicializar otras variables
19 ld a, 0d
20 ld (llego_f0), a
21 ; inicializar controlador de interrupciones 1 y 2
22 ld a, 6d
23 out (CSINT1_VINT), a
24 out (CSINT1_CLR), a
25 ld a, 4d
26 out (CSINT2_VINT), a
27 out (CSINT2_CLR), a
28 ; inicializar contador
29 ld a, 1d
30 out (CSCOUNTER_CTE), a
31 ld a, 01000000B ;se desactiva esta interrupcion
32 out (CSCOUNTER_CONTROL), a
33 ; incializar timer
34 ld a, 75d
35 out (CSTIMER_CTE), a
36 ld a, 11001111B
37 out (CSTIMER_CONTROL), a
38 ; habilitar interrupciones
39 ei
40
41
42 loop_get_tecla_test:
43    call get_tecla_nb
44    jr nz, loop_get_tecla_test
```

```
45
46          ; rotacion circular de datos:
47          ; (ix+2) <-- (ix+1)
48          ld e, (ix+1)
49          ld (ix+2), e
50          ; (ix+1) <-- (ix)
51          ld e, (ix)
52          ld (ix+1), e
53          ; (ix) <-- a (ultimo dato)
54          ld (ix), a
55
56          ; mostrar ultimo dato
57          call pbcda7seg
58          ld a, c
59          out (HEX0), a
60          ld a, b
61          out (HEX1), a
62          ; mostrar penultimo dato
63          ld a, (ix+1)
64          call pbcda7seg
65          ld a, c
66          out (HEX2), a
67          ld a, b
68          out (HEX3), a
69          ; mostramos antepenultimo dato
70          ld a, (ix+2)
71          out (LEDG), a
72
73      jr loop_get_tecla_test
```

3.6. Parte g

3.6.1. temporizador_kbd

```
1  ;;; -------------------------------------------------------------
2  ;;; 1) INIT GENERAL: stack, modo interrupciones, tabla interrupciones
3  ;;; -------------------------------------------------------------
4  ld sp, 0x0000
5  im 2
6  ; tabla
7  ld hl, tabla_int
8  ld a, h
9  ld i, a
10 ;;; -------------------------------------------------------------
11 ;;; 2) INIT PERIFIERICOS: timer, counter, puertos PSDAT_O y PSCLK_O...
12 ;;; -------------------------------------------------------------
13 ; se inicializan en nivel alto los puertos PSDAT_O y PSCLK_O:
14 ld a, 1d
```

```
15 out (salidaCLK), a
16 out (salidaDATA), a
17 ; inicializar contador:
18 ld a, 1d
19 out (CSCOUNTER_CTE), a
20 ld a, 11000000B
21 out (CSCOUNTER_CONTROL), a
22 ; incializar timer:
23 ld a, 75d
24 out (CSTIMER_CTE), a
25 ld a, 11001111B
26 out (CSTIMER_CONTROL), a
27 ;;; -------------------------------------------------------------
28 ;;; 3) INIT CONTROLADORES INTERRUPCION: controlador ps2clk, controlador
29 ;;;   timer/counter.
30 ;;; -------------------------------------------------------------
31 ; inicializar controlador de interrupciones 1:
32 ld a, 6d
33 out (CSINT1_VINT), a
34 out (CSINT1_CLR), a
35 ; inicializar controlador de interrupciones 2:
36 ld a, 4d
37 out (CSINT2_VINT), a
38 out (CSINT2_CLR), a
39 ;;; -------------------------------------------------------------
40 ;;; 4) INIT DE RUTINAS AUXILIARES:
41 ;;;   Ej:, variables necesarias para get_ps2_nb, get_packet_nb, etc.
42 ;;;   incializar variable datos_ps2:
43 ;;; -------------------------------------------------------------
44 ld a, 0d
45 ld (datos_ps2), a
46 ld (datos_ps2+1), a
47 ld (datos_ps2+2), a
48 ld ix, datos_ps2
49 ; inicializar variables de recepcion ps2:
50 ld a, 0d
51 ld (rx_done), a
52 ld (rx_byte), a
53 ld (num_bit), a
54 ld (llego_f0), a
55 ; inicializar reloj:
56 ld a, 24d
57 ld (time_s), a
58 ld a, 0d
59 ld (time_c), a
60 ld ix, time_s
61 call despreloj
62 ;;; -------------------------------------------------------------
63 ;;; RUTINA PRINCIPAL COMIENZA AQUI
64 ei
```

```
65
66
67
68  main:
69      ld a, 0xff
70      ld (pause), a
71      ld a, STATE_INITIAL
72      ld (state), a
73
74  while:
75      ld a, (state)
76
77  check_1:
78      cp STATE_INITIAL
79      jr nz, check_2
80      call s_initial
81      jr while_end  ; break
82
83  check_2:
84      cp STATE_COUNTING
85      jr nz, check_3
86      call s_counting
87      jr while_end  ; break
88
89  check_3:
90      cp STATE_FINISHED
91      jr nz, check_4
92      call s_finished
93      jr while_end  ; break
94
95  check_4:
96      cp STATE_PAUSED
97      jr nz, check_5
98      call s_paused
99      jr while_end  ; break
100
101 check_5:
102     cp STATE_EDIT
103     call s_edit
104
105 while_end:
106     jr while
107
108 ;;; -------------------------------------------------------------
109 ;;; FIN RUTINA PRINCIPAL
110 ;;; -------------------------------------------------------------
111
112
113 ;;; --------------------------------------
114 ;;; En estado "inicial"
```

```
115 ;;; ---------------------------------------
116 s_initial:
117     call get_tecla_nb
118     jr nz, s_i_done ; no se ha recibido comando desde ps2
119 s_i_got_cmd:
120     cp SCODE_SPACE
121     jr z, s_i_got_pp ; el boton PP no estaba apretado, lo ignoramos
122     cp SCODE_ENTER
123     jr z, s_i_got_restart
124     jr s_i_done
125 s_i_got_pp:
126     ;; recibimos boton PP
127     ld a, STATE_COUNTING
128     ld (state), a ; cambiar a estado "contando"
129     ld a, 0
130     ld (pause), a ; deshabilitar pausa
131     jr s_i_done
132 s_i_got_restart:
133     ld a, STATE_EDIT
134     ld (state), a
135     ld a, 0
136     ld (time_s), a
137     ld (time_c), a
138     call despreloj
139     ld a, 0xf0
140     out (LEDS), a
141 s_i_done:
142     ret
143
144 ;;; ---------------------------------------
145 ;;; En estado "contando"
146 ;;; ---------------------------------------
147 s_counting:
148     call get_tecla_nb
149     jr nz, s_c_check_time ; no se ha recibido comando desde ps2
150 s_c_got_cmd:
151     cp SCODE_ENTER
152     jr z, s_c_got_restart ; no fue restart, continuo con comparacion de
          tiempo
153     cp SCODE_SPACE
154     jr z, s_c_got_pause
155     jr s_c_check_time
156
157 s_c_got_restart:
158     ;; recibimos boton restart
159     ld a, STATE_INITIAL
160     ld (state), a  ; estado = "inicial"
161     ld a, DEFAULT_SECS ; cuenta = 24.00
162     ld (time_s), a
163     ld a, 0
```

```
164     ld (time_c), a
165     ld a, 0xff
166     ld (pause), a ; pause = true
167     ld ix, time_s
168     call despreloj
169     ld a, 0 ; apagar leds
170     out (LEDS), a
171     jr s_c_done
172
173 s_c_got_pause:
174     ld a, STATE_PAUSED
175     ld (state), a
176     ld a, 0xff
177     ld (pause), a
178     jr s_c_done
179
180 s_c_check_time:
181     ld a, (time_s)
182     ld b, a
183     ld a, (time_c)
184     or b
185     jr nz, s_c_done ; no ha llegado a 00:00
186
187 s_c_timeout:  ; llego a cero
188     ld a, STATE_FINISHED ; estado = "termine"
189     ld (state), a
190     ld a, 0xff ; pause = true
191     ld (pause), a
192     ld a, 0x0f
193     out (LEDS), a ; encender leds
194
195 s_c_done:
196     ret
197
198 ;;; ---------------------------------------
199 ;;; En estado "termine"
200 ;;; ---------------------------------------
201 s_finished:
202     call get_tecla_nb
203     jr nz, s_f_done ; no se ha recibido comando desde ps2
204 s_f_got_cmd:
205     cp SCODE_ENTER
206     jr nz, s_f_done ; no fue restart, sigo en este estado
207 s_f_got_restart:
208     ;; recibimos boton restart
209     ld a, STATE_INITIAL
210     ld (state), a  ; estado = "inicial"
211     ld a, DEFAULT_SECS
212     ld (time_s), a ; cuenta = 24.00
213     ld a, 0
```

```
214     ld (time_c), a
215     ld a, 0xff
216     ld ix, time_s
217     call despreloj
218     ld (pause), a ; pause = true
219     ld a,0
220     out (LEDS), a
221 s_f_done:
222     ret
223 
224 ;;; ---------------------------------------
225 ;;; En estado "pausado"
226 ;;; ---------------------------------------
227 s_paused:
228     call get_tecla_nb
229     jr nz, s_p_done
230 s_p_got_cmd:
231     cp SCODE_SPACE
232     jr nz, s_p_done
233 s_p_got_pp:
234     ;; recibimos boton PP
235     ld a, STATE_COUNTING
236     ld (state), a ; cambiar a estado "contando"
237     ld a, 0
238     ld (pause), a ; deshabilitar pausa
239 s_p_done:
240     ret
241 
242 ;;; ---------------------------------------
243 ;;; En estado "editar"
244 ;;; ---------------------------------------
245 s_edit:
246     call get_tecla_nb
247     jr nz, s_e_done
248 s_e_got_cmd:
249     cp SCODE_ENTER
250     jr nz, s_e_not_enter
251     ; se apreto restart:
252     ld a, 0
253     out (LEDS), a
254     ld a, STATE_INITIAL
255     ld (state), a
256     jr s_e_done
257 s_e_not_enter:
258     call scodeadigito
259     cp 100
260     jr z, s_e_done ; se presiono tecla invalida
261     ld b, a ; cargamos en b el digito nuevo
262     ld a, (time_s)
263     call binapbcd
```

```
264     and 0x0f  ; en a quedan las unidades de time_s
265     call por_10
266     add a, b
267     ld (time_s), a
268     ld ix, time_s
269     call despreloj
270 s_e_done:
271     ret
272 
273 ;;; ------------------------------------------------------------
274 ;;; reserva e inicializacion de tablas
275 ;;; ------------------------------------------------------------
276 
277 .org 0x200
278 tab_h7s:              ; _gfedcba
279     db 11000000B ; 0
280     db 11111001B ; 1
281     db 10100100B ; 2
282     db 10110000B ; 3
283     db 10011001B ; 4
284     db 10010010B ; 5
285     db 10000010B ; 6
286     db 11111000B ; 7
287     db 10000000B ; 8
288     db 10011000B ; 9
289     db 10001000B ; a
290     db 10000011B ; b
291     db 10100111B ; c
292     db 10100001B ; d
293     db 10000110B ; e
294     db 10001110B ; F
295 
296 tab_scodeadigito:
297     db 0x70 ; 0
298     db 0x69 ; 1
299     db 0x72 ; 2
300     db 0x7a ; 3
301     db 0x6b ; 4
302     db 0x73 ; 5
303     db 0x74 ; 6
304     db 0x6c ; 7
305     db 0x75 ; 8
306     db 0x7d ; 9
307 
308 .org 0x300
309 tabla_int:
310     dw rutint_counter
311     dw rutint_leds
312     dw rutint_timer
313     dw ps2clk_isr
```

```
314
315 ;;; -------------------------------------------------------------
316 ;;; reserva de memoria para variables de la aplicacion
317 ;;; -------------------------------------------------------------
318 .data
319 state:    db 0 ; estado actual (ver constantes STATE_*)
320 pause:    db 0 ; 0xFF si pausado
321 time_s:   db 0 ; segundos restantes
322 time_c:   db 0 ; centesimas restantes
323 time_dp:      db 0
324 ;;;
325 ;;; otras variables usadas para el resto de sus rutinas (si necesario)
326 datos_ps2:
327        db 0d
328        db 0d
329        db 0d
330 int_count:    db 0d
331 int_cpl:      db 0d
332 num_bit:      db 0d
333 rx_byte:      db 0d
334 rx_done:      db 0d
335 llego_f0:     db 0d
```

3.6.2. por_10

```
1  por_10:
2     push bc
3     sla a
4     ld b, a
5     sla a
6     sla a
7     add a, b
8     fin_por_10:
9        pop bc
10       ret
```

4. Cálculo de constantes para bloques timer y contador

Para que los bloque T/C combinados interrumpan cada una décima de segundo (100ms), se configura el timer para que el tiempo que cuente sea 50ms, mientras que el contador se configura con una constante de 1. Cada vez que el timer termina de contar, se decrementa una cuenta en el contador. De esta manera la interrupción se genera luego de que el timer llega a cero dos veces (100ms). El cálculo de los parámetros para el timer fue el siguiente:

$$2^{pre} * (cte + 1) * \left(\frac{1}{f_{clk}}\right) = \frac{1}{20}s = 50ms \quad (1)$$

siendo $f_{clk} = 5MHz$, $pre \leq 15$ y $cte \leq 255$.

Eligiendo $pre = 15$, la constante que minimiza el error es $cte = 75$, obteniendo de esta manera un tiempo inicial para el timer de $49,8ms$, resultando un error menor a $1ms$ $(0,2ms)$.

5. Hardware de controlador de interrupciones agregado

A continuación se presentan los esquemáticos para el controlador de interrupciones agregado con su circuito de decodificación. Además, la señal CS_INT1 en la figura 2 corresponde a la habilitación del controlador de interrupciones original suministrado.

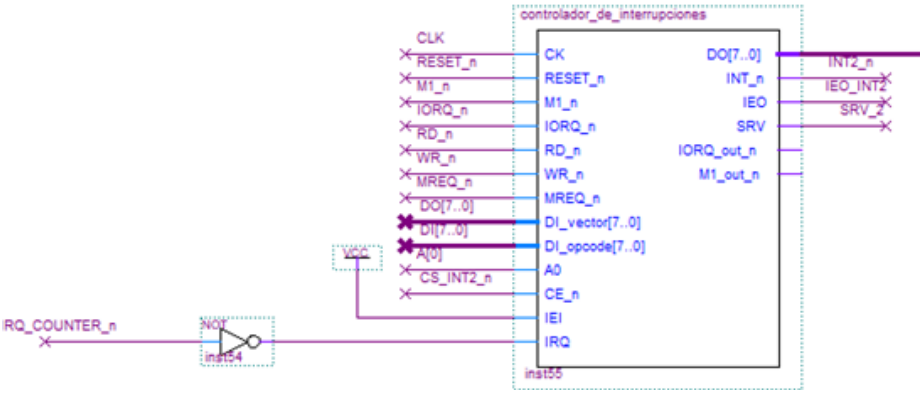

Figura 1: Hardware de controlador de interrupciones agregado

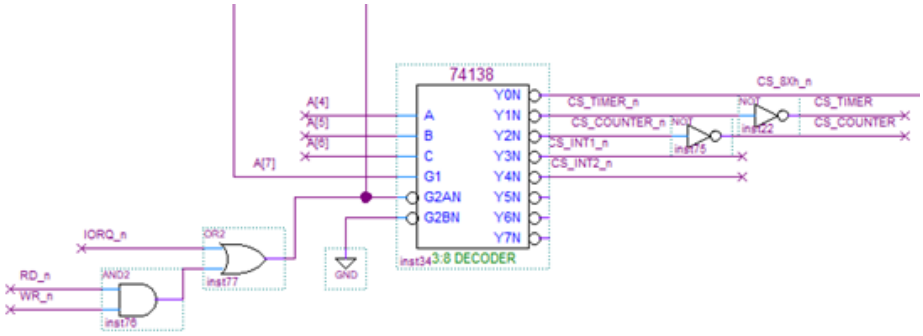

Figura 2: Decodificacion de controlador de interrupciones agregado

6. Diagrama de tiempos

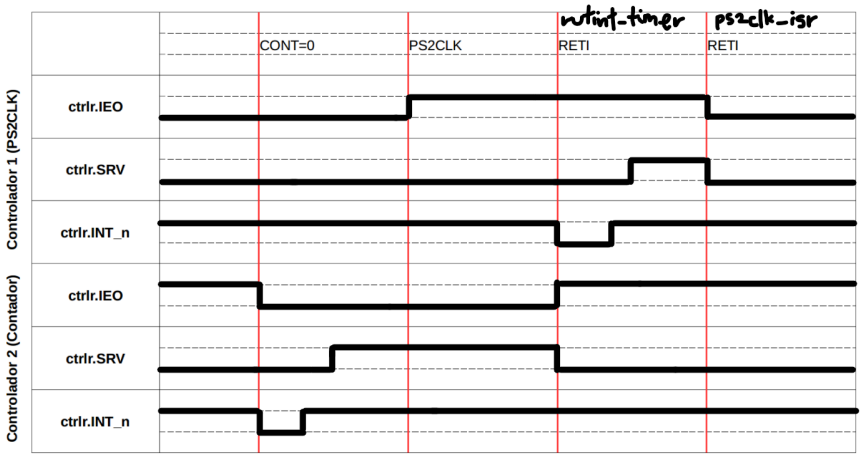

Figura 3: Diagrama planteado en parte h. Los retardos no están a escala.