

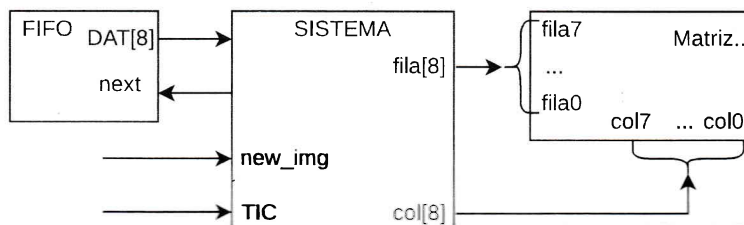
- Mínimo para aprobar: un problema
- Nombre y CI en cada hoja
- Numere las hojas, indique el total en la primera

- 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 dispone de matrices de leds de 8 filas por 8 columnas. El funcionamiento de la matriz es el siguiente: para encender el led en la posición **i, j** de la matriz, se debe poner simultáneamente un 0 lógico en las entradas **fila[i]** y **col[j]**.



La ilusión de que todos los leds permanecen encendidos o apagados en forma continua se logra seleccionando una columna por vez y poniendo los datos de las filas correspondientes a esa columna. La columna seleccionada se cambia cada **T_col** ms recorriendo toda la matriz. Haciendo esto cíclicamente la secuencia se repite cada $8 \times T_col$ ms y cada columna está seleccionada $1/8$ del tiempo.

Se quiere diseñar un controlador para la matriz de leds basado en un microprocesador Z80. La imagen a desplegar inicialmente se carga durante la inicialización del sistema mediante la subrutina **ini_img** en una tabla de 8 bytes en RAM a partir de la dirección **BASE**, en la cual se almacenará en forma consecutiva el valor de cada columna (la columna 0 en la dirección **BASE**, la columna 1 en la dirección **BASE+1**, etc.).

Se debe poder cambiar la imagen desplegada. Para eso un operador externo debe cargar el valor de las 8 columnas de la nueva imagen en un dispositivo FIFO, comenzando por la columna 0, y avisar al sistema dando un pulso a 0 en la entrada **new_img**. El FIFO pone en su salida **DAT** el dato más antiguo almacenado. Con un flanco de subida en su entrada **next**, el FIFO descarta el dato más antiguo, y en consecuencia pasa a presentarse el siguiente dato en la salida **DAT**.

Se dispone de una señal **TIC** que es una onda cuadrada de período **T_col** ms.

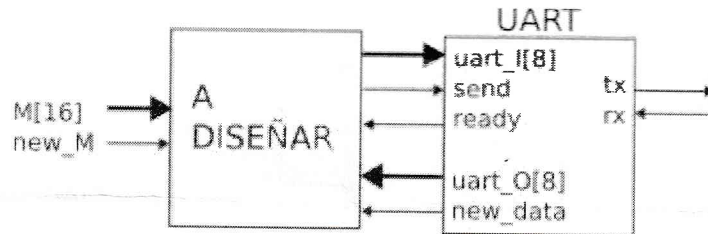
Se debe diseñar completamente el controlador de la matriz. Para eso se pide:

- Agregar todo el Hardware, incluyendo memorias (32KB de RAM y 32KB de ROM), necesario para manejar la matriz de leds y generar interrupciones cada **T_col** ms a partir de la señal **TIC**, trabajando en modo 1 de interrupciones.
- Escribir la rutina de atención a la interrupción que actualice el estado de los leds a partir del contenido de la tabla. Si se desea puede utilizarse la subrutina **BIN2ONEHOT** que recibe un byte en el acumulador y devuelve otro con un cero en el bit indicado por el valor de los 3 bits menos significativos.
- Programa principal que quede en loop infinito esperando la indicación de **new_img** para actualizar la tabla en RAM que almacena la imagen a desplegar. Cada vez que llegue un pulso en **new_img** debe leer 8 datos del dispositivo FIFO (utilizando correctamente los puertos **DAT[8]** y **next**) y almacenarlos en la tabla.
- Directivas de reserva de memoria y código de inicialización para que todo lo anterior funcione correctamente a partir de un **RESET**. Utilizar la subrutina **ini_img** para cargar la imagen inicial en la tabla en RAM luego de un reset.



PROBLEMA 2

Se debe diseñar un sistema basado en T80 equipado con un sensor y un bloque transmisor receptor (UART). El sistema recibe medidas ($M[16]$) cada vez que se da un pulso a 1 en **new_M**. Por otro lado, cada vez que desde la UART recibe el comando **0xA0** debe transmitir la medida más reciente recibida del sensor.



Las medidas en $M[16]$ son de 16 bits y están válidas solo durante la subida de **new_M** que es de corta duración. Al momento de transmitir una medida por la UART, primero se envía el byte bajo y luego el alto.

Cada vez que la UART recibe un byte por la entrada serie rx, la hace disponible en la salida **uart_O[8]** y da un pulso a 1 de corta duración en **new_data**. El dato en **uart_O[8]** se mantiene hasta que se reciba uno nuevo. Para transmitir un dato por la UART, éste se debe poner en la entrada **uart_I[8]** y dar un flanco de subida en **send**. Inmediatamente la señal **ready** baja a 0 y vuelve a subir cuando termina la transmisión y está lista para recibir una nueva palabra.

La interacción del T80 con las señales de control de la UART (ser notificado que se recibió un byte o que está listo para enviar el 2do byte de medida) debe hacerse mediante Controladores de Interrupciones (uno para cada función).

El software debe estructurarse como se describe a continuación:

- Un programa principal que luego de inicializar todo el sistema queda en loop leyendo una medida cada vez que llega el pulso en **new_M**.
- Una rutina de atención a las interrupciones generadas por **new_data** que, en caso de recibir el comando **0xA0**, se encarga de comenzar el envío de la última medida recibida; si el byte recibido es otro se descarta y no se hace nada.
- La rutina que atiende las interrupciones de **ready** envía el segundo byte de medida.

Notar que entre la transmisión del byte alto y bajo de la medida puede llegar un aviso en **new_M**. Deben tomarse las precauciones para no perder el nuevo dato y que el transmitido por la UART sea consistente. Por otro lado se asegura que no se recibirá un nuevo comando (por **uart_O[8]** y **new_data**) mientras se esté completando el envío de una medida por la UART. También se podrá suponer a los efectos del problema que no se recibirán comandos desde la UART entre las lecturas de cada uno de los dos bytes de una nueva medida.

Se pide:

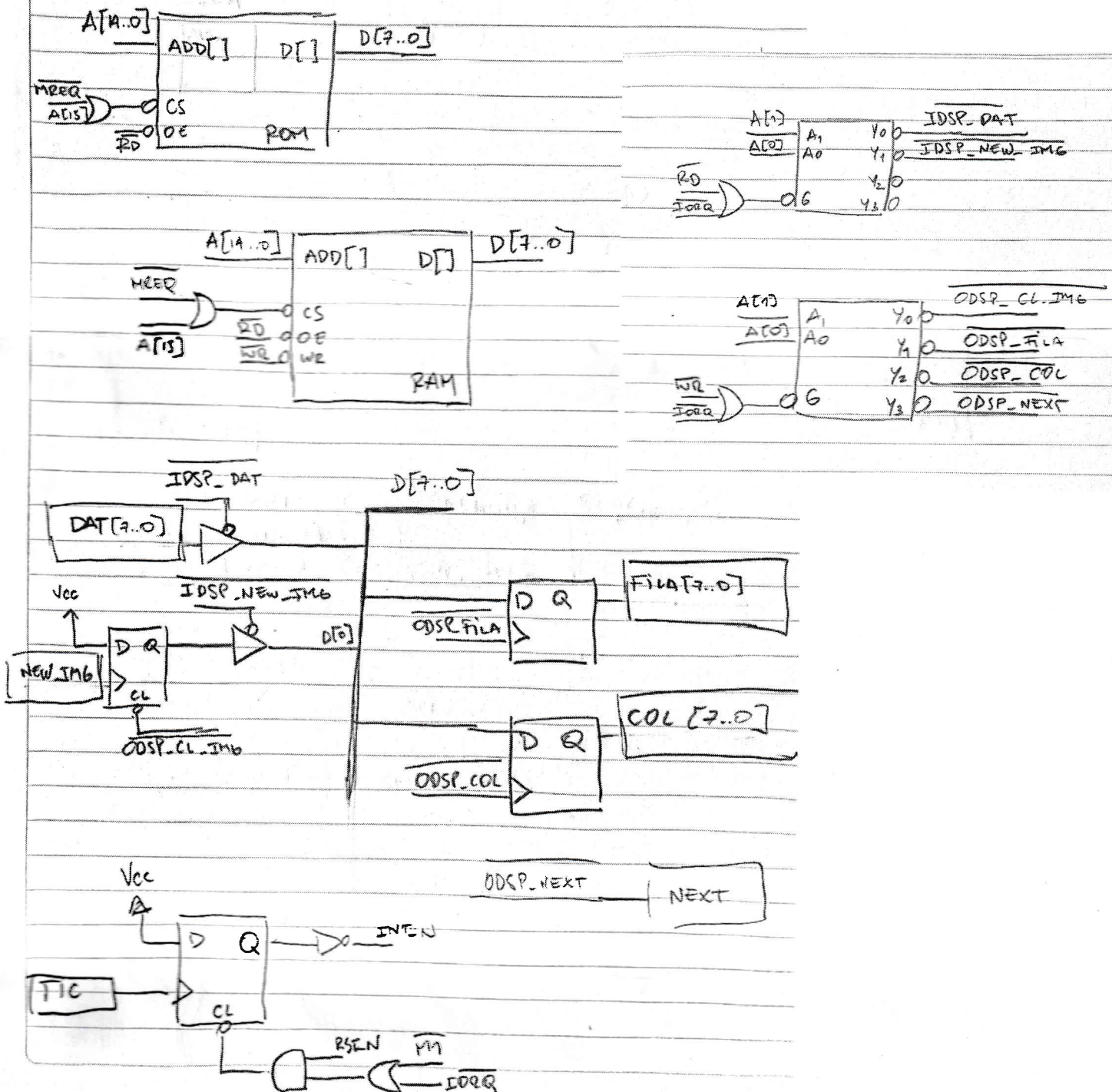
- a) HW completo del sistema (salvo memoria de 32K de ROM y 32 K de RAM que está implementada) incluyendo Controladores de interrupciones y hardware adicional para conectarlos a la UART e implementar puertos de entrada y salida.
- b) Rutinas de atención a las interrupciones.
- c) Declaración de variables y reserva de espacio en memoria. Indicar direcciones de variables, tabla de interrupciones y código.
- d) Programa principal que inicializa todo el sistema luego de un reset y loop principal. Si llega la solicitud de enviar una medida y aún no se ha leído nada, se debe enviar 0xFFFF



2/2

PROBLEMA 1

a) Hardware



FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

b) Rutinas interrupción *****
/*

PUSH

EI

A = COL_B

BIN2ONEHOT

COL = A

FILA = (BASE+COL_B)

OUT (COL)

OUT (FILA)

IF (COL_B = 7)

COL_B=0

ELSE

INC(COL_B)

END

POP

RET

*/

org 0x0038

isr_print:

PUSH AF

PUSH BC

PUSH HL

LD A, (COL_B)

LD HL, BASE

ADD A, L

;; no hay desborde con BASE=0x8000

LD L, A

LD B, (HL)

LD A, (COL_B)

CALL BIN2ONEHOT

OUT (COL), A

LD A, B

OUT (FILA), A

CP 7

JP NZ, INC

LD A, 0

JP SIGO

INC:

INC A

SIGO:

LD (COL_B), A

POP HL

POP BC

POP AF

EI

RET

c) Programa Principal *****

/*

WHILE !(NEW_IMG){}

INTRODUCCIÓN A LOS MICROPROCESADORES
FEBRERO 2024

CLEAR NEW_IMG

POINT = 0

WHILE POINT<=7{

READ (DAT)

LD (BASE+POINT), DAT

INC POINT

OUT (NEXT)}

*/

org 0x0100

prg_ppal:

IN A, (NEW_IMG)

AND 1

JP Z, prog_ppal

OUT (CLR_NEW_IMG), A

LD HL, BASE

LD B, 8

LEER_FIFO:

IN A, (DAT)

LD (HL), A

INC HL

OUT (NEXT), A

DJNZ LEER_FIFO

JP prg_ppal

d) inicializacion *****

DAT EQU 0x00

NEW_IMG EQU 0x01

CLR_NEW_IMG EQU 0x00

FILA EQU 0x01

COL EQU 0x02

NEXT EQU 0x03

ORG 0x0000

LD sp, 0

IM 1

CALL ini_img

LD a, 0

LD (COL_B), a

EI

JP prog_ppal

ORG 0x8000

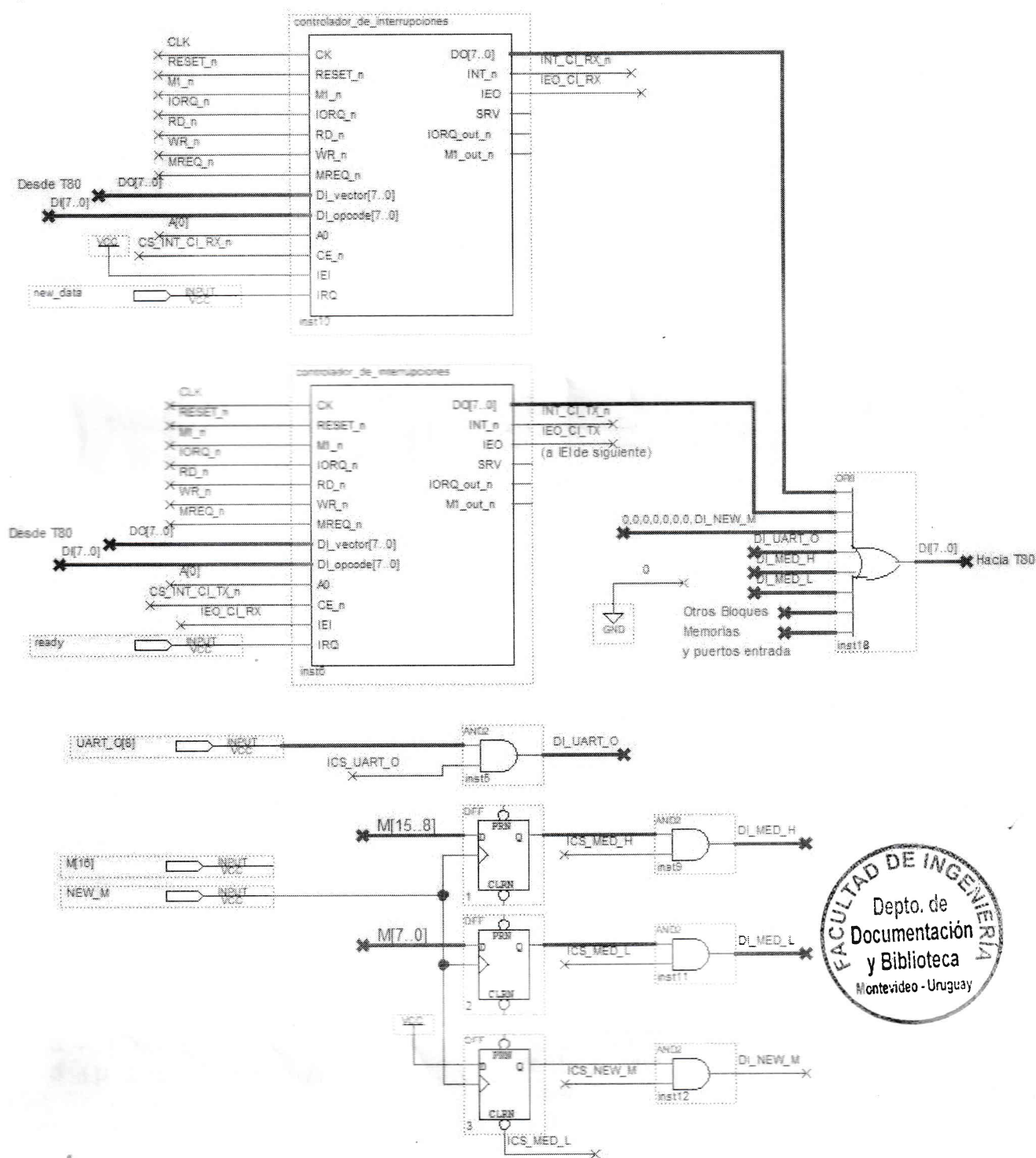
BASE: ds 8

COL_B: db 0



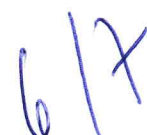
4/7

a) Hardware



Observar que el FF de Handshake se borra con la lectura del byte bajo de la medida.

3/7



d) Programa principal

```
org 0x0000
ld SP, 0
ld HL, tabla_int
ld A, H
ld I, A

ld A, VI_RX
out (VI_RX), A
out (VI_RX+1), A

ld A, VI_TX
out (VI_TX), A
out (VI_TX+1), A

ld HL, 0xFFFF
ld (ultima_medida), HL

ld A, 0xFF
```

```
ld (primer_byte), A
; (no necesario lo hace ISR_RX)
```

```
ei
jp loop
```

```
loop:
in A, (new_M)
and 0x01
jp Z, loop
;; leo byte alto
in A, (medida_H)
ld H, A
;; leo byte bajo y borro FF
in A, (medida_L)
ld L, A
ld (ultima_medida), HL
jp loop
```



7/7