Armado de una computadora basada en el 8088 de Intel

Javier Rizzo Aguirre
Ingeniería en Cibernética Electrónica
CETYS Universidad
Mexicali, México
javierrizzoa@gmail.com

Álvaro S. Moreno Partida
Ingeniería en Cibernética Electrónica
CETYS Universidad
Mexicali, México
alvarosalvador.moreno@cetys.edu.mx

***Resumen*—El siguiente proyecto se realizó como parte de la materia de Diseño con Microprocesadores, del programa de Ingeniería en Cibernética Electrónica de CETYS Universidad, con el fin de conocer y comprender las características de un microprocesador, el proceso de selección de un microprocesador, la circuitería de apoyo que es necesaria para que funcione y las aplicaciones del mapa de memoria y dispositivos.**

Palabras clave—microprocesador; memoria; dispositivos; direcciones.

I. Introducción

El diseño utilizando microprocesadores es muy relevante en la actualidad, pues permite realizar circuitos complejos de manera rápida y sencilla, ya que es mucho más fácil programar procesadores que diseñar circuitos específicos.

Si bien, el 8088 de Intel resulta obsoleto actualmente, es uno de los pilares básicos de la arquitectura x86, que es la más utilizada actualmente para computadoras personales, además es muy intuitivo para tener una primera experiencia con el funcionamiento de los microprocesadores. Hoy en día, los procesadores son mucho más complejos, sin embargo, los principios básicos siguen siendo los mismos.

II. Primera práctica: Funcionamiento básico del 8088.

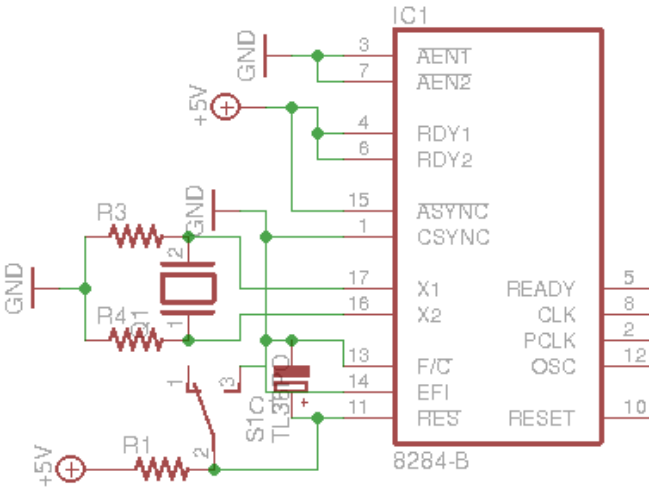
El objetivo de esta primera parte es lograr un circuito base funcional del 8088 y todo lo necesario para que este pueda ejecutar un programa guardado en una memoria ROM.

A. *El generador de reloj*

El primer e imprescindible paso para lograr el correcto funcionamiento del 8088 es conectar el generador de reloj, que es el circuito integrado 8284. Tras leer las hojas de datos tanto del 8088 como del 8284, se decidió utilizar un cristal de 12MHz, un capacitor de 22µF y una resistencia de 8.2kΩ.

Sin embargo, tras observar la salida de reloj con un osciloscopio, se decidió colocar resistencias de 500Ω entre tierra y ambos pines del cristal de cuarzo.

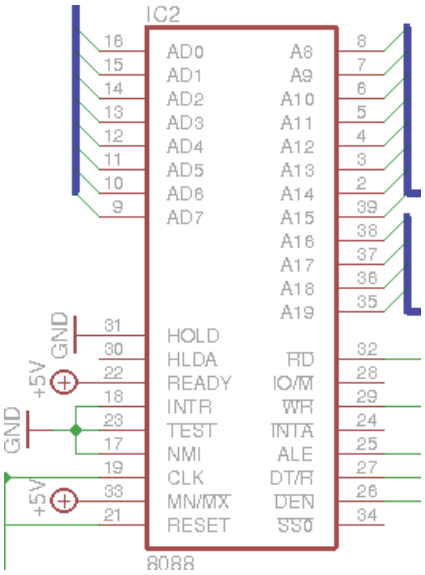
Por lo tanto las conexiones de este módulo fueron de la siguiente manera:



B. *El microprocesador*

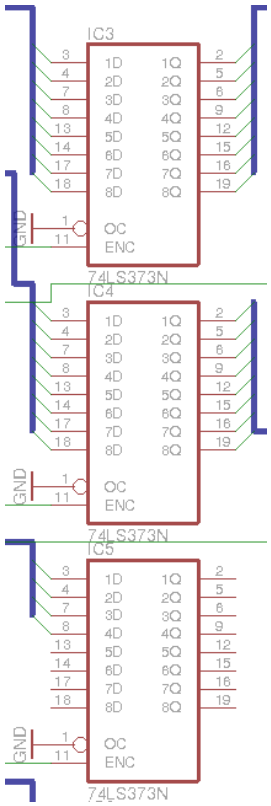
El microprocesador 8088, se podría decir, es la parte más importante del circuito. Para lograr su funcionamiento, CLK y RESET se conectan a las salidas del mismo nombre del 8284; READY se conectó a +5v, pues las memorias utilizadas son lo suficientemente rápidas como para necesitar esperar su respuesta; INTR, TEST' y NMI se conectaron a tierra, pues no fueron utilizadas; y MN/MX' se conectó a +5v porque se decidió utilizar el modo mínimo del procesador.

Los demás pines de salida o de entrada/salida, no son precisamente relevantes para este módulo, así que serán explicados en los siguientes subtemas.



C. *Los latches de direcciones*

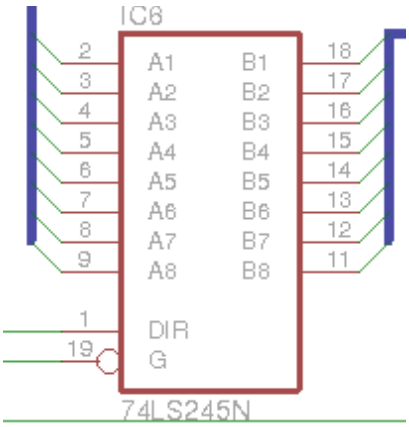
Las salidas de direcciones del 8088 están multiplexadas junto con las entradas/salidas de datos, por lo tanto es necesario guardar las direcciones en un arreglo de tres latches tipo 74373, lo cual se logró con el siguiente circuito:



donde los buses que entran a cada uno de los latches, de arriba hacia abajo, son las entradas AD0-AD7, A8-A15 y A16-A20 respectivamente; y las entradas al pin 1 de cada latch es la salida de ALE del 8088.

D. *El transmisor-receptor de datos*

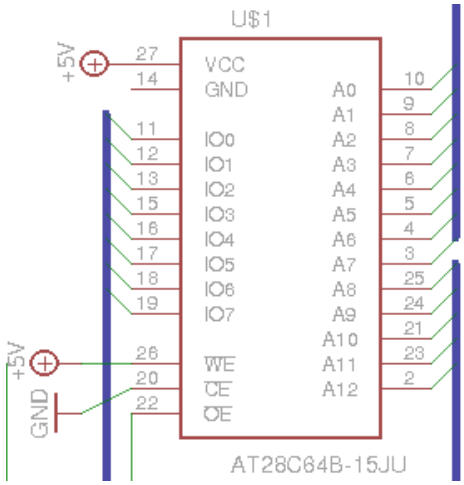
Con el objetivo de que un solo bus de datos pueda ser utilizado, a través de todo el circuito, para enviar y recibir datos, es necesario conectar un transmisor-receptor entre el 8088 y el bus de datos que recorrerá casi todos los componentes del circuito. Esto se logró utilizando un circuito integrado 74245, que recibe de un extremo las salidas AD0-AD8 del 8088 y del otro extremo el bus de datos general del circuito. A su vez, recibe las señales DT/R' y DEN' del 8088 como señales de DIR y ENABLE' respectivamente. Quedando así de la siguiente manera:



E. *La memoria ROM*

Es necesario utilizar una memoria ROM para guardar el programa a ser ejecutado. Por sus especificaciones y su disponibilidad en las tiendas de electrónica locales, se decidió utilizar una 28C64, que es una EEPROM de 8k x 8 bits.

La entrada CE' se conectó a tierra, pues no afecta que siempre esté habilitada la memoria y es necesario que esté habilitada al momento de leerla; WE' se conectó a +5v, pues la memoria será solamente leída y nunca escrita; OE' se conectó al decodificador de direcciones, que será explicado en el siguiente subtema; las entradas de direcciones se conectaron a las salidas 0-12 de los latches de direcciones; y las salidas de datos se conectaron al bus de datos, o sea, al transmisor-receptor.

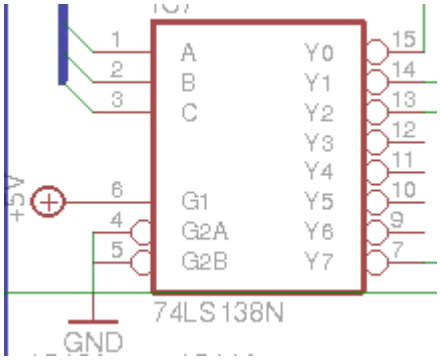


F. *El decodificador de direcciones*

Para logar un circuito más sencillo, se decidió manejar los dispositivos como si fueran memoria, por lo que cada dispositivo necesitará un decodificador de direcciones.

Se decidió utilizar un demultiplexor 74138 como base para el decodificador de direcciones, donde las señales de control A, B y C se conectaron a las señales de direcciones 13, 14 y 15 (respectivamente) de los latches y las señales de habilitación G1, G2A y G2B se conectaron a +5v, tierra y tierra respectivamente, para que de esta manera, el demultiplexor entregue un 0 en la salida seleccionada y un 1 en las demás.

Por diseño, el 8088 comienza a ejecutar su programa en la posición 0xFFFF0, por lo que resultó conveniente utilizar la salida Y7 del demultiplexor (que se selecciona con 111) como señal de activación para la ROM, conectándose directamente al pin OE' de ésta.



G. *El programa de prueba*

El circuito, teóricamente, ya es funcional. El microprocesador ya es capaz de leer un programa de la ROM, realizar operaciones según éste le indique, y guardar los resultados en sus registros internos. Sin embargo, aún no se cuenta con ningún dispositivo de salida con el cual mostrar los resultados. Es necesario, entonces, utilizar un analizador lógico para probar el circuito actual.

El programa más sencillo que se pudo idear para probar el circuito, fue un salto a sí mismo, que se logró escribiendo los bytes

```
EA0000FFFF
```

en la posición 0x1FF0 de la ROM, que es la que el 8088 entiende como 0xFFFF0.

Al momento de probar la correcta ejecución del programa, se conectó ALE como reloj del analizador lógico, y los 8 bits menos significativos de las direcciones de la ROM como datos a leer por el analizador. De esta manera, se consiguió que el analizador leyera el siguiente ciclo:

```
F0
F1
F2
F3
F4
F5
```

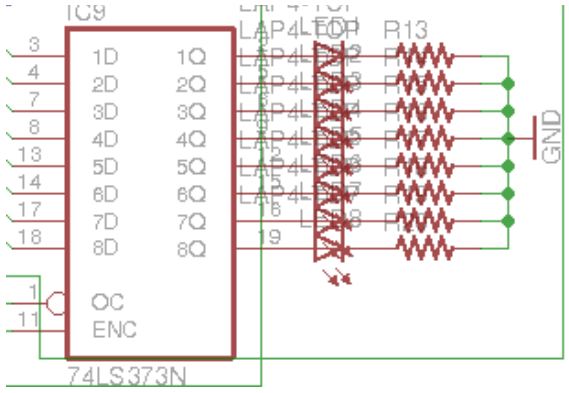
dando a entender que todo funciona correctamente.

III. SEGUNDA PRÁCTICA: LEDS

El objetivo de esta segunda parte del proyecto es lograr tener LEDs que sirvan como dispositivos de salida, para mostrar datos generados por el programa.

A. *El latch de los LEDs*

Es necesario tener un latch que mantenga los datos que se envíen, a través del bus de datos, a los LEDs, y sólo acepte nuevos datos cuando estos nuevos datos sean realmente destinados a los LEDs. Con este objetivo, se decidió utilizar nuevamente un 74373, que reciba como entradas al bus de datos, conecte sus salidas a un arreglo de 8 LEDs, y reciba como señal de habilitación de latch (LE) una nueva salida del decodificador de direcciones.



B. *El decodificador de direcciones*

Nuevamente es necesario diseñar un decodificador de direcciones para un nuevo dispositivo. Sin embargo, dada la naturaleza del ciclo de ejecución del 8088, el decodificador de direcciones para el latch de los LEDs se vuelve un poco más complicado.

Se decidió que los LEDs serían activados con la señal 000 del demultiplexor de direcciones, sin embargo, para lograr sincronizar la escritura al latch y el bus de datos, fue necesario ampliar su decodificador de direcciones según la ecuación

$$LE = CLK\ WR\ Y_0,$$

donde LE es la señal de control del latch de los LEDs, CLK es la señal de reloj del 8088, WR es la señal de escritura del 8080 y Y_0 es la salida seleccionada con 000 del demultiplexor de direcciones, estas dos últimas necesitando ser negadas previamente ya que son activas en bajo.

C. *El programa de prueba*

Para probar este segundo avance de la computadora, se decidió hacer un contador binario que mostrara cada número en el arreglo de LEDs. Esto se logró gracias al siguiente código:

```
ORG 0000H
MOV AX, 0000H
MOV DS, AX
INIT:
MOV 0000H, AL
INC AL
MOV DX, 0XEFFF
JMP DELAY
DELAY:
INC DX
JZ DX, INIT
JMP DELAY
```

Con el cual se mostró el contador y se determinó que todo funcionaba correctamente.

IV. TERCERA PRÁCTICA: SWTICHES

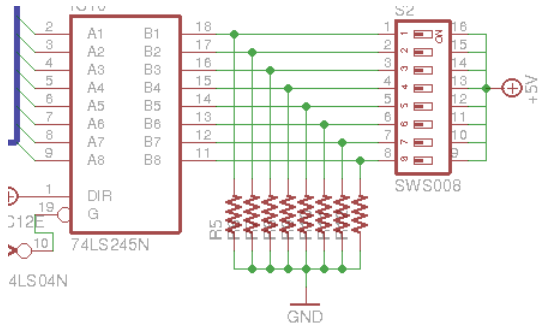
El objetivo de esta tercera parte del proyecto es lograr tener 8 switches que permitan la entrada de datos, para lograr ejecutar programas que dependan de la entrada del usuario.

A. *El buffer de tercer estado de los switches*

Como es necesario que los switches escriban al bus de datos al momento de ser leídos, pero no pueden escribir todo el tiempo pues esto interferiría con los datos que haya en el bus, es necesario utilizar un buffer del tercer estado que solo permita el paso de las señales de los switches cuando estos deben ser leídos, y en otra situación actúe como alta impedancia.

Se decidió usar un 74245 para esta función. Si bien este circuito integrado es en realidad un transmisor-receptor, puede ser utilizado como buffer del tercer estado si la señal de dirección se mantiene fija a un solo estado.

De esta manera, se conectó un 74245 con el pin DIR a +5v, los pines B1-B8 a los switches, los pines A1-A8 al bus de datos, y la señal de habilitación G al decodificador de direcciones de este dispositivo.



B. *El decodificador de direcciones*

El decodificador de direcciones en este caso es muy parecido al del latch de los LEDs, con la diferencia de que en vez de tomar en cuenta WR, se toma en cuenta RD por obvias razones, y de que se niega toda la ecuación al final porque el pin de habilitación del 74245 es activo en 0. La ecuación, por lo tanto, queda así:

$$G' = (CLK\ RD\ Y_1)'.$$

C. *El programa de prueba*

Para probar esta nueva adición al circuito, se escribió un programa que leyera el estado de los switches y lo mostrara en los LEDs, de la siguiente manera:

```
ORG 0000H
MOV AX, 0000H
MOV DS, AX
LP:
MOV AL, [2000H]
MOV [0000H], AL
JMP LP
```

V. CUARTA PRÁCTICA: RAM

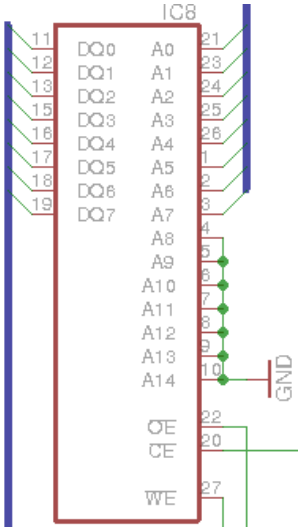
Hasta ahora, la computadora es capaz de ejecutar un programa guardado en ROM, recibir datos del usuario, y mostrar los resultados. Sin embargo, para muchas aplicaciones, los registros de propósito general del 8088 no son suficientes para almacenar todos los datos necesarios. Por esta razón, es necesaria una unidad de memoria RAM.

A. *La memoria RAM*

Debido a su disponibilidad, se seleccionó el circuito integrado 62256 para fungir como memoria RAM de la computadora.

Las entradas/salidas DQ0-DQ7 se conectaron al bus de datos y, debido a que se necesitaban menos de 256 bytes de memoria, se decidió conectar solamente las entradas de direcciones A0-A7 al bus de direcciones, dejando A8-A14 en tierra.

WE' se conectó a WR' y OE' a RD', mientras que CE' pasó a ser la entrada del decodificador de direcciones.



B. *El decodificador de direcciones*

En este caso, al igual que con la ROM, el decodificador de direcciones resultó ser muy sencillo; simplemente se conectó el pin CE' de la RAM a la salida Y2 del demultiplexor.

C. *El programa de prueba*

En esta ocasión, para demostrar el funcionamiento de la RAM, se creó un programa que leyera 10 bytes, los guardara en RAM, y uno a uno fuera leyendo y mostrándolos en los LEDs.

De esta manera, el código fuente del programa quedó de la siguiente manera:

```
ORG 0000H
MOV AX, 0000H
MOV DS, AX
MOV BX, 0X4000
MOV CX, 0X0000
CONT:
    MOV [0X0000], DL
    INC DL
    CALL DELAY
    CMP DL, 0XFF
    JG CONT
PRESBTN:
    MOV AL, [0X6000]
    AND AL, 0X01
    CMP AL, 0X00
    JZ PRESBTN
SLTBTN:
    MOV AH, [0X2000]
    MOV AL, [0X6000]
    AND AL, 0X01
    CMP AL, 0X01
    JZ SLTBTN
MOV [0X0000], BL
MOV [BX], AH
INC BL
CMP BL, 0X0A
JL PRESBTN
RST:
MOV BX, 0X4000
PNTLEDS:
    MOV AL, [BX]
    MOV [0X0000], AL
    CALL DELAY
    INC BL
    CMP BL, 0X0A
    JZ RST
    JMP PNTLEDS

PROC DELAY
    MOV CX, 0X00FF
    TAG:
    DEC CX
    CMP CX, 0X0000
    JG TAG
    RET
```

VI. CONCLUSIONES

Utilizar un microprocesador no es una tarea sencilla, sin embargo, el diseño del circuito necesario sigue una secuencia lógica e intuitiva. El hecho de ir construyendo el proyecto por partes resultó ampliamente útil, pues permitió eliminar puntos de falla previamente probados.

Con el desarrollo de este proyecto, fue posible entender el funcionamiento del 8088 de Intel y el circuito necesario para que este opere correctamente, permitiendo a los estudiantes del curso tener una primera experiencia con el funcionamiento a bajo nivel de la arquitectura x86.

VII. ANEXO: DIAGRAMA COMPLETO DE CONEXIONES