UNIVERSIDAD DE CARABOBO
FACULTAD EXPERIMENTAL DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN
ARQUITECTURA DEL COMPUTADOR

# Practica 1
# Microcontrolador Intel 8051

Icler Anaya — C.I. 24.450.482
Wilsen Hernandez — C.I. 24.993.998

## Introducción

El microcontrolador ***8051*** es un miembro de la familia **MCS-51** de **Intel** desarrollado en 1980 para uso en productos embebidos. Es un microcontrolador muy popular.

En este microcontrolador encontramos un **CPU** optimizado de **8-bit** originariamente diseñado para el control de aplicaciones simples, y con extensas capacidades para el procesamiento de expresiones ***booleanas*** (lógica de bit simple), un espacio de memoria de datos de **64K** al igual que el espacio de direcciones, 32 líneas de **entrada/salida** bidireccionales e individualmente direccionales y otras características que sustentan la fama y la buena reputación que posee este microcontrolador ya que son más de 50 compañías las que han producido variaciones de este modelo y que además estas mismas compañías tienen más de 50 versiones fabricadas por ellas mismas.

En cifras hay un estimado de más de 100 millones de núcleos ***8501*** que se venden en un año convirtiéndolo en un éxito, influyendo así en muchos de microcontroladores más recientes. Los microcontroladores son la evolución de los chips, ya que ellos se pueden programar y además cuentan con 3 principales unidades de funcionamiento como lo son: Un **CPU**, una memoria y periféricos de entrada salida.

Un microcontrolador se puede encontrar en muchos electrodomésticos, automóviles modernos, aparatos electrónicos ya que dentro de sus funciones se ha de desempeñar como si fueran una especie de computadora en miniatura por ser programables, permiten la entrada salida de información que es manejada a través de una memoria, y con el constante avance tecnológico del mundo, que tiene una característica especial, que es la miniaturización de dispositivos, circuitos, elementos internos de un aparato, etc, esto conlleva a compactar las funciones y procesos de distintos equipos en la actualidad, y que en gran parte se hace posible gracias a los microcontroladores en sus variadas versiones que irán mejorando con el tiempo y seguramente integrando y reduciendo el tamaño de los mismos quizás hasta un punto en el futuro en el que tengamos computadoras completas que quepan en lo que hoy es un microcontrolador.

## Registros

- **PC (Program counter)**: También conocido como "*Instruction Pointer*", este es una dirección de 2 bytes que le señala al microcontrolador donde se encuentra la siguiente instrucción a ser ejecutada en la memoria, es decir indica la posición en que se va el procesador en la secuencia de instrucciones.

  Siempre comienza inicializada en **0000h** y esta es incrementada cada vez que una instrucción es ejecutada (si la instrucción requiere 2 o 3 bytes entonces está incrementará de acuerdo a esta cantidad dependiendo del caso).

- **ACC (Acumulador):** Es usado como un registro general para acumular los resultados de un largo número de instrucciones. Su capacidad es de 1 byte y su dirección específica es 0E0H.

- **PSW (Program Status Word):** Su función es la de guardar un número importante de bits definidos por las instrucciones del microcontrolador. Se encuentra por default en la dirección **0D0H**, además en él se aloja la información de los indicadores empleados en la toma de decisiones de las instrucciones condicionales de los mismos indicadores o también llamados *flags*.

- **DP (Data Pointer):** Su objetivo es el de acceder a datos o código externo. Es conocido por las siglas **DPTR** y es un registro de 16 bits que tiene dos direcciones, la dirección $82_H$ (byte bajo) y la dirección $83_H$ (byte alto).

## Comportamiento del programa

Primero se selecciona el *banco de registro 0* (ubicado en la dirección **00H**) y se inicializa la memoria con la cantidad de números a generar, se guarda en **R1** el primer número de la serie, (El registro **PC** va incrementando a medida que cada instrucción del programa se va ejecutando). Se selecciona el *banco de registro 1* (ubicado en la dirección **08H**) para cargar en **R0** la dirección donde se va a almacenar el próximo número de la serie, se vuelve a seleccionar el banco de registro 0 y comienza el ciclo (**R0** disminuye en 2 unidades debido a los dos primeros números que ya se generaron).

Al comenzar el ciclo se almacena el primer número de la serie que tiene **R1** en el acumulador, luego se incrementa **R1** para obtener el siguiente número, estas dos cantidades se suman y el resultado se guarda en el acumulador. Se selecciona el *banco de registro 1* para almacenar en **R0** el resultado y luego incrementa este para guardar el valor del siguiente número de la serie.

**Data Space**

Es la región donde se almacenan temporalmente los valores de los registros “R” que utiliza el programa.

**¿En qué dirección empieza la ejecución del programa?**

Comienza en la dirección **100H**.

**¿En qué dirección termina la ejecución del programa?**

Termina en **11D** (El programa no se ejecutó por completo, *Trap Undefined Address:* **D:0080H**).

**¿Qué tamaño en bytes tiene el programa?**

Tienes 132 bytes.

**¿Cómo se lee y escribe en memoria?**

Para poder leer y escribir en memoria, es fundamental tener en cuenta que el emulador nos ofrece la posibilidad de estas dos acciones mediante su interfaz. A la derecha de las instrucciones del programa, justo debajo de donde se almacenan los registros, es donde se encuentra la memoria. Presionando **ESC** en el teclado se puede acceder a ella manualmente y bajamos hasta ellas. Se puede modificar los datos que ahí se encuentran. Para que el programa pueda leerlo, es necesario que se haga un direccionamiento directo hacia, por ejemplo, el acumulador, algo como “**mov A,30H**” esto haría que lo que esté almacenado en la dirección de memoria **30H** pase a **A**.

**¿Qué son los modos de direccionamiento? Liste los modos de direccionamiento que posee el 8051 y dé ejemplos.**

<u>Direccionamiento Directo</u>

En el direccionamiento directo, se indica la dirección a operar de forma absoluta. Para la familia de microcontroladores 8051 se dispone de 256 direcciones directas, correspondientes a (Ram interna + Registros SFR).

El **OpCode** va seguido de un byte que representa la dirección.

MOV A,30H ; A<-(30H), El contenido de la dirección 30H se mueve al acumulador
ADD A,31H ; A<-(A)+(31H), Se suma el contenido de acumulador con el de la dirección 31H; el resultado se guarda en el acumulador.

Direccionamiento Indirecto

Se especifica un registro que contiene la dirección del dato a operar.

R0, R1 cuando se accede a la memoria interna de direccionamiento indirecto 256 Bytes.
**DPTR** para el acceso a la memoria externa 64K Bytes.

MOV R0,#30H ; RO<-30H, mover al registro 0 el dato inmediato 30H (constante)
MOV A,@R0 ; A<-((R0)), mueve el contenido de la posición de memoria indicada en R0 al Acc.

Direccionamiento Inmediato

Se da cuando el operando fuente es una constante en vez de una variable, la constante puede ser añadida a las instrucciones como un byte a una dirección inmediata. Los operandos inmediatos son precedidos por el símbolo # en el lenguaje como tal, y este puede ser una constante numérica, una variable simbólica o una expresión aritmética que usa constantes, símbolos y operadores. Con solo una excepción, (La excepción es cuando es inicializado el **data pointer** se requiere una constante) todas la instrucciones que operan bajo direccionamiento inmediato usan datos de 8 bits.

Direccionamiento Relativo

Es empleado en ciertas instrucciones **JUMP** y tiene como ventaja códigos de posiciones independientes. Una dirección relativa es un valor con signo de 8 bits el cual se suma al **program counter** para formar la dirección de la siguiente instrucción.

Direccionamiento Absoluto

Solo se usa con **ACALL** (Denota la instrucción de llamado absoluto) y **AJMP**( Denota la instrucción salto absoluto). Los once bits menos significativos de la dirección de destino provienen del **opcode** y los 5 bits más significativos son los actuales 5 bits más significativos del **program counter.**

Direccionamiento Largo

Se usa únicamente con **LCALL** ( *Long Call* o llamado largo) y **LJMP** ( *Long Jump* o Salto Largo). Estas instrucciones de 3 bytes incluyen una dirección de destino completa de 16 bits como bytes 2 y 3.

Ejemplo: LJMP, $84AF2_H$ # Salta a la posición de memoria $84AF2_H$

Direccionamiento Indexado

Se utilizan dos registros para apuntar a la dirección que contiene el dato. La suma del contenido del **DPTR** más el acumulador determina la dirección a operar.

Este direccionamiento está limitado a dos únicas instrucciones que mueven datos de la ROM al acumulador.

MOVC A,@A+DPTR ; A<-((DPTR+A)), el contenido de la posición que apunta el DPTR+A se deposita en el Acc.

**¿Los registros de todos los bancos pueden ser usados para formar instrucciones de direccionamiento indirecto de registro? Explique.**

No pueden ser usados, el direccionamiento indirecto explica claramente que solo el **R0** y **R1** está habilitados para direccionamiento indirecto en memoria interna. (256k). El **DPTR** se usa para la memoria externa (64K).

**¿Cómo se hacen las comparaciones? Indique ejemplos.**

Las comparaciones que podemos encontrar en el 8051 son las siguientes:

```
JZ rel                    #salta si A = 0
JNZ rel                   #salta si A ≠ 0
DJNZ <byte>,rel           #decrementa y salta si no es igual a 0
CJNE A,<byte>,rel         #salta si A ≠ <byte>
CJNE <byte>,#dato,rel     #salta si <byte> ≠ #dato
```

Un ejemplo de ello es el siguiente:

```
          MOV       CONTADOR,#10         ;N = 10
LAZO:     ; Comienzo del conjunto de instrucciones)
          DJNZ      CONTADOR,LAZO
```

**¿Cómo se accede a 32 registros con sólo 8 variables para abordar registros?**

Mediante los bancos de registros en la memoria RAM interna, la cual se subdivide en 128 bytes de memoria bajos y en 128 bytes de memoria altos. En los primeros 128, se encuentran 4 bancos de 8 registros cada uno. Estos registros son de gran ayuda para la simplificación de los programas, debido a que cada uno de ellos nos permiten almacenar datos momentáneamente y realizar un amplio número de instrucciones del 8051. Podemos movernos sobre estos 4 bancos utilizando el PSW.

**¿Se alcanza almacenar algún número de Fibonacci en la dirección 80H? ¿Esto a que se debe?**

No es posible ya que el espacio de memoria desde **80H** hasta **FFH** está reservado para registros de funciones especiales como puertos, estado y control de Bits, *timer*, SP, acumuladores, etc.

**Modifique el programa para que todos los términos de la sucesión de Fibonacci se muestre en decimal. ¿Cuál es el último término que se puede representar sin que ocurra desbordamiento?**

El último término que se puede representar sin que ocurra desbordamiento es el 89, porque el siguiente término es el 144, y se puede apreciar que no es posible ser representado.

## Conclusión

El microcontrolador ***8051***, ha demostrado que, a pesar del tiempo que lleva desde su creación, y lanzamiento en el mercado. Es uno de los cuales que con su característica de cuatro conjuntos separados de registros (la cual es muy valorada en este microcontrolador), es decir que tiene 4 bancos de 8 registros (lo cual no se había mencionado con anterioridad), se ha podido implementar unas de las sucesiones muy conocida en la matemática, como lo es la sucesión de Fibonacci, y en la cual se pudo observar que si, se le indica al código desarrollado para esta práctica, que represente los términos en decimal, el último que no provoca un desbordamiento es el 89.

A pesar de que los microcontroladores ***8051*** modernos ofrecen muchas mejoras sobre el original, una de las más comunes que incluyen es un **watchdog timers** (un temporizador programable que "resetea" el microcontrolador si no se refresca en cierto tiempo), también osciladores internos, memoria de programa **Flash ROM** interna, código de inicialización en **ROM**, almacenamiento en **EEPROM** interna, **I²C**, **SPI**, **USB**, generadores **PWM**, conversores analógicos **A/D** y **D/A**, relojes de tiempo real **RTC**, temporizadores y contadores extra, facilidades de depuración internas, más fuentes de interrupción, modos de bajo consumo, interfaz **CAN**, etc. Estos cientos de características que simplifican y mejoran varias funciones del diseño original del ***8051.***

Si bien hoy en día la microelectrónica ha tenido grandes avances, se podría decir que las características del ***8051*** contribuyeron a esto.