Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

**INVESTIGACIONES**

Nombre: **Martinez Reyes Fernando**

Profesor: **Dr. García Ruiz Alejandro Humberto**

Asignatura: **Programación de Miprocesadores**

**8v**o. Semestre – Grupo “**I**”

2022-3

# Índice

[Índice 2](#_Toc116657064)

[1.- Qué es el registro de banderas? 3](#_Toc116657065)

[2.- Descripción de los registros de bandera 5](#_Toc116657066)

[3. Saltos condicionales 6](#_Toc116657067)

[4.- Saltos incondicionales 8](#_Toc116657068)

[Ejemplo 8](#_Toc116657069)

[5.- Tabla descriptiva de saltos condicionales 9](#_Toc116657070)

[6.- Características de la instrucción CMP 11](#_Toc116657071)

[7.- Instrucción Loop 13](#_Toc116657072)

[8.- Estructura condicional 15](#_Toc116657073)

[9.- Estructura iterativa 17](#_Toc116657074)

[10.- Registros de propósito 18](#_Toc116657075)

[11.- Descripción y utilización de los nemotécnicos INC Y DEC. 18](#_Toc116657076)

# 1.- Qué es el registro de banderas?

**Registro de bandera del microprocesador 8086**

Requisito previo: [registro de bandera en el microprocesador 8085](https://www.geeksforgeeks.org/flag-register-8085-microprocessor/)  
El registro de bandera es un registro de propósito especial. Según el valor del resultado después de cualquier operación aritmética y lógica, los bits de bandera se establecen (1) o se restablecen (0).

Diagram

Description automatically generated  
**Figura:** formato del registro de banderas  
Hay un total de 9 banderas en 8086 y el registro de banderas se divide en dos tipos:

**(a) Indicadores de estado:** hay 6 registros de indicadores en el microprocesador 8086 que se activan (1) o restablecen (0) dependiendo de la condición después de una operación de 8 o 16 bits. Estas banderas son banderas condicionales/de estado. 5 de estas banderas son las mismas que en el caso del microprocesador 8085 y su funcionamiento también es el mismo que en el microprocesador 8085. La sexta es la bandera de desbordamiento.

Las 6 banderas de estado son:

1. **Señal de bandera (S)**
2. **Bandera cero (Z)**
3. **Bandera Cary Auxiliar (AC)**
4. **Bandera de paridad (P)**
5. **Carry Flag (CY)**
6. **Indicador de desbordamiento (O):** este indicador se establecerá (1) si el resultado de una operación firmada es demasiado grande para caber en la cantidad de bits disponibles para representarlo; de lo contrario, se restablecerá (0). Después de cualquier operación, si D[6] genera cualquier acarreo y pasa a D[7] O si D[6] no genera acarreo pero D[7] genera, el indicador de desbordamiento se establece, es decir, 1. Si D[6] y D[7] ambos generan acarreo o ambos no generan ningún acarreo, luego el indicador de desbordamiento se restablece, es decir, 0.

**Ejemplo:** al agregar los bytes 100 + 50 (el resultado no está en el rango -128…127), se establecerá el indicador de desbordamiento.

MOV AL, 50 (50 es 01010000 que es positivo)

MOV BL, 32 (32 es 00110010 que es positivo)

AGREGAR AL, BL (82 es 10000010 que es negativo)

El indicador de desbordamiento se estableció cuando agregamos 2 números +ve y obtuvimos un número -ve.

**(b) Banderas de** control: las banderas de control activan o desactivan ciertas operaciones del microprocesador. Hay 3 banderas de control en el microprocesador 8086 y estas son:

1. **Bandera direccional (D):** esta bandera se usa específicamente en instrucciones de cadena.  
   Si el indicador direccional está configurado (1), acceda a los datos de la cadena desde la ubicación de memoria superior hacia la ubicación de memoria inferior.  
   Si el indicador direccional se restablece (0), acceda a los datos de cadena desde la ubicación de memoria inferior hacia la ubicación de memoria superior.
2. **Indicador de interrupción (I):** este indicador es para interrupciones.  
   Si el indicador de interrupción está establecido (1), el microprocesador reconocerá las solicitudes de interrupción de los periféricos.  
   Si el indicador de interrupción se restablece (0), el microprocesador no reconocerá ninguna solicitud de interrupción y la ignorará.
3. **Indicador de trampa (T):** este indicador se utiliza para la depuración en el chip. La configuración del indicador de trampa pone al microprocesador en modo de paso único para la depuración. En un solo paso, el microprocesador ejecuta una instrucción y entra en ISR de un solo paso.  
   Si se establece el indicador de trampa (1), la CPU genera automáticamente una interrupción interna después de cada instrucción, lo que permite inspeccionar un programa a medida que ejecuta instrucción por instrucción.  
   Si el indicador de captura se restablece (0), no se realiza ninguna función.

# 2.- Descripción de los registros de bandera

* **Indicador de acarreo (CF)** : este indicador se establece en **1** cuando hay un **desbordamiento sin firmar** . Por ejemplo, cuando agrega los bytes **255 + 1** (el resultado no está en el rango 0...255). Cuando no hay desbordamiento, este indicador se establece en **0** .
* **Indicador cero (ZF)** : se establece en **1** cuando el resultado es **cero** . Para ningún resultado cero, este indicador se establece en **0** .
* **Sign Flag (SF)** : se establece en **1** cuando el resultado es **negativo** . Cuando el resultado es **positivo** , se establece en **0** . En realidad, esta bandera toma el valor del bit más significativo.
* **Indicador de desbordamiento (OF)** : se establece en **1** cuando hay un **desbordamiento firmado** . Por ejemplo, cuando agrega los bytes **100 + 50** (el resultado no está en el rango -128...127).
* **Indicador de paridad (PF)** : este indicador se establece en **1** cuando hay un número par de bits en el resultado y en **0** cuando hay un número impar de bits. Incluso si el resultado es una palabra, ¡solo se analizan 8 bits bajos!
* **Indicador auxiliar (AF)** : se establece en **1** cuando hay un **desbordamiento sin firmar** para un nibble bajo (4 bits).
* **Indicador de habilitación de interrupción (IF)** : cuando este indicador se establece en **1** , la CPU reacciona a las interrupciones de dispositivos externos.
* **Indicador de dirección (DF)** : este indicador es utilizado por algunas instrucciones para procesar cadenas de datos, cuando este indicador se establece en **0** , el procesamiento se realiza hacia adelante, cuando este indicador se establece en **1** , el procesamiento se realiza hacia atrás.

# 3. Saltos condicionales

**salto condicional**

Esto se realiza mediante un conjunto de instrucciones de salto j<condición> dependiendo de la condición. Las instrucciones condicionales transfieren el control rompiendo el flujo secuencial y lo hacen cambiando el valor de compensación en IP.

Salto condicional

Si se cumple alguna condición específica en el salto condicional, el flujo de control se transfiere a una instrucción de destino. Existen numerosas instrucciones de salto condicional según la condición y los datos.

Las siguientes son las instrucciones de salto condicional utilizadas en datos firmados utilizados para operaciones aritméticas:

|  |  |  |
| --- | --- | --- |
| **Instrucción** | **Descripción** | **Banderas probadas** |
| JE/JZ | Salto Igual o Salto Cero | ZF |
| JNE/JNZ | Salto no igual o salto no cero | ZF |
| JG/JNLE | Salto mayor o salto no menor/igual | DE, SF, ZF |
| JGE/JNL | Salto mayor/igual o salto no menor | DE, SF |
| JL/JNGE | Saltar menos o saltar no mayor/igual | DE, SF |
| JLE/JNG | Salto menor/igual o salto no mayor | DE, SF, ZF |

Las siguientes son las instrucciones de salto condicional utilizadas en datos sin firmar utilizados para operaciones lógicas:

|  |  |  |
| --- | --- | --- |
| **Instrucción** | **Descripción** | **Banderas probadas** |
| JE/JZ | Salto Igual o Salto Cero | ZF |
| JNE/JNZ | Salto no igual o salto no cero | ZF |
| JA/JNBE | Saltar por encima o no por debajo/igual | CF, ZF |
| JAE/JNB | Saltar arriba/igual o no saltar debajo | FC |
| JB/JNAE | Saltar por debajo o no saltar por encima/igual | FC |
| JBE/JNA | Saltar por debajo/igual o no saltar por encima | FA, CF |

Las siguientes instrucciones de salto condicional tienen usos especiales y verifican el valor de las banderas:

|  |  |  |
| --- | --- | --- |
| **Instrucción** | **Descripción** | **Banderas probadas** |
| JXCZ | Saltar si CX es cero | ninguna |
| JC | Saltar si llevar | FC |
| JNC | Saltar si no llevar | FC |
| JO | Saltar si se desborda | DE |
| JNO | Saltar si no hay desbordamiento | DE |
| JP/JPY | Salto de paridad o salto de paridad par | FP |
| JNP/JPO | Saltar sin paridad o saltar paridad impar | FP |
| JS | Signo de salto (valor negativo) | SF |
| JNS | Saltar sin señal (valor positivo) | SF |

La sintaxis para el conjunto de instrucciones J<condición>:

Ejemplo,

CMP AL, BL

JE EQUAL

CMP AL, BH

JE EQUAL

CMP AL, CL

JE EQUAL

NON\_EQUAL: ...

EQUAL: ...

# 4.- Saltos incondicionales

Esto lo realiza la instrucción JMP. La ejecución condicional a menudo implica una transferencia de control a la dirección de una instrucción que no sigue a la instrucción que se está ejecutando actualmente. La transferencia de control puede ser hacia adelante, para ejecutar un nuevo conjunto de instrucciones o hacia atrás, para volver a ejecutar los mismos pasos.

Salto incondicional

Como se mencionó anteriormente, esto lo realiza la instrucción JMP. La ejecución condicional a menudo implica una transferencia de control a la dirección de una instrucción que no sigue a la instrucción que se está ejecutando actualmente. La transferencia de control puede ser hacia adelante, para ejecutar un nuevo conjunto de instrucciones o hacia atrás, para volver a ejecutar los mismos pasos.

Sintaxis

La instrucción JMP proporciona un nombre de etiqueta donde el flujo de control se transfiere inmediatamente. La sintaxis de la instrucción JMP es:

JMP label

### Ejemplo

MOV AX, 00 ; Initializing AX to 0

MOV BX, 00 ; Initializing BX to 0

MOV CX, 01 ; Initializing CX to 1

L20:

ADD AX, 01 ; Increment AX

ADD BX, AX ; Add AX to BX

SHL CX, 1 ; shift left CX, this in turn doubles the CX value

JMP L20 ; repeats the statements

# 5.- Tabla descriptiva de saltos condicionales

Table

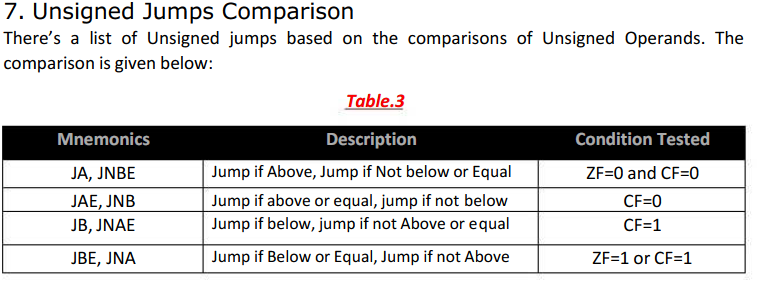
Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence



**SALTOS CONDICIONALES RENOMBRADOS**

**Table

Description automatically generated**

# 6.- Características de la instrucción CMP

¿Qué es CMP en el microprocesador 8086?

La instrucción CMP compara dos operandos. Se utiliza generalmente en ejecución condicional. Esta instrucción básicamente resta un operando del otro para comparar si los operandos son iguales o no. Se utiliza junto con la instrucción de salto condicional para la toma de decisiones.

¿Qué indicadores se ven afectados por la instrucción CMP en 8086?

Las instrucciones CMP y TEST solo afectan a los indicadores y no almacenan un resultado (estas instrucciones se utilizan para tomar decisiones durante la ejecución del programa). Estas instrucciones afectan solo a estas banderas: CF, ZF, SF, OF, PF, AF.

¿Qué es CMP en microprocesador?

Microprocesador8085. En el conjunto de instrucciones 8085, CMP es un mnemotécnico que significa "CoMPareAccumulator" y hereR significa cualquiera de los siguientes registros, o la ubicación de memoria M señalada por el par HL. R = A, B, C, D, E, H, L o M. Esta instrucción se utiliza para comparar el contenido del acumulador con el registro R dado.

¿Cuántas banderas se ven afectadas después de ejecutar la instrucción CMP?

solo se verán afectadas las banderas de acarreo y cero. ninguna bandera se verá afectada. Mientras se ejecuta la instrucción CMP B, el microprocesador compara el contenido del registro B con el contenido del acumulador.

¿Cuál es la diferencia entre CMP y sub instrucción?

¿Cuál es la diferencia entre CMP y sub instrucción? CMP se usa para comparar 2 registros restándolos entre sí, pero la respuesta no se guarda, mientras que SUB resta 2 registros y guarda la respuesta.

¿Cuál es la diferencia entre macro y procedimiento e indicar su sintaxis?

Una macro se usa para una pequeña cantidad de instrucciones; en su mayoría, menos de diez instrucciones, mientras que un procedimiento se utiliza para un gran número de instrucciones; en su mayoría, superior a diez instrucciones. Por lo tanto, esta es la principal diferencia entre macro y procedimiento.

¿Qué hace CMP?

Un panel metabólico completo (CMP, por sus siglas en inglés) es una prueba que mide 14 sustancias diferentes en la sangre. Proporciona información importante sobre el equilibrio químico y el metabolismo de su cuerpo. El metabolismo es el proceso de cómo el cuerpo utiliza los alimentos y la energía.

¿Cómo funciona CMP en 8085?

El conjunto de instrucciones del 8085 tiene dos tipos de operaciones de comparación: comparación con acumulador (CMP) y comparación inmediata con acumulador (CPI). Esta es una instrucción de 1 byte. Compara el byte de datos en el registro o memoria con el contenido del acumulador. Si A es menor que (R/M), se establece el indicador CY y se restablece el indicador Cero.

¿Cuál es la similitud y la diferencia entre la instrucción de restar y comparar?

Las instrucciones de comparación y resta en el 8085 restan un operando de otro y establecen banderas en consecuencia. La instrucción de resta almacena el resultado en el acumulador, mientras que la instrucción de comparación no lo hace; a excepción de las banderas, la instrucción de comparación "tira" el resultado.

¿Qué tipo de instrucciones hace un microprocesador 8086?

El microprocesador 8086 admite 8 tipos de instrucciones: instrucciones de transferencia de datos; Instrucciones Aritméticas; instrucciones de manipulación de bits; Instrucciones de cadena; Instrucciones de Transferencia de Ejecución de Programa (Instrucciones de Rama y Bucle) Instrucciones de Control del Procesador; instrucciones de control de iteraciones; Instrucciones de interrupción

¿Cuáles son las características de la instrucción CMP?

Mencione las características de las instrucciones CMP. – La instrucción CMP se puede utilizar para comparar dos números de 8 o 16 bits. – Siempre que se realiza una operación de comparación, el resultado de dicha operación se refleja en una de las seis banderas de estado CF, AF, OF, PF, SF y ZF.

Las diversas características de la instrucción Comparar (CMP) son las siguientes:  
  
- La instrucción CMP se puede utilizar para comparar dos números de 8 o 16 bits.  
  
- Cada vez que se realiza una operación de comparación, el resultado de dicha operación se refleja en uno de los seis indicadores de estado CF, AF, OF, PF, SF y ZF.  
  
-La operación CMP también se conoce como el método de resta, ya que utiliza el complemento a dos para ello.  
  
- El resultado de una operación CMP nunca se guarda, pero en correspondencia con el resultado de la instrucción, los estados de las banderas pueden cambiarse o restablecerse.  
  
- Los operandos de la instrucción CMP pueden residir en la memoria o pueden ser un componente de una instrucción.

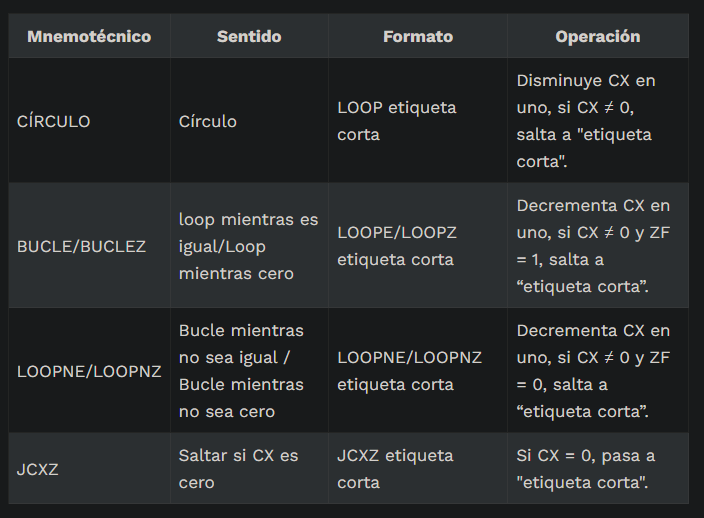
# 7.- Instrucción Loop

A veces, mientras se ejecuta un programa, será necesario repetir la ejecución de algunas instrucciones un número específico de veces. En todos los lenguajes de programación existen unas estructuras específicas para expresar esta ejecución repetida de instrucciones.

De manera similar, los diseñadores del microprocesador 8086 idearon un grupo de instrucciones para realizar la ejecución repetida de una serie de instrucciones un número específico de veces mucho más fácil. Estas instrucciones se denominan instrucciones LOOP.

Instrucciones LOOP del microprocesador 8086:

Las instrucciones de bucle hacen que el microprocesador ejecute una serie de instrucciones repetidamente. Básicamente, las instrucciones LOOP son instrucciones de salto corto en una condición, es decir, cuando se cumple la condición se toma un salto corto cuyo destino o dirección objetivo está en el rango de -128 bytes a +127 bytes desde la dirección de instrucción después de la instrucción LOOP. Las diversas instrucciones LOOP y sus funciones se enumeran a continuación.



Todas las instrucciones anteriores son instrucciones de control de iteración y se utilizan para ejecutar una serie de instrucciones repetidamente en el cumplimiento exitoso de la condición. La condición es verificar el registro CX y el indicador cero (ZF) o puede ser solo verificar el registro CX. El registro CX debe cargarse con el número de iteraciones a realizar, antes de ingresar a la sección del código terminada por las instrucciones LOOP. Veamos brevemente cada instrucción LOOP.

Instrucción de bucle:

La instrucción LOOP ejecuta el grupo de instrucciones varias veces y utiliza el modo de direccionamiento relativo. El número de iteraciones dependerá de la condición a cumplir. El registro CX realizará la operación LOOP en las instrucciones que se van a iterar.

Por cada ejecución de la instrucción LOOP, el CX se reduce automáticamente en uno sin afectar los indicadores y realiza un salto corto a la dirección de destino. Este ciclo continuará hasta que el CX sea cero. Cuando CX = 0, la ejecución del ciclo se detendrá y las instrucciones posteriores al LOOP comenzarán a ejecutarse.

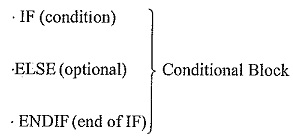
# 8.- Estructura condicional

**Declaración condicional en el programa de lenguaje ensamblador:**

Assembler admite el uso de declaraciones condicionales en el programa de lenguaje ensamblador. También permite usarlos en macrosecuencias. Estas declaraciones condicionales controlan el flujo de la ejecución del programa. Veamos las sentencias condicionales en el programa en lenguaje ensamblador y las sentencias condicionales utilizadas en las macros.

**Declaración condicional en lenguaje ensamblador Programa .IF – .ELSE – .ENDIF Declaración**

Las declaraciones condicionales se implementan en el programa de lenguaje ensamblador utilizando la estructura .IF, ELSE, ENDIF que se encuentra en el lenguaje de nivel superior. Solo la versión 6-X de MASM admite esto. Las versiones anteriores del ensamblador no admiten la instrucción IF. Aquí está el formato general para la declaración condicional IF.

[](https://www.eeeguide.com/wp-content/uploads/2018/08/Conditional-Statement-in-Assembly-Language-Program.jpg)

Como se muestra arriba, cada directiva .IF debe tener un ENDIF coincidente para finalizar una condición probada. OTRO es opcional. Proporciona una acción alternativa. El ensamblado también permite utilizar operadores relacionales con declaración .IF.

**.WHILE – Declaración .ENDW**

Al igual que la sentencia DO-WHILE en un lenguaje de nivel superior, el ensamblador admite la sentencia .WHILE- .ENDW. La declaración WHILE se usa con una condición para comenzar el ciclo y la declaración .ENDW finaliza el ciclo. Esta declaración también es compatible solo con las versiones 6.X de MASM.

**Declaraciones .BREAK y .CONTINUE**

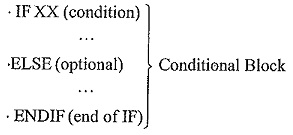
Las instrucciones .BREAK y .CONTINUE funcionan de la misma manera en un programa en lenguaje C. La instrucción .BREAK se usa para salir del bucle .WHILE. El programa 13 muestra cómo se usa la declaración .BREAK para terminar el bucle WHILE si el carácter no coincide. Esto evita más iteraciones que no son necesarias cuando no coincide ningún carácter anterior.

**.REPETIR – .UNTIL Declaración**

.REPEAT – Las sentencias .UNTIL permiten ejecutar series de instrucciones repetidamente hasta que ocurra alguna condición. .REPEAT define el inicio del ciclo y .UNTIL define el final del ciclo. Una declaración .UNTIL tiene una condición. Cuando la condición es verdadera, el bucle finaliza. Es importante tener en cuenta que las declaraciones .REPEAT y .UNTIL están disponibles para la versión 6.X de MASM. El programa 14 usa declaraciones .REPEAT y .UNTIL. Este programa enumera todos los alfabetos.

**Sentencias de montaje condicional en macros Sentencia IF-ELSE-ENDIF**

Las declaraciones de ensamblaje condicional se implementan en macros utilizando la estructura IF - ELSE - ENDIF que se encuentra en lenguajes de nivel superior. Aquí está el formato general para la familia IF de sentencias condicionales.

[](https://www.eeeguide.com/wp-content/uploads/2018/08/Conditional-Statement-in-Assembly-Language-Program-1.jpg)

Como se muestra arriba, cada directiva IF debe tener un ENDIF coincidente para terminar una condición probada. OTRO es opcional. Se utiliza para proporcionar una acción alternativa. 'XX' después de IF define varias formas utilizadas para la declaración IF.

**Declaración REPETIR**

En macro, la declaración REPEAT se usa para repetir la secuencia de macro durante un número fijo de tiempo. El conteo de repeticiones se especifica inmediatamente después de la declaración REPEAT como se muestra en el programa. El programa muestra una definición de macro que envía 26 caracteres ASCII de la A a la Z a la pantalla de video. Las declaraciones dentro de REPEAT y el primer ENDM se repiten 26 veces. Es importante tener en cuenta que esta macro tiene dos declaraciones ENDM. El primero representa el final de REPETIR y el segundo representa el final de MACRO.

**MIENTRAS Declaración**

En macro, la declaración WHILE se usa para repetir la secuencia de macro hasta que la expresión especificada con ella sea verdadera. Al igual que REPETIR, el final del ciclo se especifica mediante la instrucción ENDM. La instrucción WHILE permite utilizar operadores relacionales en su expresión. La Tabla 8.6 muestra los operadores relacionales usados ​​con declaraciones WHILE.

**PARA declaración**

Una instrucción FOR en la macro repite la secuencia de la macro para una lista de [datos](https://www.eeeonline.org/) . Por ejemplo, si pasamos dos argumentos a la macro, en la primera iteración, la instrucción FOR proporciona la secuencia de la macro utilizando el primer argumento y, en la segunda iteración, proporciona la secuencia de la macro utilizando el segundo argumento. Al igual que la declaración WHILE, el final de FOR se indica mediante la declaración ENDM. El programa muestra el uso de la instrucción FOR en la macro.

# 9.- Estructura iterativa

**Estructura iterativa**

Está formada fundamentalmente por una estructura secuencial (con otras estructuras embebidas si es necesario) y un predicado. La estructura iterativa se ejecutará tantas veces como sea necesario hasta que el predicado se evalúe como verdadero (o como falso dependiendo del tipo). Según el momento en el que se evalúe el predicado existen tres estructuras iterativas:

* Estructura **mientras**: El predicado se evalúa antes de ejecutarse la estructura secuencial asociada. La estructura secuencial se ejecutará hasta que el predicado sea evaluado como falso. Si la primera vez que se evalúa el predicado este es falso, no se ejecuta nunca.

* Estructura **repetir… hasta que**: El predicado se evalúa tras ejecutar una primera vez la estructura secuencial. Si este es falso vuelve a ejecutarse hasta que este se evalúe como verdadero.

* Estructura **iterar**: En este caso el predicado puede evaluarse en un instante previamente definido de la estructura secuencial, y en este momento dependiendo de si es verdadero o falso se continuará la ejecución de la estructura iterativa o no. Esta estructura es una generalización de las dos anteriores, no siendo estrictamente necesario su uso a la hora de definir un algoritmo de manera estructurada. De hecho hay autores que mencionan que su uso puede devenir en código de baja calidad o pobremente estructurado.

Además de esas tres estructuras iterativas se utiliza una más, que podía teóricamente construirse utilizando las anteriores pero que se añade por su gran utilidad y versatilidad. La estructura para

* Estructura **para**: Está formada por cinco elementos, una variable, un valor inicial, un valor final, un incremento y una estructura secuencial. La variable toma el valor inicial, y si este no ha llegado al valor final, se ejecuta la estructura secuencial con el valor de la variable asignado. En cada iteración se incrementa el valor de la variable en lo designado por incremento, iterándose la estructura secuencial tantas veces como sea necesario para que la variable llegue al valor final.

# 10.- Registros de propósito

**EAX:** Muy utilizado en los procesadores Intel, se emplea como Acumulador en instrucciones lógico – aritméticas y las relacionadas con las transferencias entre la memoria y la CPU.

**EBX:** Contiene la base de la dirección donde empieza una estructura de datos, bien sea un array, una pila.

**ECX:** Es habitual utilizarlo como contador en instrucciones que contengan iteraciones, es decir, que se repitan distintas veces, por ejemplo, en la operación de la notación ROR.

**EDX:** Se utiliza como apoyo al EAX. Si por ejemplo el valor del producto no cabe en EAX se agranda con este registro. También para operaciones de E/S, posiciones de E/S que van periféricos o vienen de periféricos.

# 11.- Descripción y utilización de los nemotécnicos INC Y DEC.

**INC:**

Propósito: Incrementar el operando.

Sintaxis:

**INC destino**

La instrucción suma 1 al operando destino y guarda el resultado en el mismo operando destino.

**DEC:**

Propósito: Decrementar el operando

Sintaxis:

**DEC destino**

Esta operación resta 1 al operando destino y almacena el nuevo valor en el mismo oeprando.