CONTENIDO PARCIAL 2 MICRO

[Servicios de Interrupción 2](#_Toc117439215)

[Procedimientos y cadenas 4](#_Toc117439216)

[Ensambladores en la actualidad 8](#_Toc117439217)

[Macros 12](#_Toc117439218)

# Servicios de Interrupción

Interrupción: evento que altera la secuencia en el ejecución de un programa para llevar a cabo acciones específicas.

Una interrupción hace un alto en la ejecución secuencial para ejecutar una serie de instrucciones específicas y finalmente reanudar la ejecución detenida.

Tipos de interrupciones:

De hardware:

Internas

Externas

De software:

Por BIOS

Por DOS

Interrupciones de hardware:

Interna: generadas por ciertos eventos durante la ejecución de un programa. Manejadas y requeridas por la UC. No son modificables

Externa: generadas por los dispositivos periféricos. Dependen de las señales de los periféricos.

Interrupciones de software:

Por BIOS: rutina de entrada/salida y tablas que indican los estados de los dispositivos del sistema. No tiene protección respecto al equipo. Rango: 0h – 19h.

Por DOS: emplea las funciones del sistema operativo para la manipulación del hardware. Se montan sobre las interrupciones por BIOS facilitándolas. Rango: 20h – 3fh.

Interrupciones de software por DOS:

Generadas por el ensamblador. Invocadas con la palabra reservada int, según un numero especifico asignado. Requieren condiciones previas a su invocación para ejecutar las instrucciones específicas.

Tabla de servicios:

Tabla de servicios de interrupción:

Ocupa los primero 1024 bytes de la memoria (0000h – 03FFh). Contiene 256 interrupciones con desplazamiento y posición relativa. Constituye un vector de interrupciones que apuntan al ISR.

ISR: (interrupt service routine)conjunto de instrucciones que le dan tratamiento a una interrupción.

Eventos:

Eventos en una interrupción:

Finalización de la ejecución de la instrucción previa a la interrupción.

Almacenamiento de todos los registros internos en la pila (Push CS, IP,Banderas).

IP recibe la dirección del ISR.

Ejecución las instrucciones del ISR, hasta encontrar IRET.

Devolución de los registros internos al momento de la interrupción (Pop CS, IP, Banderas).

# Procedimientos y cadenas

Métodos conceptualización:

Secuencia de instrucciones encapsuladas que cambian el flujo de control de ejecución del programa para posteriormente retornar al punto inmediato siguiente de donde fue llamado.

Los métodos:

Permiten la reutilización de código. Reducen la cantidad de código. Permiten la organización y modularizacion del programa. Simplifican el mantenimiento del código.

Tipos de procedimiento:

NEAR: es el procedimiento que esta dentro del mismo segmento de código donde está la llamada.

FAR: el procedimiento y la llamada no están en el mismo segmento de código.

Procedimientos definición:

SINTAXIS:

DEFINICION:

<id\_proc>proc near|far

<secuencia de instrucciones>

Ret

<id\_proc>endp

Uso:

Call <id\_proc>

EJEMPLO:

PROC\_EJ proc near

MOV AX,BX

RET

PROC\_EJ ENDP

CALL PROC\_EJ

Procedimientos: palabras reservadas

PROC: palabra reservada para definición de procedimientos.

ENDP: palabra reservada para designar el fin de la definición del procedimiento.

RET: palabra reservada para indicar al ensamblador el momento de salto al punto de llamado al procedimiento.

IRET: Retorno de interrupción. Restaura los indicadores del stack.

CALL: palabra reservada para hacer la llamada a los procedimientos.

Cadenas:

LEA: transfiere la dirección efectiva es decir el desplazamiento del operando fuente al destino.

OFFSET: asigna el desplazamiento de un operando o variable.

Con ambas cargamos las direcciones de memoria:

LEA destino, fuente: el operando fuente debe estar ubicado en memoria y se coloca su desplazamiento en el registro índice o apuntador especificado en destino.

Ejemplo:

LEA DX,TEXTO ;cargamos en DX la dirección efectiva del texto.

Cadenas de caracteres:

Asignar valores:

LEA DX, variable

MOV DX,OFFSET variable

Índices:

[SI]

[DI]

[BP]

[SP]

SINTAXIS:

Definición:

<id\_cad> db <cantidad> dup (<contenido>)

Uso:

[<index>]

Ejemplo:

CAD1 db 5 dup($)

LEA SI , CAD1

MOV AH, ‘A’

MOV [SI], AH

INC SI

MOV AH, ‘B’

MOV [SI],AH

# Ensambladores en la actualidad

En la actualidad

El lenguaje ensamblador y los ensambladores han evolucionado conforme las necesidades y las implementaciones de hardware lo han hecho.

Algunas iniciativas dieron como resultado:

-macro assembler (Microsoft)

-RISC V (libre)

Macro assembler:

Historia

1981 introducido por Microsoft

1981 – 1990 conocidos como MASM.EXE

1991. V6.0 cambia de nombre a “ML.EXE”

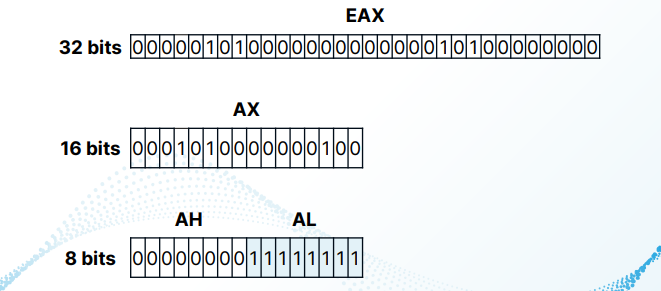
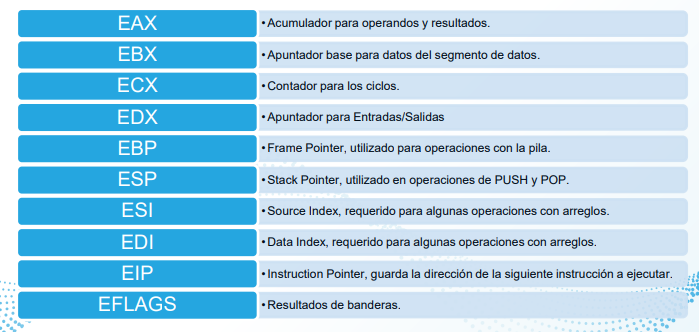
2000. V 7.0 compatibiliza con Microsoft visual C++

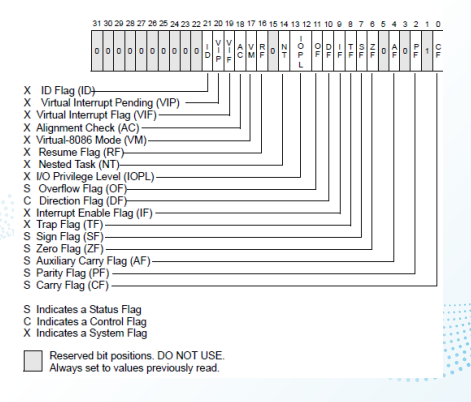
2005. V8.0 comienza el soporte x64

2017. V14.16.27023 versión actual

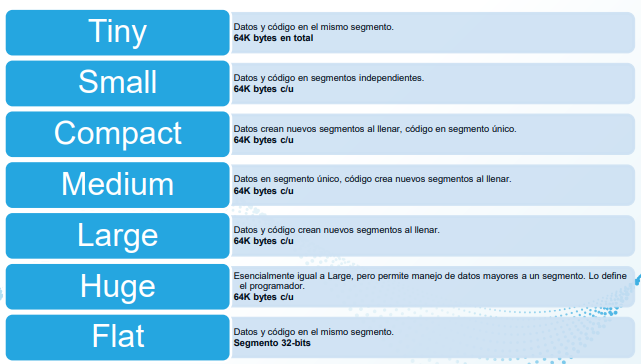
Los procesadores Intel 8086 – 80286 permitían programas con datos 8 – 16 bits (Turbo assembler etc.), el procesador Intel 80386 fue el primer procesador en permitir programas con datos de 8, 16, 32 bits con set de instrucciones x 86 (macro assembler). A partir de Windows NT las palabras convierten su tamaño oficial a 32 bits comienza el set de instrucciones x64.

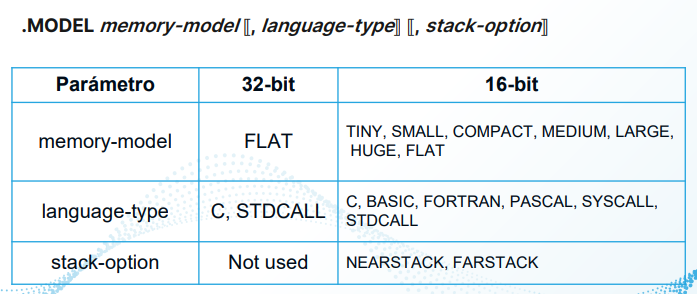
Registros:



Modelos:





Consideraciones:

Debe definirse el procesador con el que se trabajara (.386)

Por defecto el sistema numérico es hexadecimal pero se puede modificar.

La pila (.STACK) por defecto el tamaño es de 1024 bytes pero se puede modificar.

Segmento de datos (.DATA) y datos no inicializados (.DATA?)

RISC V

Historia:

Las ISA existentes eran complejas y con derechos de propiedad intelectual. Partiendo del Intel 8086 el set x86 parte con 80 instrucciones en 1978 para el 2014 alcanzo las 1338 en su versión x86 32 bits.

RISC (reduced intruction set computer) inicio como Proyecto temporal en UC Berkeley.

Surge de las necesidades de una ISA libre y cambiando la tendencia incremental de las ISAs existentes a un modular con un núcleo fundamental y adaptándolo a las necesidades específicas.

El núcleo fundamental del ISA de RISC – V es llamado RV32I

Las extensiones se indican mediante banderas representadas mediante concatenación al núcleo fundamental (ejemplo RV32IMF agrega multiplicación RV32M y punto flotante precisión simple RV32F a las instrucciones base obligatorias RV32I).

# Macros

Conceptualización:

Colección de instrucciones que se repiten frecuentemente durante la ejecución de un programa escrito en lenguaje ensamblador

Características:

Al referenciar una macro en tiempo de ensamblaje se sustituye por el conjunto de instrucciones que representa.

Siempre deben de estar definidas previamente a ser referenciadas.

Manejan etiquetas locales y parámetros.

Sintaxis:

<identificador>MACRO<parámetros>

(LOCAL <etiqueta>)?

<instrucciones>

ENDM

Parámetros:

Sumar MACRO a,b,total

PUSH AX

MOV AX,a

ADD AX,b

MOV total ,AX

POP AX

ENDM

Etiquetas:

Dividir MACRO a,b,cociente,resto

LOCAL FIN

PUSH AX

CMP b,0h

JE FIN

MOV AX,a

DIV b

MOV cociente,AL

MOV resto,AH

FIN:

POP AX

ENDM

Ventajas de macros:

Reducen en forma lógica la cantidad de código.

No utilizan saltos para su ejecución.

Permiten el manejo de “parámetros”.

Son reutilizables en “librerías”.

