CONTENIDO PARCIAL 2 MICRO

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”

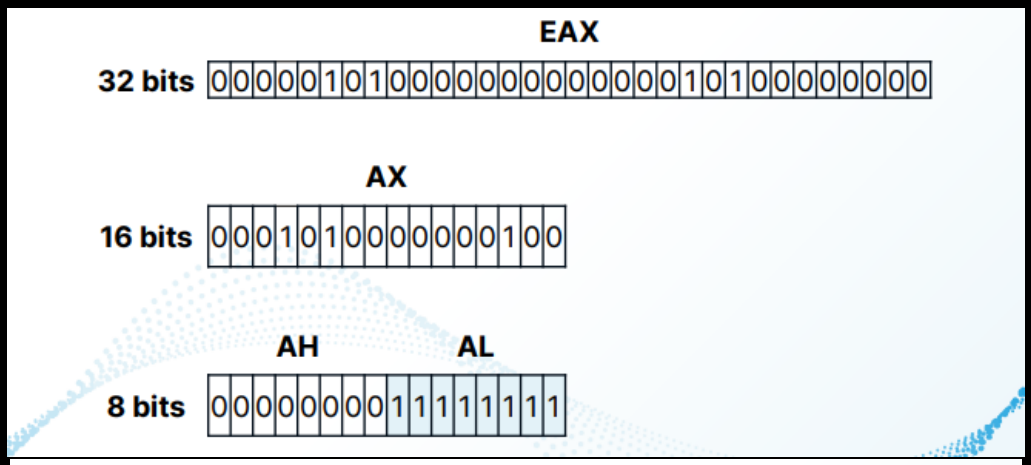
2000. V 7.0 compatibiliza con Microsoft visual C++

2005. V8.0 comienza el soporte x64

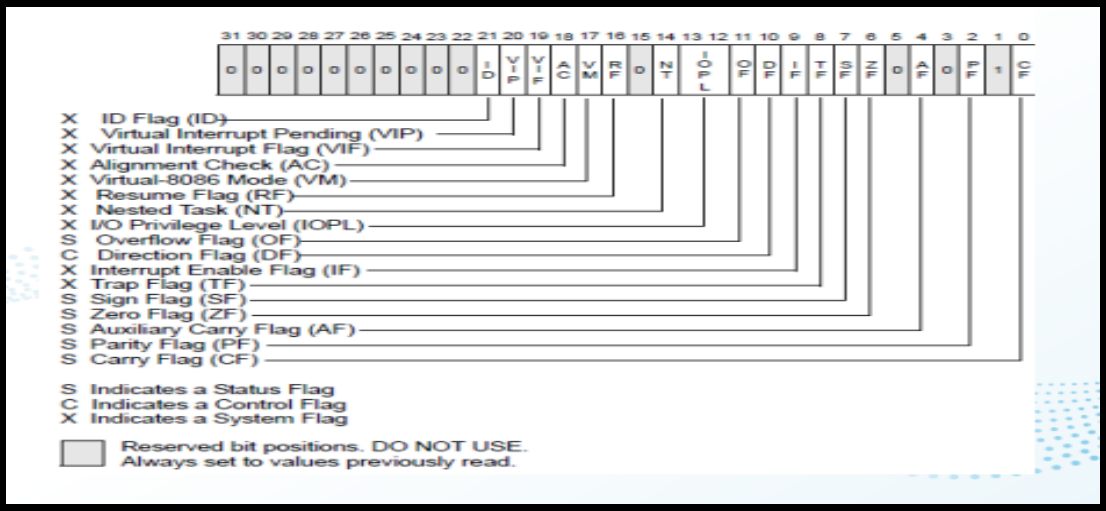
2017. V14.16.27023 versión actual

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.

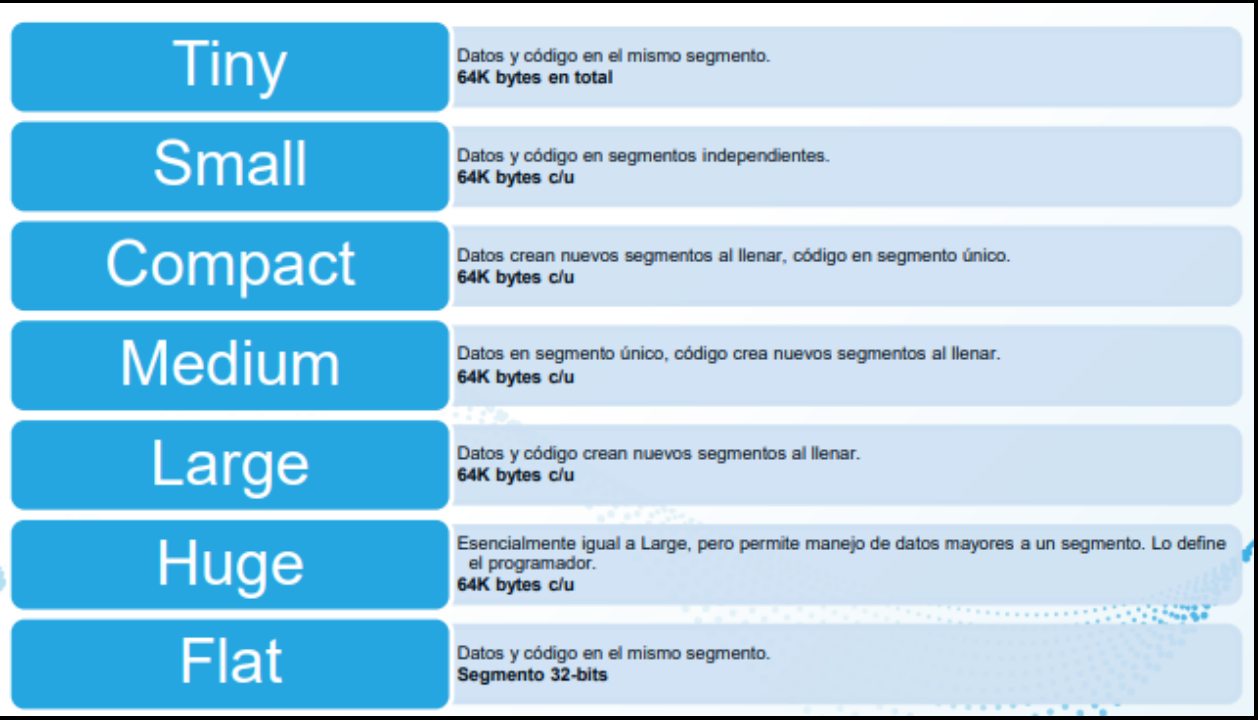
Registros:



EAX	• Acumulador para operandos y resultados.
EBX	• Apuntador base para datos del segmento de datos.
ECX	• Contador para los ciclos.
EDX	• Apuntador para Entradas/Salidas
EBP	• Frame Pointer, utilizado para operaciones con la pila.
ESP	• Stack Pointer, utilizado en operaciones de PUSH y POP.
ESI	• Source Index, requerido para algunas operaciones con arreglos.
EDI	• Data Index, requerido para algunas operaciones con arreglos.
EIP	• Instruction Pointer, guarda la dirección de la siguiente instrucción a ejecutar.
EFLAGS	• Resultados de banderas.



Modelos:



.MODEL *memory-model* [[, *language-type*]] [[, *stack-option*]]

Parámetro	32-bit	16-bit
memory-model	FLAT	TINY, SMALL, COMPACT, MEDIUM, LARGE, HUGE, FLAT
language-type	C, STDCALL	C, BASIC, FORTRAN, PASCAL, SYSCALL, STDCALL
stack-option	Not used	NEARSTACK, FARSTACK

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”.

PARCIAL 2

1. En lenguaje Ensamblador para el modelo de procesador intel 8086, el error "jump destination too far

by XX bytes" puede solventarse con:

-Reestructuración del código fuente

-Utilizando saltos intermedios

2. Tanto los macros como los procedimientos deben definirse previos a ser referenciados

-Falso

3. En la memoria en sus primeros 1024 bytes (000h - 3FFh), ¿Qué se almacena?

-Vector de interrupciones

4. A un arreglo multidimensional se le conoce como Matriz

-verdadero

5. La Arquitectura Clásica del Computador enfoca sus características en el procesamiento, almacenamiento

y transferencia de datos

-verdadero

6. La diferencia entre un Procedimiento y un Macro es que el primero copia la porción de cóidgo y se continua

la ejecución secuencia, mientras que el segundo modifica el IP para ejecutar la posición de código específico

y luego retomar el punto en que se interrumpio la secuencia.

-Falso

7. La portabilidad es una ventaja del lenguaje ensamblador

-Falso

8. Empareje cada palabra reservada, en Lenguaje Ensambaldor, con su función respectiva:

-Utilizada para hacer llamada a procedimientos definidos	CALL
-Utilizada para declarar el fin de la definición de un programa	END
-Utilizada para indicar, al ensamblador, el momento de salto al punto donde fue llamado del procedimiento y continuar con la ejecución secuencial	RET
-Utilizada para indicar, al ensamblador, el momento de salto al punto de ejecución de interrupción y continuar con la ejecución secuencial	IRET
-Utilizada para la definición de procedimientos	PROC
-Utilizada para declarar el fin de la definición de un proce-dimiento.	ENDP

9. A la agrupación de instrucciones que se repiten constantemente a lo largo de un programa, en Lenguaje

de Maquina, se le denomina "Macros"

-Falso

10. Los Registros trabajan directamente con la CU y se mueven a la velocidad del ALU, teniendo capaci-

dades de almacenamiento permanente

-Falso

11. Cada posición de una cadena de caracteres, equivale a una posición en un vector de bajo nivel

y no un espacio reservado de memoria

-Falso

12. La arquitectura Harvard se le conoce como la arquitectura Clásica de Computador.

-Falso

13. En Lenguaje Ensamblador, podemos definir parámetros en los Procedimientos, pero para los Macros no,

con los últimos se debe modificar previamente el entorno.

-Falso

14. El instructor Pointer no es visible al usuario, pero si modificable por el mismo, siendo un registro

de control y estado

-Verdadero

15. Cada segmento definido es de 64 bytes de memoria

-Falso

16. La diferencia entre un Macro y un procedimiento es que el primer copia la porción de código y se

continua la ejecución secuencia, mientras que el segundo modifica el IP para ejecutar la porción

de código específico y luego retornar al punto en que se interrumpió la secuencia

-Verdadero

17. Ordene secuencialmente los eventos que se dan al ejecutar una interrupción en Lenguaje Ensamblador

-MOver la dirección del ISR, que corresponde a la interrupción,

al IP — Tercer evento

-Ejecutar instrucciones del ISR hasta encontrar la palabra

reservada "IRET" — Cuarto evento

-Finalizar la instrucción previa al detectar la interrupción — Primer evento

-Sacar de Pila el CS, IP y Banderas — Quinto evento

-Insertar en Pila el CS, IP y Banderas — Segundo evento

18. El ensamblador traduce lo programado en Lenguaje de Maquina a Lenguaje Ensamblador

-Falso

19. Un nemónico es una cadena de "0", "1"

-Falso

20. El instruction Register contiene la siguiente instrucción a ejecutar

-Falso

21. Los lenguajes más cercanos a la arquitectura de hardware se denominan lenguajes de alto nive,

debido a su nivel de complejidad

-Falso

22. Las computadoras únicamente pueden interpretar el paso o interrupción de corriente eléctrica por lo cual, el sistema hexadecimal se ajusta a las necesidades de la programación

-Falso

23. Tomando la formula de Mapeo Lexicográfico.

Dirección efectiva = Dirección de Memoria + (fila i * tamaño de un elemento) + (columa j * número de elementos por columna * tamaño de elemento)

El almacenamiento en memoria, de una Matriz bidemensional M[i,j], se dio por:

-Columnas

24. La simplicidad de programación es una caracteristica del Lenguaje Ensamblador

-Falso

25. El lenguaje de Maquina evolucióno en el Lenguaje Ensamblador

-Verdadero

26. Empareje cada concepto con el elemento básico correspondiente a un Programa en Lenguaje Ensambaldor

-Dan nicio a los segmentos, indican al Ensamblador la estructura del programa	Directivas
-Indican los tamaños del programa, es decir, la cantidad máxima de código y datos	Modelos
-Conjuntos de nemónicos e identificadores válidos que ejecutan una acción	Instrucciones
-Nombres que se le dan a los elementos de un programa, indicados por el programados	Identificadores

27. Empareje cada definición con el tipo de interrupcion correspondiente

-Interrupcion prvocada por el ensamblador, emplea las funciones del Sistema Operativo para la manipulación de Hardware — Interrupción de DOS de software

-Interrupción generada por dispositivos periféricos — Interrupción Externa de Hardware

-Interrupción generada por eventos ocurridos durante la ejecución del programa, manejados completamente por el hardware — Interrupción interna de Hardware

-Interrupción provocada por el ensamblador, contiene las rutinas de I/O y tablas que indican los estados de los dispositivos del sistema — Interrupción de BIOS de Software

28. El caché y la Memoria Principa(RAM) son los elementos mas veloces de la computadora

-Falso

29. A la colección de instrucciones que le dan tratamiento a una interrupción, se le conoce como

-ISR

30. Un método lleva un salto en forma implicita, indicando el retorno al punto de salto inicial por medio de la palabra reservada RET

-Verdadero

31. Tomando la formula de Mapeo Lexicográfico.

Dirección efectiva = Dirección de Memoria + (fila i * tamaño de elementos por fila * tamaño de elementos) + (columa j * tamaño de elemento)

El almacenamiento en memoria, de una Matriz bidemensional M[i,j], se dio por:

-Filas

32. A la colección ordenada de datos de cualquier tipo, que poseen el mismo formato en sus datos consecutivos se le conoce como Arreglo

-Falso

33. Lo programado en Lenguaje Ensamblador se traduce en Lenguaje de Máquina por medio de un Ensamblador

-Verdadero

34. Selecciones algunas de las ventajas que proporcionan los Métodos al utilizarse en la programación en Lenguaje Ensamblador

-Reducen la cantidad de código

-Permiten la reutilización de código

-Permiten la organización y modlarización del programa

-Simplifican el mantenimiento de codigo

35. A la obtención de elementos en un arreglo multidimensional por medio de una fórmula matemática definida se le conoce como Mapeo Lexicográfico

-Verdadero

36. El ensamblador traduce lo programado en Lenguaje de Máquina a Lenguaje Ensamblador

-Falso

37. Al referenciar un Macro, en tiempo de ensamblaje, se sustituye la llamada por conjunto de instrucciones que contiene

-Verdadero

38. Elija los componentes de la unidad central de proceso

-Registros

-Unidad Aritmético-Lógica

Unidad de control

39. Empareje cada especificacipon con el Registro correspondiente

-Usualmente conserva la base de los datos que hay en la memoria	Registro Base(BX, EBX)
-Usualmente conserva el resultado temporal después de una operación aritmética o lógica	Registro Acumulador(AX, EAX)
-Usualmente contiene el conteo de ciertas instrucciones para corri-mientos y rotaciones del número de bytes o contador LOOP	Registro Contador(CX, ECX)
-Usulamente contiene la parte mas significativa del producto luego de una multiplicación o del di-vidento antes de una división; de uso general	Registro de Datos(DX, EDX)

40. Cada posición de una cadena de caracteres, equívale a una posición en un vector de bajo nivel y no un espacio reservado en memoria

-Falso