Introducción Al Lenguaje Ensamblador

M. en C. Karina Y. Sosa González

18/02/2015

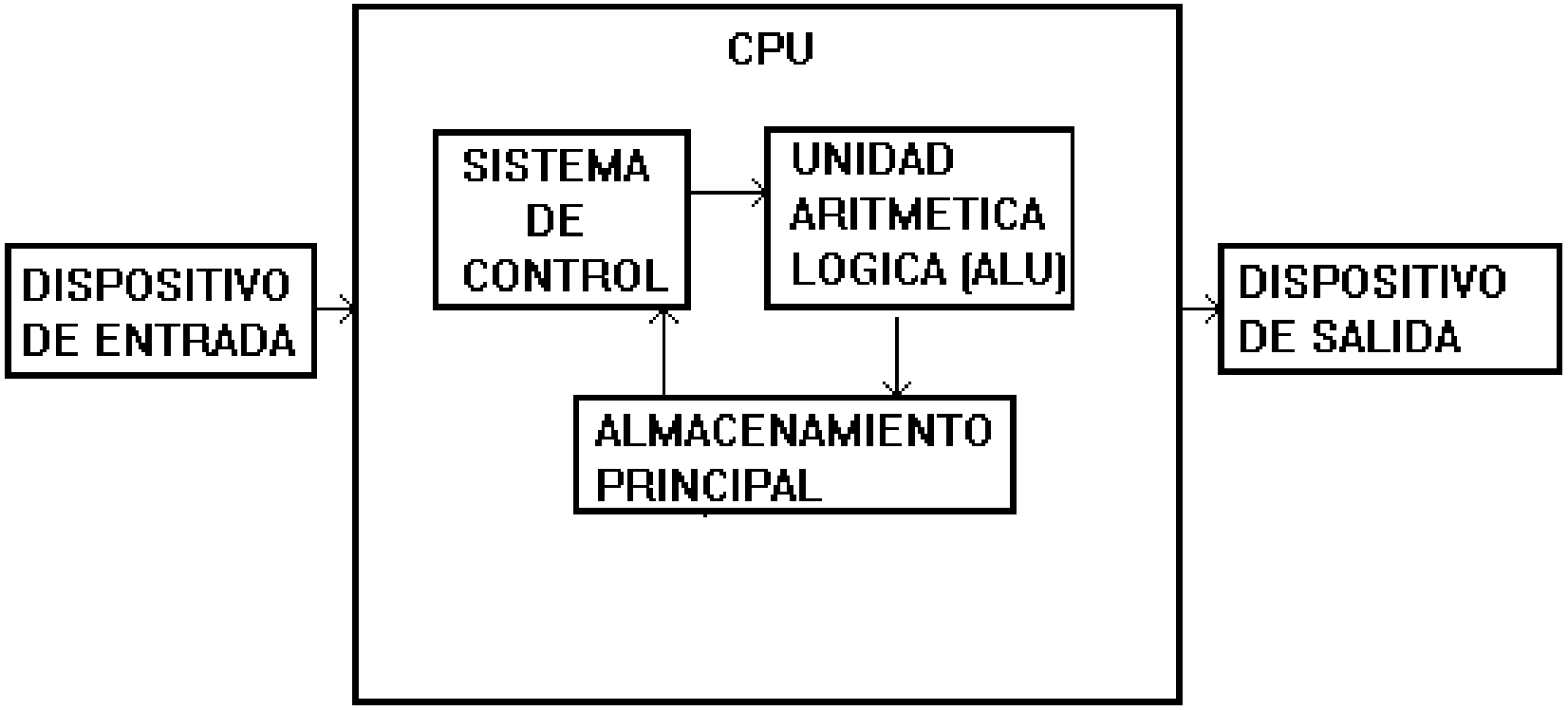
UNIDAD 2

AGENDA

2. Organización de la unidad central de procesamiento

a) organización de la CPU
b) Conjunto y formato de instrucción
c) Lenguaje ensamblador
d) Tipos de instrucciones y modos de direccionamiento
e) Ciclos de instrucción: búsqueda y ejecución

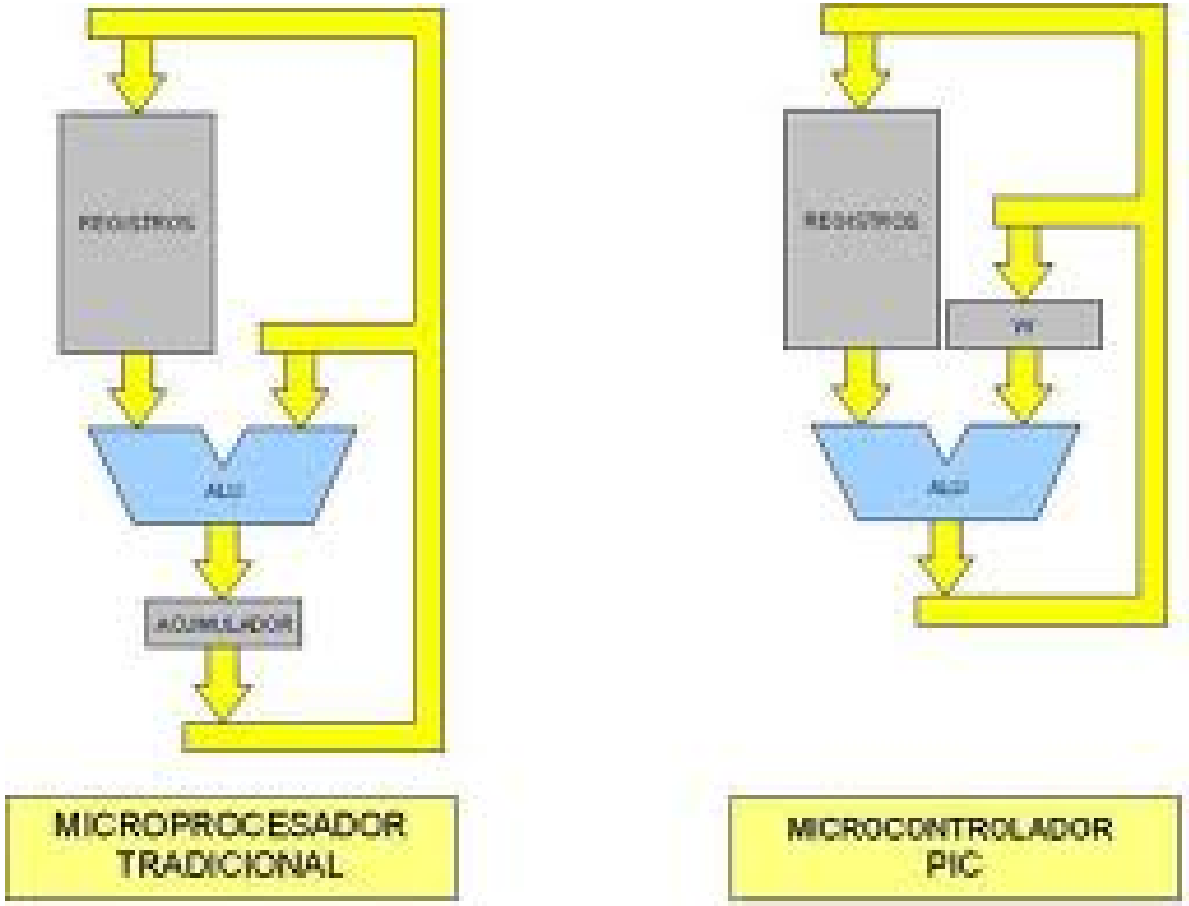
Organización de la CPU



Organización de la CPU (ALU)

- ALU = Unidad Aritmético – Lógica
- Realiza cálculos (aritméticos y lógicos)
- Utiliza Banderas
- Unidad de control, registros y E/S llevan datos

Organización de la CPU (ALU)

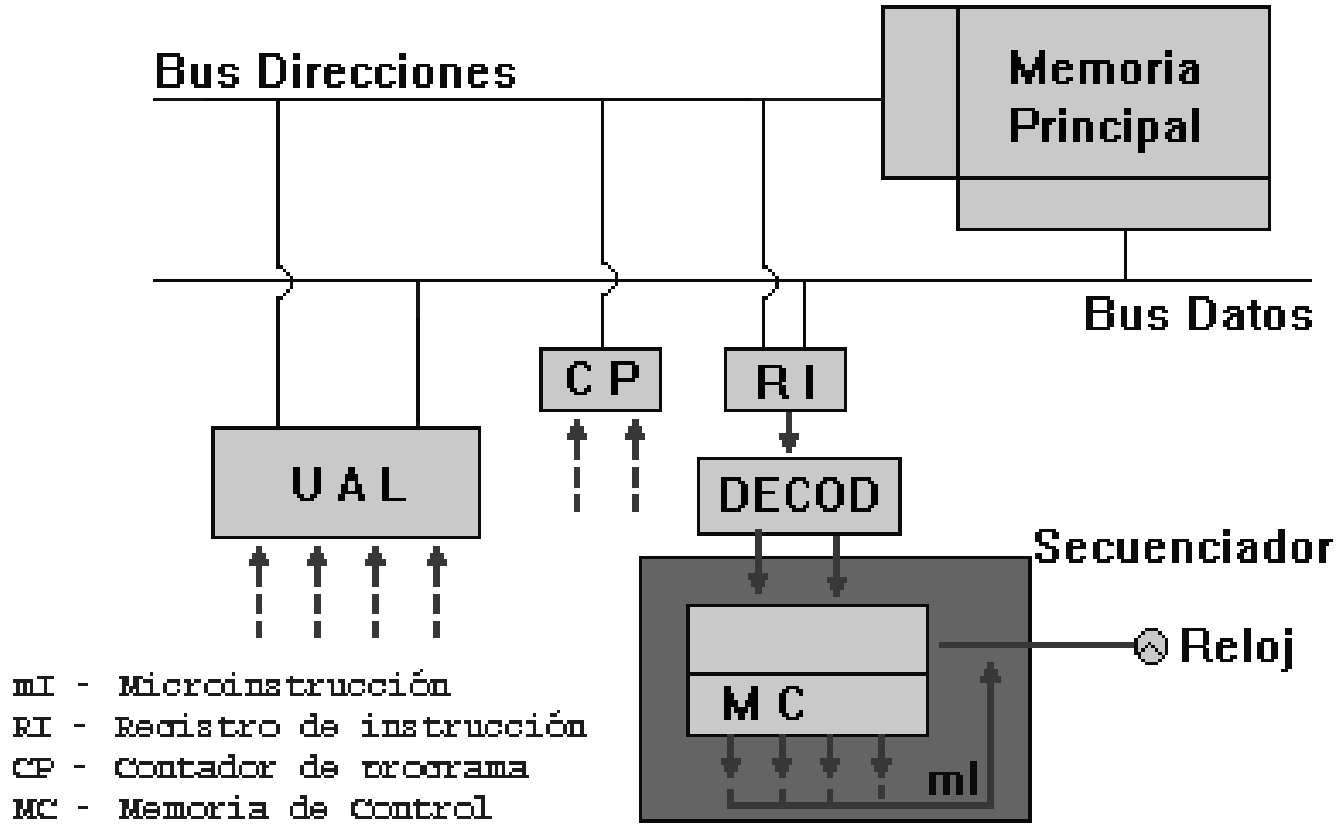


M. en C. Karina Y. Sosa González

Organización de la CPU (CU)

- CU = Unidad de control
- Ejecución de una instrucción = conjunto de ciclos
  - Captación (fetch) : obtener instrucción de la memoria
  - Indirecto : obtener operados
  - Ejecución
  - Interrupción : instrucciones que requieren atención urgente
- Ciclos compuestos de microoperaciones
- Utiliza los registros de control y estado

Organización de la CPU (CU)



Organización de la CPU (Almacenamiento)

- Registros = Almacenamiento temporal interno del CPU
  - Acceso a alta velocidad
  - Localización de la ultima instrucción
  - Almacenar instrucciones y datos mientras se ejecuta alguna instrucción
  - Dos tipos (disponibles al usuario y de control o estado)
  - Los de control (PC, IR, MAR, MDR, PSW)

- Memorias externas de distintos tipos

Organización de la CPU ¿Qué es o para que sirve?

- Función: ejecutar secuencias de instrucciones.
  - Buscar instrucciones (memoria)
  - Interpretar instrucciones
  - Buscar datos (memoria o E/S)
  - Procesar datos
  - Escribir datos (memoria o E/S)
- Programación computadora.
  - Preparar secuencia de instrucciones (conjunto de instrucciones)
  - Almacenar datos (memoria)
  - Instruir al CPU para que inicie la ejecución

Tipos de Instrucciones

- Lenguaje Maquina
- Lenguaje Ensamblador
- Lenguajes Estructurados
- Lenguajes de Programación Orientados a objetos
- Lenguajes de Programación Orientados a Aspectos

Tipos de Instrucciones (Lenguaje Maquina)

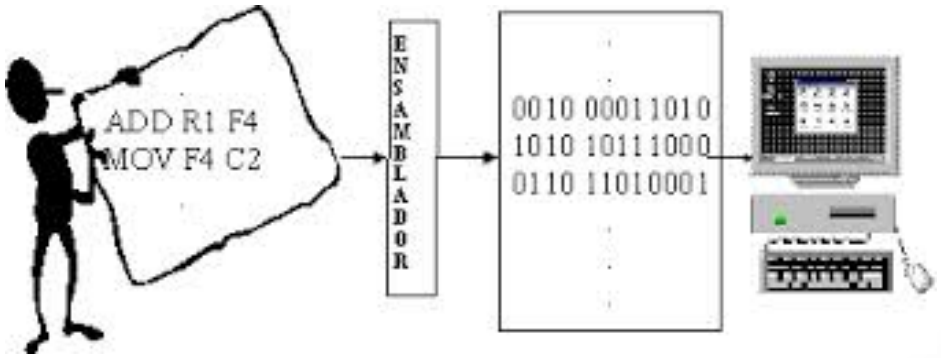
- Es aquel escrito directamente para que una computadora lo entienda.
- Son cadenas binarias (series de 1 y 0) que especifican operaciones y posiciones de memoria implicadas en la operación.
- También es conocido como código binario
- Dependen del hardware, por tanto, difieren de una computadora a otra.

Tipos de Instrucciones (Lenguaje Maquina)

Ventajas	Desventajas
Transfiere un programa a la memoria sin necesidad de interprete Velocidad de ejecución	Dificultad y lentitud en la codificación Poca fiabilidad Gran dificultad para verificar y poner a punto los programas Solo ejecutables en el mismo CPU

Tipos de Instrucciones (Lenguaje Ensamblador)

- También conocido como bajo nivel
- Son mas fáciles que los lenguajes maquina, pero dependen de la maquina en particular
- Se usan nemotécnicos
- Requiere de fase de traducción al lenguaje maquina
  - Ensamblador = programa fuente
  - Programa traducido = programa objeto

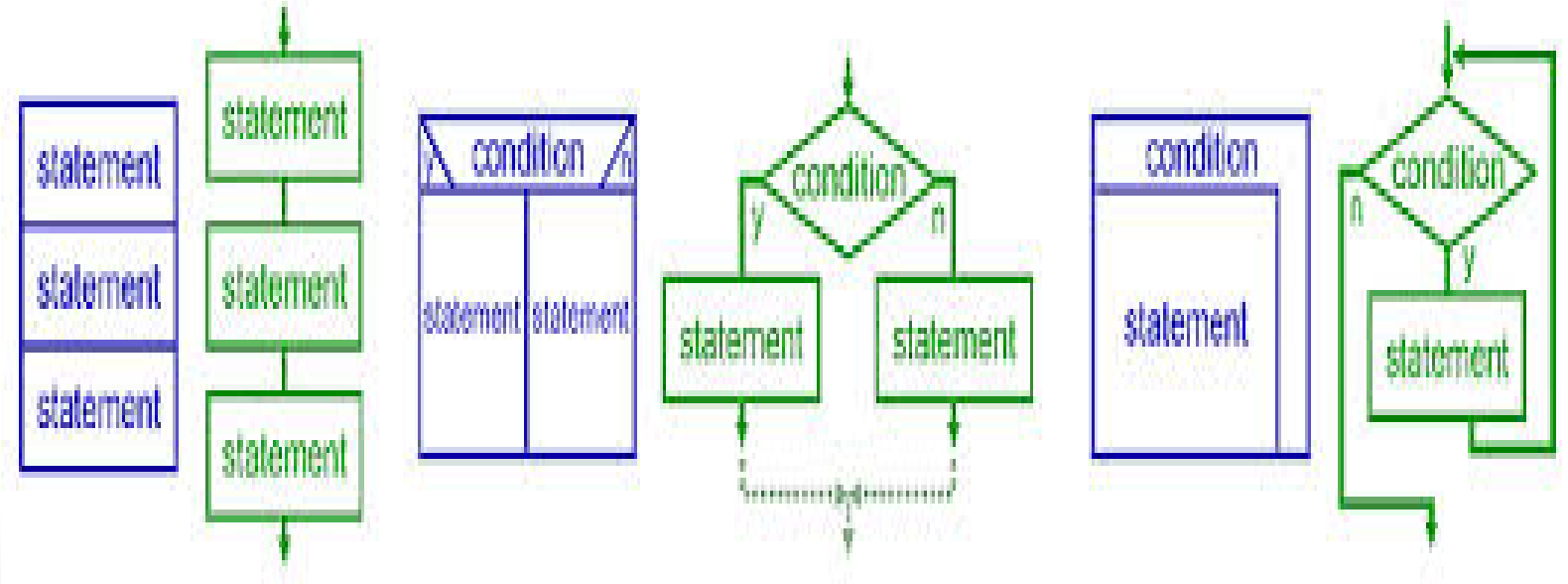


Tipos de Instrucciones (Lenguaje Ensamblador)

Ventajas	Desventajas
Mayor facilidad de codificación Velocidad de calculo rápida Aplicaciones reducidas = tiempo real, control de proceso y dispositivos electrónicos	Dependencia total del hw No es transportable La formación del programador es mas completa = requiere conocimiento de la maquina

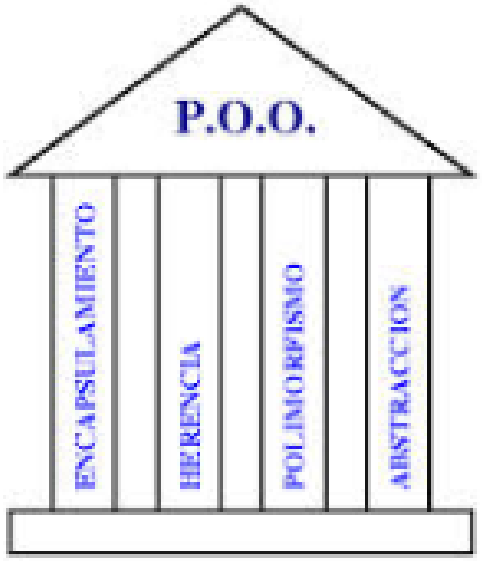
Tipos de Instrucciones (Lenguaje Estructurado)

- Programas fáciles
- Es independiente del HW
- Portables o transportables de PC a PC



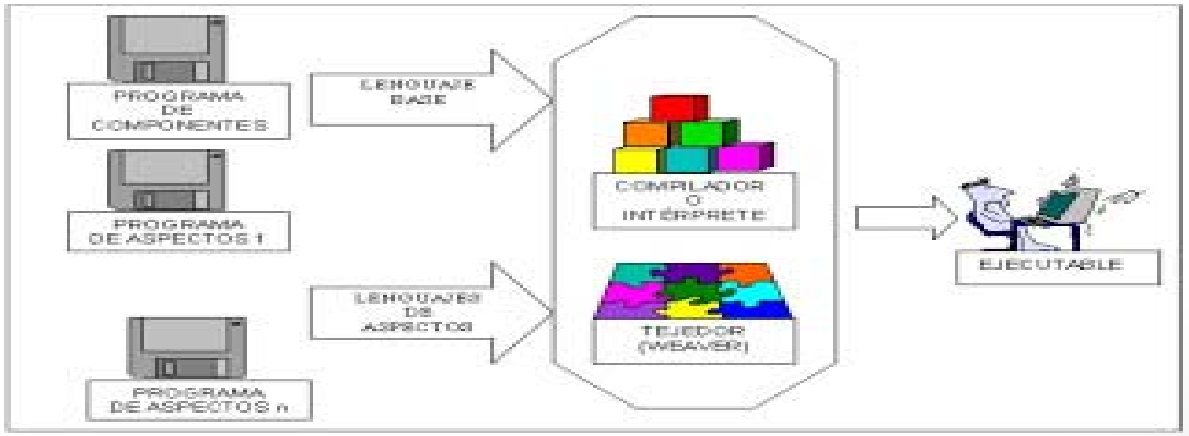
Tipos de Instrucciones (Lenguaje Orientado a Objetos)

- Programas armados de manera mas fácil
- Es independiente del HW (portable)
- Se otorgan estructuras muy usadas para que el usuario las tome y les incorpore la utilidad que desea
- Diversos aspectos



Tipos de Instrucciones (Lenguaje Orientado a Aspectos)

- Unidad modular que se disemina por las estructuras de unidades funcionales
- Separa aspectos que se encuentran en varios módulos del sistema y los extrae de forma independiente para hacer cambios específicos



Repertorio de Operaciones

- Transferencia de datos
- Aritméticas
- Manipulación de bits (lógica)
- Control de programa
- Manipulación de cadenas
- Control de CPU

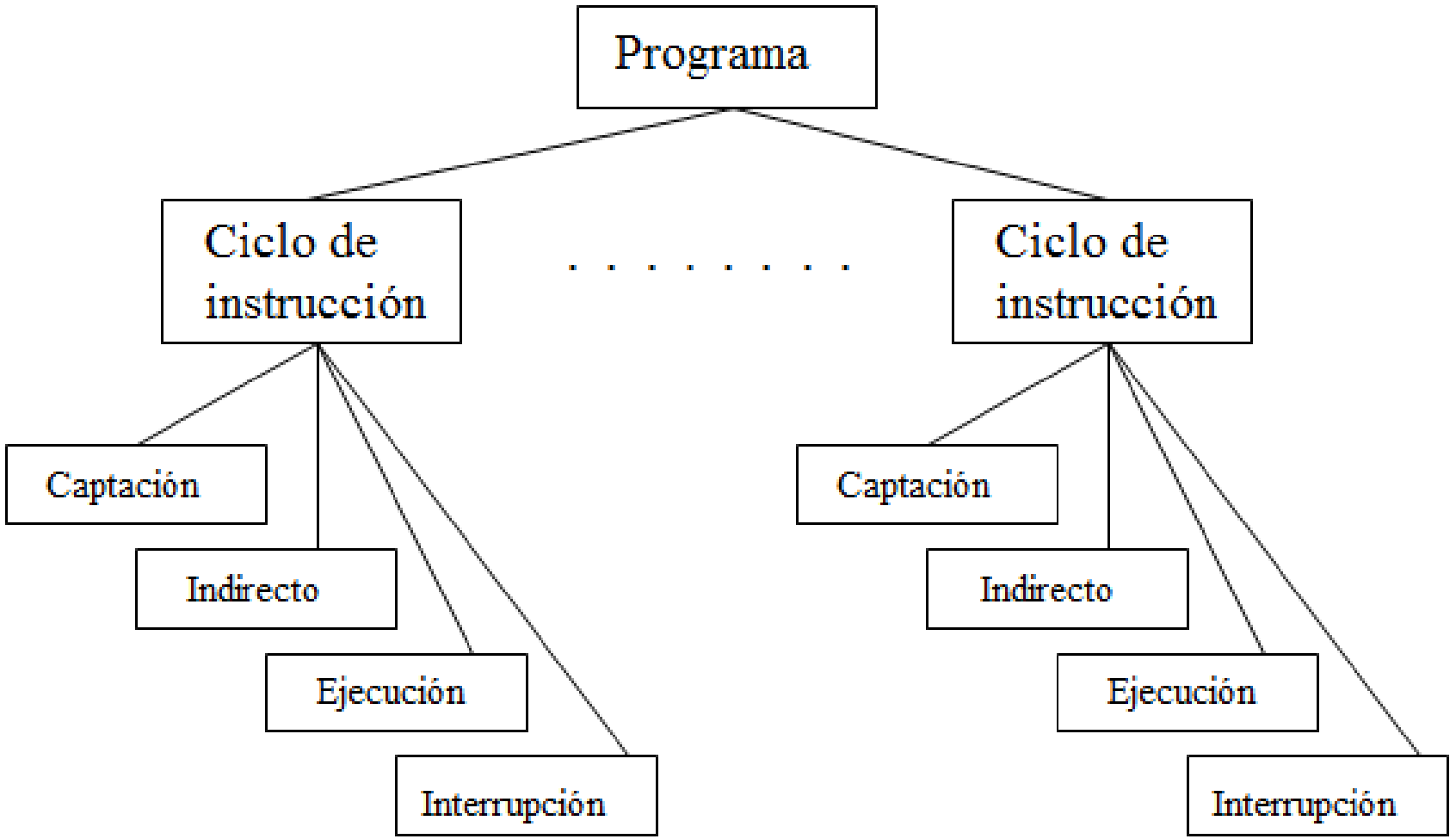
UNIDAD 2

Ciclos de instrucciones

- La función de una maquina es ejecutar programas
- La ejecución de un programa consiste en ejecutar una secuencia de instrucciones maquina
- Cada instrucción esta a su vez compuesta de un conjunto de ciclos llamados CICLOS DE INSTRUCCIONES
- Cada uno de los ciclos de instrucciones esta compuesto de una serie de pasos conocidos como micro operaciones

UNIDAD 2

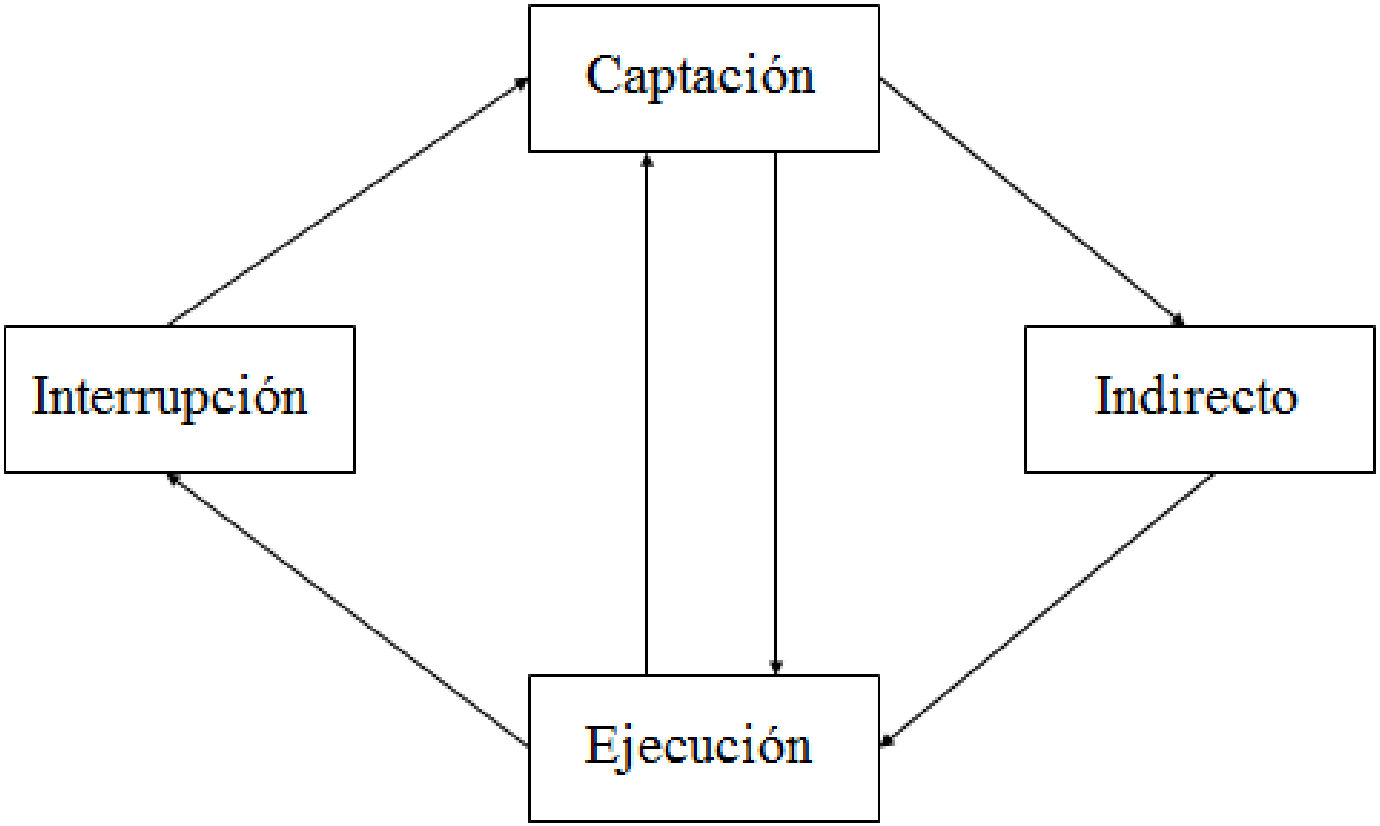
Ciclos de instrucciones



UNIDAD 2

Ciclos de instrucciones

Ciclo de Instrucción



ESPERA!!! Conceptos Importantes:

- Definición de Bit, Byte, Nible y Palabra
- Alfabeto Binario, octal, decimal, hexadecimal
- Representación y conversión de valores en diferentes bases
- Operaciones en complementaos a la base y base disminuida
- Representación de números enteros con signo
- Aritmética de números enteros con signo y BCD
- Representación de números reales

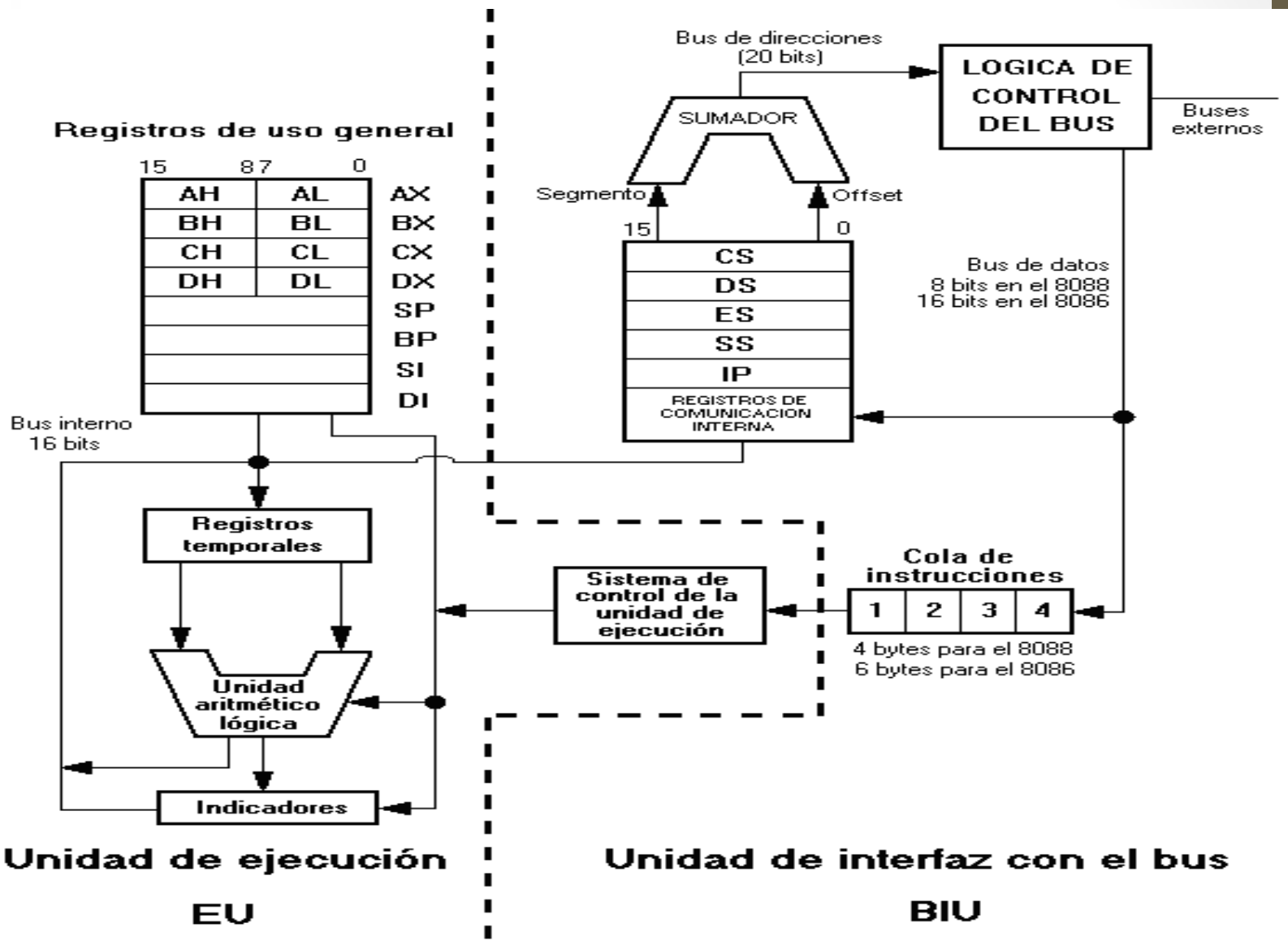
Generalidades

- Para entender la forma en la cual actúa el lenguaje ensamblador, usaremos como ejemplo el Intel8086.
- Características:
  - Estructura general de todos los Intel (Compatibles)
  - Registros generales
  - Registro específicos
  - ALU de 8 bits
  - Tamaño de palabra de 8 bits
  - Direccionamiento de hasta 20 bits en memoria
  - 7 Modos de direccionamiento

UNIDAD 2

Lenguaje Ensamblador

Estructura general



Estructura General

Cuenta con 2 bloques con 4 registros de propósito general:

AX : Acumulador General
BX : Acumulador Base
CX : Acumulador Contador
DX : Acumulador de Datos

SP : Apuntador de Pila
BP : Apuntador Base
SI : Índice Fuente
DI : Índice Destino

Estructura General

Registros de propósito especifico:

CS : Segmento de Código
SS : Segmento de Pila
DS : Segmento de Datos
ES : Segmento Extra
IP : Puntero de Instrucción
FLAGS : Registro de Banderas

Registro de indicadores (16 bits)																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Flag	--	--	--	--	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

UNIDAD ARITMETICO LOGICA:

- Se encarga de realizar las operaciones aritméticas básicas (suma, resta, desplazamiento, corrimiento, multiplicación y división) y logics (AND, OR, XOR y TEST).
- Dentro del 8086, se pueden realizar operaciones de 8 y 16 bits.
- Para lo cual los cuatro principales registros de propósito general pueden fragmentarse en parte alta y parte baja (H y L).

SISTEMA DE CONTROL DE LA UNIDAD DE EJECUCION

- Su función es decodificar las instrucciones que le son enviadas directamente de la cola. Una vez que sabe exactamente cual es el significado de la instrucción (por medio de una tabla de búsqueda - Look of Table), envía la orden a la unidad aritmética - lógica.

- La tabla de búsqueda es comúnmente llamada CROM (Control Read Only Memory) o Memoria de Control.

UNIDAD 2

Lenguaje Ensamblador

Dentro de la estructura del ensamblador permite:

- **ETIQUETAS**
- **DEFINICIONES**
- **COMENTARIOS**
- **SENTENCIAS**

UNIDAD 2

Lenguaje Ensamblador

Una sentencia (o instrucción) en ensamblador puede tener la siguiente estructura:

[Nombre] [Operación] [Operandos] [;Comentario]

- **Nombre: Normalmente conocido como etiqueta.**
- **Operación: Indica que acción deberá realizarse sobre los operandos.**
- **Operandos: Uno o mas ítems con los que la instrucción opera**
- **Comentario: Es opcional, no tiene ninguna influencia sobre el código.**

principio: MOV AX, 7 ; movemos el valor 7 al reg AX

UNIDAD 2

Lenguaje Ensamblador
MODOS DE DIRECCIONAMIENTO

INMEDIATO:

El usuario especifica un byte o palabra como operando fuente. Esta constante se ensambla como parte de la instrucción.

Ejemplo:

MOV AX, 18D

Se coloca el número 18 decimal en el registro AX.

UNIDAD 2

Lenguaje Ensamblador
MODOS DE DIRECCIONAMIENTO

DIRECTO:

En este direccionamiento la dirección efectiva está contenida dentro de la instrucción. Es decir, el registro al terminar la instrucción tendrá el valor contenido dentro de la dirección anteriormente mencionada.

MOV AX, [16h]

- **Para calcular la dirección verdadera, suponiendo que DS está en 200h, el cálculo ser realiza de la siguiente manera:**

  DS * 10h + 16h = 200h * 10h + 16h = 2016h

DE REGISTRO:

En esta forma de direccionamiento el operando se obtiene de un registro o se coloca en éste. La instrucción:

MOV AX, BX

El registro AX toma el valor contenido dentro de BX. Es decir, de manera general, se puede transpolar a C donde AX = BX.

RELATIVO A LA BASE:

Al hacer uso de este modo de direccionamiento, la dirección efectiva del operando se obtiene al sumar un desplazamiento a los siguientes registros: BX o BP. En este caso, los registros deben contener la dirección de desplazamiento.

MOV AX, [BX + 2]
MOV AX, [BX][2]

UNIDAD 2

Lenguaje Ensamblador
MODOS DE DIRECCIONAMIENTO

INDEXADO DIRECTO:

La dirección efectiva es la suma del contenido de un registro índice (SI o DI) y un desplazamiento. Un ejemplo común lo constituye una secuencia de instrucciones donde primero se carga una dirección en un registro índice y después la misma se combina con una localidad de memoria.

MOV SI, 2
MOV AX, [16h][SI]

En este caso, en el registro AX se coloca el contenido de la localidad de memoria cuya dirección es la 16h + 2.

UNIDAD 2

Lenguaje Ensamblador
MODOS DE DIRECCIONAMIENTO

INDEXADO DE BASE:

La dirección efectiva es la suma de los contenidos del registro base (BX), un registro índice (SI o DI) y un desplazamiento.

MOV AX, [BX] [SI + 2]
MOV AX, [BX + SI + 2]
MOV AX, [BX + SI]

El desplazamiento puede ser opcional.

UNIDAD 2

Lenguaje Ensamblador
MODOS DE DIRECCIONAMIENTO

Registro	**MOV AX, BX**	Registro BX → Registro AX
Inmediato	**MOV AX, 18D**	Dato 3Ah → Registro CH
Directo	**MOV AX, [16h]**	Registro AX → 10000h + 1234h → Direccion de Memoria 11234h
Registro indirecto	**MOV AX, [BX]**	Registro CL → 10000h + 0300h → Direccion de Memoria 10300h
Relativo a la base	**MOV AX, [BX + 2]**	Registro SP → 10000h + 0300h+0200h → Direccion de Memoria 10500h
Indexado Directo	**MOV AX, [16h][SI]**	Direccion de Memoria 10304h → 10000h + 0300h+4 → Registro CL
Indexado de Base	**MOV AX, [BX] [SI + 2]**	Registro AX → 10000h +1000h 0300h+0200h → Direccion de Memoria 10700h

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Este tipo de instrucciones son útiles para manipular los datos que se encuentran contenidos dentro de los registros del microprocesador, las memorias externas o inclusive, datos que nosotros deseemos ingresar al programa. El listado completo es el siguiente:

1. **Transferencia de datos**
2. **Carga**
3. **Pila**
4. **Almacenamiento**
5. **Manejo de cadenas**

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Transferencia

MOV
MOVSB
MOVSW

Almacenamiento

STOSB
STOSW
SAHF

Carga

LODSB
LODSW
LAHF
LDS
LEA
LES

Pila

POP
POPF
PUSH
PUSHF

Manejo De cadenas

REP
REPE
REPZ
REPNE
REPNZ
SCASB
SCASW

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Mueve uno o dos bytes (dependiendo del destino desde fuente)

MOV x, y

Direccionamiento:

Registro, [localidad]
[localidad], registro
Registro , registro
[localidad], dato
Registro, dato

Explicación:

Esta instrucción es útil para el movimiento de uno o dos bytes cuando se trata de localidades de memoria, datos e inclusive registros.

Mueve una cadena de longitud un byte

MOVSB

Explicación:

Transfiere un byte de la localidad del segmento de datos direccionada por SI a la localidad del segmento adicional direccionada por DI. Los apuntadores aumentan o disminuyen dependiendo de la bandera de dirección. Si D = 0 incrementa y D = 1 decremento.

Mueve una cadena de longitud una palabra

MOVSW

Explicación:

Transfiere una palabra de la localidad del segmento de datos direccionada por SI a la localidad del segmento adicional direccionada por DI. Los apuntadores aumentan o disminuyen dependiendo de la bandera de dirección. Si D = 0 incrementa y D = 1 decremento.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Como ayuda se tienen las siguientes instrucciones que manipulan algunas banderas del registro FLAGS

CLC	**-**	**Limpia bandera de acarreo (CF = 0)**
CLD	**-**	**Limpia bandera de dirección (DF = 0)**
CLI	**-**	**Limpia bandera de interrupción (IF = 0)**
CMC	**-**	**Complementa bandera de acarreo**
STC	**-**	**Activa bandera de acarreo (CF = 1)**
STD	**-**	**Activa bandera de dirección (DF = 1)**
STI	**-**	**Activa bandera de interrupción (IF = 1)**

Carga cadenas de un byte

LODSB

Explicación:

Carga en AL un dato almacenado en la dirección de desplazamiento dentro del segmento de datos que indica el registro SI. Después de cargar el dato, el contenido de SI es incrementado o decrementico en uno dependiendo de la bandera DF.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Carga cadenas de un palabra

LODSW

Explicación:

Carga en Ax un dato almacenado en la dirección de desplazamiento dentro del segmento de datos que indica el registro SI. Después de cargar el dato, el contenido de SI es incrementado o decrementado en dos (porque la memoria es de 1 byte) dependiendo de la bandera DF.

Carga el registro de banderas

LAHF

Explicación:

Transfiere los ocho bits del extremo derecho del registro de banderas al registro AH.

Esta instrucción fue diseñada como instrucciones puente, porque permitían traducir el lenguaje para el 8085.

Transfiere una palabra desde la pila

POP y

Direccionamiento

Registro [localidad]

Explicación:

La ultima palabra contenida dentro de la pila es tomada y almacenada en el operando que acompaña la instrucción.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE MOVIMIENTO DE DATOS

Transfiere una palabra a la pila

PUSH y

Direccionamiento

Registro
[localidad]
dato

Explicación:

La ultima palabra contenida dentro del operando que acompaña la instrucción con destino a la pila.

Transfiere al registro FLAGS desde la pila

POPF

Explicación:

La ultima palabra contenida dentro de la pila es tomada y transferida al registro de banderas. Una vez hecha la transferencia se incrementa en 2 el registro SP.

Transfiere el registro FLAGS a la pila

PUSHF

Explicación:

Primero se decrementa en 2 la pila y después se transfiere el registro de banderas (en los bits señalados para la instrucción POPF) a la palabra cuya dirección esta dada por la pila.

Almacena un byte en la localidad del segmento

STOSB

Explicación:

La instrucción almacena el byte de AL en la localidad de memoria del segmento adicional direccionada por DI.
Después de almacenar el byte el contenido de DI se incrementa o decrementa dependiendo de la bandera DF.

Almacena una palabra en la localidad del segmento

STOSW

Explicación:

La instrucción almacena la palabra de Ax en la localidad de memoria del segmento adicional direccionada por DI y en DI +/- 1.

Después de almacenar la palabra el contenido de DI se incrementa o decrementa dependiendo de la bandera DF en 2.

Almacena el registro de banderas

SAHF

Explicación:

Trasfiere los bits 7, 6, 4, 2 y 0 del registro AH al registro de banderas. Estos bits corresponden a las banderas SF, ZF, AF, PF, CF.

Repite la operación cadena

REP instrucción

Explicación:

Esta instrucción causa que la operación cadena se repita mientras Cx sea diferente de 0. Cuando se inicia la repetición, se prueba el estado de la bandera ZF y si este cambia después de la ejecución de instrucción CMPS y SCAS, la repetición termina.

Busca cadenas de byte

SCASB

Explicación:

Compara el registro AL con un bloque de bytes en memoria. La instrucción resta la memoria de AL sin afectar el registro. El código de operación utilizado para la comparación de bytes. El contenido de la localidad de memoria del segmento es direccionado por DI. Recuerde que utiliza DF para incrementar o decrementar.

Busca cadenas de palabra

SCASW

Explicación:

Compara el registro Ax con un bloque de 2 bytes en memoria. La instrucción resta la memoria de Ax sin afectar el registro. El código de operación utilizado para la comparación de 2 bytes. El contenido de la localidad de memoria del segmento es direccionado por DI. Recuerde que utiliza DF para incrementar o decrementar de dos en dos.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación ADC

ADC x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Lleva a cabo la adición de dos operandos y suma un uno al resultado si la bandera CF está activa.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación ADD

ADD x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Lleva a cabo la adición de dos operandos y no considera acarreo alguno al sumar, pero sí en la parte de resultado.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación DIV

DIV y

Direccionamiento:

Registro
[ ]

Explicación:

División sin signo. El numerador con bytes esta en AH y AL, mientras que para palabras está en Ax y Dx. El cociente está en AL o Ax y el residuo en AH o Dx.

Operación IDIV

IDIV y

Direccionamiento:

Registro
[ ]

Explicación:

Lleva a cabo la división con signo. Para lo anterior considera los mismos registros que para la instrucción DIV.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación IMUL

IMUL y

Direccionamiento:

Registro
[ ]

Explicación:

Multiplica con signo AL o Ax a la fuente y regresa el producto en AL y AH si la operación fue sobre 8 bits; ó en Ax y Dx si la operación fue sobre 16 bits.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación MUL

MUL y

Direccionamiento:

Registro
[ ]

Explicación:

Multiplica con signo AL o Ax a la fuente y regresa el producto en AL y AH si la operación fue sobre 8 bits; ó en Ax y Dx si la operación fue sobre 16 bits.

Operación SBB

SBB x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Resta los dos operandos y resta el resultado en uno si CF está activada. El operando fuente siempre se resta del destino.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación SUB

SUB x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Resta el operando fuente del destino.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación DEC

DEC x

Direccionamiento:

Registro
[ ]

Explicación:

Esta instrucción resta en uno el destino y regresa el resultado al mismo operando. Es decir, se lleva a cabo la siguiente operación:

$$X = X - 1$$

Operación INC

INC x

Direccionamiento:

Registro
[ ]

Explicación:

Esta instrucción suma en uno el destino y regresa el resultado al mismo operando.
Es decir, se lleva a cabo la siguiente operación:

$$X = X + 1$$

Operación SAL

SAL x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Desplaza al destino a la izquierda un número determinado de bits, dado por un conteo. Las posiciones vacías se llenan con ceros.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES ARITMETICAS

Operación SHL

SHL x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Desplaza al destino a la izquierda un número determinado de bits, dado por un conteo. Las posiciones vacías se llenan con ceros.

Operación SAR

SAR x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Desplaza al destino a la derecha un número determinado de bits, dado por un conteo. Las posiciones vacías se llenan con ceros. El bit de menor orden reemplaza a la bandera de acarreo, misma que pierde su estado original.

Operación SHR

SHR x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Desplaza al destino a la derecha un número determinado de bits, dado por un conteo. Las posiciones vacías se llenan con ceros. El bit de menor orden reemplaza a la bandera de acarreo, misma que pierde su estado original.

Operación RCL

RCL x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Rota hacia la izquierda el operando destino, un número de bits especificado por conteo. La bandera de acarreo se incluye en esta rotación.

Operación RCR

RCR x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Rota hacia la derecha el operando destino, un número de bits especificado por conteo. La bandera de acarreo se incluye en esta rotación.

Operación ROL

ROL x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Rota hacia la izquierda el operando destino, un número de bits especificado por conteo. El bit más significativo pasa a ocupar el lugar del menos significativo en cada rotación.

Operación ROR

ROR x, c

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Rota hacia la derecha el operando destino, un número de bits especificado por conteo. El bit más significativo pasa a ocupar el lugar del menos significativo en cada rotación.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES LOGICAS

Operación AND

AND x, y

Explicación:

Lleva a cabo la conjunción, bit a bit, de dos operandos considerando su tabla de verdad:

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES LOGICAS

Operación NEG

NEG x

Direccionamiento:

Registro
[ ]

Explicación:

Esta instrucción genera el complemento a dos del operando destino.
Recuerde que el complemento a dos se compone sumando uno al numero destino y después invirtiendo todos los números.

Operación NOT

NOT x

Explicación:

Lleva a cabo la negación, bit a bit, del operando destino considerando su tabla de verdad:

X	Z
0	1
1	0

Direccionamiento:

Registro
[ ]

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES LOGICAS

Operación OR

OR x, y

Explicación:

Efectúa, bit a bit, la disyunción inclusiva lógica de dos operandos:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Operación TEST

TEST x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Compara dos números, bit a bit; es decir realiza la conjunción pero no devuelve resultado de la operación, solo tiene efecto sobre el estado de las banderas.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES LOGICAS

Operación XOR

XOR x, y

Explicación:

Lleva a cabo la disyunción exclusiva, bit a bit, de dos operandos (tabla de verdad):

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES LOGICAS

Operación CMP

CMP x, y

Direccionamiento:

Registro, [ ]
[ ], registro
Registro , registro
[ ], dato
Registro, dato

Explicación:

Si el resultado de la operación es ZF = 1, entonces quiere decir que la comparación es valida. El resultado no se guarda.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

La forma de utilización de cualquier condición de salto es:

JCC etiqueta-corta

Lo que significa que antes de la comprobación de banderas debe existir una operación aritmética o lógica que afecte de manera directa el estado del registro de las banderas. Una vez hecho lo anterior, entonces procedemos a comprobar la bandera seleccionada y realizar (o no) el salto condicionado.

SALTO INCONDICIONAL

JMP etiqueta-corta

Explicación:

No importa el estado de las banderas del registro FLAGS, se realiza el salto de manera directa a la etiqueta corta.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Condición
JC	Salta si carry	CF = 1
JNC	Salta si no carry	CF = 0
JZ	Salta si cero	ZF = 1
JNZ	Salta si no cero	ZF = 0
JS	Salta si signo	SF = 1
JNS	Salta si no signo	SF = 0

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Condición
JO	Salta si overflow	OF = 1
JNO	Salta si no overflow	OF = 0
JP	Salta si paridad	PF = 1
JPE	Salta si paridad de 1's	PF = 1 (1's)
JNP	Salta si no paridad	PF = 0
JPO	Salta si paridad de 0's	PF = 0 (0's)

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Cond.	FLAGS
JA	Salta si mayor	A > B	/C /Z
JNBE	Salta si no menor/igual	A > B	/C /Z
JAE	Salta si mayor o igual	A >= B	/C
JNB	Salta si no menor	A >= B	/C
JE	Salta si igual	A == B	Z
JNE	Salta si no igual a cero	A != B	/Z

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Cond.	FLAGS
JBE	Salta si menor o igual	A <= B	C ó Z
JNA	Salta si no mayor	A <= B	C ó Z
JB	Salta si menor	A < B	C
JNAE	Salta si no mayor o igual	A < B	C

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Cond.	FLAGS
JG	Salta si mayor	A > B	/S ó Z
JNLE	Salta si no menor/igual	A > B	/S ó Z
JGE	Salta si mayor o igual	A >= B	/S
JNL	Salta si no menor	A >= B	/S
JE	Salta si igual	A == B	Z
JNE	Salta si no igual	A != B	/Z

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES DE CONTROL

JCC	Descripción	Cond.	FLAGS
JLE	Salta si menor o igual	A <= B	S O ó Z
JNG	Salta si no mayor	A <= B	S O ó Z
JL	Salta si menor	A < B	S O
JNGE	Salta si no mayor o igual	A < B	S O

CICLO HASTA QUE TERMINA EL CONTEO

LOOP etiq-corta

Explicación:

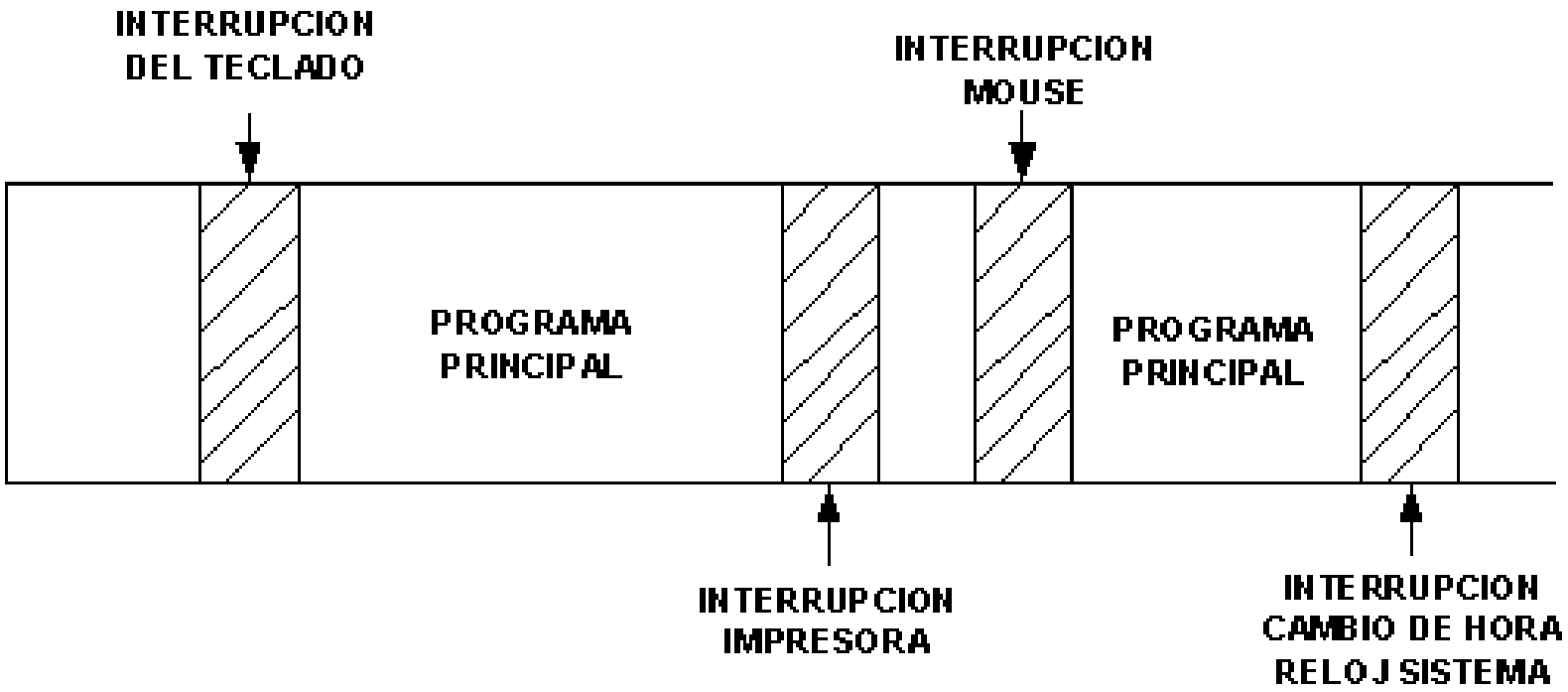
La instrucción decrementa el registro Cx en 1, transfiere el control del programa a la etiqueta corta si Cx ≠ 0 (o lo que es lo mismo, salta mientras que el registro Cx no sea igual a cero).

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES INTERRUPCIONES

Una interrupción es una instrucción que detiene la ejecución de un programa y coloca el apuntador de instrucciones en una localidad de memoria específica, dicha localidad contiene una subrutina que se desarrollará; una vez terminado el proceso y atendida la interrupción, se devuelve el control a la parte del programa que fue interrumpido mismo que reanudará el procesamiento del programa.

Importancia de las prioridades.



UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES INTERRUPCIONES

Sí existe una interrupción, realiza la siguiente secuencia:

1. **El contenido de registro de banderas es guardado en la pila.**
2. **Se limpian las IF y TF. Esto deshabilita la terminal INTR y la propiedad de trampa o de paso sencillo.**
3. **El contenido del registro CS (segmento de código) es guardado en la pila.**
4. **El contenido de IP (apuntador de instrucción) es guardado en pila.**
5. **El contenido del vector de interrupción es leído, y ubicado tanto en el IP como en el CS, de manera que la siguiente instrucción se ejecuta en el procedimiento de servicio de interrupción direccionado por el vector.**

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES INTERRUPCIONES

Interrupción por Software:

Una interrupción puede ser vista como una función. Dicha función hace que el programa se vuelva más simple, lo anterior es posible utilizando un código (carácter) que llame una función en especifico (interrupción) misma que hará todo el trabajo. Existen funciones que trabajan desde el disco duro y otras desde fuera (hardware).

Interrupción por Hardware:

Cuando lo que provoca una interrupción es el cambio de estado en una entrada especifica, se dice que tenemos una interrupción de tipo hardware.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES MACROS

La definición de macros es muy parecida a los procedimientos. Son casi idénticos, pero los macros solo existen si dentro del programa que se compila han sido llamados. Una vez que el programa ha sido compilado, los macros se convierten en parte de las instrucciones de manera real. Si un macro no es utilizado, entonces es ignorado por el compilador.

La definición de un macro es:

```
Nombre      MACRO [parámetros, …]
            < Instrucciones >
ENDM
```

Los macros deben ser definidos ANTES del código.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES MACROS

En el caso de que se tengan varias macros, es conveniente ponerlas en un archivo externo. Para que dicho archivo pueda ser incluido dentro del programa, solo es necesario incluirlo.

```
IF1
        INCLUDE fichero.ext
ENDIF
```

IF1 asegura que el ensamblador lea el fichero solo en la primera pasada, para acelerar el ensamblaje y evitar que aparezcan en el listado (generado en segunda fase)

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES MACROS

Importante:

- **Cuando se desea llamar una macro, solo es necesario poner el nombre y los parámetros (similar a una función en C)**
- **Para indicar que ha terminado la macro, solo es necesario agregar ENDM**
- **Cada vez que se llama una macro, el compilador genera el código. Por lo que si se llama 10 veces, se genera el mismo código 10 veces, haciéndolo mucho más grande que el original (cuidado con eso).**

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES MACROS

```
Macro1 MACRO p1, p2, p3
        mov Ax, p1
        mov Bx, p2
        mov Cx, p3
ENDM

ORG 100h
Macro1 1, 2, 4
Macro1 4, Dx, 8
RET
```

```
mov Ax, 01
mov Bx, 02
mov Cx, 04
mov Ax, 04
mov Bx, Dx
mov Cx, 08
```

Ejemplo de una macro sencilla.
No contiene etiquetas.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES MACROS

Para evitar los errores de declaración múltiple en una macro, se puede utilizar la directiva LOCAL. Se puede usar seguida de nombres de variables, etiquetas y nombres de procedimientos.

```
MINIMO      MACRO dato1, dato2, resultado
            LOCAL ya
            mov, Ax, dato1
            cmp Ax, dato2
            jb ya
            mov Ax, dato2
ya:         mov resultado, Ax
ENDM
```

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES PROCEDIMIENTOS

Un procedimiento es una parte de código que puede ser llamado desde un programa para realizar alguna tarea especifica.

Los procedimientos hacen que los programas sean mas estructurados y fáciles de entender. El retorno de los programas después de la llamada de un procedimiento es a la instrucción siguiente del lugar donde fueron llamados.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES PROCEDIMIENTOS

La sintaxis de un procedimiento es:

```
Nombre  PROC
        ; aquí esta la parte de código
        ; que contiene el procedimiento
RET
Nombre ENDP
```

Nombre : El nombre del procedimiento
RET : Es una instrucción que regresa el control al punto pasado
PROC/ENDP : Directivas del compilador

Características:

1. El procedimiento siempre se declara después del programa principal.
2. No contienen parámetros.
3. Las llamadas pueden ser tan grandes como el segmento de código lo permita, inclusive puede estar en un segmento diferente. Por lo que es recomendable el uso de procedimientos en lugar de saltos condicionados.
4. La mejor forma de utilización de los procedimientos es utilizando los saltos condicionados como enlace entre el código principal y las rutinas que manejan los procedimientos.

UNIDAD 2

Lenguaje Ensamblador
INSTRUCCIONES PROCEDIMIENTOS

```
ORG 100h
MOV AL, 1
MOV BL, 2
CALL m2
CALL m2
CALL m2
CALL m2
RET ; Fin del programa

m2 PROC
MUL BL ; AX = AL * BL.
RET ; return to caller.
m2 ENDP
END
```

Este programa eleva el número dos a la base tres. Es decir realiza la operación 2^3.

M. en C. Karina Y. Sosa González

DUDAS, QUEJAS O SUGERENCIAS…