

**INTITUTO TECNOLÓGICO SUPERIOR DEL SUR DE GUANAJUATO**

**5° SEMESTRE (AGOSTO - DICIEMBRE)**

* **CARRERA:** Ing. en Sistemas Computacionales
* **MATERIA:** Lenguajes de interfaz
* **PROFESOR:** Jorge Guzmán Ramos
* **ALUMNO:** Jorge Luis Medina Gaytán (S14120040)
* **GRUPO:** S66A

**Lenguajes de Interfaz**

**(PORTAFOLIO DE EVIDENCIAS)**

**UNIDAD 1: Introducción al lenguaje ensamblador**

**23/01/2017**

**1.1 – Importancia de la programación en lenguaje ensamblador.**

* Comunicación directa con la máquina (traducción de bajo nivel a hexadecimal).
* Mayor eficiencia.
* Facilidades de codificación comparada con el lenguaje máquina en binario.
* Óptima utilización de recursos de la máquina.

**1.2 – El procesador y sus registros internos.**

* Registros generales.
* Registros de control.
* Registros de estado.

**24/01/2017**

**1.2 – El procesador y sus registros internos.**

* De control: están en la unidad de control y son MAR, MBR, PC, IR e ICC.
* De estado: ICC, FLAGS (PSW no es registro y está en RAM).

**1.3 – La memoria principal.**

* Von Newman: SW + Datos.
* Harvard: SW o Datos.

ME – Memory Enable

CS – Chip Set

RAM (+ Caches L1 y L2)

**1.4 – El concepto de interrupciones.**

Mi programa

Recuperación de errores

E/S

Int.

Int.

-

-Error

-

-

-

-

-

-

-

-

-

-

-

- Leer

-

-

Se guardan las condiciones de retorno: registros guardados.

\* Int. = Interrupción

\* Errores de máquina

\* Instrucciones inválidas

\* Aprovechamiento del procesador

**26/01/2017**

**1.2 – El procesador y sus registros internos.**

MAR Address

MBR Buffer

ADD Op1, Op2

* ADD Var1, Var2
* MOV AX, Var1

ADD AX, Var2

**1.4 – El concepto de interrupciones.**

INT 10H 🡺 INT 16

**26/01/2017**

**1.5 – Llamadas a servicios del sistema.**

INT 21H ; Llamada a servicio del S.O.

INT 10H ; Llamada a servicio del BIOS.

* **BORLAND Inc.**

TASM.- Turbo Assembler (Acepta mayúsculas y minúsculas)

* **Microsoft**

MASM.- Microsoft Assembler

MOV AH, 09H ; Petición de despliegue en pantalla.

LEA DX, Mensaje ; Mensaje a desplegar.

**26/01/2017**

**1.6 – Modos de direccionamiento.**

[ ] 🡺 Dirección para RAM.

**1.7 – Proceso de ensamblado y ligado.**

**2**

**1**

Traduce a lenguaje de máquina.

**.ASM**

**Compilador (TASM)**

**Programa fuente**

**.OBJ**

**3**

**4**

**Programa ejecutable**

**Enlazador (TLINK)**

Resuelve referencias externas (CALL).

Se utiliza un solo paso en lenguajes de alto nivel.

Se utiliza un editor de texto sin formato.

D: TASM\TASM MyProg

D: TASM\TLINK MyProg

D: TSM\MyProg

**31/01/2017**

**1.8 – Desplegado de mensaje en el monitor.**

Llamadas a servicio del sistema o BIOS:

* A través de una interrupción.
* Se envían parámetros de control a través de registros generales.
* El servicio solicitado lo hace un programa, por lo tanto usa los recursos del sistema.

MOV AH, 09H

MOV DX, Offset Mensaje 🡸🡺 LEA Mensaje

INT 21H

MOV AH, 09H

LEA DX, Mensaje2

INT 21H

Inicio MOV AH, 09H

**UNIDAD 2: Programación básica**

**02/02/2017**

**2.2 – Ciclos numéricos.**

**2.3 – Captura básica de cadenas.**

Datos SEGMENT

Etiqueta LABEL BYTE

Mcortoong DB 30

Relong DB ?

Nombre DB 31 DUP(‘ ‘)

Peticion DB 10,13,”Teclea tu nombre”,”$”

Datos ENDS

Codigo SEGMENT

Assumme …

Mov AX, Datos

Mov DS, AX

Mov AX, 09H

Mov DX, offet Peticion

INT 21H

Mov AH, 0AH

LEA DX, Etiqueta

INT 21H

Mov BH, 00

Mov BL, Relong

Mov Nombre[Bx + 1], “$”

Mov AH, 09H

Lea DX, Nombre

INT 21H

**07/02/2017**

**2.4 – Comparación y prueba.**

Etiqueta CMP op1, op2

\* El operando 1 se compara con el operando 2 y se coloca el indicador (flag) correspondiente.

CMP Reg / Memoria, Reg / Memoria / Inmediato

Etiqueta Snnn Etiqueta 2

Donde nnn:

* A: Mayor que (after)
* B: Menor que (Before)
* E: Equals (Igual)
* Z: Cero
* S: Negativo
* C: Acarreo (Carry)
* O: Desbordamiento (Overflow)
* P: Par
* N: Negación

**09/02/2017**

**2.5 – Saltos.**

**2.6 – Ciclos condicionales.**

Etiqueta JMP Etiqueta2

Etiqueta Jnnn Etiqueta2

Ciclos:

…

…

…

CMP op1, op2

Jnnn Ciclo

**2.7 – Incremento y decremento.**

Etiqueta INC Reg / Memoria

Etiqueta DEC Reg / Memoria

**09/02/2017**

**2.8 – Captura de cadenas con formato.**

MOVS - Mueve (copia) byte a byte, palabra o sobre palabra una cadena a otra localidad de memoria.

LODS - Carga desde memoria un byte de AL, una palabra de AX o una sobre de EAX.

STOS - Almacena el contenido de los registros AL, AX o EAX en memoria.

CMPS - Compara dos cadenas.

SCAS - Compara el contenido de AL, AX o EAX con el contenido de una localidad de memoria.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operación** | **Instrucción** | **Operandos** | **Byte** | **Palabra** | **Doble palabra** |
| Mover | MOVS | ES:DI, DS:SI | MOVSB | MOVSW | MOVSD |
| Carga | LODS | AX,DS:SI | LODSB | LODSW | LODSD |
| Almacenar | STOS | ES:DS,AX | STOSB | STOSW | STOSD |
| Comparar | CMPS | DS,SI, ES,DI | CMPSB | CMPSW | CMPSD |
| Rastrear | SCAS | ES:DI,AX | SCASB | SCASW | SCASD |

**16/02/2017**

**2.9 – Instrucciones aritméticas.**

Etiqueta ADD op1, op2

\* Suma op1 a op2 y el resultado queda en op1 (SUB Minuendo, Sustrayendo).

|  |  |  |  |
| --- | --- | --- | --- |
| **Instrucción** | **Multiplicando** | **Multiplicador** | **Producto** |
| DIV CL | AL | CL | AX |
| DIV BX | AX | BX | DX:AX o EAX |
| DIV EDX | EAX | EBX | EDX: EAX o RAX para 64 bits |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instrucción** | **Dividendo** | **Divisor** | **Cociente** | **Residuo** |
| DIV CL | AX | CL | AL | AH |
| DIV CX | DX:AX | CX | AX | DX |
| DIV EBX | EDX:EAX | EBX | EAX | EDX |

**Multiplicación por corrimiento:**

SHL Op1, Corrimiento

SHL AL, 2

**División por corrimiento:**

SHR Op1, Corrimiento

**28/02/2017**

**2.10 – Manipulación de la pila.**

* POPA ; Recupera todos los registros
* POPF ; Recupera el EFLAGS (2 bytes) / (POPFD 4 bytes)
* PUSHA ; Guarda todos los registros generales
* PUSHF ; Guarda FLAGS / (PUSHFD guarda EFLAGS)

**2.12 – Instrucciones lógicas.**

Etiqueta AND Op1, Op2

OR Op1, Op2

NOT Op1

XOR Op1, Op2

**Fecha del sistema:**

MOV AH, 2AH ; Obtener la fecha de BIOS

INT 21H ; Llamada al sistema

Se obtiene en:

AL= día de la semana (0 = domingo)

CX = año

DH = mes (01 a 12)

DL = día (01 a 31)

**07/03/2017**

**2.13 – Desplazamiento y rotación.**

[Etiqueta] SHL / SHR

SHL (Izquierda)

SHL AL, 1

AL, CL

SHR (Derecha)

AL, 2

Registro / Memoria, CL / Inmediato

AL = 0100 1011 (75)

AL = 1001 0110 (150)

AL = 0001 0010 (18)

SHL = Multiplicar por 2n donde n son las posiciones de desplazamiento.

SHR = Divide entre 2n donde n son las particiones de desplazamiento.

**UNIDAD 3: Modularización**

**15/03/2017**

**3.1 – Procedimientos.**

Nombre PROC

-

-

-

RET

Nombre ENDP

MOV BH, 1EH ; Coloca color de fondo (1) y texto (E)

|  |  |  |
| --- | --- | --- |
| **Opaco** | **Brillante** |  |
| 0 | 8 | **Negro** |
| 1 | 9 | **Rey** |
| 2 | A | **Verde** |
| 3 | B | **Cielo** |
| 4 | C | **Rojo** |
| 5 | D | **Magenta** |
| 6 | E | **Amarillo** |
| 7 | F | **Blanco** |

LRC – Longitudinal Redundancy Check

CRC – Cyclic Redundancy Check

**30/03/2017**

**3.2 – Macros.**

Nombre PROC (near | far)

-

-

-

RET

Nombre ENDP

Ejecución: CALL Nombre

\*\* No admite parámetros, se pasa información a través de los registros generales.

Nombre MACRO Par1, Par2, \_, Par15

-

-

-

ENDM

Ejecución: Nombre Par1, Par2, \_, Par15

\*\* Se debe definir antes de ser ejecutada.

**Módulo Inicio:** Todo lo que se hace una sola vez y al principio.

**Módulo Principal:** Contiene la lógica total del programa organizado a través de CALLS.

**Módulo Final:** Contiene todo lo que se hace una sola vez y al terminar el programa.

**UNIDAD 4: Programación de dispositivos**

**24/04/2017**

**4.1 – Buffer de video en modo texto.**

Establecer tamaño de cursor:

MOV AH, 01H ; Agrandar el cursor

MOV CH, 00 ; Línea inicial

MOV CL, 14 ; Línea final

INT 10H ; Llamado al BIOS

**4.5 – Programación hibrida.**

# include stdio.h

# include conio.h

main ( ){

int dato1, dato2, producto;

printf (“Teclea dos números: “);

scanf (“%d”, “%d”, &dato1, &dato2);

asm push ax

asm push cx

asm mov cx, dato1

asm mov ax, 0h

mult:

asm add ax, dato2

asm loop mult

asm mov producto, ax

asm pop cx

asm pop ax

printf (“El producto es: %d”, producto);

}

**08/05/2017**

**4.2 – Acceso a discos en lenguaje ensamblador**

- Manejo de archivos con método de acceso (S.O)

- Manejo de archivos en modo nativo (BIOS)

MOV AH, 16H ; Crea un archivo

LEA DX, Nombre ; Nombre FCP

INT 21H ; Llamado al DOS

MOV AH, 1AH ; Área de transformación DTA

LEA DX, Nombre ; Nombre de la DTA

INT 21H ; Llamado al DOS

\* DTA.- Data Transfer Area

**09/05/2017**

**4.2 – Acceso a discos en lenguaje ensamblador**

MOV AH, 0FH ; Petición de abrir

LEA DX, Nombre ; Nombre del FCB

INT 21H ; Llamada al DOS

MOV AH, 14H ; Petición de leer

LEA DX, Nombre ; FCB para leer un registro

INT 21H ; Llamada al DOS

MOV AH, 10H ; Petición de cerrar

LEADX, Nombre ; Nombre del FCB

INT 21H ; Llamado al DOS

**Formato del FCB:**

|  |  |
| --- | --- |
| **Byte** | **Contenido** |
| **0** | Unidad de disco; 0 = A, 1 = B, 2 = C … |
| **1 – 8** | Nombre del archivo justificado a la izquierda. |
| **9 – 11** | Extensión del nombre (.EXE, .TXT, .CIB, etc.). |
| **12 – 13** | Número del bloque actual. |
| **14 - 15** | Tamaño del registro lógico. |
| **16 – 19** | Tamaño del archivo. |
| **20 – 21** | Fecha de última actualización. |
| **22 – 31** | Reservado para el S.O. |
| **32** | Número de registro actual dentro del bloque. |
| **33 - 36** | Número relativo de registro. |

**10/05/2017**

**4.2 – Acceso a discos en lenguaje ensamblador**

**Lectura / escritura absoluta:**

MOV AL, Unidad ; # 0 para A, 1 para B,...

MOV BX, Direc ; Coloca la dirección del dato

MOV CX, Sector ; Numero de sectores para leer / escribir

MOV DX, #Sector ; Numero relativo del sector

INT 25H / 26H ; Petición de leer / escribir

POPF ; Obtiene los indicadores (flags) si se desea

JC Error ; Ir manejar el error

1000 0000 Falla de conexión

0100 0000 Falla de búsqueda

0000 1000 Falla de paridad (CRC)

0000 0100 Sector no encontrado

0000 0011 Disco protegido físicamente contra escritura

0000 0010 Error no clasificado

\*Cyclic Redundancy Check (CRC)