UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
Facultad de Ingeniería, Arquitectura y Diseño

Ingeniero en Software y Tecnologías Emergentes

Manual de Talleres de la materia Organización de Computadoras

AUTOR(ES)
Jonatan Crespo Ragland

Taller No. 8 Instrucciones de control de flujo

Objetivo: Desarrollar los códigos de operación y proceso de ensamblaje

Fundamentos teóricos del taller

- Lenguaje Ensamblador x86

Instrucciones para el desarrollo del taller

1. Desarrolla los siguientes puntos

Recursos

1. Apuntes de clase.
2. Fuentes bibliográficas.
3. Lápiz.
4. Equipo de cómputo.

Tiempo e instrucciones de entrega

Duración: 2 horas.

Desarrollar la lógica o pseudocódigo de los siguientes problemas en ensamblador x86, utilizando solamente la lógica de los registros de bandera y saltos condicionales vistos en clase. No es necesario que compilen correctamente y se pueden utilizar bloques a manera de comentario, aunque sí deben estar en su repositorio de código. Ejemplo: (Código de lectura aquí...)

1. Comparador de Números: Escribir un programa que reciba dos números y determine si son iguales, si uno es

mayor que el otro, o si son negativos.

; Programa: Comparador de dos números

; Objetivo: Determinar si son iguales, si uno es mayor, o si son negativos

```
MOV AX, num1 ; Cargar primer número
MOV BX, num2 ; Cargar segundo número
CMP AX, BX ; Compara AX - BX

JE SON_IGUALES ; Si ZF=1, son iguales
JG MAYOR ; Si SF=0 y ZF=0, AX > BX
JL MENOR ; Si SF=1, AX < BX

SON_IGUALES:
 ; (mensaje) “Los números son iguales”
 JMP FIN

MAYOR:
 ; (mensaje) “El primer número es mayor”
 JMP VERIFICAR_SIGNO

MENOR:
 ; (mensaje) “El segundo número es mayor”

VERIFICAR_SIGNO:
 TEST AX, AX
 JS NEGATIVO_AX
 TEST BX, BX
 JS NEGATIVO_BX
 JMP FIN
```

```
NEGATIVO_AX:
 ; (mensaje) “El primer número es negativo”
 JMP FIN

NEGATIVO_BX:
 ; (mensaje) “El segundo número es negativo”

FIN:
 ; Fin del programa
```

2. Clasificación de Números: Leer un número y clasificarlo como positivo, negativo o cero.

```
; Clasifica un número como positivo, negativo o cero

MOV AX, num
CMP AX, 0
JE CERO
JS NEGATIVO
JNS POSITIVO

CERO:
 ; “El número es cero”
 JMP FIN

NEGATIVO:
 ; “El número es negativo”
 JMP FIN
```

POSITIVO:

; “El número es positivo”

FIN:

3. Par o Impar: Leer un número y determinar si es par o impar usando únicamente la bandera de paridad (PF).

; Determina si un número es par o impar usando la bandera PF

MOV AL, num

TEST AL, 1 ; Verifica el bit menos significativo

; La bandera de paridad (PF) indica si el número de bits en 1 del resultado es par

JP PAR ; Si PF=1 → par

JNP IMPAR ; Si PF=0 → impar

PAR:

; “El número es par”

JMP FIN

IMPAR:

; “El número es impar”

FIN:

4. Simulación de Overflow: Pedir dos números y sumarlos, verificando si ocurre desbordamiento con la bandera OF (Overflow Flag). Imprimir un mensaje si se detecta overflow.

; Detectar overflow al sumar dos números con bandera OF

```
MOV AL, num1
ADD AL, num2
JO DESBORDAMIENTO ; Si OF=1 → ocurrió overflow
JNO SIN_OVERFLOW ; Si OF=0 → no hubo overflow

DESBORDAMIENTO:
 ; “Se detectó un desbordamiento (overflow)”
 JMP FIN

SIN_OVERFLOW:
 ; “No hubo desbordamiento”
FIN:
```

5. Simulación de Acarreo: Realizar una suma entre dos números y verificar si hay un acarreo con la bandera CF (Carry Flag). Mostrar si se generó un acarreo o no.

```
; Detectar acarreo con la bandera CF

MOV AL, num1
ADD AL, num2
JC ACARREO ; Si CF=1 → hay acarreo
JNC SIN_ACARREO ; Si CF=0 → no hay acarreo

ACARREO:
 ; “Se generó un acarreo”
 JMP FIN
SIN_ACARREO:
 ; “No se generó acarreo”
```

FIN:

6. Mínimo y Máximo de Tres Números: Leer tres números e identificar el menor y el mayor.

; Identificar el menor y el mayor de tres números

```
MOV AX, num1
MOV BX, num2
MOV CX, num3

; --- Buscar máximo ---
CMP AX, BX
JL MAYOR_BX
JGE SIG_MAX

MAYOR_BX:
 MOV AX, BX
SIG_MAX:
 CMP AX, CX
 JL MAYOR_CX
 JMP SIG_MIN
MAYOR_CX:
 MOV AX, CX
SIG_MIN:
 MOV DX, AX ; DX = máximo

; --- Buscar mínimo ---
MOV AX, num1
CMP AX, BX
JG MENOR_BX
```

```
JLE SIG_MIN2
MENOR_BX:
 MOV AX, BX
SIG_MIN2:
 CMP AX, CX
 JG MENOR_CX
 JMP FIN
MENOR_CX:
 MOV AX, CX

; AX = mínimo, DX = máximo
FIN:
```

7. Ordenamiento de Dos Números

Leer dos números e intercambiarlos si no están en orden ascendente usando solo saltos condicionales.

```
; Ordenar dos números en orden ascendente

MOV AX, num1
MOV BX, num2
CMP AX, BX
JLE ORDENADOS ; Si ya están en orden, saltar
XCHG AX, BX ; Si no, intercambiarlos

ORDENADOS:
 ; AX <= BX
FIN:
```

8. Ciclo de Conteo sin Comparaciones: Implementar un contador de 0 a 9.

```
; Contador de 0 a 9 sin comparaciones

MOV CX, 10 ; Contar 10 veces
MOV AL, 0

CICLO:
 ; Mostrar AL
 INC AL
 LOOP CICLO ; Usa CX como contador automático (decrementa y salta si no llega a 0)

FIN:
```

Referencias

Anexos

JE	Salta si los operandos son iguales
JNE	Salta si los operandos no son iguales
JG	Salta si el primer operando es mayor que el segundo
JGE	Salta si el primer operando es mayor o igual al segundo
JL	Salta si el primer operando es menor que el segundo

JLE	Salta si el primer operando es menor o igual al segundo
JA	Salta si el primer operando es mayor sin acarreo
JAE	Salta si el primer operando es mayor o igual sin acarreo
JB	Salta si el primer operando es menor con acarreo
JBE	Salta si el primer operando es menor o igual con

	acarreo
JC	Salta si hubo acarreo (Carry Flag = 1)
JNC	Salta si no hubo acarreo (Carry Flag = 0)
JO	Salta si hubo desbordamiento (Overflow Flag = 1)
JNO	Salta si no hubo desbordamiento (Overflow Flag = 0)
JS	Salta si el resultado es negativo (Sign Flag = 1)
JNS JP / JPE JNP / JPO JZ	Salta si el resultado no es negativo (Sign Flag = 0) Salta si el resultado tiene paridad par (Parity Flag = 1) Salta si el resultado tiene paridad impar (Parity Flag = 0) Salta si el resultado es cero (Zero Flag = 1)

AND	Realiza una operación lógica AND entre dos operandos
OR	Realiza una operación lógica OR entre dos operandos
XOR	Realiza una operación lógica XOR entre dos operandos
NOT	Realiza una operación lógica NOT (negación) sobre un operando
TEST	Realiza una operación AND entre dos operandos y ajusta las banderas sin almacenar el resultado
SHL / SAL	Desplazamiento lógico a la izquierda (Shift Left)
SHR	Desplazamiento lógico a la derecha (Shift Right)
ROL	Rotación a la izquierda (Rotate Left)

CF	Carry Flag: Indica si hubo un acarreo en una operación de suma o si ocurrió un "préstamo" en una resta.
PF	Parity Flag: Indica si el número de bits 1 en el resultado

	es par (1) o impar (0).
AF	Auxiliary Carry Flag: Se activa si hay un acarreo entre los 3 y 4 bits más bajos en una operación.
ZF	Zero Flag: Se activa si el resultado de la operación es 0.
SF	Sign Flag: Indica si el resultado de la operación es negativo (el bit más significativo es 1).
TF	Trap Flag: Se activa para permitir la ejecución de una instrucción de "trampa" (utilizado en depuración).
IF	Interrupt Enable Flag: Indica si las interrupciones están habilitadas (1) o deshabilitadas (0).
DF OF	Direction Flag: Determina si el procesamiento de cadenas de caracteres ocurre hacia arriba o hacia abajo en memoria. Overflow Flag: Se activa si ocurrió un desbordamiento aritmético.

DB	Define un byte o un conjunto de bytes.
DW	Define una palabra (2 bytes).
DD	Define una doble palabra (4 bytes).
DQ	Define una cuádruple palabra (8 bytes).
DT	Define un tipo de 10 bytes (utilizado para valores de punto flotante de precisión extendida).
RESB	Reserva espacio en bytes. No inicializa los valores.
RESW	Reserva espacio en palabras (2 bytes).
RESD	Reserva espacio en doble palabra (4 bytes).