



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 1 de 14

I. TEMA: PROGRAMACION EN ENSAMBLADOR – INSTRUCCIONES DE SALTO

II. OBJETIVOS DE LA PRACTICA

Al finalizar la presente práctica, el estudiante:

1. Entiende la lógica de funcionamiento de las instrucciones de bifurcación de los procesadores con arquitectura X86.
2. Implementa, utilizando el lenguaje ensamblador de los procesadores con arquitectura X86, programas con estructuras de control de bifurcación.
3. Escribe programas en lenguaje Ensamblador para Linux utilizando el compilador NASM

III. TRABAJO PREPARATORIO.

1. Conceptos básicos de organización y arquitectura de computadores.
2. Conocimientos básicos del conjunto de instrucciones ensamblador de los microprocesadores con arquitectura X86

IV. MATERIALES.

1. Sistema operativo Linux
2. Compilador NASM
3. Librería io.mac

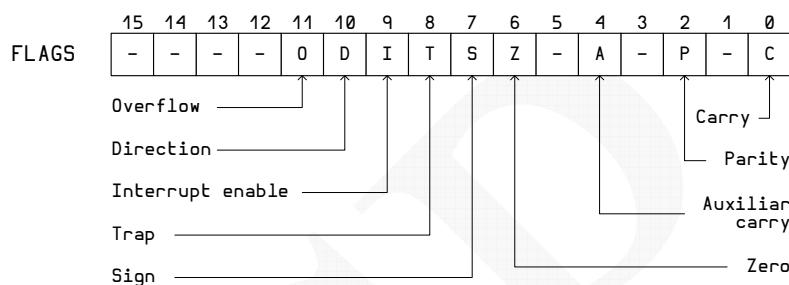


V. MARCO TEORICO

REGISTRO DE BANDERAS X86

El registro de banderas en los procesadores 8086, almacena información sobre algunas condiciones que pueden presentarse como resultado de las instrucciones que se ejecutan y también sobre condiciones que permiten controlar el modo de operación del procesador.

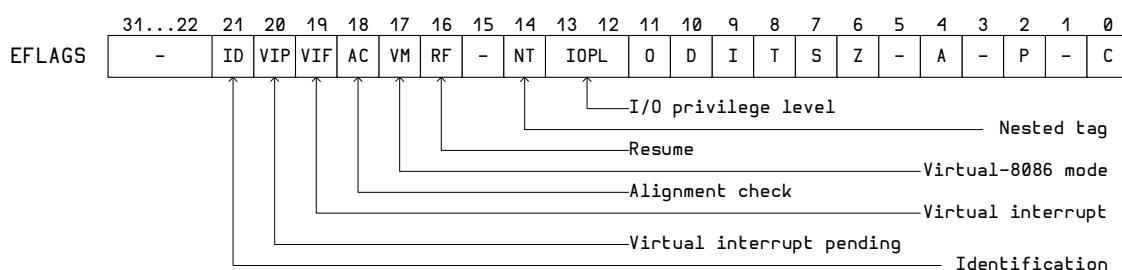
El registro de banderas (Flags Register) del procesador 8086, conocido también como palabra de estado del procesador (PSW – Processor State Word) tiene el siguiente formato:



El propósito de cada bandera o indicador es:

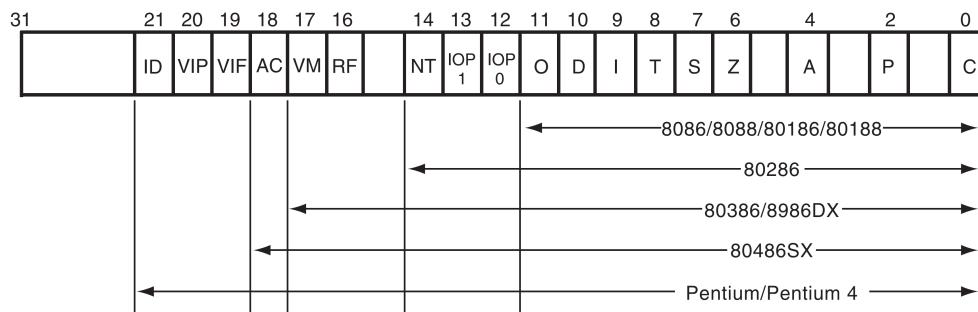
- O (Desbordamiento)** - Se activa cuando una operación, con signo, genera un resultado cuyo valor excede la capacidad de almacenamiento de los registros.
- D (Dirección)** - Se utiliza para determinar la dirección de incremento (positivo o negativo) automático para operaciones con cadenas utilizando registros DI o SI.
- I (Interrupción)** - Se activa cuando un dispositivo requiere atención del procesador.
- T (Trampa)** - Se utiliza para tareas de depuración de programas.
- S (Signo)** - Se modifica de acuerdo al signo del resultado de la última operación.
- Z (Cero)** - Se activa si el resultado de la última operación es igual a cero.
- A (Acarreo auxiliar)** - Se activa cuando ocurre un acarreo entre los bits 3 y 4 (BCD).
- P (Paridad)** - Se activa si el número de unos (1) en un número es par.
- C (Acarreo)** - Se activa cuando hay acarreo desde el bit más significativo en una operación de suma o un préstamo en una operación de resta.

En los procesadores de Intel actuales el registro de banderas (EFLAGS) incluye banderas para dar soporte a capacidades requeridas por aplicaciones modernas:





Estas banderas se han ido agregando en cada uno de los modelos de procesadores Intel, como puede apreciarse en la figura tomada del texto de Barry Brey (“*The Intel Microprocessors*” 8va edición)



INSTRUCCIONES DE SALTO

La familia de microprocesadores X86, incluye un gran número de instrucciones de salto. El propósito de estas es permitir que se ejecute una u otra secuencia de instrucciones, dependiendo de condiciones resultantes de la ejecución de instrucciones previas, las cuales son mantenidas en el registro de banderas del procesador.

Una instrucción de salto sigue la siguiente sintaxis

Jxx Destino

En donde:

Jxx – Es la instrucción de salto. En este caso las “x” representan las variaciones de esta instrucción (basadas en la condición que verifican)

Destino– La dirección a donde se saltará en caso que el salto sea efectivo, caso contrario, se ejecutará la siguiente instrucción a Jxx.

Los saltos pueden ser incondicionales o condicionales.

Instrucción de salto incondicional.

Esta instrucción se utiliza para saltar, independientemente de cualquier condición. La sintaxis de esta instrucción es la siguiente:

JMP dest

Operación: Salta incondicionalmente a *dest*.

Instrucciones de salto condicional

Son instrucciones de salto que verifican alguna condición en el sistema, y dependiendo de la misma, realizan o no el salto.

Estas instrucciones pueden agruparse en:



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 4 de 14

1. Saltos basados en el valor de una bandera aritmética única

JC etiqueta	Saltar si hubo arrastre/préstamo (CF = 1).
JNC etiqueta	Saltar si no hubo arrastre/préstamo (CF = 0).
JZ etiqueta	Saltar si el resultado es cero (ZF = 1).
JNZ etiqueta	Saltar si el resultado no es cero (ZF = 0).
JS etiqueta	Saltar si el signo es negativo (SF = 1).
JNS etiqueta	Saltar si el signo es positivo (SF = 0).
JP/JPE etiqueta	Saltar si la paridad es par (PF = 1).
JNP/JPO etiqueta	Saltar si la paridad es impar (PF = 0).

2. Saltos basados en comparaciones sin signo

JB etiqueta/JNAE etiqueta	Saltar a <i>etiqueta</i> si es menor.
JBE etiqueta/JNA etiqueta	Saltar a <i>etiqueta</i> si es menor o igual.
JE etiqueta	Saltar a <i>etiqueta</i> si es igual.
JNE etiqueta	Saltar a <i>etiqueta</i> si es distinto.
JAE etiqueta/JNB etiqueta	Saltar a <i>etiqueta</i> si es mayor o igual.
JA etiqueta/JNBE etiqueta	Saltar a <i>etiqueta</i> si es mayor.

3. Saltos basados en comparaciones con signo

JL etiqueta/JNGE etiqueta	Saltar a <i>etiqueta</i> si es menor.
JLE etiqueta/JNG etiqueta	Saltar a <i>etiqueta</i> si es menor o igual.
JE etiqueta	Saltar a <i>etiqueta</i> si es igual.
JNE etiqueta	Saltar a <i>etiqueta</i> si es distinto.
JGE etiqueta/JNL etiqueta	Saltar a <i>etiqueta</i> si es mayor o igual.
JG etiqueta/JNLE etiqueta	Saltar a <i>etiqueta</i> si es mayor.

Instrucción de comparación

Permite actualizar los bits del registro de banderas del microprocesador, de manera que luego estas banderas sean utilizadas por una instrucción de salto posterior

CMP dest,src	Operación: <i>dest - src</i> (sólo afecta flags del registro de banderas).
---------------------	--



IMPLEMENTACIÓN DE ESTRUCTURAS DE CONTROL

Las instrucciones de control de alto nivel tales como las estructuras ***if – then – else*** y ***switch – case*** se convierten, después del proceso de compilación, en una secuencia de instrucciones ensamblador, todas basadas en instrucciones de salto o bifurcación como las estudiadas.

A continuación, veremos un ejemplo de esas equivalencias, aclarando que estas no son la única alternativa de correspondencia.

Estructura IF – THEN – ELSE

Una instrucción de bifurcación que en el lenguaje de programación C se expresa de la siguiente forma:

```
if (X > Y) {  
    mayor = X;  
}  
else {  
    mayor = Y;  
}
```

Sería equivalente al siguiente código en ensamblador:

```
mov ax, word [X]  
mov bx, word [Y]  
  
cmp ax, bx  
  
js segundoEsMayor  
mov [mayor], ax  
jmp fin  
  
segundoEsMayor:  
    mov [mayor], bx  
  
fin:  
    mov bx, [mayor]
```

Estructura SWITCH – CASE

Una instrucción de alternativa múltiple que en el lenguaje de programación C se expresa de la siguiente forma:

```
switch (opcion) {  
    case 1:  
        printf("Ha ingresado el numero 1\n");  
        Bloque_de_instrucciones_01;  
        break;
```



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 6 de 14

```
case 2:  
    printf("Ha ingresado el numero 2\n");  
    break;  
default:  
    printf("Ha ingresado un numero diferente de 1 y  
           2\n");  
    break;  
}
```

Sería equivalente al siguiente código en ensamblador:

```
cmp word[opcion], 1  
je esUno  
  
;no es uno  
cmp word[opcion], 2  
je esDos  
  
;no es dos  
jmp esOtro  
  
esUno:  
PutStr msgUno  
jmp final  
  
esDos:  
PutStr msgDos  
jmp final  
  
esOtro:  
PutStr msgOtro  
  
final:  
mov ax,1 ;Salimos del programa  
int 80h
```



VI. TRABAJO DE LABORATORIO.

1. Escriba un programa que lea dos números enteros y muestre el mayor de ellos

Solución

```
;Nombre      : mostrarMayor.asm
;Proposito   : muestra el mayor de dos numeros
;Autor       : Edwin Carrasco
;FCreacion  : 23/06/2010
;FModif.    : 11/09/2019
;compilar   :
;           nasm -f elf mostrarMayor.asm
;           ld -m elf_i386 -s -o mostrarMayor
mostrarMayor.o io.o

%include "io.mac"

section .data
mensaje: db "ESTE PROGRAMA MUESTRA EL MAYOR DE DOS
NUMEROS",10,0
nro1: db "Ingrese el primer número: ", 0
nro2: db "Ingrese el segundo número: ", 0
salida: db "El mayor es: ", 0, 10

section .text
global _start

_start:
PutStr mensaje
PutStr nro1
GetInt ax
PutStr nro2
GetInt bx

;Comparar números
cmp ax, bx
js negativo

;ax es mayor
PutStr salida
PutInt ax
jmp final

negativo:
;bx es el menor
PutStr salida
PutInt bx
```



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 8 de 14

```
final:  
    nwln  
    mov ax, 1  
    int 80h
```

```
root@aar:/home/ecp/codigo/AC# nasm -f elf mostrarMayor.asm  
root@aar:/home/ecp/codigo/AC# ld -m elf_i386 -s -o mayor mostrarMayor.o io.o  
root@aar:/home/ecp/codigo/AC# ./mayor  
ESTE PROGRAMA MUESTRA EL MAYOR DE DOS NUMEROS  
Ingrese el primer numero: 4  
Ingrese el segundo numero: 8  
El mayor es: 8
```



2. Escriba un programa que determine si un número es par

Solución

```
;Nombre      : testPar.asm
;Proposito   : determina si un numero es par
;Autor       : Edwin Carrasco
;FCreacion  : 23/06/2010
;FModif.    : 11/09/2019
;compilar   :
;           nasm -f elf testPar.asm
;           ld -m elf_i386 -s -o testPar testPar.o io.o

%include "io.mac"
section .data
msg: db "ESTE PROGRAMA DETERMINA SI UN NUMERO ES
      PAR",10 ,0
nro: db "Ingrese el numero: ", 0
msgPar: db "El numero es par", 0
msgImpar: db "El numero es impar", 0

section .text

global _start

_start:
PutStr msg
PutStr nro
GetInt ax
rcr ax, 1
jc esImpar
;es par
PutStr msgPar
jmp final

esImpar:
PutStr msgImpar

final:
nwln
mov ax, 1
int 80h
```

```
root@aar:/home/ecp/codigo/AC# nasm -f elf testPar.asm
root@aar:/home/ecp/codigo/AC# ld -m elf_i386 -s -o testPar testPar.o io.o
root@aar:/home/ecp/codigo/AC# ./testPar
ESTE PROGRAMA DETERMINA SI UN NUMERO ES PAR
Ingrese el numero: 177
El numero es impar
```



3. Escriba un programa que lea la nota de un alumno e indique si el alumno esta reprobado, desaprobado, aprobado o es sobresaliente. Las notas deben ingresarse en tiempo de ejecución. El mensaje que se debe mostrar, según sea la nota, se indica en la tabla adjunta:

NOTA	MENSAJE
0 – 6	Reprobado
7 – 10	Desaprobado
11 – 16	Aprobado
17 – 20	Sobresaliente

Solución

```
;Nombre      : ranking.asm
;Proposito   : Clasifica a un alumno de acuerdo a su nota
;Autor       : Edwin Carrasco
;FCreacion  : 06/10/2021
;FModif.    : ---
;compilar    : nasm -f elf ranking.asm
;              ld -m elf_386 io.o ranking.o -s -o ranking
;              ./ranking

%include "io.mac"

section .data
msgProposito db "ESTE PROGRAMA MUESTRA UN MENSAJE SEGUN LA
                  NOTA OBTENIDA POR UN ESTUDIANTE",10,0
msgPedirNota db "Ingrese la nota del estudiante: ",0
msgMostrarSalida db "La clasificacion del estudiante es :", 0
msgReprobado db " - REPROBADO",10,0
msgDesaprobado db " - DESAPROBADO",10,0
msgBueno db " - APROBADO", 10,0
msgSobresaliente db " - SOBRESALIENTE", 10,0

section .text
global _start

_start:
xor bx, bx
mov cx,1
mov dx, 0
PutStr msgProposito

;Leer datos
leer/nota:
PutStr msgPedirNota
GetInt ax

;Procesar
PutStr msgMostrarSalida
```



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 11 de 14

```
cmp ax,6           ; si nota < 7
jg no_reprobo    ; reprobado
PutStr msgReprobado
jmp final

no_reprobo:
cmp ax,10          ; Si nota < 11
jg no_desaprobo   ; desaprobado
PutStr msgDesaprobado
jmp final

no_desaprobo:
cmp ax,16          ; Si nota < 16
jg sobresaliente   ; aprobado
PutStr msgBueno
jmp final

sobresaliente:
PutStr msgSobresaliente

final:
mov ax,1 ;Salimos del programa
int 80h
```

```
ecp@ecp-0AC:~/codigo_ASM$ #Prueba de funcionalidad
ecp@ecp-0AC:~/codigo_ASM$ nasm -f elf ranking.asm
ecp@ecp-0AC:~/codigo_ASM$ ld -m elf_i386 io.o ranking.o -s -o ranking
ecp@ecp-0AC:~/codigo_ASM$ ./ranking
ESTE PROGRAMA MUESTRA UN MENSAJE SEGUN LA NOTA OBTENIDA POR UN ESTUDIANTE
Ingrrese la nota del estudiante: 15
La clasificacion del estudiante es : - APROBADO
```



UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ORGANIZACIÓN Y ARQUITECTURA DEL COMPUTADOR
GUIA DE LABORATORIO

ECP 12 de 14

VII. PRACTICAS DE LABORATORIO

1. Escriba un programa que lea la longitud de tres segmentos y determine si estos pueden formar un triángulo.
2. Escriba un programa que lea el año, mes y día de nacimiento de dos personas e indique cuál de los dos es mayor. Asuma que no existen años bisiestos.
3. El impuesto a la renta de 5ta categoría se calcula de acuerdo a la siguiente tabla

REMUNERACIONES				
GOBIERNO REGIONAL				S/. 12,345.00
MINSA				S/. 67,890.00
				TOTAL S/. 80,235.00
UNIDAD IMPOSITIVA				S/. 4,300.00
NO AFECTO (7 UIT)				S/. 30,100.00
				DIFERENCIA AFECTA (RENTA NETA) S/. 50,135.00
IMPUESTO				
Hasta 5 UIT	0.08	S/. 21,500.00	S/. 21,500.00	S/. 1,720.00
De 5 a 20 UIT	0.14	S/. 86,000.00	S/. 28,635.00	S/. 4,008.90
De 20 a 35 UIT	0.17	S/. 150,500.00	S/. 0.00	S/. 0.00
De 35 a 45 UIT	0.20	S/. 193,500.00	S/. 0.00	S/. 0.00
Mas de 45 UIT	0.30	S/. 193,500.00	S/. 0.00	S/. 0.00
				TOTAL IMPUESTOS S/. 5,728.90
DEDUCCIONES				
GOBIERNO REGIONAL				S/. 1,357.99
MINSA				S/. 2,468.00
				TOTAL DEDUCCIONES S/. 3,825.99
				RETENCIONES EN EXCESO/DEUDA S/. 1,902.91

Escriba un programa que lea las remuneraciones, las deducciones y la UIT y determine las retenciones en exceso o deuda.



VIII. EVALUACION

La evaluación de las actividades realizadas en la presente guía de práctica se hará en función de la siguiente tabla:

ACTIVIDAD	Procedimental	
	Sesión 01	Sesión 02
Resolución del ejercicio propuesto 01	--	04
Resolución del ejercicio propuesto 02	--	07
Resolución del ejercicio propuesto 03	--	09
TOTAL	--	20



IX. REFERENCIAS

1. Brey Barry. “*Los Microprocesadores Intel. Arquitectura, Programación e Interfaces*”. Prentice Hall 3Ed.
2. Brey Barry “*The Intel Microprocessors. Architecture, Programming, and Interfacing*” Prentice Hall 8ed.
3. Carter P. “*PC Assembly Language*” 2005
4. Hyde Randall. “*Art of Assembly Language Programming*”. Nostarch Press 1Ed.

