





LAB 2

tema 2


Flag registers


- O: overflow
- D: direction
- I: enabled interrupt
- T: traps
- S: signed (0 si el valor es negativo)
- Z: zero
- A: auxiliar carry
- P: parity (1 se es par y 0 si es impar) (**el numero 2 es impar ya que solo tiene un bit 1 puesto a uno) 0010
- C: carry


Instrucciones

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	 imagenes	:≡ Lesson
<u>LEA</u>	lea target, source	get source addres y lo guarda in target	lea dx, op1		Lesson 2
<u>JMP</u>	jmp etiqueta	salto incondicional a otra parte del codigo	jmp suma		Lesson 2

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	 imagenes	:≡ Lesson
<u>Loop</u>	loop label	bucle, el que lleva la cuenta es CX. Por cada vuelta decrementa cx en uno	MOV CX, 4 Bucle: INC BX ADD BX, CX LOOP Bucle		Lesson 2
<u>cmp</u>	cmp destino, origen	IF pero no almaceno el resultado (igual que sub pero este si que almacena el resultado)	cmp ax, bx (Si ax=bx el resultado es Z, si bx>ax da resultado negativo y bx<ax para cualquier otro caso)		Lesson 2
<u>INT</u>	int tipo_interrupcion	interrupciones, guardo ip y flags	int 21h		Lesson 2
<u>IRET</u>	iret	recupero registros y ip (utilizado cuando se finaliza una interrupción)	iret		Lesson 2
<u>SHL</u>	shl target, numero_de_veces	desplazamiento logico hacia la izquierda			Lesson 2
<u>PUSH</u>	push origen	Decrementa SP (puntero de pila) en 2 y lo mueve a la cima de la pila	PUSH BX ⇒ guarda BX en la cima de la pila		Lesson 1
<u>mov</u>	mov destino, origen	Mueve un byte o word de un sitio a otro	MOV CX, 112h ⇒ CX = 112h		Lesson 1

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	 imagenes	:≡ Lesson
<u>POP</u>	pop target	Guarda en target lo que se encuentre en la cima de la pila	pop BX ⇒ Guarda en BX lo que esté en la cima de la pila		Lesson 1
<u>ADD</u>	add destino, origen	suma 2 operandos	ADD CL, BL ⇒ CL = CL + BL ADD AL, 12h ⇒ AL = AL + 12h ADD CX, DX ⇒ CX = CX + DX		Lesson 1
<u>ADC</u>	adc destino, origen	Suma 2 operandos y el carryflag	ADC CL, BL ⇒ CL = CL + BL + CF ADC AL, 12h ⇒ AL = AL + 12h + CF ADC CX, DX ⇒ CX = CX + DX + CF		Lesson 1
<u>SBB</u>	sbb destino, origen	Resta 2 operandos y el carryflag	SBB CX, DX ⇒ CX = CX - DX - CF		Lesson 1
<u>SUB</u>	sub destino, origen	Resta 2 operandos	SUB CL, BL ⇒ CL = CL - BL SUB AL, 12h ⇒ AL = AL - 12h SUB CX, DX ⇒ CX = CX - DX		Lesson 1

Aa Nombre	≡ Formato	≡ Descripción	≡ Ejemplo	 imágenes	Lesson
<u>MUL</u>	mul origen	Multiplica 2 números SIN SIGNO. Puedo trabajar con 8 bits(ah/al) o con 16(AX) El resultado se guardará en AX si el resultado es de 8 bits El resultado se almacena concatenadamente DX y AX si es de 16 bits(word)	AX = 1234h BX = 1000h MUL BX ⇒ DX = 0123h, AX = 4000h AX = 17h BX = 10h MUL BL ⇒ AX = 0170h		Lesson 1
<u>IMUL</u>	imul origen	Multiplica 2 números CON SIGNO. Se almacena en AX si es de 8 bits y en DX y AX si es de tamaño palabra	AL = FEh = -2 BL = 12h = 18 iMUL BL ⇒ AX = FFDCh = -36		Lesson 1
<u>NEG</u>	neg target	Cambia el signo, sistema de representación Ca2	neg AL ⇒ si AL=F2h ahora AL=0eh		Lesson 1
<u>DEC</u>	dec target	Resta 1	ax=1234h dec AX ⇒ AX=1233h dec Ah ⇒ ah=11h		Lesson 1
<u>DIV</u>	div origen	AL se queda el cociente y AH el resto	AX = 0013h = 19 BL = 02h = 2 DIV BL ⇒ AH = 1, AL = 9		Lesson 1

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	 imagenes	:≡ Lesson
<u>IDIV</u>	idiv origen		AX = FFEDh = -19 BL = 02h = 2 – IDIV BL; AH = 1, AL = F7h = -9		Lesson 1
<u>INC</u>	inc target	Suma 1	ax=1234h inc AX ⇒ AX=1235h inc Ah ⇒ ah=13h		Lesson 1

- Saltos

- saltos incondicionales

jmp nombre

- saltos condicionales

j[condicion] nombre

J

N niega a lo que venta

Z cero

E igual

C carry

S signo

O overflow

P paridad

PE paridad par

PO paridad impar

despues de estos puedo poner E para poner igualdad

L less

A above

B bellow

G great

Ejmplos

JLE

JP

JNO

Memoria

width 1Byte

alto 1MB (20 bits)

cada entrada tiene 5 números hexadecimales (01234h) (0A0123h) siempre empieza por 0

Yo trabajo con 16 bits por lo que puedo segmentar

tienen 64K \Rightarrow 16bits por posicion en el segmento

data segment

estra segment

code segment

stack segment

la dirección será segmentada (lo mismo que offset)

segmento XXXX

efective addres XXX

XXXX:XXXX

data segment : offset

48 = 30

1234:2358

lea dx, op1

DS=1234H

DX=2358h (offset)

si al pulsar una tecla me da 38h, le tengo que restar 30h y me dice que la que he pulsado es el tecla 8

Para pasar de mayusculas a minusculas, le sumo 20h

para hacer un salto de linea

CR, retorno de carro (0Dh)

LF line feed (0Ah)

Tipos de strings

- longitud fija
 - la máxima longitud es definida para todos

P	E	P	E					A	N	T	O	N	I	O		R	O	S	A				
---	---	---	---	--	--	--	--	---	---	---	---	---	---	---	--	---	---	---	---	--	--	--	--

- longitud variable
 - Definimos la longitud de dos tipos de forma

– Delimiter character

*	P	E	P	E	*	A	N	T	O	N	I	O	*	R	O	S	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

– Explicit length

4	P	E	P	E	7	A	N	T	O	N	I	O	4	R	O	S	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Interrupciones

- Existen 2 tipos de interrupciones
 - interrupciones Hardware
 - interrupciones software

Al realizar una interrupción, guardamos el IP y flags registers

Tipos de interrupciones 21h

- 01h → El programa espera hasta pulsar una tecla, leer la tecla y la guarda en AL (mostrando por pantalla)
- 02h → Escribe un carácter que esté guardado en DL
- 08h → El programa espera hasta pulsar una tecla, lee la tecla y la guarda en AL (sin mostrar por pantalla) (PASSWORD)
- 09h → Muestra un string por pantalla ACABADO EN \$, que estuviese guardado en DS:DX
- 0Ah → Lee una cadena de caracteres y se guarda en DS:DX
- 4ch → error code

int 10h Video

- 00h → set video mode
- 02h → posicionar el cursor
- 06h → scroll up
- 07h → scroll down
- 09h → escribir un caracter y atributo
- 0ah → escribir un caracter y ultimo atributo
- 0eh → escribe un caracter y pasa al a siguiente columna

Ejemplo

00h

AL=1 (40x25)

AL=3 (80x25)

02h

dh= fila (0-24)

dl = columna (0-39)(0-79)

bh= pagina 0

06h/07h

al = numero de lineas to scroll 0

09h

bh=0 pagina

bl = 7; escribe en blanco

cx = 4; escribe 4 veces

al = 'A'

int 16h servicios de teclado

- 00h / 10h → lee un carácter, espera hasta leerlo (10h permita las teclas desde F1 - f12)
- 01h / 11h → Devuelve el estado del buffer
 - ZF=1 si buffer vacio
 - ZF=0 si una letra está en el buffer

Adicional:

DT

Define 10 bytes length data

DD

Define 32 bytes length data