



LAB 1

- kernel driver not installed (rc=-1908)

```
# /etc/init.d/vboxdrv setup
```

ensamblar y linkear

- masm /zi archivo.asm
- link /co archivo.obj

dosseg ⇒ Preparación del modo de ejecución

.model small ⇒ definición del modo de compilado

.stack 100h ⇒ reserva 100 bytes para la pila

.data ⇒ Declaración de datos

.code

mov ax, @data ⇒ establecen las direcciones de memoria de los datos

mov ds, ax

mov ah, 4ch ⇒ Preguntan por el fin del programa

int 21h

En cv para ver variables

- tabla db 1,2,3,4,5,6,7,8
- num1 db 22h

- num2 dw 1234h
wb num1(8 bits)
ww num2 (16 bits)
wb table L8

AX acumulador

BX base

CX contador

DX datos

DB define byte

dw define word

- **Pointer registers: 16bits**

Data registers:


- AX (AH, AL)
- BX (BH, BL)
- CX (CH, CL)
- DX (DH, DL)


Pointer registers:


- SP - Stack Pointer
- BP - Base Pointer
- SI - Source Index Register
- DI - Destination Index Register
- IP - Instruction Pointer



PC Contador de programa

Instrucciones

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	:≡ Lesson	 imagenes
<u>mov</u>	mov destino, origen	Mueve un byte o word de un sitio a otro	MOV CX, 112h ⇒ CX = 112h	Lesson 1	
<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	

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	≡ Lesson	 imagenes
<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>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	
<u>SBB</u>	sbb destino, origen	Resta 2 operandos y el carryflag	SBB CX, DX ⇒ CX = CX - DX - CF	Lesson 1	

Aa Nombre	≡ Formato	≡ Descripcion	≡ Ejemplo	≡ Lesson	 imagenes
<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>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	
<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	

Aa Nombre	≡ Formato	≡ Descripción	≡ Ejemplo	≡ Lesson	 imágenes
<u>DEC</u>	dec target	Resta 1	ax=1234h dec AX ⇒ AX=1233h dec Ah ⇒ ah=11h	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>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	
<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	
