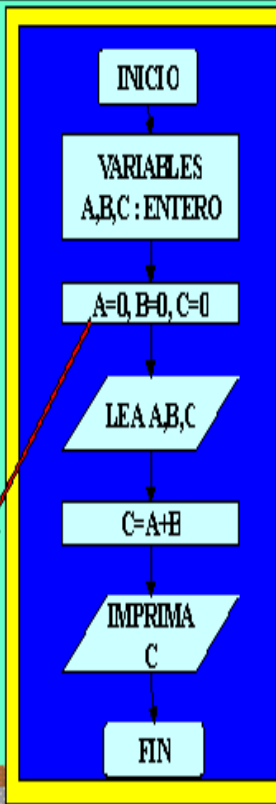
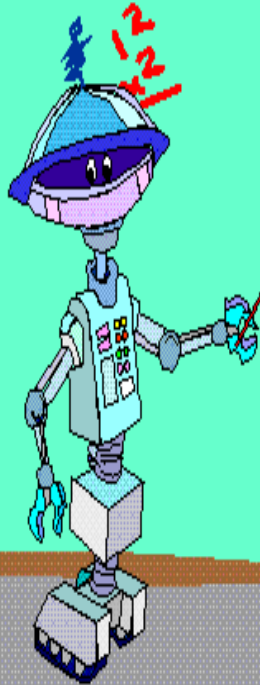


INSTRUCCIONES BÁSICAS



Instrucciones de Movimientos

DB = 8BITS
DW =16BITS
DD= 32BITS

✓ MOV destino, fuente

- Transfiere un byte o una palabra desde el operando fuente al operando destino.
- Destino: registro o elemento de memoria.
- Fuente: registro, elemento de memoria o valor inmediato.





✓ PUSH fuente

Coloca una palabra (2 bytes) en la pila.

- Decrementa el puntero de pila (SP) en 2.
- Transfiere el operando fuente a la cima de la pila.

✓ POP destino

Saca una palabra de la pila.

- Transfiere el elemento que se encuentra en la cima de la pila al operando destino.
- Incrementa el puntero de pila (SP) en 2.

✓ LEA destino, fuente

- Transfiere la dirección efectiva (desplazamiento) del operando fuente al operando destino.
- Fuente: Operando de memoria (byte o palabra).
- Destino: Registro de 16 bits. No se puede utilizar un registro de segmento.



- ✓ LDS destino, fuente

- Transfiere un puntero de 32 bits (dirección compuesta por segmento y desplazamiento) correspondiente al segundo operando.
- Destino: Debe ser un registro, pero no de segmento. El valor de segmento indicado por el segundo operando de la instrucción se carga en el registro DS.
- Fuente: Operando de memoria de doble palabra (32 bits).

✓ LES destino, fuente

- Igual que LDS, pero el segmento se transfiere al registro ES.

MOV EAX, 1

XOR EAX, EAX
INC EAX

Instrucciones Aritméticas

✓ ADD destino, fuente

- Suma los dos operandos. El resultado se almacena en el destino.
- Los dos operandos deben ser del mismo tipo (byte o palabra).

✓ ADC destino, fuente

- Suma los dos operandos, incrementando el resultado en 1 si la bandera de acarreo está activada.





✓ SUB destino, fuente

- Resta el operando fuente al operando destino. El resultado se almacena en el operando destino.
- Los dos operandos deben ser del mismo tipo (byte o palabra).

✓ SBB destino, fuente

- Resta el operando fuente al operando destino. Resta 1 al resultado si la bandera de acarreo está activada.

- ✓ MUL fuente

- Multiplica, sin considerar el signo, el acumulador (AL o AX) por el operando fuente.
- Si fuente es byte, se multiplica por AL y el resultado se almacena en AX.
- Si fuente es palabra, se multiplica por AX y el resultado se almacena en DX:AX (parte alta en DX y parte baja en AX).

✓ IMUL fuente

- Multiplica, considerando el signo, el acumulador (AL o AX) por el operando fuente.
- Si fuente es byte, se multiplica por AL y el resultado se almacena en AX.
- Si fuente es palabra, se multiplica por AX y el resultado se almacena en DX:AX (parte alta en DX y parte baja en AX).



✓ DIV fuente

- Divide, sin considerar el signo, AX o DX:AX entre fuente.
- Si fuente es byte, divide AX entre fuente. El cociente se almacena en AL y el resto en AH.
- Si fuente es palabra, divide DX:AX entre fuente. El cociente se almacena en AX y el resto en DX.

✓ IDIV fuente

- Divide, considerando el signo, AX o DX:AX entre fuente.
- Si fuente es byte, divide AX entre fuente. El cociente se almacena en AL y el resto en AH.
- Si fuente es palabra, divide DX:AX entre fuente. El cociente se almacena en AX y el resto en DX.

✓ INC destino

- Incrementa el operando destino en 1.

✓ DEC destino

- Decrementa el operando destino en 1.

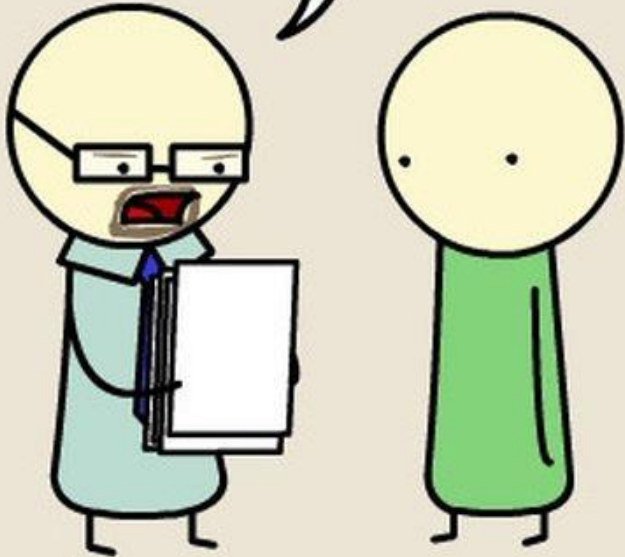
✓ NEG destino

- Niega el operando destino (Complemento a 2)



ASSEMBLY

DID YOU REALLY HAVE TO REDEFINE EVERY
WORD IN THE ENGLISH LANGUAGE?



Instrucciones
Lógicas



- ✓ AND destino, fuente
 - Operación lógica AND a nivel de bits.
- ✓ OR destino, fuente
 - Operación lógica OR a nivel de bits.
- ✓ XOR destino, fuente
 - Operación lógica XOR a nivel de bits.
- ✓ NOT destino
 - Operación lógica NOT. Complementa los bits del operando.

python

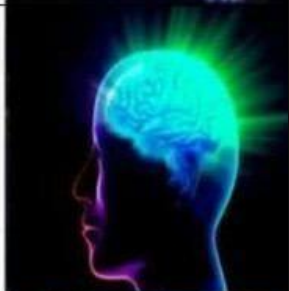
python

```
print("hello world")
```



java

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World")
    }
}
```



c++

```
#include<iostream>
using namespace std ;
int main()
{
    cout<<"hello world "<<endl;
    return 0;
}
```



Assembly :

```
global _main
extern _printf
section .text
_main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db 'Hello, World', 10, 0
```



Instrucciones Comparativas

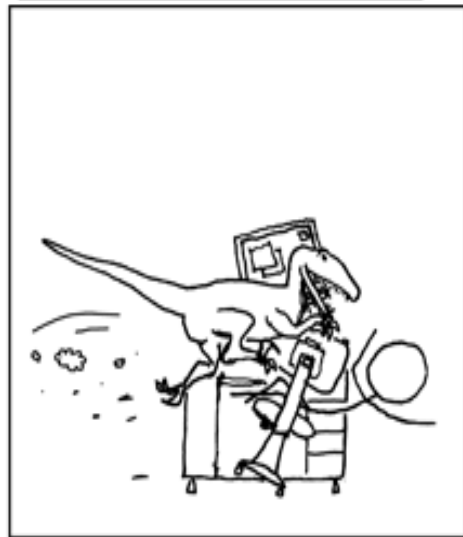
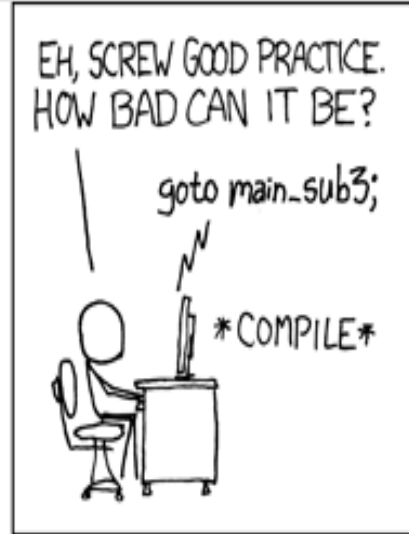
✓ TEST fuente1, fuente2

- Realiza la operación AND entre los dos operandos sin almacenar el resultado. Sólo actualiza las banderas del registro de estado.

✓ CMP fuente1, fuente2

- Lleva a cabo la operación de resta entre los dos operandos, sin almacenar el destino.
- Actualiza las banderas.





Instrucciones de Saltos

✓ JMP dir

- Salto incondicional a dir. El operando será normalmente una etiqueta mediante la cual se referencia la instrucción del programa donde se quiere realizar el salto.



- ✓ JE dir
 - Salto si igual ($Z=1$). Comúnmente se usa como resultado de CMP.
- ✓ JZ dir
 - Salto si igual ($Z=1$). Comúnmente se usa cuando un resultado da 0.
- ✓ JNE / JNZ dir
 - Salto si no igual ($Z=0$).
- ✓ JS dir
 - Salto si signo negativo ($S=1$).

- ```

test eax, eax
jnz @B010
ret 0

@B010:
mov al, byte ptr [edi]
neg eax
ref 0

@B020:
test edx, edx
jnz @B030
movzx eax, byte ptr [edi]
ret 0

@B030:
mov edi, edi
push esi
push ebx
push ecx
push edx
call @PLT@@@
add esp, 10h
ret 0

```



- ✓ JGE dir
  - Salto si el resultado de la última instrucción es mayor o igual que 0.
- ✓ JP dir
  - Salto si paridad par ( $P=1$ ).
- ✓ JNP dir
  - Salto si paridad impar ( $P=0$ ).
- ✓ JCXZ dir
  - Salto si  $CX=0$ .

✓ **LOOP etiqueta**

- Salta a etiqueta si el registro CX es distinto de 0. Además, decrementa el valor de CX.
- Esta instrucción se utiliza para implementar ciclos. Para ello, se inicializa CX con el número de iteraciones que debe realizar el ciclo y, posteriormente, se colocan aquellas instrucciones que se desean ejecutar de forma iterativa, delimitadas por la etiqueta y la instrucción LOOP.

✓ **LOOPE / LOOPZ etiqueta**

- Salta si la bandera Z es 1 y CX es distinto de 0. Decrementa CX.

✓ **LOOPNE / LOOPNZ etiqueta**

- Salta si la bandera Z es 0 y CX es distinto de 0. Decrementa CX.

✓ **INT n**

- Ejecuta el manejador de la interrupción especificada en el operando.





¿DUDAS?

