Introducción:

Este manual tiene como fin ayudar a comprender los aspectos técnicos del programa para poder facilitar el uso y funcionamiento del mismo, a continuación se explica, el ensamblador utilizado para esta práctica, los requisitos del ensamblador y la forma de poder ejecutar programas utilizando este ensamblador

Descripción del Emulador:

NASM:

El Netwide Assembler o Nasm, es un ensamblador libre para la plataforma Intel x86. Puede ser usado para escribir programas tanto de 16 bits como de 32 bits (IA-32). En el NASM, si se usan las bibliotecas correctas, los programas de 32 bits se pueden escribir de una manera tal para que sean portables entre cualquier sistema operativo x86 de 32 bits. El paquete también incluye un desensamblador, el NDISASM.

Historia:

El NASM fue escrito originalmente por Simon Tatham con ayuda de Julian Hall, y actualmente es desarrollado por un pequeño equipo en SourceForge que le hace mantenimiento. Fue lanzado originalmente bajo su propia licencia, pero más adelante fue cambiada por la licencia GNU Lesser General Public License, seguido de un número de problemas políticos causado por la selección de la licencia. La próxima versión del NASM, la 2.00, actualmente está siendo desarrollada bajo la bifurcación 0.99, e incluirá soporte para el x86-64 (x64/AMD64/Intel 64), junto con la respectiva salida de archivo objeto de 64 bits.

Características:

- El NASM puede generar varios formatos binarios en cualquier máquina, incluyendo COFF (y el ligeramente diferente formato Portable Executable usado porMicrosoft Windows), el a.out, ELF, Mach-O, y el formato binario nativo Minix. El NASM incluso define su propio formato binario, RDOFF, que es usado actualmente solamente por el proyecto del sistema operativo RadiOS).
- La variedad de formatos de la salida permite a uno portar los programas a virtualmente cualquier sistema operativo x86. Además, el NASM puede crear archivos binarios planos, usables para escribir boot loaders (cargadores de arranque), imágenes ROM, y varias facetas del desarrollo sistemas operativos. El NASM incluso puede correr en plataformas diferentes del x86, como SPARC y PowerPC, aunque no puede producir programas usables por esas máquinas.
- El NASM usa la tradicional sintaxis de Intel para el lenguaje ensamblador x86, mientras que otros ensambladores libres, como el ensamblador del GNU (GAS), utilizan la sintaxis de AT&T. También evita características como la generación automática de sobreescritura (override) de segmentos y la relacionada directiva *ASSUME* usada por el MASM y los ensambladores compatibles, pues estas pueden ser a menudo confusas -- los programadores deben seguir por sí mismos el contenido de los registros de segmento y la localización de variables a los que éstos se refieren

Ejemplo Comando de Nasm:

```
nasm -f elf Practica2.asm
ld -m elf_i386 -s -o practica Practica2.o
./practica
```

Ejemplo Comando de Ejecución de NASM para DosBox

```
nasm Practica2.asm -fbin -o practica.com
dosbox ./practica.com -exit
```

DosBox:
Es un emulador que recrea un entorno similar al sistema DOS con el objetivo de poder ejecutar programas y videojuegos originalmente escritos para el sistema operativo MS-DOS de Microsoft en ordenadores más modernos o en diferentes arquitecturas (Como Power PC). También permite que estos juegos funcionen en otros sistemas operativos como GNU/Linux. Fue porque Windows XP ya no se bas en MS-DOS y pasó a basarse a Windows NT.

Descripción del Sistema Operativo:

Sistema Operativo utilizado: Debian 8.5 - Jessie

Debian: Debian o Proyecto Debian (en inglés: Debian Project) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre. El sistema se encuentra precompilado, empaquetado y en formato deb para múltiples arquitecturas de computador y para varios núcleos.
El proyecto Debian fue anunciado inicialmente 1993 por Ian Murdock. El nombre Debian proviene de la combinación del nombre de su entonces novia Deborah y el suyo, por lo tanto, Deb (orah) e Ian. Debian 0.01 fue lanzado el 15 de septiembre de 1993, y la primera versión estable fue hecha en 1996.

Requisitos del sistema:

Tipo de instalación	RAM (mínimo)	RAM (recomendado)	Disco duro
Sin escritorio	128 Megabytes	512 Megabytes	2 Gigabytes
Con escritorio	256 Megabytes	1 Gigabyte	10 Gigabyte

Requerimientos de la aplicación:

1. Tener instalado gcc (Compilador de C)
2. Tener instalado dosbox (Segunda forma de poder compilar el programa)
3. Una computadora de 64 bits
4. Un sistema operativo base linux de 64 bits

Pasos para poder compilar y ejecutar un programa:

1. Generar el archivo .asm
2. Compilar el archivo .asm con el siguiente código: **nams -f elf64 NombreArchivo.asm**, este comando traducirá el código en el archivo .asm a un código binario que el procesador pueda entender, como resultado de la ejecución de este código se tendrá un archivo .o donde estará el código binario que el procesador entiende.
3. Generar el ejecutable en base al archivo .o generado en el paso anterior, para esto se introduce lo siguiente en consola: **ld NombreEjecutable NombreArchivo.o**
4. Ejecutar el ejecutable generador el paso anterior se hará de la siguiente forma: **./NombreEjecutable**
5. En consola se podrá ver el resultado del programa escrito

Pasos para poder comilar y ejecutar un programa con dosbox:

1. Generar el archivo .asm
2. Compilar el .asm con el siguiente código: **nasm NombreArchivo.asm -fbin -o NombreCom.com,** este comando traducirá el .asm a código binario entendible por dosbox, como resultado de la ejecución se genera un archivo .com que será el archivo que se compilara en dosbox
3. Ejecutar el siguiente comando en la ruta donde se haya generado el archivo .com: **dosbox ./NombreCom.com -exit**, esto hará que dosbox compile el .com y ejecute el programa, el -exit es para indicarle a dosbox que cuando termine la ejecución se cierre

Código de la solución (Fragmentos más relevantes):

Recepción de la ecuación:

```
Ingresar_Ecuacion:

        call Limpiar_Numeros

        mov ah,00h
        mov al,03h
        int 10h

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgGrado
        int 21h

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,8h
        int 21h

        cmp al,34h
        ja Error_Grado

        mov [Grado],al

        mov ah,9h
        mov dx,Grado
        int 21h

        mov al,[Grado]
        cmp al,34h
        je Grado4
        cmp al,33h
        je Grado3
        cmp al,32h
        je Grado2
        cmp al,31h
        je Grado1
```

```
Grado4:
        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,Msg4
        int 21h

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgA
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoA],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Na],ax

        mov ah,9h
        mov dx,MsgB
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoB],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Nb],ax

        mov ah,9h
        mov dx,MsgC
        int 21h

        call recibir_numero
```

```
mov ax,[Signo]
mov [SignoC],ax

call AsciiToDecimalNum

mov ax,[Resultado]
mov [Nc],ax

mov ah,9h
mov dx,MsgD
int 21h

call recibir_numero

mov ax,[Signo]
mov [SignoD],ax

call AsciiToDecimalNum

mov ax,[Resultado]
mov [Nd],ax

mov ah,9h
mov dx,MsgE
int 21h

call recibir_numero

mov ax,[Signo]
mov [SignoE],ax

call AsciiToDecimalNum

mov ax,[Resultado]
mov [Ne],ax

mov ah,9h
mov dx,SaltoLinea
int 21h

mov ah,9h
mov dx,MsgAlmacenada
int 21h

mov ah,8h
```

```
        int 21h

        jmp imprimir_menu2

Grado3:
        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,Msg3
        int 21h

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgA
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoA],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Na],ax

        mov ah,9h
        mov dx,MsgB
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoB],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Nb],ax

        mov ah,9h
```

```
        mov dx,MsgC
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoC],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Nc],ax

        mov ah,9h
        mov dx,MsgD
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoD],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Nd],ax

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgAlmacenada
        int 21h

        mov ah,8h
        int 21h

        jmp imprimir_menu2

Grado2:

        mov ah,9h
        mov dx,SaltoLinea
        int 21h
```

```
mov ah,9h
mov dx,Msg2
int 21h

mov ah,9h
mov dx,SaltoLinea
int 21h

mov ah,9h
mov dx,MsgA
int 21h

call recibir_numero

mov ax,[Signo]
mov [SignoA],ax

call AsciiToDecimalNum

mov ax,[Resultado]
mov [Na],ax

mov ah,9h
mov dx,MsgB
int 21h

call recibir_numero

mov ax,[Signo]
mov [SignoB],ax

call AsciiToDecimalNum

mov ax,[Resultado]
mov [Nb],ax

mov ah,9h
mov dx,MsgC
int 21h

call recibir_numero

mov ax,[Signo]
mov [SignoC],ax

call AsciiToDecimalNum
```

```
        mov ax,[Resultado]
        mov [Nc],ax

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgAlmacenada
        int 21h

        mov ah,8h
        int 21h

        jmp imprimir_menu2

Grado1:

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,Msg1
        int 21h

        mov ah,9h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,MsgA
        int 21h

        call recibir_numero

        mov ax,[Signo]
        mov [SignoA],ax

        call AsciiToDecimalNum

        mov ax,[Resultado]
        mov [Na],ax

        mov ah,9h
```

```
                mov dx,MsgB
                int 21h

                call recibir_numero

                mov ax,[Signo]
                mov [SignoB],ax

                call AsciiToDecimalNum

                mov ax,[Resultado]
                mov [Nb],ax

                mov ah,9h
                mov dx,SaltoLinea
                int 21h

                mov ah,9h
                mov dx,MsgAlmacenada
                int 21h

                mov ah,8h
                int 21h

                jmp imprimir_menu2

        Error_Grado:

                mov ah,9h
                mov dx,SaltoLinea
                int 21h

                mov ah,9h
                mov dx,MsgErrorGrado
                int 21h

                mov ah,8h
                int 21h

                jmp imprimir_menu2
ret
```

Imprimir Eucación:

```
Imprimir_Funcion:
        mov ah,00h              ; limpiar pantalla
        mov al,03h
        int 10h                 ;llame a la interrupción de video

        mov ah,09h
        mov dx,MsgFuncion
        int 21h

        mov al,[Grado]
        cmp al,34h
        je ImprimirGrado4
        cmp al,33h
        je ImprimirGrado3
        cmp al,32h
        je ImprimirGrado2
        cmp al,31h
        je ImprimirGrado1
        jmp Salir_funcion

        ImprimirGrado4:
                mov ah,09h
                mov dx,SaltoLinea
                int 21h

                mov ah,9h
                mov dx,SignoA
                int 21h

                mov ax,[Na]
                mov [Resultado],ax

                call DecimalToAscii

                mov ah,9h
                mov dx,MsgX
                int 21h

                mov ah,9h
                mov dx,Potencia
                int 21h

                mov ah,9h
                mov dx,Cuatro
```

```
int 21h

mov ah,9h
mov dx,SignoB
int 21h

mov ax,[Nb]
mov [Resultado],ax

call DecimalToAscii

mov ah,9h
mov dx,MsgX
int 21h

mov ah,9h
mov dx,Potencia
int 21h

mov ah,9h
mov dx,Tres
int 21h

mov ah,9h
mov dx,SignoC
int 21h

mov ax,[Nc]
mov [Resultado],ax

call DecimalToAscii

mov ah,9h
mov dx,MsgX
int 21h

mov ah,9h
mov dx,Potencia
int 21h

mov ah,9h
mov dx,Dos
int 21h

mov ah,9h
mov dx,SignoD
```

```
        int 21h

        mov ax,[Nd]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,9h
        mov dx,MsgX
        int 21h

        mov ah,9h
        mov dx,SignoE
        int 21h

        mov ax,[Ne]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,8h
        int 21h

        jmp imprimir_menu2
ret

ImprimirGrado3:
        mov ah,09h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,SignoA
        int 21h

        mov ax,[Na]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,9h
        mov dx,MsgX
        int 21h

        mov ah,9h
        mov dx,Potencia
```

```
int 21h

mov ah,9h
mov dx,Tres
int 21h

mov ah,9h
mov dx,SignoB
int 21h

mov ax,[Nb]
mov [Resultado],ax

call DecimalToAscii

mov ah,9h
mov dx,MsgX
int 21h

mov ah,9h
mov dx,Potencia
int 21h

mov ah,9h
mov dx,Dos
int 21h

mov ah,9h
mov dx,SignoC
int 21h

mov ax,[Nc]
mov [Resultado],ax

call DecimalToAscii

mov ah,9h
mov dx,MsgX
int 21h

mov ah,9h
mov dx,SignoD
int 21h

mov ax,[Nd]
mov [Resultado],ax
```

```
        call DecimalToAscii

        mov ah,8h
        int 21h

        jmp imprimir_menu2
ret

ImprimirGrado2:
        mov ah,09h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,SignoA
        int 21h

        mov ax,[Na]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,9h
        mov dx,MsgX
        int 21h

        mov ah,9h
        mov dx,Potencia
        int 21h

        mov ah,9h
        mov dx,Dos
        int 21h

        mov ah,9h
        mov dx,SignoB
        int 21h

        mov ax,[Nb]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,9h
        mov dx,MsgX
```

```
        int 21h

        mov ah,9h
        mov dx,SignoC
        int 21h

        mov ax,[Nc]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,8h
        int 21h

        jmp imprimir_menu2
ret

ImprimirGrado1:
        mov ah,09h
        mov dx,SaltoLinea
        int 21h

        mov ah,9h
        mov dx,SignoA
        int 21h

        mov ax,[Na]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,9h
        mov dx,MsgX
        int 21h

        mov ah,9h
        mov dx,SignoB
        int 21h

        mov ax,[Nb]
        mov [Resultado],ax

        call DecimalToAscii

        mov ah,8h
        int 21h
```

```
                jmp imprimir_menu2
        ret

        Salir_funcion:
                mov ah,8h
                int 21h
                jmp imprimir_menu2
ret
```