



Práctica 1 - ASM i386

Organización y Arquitectura de Computadoras

Docente: Garcia Rocha Jose Isabel

Alumno:

- Chaparro Herrera Hugo Giovanni - 2200357

Tijuana, Baja California a 29 de agosto de 2025



Objetivo

Conocer y dominar el uso de una máquina virtual con sistema operativo Linux analizando sus recursos de hardware y software, para conocer sus capacidades y limitaciones de forma organizada y responsable.

Desarrollo

section .data

Se utiliza para la declaración de valores constantes o inicializados previo a ejecución. Estos valores no cambian en tiempo de ejecución. Comúnmente, en esta sección se declaran valores como un mensaje y la longitud del mismo.

```
section .data
    msg db 'Hello, world!', 0x0A
    len equ $-msg
```

section .bss

A diferencia de la sección anterior, aquí se declaran variables o espacios de memoria cuyo contenido podemos modificar durante la ejecución del programa.

```
section .bss
    n_lineas equ 10
    buffer resb 10
```

En el ejemplo anterior, se declaró una variable llamada “n_lineas” que contendrá el valor 10 decimal y un buffer que contendrá 10 bytes de tamaño.

section .text

En esta sección se colocan las sentencias que le indicarán al ensamblador que hacer, comúnmente en esta sección se inicia con la declaración ‘global _start’. la cual, le indica al kernel donde empieza la ejecución del programa.

```
section .text
    global _start

_start:
    ; código a ejecutar
```



directiva global

Se utiliza **global** con la finalidad de hacer visible el símbolo a **-ld** para que pueda ser referenciado con otros archivos. Comúnmente se realiza para poder modularizar código en diferentes archivos.

Comandos basicos de linux

1. **ls**: Muestra la información de los archivos contenidos dentro de un directorio.
2. **sudo**: Permite al usuario ejecutar comandos con privilegios de administrador/superuser.
3. **pwd**: Muestra el directorio actual.
4. **mkdir**: Crea un nuevo directorio.
5. **cd**: Sirve para navegar entre directorios.
6. **rmdir**: Elimina directorios vacíos de la lista de directorios.
7. **cp**: copia archivo de un directorio a otro.
8. **mv**: Renombra y reemplaza archivos.
9. **rm**: Borra archivos.
10. **uname**: Muestra la información del sistema operativo.
11. **locate**: Busca un archivo en la base de datos.
12. **touch**: Crea archivos.
13. **ln**: Crea accesos directos a otros archivos.
14. **cat**: Muestra el contenido de un archivo en la terminal.
15. **clear**: Limpia la terminal.
16. **ps**: Muestra los procesos en la terminal.
17. **man**: Abre el manual de comandos de Linux.
18. **grep**: Busca una cadena en concreto en una salida.
19. **echo**: Imprime una cadena.

Extensiones de archivos relacionados con ensamblador

.asm

El archivo con extensión **.asm** contiene el código fuente en ensamblador, aquí es donde el programador escribe las instrucciones a ejecutar en lenguaje ensamblador.

.o/.obj

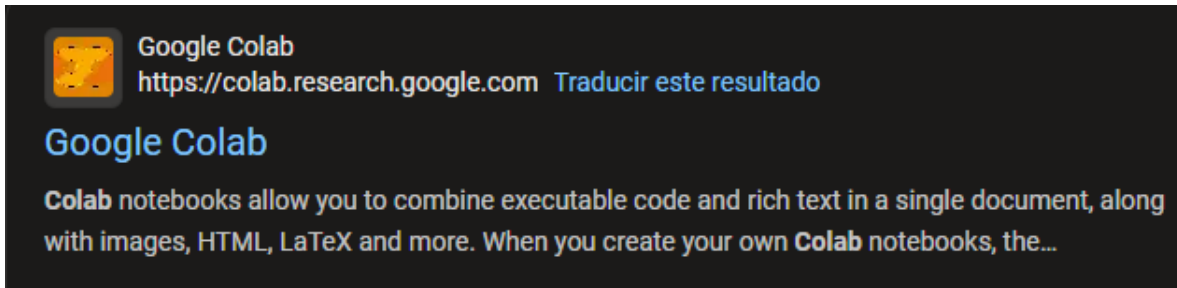
El archivo con extensión **.o** contiene el archivo tipo objeto, este es el resultado del proceso de ensamblaje del archivo que contiene el código fuente en lenguaje ensamblador. Dentro de este archivo se aloja el código en lenguaje máquina que se utilizara para comunicarse con el computador.

.exe

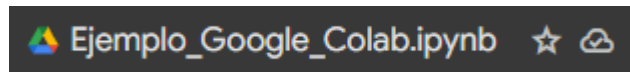
El archivo con extensión **.exe** contiene el resultado final tras la vinculación de los archivos objeto. Este es el archivo que se ejecuta para poder correr el programa.

Tutorial sobre el uso de Google Colab con ensamblador

1. Entrar a Google Colab.



2. Crear un archivo nuevo con nombre “Ejemplo_Google_Colab.ipynb”.



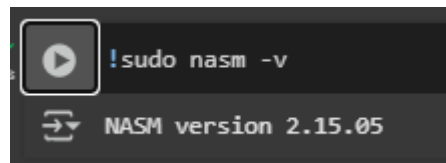
3. Instalar nasm.

Para la realización de este paso, es necesario colocar la siguiente línea en una nueva línea de código. `!sudo apt-get install nasm`

```
[ ] !sudo apt-get install nasm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nasm
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 375 kB of archives.
After this operation, 3,345 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 nasm amd64 2.15.05-1 [375 kB]
Fetched 375 kB in 0s (1,353 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line 1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package nasm.
(Reading database ... 126371 files and directories currently installed.)
Preparing to unpack ../nasm_2.15.05-1_amd64.deb ...
Unpacking nasm (2.15.05-1) ...
Setting up nasm (2.15.05-1) ...
Processing triggers for man-db (2.10.2-1) ...
```

4. Revisar la versión de nasm instalado.

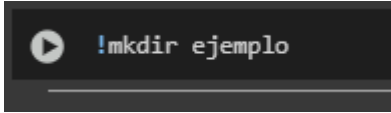
De igual manera, en una nueva línea de código es necesario colocar la siguiente instrucción: `!sudo nasm -v` Con la finalidad de verificar la versión nasm instalada.





5. Crear una carpeta cuyo nombre sea “ejemplo”.

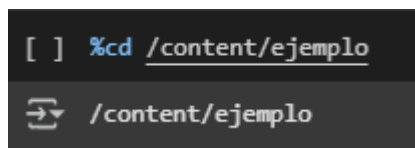
Para poder realizar este paso, se necesita ejecutar la siguiente instrucción en una línea de código: `mkdir ejemplo`



```
!mkdir ejemplo
```

6. Cambiar del directorio a `/content`.

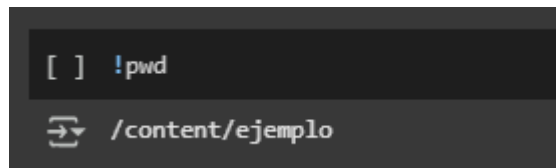
Para cambiar de directorio, es necesario ejecutar la siguiente instrucción: `%cd /content/ejemplo`



```
[ ] %cd /content/ejemplo  
↔ /content/ejemplo
```

7. Comprobar que este cambio se haya realizado.

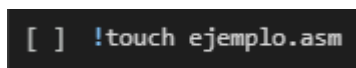
Para comprobar este dato, solamente se necesita ejecutar: `!pwd`



```
[ ] !pwd  
↔ /content/ejemplo
```

8. Crear un archivo con extensión `.asm` y con nombre “ejemplo”.

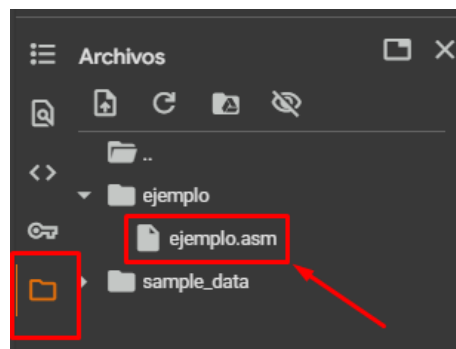
Es necesario ejecutar la siguiente instrucción `!touch ejemplo.asm`



```
[ ] !touch ejemplo.asm
```

9. Una vez creado el archivo, ingresar el siguiente código.

Para poder abrir el archivo, es necesario desplazarnos a la sección correspondiente de la carpeta, la cual está ubicada en la parte izquierda.





Conclusiones

```
global _start
section .text

_start:
    ; sys_write(stdout, message, length)
    mov eax,4
    mov ebx, 1
    mov ecx, message
    mov edx, length
    int 80h

    ; sys_exit (return code)
    mov eax, 1                ; sys_exit syscall
    mov ebx, 0                ; return 0
    int 80h

section .data
    message: db 'Hello, world', 0x0A ; mensaje y linea nueva
    length: equ 4-message          ; longitud de la cadena
```

Conclusiones

La práctica permitió familiarizarse con los fundamentos del lenguaje ensamblador i386 y el entorno Linux, consolidando el conocimiento sobre la estructura de un programa en ASM, incluyendo las secciones .data, .bss, .text y el uso de directivas como global. Además, se reforzó el manejo básico de comandos en Linux, esenciales para la navegación, edición y ejecución de archivos en un sistema operativo tipo Unix.

El uso de Google Colab como entorno de desarrollo demostró ser una herramienta accesible y funcional para compilar y ejecutar código ensamblador, facilitando la comprensión del flujo de trabajo desde la creación del archivo .asm hasta su ejecución como .exe. A pesar de las dificultades iniciales con el formato ELF y el proceso de ensamblaje, la práctica evidenció que con instrucciones claras y experimentación guiada, es posible superar barreras técnicas y adquirir habilidades prácticas en programación de bajo nivel.

Dificultades durante el desarrollo

Durante la realización de la práctica, la única dificultad que se presentó fue la identificación de cada parte de la instrucción de ensamblaje y enlazado de archivos. Debido a que, desconocía a qué se diferenciaba al momento de escribir `elf` antes de hacer el ensamblaje del mismo.

Sin embargo, las instrucciones proporcionadas por el profesor fueron bastante concretas al momento de realizar cada paso de la impresión del hola mundo.



Referencias Bibliográficas

- [02. NASMx86: partes del código fuente](#)
- [Guide to x86 Assembly](#)
- [Using as - Assembler Directives](#)
- [NASM Assembly Language Tutorials - asmtutor.com](#)
- [Assembly - System Calls](#)