Arquitecturas Intel 64 e IA-32

Organización del Computador II

20 de marzo de 2023

Toolchain

La presente guía introducirá el toolchain que estaremos usando en la materia. La idea es que comprueben y comprendan el funcionamiento básico de las herramientas de compilación y debugging.

Ejercicio 1 Creación de la cuenta individual de GIT de la materia

Cada estudiante deberá tener una cuenta del Departamento de Computación que le dará acceso a un cuenta de email y al Git del DC.

Si ya poseen la cuenta, no es necesario crear otra. Verifiquen que puedan acceder al gitlab: `https://git.exactas.uba.ar/users/sign_in`

Ejercicio 2 Instalación de Software requerido para la materia

Les pedimos que se armen un entorno de trabajo con el software que precisarán para la materia.
Pueden encontrar algunos pasos orientativos en el campus bajo la solapa "Material de Cursada".

a) Linux

b) GCC

c) GDB (y gdb-dashboard!)

d) NASM

e) valgrind

f) git

g) Editor de texto de preferencia (por ejemplo, vscode, vim u otro)

Más adelante, para la 2da parte de la materia, necesitaremos instalar QEMU, pero por el momento no es necesario.

Ejercicio 3 Hola mundo

a) Escriban el programa en ASM "Hola Mundo"presentado en la clase práctica. Compílenlo y pruébenlo en sus computadoras acorde a las indicaciones provistas en clase.

b) Modifiquen el programa para que imprima su nombre, apellido, número de libreta y alguna frase que quieran.

c) Prueben el programa y suban el código de este último al repo git. Recuerden que pueden tener un archivo .gitignore en el repo para filtrar y que únicamente se suban el código del programa y no archivos objetos o ejecutables. En este link hay archivos gitignore para los entornos y lenguajes más usados: `https://github.com/github/gitignore`

Ejercicio 4 GDB test run

Esta es una actividad, para explorar un poco **gdb** y **gdb-dashboard**.

GDB es un debugger de código, les permite ir ejecutando el código línea por línea e ir inspeccionado las variables, valores de los registros, estado de la memoria, etc.

Para iniciar el *gdb* escriban en la terminal: `gdb` *`nombre_archivo_ejecutable`*

Dentro de gdb tienen un prompt.

- Para poner un breakpoint en la primer linea pongan: **b 1** o **break 1**
- Con **r** o **run** inician la ejecución
- Con **n** o **next** avanzan a la siguiente línea
- Con **c** o **continue** continúan la ejecución
- Con **info reg** ***nombre_registro*** pueden inspeccionar los valores de los registros. Por ejemplo, **info reg rax** , otro ejemplo: **info reg eflags**. Si instalaron el gdb-dashboard, ya tienen esa información disponible.
- Con **q** o **quit** pueden salir de gdb

a) Escriba un programa que sume dos números de 8 bits. Recuerden agregar la llamada al sistema operativo para indicar que termina el programa con la syscall sys_exit.

b) Investigue que pasaría si el resultado de la suma supera los 8 bits usando **GDB**.

c) Ahora, escriba un programa que sume dos números de 64 bits.

d) Pruebe su código sumando combinaciones de positivos, negativos y ceros.

e) Busque qué tipo de representación numérica para enteros usa Intel. ¿En qué casos se activarían los flags de Zero, Carry y Overflow? Arme algún ejemplo numérico con el que logre activar estos flags. Compare el caso 8 bits y 64 bits. Inspeccione con gdb el registro EFLAGS y los registros que utiliza en la suma.

f) Agregue a su programa una condición que imprima en pantalla el resultado y si hubo overflow. Recuerde verificar los flags.

g) Recuerde guardar su trabajo en el repo git