Arquitecturas Intel 64 e IA-32

Organización del Computador II

21 de marzo de 2024

Enunciado nro. 1

En esta actividad, vamos a trabajar con el primer y segundo volumen de los manuales Intel® 64 and IA-32 Architectures Software Developer's Manual. Nuestro objetivo de aprendizaje es familiarizarnos con los manuales, buscar características fundamentales de las arquitecturas Intel® 64 e IA-32 y entender las explicaciones de algunas de las instrucciones básicas de la programación en el lenguaje ensamblador de Intel.

Pueden descargar los manuales desde los links:

- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2: Instruction Set Reference, A-Z

1. Arquitectura básica IA-32 e Intel 64

En esta sección, vamos a trabajar con el primer volumen de los manuales de Intel para entender la arquitectura básica de los procesadores Intel.

Ejercicio 1 IA-32 Entorno de ejecución
Busquen en el manual la sección ***3.2 Overview of the basic execution environment*** en *Vol. 1 3-2*. En base a lo que dice la sección ***Address Space*** y la figura *3-1 IA-32 Basic Execution Environment for Non-64 bit Modes*. Indiquen:

a) ¿Cuál es el tamaño en bits de una dirección de memoria en la arquitectura IA-32 y cuál es la unidad más pequeña que podemos direccionar?

b) ¿Cuántos registros de propósito general hay en IA-32 y que tamaño tienen? Pueden también consultar la sección ***3.4.1 General Purpose Registers***

c) Busquen en el manual que guarda el registro EIP (Instruction Pointer) e indiquen su tamaño en bits. ¿Por qué motivo creen que el EIP tiene ese tamaño en bits?

Ejercicio 2 Flags
Busquen en el manual la sección ***3.4 BASIC PROGRAM EXECUTION REGISTERS*** en *Vol. 1 3-10*. Indiquen:

a) Busquen en la sección del manual qué guarda el registro EFLAGS e indiquen su tamaño en bits.

b) En el formato del registro, busquen los siguiente bits e indiquen para qué son y en qué posición del registro están almacenados:
Flag de Overflow, Flag de Signo, Flag de Interrupciones.

c) Indiquen si en la arquitectura Intel 64 se usa el mismo registro. En caso que sea otro, indiquen su tamaño y la relación tendría con el de IA-32.

Ejercicio 3 Stack y llamadas a función
Busquen en el manual la sección ***Overview of the basic execution environment*** en *3-2 Vol. 1*. En el párrafo donde menciona la pila (Stack), en la página *3-4 Vol. 1*. Indiquen:

a) ¿Para qué es necesaria la pila? ¿Donde está ubicada?

Investiguemos un poco más como maneja la pila Intel y qué debemos hacer nosotros como programadoras y programadores. Para eso, vayan a la sección ***6.2 Stacks*** en *Vol. 1 6-1*. Observen el dibujo de la pila *Figure 6-1. Stack Structure*.
Luego, en las secciones ***6.2.4.1 Stack-Frame Base Pointer*** y ***6.2.4.2 Return Instruction Pointer*** van a encontrar información sobre los registros ESP y EBP. Expliquen con sus palabras:

a) ¿Para qué sirven los registros ESP y EBP?¿Que consideraciones debemos tener al trabajar con cada uno ellos?

b) En el primer parrafo de la sección ***6.2.4.2 Return Instruction Pointer***, ¿Qué registro se pushea en la pila al hacer un CALL? Discutan con sus compañeros por qué creen que ocurre eso

c) En el primer parrafo de la sección ***6.2.4.2 Return Instruction Pointer***, ¿Qué ocurre al hacer un RET? Discutan con sus compañeros por qué creen que ocurre eso

d) En el segundo parrafo de la sección ***6.2.4.2 Return Instruction Pointer***, ¿Qué debe asegurarse el programador antes de llamar a un RET cuando esta escribiendo una subrutina? ¿Cómo lo asegura?

e) ¿Cuál es el ancho de la pila en modo 32 bits y en 64 bits? (tamaño del dato de PUSH y POP)

f) Luego de responder las preguntan anteriores, discutan en grupo si el EBP podría ser usado para guardar datos que no sean la base de la pila. ¿Qué opinan?

2. Set de Instrucciones IA-32 e Intel 64

Ya hemos explorado la arquitectura básica de Intel. En esta sección vamos a conocer algunas de las instrucciones más básicas.

Ejercicio 4 Set de instrucciones En el segundo volumen del manual ***Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2: Instruction Set Reference, A-Z*** busquen las siguientes instrucciones:

a) DEC, ADD, MOV, JZ, JE

Expliquen con sus palabras

a) Observen el formato de la instrucción y respondan, ¿cuántos operandos recibe, de qué tipo son y qué tamaño tienen?. Por ejemplo, 2 operandos de los tipo registro-registro de 8 a 64 bits, memoria-registro, memoria-memoria

b) ¿Qué hace cada instrucción?

c) Den uno o más ejemplos de su uso (traten que varíen los operandos y tamaño de dato leído). Por ejemplo, SUB EAX, 0x00000001 o ADD RAX, 1

d) ¿Qué diferencia existe entre JZ y JE?

Los jumps figuran en la sección ***Jcc—Jump if Condition Is Meet*** del segundo volumen.

Adicionalmente pueden apoyarse en sitios de Internet que sumarizen la información para ayudarse a responder las preguntas sobre las instrucciones.

Checkpoint 1

3. Herramientas de desarrollo

Ejercicio 5 Toolchain

En este ejercicio vamos a tener un primer acercamiento a algunas de las herramientas que utilizaremos en el curso.

Nos servirá también para verificar que el entorno de desarrollo está correctamente instalado.

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`

Instalación de Software requerido para la materia

Les pedimos que se armen un entorno de trabajo con el software que precisarán para la materia (en las máquinas de los labos *deberían* estar todas las herramientas instaladas).

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.

Hola mundo
Se pide:

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`

d) Debuggear el código con gdb, frenando su ejecución en la línea de la primera *syscall* y verificar el valor de los registros.
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

e) (Opcional) Escribir un `Makefile` para compilar y linkear el código.

Checkpoint 2