# PRÁCTICA 1: SUM1TON

+shell ls -l sum1toN_gdb_asm.txt

+file sum1toN  ->abrir archivo en gdb

Reading symbols from sum1toN...

+info sources  ->información de las fuentes

/home/eduardo/Descargas/sum1toN:

/home/eduardo/Descargas/sum1toN.s

+b _start ->break point en el que comenzar el programa

Punto de interrupción 1 at 0x1000: file sum1toN.s, line 17. ->_start está en la línea 17

+run ->ejecutar programa

Starting program: /home/eduardo/Descargas/sum1toN_

Breakpoint 1, _start () at sum1toN.s:17

+ptype n ->imprimir en pantalla el tipo de n

type = <data variable, no debug info> ->nos muestra que n es una variable

+p n

'n' has unknown type; cast it to its declared type

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+p (int) n ->imprimir en pantalla el valor de n, en el que hay que castearlo a entero al ser una etiqueta

$1 = 5

+n  ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+x &n  ->examinar n (dirección de memoria)

0x56558000:  0x00000005

+x /1bw &n  ->examinar n en formato

Eduardo Ezponda Igea

0x56558000:  0x00000005

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+p $ecx ->imprimir el valor del registro ecx

$2 = 9

+p $edx ->imprimir el valor del registro edx

$3 = 3

+until

+info registers ->imprimir el valor de los registros

| | | |
|---|---|---|
| eax | 0xf7ffda40 | -134227392 |
| ecx | 0xc | 12 |
| edx | 0x3 | 3 |
| ebx | 0xf7ffd000 | -134230016 |
| esp | 0xffffd1e0 | 0xffffd1e0 |
| ebp | 0x0 | 0x0 |
| esi | 0xffffd1ec | -11796 |
| edi | 0x56556000 | 1448435712 |
| eip | 0x5655600d | 0x5655600d <bucle+2> |
| eflags | 0x206 | [ PF IF ] |
| cs | 0x23 | 35 |
| ss | 0x2b | 43 |
| ds | 0x2b | 43 |
| es | 0x2b | 43 |
| fs | 0x0 | 0 |
| gs | 0x63 | 99 |

+layout split ->ensamblador (direcciones de memoria)

+focus cmd ->pantalla abajo

Focus set to cmd window.

+focus src ->pantalla arriba

Focus set to src window.

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

+n ->con el comando n(next) pasamos a la siguiente línea del programa

[Inferior 1 (process 15290) exited with code 017]

+start ->comenzar de nuevo el programa

Función «main» no definida.

Conteste «y» o «[n]».

Punto de interrupción temporal 2 (-qualified main) pendiente.

Starting program: /home/eduardo/Descargas/sum1toN_

Breakpoint 1, _start () at sum1toN.s:17

+exit ->salir del gdb

# PRÁCTICA: CUESTIONARIO

## 20.1.1 Cuestiones teóricas:

1. Cuál es la principal diferencia entre el lenguaje ensamblador AT&T y el propio de Intel.

La principal diferencia es que en el lenguaje ensamblador AT&T en las operaciones con registros, se encuentra el primer lugar la fuente, y a continuación, el destino.

Sin embargo, en el propio Intel, se encuentra primero el destino, y a su derecha

la fuente.

Ejemplo:

Intel: add %ecx, %edx; ##ecx = ecx + edx

Lenguaje ensamblador AT&T: add %ecx, %edx; ## edx = edx + ecx

Como se puede observar, en el primer caso el resultado se guardará en el registro c, mientras que en el caso del lenguaje ensamblador AT&T en el registro d.

2. ¿Qué fases comprende el toolchain?

Se conoce por Toolchain a los distintos programas o herramientas que intervienen en la obtención de una
aplicación en lenguaje máquina a partir de módulos en lenguajes símbolicos como C, ensamblador y
módulos librería.
Las 3 herramientas principales en el proceso de traducción son : el compilador, el ensamblador y el linker.

3. Lista las herramientas de desarrollo a utilizar durante la realización de las prácticas mediante los dos procesadores utilizados.

Gcc, gdb, ensamblador, linker.

4. Libro: Programming from the Ground-Up
a. Qué es GNU/Linux

Linux es una familia de sistemas operativos tipo Unix compuesto por software libre y código abierto. Aunque se conoce principalmente como linux, realmente éste solo es el núcleo o el kernel, mientras que el sistema completo se basa en una gran cantidad de componentes del proyecto GNU.

Eduardo Ezponda Igea

b. Qué es GNU

GNU es un sistema operativo el cual mantiene la libertad de los propios usuarios (software libre). A grandes rasgos, esto significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software.

c. Qué es gcc

Gcc es una herramienta que contiene todo lo necesario para crear los programas en todo tipo de lenguajes de programación.

d. Qué gestiona el kernel

Linux es el nombre del kernel/núcleo. El núcleo es la parte central de un sistema operativo que mantiene un seguimiento de todo. El núcleo es tanto una cerca como una puerta. Como una puerta, permite que los programas accedan al hardware de manera uniforme. Sin el núcleo, tendría que escribir programas para tratar con cada modelo de dispositivo jamás hecho. El núcleo maneja todas las interacciones específicas del dispositivo para que no tenga que hacerlo. También maneja el acceso a archivos y la interacción entre procesos. El núcleo también controla el flujo de información entre programas.

Teclado →Kernel→ (Sistema de ventanas)→ Programa

e. ¿Se puede acceder simultáneamente a instrucciones y datos?¿Por qué?

La CPU lee las instrucciones de memoria una a la vez y las ejecuta. Esto se conoce como el ciclo fetch-execute. Para ello se utiliza el contador de programa, en el que lee a esa dirección de memoria que contiene la instrucción necesitando los datos.

f. Cuál es la función del registro PC

El contador del programa se utiliza para indicar al ordenador de dónde obtener la siguiente instrucción. Mencionamos anteriormente que no hay diferencia entre la forma en que se almacenan los datos y los programas, solo que la CPU los interpreta de manera diferente. El contador del programa contiene la dirección de memoria de la siguiente instrucción a ejecutar. La CPU comienza mirando el contador del programa, y obteniendo cualquier número almacenado en la memoria en la ubicación especificada. Luego se pasa al decodificador de instrucciones que averigua qué significa la instrucción.

g. Cuáles son los dos tipos de registros de la CPU

Además de la memoria en el exterior del procesador, el propio procesador tiene algunas ubicaciones especiales de memoria de alta velocidad llamadas registros. Hay dos tipos de registros - registros generales y registros especiales. Los registros de propósito general son donde ocurre la acción principal. La suma, resta, multiplicación, comparación y otras operaciones generalmente usan registros de propósito general para el procesamiento. Sin embargo, las computadoras tienen muy pocos registros de propósito general. La mayoría de la información se almacena en la memoria principal, se lleva a los registros para su procesamiento, y luego se vuelve a poner en la memoria cuando se completa el procesamiento. los registros especiales son registros que tienen fines muy específicos.

h. Qué significa Word Size

En los ordenadores que se usan actualmente, los registros son de 4 bytes de largo. El tamaño de un registro típico se llama “computer’s word size”. En este caso tendríamos palabras de 4 bytes.

i. Qué es una varible puntero.

Las direcciones también tienen cuatro bytes (1 palabra) de largo, y por lo tanto también caben en un registro. Los procesadores x86 pueden acceder hasta 4294967296 bytes si se instala suficiente memoria. Un número llega a ser una dirección cuando tratas de mirar al lugar al que apunta el byte que estás mirando. Las direcciones que se almacenan en la memoria también se llaman punteros, porque en lugar de tener un valor regular en ellas, apuntan a una ubicación diferente en la memoria.

j. Lista cuatro modos diferentes de direccionar un operando.

1. Direccionamiento inmediato:

El modo más simple es el modo inmediato, en el que los datos a acceder están incrustados en la instrucción misma. Por ejemplo, si queremos inicializar un registro a 0, en lugar de darle al equipo una dirección para leer el 0, especificaríamos el modo inmediato y le daríamos el número 0.

2. Direccionamiento directo:

En el modo de direccionamiento directo, la instrucción contiene la dirección de memoria a acceder. Por ejemplo, podría decir, por favor, cargue este registro con los datos de la dirección 2002. La computadora iría directamente al byte número 2002 y copiaría el contenido en nuestro registro.

3. Direccionamiento indexado:

En el modo de direccionamiento indexado, la instrucción contiene una dirección de memoria a la que acceder, y también especifica un registro de índice para compensar esa dirección. Por ejemplo, podríamos especificar la dirección 2002 y un registro de índice. Si el registro de índice contiene el número 4, la dirección real desde la que se cargan los datos sería 2006. De esta manera, si usted tiene un conjunto de números a partir de la ubicación 2002, puede ciclo entre cada uno de ellos utilizando un registro de índice.

4. Direccionamiento indirecto:

En el modo de direccionamiento indirecto, la instrucción contiene un registro que contiene un puntero a donde se debe acceder a los datos. Por ejemplo, si usamos el modo de direccionamiento indirecto y especificamos el registro %eax, y el registro %eax contenía el valor 4, se usaría cualquier valor que estuviera en la ubicación de memoria 4.

5. Direccionamiento de registro:

En el modo de direccionamiento de registro, la instrucción contiene un registro para acceder, en lugar de una ubicación de memoria. El resto de los modos se ocupará de las direcciones.

20.1.2 Cuestiones prácticas:

1. Comando de compilación del programa fuente ensamblador mediante el front-end gcc qué incluya la tabla de símbolos para el depurador

gcc -m32 –save-temps -g -o sum1toN sum1toN.s

2. Comando de enlace (linker) del módulo objeto reubicable.

ld -melf_i386

3. Declaración en lenguaje C de la variable n tipo entero con signo de un byte.

- n: .byte

4. Instrucción en lenguaje ensamblador del programa sum1toN.s que realiza una suma.

Add destino, fuente

5. Comandos del depurador gdb para la impresión del contenido de la variable n

p (int) n, p $n

6. Ejecutar sum1toN, compilado de sum1toN.s, paso a paso mediante el depurador GDB ejecutando los comandos necesarios para:
a. imprimir el contenido de la variable n y su dirección en memoria principal

p (int) n, x &n
b. imprimir la dirección de la etiqueta bucle

x &bucle
c. imprimir el contenido del registro ECX al salir del bucle

p $ecx

7. Cambiar el tamaño de los operandos de la suma a 2 bytes

a. Cambiar el tamaño del operando n → n: .word 5

8. Cambiar la instrucción add %edx,%ecx por la instrucción addw %dx,%cx

Al cambiar la instrucción, te notifica de un error en el registro %ec.

9. En GDB qué comando hay que utilizar para ejecutar todas las iteracciones del bucle del programa de forma continuada.

Until

10. Comparando las versiones en lenguajes C y ASM de los módulos fuente, por qué la instrucción until del depurador GDB en el caso del módulo fuente en lenguaje C se ejecuta durante la sentencia while.

Porque si ejecutas until durante la sentencia while, éste funcionaría únicamente como una instrucción next avanzando a la siguiente línea del programa. Sin embargo, de la otra forma se ejecutarían todas las iteraciones del bucle a la vez.

Eduardo Ezponda Igea