Clase 18 (sincrónica, viernes 4 de junio): Ejercicio sobre recursividad, función de Ackermann

Ya vimos que el uso de la pila es intensivo durante la ejecución de un programa. Se usa para almacenar registros temporalmente, para memoria de trabajo (ejemplo, variables locales en funciones de C/C++), para almacenar las direcciones de retorno, entre otros.

Además, hablaremos más adelante que la pila puede ser sensible a problemas de seguridad si no se toman las medidas adecuadas (una de las tareas programadas está relacionada a esa temática). Si aún quedan dudas, se recomienda revisar posteriormente lo siguiente:

- Del libro x86-64 Assembly Language Programming with Ubuntu:

  Capítulo 9

- Del libro Computer Systems: a Programmer's Perspective:

  Sección 3.7.4

  Sección 3.10.5

Para terminar de reforzar este tema, en la clase de hoy, se espera que en equipos de hasta 3 personas hagan lo siguiente:

1. Hacer un programa en C/C++ que invoque a un método externo llamado ackermann.

2. En lenguaje ensamblador implementar la función ackermann, la cuál

deberá calcular un número de Ackermann, dada una entrada particular. Esta función devuelve un entero y tiene como parámetros dos enteros. Debe seguir las convenciones de C/C++ para el uso de la pila al invocar el método.

3. Hacer un dibujo de la pila donde ilustre los **tres** primeros llamados recursivos. Este dibujo debe incluir lo que se almacena en la pila después de cada vez que se hace uso de ella (push, pop, llamados, modificación de rsp/rbp, etc.). Puede hacer el dibujo en papel e incluir fotos, usar una hoja de Excel, etc.

Sobre la función de Ackermann:

Esta función recursiva tiene la particularidad de que crece extremadamente rápido. Es usada en contextos de complejidad computacional donde se quieren medir elementos de la arquitectura donde se está ejecutando la función.

La fórmula recursiva la pueden encontrar aquí: https://en.wikipedia.org/wiki/Ackermann_function

Para resolver (desde cero) esta función de forma recursiva requiere de un muy buen entendimiento del funcionamiento de la pila, principalmente si necesita variables locales (que se almacenarán en la pila) y sobre cómo se devuelven los valores de un llamado a otro.

Escriba su función usando solo el espacio necesario para memoria de trabajo (es decir, no puede hacer uso de memoria definida en el .data).

Usen solo números pequeños, sino, dependiendo de la memoria de la máquina, el sistema operativo, etc., podría tardar mucho tiempo

ejecutándose, o bien podría quedarse sin pila en algún momento (se recomienda hacer cálculos de Ackermann (1, 4), Ackermann (2, 4), puede explorar valores más allá de esos, pero considere que se podría caer su programa).

Si necesita depurar, hágalo con valores pequeños, ejemplo Ackermann (1, 2), cuyo resultado es 4 y requiere solo 11 llamados recursivos.

Vea esto como un ejercicio meramente académico para entender el funcionamiento de la pila en lenguaje ensamblador. También considérelo como un reto, por ejemplo, **¿podrá mi código en ensamblador ser más eficiente y calcular un número de Ackermann más grande que el que es capaz de calcular C/C++ con el código generado por el compilador por default?, ¿podrá mi código requerir menos espacios para almacenarse que el código que genera un compilador de C/C++?**

Last modified: viernes, 4 junio 2021, 11:56 a.m.