Instituto Tecnológico de Costa Rica

Ingeniería en Computación

Sede Central – Cartago

IC-3101 Arquitectura de Computadores

**Proyecto #1 – Calculadora NASM**

Profesor:

Esteban Arias Méndez

Estudiantes:

Óscar Cortés Cordero – 2016136191

Randall Delgado Miranda – 2016238510

Fecha de entrega: 8 de mayo

I Semestre 2017

**Abstract:** The following project consists in a calculator, developed in Assembly Language, specifically in NASM for the Linux operative system. It consists of a program in which a series of basic mathematical operations can be performed, like adding, substracting, multiplying and dividing in any given base. It also has the capability of calculating a number’s binary complement, converting floating-point numbers to another base, storing variables, and a list of useful commands. With the use of simple assembly instructions, all of the features mentioned above are added to create the program.

Introducción

La herramienta de NASM resulta muy útil para el desarrollo de programas portables en ensamblador, especialmente por su manejo actual de 32 bits, lo cual permite un mayor manejo de valores. En este proyecto, el objetivo consiste en desarrollar una calculadora, la cual permita operar 2 o más valores en las bases numéricas principales: decimal, binario, octal y hexadecimal. Además, debe ser capaz de convertir números en punto flotante, calcular el complemento de un número dado, y guardar variables para su futuro uso.

La única herramienta usada para el desarrollo del programa fue el mismo NASM, y el editor de texto incorporado en Linux, Geany. La documentación utilizada para NASM fue tomada de la página oficial, del siguiente enlace: <http://www.nasm.us/xdoc/2.13.01/nasmdoc.pdf>. También se llegó a consultar del libro de Sivarama P. Dandamudi, “Guide to Assembly Language Programming in Linux”, tomado del enlace de Google Drive brindado por el profesor en el foro.

El objetivo principal no es sólo completar el proyecto en sí, sino más bien lograr un dominio más amplio del lenguaje ensamblador, para futuros proyectos que se presenten e involucren el uso de éste. De paso, se aprende también acerca del manejo y organización de la memoria, direccionamientos, y modalidades un poco más complejas como lo son el manejo del punto flotante mediante instrucciones FPU, y reconocimiento de sintaxis.

Desarrollo

Para el desarrollo de todo el proyecto, no se utilizó código externo como guía para su elaboración, sino que se recurrió a la experimentación, y la consulta de la documentación de la página de NASM, y en parte al libro de Sivarama, ya que descubrir por cuenta propia cómo funciona cierta rutina resulta ser más retroalimentativo que basarse en un código ya realizado.

Para la elaboración de las sumas, restas, multiplicaciones y divisiones, se realiza en un primer inicio el reconocimiento de a cuál base pertenece cada operando, para identificar su conversión respectiva. Una vez se identificó esto último de la línea de comandos, se procede a revisar en cuál base desea el usuario que se dé el resultado, para brindarlo al final de la operación. Para operaciones combinadas y paréntesis, se utiliza la conversión de operaciones a operación prefija, y se utiliza un macro que asigna un valor de prioridad a cada operador.

Para la muestra del complemento, cuando se lee el símbolo “~”, ya se sabe que se desea buscar el complemento a 2 de ese número. Por lo tanto, una vez se identifica su base, se convierte a binario (a menos que ya lo esté, o que esté en octal/hexadecimal, que requiere pasarse a decimal primero), y luego se saca su complemento mediante la instrucción “not”, y la suma de 1.

La evaluación de comandos se hace desde el inicio, al comparar si el primer carácter es un “#”, y si lo es, se hace una comparación de cada comando. Para los comandos #ayuda y #salir, simplemente se imprime la variable asignada. Para el comando #procedimiento, se tiene un flag que indica si se muestra el procedimiento o no.

Análisis de resultados

Como parte principal de una calculadora, que es resolver sumas, restas, multiplicaciones y divisiones, todo funciona bien para cualquier base dada, y el resultado se devuelve en la base que indique el usuario. Se respetan los órdenes de precedencia y los paréntesis, si están presentes. Sin embargo, presenta un error cuando en una multiplicación o división, el segundo operando tiene un “-” adelante, implicando ser negativo. Esto no se da con el primer operando.

El complemento de un número dado se devuelve de manera exitosa, en cualquier base que se brinde. Se muestra correctamente su valor complementado a 2 en binario. Las variables, lastimosamente, no se pudieron llegar a implementar, debido a su nivel de dificultad y al tiempo disponible, ya que se les dio más enfoque a las partes principales del proyecto. Del mismo modo se dio con las conversiones de punto flotante, ya que, aunque se investigó sobre ello, la carga académica complicó el poder implementarlo.

Los errores que puede presentar el programa relacionados con la parte implementada (por ejemplo, errores de sintaxis, división entre cero, números con base incorrecta o desbordamiento) fueron validados correctamente. En caso de ocurrir uno de ellos, se despliega un mensaje al usuario.

Los comandos se realizan correctamente, mientras se puedan implementar o tengan algún uso. Por ejemplo, el comando #bits no tiene ninguna utilidad sin haber implementado la parte de punto flotante, o el comando #var, ya que no se pueden definir variables, pero el comando #procedimiento sí, ya que se puede elegir entre mostrar el resultado de la operación en binario (aunque no se muestra un paso a paso de la operación). También se implementaron los comandos #ayuda y #salir.

No se logró implementar ninguna característica extra al proyecto, a pesar de que no eran de gran dificultad. Sin embargo, hay que darles prioridad a las características principales, y luego preocuparse por los puntos adicionales que se puedan obtener.

Conclusiones

Como conclusión acerca del proyecto, cabe decir que resultó ser un proyecto muy interesante, ya que se acerca mucho a lo que es la creación de un lenguaje de programación, en el sentido de que sigue una sintaxis propia, y permite el manejo de variables con las cuales recurre al uso de memoria. Tal vez no sea un proyecto tan dinámico en sí, pero su desarrollo tampoco resultó ser tedioso, incluso se podría decir que resultó llamativo, a su manera.

Sin embargo, no se logró cumplir el objetivo propuesto, el cual consistía en completar el proyecto, aunque se obtuvo gran conocimiento acerca del lenguaje ensamblador, de NASM y su forma de trabajar, y brindó una mejor capacidad lógica para resolver problemas, que de otra forma serían más difíciles de no ser por proyectos de este tipo, que ayudan a practicar para mejorar la capacidad de desarrollar programas de manera eficaz y eficiente.

Dos factores que en conjunto contribuyeron a la incompletitud del proyecto fueron el tiempo y la carga académica de otros cursos. Se debe admitir que faltó organización a la hora de desarrollar este proyecto, y se tuvieron que haber organizado los roles con mucha más anticipación, ya que sería mentira afirmar que no hubo suficiente tiempo. No obstante, el resto de carga académica dificultó también una correcta organización del trabajo, pero no justifica la falta de organización temprana.

Como última observación, cabe destacar que, al ser el primer proyecto del curso, resultó bastante manejable de realizar, pese a la ausencia de ciertas características. Aunque tenía su grado de dificultad, no habían partes que fueran imposibles de realizar; todo estaba al alcance del estudiante para poder realizar el proyecto de la mejor manera, y eso es lo que define un buen proyecto.

Apéndice