Programación de Computadores

Valeria Morales Alvarado, Mathiw Rojas Jiménez, Estefanny Villalta Segura, Felipe Quesada Sánchez

Ingeniería en Computadores, TEC

Cartago, Costa Rica

valeriamorales40@gmail.com

matrj08@gmail.com

stephanie1117@hotmail.es

pipequesan@gmail.com

resumen-*la programación de computadores tiene todo una historia, en sus primeras fases se encuentra el lenguaje máquina el cual permitía la creación de instrucciones por medio de 0 y 1, asimismo que una computadora tuviera solo un programa el cual manejara la parte del hardware sin poder editarlo o mejorarlo, con el paso del tiempo la programación de computadores fue evolucionando cambiando así este tipo de obstáculos, siendo más accesible el cambio de instrucciones en los circuitos programables o procesadores, con mejoras en la ejecución de programas y creando nuevos tipos de lenguajes para dar instrucciones a la máquina, como el lenguaje ensamblador, el cual facilita dicha tarea.*

I. Introducción

El lenguaje de máquina es un sistema que se relaciona con la forma en la cual cada computador interpreta una serie de instrucciones. Dentro de este tema encontramos características del mismo relacionadas con subtemas como programa almacenado, ejecución de programa, y ensambladores. Todas ellas muy importantes en el análisis de esta área de la programación.

II. Lenguaje Máquina

El lenguaje máquina es el lenguaje que utilizan los computadores para procesar las instrucciones que le son ordenadas. Este lenguaje se basa en el binario, o sea, en unos (1) y ceros (0). Todo lenguaje de programación como punto final de traducción, debe de llegar a lenguaje máquina, para poder llegar a ser interpretado y ejecutado, como en Fig. 1.

Fig. 1 Ejemplo de traducción a máquina.

Este lenguaje depende de la arquitectura de cada computador, o sea, no existe un único lenguaje máquina. Cada familia de computadores posee su propia arquitectura básica, como por ejemplo: la familia Intel x86 posee la misma arquitectura básica y la familia IBM System/370 posee la misma arquitectura, pero no es lo mismo la arquitectura de Intel que la de IBM. Esto crea incompatibilidad de código entre diferentes computadores de diferentes familias, pero dentro de la misma familia, hacia atrás, si existe compatibilidad de código, esto quiere decir que, los modelos más nuevos, entienden el lenguaje máquina de los más viejos, pero no al revés.

El lenguaje máquina, se basa en las siguientes instrucciones sencillas:

- Registros específicos para operaciones aritméticas, direccionamiento o control de funciones
- Posiciones de memoria específicas(offset)
- Modos de direccionamiento usados para interpretar operando

Y con un conjunto de estas instrucciones, es que el lenguaje máquina puede lograr ejecutar cualquier otra instrucción más compleja.

III. Programa Almacenado

El concepto de programa almacenado se establece como el almacenamiento y ejecución las instrucciones de un programa en la memoria central (llámese también la memoria RAM) de un computador, haciéndola capaz de mostrar tanto los datos como resultados de dicho programa.

El concepto de un programa almacenado viene planteándose desde que Turing presentó su trabajo sobre la máquina universal de Turing, basándose en lo teóricamente establecido

en este documento. El nombre de John Von Newmann y el modelo de Von Newmann suelen ser reconocidos como los principales pioneros del programa almacenado debido a ser los primeros en plantear y mostrar avances en el área. Desde entonces los diversos profesionales que han intentado hacer un acercamiento a la aplicación de este concepto ha sido ampliado, hasta que en el año 1948 donde la SSEM "Baby" de la Universidad de Mánchester hiciera el primer debut de una computadora de programa almacenado.

Lo que hace a un programa almacenado diferente es que en las primeras versiones de computadoras, estas cumplían con un único propósito por lo cual las instrucciones nunca estaban almacenadas en los equipos. Al ya poderse almacenar instrucciones en la memoria central, en caso de ser necesario, siempre se podía modificar lo almacenado a conveniencia de lo que se necesitase.

El éxito de los programas almacenados radica en la separación del programa de la máquina; impulsando mayor libertad en el diseño de computadoras que fueran flexibles y a las cuales con este nuevo método, se le podían ingresar diferentes programas a su memoria para ser utilizados.

IV. Ejecución de Programa

Un programa, como ya sabemos, es una serie de instrucciones que le damos a la computadora mediante un algoritmo específico. Para llegar a la ejecución del programa hay que completar una serie de pasos, principalmente dar al programa la función que queremos que realice. Cada vez que llamamos al programa que hemos creado previamente es a lo que le llamamos ejecución del programa, estamos ejecutando las instrucciones que le damos a la computadora.

Durante este proceso pueden existir inconvenientes sin embargo la mayoría de las veces se pueden arreglar. A la hora de encontrar y solucionar errores en el programa debemos tener en cuenta que hay tres tipos de error: el error de sintaxis, el de runtime y el de semántica. En ese caso en la programación en Python, el cual es de alto nivel, a comparación del lenguaje de máquina que es de bajo nivel, es común que al correr el programa, Python nos indique los errores de sintaxis, los cuales son deficiencias en la escritura. La palabra runtime hace referencia al tiempo en el que se ejecuta un programa, desde la memoria principal hasta que llegue a una conclusión; ya sea que obtenemos un resultado, o que ocurre el error de runtime, el cual indica que el programa finaliza la ejecución sin haber llegado a ningún resultado. Finalmente encontramos el error de semántica, el cual se refiere a que el programa funciona sin embargo no devuelve el resultado esperado, lo cual significa que no enviamos las instrucciones de forma correcta, por lo cual le pedimos al programa que realizará otra función. Estos tres errores son factores importantes que pueden interferir con la ejecución de programa, en especial sintaxis y runtime ya que no nos dejan obtener ningún resultado. Al programar debemos asegurarnos de que no haya ningún error y una vez corregidos, la ejecución del programa nos lleva a la salida. La salida es la expresión utilizada para describir el resultado de ingresar un parámetro y realizar una función.

El origen del lenguaje de máquina viene desde la creación del hardware de la computadora, ya que cada procesador tiene su propio lenguaje de máquina. Este está ligado al lenguaje de ensamblador el cual es un lenguaje de bajo nivel lo que significa que tiene una interacción más directa con la máquina. El ensamblador es un programa utilitario, término que se refiere a software que da la solución al problema propuesto. Este traduce el código de ensamblador a lenguaje de máquina lo que permite realizar todos los procesos e instrucciones. En este sentido al transformar el las instrucciones de forma que la computadora entienda los datos de máquina se da la ejecución del programa. En este caso, como es de bajo nivel presenta características que se relacionan con la ejecución diferente a los lenguajes de alto nivel. Entre estos esta que es más difícil de comprender y más complicado de portar ya que va desde el hardware del sistema, y que en este también se puede controlar el tiempo de ejecución.

En este caso podemos ver la comparación de la ejecución de programa en un lenguaje de alto nivel como en uno de bajo nivel y las características de cada uno con respecto a la ejecución y flujo del programa. En el lenguaje de ensamblador también podemos encontrarnos con otro tipo de problemas a la hora de buscar la salida. En ocasiones es probable que el programa se detenga marcando ciertas instrucciones, a esto se le llama punto de ruptura o breakpoint. Esta sucede cuando hay algún error en el programa y lo que marca es la zona donde hay un conflicto. Para solventarlo debemos detener el programa y analizar el contenido de la memoria o registro para encontrar el problema. Una vez el programa está listo, debemos mandar la orden de correr o ejecutar lo que hemos escrito. Así, una vez más, podemos asegurarnos de que todo esté funcionando. En caso de que el proceso no sea tan efectivo como esperábamos debemos empezar a analizar los posibles puntos de error que encontramos en el programa. Para este proceso se pueden usar listas de ensamblado y ejecutar las instrucciones paso a paso. A estos métodos de solución de errores se les llama también técnicas de depuración. Estas fallas también pueden afectar el flujo del programa, sin embargo, una vez estén resueltos procedemos a ejecutar el programa y obtener nuestra salida.

V. Ensambladores

El lenguaje máquina es un tipo de lenguaje que está encargado de darle instrucciones una serie de circuitos, sin embargo, este es un tanto difícil y tedioso de programar, así que se creó un nuevo lenguaje el cual se pudiera hacer operaciones por medio de una colección de símbolos mnemónicos para facilitar la comunicación entre el ser humano y la máquina, asimismo la programación de instrucciones a la parte del hardware de la computadora o circuito programable.

Los lenguajes ensambladores permiten una comunicación más simple entre el humano y la máquina gracias a que traduce lo escrito por el humano a lenguaje máquina.

A. Tipos de Ensambladores

Se les clasifica dependiendo de sus características. Los lenguajes ensambladores tienen una entrada (programa fuente) y una salida (programa objeto) y cada uno tiene distintas formas de resolver el proceso de tener una la entrada un programa y pasarlo a otro en la salida. Algunos lenguajes, procesadores o ensambladores son:

1) Macroensambladores o moduladores: Este tipo de ensamblador son un tanto difíciles de programar y entender, a pesar de que tienen la posibilidad de no establecerse en el programa se consideran residentes por su complejidad. Estos funcionan con macroinstrucciones (las macroinstrucciones son varias instrucciones que en conjunto forma un algoritmo para realizar una tarea más compleja).

2) Ensambladores Cruzados: Si se tienen dos computadoras con procesadores diferentes y en una se utiliza un ensamblador el cual el programa objeto producto del traducido de uno fuente funciona la para la otra computadora con el procesador distinto, sería el llamado lenguaje ensamblador cruzado.

3) Ensambladores Residentes: Pueden ser de varios tipos, siempre y cuando permanezcan en la memoria de la computadora se van a considerar ensambladores residentes. Una de sus ventajas es que como esta en la memoria se puede probar, crear y depurar el código en el momento y no como otros los cuales se tendrían que transportar.

4) Ensambladores de una Fase: Los ensambladores de una fase van paso por paso revisando el código y traduciendo a lenguaje máquina, por esta razón primero se definen los símbolos, es decir, que significa cada línea de código, para luego traducir de una forma correcta y eficiente.

5) Ensambladores de dos Fases: Estos tipos de ensamblador sin similares a los de una fase, en vez de traducir el programa en una fase lo hacen en dos etapas. La primera es para reconocer qué significa cada símbolo y la segunda es para traducir el código, ya que no le es necesario saber que significa cada símbolo porque está en su memoria. Un tipo de ensamblador que se basa en el de dos fases es el ensamblador 8086 que muy utilizado en la actualidad por ejemplo por Windows.

Los lenguajes ensambladores, a pesar de ser bastante arcaicos, son indispensables en la actualidad para el funcionamiento y comunicación correcta y eficiente de cualquier computador.

Algunas de las características por mencionar son:

- Incrementan la eficiencia del programa: Si no existe un lenguaje ensamblador los programadores deberán escribir el código de un lenguaje de alto nivel, por ejemplo, en código máquina.
- Incrementa la productividad: Al escribir en otros lenguajes de programación que no sea el lenguaje máquina se optimiza tiempo y se logra más eficiencia al escribir un código ya que es más accesible para los programadores.
- Flexibilidad al escribir un programa: En lenguaje ensamblador es como un traductor, esto significa que yo puedo darle órdenes a una computadora por medio de lenguajes de alto nivel como por ejemplo c++ y así manejar la parte del hardware, un ejemplo son las computadoras programables raspberry py.
- Son poco portables, esto porque dicho lenguaje está hecho específicamente para un tipo de hardware, asimismo a pesar de que su fin fue facilitar la forma de dar instrucciones a un computador, son difíciles de entender directamente, por consiguiente más difíciles de programar, depurar, mantener y actualizar, sin embargo, es algo que se puede superar con práctica y estudio.

VI. Conclusiones

Se demuestra que el lenguaje de máquina, es el lenguaje de programación de más bajo nivel. El cual posee características como programa almacenado; la relación con la memoria electrónica del sistema y la conservación de datos, ejecución de programa; el proceso, flujo del programa y respectivos errores, y ensambladores; programa utilizado para traducir un código directamente al microprocesador.

Referencias

[1] A. Suárez. (s.f). *Lenguaje máquina y ensamblador*. [Online] Recuperado de:http://arquitecturadelcomputadorsemestre6.blogspot.com/2014/10/lenguaje-maquina-y-lenguaje-ensamblador.html

[2] *Lenguaje máquina: La interfaz entre el hardware y el software.* (s.f). [Online] Recuperado de: https://www.esi.uclm.es/www/isanchez/eco0910/maquina.pdf

[3] L. Salazar. (s.f). *ITCM Modelos de computadoras KND*. [Online] Recuperado de: http://itcmmodelosdecomputadorasknd.blogspot.com/2009/12/concepto-de-programa-almacenado_04.html

[4] Universidad Complutense de Madrid. (2001). *Programación en ensamblador*. [Online]. Recuperado de: http://www.dacya.ucm.es/hidalgo/estructura/ensamblador.pdf

[5] A. Tanenbaum. (2000). *Organización de compuatdoras: Un enfoque estructurado*. Cuarta edición. Amsterdam, Holanda. Pearson Educación. Recuperado de: http://f.javier.io/rep/books/Tanenbaum,%20Andrew%20S.-%20Organizaci%C3%B3n%20de%20Computadoras.%20Un%20Enfoque%20Estructurado.%20Cuarta%20Edici%C3%B3n.%20M%C3%A9xico,%20Prentice%20Hall,%202000.pdf

[6] A. Aguirrez. (s.f.). *Tipos de lenguaje ensamblador*. [Online]. Recuperado de: https://sites.google.com/site/ensambladoralexa/1-1-3-tipos-de-lenguaje-ensamblador

[7] A. Gómez. (s.f). *Tipos de ensambladores*. [Online]. Recuperado de: https://informatica4194.webnode.mx/contactanos/tipos-de-ensambladores/

[8] J. Gómez. (1998). *Estructura de computadores: Ensamblador MIPS*. [Online]. Recuperado de: http://www2.elo.utfsm.cl/~lsb/elo311/aplicaciones/spim/Tutorial_MIPS.pdf

[9] La Revista Informática. (s.f). *Lenguaje de programación ensamblador*. [Online]. Recuperado de: http://www.larevistainformatica.com/Ensamblador.htm

[10] Universidad de Virginia. (Noviembre, 2018). *x86 Assembly Guide*. [Online]. Recuperado de: http://www.cs.virginia.edu/~evans/cs216/guides/x86.html
[12] Internet Wayback. (Septiembre, 1996). *The Art of Assembly Language Programming*. [Online]. Recuperado de: https://web.archive.org/web/20041231081205/http://cs.smith.edu/~thiebaut/ArtOfAssembly/artofasm.html