TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Unidad 4

ALUMNOS- Christiam Iván Rodríguez moreno
NO.CONTROL: 13480619
CARRERA: Ingeniería en Sistemas Computacionales
ASESOR INTERNO: Juan Pablo Rosas Baldazo

Cd. Guadalupe, N.L. Mayo, 2018

Índice

Introducción

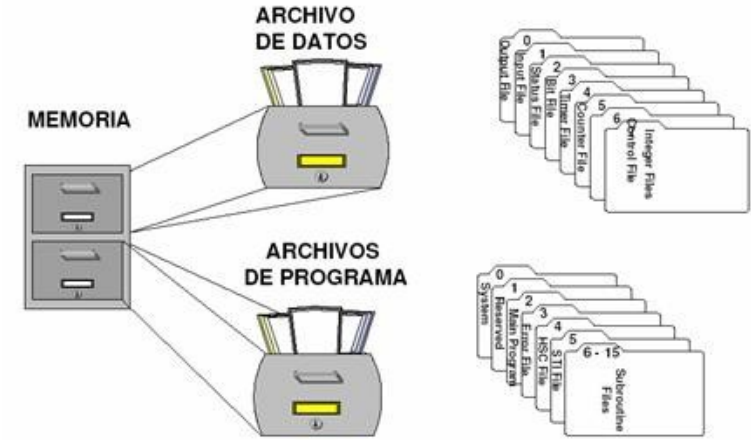
La generación de código es la fase más compleja de un compilador, puesto que no sólo depende de las características del lenguaje fuente sino también de contar con información detallada acerca de la arquitectura objetivo, la estructura del ambiente de ejecución y el sistema operativo que esté corriendo en la máquina objetivo.
La generación de código por lo regular implica también algún intento por optimizar, o mejorar, la velocidad y/o el tamaño del código objetivo recolectando más información acerca del programa fuente y adecuando el código generado para sacar ventaja de las características especiales de la máquina objetivo, tales como registros, modos de direccionamiento, distribución y memoria caché.

¿Que es un registro?

Los registros son la memoria principal de la computadora. existen diversos registros de propósito general y otros de uso exclusivo.
Algunos registros de propósito general son utilizados para cierto tipo de funciones. Existen registros acumuladores, puntero de instrucción, de pila, etc.
Distribución la distribución es el proceso en el que el programa generado puede ejecutarse en otras máquinas.
Con respecto al ensamblador, la mayoría del direccionamiento se hace relativo para que el programa sea re localizable por un programa llamado cargador.
En el caso de programas compilados se necesitan de las librerías, si son estáticas se incluyen en el ejecutable por lo que el programa se hace gráfico, si son dinámicas no pero el programa es más pequeño. Debido a la complejidad del software actual se necesitan de asistentes para poder instalar y ejecutar un programa.

¿Quiénes lo utilizan?

Antes de nada, para el desarrollo de esta parte hablaremos indistintamente de registros de activación o de marcos de pila. Esto se debe a que en la documentación encontrada sobre el manejo de los registros ebp y esp se hace mención a dicho concepto de marco de pila. Puesto que el lenguaje permite recursividad, los registros de activación se asignan dinámica mente.



Distribución

La UCP o CPU tiene 14 registros internos, cada uno de ellos de 16 bits (una palabra). Los

bits están enumerados de derecha a izquierda, de tal modo que el bit menos significativo es el bit 0. Los registros se pueden clasificar de la siguiente forma:

Instrucción Significado (Definidas por el

fabricante del CPU)

0000 0000 – No hacer nada

0000 0001 – Sumar V1 y V2 y guardar el resultado

0000 0011 – Muestra en pantalla la variable R

0000 0111 – Guarda en memoria lo que se te da en

una variable llamada V1

0000 1111 – Guarda en memoria lo que se te da en

una variable llamada V2

0001 0000 – Mover de la memoria el valor de Suma

a la variable R

Registros de datos:

AX: Registro acumulador. Es el principal empleado en las operaciones aritméticas.

BX: Registro base. Se usa para indicar un desplazamiento.

CX: Registro contador. Se usa como contador en los bucles.

DX: Registro de datos.

Estos registros son de uso general y también pueden ser utilizados como registros de 8 bits, para utilizarlos como tales es necesario referirse a ellos como por ejemplo: AH y AL, que son los bytes alto y bajo del registro AX. Esta nomenclatura es aplicable también a los registros BX, CX y DX.

4.2 Lenguaje ensamblador

¿Qué es Lenguaje ensamblador?

El lenguaje ensamblador es un tipo de lenguaje de bajo nivel utilizado para escribir programas informáticos, y constituye la representación más directa del código máquina específico para cada arquitectura de computadora

Lenguaje

Un programa escrito en lenguaje ensamblador consiste en una serie de Instrucciones que corresponden al flujo de órdenes ejecutables que pueden ser cargadas en la Memoria de un sistema basado en Microprocesador. Por ejemplo, un Procesador x86 puede ejecutar la siguiente instrucción Binaria como se expresa en código de máquina:

•Binario: 10110000 01100001 (Hexadecimal: 0xb061)

La representación equivalente en lenguaje ensamblador es más fácil de recordar:

•MOV al, 061h

El código máquina, o lenguaje de máquina, está formado por instrucciones sencillas, que dependiendo de la estructura del procesador- pueden especificar:

•Registros específicos para operaciones aritméticas, direccionamiento o control de

funciones.

•Posiciones de memoria específicas (offset).

•Modos de direccionamiento usados para interpretar operandos.Las operaciones más complejas se realizan combinando estas instrucciones sencillas,que pueden ser ejecutadas secuencialmente o mediante instrucciones de control de flujo.Casi todas las instrucciones utilizan 2 operandos específicos para realizar su función.

Ejemplo, cuando deseamos mover un valor constante hacia un registro de almacenamiento debemos especificar ambos operandos. Las operaciones disponibles en la mayoría de los conjuntos de instrucciones incluyen:

•mover

llenar un registro con un valor constante (Ej.: mov al, `20`). o mover datos de una posición de memoria a un registro o viceversa (Ej.: mov al, [si]) o escribir y leer datos de dispositivos (Ej.: lea dx, offset cadena)

•computar

sumar, restar, multiplicar o dividir los valores de dos registros, colocando el resultado en uno de ellos o en otro registro (Ej.: sum, mul, div, entres otras

instrucciones). o realizar operaciones binarias, incluyendo operaciones lógicas (AND/OR/XOR/NOT) o comparar valores entre registros (mayor, menor, igual) (Ej.: cmp)

•afectar el flujo del programa saltar a otra posición en el programa y ejecutar instrucciones allí(Ej.: jmp) o saltar si se cumplen ciertas condiciones (IF) (Ej.: jnb, jnz, jb, jz, jne, je, entre otros) o saltar a otra posición, pero guardar el punto de salida para retornar (Ej.: CALL, llamada a subrutinas) Algunas computadoras incluyen instrucciones complejas dentro de sus capacidades. Una sola instrucción compleja hace lo mismo que en otras computadoras puede requerir una larga serie de instrucciones, por ejemplo:

•salvar varios registros en la Pila de una sola vez

•mover grandes bloques de memoria

Características:

El programa lee un archivo escrito en lenguaje ensamblador y sustituye cada uno de los códigos memotécnicos por su equivalente código máquina. Los programas se hacen fácilmente portables de máquina a máquina y el cálculo de bifurcaciones se hace de manera fácil.

Clasificación:

•**Ensambladores básicos:**Son de muy bajo nivel, y su tarea consiste básicamente, en

ofrecer nombres simbólicos a las distintas instrucciones, parámetros y cosas tales como

los modos de direccionamiento

•**Ensambladores modulares, o macro ensambladores:**Descendientes de los

ensambladores básicos, fueron muy populares en las décadas de los 50 y los 60, fueron

antes de la generalización de los lenguajes de alto nivel. Un macroinstrucción es el

equivalente a una función en un lenguaje de alto nivel.

¿Que Lenguaje Máquina?

•El Lenguaje Maquina es el conjunto de datos que la parte física de la computadora (Hardware) es capaz de comprender e interpretar “El Código Binario” comprendido por los Valores 0 y 1 con tensiones comprendidas entre 0 y 4 Voltios y 4 y 5 Voltios respectivamente, la secuencias de estos valores formaran cadenas de información para que se realice una instrucción.Instrucciones del procesador
De forma arbitraria podríamos asignar a cada Byte una acción que controle alguna función del procesador, por ejemplo podríamos definir las siguientes reglas:

Instrucción Significado (Definidas por el

fabricante del CPU)

0000 0000 – No hacer nada

0000 0001 – Sumar V1 y V2 y guardar el resultado

0000 0011 – Muestra en pantalla la variable R

0000 0111 – Guarda en memoria lo que se te da en

una variable llamada V1

0000 1111 – Guarda en memoria lo que se te da en

una variable llamada V2

0001 0000 – Mover de la memoria el valor de Suma

a la variable R

A este grupo de instrucciones que definen que puede hacer el procesador y la forma en que se utilizan se les denomina: **Conjunto de Instrucciones del**

Procesador, y son específicas del procesador o de la familia del procesador.
Del mismo modo que podemos definir las instrucciones del procesador, podemos hacerlo con los datos que utilizaremos.

Dato Significado (Ya sea por un estándar o

por nosotros)

0000 0001 – 1 (el número 1)

0000 0011 – 2 (el número 2)

0000 0100 – 3 (el número 3)

Con estas reglas primitivas podríamos hacer lo siguiente (siempre recuerda que Nosotros

definimos qué significa cada combinación, así como las reglas que queremos usar de

forma arbitraria):

Instrucción Dato

00000111 00000001

00001111 00000011

00000001

0001 0000

00000011

00000000

Podríamos “traducir” lo anterior como:

1.Guarda en en la variable V1 el número 1

2.Guarda en en la variable V2 el número 2

3.Realizamos las suma del valor de V1 más el valor de V2 (1 + 2, en nuestro ejemplo)

4.Asignamos a la variable R el valor de suma (R = 3)

5.Mostrar en pantalla la variable R

6.No hagas nada

Poniendo todo junto podríamos ponerlo de la siguiente forma (para entenderlo mejor

nosotros):

Instrucción Dato Pseudo código

00000111 00000001 Guarda en en la variable V1 el número 1

00001111 00000011 Guarda en en la variable V2 el número 2

00000001 Realizamos las suma del valor de V1 más el valor de V2 (1 + 2, en nuestro

ejemplo)

00010000 Asignamos a la variable R el valor de suma (R = 3)

00000011 Mostrar en pantalla la variable R

00000000 No hagas nada

Y el resultado sería mostrar en la pantalla la suma de 2 números.

Aunque el procesador solo leerá:

000001110000000100001111000000110000000100010000000000110000000 0

En el caso de los procesadores comerciales, los fabricantes son quienes proporcionan

a los desarrolladores y programadores una serie de instrucciones que “entiende” el procesador. Éste es el Conjunto de Instrucciones del Procesador. Y todos los programas deben utilizar este conjunto de instrucciones de a cuerdo a las reglas y estructura que el fabricante implementa en el procesador.

Instrucción Dato Pseudo código

00000111 00000001 Guarda en en la variable V1 el número 1

00001111 00000011 Guarda en en la variable V2 el número 2

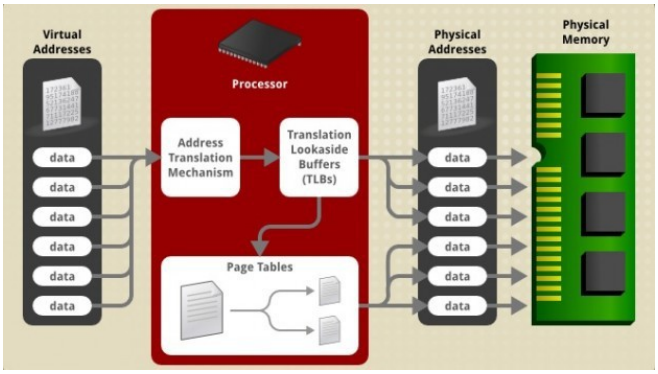
00000001 Realizamos las suma del valor de V1 más el valor de V2 (1 + 2, en nuestro ejemplo)

00010000 Asignamos a la variable R el valor de suma (R = 3)

00000011 Mostrar en pantalla la variable R00000000 No hagas nada

4.4 Administrador de memoria

La administración de la memoria es un proceso hoy en día muy importante, de tal modo que su mal o buen uso tiene una acción directa sobre el desempeño de memoria. En general un ensamblador tiene un administrador de memoria más limitado que u compilador; en la mayoría de los lenguajes de programación el uso de punteros no estaba vigilado por lo que se tienen muchos problemas con el uso de memoria. Los lenguajes más recientes controlan el uso de punteros y tienen un programa denominado recolector de basura que se encarga de limpiar la memoria no utilizada mejorando el desempeño.La memoria principal puede ser considerada como un arreglo lineal de localidades dealmacenamiento de un byte de tamaño. Cada localidad de almacenamiento tiene asignada una dirección que la identifica Se distinguen los siguientes propósitos del sistema de administración de memoria:Protección.Si varios programas comparten la memoria principal, se debería asegurar que el programa no sea capaz de cambiar las ubicaciones no pertenecientica él. Aunque una acción de escritura puede tener efectos más graves que una de lectura, esta última tampoco debería estar permitida, para proporcionar algo de privacidad al programa.



Compartimiento.

Este objetivo parece contradecir al anterior, sin embargo a veces es necesario para los usuarios poder compartir y actualizar información (por ejemplo, en una base de datos) y, si se organiza la tarea de entrada a la misma, se puede evitar el tener varias copias de la rutina.

Reubicación.

La técnica de multiprogramación requiere que varios programas ocupen la memoria al mismo tiempo. Sin embargo no se sabe con anticipación donde será cargado cada programa por lo que no es práctico usar direccionamiento absoluto de memoria.

Organización física.

Debido al costo de una memoria principal rápida, éste se usa en conjunto con una memoria secundaria mucho más lenta (y por consiguiente, barata) a fines de extender su capacidad.

Organización lógica.

Aunque la mayor parte de las memorias son organizadas linealmente con un

direccionamiento secuencial, esto difícilmente concuerde con el camino seguido por el

programa, debido al uso de procedimientos, funciones, subrutinas, arreglos, etc

Conclusion

Nos enseña un como funcionan los lenguajes y la diferencia del lenguaje maquina y el ensambaldor nos ayuda a aprender algo de el tipo de almacenamiento en memoria y que ventajas y desventajas se tienen a la hora de utilizarlos,
el código intermedio puedes irse optimizando hacia el código objeto de bajo nivel.
Y que cada equipo tiene un diferente procesador para su lenguaje máquina, por lo cual se crean los lenguajes ensambladores, que a pesar de ya no ser tan utilizadas en la actualidad, fue de vital importancia en el pasado para el desarrollo de software y cuando se requería la manipulación directa del hardware o en rendimientos inusuales de los equipos.

Bibliografía

Beck. Software de Sistemas, Introducción a la programación de Sistemas. Addison-Wesley Iberoamericana.

Guerra Crespo. Héctor. Compiladores. Ed. Tecnológica didáctica.

Vázquez Gaudioso. Elena, García Saiz. Tomas. Introducción a la teoría de autómatas, gramática y lenguajes. Ed. Universitaria Ramón Areces

Meloni. Brenda, Giro. Juan, Vázquez. Juan, Constable. Leticia. Lenguajes formales y teoría de autómatas. Ed. Alfaomega

Millan Borrajo. Daniel, Fernández Martínez. Paloma, Isasi Viñuela. Pedro. Lenguajes, Gramáticas y Autómatas: Un enfoque práctico. Ed. Addison Wesley

Reporte

Lo que yo entendi de la unidad es primero que nada que es un registro el cual es la distribucion de un proceso que por lo general son de uso exclusivo ¿como se utilizan o quien los utiliza? Se utilizan regularmente para el manejador de marcos de pilas en las documentaciones que se encuentran y generalmente son de estado dinamico tan bien que es la distribucion que por lo general se conforma de 14 registros o datos exclusivos y se utilizan para clasificar las variables Aprendimos tan bien un poco del lenguaje ensamblador que regularmente son series de instrucciones dadas a traves de el microprocesador y regularmente son en binarios o en codigo maquina sus caracteristicas regularmente los lenguajes ensambladores cuentan con que solo ofrecen nombres simbolicos de instrucciones y parametros dados a diferencia de los lenguajes modulares que se utilizan para resolver problemas mas complejos y que se resuelve regularmente en partes aprendimos tan bien un poco de los que es el lenguaje maquina que a comparacion de los otros dos este solo funciona con 0 y 1 y que solo comprende un conjunto de acciones dadas o instrucciones

Aprendimos un poco de como administrar o que es la administracion de memoria y esto dice que se utiliza para la administacion en memoria y que regularmente esta almacenada en un apartamento local son considerados como arreglos lineales

Conceptos

Código intermedio: El Front-end traduce el programa fuente en una representación de código intermedio, y el back-end traduce esta representación en código final
Código objeto: Al código que resulta de la compilación del código fuente. Puede ser en lenguaje maquina o bytecode, y puede distribuirse en varios archivos que corresponden a cada código fuente compilado
Lenguaje de bajo nivel: Aquel en el que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan
Arquitectura computacional: Es el diseño conceptual y la estructura operacional fundamental de un sistema de computadoras
Memoria: Es el dispositivo que retiene, memoriza o almacena datos informáticos durante algún periodo de tiempo