Luis Enrique Sosa Hernández
David Samuel Couary Juárez
Lenguajes y autómatas II
Maestra:
Martínez Moreno Martha

12:00-13:00

GENERACION DE CÓDIGO INTERMEDIO

INSTITUTO TECNOLÓGICO DE VERACRUZ

Contenido

Introducción

En este documento se presenta la forma de representar un código de asignaciones en tres lenguajes diferentes, así como una breve explicación de los mismos, lenguaje R, lenguaje ensamblador y jasmin assembler, los cuales son utilizados en la generación de código intermedio una de las fases del compilador.

Código en lenguaje R

```
#Asignaciones en R

a=10;

b=10.50

c="Hola mundo"

d=TRUE

e=FALSE

a=a+15+20
```

Como podemos observar R es un lenguaje que no tiene estructura.

Es un lenguaje poco typeado, lo que significa que no es necesario asignarle un tipo de dato a nuestras variables, también es importante mencionar que las variables pueden cambiar de tipo de dato, según le convenga al programador.

Los tipos de dato que se manejan, son enteros, flotantes, cadena de caracteres y booleanos, esto en el caso para las asignaciones simples, también maneja estructura como arreglos y objetos.

Código en ensamblador

```
.model small
.stack 64
.data
;Enteros
a dw 10d;var1=255
;Flotantes
b dd 10.50;var6 65536.0
;Cadenas
c db "Hola mundo",'$';var9="Hola mundo"
;Boleanos
d db 1d;var12=true;
e db 0d;var12=false;
;Tambien se pueden declarar variables sin dato con ?
f dw ?;Esto no se puede en R
.code
Turbo PROC FAR;declaracion del procedimiento principal
;Suma de tres numeros utilizando direccionamiento inmediato
mov ax,a;pone el valor de la variable a en ax
add ax,15d;suma a ax el numero 15d
```

add ax,20d;suma a ax el numero 20d

mov a,ax;guarda el resultado en la variable a

mov ah,4Ch;Devuelve el control al DOS

int 21h

TURBO ENDP ;fin del procedimiento principal

END;fin del programa

En ensamblador podemos notar significativas diferencias, la principal es que las líneas de código se triplican.

Aquí ya es necesario declarar las variables con un tipo de dato, debe recalcarse que la base numérica por defecto es Hexadecimal, por lo cual, si quiere usarse decimal, es posible indicárselo a ensamblador mediante el símbolo d al final de cada valor.

Ensamblador cuenta con una estructura definida la cual es:

.model small

.stack 64

.data

Dentro de data van todas las declaraciones de variables que se requieran

Para las asignaciones se sigue la siguiente estructura, por ejemplo:

- a dw 10d

donde a es el nombre de la etiqueta, dw el tipo de dato y 10d el valor en decimal

.code

En esta sección va todo el código necesario para realizar las instrucciones que se requieran , como realizar una suma.

Turbo PROC FAR

TURBO ENDP

END

Como dato importante, en ensamblador, para realizar la asignación con operación, es necesario hacer uso de los registros como se ve en el ejemplo de código, y al final regresar el ultimo valor de registro a la variable en la cual queremos asignar el resultado de la operación aritmética.

Código en Jasmin

```
; example.j


; Generated by ClassFileAnalyzer (Can)

; Analyzer and Disassembler for Java class files

; (Jasmin syntax 2, http://jasmin.sourceforge.ne

;

; ClassFileAnalyzer, version 0.7.0

.bytecode 52.0

.source example.java

.class example

.super java/lang/Object


.method <init>()V

  .limit stack 1

  .limit locals 1
```

```
  .line 1
  0: aload_0
  1: invokespecial java/lang/Object/<init>()V
  4: return
.end method

.method public static main([Ljava/lang/String;)V
  .limit stack 2
  .limit locals 7
  .line 3
  0: bipush 10
  2: istore_1
  .line 4
  3: ldc2_w 10.5
  6: dstore_2
  .line 5
  7: ldc "hola mundo"
  9: astore 4
  .line 6
  11: iconst_1
  12: istore 5
  .line 7
  14: iconst_0
  15: istore 6
  .line 8
```

```
  17: iload_1
  18: bipush 15
  20: iadd
  21: bipush 20
  23: iadd
  24: istore_1
  .line 10
  25: return
.end method
```

En jasmin encontramos semejanza con ensamblador , como que las líneas de código son extensas, y tienen una estructura parecida.

La diferencia es que jasmin no trabaja con registros, trabaja exclusivamente con la pila o stack.

Estructura

```
.bytecode 52.0
.source example.java
.class example
.super java/lang/Object
```

Cuenta con un constructor en el cual se inicializan los valores

```
.method <init>()V
  .limit stack 1
  .limit locals 1
  .line 1
  0: aload_0
  1: invokespecial java/lang/Object/<init>()V
  4: return
```

Cuenta con su método principal en el cual se colocan las declaraciones asignaciones e instrucciones.

.method public static main([Ljava/lang/String;)V

Para las asignaciones en jasmin hacemos uso de la pila de la siguiente manera

Ejemplo

- 0: bipush 10
- 2: istore_1
- Donde bipush es el tipo de dato 10 el valor que se le asigna y istore_1 el lugar donde se almacenara ese dato

25: return

.end method

Conclusión

Podemos concluir que las diferencias entre un lenguaje de alto nivel y uno de bajo nivel son abismales. Y específicamente en la cantidad de líneas de código.

Los lenguajes de alto nivel, nos han brindado muchas comodidades a los programadores, como ahorrarnos muchas líneas de código al programar, gracias a los compiladores de los actuales lenguajes de programación, podemos escribir código de una manera más sencilla, sin necesidad de involúcranos tanto con el hardware.

Con ensamblador y jasmin nos encontramos más cerca del lenguaje máquina, por lo tanto, es mucho más complejo y difícil de programar, pero son necesarios para la generación de código intermedio.

Por lo tanto, si el hardware no fuera hard seria soft.

Anexo:

Para la generación del código en jasmin, se usó una herramienta llamada classfileanalyzer la cual se puede recuperar del siguiente enlace:

http://classfileanalyzer.javaseiten.de/

En la misma página se explican todos los pasos a seguir para le generación del código en jasmin.

Referencias

Roeder, H. ClassFileAnalyzer - Analyzer and Disassembler for Java class files. Classfileanalyzer.javaseiten.de. Recuperado 18 octubre 2017, de http://classfileanalyzer.javaseiten.de/