Generazione del bytecode

CodeGenerator

Memorizza le istruzioni che man mano vengono generate durante la traduzione (si tratta di un meccanismo di traduzione on-the-fly guidato dalla sintassi)

La lista di istruzioni viene memorizzata in una struttura apposita, come oggetti di tipo Instructions.

Le istruzioni verranno generate on-the-fly durante il processo di Parsing (di Parsificazione).

All'interno della classe CodeGenerator sono presenti due metodi emit() con o senza operando che permettono di aggiungere istruzioni all'interno del codice.

Inoltre è presente il metodo emitLabel() che permette di aggiungere etichette di salto nel codice, insieme al metodo newLabel() che permette la gestione di incremento delle etichette, in quanto durante la creazione di etichette per salto, quest'ultime non possono avere lo stesso identificativo, ovvero non possono essere presenti due etichette identificate con L0, L1, ecc..

Il metodo toJasmin() permettere infine di produrre il file finale denominato Output.j

Il formato di Output.j che viene generato, è riconosciuto da Jasmin per la generazione del file .class

Instruction

Permette la creazione di istruzioni.

Sono presenti due metodi che permettono la creazione di istruzioni con o senza operando.

Il metodo toJasmin() restituisce l'istruzione nel formato adeguato per l'assembler Jasmin

OpCode

Enumerazione dei nomi mnemonici relativi ai comando del linguaggio target che fa riferimento alle istruzioni della JVM

SymbolTable

Consente di tenere traccia degli identificatori che vengono richiamati nel programma sorgente andando a fare delle operazioni di inizializzazione o aggiornamento di variabili.

Occorre avere una struttura dati che permetta l'implementazione della tabella dei simboli.

In questo caso viene utilizzato un oggetto di tipo HashMap

Questa struttura dati consente di andare a gestire l'associazione di un identificatore a un particolare indirizzo della tabella dei simboli a partire dall'indirizzo 0 e mediante il metodo

LookupAdress() è possibile ricercare un identificatore all'interno della tabella dei simboli e di restituire il valore -1 nel caso in cui quell'identificatore non è stato inserito nella tabella dei simboli.

A livello di linguaggio bytecode, gli indirizzi della symbol table, verranno utilizzati come argomento dei comandi iload e istore

Ad esempio dopo un operazione di lettura bisogna memorizzare il valore con l'istruzione di bytecode istore 0, dove 0 indica la posizione in cui abbiamo memorizzato il valore della variabile letta nella symbol table.

Nella symbol table si trova l'associazione indirizzo – valore

Le istruzioni della grammatica in uso che consentono l'aggiornamento o la dichiarazione di variabili sono:

assign (assegnare un valore ad un identificatore o ad una lista di identificatori)

read (legge in input un intero non negativo e controlla se la variabile ha già un valore e bisogna aggiornarlo oppure se la variabile non è presente è bisogna assegnare un nuovo valore)

Tag

Viene utilizzata per rappresentare i nomi dei token

Per i token che corrispondono a un solo carattere (ad esempio: +) si può utilizzare il codice ASCII del carattere

Token

Viene utilizzata per rappresentare i token

Word

Questa classe viene derivata da Token, e viene utilizzata per rappresentare i token che corrispondono a:

identificatori

parole chiave

operatori relazionali

elementi della sintassi che consistono di più caratteri

NumberTok

Permette di rappresentare i token che corrispondo a numeri