

Linguaggi Formali e Traduttori

5.3 Codice intermedio

- Sommario
- Java Virtual Machine
- Componenti della JVM
- Struttura di un frame della JVM
- Gestione della pila degli operandi
- Operazioni aritmetiche e su bit
- Gestione degli array
- Controllo di flusso

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Problema

- Stabilito che il programma da tradurre è sintatticamente corretto, il compilatore lo deve tradurre in un programma equivalente scritto in **codice intermedio**.

In questa lezione

- Adottiamo il bytecode di Java come codice intermedio per la traduzione.
- Riepiloghiamo la struttura e il significato delle istruzioni più importanti della Java Virtual Machine (JVM).

Riferimenti esterni

- [Java Language and Virtual Machine Specifications](#)
- [JVM Instruction set](#)

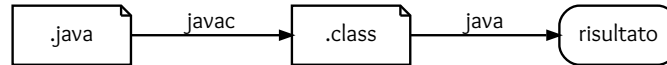
Java Virtual Machine

La **Java Virtual Machine** (JVM) è un **interprete** in grado di eseguire **bytecode**.

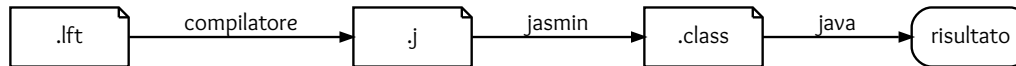
Caratteristiche principali della JVM

- Macchina virtuale basata su **pila**.
- Istruzioni di basso livello (gestione della pila) ed alto livello (**oggetti**).
- **Garbage collector** (la memoria inutilizzata viene reclamata automaticamente).

Uso tipico della JVM



In questo corso



- Il file `.j` contiene bytecode JVM in formato mnemonico (facile da produrre/leggere).
- Il file `.class` contiene bytecode JVM in formato binario.
- Usiamo **Jasmin** per tradurre il bytecode dal formato mnemonico a quello binario.

Componenti della JVM

- Un' **area metodi** che contiene il bytecode corrispondente ai metodi di tutte le classi usate da un'applicazione.
- Un insieme di **registri** che contengono informazioni essenziali sullo stato della macchina virtuale. Tra tutti, il **program counter** contiene l'indirizzo della prossima istruzione da eseguire.
- Una **pila di frame**, uno per ogni metodo in esecuzione. A sua volta, ogni frame è composto da:
 - una **pila degli operandi** usata per la valutazione di risultati temporanei;
 - un **array di variabili locali** usato per la memorizzazione delle variabili locali e degli argomenti del metodo.
- Un **heap** in cui vengono allocati gli oggetti.

Nota

Nelle applicazioni con thread multipli in esecuzione esistono copie distinte della pila dei frame e di alcuni registri per ciascun thread.

Struttura di un frame della JVM

Ogni **frame** corrisponde a un metodo in esecuzione e contiene:

- **argomenti e variabili locali** (indirizzati a partire da 0);
- **pila degli operandi** (cresce/calza durante l'esecuzione del metodo).

Nome	Slot n.	Valore
<i>a</i>	0	42
<i>b</i>	1	true
<i>x</i>	2	7
<i>y</i>	3	23
<i>z</i>	4	'c'
—	—	5
—	—	7
		⋮

```
static void m(int a, boolean b) {  
    int x, y;  
    char z;  
  
    ... 5 * x ...  
}
```

Nota

- Nei metodi non statici il primo argomento è il riferimento all'oggetto ricevente (this).

Gestione della pila degli operandi

Istruzione	Prima	Dopo	Descrizione
ldc <i>v</i>		<i>v</i>	carica <i>v</i> sulla pila
iload <i>&x</i>		<i>v</i>	carica il valore di <i>x</i> sulla pila
istore <i>&x</i>	<i>v</i>		assegna <i>v</i> a <i>x</i>
pop	<i>v</i>		rimuove il valore in cima alla pila
dup	<i>v</i>	<i>v v</i>	duplica il valore in cima alla pila
swap	<i>v</i> ₁ <i>v</i> ₂	<i>v</i> ₂ <i>v</i> ₁	scambia i due valori in cima alla pila

Note

- Il valore in cima alla pila è quello più a destra.
- Le istruzioni **iload** e **istore** hanno come argomento l'indirizzo – e non il nome – della variabile *x* nel frame del metodo corrente.

Operazioni aritmetiche e su bit

Istruzione	Prima	Dopo	Descrizione
ineg	v	v	negazione
iadd	$v_1 \ v_2$	v	somma $v_1 + v_2$
isub	$v_1 \ v_2$	v	sottrazione $v_1 - v_2$
imul	$v_1 \ v_2$	v	moltiplicazione $v_1 \times v_2$
idiv	$v_1 \ v_2$	v	divisione v_1 / v_2
irem	$v_1 \ v_2$	v	resto della divisione v_1 / v_2
iand	$v_1 \ v_2$	v	congiunzione bit a bit
ior	$v_1 \ v_2$	v	disgiunzione bit a bit

Note

- Il valore in cima alla pila è quello più a destra.
- Nelle operazioni binarie (es. **isub**) il secondo operando è quello in cima alla pila.

Gestione degli array

Istruzione	Prima	Dopo	Descrizione
newarray	n	a	crea un array di n elementi
arraylength	a	n	dimensione dell'array a
iaload	$a\ i$	v	carica $a[i]$ sulla pila
iastore	$a\ i\ v$		assegna v ad $a[i]$

Nota

- a è un riferimento all'array nell'heap.

Controllo di flusso

Istruzione	Prima	Dopo	Descrizione
goto l			salta a l
if_icmpeq l	$v_1 \ v_2$		salta a l se $v_1 = v_2$
if_icmpne l	$v_1 \ v_2$		salta a l se $v_1 \neq v_2$
if_icmple l	$v_1 \ v_2$		salta a l se $v_1 \leq v_2$
if_icmpge l	$v_1 \ v_2$		salta a l se $v_1 \geq v_2$
if_icmplt l	$v_1 \ v_2$		salta a l se $v_1 < v_2$
if_icmpgt l	$v_1 \ v_2$		salta a l se $v_1 > v_2$
invokestatic m	$v_1 \cdots v_n$	$v?$	invoca $m(v_1, \dots, v_n)$
return			termina il metodo
ireturn	v		termina il metodo restituendo v

Nota

- $v?$ è presente solo se il metodo invocato ha un tipo di ritorno diverso da void.