

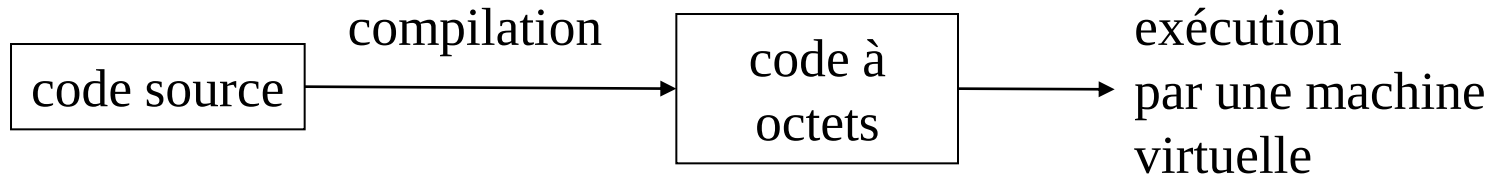
Code à octets

Introduction historique

Organisation du code à octets

Organisation de la machine virtuelle

Introduction



Code à octets (*bytecode*)

Code intermédiaire en binaire

Code intermédiaire :

- indépendant de la machine cible
- non exécutable directement par les processeurs
- pas de distinction entre registres et mémoire centrale

Binaire :

chaque code d'opération (iload, goto...) est codé sur un octet

Introduction historique

Haut niveau	Bas niveau				Exécution
	sans distinction registre/mémoire centrale		avec distinction registre/mémoire centrale		
	mnémonique	binaire	mnémonique	binaire	
				code machine 	processeur
			langage d'assemblage	code machine 	processeur
C, C++	code intermédiaire		langage d'assemblage	code machine 	processeur

écrire une fois, compiler n'importe où

Introduction historique

Haut niveau	Bas niveau				Exécution
	sans distinction registre/mémoire centrale		avec distinction registre/mémoire centrale		
	mnémonique	binaire	mnémonique	binaire	
	script shell				interpréteur de commandes
Java		code à octets			machine virtuelle Java
	langage d'assemblage Java	code à octets			machine virtuelle Java

compiler une fois, exécuter n'importe où

Code à octets pour une machine virtuelle Java

Organisation du code à octets

Un fichier pour chaque classe ou interface

Références entre fichiers

Le code pour une classe ou interface contient :

- la table des symboles avec les noms des membres et des types
- le code des méthodes, séparément pour chaque méthode

Organisation du code

Un fichier pour chaque classe ou interface

Identification du format et de la version du code à octets

Groupe des constantes

Permissions

Indice du nom de la classe ou interface

Indice du nom de la classe mère

Indices des noms des interfaces

Champs déclarés

Méthodes déclarées ou redéclarées

Attributs

Groupe des constantes

Deux octets pour le nombre de constantes

La séquence des constantes

Pour chaque constante :

- un octet pour la catégorie de constante (nom de classe, de champ, de méthode ou d'interface, identificateur, constante numérique...)
- le contenu de la constante, en un ou plusieurs champs suivant la catégorie

Le code des méthodes

Une méthode est représentée par

- des informations en longueur fixe
- une séquence d'attributs, dont le code et les exceptions

Chaque attribut est représenté par

- certaines informations en longueur fixe
- d'autres en longueur variable

L'attribut code contient entre autres la séquence des instructions

Instructions

Une instruction occupe 1 ou plusieurs octets

1 octet pour le code d'opération (iload, goto...)

1 ou 2 octets supplémentaires par opérande

Certains codes d'opération précisent

- le type : iload, lload, fload, dload, aload (int, long, float, double, reference)
- l'opérande : iload_0, iload_1, iload_2, iload_3

Machine à pile

load : empiler une variable

store : dépiler et sauvegarder une variable

push : empiler une constante

pop : dépiler

Une instruction à deux opérandes dépile deux cases de la pile,
puis empile le résultat

La mémoire de la machine virtuelle

Compteur d'instructions

Piles d'exécution : une par fil d'exécution (*thread*)

- contiennent les blocs d'exécution des méthodes

Tas

- contiennent les objets (instances de classes et tableaux)

Zone des méthodes, y compris la séquence d'instructions

Groupe de constantes

- contient les tables des symboles des classes et interfaces utilisées

Appel d'une méthode

L'instruction `invokevirtual` prend comme opérande une entrée du groupe de constantes qui décrit la classe de l'objet et la méthode appelée