8. Виртуална машина

8.1. Структура

Виртуалната машина е абстрактна изчислителна машина. Също както реална машина има множество от инструкции. Виртуалната машина е независима от входния език на компилираната програма.

Тъй като виртуалната машина е създадена за учебни цели, нейните инструкции и операнди се представят с кодове от целочислен тип.

Виртуалната машина има следната структура:

- програмен брояч и масив от инструкции за изпълнение

- стек

- heap

Използват се следните структури от данни:

- масив от инструкции – съдържа кодовете на генерираните инструкции в числен вид.

- операнден стек – съдържа стойностите на операндите за изпълнение на съответните инструкции.

- стек от кадри (frames) – при всяко извикване на функция се създава нов стеков кадър. В кадъра се съдържа адрес за връщане и масив от локални променливи. При излизане от функцията стековият кадър се унищожава.

- масив за глобални променливи – съдържа глобални променливи, които са достъпни във всяка точка от програмата.

- heap – използва се за съхранение на динамично създадени масиви. Реализира се като масив от масиви.

8.2. Команди

Командите са представени в следния вид:

- **мнемоничен код на инструкцията**

Код: числов код на инструкцията

Формат: мнемоничен код, операнд1, операнд2, …

Операнден стек: ... (съдържание на стека преди изпълнение) -->
... (съдържание на стека след изпълнение)

Описание на командата

8.2.1. Операции с константи

- **ICONST**

Записване на целочислена константа в операндния стек.

Код: 0

Формат: iconst, value

Операнден стек: ... --> ..., value

Value трябва да е от целочислен тип. Стойността се записва на върха на операндния стек.

8.2.2 Операции за зареждане

- **GLOAD**

Зареждане на стойност от масива с глобални променливи.

Код: 1

Формат: gload, index

Операнден стек: ... --> ..., value

Стойността на глобалната променливата, намираща се на index се записва на върха на операндния стека.

- ILOAD

Зареждане на примитивна стойност от масива с локални променливи.

Код: 2

Формат: iload, index

Операнден стек: ... --> ..., value

Стойността на локалната променлива, намираща се на index, се записва на върха на операндния стека.

- ALOAD

Зареждане на референция от масива с локални променливи.

Код: 3

Формат: aload, index

Операнден стек: ... --> ..., value

Стойността на локалната променлива, намираща се на index, се записва на върха на операндния стек.

- IALOAD

Зареждане на цяло число от масив.

Код: 4

Формат: iaload

Операндния стек: ..., arrayref, index --> ..., value

Arrayref трябва да е референция към масив от целочислен тип. Arrayref и index биват премахнати от върха на стека. Стойността от целочисления масив, намираща се на позиция index в масива с референция arrayref, е извлечена и записана на върха на операндния стек.

- BALOAD

Зареждане на булева стойност от масив.

Код: 5

Формат: baload

Операндния стек: ..., arrayref, index --> ..., value

Arrayref трябва да е референция към масив от булев тип. Arrayref и index биват премахнати от върха на стека. Стойността от булевия масив, намираща се на позиция index в масива с референция arrayref, е извлечена и записана на върха на операндния стека.

- CALOAD

Зареждане на символ от масив.

Код: 6

Формат: caload

Операнден стек: ..., arrayref, index --> ..., value

Arrayref трябва да е референция към масив от символен тип. Arrayref и index биват премахнати от върха на стека. Стойността от символния масив, намираща се на позиция index в масива с референция arrayref, е извлечена и записана на върха на операндния стек.

8.2.3. Операции за запис на стойности

- GSTORE

Записване на стойност в масива с глобални променливи.

Код: 7

Формат: gstore, index

Операнден стек: ..., value --> ...

Стойността на върха на операндния стек е премахната и записана в масива с глобални променливи на позиция index.

- **ISTORE**

Записване на примитивна стойност в масива с локални променливи.

Код: 8

Формат: istore, index

Операнден стек: ..., value --> ...

Стойността на върха на операндния стек е премахната и записана в масива с локални променливи на позиция index.

- **ASTORE**

Записване на референция в масива с локални променливи.

Код: 9

Формат: astore, index

Операнден стек: ..., value --> ...

Стойността на върха на операндния стек е премахната и записана в масива с локални променливи на позиция index.

- IASTORE

Запазване на стойност в целочислен масив.

Код: 10

Формат: iastore

Операнден стек: ..., arrayref, index, value --> ...

Arrayref трябва да е референция към масив от целочислен тип. Value трябва да е от целочислен тип. Arrayref, index и value биват премахнати от върха на стека. Целочислената стойност value е запазена на позиция index в масива с референция arrayref.

- BASTORE

Запазване на стойност в булев масив.

Код: 11

Формат: bastore

Операнден стек: ..., arrayref, index, value --> ...

Arrayref трябва да е референция към масив от булев тип. Value трябва да е от булев тип. Arrayref, index и value биват премахнати от върха на стека. Булевата стойност value е запазена на позиция index в масива с референция arrayref.

- CASTORE

Запазване на стойност в символен масив.

Код: 12

Формат: castore

Операнден стек: ..., arrayref, index, value --> ...

Arrayref трябва да е референция към масив от символен тип. Value трябва да е от символен тип. Arrayref, index и value биват премахнати от върха на стека. Символната стойност value е запазена на позиция index в масива с референция arrayref.

- CASTOREALL

Запазване на стойности в символен масив.

Код: 13

Формат: castore

Операнден стек: ..., arrayref, value1, …, value n, nargs --> ...

Arrayref трябва да е референция към масив от символен тип. Value1 … value n трябва да са от символен тип, а nargs е броят на елементите. Arrayref, values и nargs биват премахнати от върха на стека. Символните стойности са запазени в масива с референция arrayref.

8.2.4. Аритметични и логически операции

- IADD

Целочислено събиране.

Код: 14

Формат: iadd

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията е value1 + value2, който е записан на върха на стека.

- ISUB

Целочислено изваждане.

Код: 15

Формат: isub

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията е value1 - value2, който е записан на върха на стека.

- IMUL

Целочислено умножение.

Код: 16

Формат: imul

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията е value1 * value2, който е записан на върха на стека.

- IDIV

Целочислено деление.

Код: 17

Формат: idiv

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията е value1 / value2, който е записан на върха на стека.

- **IREM**

Целочислено деление с остатък.

Код: 18

Формат: irem

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията е value1 - (value1 / value2) * value2, който е записан на върха на стека.

- **INEG**

Целочислено отрицание.

Код: 19

Формат: ineg

Операнден стек: ..., value --> ..., result

Стойността value трябва да е от целочислен тип. Тя бива премахната от операндния стек. Резултатът (result) от операцията е - value, който е записан на върха на стека.

- IAND

Логическо И.

Код: 20

Формат: iand

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията е value1 & value2, който е записан на върха на стека.

- IOR

Логическо ИЛИ.

Код: 21

Формат: ior

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията е value1 | value2, който е записан на върха на стека.

- NOT

Логическо отрицание.

Код: 22

Формат: not

Операнден стек: ..., value --> ..., result

Стойността value трябва да е от булев тип. Бива премахната от операндния стек. Резултатът(result) от операцията е value xor 1, който е записан на върха на стека.

8.2.5. Операции за сравнение

- ICMPEQ

Операция за проверка на еквивалентност.

Код: 23

Формат: icmpeq

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията value1 == value2 е булев (0 или 1) и е записан на върха на стека.

- ICMPNE

Операция за проверка на нееквивалентност.

Код: 24

Формат: icmpne

Операндния стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията value1 != value2 е булев (0 или 1) и е записан на върха на стека.

- ICMPLT

Операция проверка за по - малко.

Код: 25

Формат: icmplt

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията value1 < value2 е булев (0 или 1) и е записан на върха на стека.

- ICMPLE

Операция проверка за по – малко или равно.

Код: 26

Формат: icmple

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стек. Резултатът (result) от операцията value1 <= value2 е булев (0 или 1) и е записан на върха на стека.

- ICMPGT

Операция проверка за по - голямо.

Код: 27

Формат: icmpgt

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията value1 > value2 е булев(0 или 1) и е записан на върха на стека.

- ICMPGE

Операция проверка за по – голямо или равно.

Код: 28

Формат: icmpge

Операнден стек: ..., value1, value2 --> ..., result

Двете стойности value1 и value2 трябва да са от целочислен тип. Те биват премахнати от операндния стека. Резултатът (result) от операцията value1 >= value2 е булев (0 или 1) и е записан на върха на стека.

- IF_TRUE

Условен преход при изпълнено условие (true).

Код: 29

Формат: if_true, address

Операнден стек: ..., value --> ...

Стойността value трябва да е от целочислен тип. Тя бива премахната от операндния стек. Ако value == 1, програмата продължава с изпълнение на инструкция, намираща се на address.

- IF_FALSE

Условен преход при неизпълнено условие (false).

Код: 30

Формат: if_false, address

Операнден стек: ..., value --> ...

Стойността value трябва да е от булев тип. Тя бива премахната от операндния стек. Ако value == 0, програмата продължава с изпълнение на инструкция, намираща се на address.

8.2.6. Операции за управление.

- GOTO

Безусловен преход.

Код: 31

Формат: goto, address

Операнден стек: не се променя

Address трябва да е адрес на инструкция. Изпълнението на програмата продължава на address.

- **IRETURN**

Връщане на целочислена стойност от функция.

Код: 32

Формат: ireturn

Операнден стек: ..., value --> празно

Стойността на value трябва да е от целочислен, булев или символен тип. Тя бива премахната от операндния стек на текущия кадър и бива записана на върха на стека на предходния кадър. Текущият кадър бива унищожен, а управлението е върното на предходния кадър.

- **ARETURN**

Връщане на референтна стойност от функция.

Код: 33

Формат: areturn

Операнден стек: ..., value --> празно

Стойността на value трябва да е от референтен тип. Тя бива премахната от операндния стек на текущия кадър и бива записана на върха на стека на предходния кадър. Текущият кадър бива унищожен, а управлението е върното на предходния кадър.

- RETURN

Връщане на void стойност от функция.

Код: 34

Формат: return

Операнден стек: ... --> празно

Функцията трябва да връща стойност от тип void. Текущият кадър бива унищожен, а управлението е върното на предходния кадър.

- CALL

Извикване на подпрограма.

Код: 35

Формат: call, address

Операнден стек: ..., [arg1, [arg2, ... [argn]]], nargs --> ...

Nargs на брой параметри (съответно arg1, arg2, ..., argn) се премахват от операндния стек. Създава се нов кадър и в масива му от локални промеливи се записват параметрите в съответната последователност. Изпълнението на програмата продължава на address.

- HALT

Прекратяване изпълнението на виртуалната машина.

Код: 36

Формат: halt

Операнден стек: без промяна

8.2.7. Операции с референтни типове

- NEWARRAY

Създаване на масив.

Код: 37

Формат: newarray, type

Операнден стек: ..., count --> arrayref

Стойността count представя броя на елементите на масива. Тя трябва да е от целочислен тип. Стойността се премахва от операндния стек. Type представя

типа на масива и трябва да е една от следните целочислени стойности за съответния тип:

int – 0

char – 1

boolean – 2

Масив от тип type и дължина count е алокиран от heap-а. Референцията (arrayref) към този масив е записана на върха на операндния стек.

- **ARRAYLENGTH**

Връща размер на масив.

Код: 38

Формат: arraylength

Операнден стек: ..., arrayref --> length

Стойността arrayref трябва да е от референтен тип и да реферира масив. Тя е премахната от операндния стек. Дължината (length) на реферирания масив бива определена и записана на върха на операндния стек.

8.2.8. Входно-изходни операции

- **PRINT**

Визуализира стойности на стандартния изход.

Код: 39

Формат: print

Операнден стек: ..., value1, type1, …, value n, type n, nargs --> …

Value1, type1 … value n, type n са съответно стойности и техните типове, а nargs е броят на стойностите.

Type:

int – 0

char – 1

boolean – 2

string – 3

Value, type премахнати от операндния стек. Стойностите биват визуализирани на стандартния изход.

- READ

Прочита и записва стойности от стандартния вход.

Код: 40

Формат: read

Операнд стек: …, ctx1, index1, type1, ..., ctx n, index n, type n, nargs --> …

Ctx е контекста, в който ще се запише текущата прочетената стойност, index е позицията в контекста, type е типът на стойността, а nargs е броя на стойностите.

Type: Ctx:

int	– 0	global	– 0
char	– 1	local	– 1
boolean	– 2		
string	– 3		

Ctx, index, type, nargs биват премахнати от операндния стек, а стойностите биват записани на съответната позиция в съответния контекст.