Chapter 11-2 Jasmin Code Generation

Wuu Yang

National Chiao-Tung University, Taiwan, R.O.C.

first draft: June 16, 2018

current version: July 21, 2019

Copyright ©July 21, 2019 by Wuu Yang. All rights reserved.

Chapter outline: Generating Jasmin Code

Reference: `Jasmin Instructions.pdf`, `Jasmin User Guide.pdf`.

The whole minipascal program is translated into a single class in the Jasmin program:

```
.class public Sort
.super java/lang/Object
.field public static g6 [I
.field public static g10 I
.field public static g9 I
.field public static g7 I
.field public static g8 I

.method public static f0(I)V
...
return
.end method

.method public static f1(F)V
...
return
.end method
```

```
.method public static main([Ljava/lang/String;)V
.limit stack 50
.limit locals 50
ldc 50
multianewarray [I 1
putstatic Sort/g6 [I
...
return
.end method
```

Global Variables

The global variables in a Minipascal program are declared as class variables (i.e., `static field`) in the class.

```
.field public static g6 [I
.field public static g10 I
.field public static g9 I
.field public static g7 I
.field public static g8 I
```

You may access these global variables with

```
getstatic Sort/g6 [I
getstatic Sort/g10 I
putstatic Sort/g10 I
```

Read an Integer into a Local Variable

The local variables in a Minipascal function are implemented as local variables in a Jasmin method:

```
VAR input: integer;   // input is a local variable
input := readlnI();

getstatic java/lang/System/in Ljava/io/InputStream;
invokevirtual java/io/InputStream/read()I
istore 2
```

Print an Integer

To print an integer you may use the following method:

```
.method public static f0(I)V
.limit locals 5
.limit stack 5
getstatic java/lang/System/out Ljava/io/PrintStream;
iload 0
invokevirtual java/io/PrintStream/println(I)V
return
.end method
```

Functions in the Minipascal Programs

Each function in the Minipascal programs is translated into a public static method in the Jasmin program.

You may access these functions with

```
invokestatic Sort/f0(I)V
```

Symbolic Labels

You may use symbolic labels instead of the actual addresses in the Jasmin programs:

```
L37:
if_icmple L36
goto  L3
```

Global Arrays

A global array is implemented as a static field. The memory space is allocated at the beginning of the main program.

```
arr: ARRAY [1..50] OF Integer;  // arr is a global array
... := arr[start] ...
```

```
.field public static g6 [I ; g6 is an array of integers
```

```
getstatic Sort/g6 [I
iload 0
ldc 1
isub  ; start - 1 due to the lower bound of arr is 1, not 0.
iaload    ; load arr[start] onto the stack
```

```
.method public static main([Ljava/lang/String;)V
.limit stack 50
.limit locals 50
ldc 50
multianewarray [I 1          ; allocate memory space for g6
putstatic Sort/g6 [I
...
return
.end method
```

Example. Store a value into the global array

```
getstatic Sort/g6 [I      ; get arr
getstatic Sort/g7 I       ; get size
ldc 1
isub                      ; index is size - 1
iload 2                   ; load input
iastore                   ; arr[size] = input;
```

Example. Here is a complete sample function:

```
PROCEDURE ReadArr(VAR a: ARRAY [1..50] OF Integer);
    VAR input: integer;
    BEGIN
        size := 0;
        input := readlnI();
        WHILE input != 0 DO
        BEGIN
            size := size + 1;
            arr[size] := input;
            input := readlnI();
        END
    END;
```

```
.method public static f11([I)V
.limit stack 50
.limit locals 50
ldc 0
putstatic Sort/g7 I      ; size = 0
invokestatic Sort/f3()I
istore 2   ; input := readlnI();
goto  L3
L2:
getstatic Sort/g7 I
ldc 1
iadd
putstatic Sort/g7 I  ; size := size + 1
getstatic Sort/g6 [I  ; get arr
getstatic Sort/g7 I   ; get size
ldc 1
isub            ; size - 1
iload 2        ; load input
```

```
iastore          ; arr[size] = input;
invokestatic Sort/f3()I
istore 2         ; input := readlnI()
L3:
iload 2
ldc 0
if_icmpne L2     ; if input != 0 then jump to L2
return
.end method
```