Assignment 9

Learning goals:

Apply the concepts and methods presented in chapter “Code Generation” of the lecture to some exercise problems.

Presentation:

Please prepare the solutions for informal presentation in class, and fill in the online check list in ZEUS.

Task 1 Compiling expressions

Using the attributed grammar described on lecture slides 40-51, determine the sequence of MJVM instructions generated for the expression below. Moreover, represent the concrete syntax tree and operand descriptors as on lecture slide 52, and explain how operand descriptors are used to generate the instruction sequence during parsing.

```
(3 * a[i] + x) % P
```

`a` is a local variable at address 0 pointing to an integer array, `i` is a local variable at address 1, `x` is a global variable at address 7, and `P` is a constant with value 27.

Task 2 MJ compiler and decoder

Use the given expression of Task 1 in a complete (minimal) MJ source program and compile it using `MJ.Compiler` (see lecture slide 40 of the “Overview” chapter). Decode the generated object file using `MJ.Decode`, then locate and explain the generated code for the expression of Task 1. Addresses of variables may differ from Task 1.

Task 3 Compiling assignments

Using the attributed grammar described on lecture slides 40-56, determine the sequence of MJVM instructions generated for the assignment sequence below. Moreover, for each generated instruction, give the production and semantic action (method call of *Code* class) of the attributed grammar that emits the instruction. Also reference the lecture slide number where each semantic action is visible.

```
n = 3;
obj = new Dataset;
obj.val = new int[10];
obj.val[0] = n;
```

`n` is a local variable at address 0, `obj` is a global variable at address 1, `Dataset` is a class with 2 fields, where `val` is its first field. All variables and fields have been declared such that the given assignments are valid.

Task 4 MJ compiler and decoder

Same task as Task 2, but for the given assignment sequence of Task 3.

Task 5 Jump labels

The following Java code snippet uses `Code` and `Label` classes of the `MJ.CodeGen` package of the MicroJava compiler to generate code for an if-else statement. `x`, `y`, and `z` are operand descriptors referring to local variables at addresses 0, 1, and 2, respectively.

```
Label elseLabel = new Label();
Label endLabel = new Label();
Code.load(x); Code.load(y);
Code.put(Code.jge); elseLabel.putAdr();
Code.load(x); Code.assignTo(z);
Code.put(Code.jmp); endLabel.putAdr();
elseLabel.here();
Code.load(y); Code.assignTo(z);
endLabel.here();
```

Determine the MJVM instructions generated by this code snippet, assuming that the first generated instruction is located at code address 100. For each MJVM instruction, represent its code address, instruction including explicit operands, and instruction length (in bytes). Moreover, explain when the final jump target addresses are written to the code buffer during execution of the given Java code snippet.