Decoders: Ajay, Bien, Sarah
CMPE-152
5/9/19

Compiler Design Report

High-level Description of Compiler

Our language is called **C-ish**, because it takes aspects of C and allows the code to be written in a way that compiles for the Java Virtual Machine (JVM). It emits Jasmin Assembly code that can directly be run on the JVM. We allow for the following constructs in our language:

- Types with type checking, including integers and floats.
- Basic arithmetic operations with operator precedence.
- Assignment statements.
- A conditional control statement in the form of IF loops.
- A looping control statement in the form of DO WHILE loops.
- Functions with calls and returns in the form of a basic add operation that loads parameters passed via the local variable stack. In our add function, parameters are passed by *value*.
- Allows for non-trivial sample programs to be written in the source language C-ish.
- Our compiler generates Jasmin assembly code that can be successfully assembled, and the resulting .class file can be executed with no crashes (such as null pointer exceptions).

Instructions for Building and Running Our Compiler

1. First, drop the *jasmin.jar* & *PascalRTL.jar* files into the root of the project directory (if not already present).
2. Run the following commands in order to create a *.class file and execute it with the JVM and compiler with Pascal RTL library:
    a. java -jar jasmin.jar C_main.j
    b. java -cp .:PascalRTL.jar C_main

UML Diagrams

The UML Diagrams image is included in our Decoders zip file as ‘Compiler UML.jpg’ .

Language Grammar

The Syntax Diagrams for our Language grammar is included in our Decoders zip file as ‘C.g4.html’. This webpage contains all of our syntax diagrams for the Compiler we created. Please unzip the whole Decoders.zip file to view the diagrams, as the images folder is needed too.

```
grammar C;

@header {
```

Decoders: Ajay, Bien, Sarah
CMPE-152
5/9/19

```
#include "wci/intermediate/TypeSpec.h"
using namespace wci::intermediate;
}

program     : ( func )* header main_block ;
header      : VOID MAIN '(' (parm_decl_list)* ')' ;
main_block  : '{' (decl_list)? main ;
main     : (stmt_list)? main_closing '}' ;
main_closing: RETURN ';' ;

decl_list      : var_list ( ';' var_list)* ';';
var_list       : var_type var_name ( ',' var_name)* ;
var_name       : IDENTIFIER ; //varId
var_type      : types ; //typeId


types locals [ TypeSpec *type = nullptr ]
        : INTEGER #integerType
        | FLOAT #floatType
        | VOID #voidType
        ;

stmt
    : assignment_stmt ';' #assignmentStmt
    | if_stmt #ifStmt
    | do_while_stmt ';' #doWhileStmt
    | func_assignment #funcAssignmentStmt
    | func_call ';' #funcCallStmt
    | printf_stmt ';' #printfStmt //not overriding this one.
    ;

stmt_list          : stmt ( stmt )*;
assignment_stmt    : variable '=' expr ;
if_stmt                 : if_part then_part ;
if_part            : IF if_rel_expr ;
then_part          : '{' stmt_list '}';
do_while_stmt      : do_stmt while_stmt ;
do_stmt            : DO '{' stmt_list '}' ;
```

```
while_stmt         : WHILE do_while_rel_expr;
func_assignment    : func_name '=' stmt ;
func_call          : func_name '(' (call_param_list)* ')';
call_param_list    : parm ( ',' )? ;

func         : func_decl func_block ;
func_decl    : var_type var_name '(' ( parm_decl_list )? ')' ;
func_block   : '{' (decl_list)? (func_stmt_list)? func_closing '}' ;
func_closing : RETURN func_name ';' ;

func_stmt_list
                 : func_add #func_add_stmt
                 | #empty_func_stmt
                 ;

func_add    : variable '=' variable '+' variable ';' ;

parm_decl_list : parm_decl ( ',' parm_decl )* ;
parm_decl      : var_type var_name ;


if_rel_expr        : '(' expr rel_op expr ')' ;
do_while_rel_expr  : '(' expr rel_op expr ')' ;


printf_stmt      : 'printf' '("' our_string '"' (',' expr)* ')';
our_string       : IDENTIFIER '-->' PRINT_IDENTIFIER;

variable : IDENTIFIER;

func_name locals [ TypeSpec *type = nullptr ]
    : variable;

parm locals [ TypeSpec *type = nullptr ]
    : expr;

expr locals [ TypeSpec *type = nullptr ]
    : expr mul_div_op expr #mulDivExpr
```

```
    | expr add_sub_op expr #addSubExpr
    | number #unsignedNumberExpr
    | signedNumber #signedNumberExpr
    | variable #variableExpr
    | '(' expr ')' #parenExpr
    | t_f_op #boolExpr
    ;

signedNumber locals [ TypeSpec *type = nullptr ]
    : sign number
    ;
sign      : '+' | '-' ;

number locals [TypeSpec *type = nullptr]
    : INTEGER_NUM #integerNumConst
    | FLOAT_NUM #floatNumConst
    ;

mul_div_op : MUL_OP | DIV_OP ;
add_sub_op : ADD_OP | SUB_OP ;
rel_op     : EQ_OP | NE_OP | LT_OP | LE_OP | GT_OP | GE_OP ;
t_f_op     : TRUE | FALSE ;

INTEGER  : 'int' ;
FLOAT    : 'float' ;
VOID     : 'void' ;
RETURN   : 'return' ;
IF       : 'if' ;
TRUE     : 'true' ;
FALSE    : 'false' ;
MAIN     : 'main' ;
DO       : 'do' ;
WHILE    : 'while' ;

IDENTIFIER     :  [a-zA-Z][a-zA-Z0-9]* ;
PRINT_IDENTIFIER: '%' [a-z];

INTEGER_NUM    : [0-9]+ ;
```

```
FLOAT_NUM     : INTEGER_NUM '.' INTEGER_NUM  ;

MUL_OP     : '*' ;
DIV_OP     : '/' ;
ADD_OP     : '+' ;
SUB_OP     : '-' ;

EQ_OP     : '==' ;
NE_OP     : '<>' ;
LT_OP     : '<' ;
LE_OP     : '<=' ;
GT_OP     : '>' ;
GE_OP     : '>=' ;

NEWLINE : '\r'? '\n' -> skip;
WS      : [ \t]+ -> skip;
```

Parse Tree for C-ish

This parse tree is included in our submission in the Decoders.zip file in 'C_Parse_Tree.png'.

Generated Code Template

- *Two data types:*

```
; int i, j
     .field private static i I
     .field private static j I

; float alpha,beta5x
     .field private static alpha F
     .field private static beta5x F
```

- *Basic arithmetic operations with operator precedence:*

```
; integer operations
; j=i+8
     getstatic    C_main/i I
     ldc    8
     iadd
     putstatic    C_main/j I
; i=-2+3*j
```

```
      ldc    2
      ineg
      ldc    3
      getstatic    C_main/j I
      imul
      iadd
      putstatic    C_main/i I

; float operations
; alpha=9.3
      ldc    9.3
      putstatic    C_main/alpha F
; beta5x=alpha
      getstatic    C_main/alpha F
      putstatic    C_main/beta5x F
; beta5x=alpha/3.7-alpha*2.88
      getstatic    C_main/alpha F
      ldc    3.7
      fdiv
      getstatic    C_main/alpha F
      ldc    2.88
      fmul
      fsub
      putstatic    C_main/beta5x F
; beta5x=8.45*(alpha+9.12)
      ldc    8.45
      getstatic    C_main/alpha F
      ldc    9.12
      fadd
      fmul
      putstatic    C_main/beta5x F
```

- *Assignment statements:*

```
; i=32
      ldc    32
      putstatic    C_main/i I

; j=i+8 ;//Assignment statement that uses an existing identifier
```

```
     getstatic    C_main/i I
     ldc    8
     iadd
     putstatic    C_main/j I
```

- *Conditional control statement:*

```
; if (i<>70) {//code} ;//<> is != in C language
     getstatic    C_main/i I
     ldc    70
     if_icmpne    IF4
     iconst_0
     goto    IF5
IF4:
     iconst_1
IF5:
     ;more code
```

- *Looping control statement:*

```
; do{i=i+1;} while(i<125)
DO_WHILE0:
; i=i+1
     getstatic    C_main/i I
     ldc    1
     iadd
     putstatic    C_main/i I
; while(i<125) & do code
     getstatic    C_main/i I
     ldc    125
     if_icmplt    DO_WHILE1
     iconst_1
     goto    DO_WHILE2
DO_WHILE1:
     iconst_0
DO_WHILE2:
     ifne    DO_WHILE3
     goto    DO_WHILE0
DO_WHILE3:
     ; more code
```

Decoders: Ajay, Bien, Sarah
CMPE-152
5/9/19

- *Function with parameters passed by value:*

```
;int add (int x, int y) {
;     x = x + y;
;     return x;
;}
.method public static add(II)I
      iload_0
      iload_1
      iadd
      istore_0
      iload_0
      ireturn
.limit stack 8
.limit locals 8
.end method
```

- *Function call:*

```
; add(i,j);
      getstatic    C_main/i I
      getstatic    C_main/j I
      invokestatic C_main/add(II)I
      putstatic    C_main/j I
```

- *Print function:*

```
; printf("j-->%d",j)
      getstatic    C_main/j I
      getstatic    java/lang/System/out Ljava/io/PrintStream;
      ldc    "j-->%d\n"
      iconst_1
      anewarray    java/lang/Object
      dup
      iconst_0
      getstatic    C_main/j I
      invokestatic    java/lang/Integer.valueOf(I)Ljava/lang/Integer;
      aastore
```

Decoders: Ajay, Bien, Sarah
CMPE-152
5/9/19

```
      invokevirtual
java/io/PrintStream.printf(Ljava/lang/String;[Ljava/lang/Object;)Ljava
/io/PrintStream;
      pop
      pop
```