COM S 440/540 Project part 5

Code generation: control flow

1 Requirements for part 5

When executed with a mode of 5, your compiler should read the specified input file, and check it for correctness (including type checking) as done in part 3. If there are no errors, then your compiler should output an equivalent program in our target language (still Java assembly). For this part of the project, your compiler must generate correct code for expressions (from part 4), and for (possibly nested) branching statements and loops. As usual, error messages should be written to standard error, and your compiler may make a "best effort" to continue processing the input file, or exit. Specifically, your part 4 solution should be modified to include the following.

- Type checking of conditions (expressions) within if statements, while loops, do-while loops, and for loops: the condition expression should be a numeric type (char, int, or float).
- Type checking of update expressions in for loops.
- Code generation for if statements (with and without else), while loops, do-while loops, and for loops.
- Code generation for break and continue statements; error messages when they are not within a loop.

2 Checking your generated code

The test scripts from Part 4 have been updated and can be used to test your compilers:

RunTest.sh will run your compiler with mode `-5` on the C source code, assemble the output into a `.class` file, and run the `.class` file on a JVM.

DotTest.sh will run your compiler on the C source code, and check (only) the `.class`, `.super`, `.method`, `.code`, and `.end` portions of the generated assembly.

AsmTest.sh will run your compiler on the C source code, and compare your generated assembly against carefuly crafted specifications.

As always, students are encouraged to ensure that their compilers work well with the scripts.

3 Grading

For all students: implement as many or as few features listed below as you wish, but keep in mind that some features will make testing your code *much* easier, and a deficit of points will impact your overall grade. Excess points will count as extra credit.

For code generation "without short circuiting", your compilers will be tested using integer variables for the condition. The basic tests for each construct are as follows, where `x` is an integer variable.

```
if (x) { /* statements */ }

if (x) { /* statements */ } else { /* statements */ }
```

```
while (x) { /* statements */ }

do { /* statements */ } while (x);

for (/* initialize */; x; /* update */) { /* statements */ }
```

More advanced tests will use an integer expression, or a single comparison as the condition.

Code generation "with short circuiting" will be tested with a variety of conditions (comparisons and numeric values) connected with operators `&&`, `||`, and `!`.

Points	Description
15	**Documentation**
3	`README.txt` How to build the compiler and documentation. Updated to show which part 5 features are implemented.
12	`developers.pdf` New section for part 5, that explains the purpose of each source file, the main data structures used (or how they were updated), and gives a high-level overview of how the target code is generated.
7	**Ease of grading** How easy was it for the graders to build your compiler and documentation? Does your compiler work with the grading scripts?
8	**Still works in modes 0 through 4**
15	**Expressions and function calls** This includes the most basic functionality from the previous part of the project, namely function calls and assignments to variables, that will be necessary to test this part of the project.
5	**Type checking** Check that any expressions used as conditions in if statments and while, do-while, or for loops have a numeric type (char, or int, or float). Generate an appropriate error message otherwise.
5	**Error checking `break` and `continue`** Generate an appropriate error message for `break` and `continue` statements outside of a loop.
55	**Without short circuiting**
5	if–then
5	if–then–else
5	while
5	do–while
8	for
5	ternary operator `?:`
5	`break` (requires a working loop)
5	`continue` (requires a working loop)

12	comparisons: `==`, `!=`, `>`, `>=`, `<`, `<=`
30	**With short circuiting**
5	and, or, not
5	comparisons: `==`, `!=`, `>`, `>=`, `<`, `<=`
5	Boolean assignments
5	if–then, if–then–else, ternary operator
5	while, do–while
5	for
100	**Total for students in 440 (max points is 120)**
120	**Total for students in 540 (max points is 140)**

4 Submission

Part	Penalty applied
Part 0	50% off
Part 1	40% off
Part 2	30% off
Part 3	20% off
Part 4	10% off

Table 2: Penalty applied when re-grading

Be sure to commit your source code and documentation to your git repository, and to upload (push) those commits to the server so that we may grade them. In Canvas, indicate which parts you would like us to re-grade for reduced credit (see Table 2 for penalty information). Otherwise, we will grade only part 5.

5 Examples

5.1 Input: `hello.c`

```
1
2  void prints(const char s[])
3  {
4      int i;
5      i = 0;
6      while (s[i]!=0) {
7          putchar((int)s[i++]);
8      }
9  }
10
11 int main()
12 {
13     prints("Hello, world!\n");
14     return 0;
15 }
```

5.2 Output: `hello.j`

```
.class public hello
.super java/lang/Object


.method public static prints : ([C)V
    .code stack 2 locals 2
        ; expression statement at INPUTS/hello.c line 5
        iconst_0
        istore_1 ; i
        ; begin while loop at INPUTS/hello.c line 6
        goto L1
    L2:
        ; expression statement at INPUTS/hello.c line 7
        aload_0 ; s
        iload_1 ; i
        iinc 1 1
        caload
        invokestatic Method lib440 putchar (I)I
        pop
    L1:
        ; while condition at INPUTS/hello.c line 6
        aload_0 ; s
        iload_1 ; i
        caload
        iconst_0
        if_icmpne L2
        ; end while loop at INPUTS/hello.c line 6
        ; implicit return at INPUTS/hello.c line 9
        return
    .end code
.end method


.method public static main : ()I
    .code stack 1 locals 0
        ; expression statement at INPUTS/hello.c line 13
        ldc "Hello, world!\n"
        invokestatic Method lib440 java2c (Ljava/lang/String;)[C
        invokestatic Method hello prints ([C)V
        ; return at INPUTS/hello.c line 14
        iconst_0
        ireturn
        ; implicit return at INPUTS/hello.c line 15
        ;DEAD   return
    .end code
.end method


.method <init> : ()V
    .code stack 1 locals 1
        aload_0
        invokespecial Method java/lang/Object <init> ()V
        return
```

```
    .end code
.end method

.method public static main : ([Ljava/lang/String;)V
    .code stack 1 locals 1
        invokestatic Method hello main ()I
        invokestatic Method java/lang/System exit (I)V
        return
    .end code
.end method
```

5.3 Input: `fib.c`

```
/*
  Computes and prints the first 40 Fibonacci numbers.
*/

int main()
{
    int i, f1, f2, f3;
    f1 = 0;
    f2 = 1;
    putint(0);
    putchar(32);
    putchar(58);
    putchar(32);
    putint(0);
    putchar(10);
    i=40;
    while (i) {
        putint(40 - --i);
        putchar(32);
        putchar(58);
        putchar(32);
        putint(f2);
        putchar(10);
        f3 = f1 + f2;
        f1 = f2;
        f2 = f3;
    }
    return 0;
}
```

5.4 Output: `fib.j`

```
.class public fib
.super java/lang/Object


.method public static main : ()I
    .code stack 2 locals 4
        ; expression statement at INPUTS/fib.c line 8
        iconst_0
        istore_1 ; f1
```

```
10          ; expression statement at INPUTS/fib.c line 9
11          iconst_1
12          istore_2 ; f2
13          ; expression statement at INPUTS/fib.c line 10
14          iconst_0
15          invokestatic Method lib440 putint (I)V
16          ; expression statement at INPUTS/fib.c line 11
17          bipush 32
18          invokestatic Method lib440 putchar (I)I
19          pop
20          ; expression statement at INPUTS/fib.c line 12
21          bipush 58
22          invokestatic Method lib440 putchar (I)I
23          pop
24          ; expression statement at INPUTS/fib.c line 13
25          bipush 32
26          invokestatic Method lib440 putchar (I)I
27          pop
28          ; expression statement at INPUTS/fib.c line 14
29          iconst_0
30          invokestatic Method lib440 putint (I)V
31          ; expression statement at INPUTS/fib.c line 15
32          bipush 10
33          invokestatic Method lib440 putchar (I)I
34          pop
35          ; expression statement at INPUTS/fib.c line 16
36          bipush 40
37          istore_0 ; i
38          ; begin while loop at INPUTS/fib.c line 17
39          goto L1
40      L2:
41          ; expression statement at INPUTS/fib.c line 18
42          bipush 40
43          iinc 0 -1
44          iload_0 ; i
45          isub
46          invokestatic Method lib440 putint (I)V
47          ; expression statement at INPUTS/fib.c line 19
48          bipush 32
49          invokestatic Method lib440 putchar (I)I
50          pop
51          ; expression statement at INPUTS/fib.c line 20
52          bipush 58
53          invokestatic Method lib440 putchar (I)I
54          pop
55          ; expression statement at INPUTS/fib.c line 21
56          bipush 32
57          invokestatic Method lib440 putchar (I)I
58          pop
59          ; expression statement at INPUTS/fib.c line 22
60          iload_2 ; f2
61          invokestatic Method lib440 putint (I)V
62          ; expression statement at INPUTS/fib.c line 23
```

```
63          bipush 10
64          invokestatic Method lib440 putchar (I)I
65          pop
66          ; expression statement at INPUTS/fib.c line 24
67          iload_1 ; f1
68          iload_2 ; f2
69          iadd
70          istore_3 ; f3
71          ; expression statement at INPUTS/fib.c line 25
72          iload_2 ; f2
73          istore_1 ; f1
74          ; expression statement at INPUTS/fib.c line 26
75          iload_3 ; f3
76          istore_2 ; f2
77      L1:
78          ; while condition at INPUTS/fib.c line 17
79          iload_0 ; i
80          ifne L2
81          ; end while loop at INPUTS/fib.c line 17
82          ; return at INPUTS/fib.c line 28
83          iconst_0
84          ireturn
85          ; implicit return at INPUTS/fib.c line 29
86          ;DEAD   return
87      .end code
88  .end method
89
90
91  .method <init> : ()V
92      .code stack 1 locals 1
93          aload_0
94          invokespecial Method java/lang/Object <init> ()V
95          return
96      .end code
97  .end method
98
99  .method public static main : ([Ljava/lang/String;)V
100     .code stack 1 locals 1
101         invokestatic Method fib main ()I
102         invokestatic Method java/lang/System exit (I)V
103         return
104     .end code
105 .end method
```

5.5 Input: `short.c`

```
1
2  /*
3   * Simple short-circuiting tests
4   */
5
6  void and(int x)
7  {
8      if ((putchar(65) < x) && (putchar(75) < x) && (putchar(85) < x))
```

```
 9      {
10          putchar(33);
11      } else {
12          putchar(46);
13      }
14      putchar(10);
15  }
16
17  void or(int x)
18  {
19      if ((putchar(65) > x) || (putchar(75) > x) || (putchar(85) > x))
20      {
21          putchar(33);
22      } else {
23          putchar(46);
24      }
25      putchar(10);
26  }
27
28  int main()
29  {
30      and(60);
31      and(70);
32      and(80);
33      and(90);
34
35      or(60);
36      or(70);
37      or(80);
38      or(90);
39      return 0;
40  }
```

5.6 Output: short.j

```
1   .class public short
2   .super java/lang/Object
3
4
5   .method public static and : (I)V
6       .code stack 2 locals 1
7           ; if statement at INPUTS/short.c line 8
8           bipush 65
9           invokestatic Method lib440 putchar (I)I
10          iload_0 ; x
11          if_icmpge L1
12          bipush 75
13          invokestatic Method lib440 putchar (I)I
14          iload_0 ; x
15          if_icmpge L1
16          bipush 85
17          invokestatic Method lib440 putchar (I)I
18          iload_0 ; x
19          if_icmpge L1
```

```
20          ; expression statement at INPUTS/short.c line 10
21          bipush 33
22          invokestatic Method lib440 putchar (I)I
23          pop
24          goto L2
25      L1:
26          ; expression statement at INPUTS/short.c line 12
27          bipush 46
28          invokestatic Method lib440 putchar (I)I
29          pop
30      L2:
31          ; expression statement at INPUTS/short.c line 14
32          bipush 10
33          invokestatic Method lib440 putchar (I)I
34          pop
35          ; implicit return at INPUTS/short.c line 15
36          return
37      .end code
38  .end method
39
40
41  .method public static or : (I)V
42      .code stack 2 locals 1
43          ; if statement at INPUTS/short.c line 19
44          bipush 65
45          invokestatic Method lib440 putchar (I)I
46          iload_0 ; x
47          if_icmpgt L1
48          bipush 75
49          invokestatic Method lib440 putchar (I)I
50          iload_0 ; x
51          if_icmpgt L1
52          bipush 85
53          invokestatic Method lib440 putchar (I)I
54          iload_0 ; x
55          if_icmple L3
56      L1:
57          ; expression statement at INPUTS/short.c line 21
58          bipush 33
59          invokestatic Method lib440 putchar (I)I
60          pop
61          goto L2
62      L3:
63          ; expression statement at INPUTS/short.c line 23
64          bipush 46
65          invokestatic Method lib440 putchar (I)I
66          pop
67      L2:
68          ; expression statement at INPUTS/short.c line 25
69          bipush 10
70          invokestatic Method lib440 putchar (I)I
71          pop
72          ; implicit return at INPUTS/short.c line 26
```

```
73          return
74      .end code
75  .end method
76
77
78  .method public static main : ()I
79      .code stack 1 locals 0
80          ; expression statement at INPUTS/short.c line 30
81          bipush 60
82          invokestatic Method short and (I)V
83          ; expression statement at INPUTS/short.c line 31
84          bipush 70
85          invokestatic Method short and (I)V
86          ; expression statement at INPUTS/short.c line 32
87          bipush 80
88          invokestatic Method short and (I)V
89          ; expression statement at INPUTS/short.c line 33
90          bipush 90
91          invokestatic Method short and (I)V
92          ; expression statement at INPUTS/short.c line 35
93          bipush 60
94          invokestatic Method short or (I)V
95          ; expression statement at INPUTS/short.c line 36
96          bipush 70
97          invokestatic Method short or (I)V
98          ; expression statement at INPUTS/short.c line 37
99          bipush 80
100         invokestatic Method short or (I)V
101         ; expression statement at INPUTS/short.c line 38
102         bipush 90
103         invokestatic Method short or (I)V
104         ; return at INPUTS/short.c line 39
105         iconst_0
106         ireturn
107         ; implicit return at INPUTS/short.c line 40
108         ;DEAD   return
109     .end code
110 .end method
111
112
113 .method <init> : ()V
114     .code stack 1 locals 1
115         aload_0
116         invokespecial Method java/lang/Object <init> ()V
117         return
118     .end code
119 .end method
120
121 .method public static main : ([Ljava/lang/String;)V
122     .code stack 1 locals 1
123         invokestatic Method short main ()I
124         invokestatic Method java/lang/System exit (I)V
125         return
```

```
126     .end code
127 .end method
```

5.7 Input: break1.c

```
1
2  int main()
3  {
4      int a;
5      for (a=48 ; ; )
6      {
7          a = a + 1;
8          if (a>55) break;
9          putchar(a);
10     }
11     putchar(10);
12     return 0;
13 }
```

5.8 Output: break1.j

```
1  .class public break1
2  .super java/lang/Object
3
4
5  .method public static main : ()I
6      .code stack 2 locals 1
7          ; begin for loop at INPUTS/break1.c line 5
8          ; with break label
9          ; for initialization at INPUTS/break1.c line 5
10         bipush 48
11         istore_0 ; a
12     L2:
13         ; expression statement at INPUTS/break1.c line 7
14         iload_0 ; a
15         iconst_1
16         iadd
17         istore_0 ; a
18         ; if statement at INPUTS/break1.c line 8
19         iload_0 ; a
20         bipush 55
21         if_icmple L1
22         ; break at INPUTS/break1.c line 8
23         goto L3
24     L1:
25         ; expression statement at INPUTS/break1.c line 9
26         iload_0 ; a
27         invokestatic Method lib440 putchar (I)I
28         pop
29         ; empty for condition
30         goto L2
31         ; end for loop at INPUTS/break1.c line 5
32     L3:
33         ; expression statement at INPUTS/break1.c line 11
```

```
34          bipush 10
35          invokestatic Method lib440 putchar (I)I
36          pop
37          ; return at INPUTS/break1.c line 12
38          iconst_0
39          ireturn
40          ; implicit return at INPUTS/break1.c line 13
41          ;DEAD   return
42      .end code
43  .end method
44
45
46  .method <init> : ()V
47      .code stack 1 locals 1
48          aload_0
49          invokespecial Method java/lang/Object <init> ()V
50          return
51      .end code
52  .end method
53
54  .method public static main : ([Ljava/lang/String;)V
55      .code stack 1 locals 1
56          invokestatic Method break1 main ()I
57          invokestatic Method java/lang/System exit (I)V
58          return
59      .end code
60  .end method
```

5.9 Input: break2.c

```
1  int main()
2  {
3      int a;
4      for (a=48 ; a<=55; a = a+1)
5      {
6          putchar(a);
7      }
8      putchar(10);
9      break; // should be an error
10     return 0;
11 }
```

5.10 Error(s) for break2.c

```
Code generation error in file break2.c line 9
  break not inside a loop
```