Nick Capurso

COSC 40603 – Dr. Rinewalt

Compiler Project Readme

Included in this Document:

1. Descriptions of Included Java Files:

- Grammar.jj – The main grammar file from which generates the Parser class and other JavaCC-derived files.
- CodeGenerator.java – The class that handles code generation after quads are generated by the parser.
- Quad.java – Custom class representing a quad.
- SymbolTableEntry.java – Custom class representing a symbol table entry.

2. What Works

- All quads successfully generated.
- Code generation successfully generates all machine instructions from quads.
- Added unlabeled EXIT statement functionality.
- Added dynamic memory allocation for temporary variables (in the case they need to be written to memory).
- Generates immediate/quick versions of machine instructions when using a constant as an operand.
- Error handling.

3. What Doesn’t Work

- Nothing, hopefully.
- Sometimes, EOF (Ctrl-D) doesn’t work in the Eclipse plugin. Thus, I added a keyword “$QUIT” that will quit the program/parser.

4. Constraints

- Variable identifiers and loop labels are case-insensitive as defined by the instructions, “Upper and lower case letters are equivalent in all tokens except strings.”
- Reserved words **MUST** be typed in either all capital letters or all lowercase letters (i.e. “ELSIF” or “elsif”).

5. Test Program (Capitalized Reserved Words)

```
a := 2;
b:= -a;
c:= a+2*b-5;
d:=100/c;
IF d>=b THEN a:=3; ELSE d:=10;END IF;
IF a&b THEN PUT (a,b); END IF;
WHILE a|b&^c LOOP a:=a-1; END LOOP;
IF a<b &c>d | a/=d & b=c<=d THEN a:=SQRT(b*ABS(c)); ELSIF a>0 THEN b:=+a; ELSIF
a<5 THEN GET(a); END IF;
WHILE a>0 LOOP
    b:=5;
    x:WHILE c<-7 LOOP
            IF ^d THEN EXIT x;END IF;
            y:WHILE b&c LOOP
                    PUT("hello world");
                    IF a THEN EXIT x; ELSE EXIT y; END IF;
                    WHILE a LOOP
                            IF b THEN EXIT x; ELSE EXIT y; END IF;
                    END LOOP;
                    b:=c|9;
                    a:=a<b;
                    IF a THEN EXIT x; ELSE EXIT y; END IF;
            END LOOP;
            IF b THEN EXIT x; END IF;
            a:=b-c;
    END LOOP;
END LOOP;
```

6. Test Program Output

```
;--------------------Program Start------------------
       ORG           $1000
1000   MOVEQ.L       #2,D1
1002   MOVE.L        D1,D2
1004   NEG           D2
1006   MOVEQ.L       #2,D3
1008   MUL.L         D2,D3
100a   MOVE.L        D1,D4
100c   ADD.L         D3,D4
100e   SUBI.L        #5,D4
1010   MOVEQ.L       #100,D3
1012   DIV.L         D4,D3
1014   MOVE.L        D3,D5
1016   CMP.L         D2,D5
1018   BGE           101e
101a   CLR.L         D5
101c   BRA           1020
101e   MOVEQ.L       #1,D5
1020   MOVE.L        D1,A
1022   MOVE.L        D2,B
1024   MOVE.L        D3,D
1026   MOVE.L        D4,C
1028   TST           D5
102a   BEQ           1032
102c   MOVEQ.L       #3,D1
102e   MOVE.L        D1,A
1030   JMP           1036
1032   MOVEQ.L       #10,D1
1034   MOVE.L        D1,D
1036   MOVE.L        A,D1
1038   AND.L         B,D1
103a   TST           D1
103c   BEQ           1046
103e   MOVE.L        A,D0
1040   TRAP          #2
1042   MOVE.L        B,D0
1044   TRAP          #2
1046   MOVE.L        C,D1
1048   NOT           D1
104a   MOVE.L        B,D2
104c   AND.L         D1,D2
104e   MOVE.L        A,D1
1050   OR.L          D2,D1
1052   TST           D1
1054   BEQ           105e
1056   MOVE.L        A,D1
1058   SUBI.L        #1,D1
105a   MOVE.L        D1,A
105c   JMP           1046
105e   MOVE.L        A,D1
1060   CMP.L         B,D1
1062   BLT           1068
```

```
1064  CLR.L      D1
1066  BRA        106a
1068  MOVEQ.L    #1,D1
106a  MOVE.L     C,D2
106c  CMP.L      D,D2
106e  BGT        1074
1070  CLR.L      D2
1072  BRA        1076
1074  MOVEQ.L    #1,D2
1076  AND.L      D2,D1
1078  MOVE.L     A,D2
107a  CMP.L      D,D2
107c  BNE        1082
107e  CLR.L      D2
1080  BRA        1084
1082  MOVEQ.L    #1,D2
1084  MOVE.L     B,D3
1086  CMP.L      C,D3
1088  BEQ        108e
108a  CLR.L      D3
108c  BRA        1090
108e  MOVEQ.L    #1,D3
1090  CMP.L      D,D3
1092  BLE        1098
1094  CLR.L      D3
1096  BRA        109a
1098  MOVEQ.L    #1,D3
109a  AND.L      D3,D2
109c  OR.L       D2,D1
109e  TST        D1
10a0  BEQ        10b0
10a2  MOVE.L     C,D1
10a4  ABS        D1
10a6  MOVE.L     B,D2
10a8  MUL.L      D1,D2
10aa  SQRT       D2
10ac  MOVE.L     D2,A
10ae  JMP        10dc
10b0  MOVE.L     A,D1
10b2  CMPI.L     #0,D1
10b4  BGT        10ba
10b6  CLR.L      D1
10b8  BRA        10bc
10ba  MOVEQ.L    #1,D1
10bc  TST        D1
10be  BEQ        10c6
10c0  MOVE.L     $24,D1
10c2  MOVE.L     D1,B
10c4  JMP        10dc
10c6  MOVE.L     A,D1
10c8  CMPI.L     #5,D1
10ca  BLT        10d0
10cc  CLR.L      D1
10ce  BRA        10d2
10d0  MOVEQ.L    #1,D1
```

```
10d2   TST           D1
10d4   BEQ           10dc
10d6   CLR.L         D0
10d8   TRAP          #1
10da   MOVE.L        D0,A
10dc   MOVE.L        A,D1
10de   CMPI.L        #0,D1
10e0   BGT           10e6
10e2   CLR.L         D1
10e4   BRA           10e8
10e6   MOVEQ.L       #1,D1
10e8   TST           D1
10ea   BEQ           116c
10ec   MOVEQ.L       #5,D1
10ee   MOVE.L        D1,B
10f0   MOVEQ.L       #7,D1
10f2   NEG           D1
10f4   MOVE.L        C,D2
10f6   CMP.L         D1,D2
10f8   BLT           10fe
10fa   CLR.L         D2
10fc   BRA           1100
10fe   MOVEQ.L       #1,D2
1100   TST           D2
1102   BEQ           116a
1104   MOVE.L        D,D1
1106   NOT           D1
1108   TST           D1
110a   BEQ           110e
110c   JMP           116a
110e   MOVE.L        B,D1
1110   AND.L         C,D1
1112   TST           D1
1114   BEQ           115a
1116   MOVEA.L       helloworld,A0
1118   TRAP          #3
111a   MOVE.L        A,D1
111c   TST           D1
111e   BEQ           1124
1120   JMP           116a
1122   JMP           1126
1124   JMP           115a
1126   MOVE.L        A,D1
1128   TST           D1
112a   BEQ           113a
112c   MOVE.L        B,D1
112e   TST           D1
1130   BEQ           1136
1132   JMP           116a
1134   JMP           1138
1136   JMP           115a
1138   JMP           1126
113a   MOVE.L        C,D1
113c   ORI.L         #9,D1
113e   MOVE.L        A,D2
```

```
1140   CMP.L        D1,D2
1142   BLT          1148
1144   CLR.L        D2
1146   BRA          114a
1148   MOVEQ.L      #1,D2
114a   MOVE.L       D1,B
114c   MOVE.L       D2,A
114e   TST          D3
1150   BEQ          1156
1152   JMP          116a
1154   JMP          1158
1156   JMP          115a
1158   JMP          110e
115a   MOVE.L       B,D1
115c   TST          D1
115e   BEQ          1162
1160   JMP          116a
1162   MOVE.L       B,D1
1164   SUB.L        C,D1
1166   MOVE.L       D1,A
1168   JMP          10f0
116a   JMP          10dc
116c   MOVE.B       #9,D0  ;Set up halt trap
116e   TRAP         #5     ;Halt program

;----------Non-Temporary & String Storage----------
       ORG          $3000
3000   A      DC.W  0
3002   B      DC.W  0
3004   C      DC.W  0
3006   D      DC.W  0
3008   helloworld   DC.B  'hello world'
3015          DC.B  0      ;Null terminator

;-----------------Temporary Storage----------------
       ORG          $4000
```

7. Additional Features, Test Programs, and Error Handling

7.1. Unlabeled Exits

Although not required by our language, unlabeled exits are easy to implement (note: the example uses loops without a while-clause, which is also permitted by our language).

Example:

```
x: LOOP
       y := 0;
       LOOP
              y := 1;
              EXIT x;
              EXIT;
       END LOOP;
       EXIT;
END LOOP;
```

Output:

```
;-------------------Program Start------------------
       ORG           $1000
1000   MOVEQ.L       #0,D1 (Beginning of Loop X)
1002   MOVE.L        D1,Y
1004   MOVEQ.L       #1,D1 (Beginning of inner, unlabeled loop)
1006   MOVE.L        D1,Y
1008   JMP           1012 (EXIT x; – Jumps to the end of the program)
100a   JMP           100e (EXIT; - Jumps to the end of the inner loop)
100c   JMP           1004 (Inner loop jump back to beginning)
100e   JMP           1012 (EXIT; - Jumps to the end of the program)
1010   JMP           1000 (Loop x jump back to beginning)
1012   MOVE.B        #9,D0  ;Set up halt trap
1014   TRAP          #5     ;Halt program

;----------Non-Temporary & String Storage----------
       ORG           $3000
3000   Y      DC.W   0

;-----------------Temporary Storage----------------
       ORG           $4000
```

7.2. Immediate/Quick Versions of Machine Instructions

Example:

```
x := 0;
y := x - 5;
z := y | 4;
```

Output:

```
;-------------------Program Start------------------
       ORG           $1000
1000   MOVEQ.L       #0,D1 (Move quick 0 (for x := 0))
1002   MOVE.L        D1,D2
1004   SUBI.L        #5,D2 (Subtract immediate (x – 5)
1006   MOVE.L        D2,D3
1008   ORI.L         #4,D3 (OR immediate ( y | 4 )
100a   MOVE.L        D1,X
100c   MOVE.L        D2,Y  (Move all to memory, end of BB)
100e   MOVE.L        D3,Z
1010   MOVE.B        #9,D0  ;Set up halt trap
1012   TRAP          #5     ;Halt program

;----------Non-Temporary & String Storage----------
       ORG           $3000
3000   X      DC.W   0
3002   Y      DC.W   0
3004   Z      DC.W   0

;-----------------Temporary Storage----------------
       ORG           $4000
```

7.3. Dynamic Temporary Variable Memory Allocation

In the case that all data registers get filled up in the middle of a basic block, one is chosen (D1) and its contents are written to memory. If, at this point, a temporary variable is still live, then it must be allocated storage so it can be moved back into a register later.

Example (no jumps, thus there will be only one basic block):

```
a := 0; b := 0; c := 0; d :=0; //declarations
e := a<b | d >= a & ((a < b) & ((b < c) & (b & (c < d))));
```

```
(Because of the parenthesized expressions, the temporary variables keeping track of
the evaluated left-hand-side expressions of the ANDs are kept live until the right-
hand-side is finally be evaluated and the AND can be done.)
```

Output (chopped up because of length):

```
;--------------------Program Start------------------
       ORG          $1000
1000   MOVEQ.L      #0,D1
1002   MOVEQ.L      #0,D2
1004   MOVEQ.L      #0,D3
1006   MOVEQ.L      #0,D4
1008   MOVE.L       D1,D5
100a   CMP.L        D2,D5
100c   BLT          1012   (a < b)
100e   CLR.L        D5
1010   BRA          1014
1012   MOVEQ.L      #1,D5
*** Lots of Comparisons ***
103a   MOVE.L       D1,$3 (Registers fill up, D1's contents need to be moved to
                           memory, which includes a live temporary $3)
*** More Comparisons ***
1048   MOVE.L       D1,$4 (Registers fill up again, $4 has to be moved to memory)
104a   MOVE.L       D2,D1
104c   AND.L        $4,D1
104e   MOVE.L       D1,$5 (Again here, $5 moved to memory)
1050   MOVE.L       $3,D1 ($3 moved back into a register to do the AND)
1052   AND.L        $5,D1
*** Executing Until Program End ***

;----------Non-Temporary & String Storage----------
       ORG          $3000
3000   A      DC.W  0
3002   B      DC.W  0
3004   C      DC.W  0
3006   D      DC.W  0
3008   E      DC.W  0
;-----------------Temporary Storage----------------
       ORG          $4000
4000   $3     DC.W  0
4002   $4     DC.W  0 (Storage has been allocated to only temporaries $3, $4, $5)
4004   $5     DC.W  0
```

7.4. Error Throwing: Undeclared Variable

Throws a RuntimeException with a custom message if an undeclared variable is referened.

Example:

```
y := x / 100;
```

Output:

```
Exception in thread "main" java.lang.RuntimeException: Variable x not declared!
```

7.5. Error Throwing: Non-Unique Loop Label

Throws a RuntimeException with a custom message if a loop has been labeled with a label that already exists on another, encompassing loop (otherwise, an ambiguous EXIT statement could be created).

Example:

```
x: WHILE 5 > 4 LOOP
      x: WHILE 6 > 5 LOOP
```

Output (Interrupted After Writing Second Loop):

```
Exception in thread "main" java.lang.RuntimeException: Loop X already exists, may
create ambiguous EXIT statements
```