Intermediate Code Generation - Part 2

Y.N. Srikant

Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012

NPTEL Course on Principles of Compiler Design

Outline of the Lecture

- Introduction (covered in part 1)
- Different types of intermediate code (covered in part 1)
- Intermediate code generation for various constructs

Code Template for *Function* Declaration and Call

Assumtion: No nesting of functions

result foo(parameter list){ variable declarations; Statement list; }

func begin foo

/* creates activation record for foo - */

/* - space for local variables and temporaries */

code for Statement list

func end /* releases activation record and return */

x = bar(p1,p2,p3);

code for evaluation of p1, p2, p3 (result in T1, T2, T3)

/* result is supposed to be returned in T4 */

param T1; param T2; param T3; refparam T4;

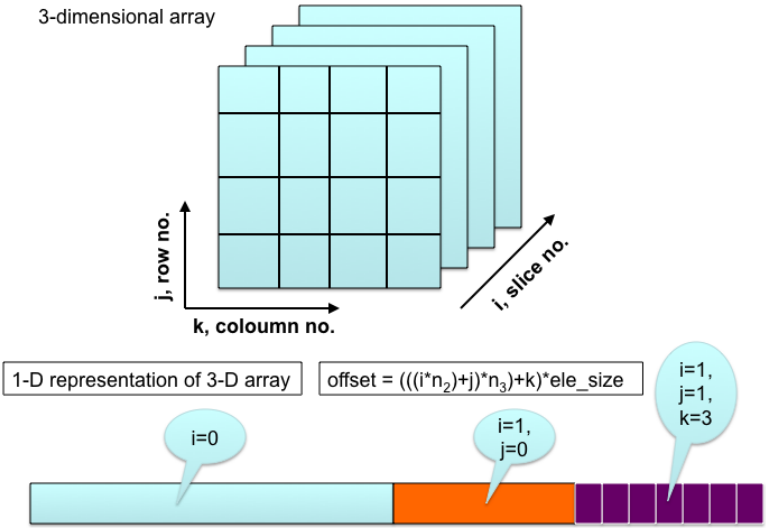
call bar, 4

/* creates appropriate access links, pushes return address */

/* and jumps to code for bar */

x = T4

1-D Representation of 3-D Array



```
int a[10][20][35], b;
b = exp1;
code for evaluation of exp1 (result in T1)
b = T1
/* Assuming the array access to be, a[i][j][k] */
/* base address = addr(a), offset = (((i*n2)+j)*n3)+k)*ele_size */
a[exp2][exp3][exp4] = exp5;

10: code for exp2 (result in T2)  || 141: T8 = T7+T6
70: code for exp3 (result in T3)  || 142: T9 = T8*intsize
105: T4 = T2*20                   || 143: T10 = addr(a)
106: T5 = T4+T3                   || 144: code for exp5 (result in T11)
107: code for exp4 (result in T6) || 186: T10[T9] = T11
140: T7 = T5*35
```

Short Circuit Evaluation for Boolean Expressions

- (exp1 && exp2): value = if ($\sim$exp1) then FALSE else exp2
  - This implies that exp2 need not be evaluated if exp1 is FALSE
- (exp1 || exp2):value = if (exp1) then TRUE else exp2
  - This implies that exp2 need not be evaluated if exp1 is TRUE
- Since boolean expressions are used mostly in conditional and loop statements, it is possible to realize perform short circuit evaluation of expressions using control flow constructs
- In such a case, there are no explicit '||' and '&&' operators in the intermediate code (as earlier), but only jumps
- Much faster, since complete expression is not evaluated
- If unevaluated expressions have side effects, then program may have non-deterministic behaviour

Control-Flow Realization of Boolean Expressions

if ((a+b < c+d) || ((e==f) && (g > h-k))) A1; else A2; A3;

```
100:         T1 = a+b
101:         T2 = c+d
103:         if T1 < T2 goto L1
104:         goto L2
105:L2:      if e==f goto L3
106:         goto L4
107:L3:      T3 = h-k
108:         if g > T3 goto L5
109:         goto L6
110:L1:L5:   code for A1
111:         goto L7
112:L4:L6:   code for A2
113:L7:      code for A3
```