CS480

Ch 8

(8.1 - 8.7 pp. 463-508)

Intermediate Code Generation

April 1, 2009

This will be on Monday's exam

Intermediate Code

- Ties front end to back end
- Can be machine/language independent

Parser –> Static Checker –> Intermediate Code Generator –> (Optimizer) –> Code Generator

- We don't really have distinct Intermediate and Machine code

Three Address Code

x = y ***op*** z

- At *most* three addresses used
  - x = y is valid 3 Address Code
  - x = -y is valid 3 Address Code
- May need to break down expressions
  - x = y + z * q;
- Our "Quads" are three address code

Types of Statements

- Assignment Statement

  x = y *binop* z; x = *unop* y

- Unconditional Jumps: goto Label
- Conditional Jumps: if (x *relop* y) goto Label
- Function Call: Foo (a,b,c)
- Function Return: return y
- Indexed Assignments: x = a[i] ; a[i] = x
- Address and Pointer Assignments: x = &a ; b = *a; *a = b

Syntax Directed Translation into 3 Address code

page 468

- Semantic Rules!
- Attributes? How? Where? When?

S -> id = E ||

E -> E_1 + E_2 ||

- What happens if E + E is deep in the parse tree?
- What if E_1 is an array? Pointer? The expression (a + b) ?

OpCode Execution

```
void intExecute(int wOpcode, /*opcode of current quad to be executed */
           int wOperand1,    /*operand1 value if necessary for opcode*/
           int wOperand2,    /*operand2 value if necessary for opcode*/
           int wOperand3)    /*operand3 value if necessary for opcode*/
{
  . . .
  switch (wOpcode)
  {
    case OP_ADD:        gStack[wOperand3] = wOperand1 + wOperand2;
                        break;
    . . .

    case OP_DEREFERENCE:  gStack[wOperand3] = gStack[wOperand1];
                          break;
```

Addressing Modes

```
int intDecode(int wMode,          /* mode of the operand */
              int wAddress)       /* address of Op */
```

```
{
  switch(wMode)
  {
    case IMMEDIATE:      return(wAddress); // 0

    case GLOBAL_LVALUE: return(wAddress); // 1

    case GLOBAL_RVALUE: return(gStack[wAddress]); // 2

    case LOCAL_LVALUE:  return(gAP + wAddress); // 3

    case LOCAL_RVALUE:  return(gStack[gAP + wAddress]);//4
  };
}
```

Example

1	int Foo(b)
2	int b;
3	{
4	output(b);
5	}
6	
7	main ()
8	{
9	int a;
10	a = 100;
11	a = a + 1;
12	Foo (a);
13	}

addr	Lvl	Insn	AM	Op1	AM	Op2	AM	Op3
0	1	27	0	0	0	0	0	0
1	2	22	0	0	0	0	0	0
2	2	20	4	-3	0	0	0	0
3	2	25	0	1	0	0	0	0
4	2	26	0	0	0	0	1	0
5	2	23	0	0	0	0	0	0
6	2	22	0	3	0	0	0	0
7	2	26	2	1	0	0	3	1
8	2	1	4	1	2	2	3	2
9	2	26	4	2	0	0	3	1
10	2	20	4	1	0	0	0	0
11	2	21	0	1	0	0	0	1
12	2	26	2	0	0	0	3	3
13	2	26	0	0	0	0	1	0
14	2	23	0	0	0	0	0	0
15	1	21	0	0	0	0	0	6
16	1	28	0	0	0	0	0	0

RTS
0
100
1

Various Statements

- Assignments

```
int *a, x[10] p, b, c;
a = &b;
*p = a;
x[b] = c;
c = b++;
```

Short-Circuit Code

```
if( x == 0 && ( y = x + 1) )
{
  //do something
}
```

Control Flow

- If
- If/else
- While
- How is **`for`** different from **`while`**?

Function Calls

```
int Foo(a, b)
int a; int *b;
{
  return a * *b;
}

...
Foo(1, &x);
...
```