CS480

Ch 8

(8.1 - 8.7 pp. 463-508)

Intermediate Code Generation

March 22, 2013

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 ||

$E \rightarrow E_1 + E_2$ ||

- 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;
b = 1;
 c = 9;
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);
...
```

SA2

- output (top down)

- invoke main()

- BU Parsing
  - stack: add (Level, Address)

  - function calls

```
foo(x);
bar; /* zero arguments */
```

Interpreter Debugger

Start the interpreter in Trace Mode.

./interpreter file.q -d

(the file name MUST be the first argument, and any second argument starts Trace Mode).

Once in Trace Mode you are prompted with a menu that lets you:

s	Dump the Stack and then step.
b	Set a breakpoint at an OpCode type (numeric)
<cr>	Single step by pressing [Enter]

Debugger

The AP and SP are marked in the stack.

Operand1-3 are post-`intDecode()`

```
./interpreter assignments.quad -d

TRACE: sp = 16, pc = 0, ap = 0
TRACE: quad = 27 0 0 0 0 0 0
TRACE: operand1 = 0, operand2 = 0, operand3 = 0
TRACE: enter [s = stackdump b = breakpoint <cr> = step]: s

_____________________________________________________________

                R U N T I M E   S T A C K
   0A    0    10     0     0     0     0     0     0     0
   0     0     0     0     0     0     0S    0     0     0
_____________________________________________________________

TRACE: sp = 16, pc = 1, ap = 0
TRACE: quad = 26 1 4 0 0 1 3
TRACE: operand1 = 4, operand2 = 0, operand3 = 3
TRACE: enter [s = stackdump b = breakpoint <cr> = step]:
```

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

main()
{
  b = 1;
  c = 9;
  a = &b;
  p = *a;
  x[b] = c;
  c = b++;
}
```

Lvl	Opcode	AM	OP1	AM	OP2	AM	OP3
1	27	0	0	0	0	0	0
1	26	1	4	0	0	1	3
2	22	0	4	0	0	0	0
2	26	1	15	0	0	3	1
2	26	4	1	0	0	1	1
2	9	2	1	0	0	3	2
2	26	4	2	0	0	1	14
2	1	2	3	2	15	3	3
2	26	2	16	0	0	4	3
2	26	2	15	0	0	3	4
2	26	4	4	0	0	1	16
2	7	0	0	0	0	1	15
2	26	0	0	0	0	1	0
2	23	0	0	0	0	0	0
1	21	0	0	0	0	0	2
1	28	0	0	0	0	0	0

RTS

17
0
0
10
...