Plan for today

Finish Tree patterns for MiniJava

Instruction selection for x86

Suggested exercises

Tree Patterns

Approach
- organize them by Tree.Exp and Tree.Stm node and for each one figure out if
  munchNodeNAME is needed
- determine which nodes correspond to code generation for MIPS

Tree.Exp nodes
- ExpCONSTR(int i)
- ExpNAME(Label n) - parent node will do any code gen
- ExpTEMP(Temp t) - code gen not needed because result already in Temp
- ExpBINOP(int binop, Exp left, Exp right)
  - code gen based on?
  - how do we get Temps for left and right?
- ExpMEM(Exp exp)
  - where can this show up in Tree.Stm?
- ExpCALL(Exp func, List<Exp> args)
  - what should happen here?

Tree.Stm nodes
- StmMOVE(Temp t, Exp rhs)
- StmMOVE(ExpMEM lhs, Exp rhs)
- StmEXP(Exp e)
- StmJUMP(Label targ)
- StmJUMP(int relop, Exp lhs, Exp rhs, Label true, Label f)
- StmLABEL(Label l)

Instruction selection for x86

Registers
- EAX, the accumulator
- EBX, the base register
- ECX, the counter register
- EDX, the data register
- • ESI, the source register
- • EDI, the destination register
- • ESP, the stack pointer register
- • EBP, the frame pointer register

Representations
- Constants prefixed with ‘$’, $3, $4, $-5, etc
- Registers prefixed with ‘%’, %eax, %esp, etc.

Some Instructions
- movl -12(%ebp), %eax
- imull -4(%ebp), %eax
- cmpl -4(%ebp), %eax
- jge .L2
- M[ebp-4] * %eax
- if ( M[ebp-4] >= %eax) goto .L2
Suggested Exercises

Compile the following three C programs

- using ‘gcc -S filename.c’ on a 32-bit machine
- read http://cs.wellesley.edu/~cs342/crash-course.pdf, pages 5-9

Answer the following questions

- The Pentium requires that the destination of a multiply must be %eax. What instruction in the assembly output for the below programs seems to suggest that is probably true?
- Identify and explain the lines of assembly code (5 lines that start at cmpl) generated for the ‘if’ statement.
- What is the return register?

```c
int main() {
    return 20;
}
```
```
int main() {
    int x, y;
    x = 2; y = 42;
    if (x > y) {
        return 7;
    } else {
        return 8;
    }
}
```
```
int main() {
    int x, y;
    x = 2; y = 42;
    return x * y;
}
```