An Example of some of the Code Generator Code

Here is some code snippits taken straight from a code generator phase of a compiler. They contain examples of backpatching as well as a couple of simple statements and the decrement operator. I removed some code from compound for overlaying the memory of lexically parallel compound statments. These are only programming suggestions.

```
// the classic backpatch using the emit functions
int backPatchJumpToHere(int addr, char *comment)
{
    int currloc;

    currloc = emitSkip(0);
    emitBackup(addr);
    emitRMAbs("LDA", PC, currloc, comment);
    emitBackup(currloc);
}

        .
        .
        .


    case WhileK:
        emitComment("WHILE");
        currloc = emitSkip(0);             // return to here to do the test
        codegen_expression(currnode->child[0]);

        emitRM("JGT", AC, 1, PC, "Jump to while part");
        emitComment("DO");

        skiploc = breakloc;              // save the old break statement return point

        breakloc = emitSkip(1);            // addr of instr that jumps to end of loop
                                           // this is also the backpatch point

        codegen_general(currnode->child[1]);

        emitRMAbs("LDA", PC, currloc, "go to beginning of loop");
        backPatchJumpToHere(breakloc, "No more loop");   // backpatch jump to end of loop

        breakloc = skiploc;              // restore for break statement

        emitComment("ENDWHILE");
        break;
        .
        .
        .

    case ReturnK:
        emitComment("RETURN");
        if (currnode->child[0]) {
            codegen_expression(currnode->child[0]);
        }

        emitRM("LDA", RT, 0, AC, "Copy result to rt register");
        emitRM("LD", AC, RETURN_OFFSET, FP, "Load return address");
        emitRM("LD", FP, OFPOFF, FP, "Adjust fp");
        emitRM("LDA", PC, 0, AC, "Return");
        break;
        .
        .
        .


    case CompoundK:
        emitComment("BEGIN compound statement");
        codegen_general(currnode->child[0]);
        codegen_general(currnode->child[1]);
        emitComment("END compound statement");
        break;
    }
        .
        .
        .


    case DEC:
        emitRM2("LD", AC, lhs->offset, offReg,
                "load lhs variable", lhs->attr.name);
        emitRM2("LDA", AC, -1, AC,
                "decrement value of", lhs->attr.name);
        emitRM2("ST", AC, lhs->offset, offReg,
                "Store variable", lhs->attr.name);
        break;
        .
        .
        .

    case OpK:
        // process lhs
        codegen_expression(currnode->child[0]);

        // process rhs if binary operator
        if (currnode->child[1]) {
            emitRM("ST", AC, toff--, FP, "Save left side");
            codegen_expression(currnode->child[1]);
            emitRM("LD", AC1, ++toff, FP, "Load left into ac1");
        }
        .
        .  [some more OpK code]
        .
        case '+':
            emitRO("ADD", AC, AC1, AC, "Op +");
            break;
```