编译课设申优文章

17231171 赵祎君

一、课程设计整体架构

1. **词法分析与语法分析**

该部分为课程设计的基础部分，按照上课讲述的递归下降分析法构造分析子程序进行分析即可，注意预留出错误处理的接口，以对应后续的错误处理程序。

文法中有一些不符合LL(1)规定的地方，可以通过改变文法的方式解决，也可以针对文法特殊处理，解决方法在后文困难点解决方案中说明。

1. **错误处理**

在错误处理阶段之前，需要先按照要求建立起符号表，在建立过程中可以捕捉b型错误，并拒绝填符号表。

在词法分析程序中，捕捉并处理a型错误。

在赋值语句、读写语句、循环语句等出现标识符的地方，捕捉并处理c型错误。

在函数值参数表的分析子程序中，捕捉并处理d和e型错误。

在条件分析子程序中，捕捉并处理f型错误。

对返回函数，查看所述函数的性质，判断返回值类型是否合理，捕捉并处理g和h型错误。

在赋值语句、因子等可能出现数组操作的地方，捕捉并处理i型错误。

在赋值语句处，通过符号表判断左值的类型。捕捉并处理j型错误。

在语法分析程序中，对于需要出现分号、右小括号、右中括号、while的地方，分别捕捉并处理k、l、m、n型错误。

在常量定义中，判断等号前后类型是否符合，捕捉并处理o型错误。

对于其他预留了错误处理接口，但未给出要求的语法、语义错误，归在自定义的z型错误中。

上述所有的错误类型中，除b型错误，都可以采用“忽略“”的方法进行局部化处理，即缺少成分的地方假设该成分存在，继续进行下一动作。

1. **代码生成**

首先借助语法分析程序生成中间代码，再通过中间代码生成mips程序。

中间代码设计为四元式的形式，但是操作符包括但不局限于传统的算术运算和逻辑运算，设置标签、跳转语句、读写操作、函数声明、调用与返回都可以作为特殊的操作符转化为四元式的形式存储进格式统一的中间代码中。

在生成中间代码的阶段，对于计算过程中所需要的中间变量，可以设置函数统一分配，但要注意命名时特殊处理，避免与原有变量名重复。组织与符号表结构相似的中间变量表，存储这些中间变量，以便后续管理。

对于push函数参数的顺序，处理技巧在后文困难点解决方案中说明。

在生成mips函数之前，先对符号表进行扫描，确定每个函数所需要的栈空间大小，并计算函数参数、函数局部变量与常量、函数中间变量在函数栈空间中的偏移。

在中间代码生成mips代码的阶段，首先查看符号表，在.data阶段为全局的常量与变量分配地址空间。接下来进入.text，需要先将栈指针下移main函数所需的空间，再加载j main指令以确保程序从main开始执行。

对于其余中间代码，结合自己对中间代码意义的定义与mips语句的对应，简单生成mips代码即可。

此阶段可以暂不考虑寄存器的分配问题，对每条需要计算的中间代码，可以先将其操作数的值从主存中取出，赋给寄存器$t0与$t1，运算结果存储在寄存器$t2中，再将$t2寄存器的值存入结果在主存中对应的位置。

对于函数参数、调用以及返回的中间代码处理，是较为困难的地方，将在后文困难点解决方案中说明。

1. **代码优化**

代码优化部分分为中间代码优化与目标代码的优化两部分。

对于中间代码优化，主要优化方式为复制传播与函数内联，并进行活跃变量分析，进行死代码删除并为寄存器分配做准备。

对于目标代码的优化，主要体现在寄存器分配与代码形式的优化上。

具体的优化方式在后文困难点解决方案中说明。

二、困难解决及技巧

1. **词法分析与语法分析**

对于变量说明和有返回值函数定义，其＜类型标识符＞＜标识符＞的部分是相同的，因此需要读到至少第三个单词才能判断是哪一语法成分。为了简化程序处理方式，这里可以设立两个临时变量将类型标识符和标识符储存下来，作为参数传递给递归下降子程序，免去了更改语法的复杂操作。

对于无返回值函数定义和主函数，也是这种处理方式。

对于语句中有返回值函数调用语句和无返回值函数调用语句的区分问题，可以在语法分析阶段建立符号表，也可以只存储函数与返回值的信息，通过查表区分。

需要注意的是，为了方便后续mips代码的生成，在词法分析中，需要对字符串中’\’、’\n’这样的字符进行转义；在表达式的分析程序中，需要向上返回表达式的类型。

1. **错误处理**

对于符号表的组织形式，由于课设语言写出的程序都为非分程序，我们可以简单的为每个函数建立一个子符号表，然后将这些符号表与全局符号表一起组织为整个程序的符号表，减少了工作量。

在处理d型错误时，可以建立函数-顺序函数参数的数组，以减轻工作量。

1. **代码生成**

在进行表达式相关的中间代码生成时，为了减少中间变量的使用，可以由调用表达式分析子程序的程序申请中间变量，传入表达式分析子程序进行赋值，而不是由因子分析程序申请中间变量，向上依次传递给项分析子程序、表达式分析子程序进行赋值。

在因子分析子程序中，对于标识符显示为常量的情况，可以直接查符号表返回常量的值，从而减少后续工作的步骤。

对于函数参数的顺序，可以采用先按代码顺序生成中间代码，再对中间代码按倒序进行重排的方法。即记录计算每个参数值的中间代码，将其按块取出，再倒序放回，同时将push语句按倒序排列在最后，紧挨funcall语句，避免影响参数中的函数调用。

将操作数的值加载进寄存器时，需要根据操作数的种类分别处理。当操作数为立即数时，需要采用li指令；当操作数为普通变量时，需要查符号表确定其在函数栈空间内的偏移，用lw指令将值取出；当操作数为数组时，需要计算数组偏移，才能取出对应的值。将结果存入主存时，也需要根据结果的种类采用不同形式的sw指令。

对于函数参数入栈操作，只需要提前将参数值存入即将分配的函数栈空间中相应位置即可。

对于函数调用语句，需要下移栈指针，存贮返回地址并跳转到函数。

对于函数声明语句，因其为函数调用后跳转到的地方，因此需要将$ra寄存器，即函数的返回地址存储在主存中，以免递归调用函数无法返回。

对于返回语句，如果返回值，则将返回值写入$v0寄存器，然后取出$ra寄存器的值取回，移回栈指针并跳转到$ra中存储的返回地址。

1. **代码优化**

**函数内联**

对于没有跳转语句、函数调用语句与局部数组的函数，可以进行函数内联，从而减少函数调用的次数，从而减少函数调用带来的必然的跳转指令与一系列对主存的操作。

在生成中间代码的过程中，如果调用的函数是可内联的，则申请中间变量赋值为函数参数的值，然后将函数原本的中间代码整体复制到调用函数的地方，并进行变量重新分配与映射，使之与原函数内容、调用函数的内容都不冲突。如果函数有返回值，则将原本的返回语句改为赋值语句。

**复制传播**

对于出现赋值情况的中间代码，如赋值语句、运算语句等，可以将被赋值的变量在后续代码中替换为赋值变量，这一对应关系将在赋值变量被重新赋值后消失。

这样就有可能将原本不是死代码的一些代码转变为死代码，在后续分析中删除，从而减少中间代码数目。

**活跃变量分析与死代码删除**

按照理论课的内容进行构建与删除即可，需要额外注意并不是中间代码中所有的操作数和结果都可以参与活跃变量分析，需要进行分辨哪些是真正的“变量”。

**寄存器分配**

对于s寄存器，在生成mips代码前进行分析，确定哪些变量需要使用哪些s寄存器，这一部分根据理论课的冲突图构建与图着色算法进行确定与分配即可。

对于t寄存器，在生成mips的过程中，如果该语句需要t寄存器辅助运算，则向t寄存器池申请t寄存器，t寄存器池根据t寄存器的使用情况，按照未被占用的寄存器、存储立即数的寄存器、存储不活跃的变量的寄存器的优先顺序分配t寄存器，如果分配了正在被活跃变量占用的寄存器，则需要将变量的值先写回主存中。

S寄存器的生命周期为一个函数，T寄存器的生命周期为一个基本块，在函数/基本块结束之前，需要将仍然活跃的、储存在s/t寄存器中的变量的值写回主存。

**代码形式的优化**

对于for循环语句和while循环语句，尽可能的用beq和bne指令代替j指令，从而减少跳转语句的数目，可以通过更改语句执行顺序的方式达成。

对于操作数为立即数的情况，尽可能的将指令转换为I型指令或可接受立即数的伪指令，从而减少寄存器的分配与写回。