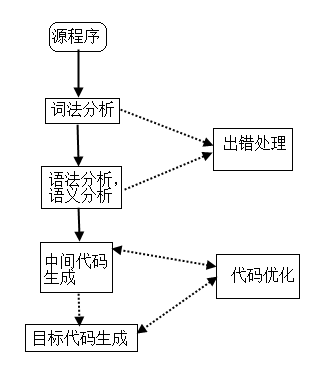
## 详细设计

### 程序结构

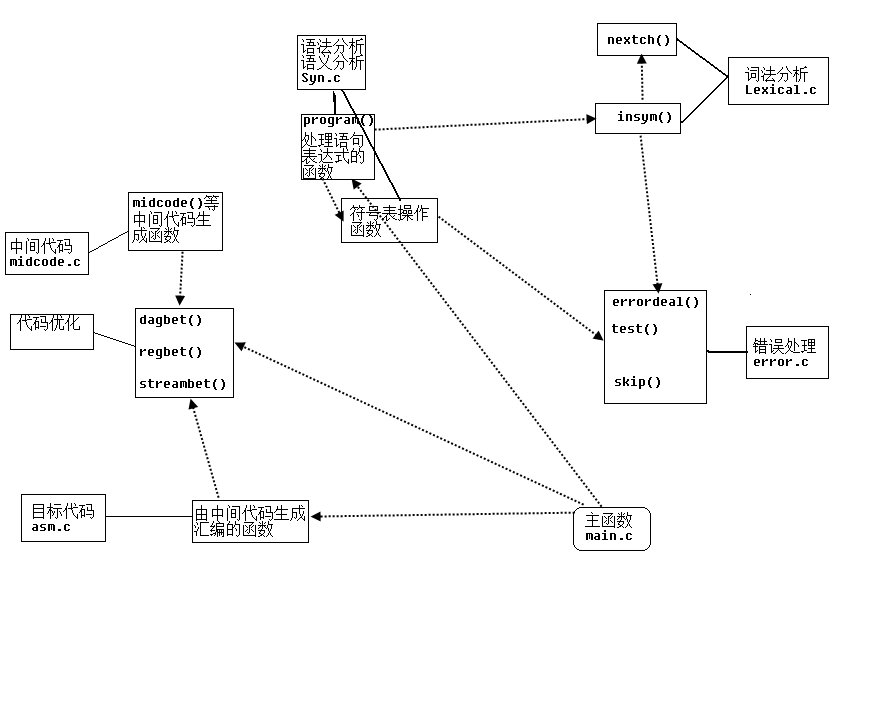
程序分为：词法分析、语法分析语义分析、生成中间代码、汇编代码生成、代码优化、出错处理



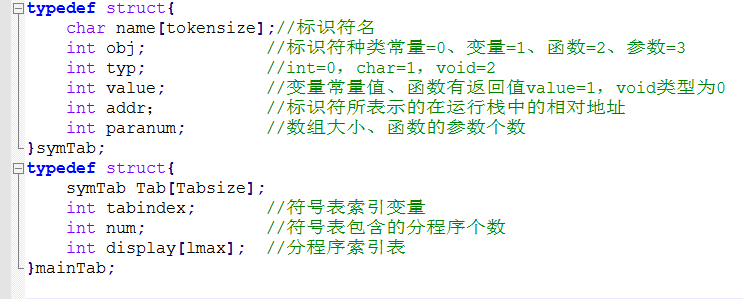
### 类/方法/函数功能

|  |  |
| --- | --- |
| 函数 | 功能 |
| **词法分析** |  |
| void nextch() | 读取文件下一个字符,并将读字符指针指向下一个字符 |
| void insym() | 读下一个单词，判断单词类型 |
| int isSpace() | 判断是否为空格 |
| int isNewline() | 判断是否为换行符 |
| int isTab() | 判断是否为制表符 |
| int isLetter() | 判断是否为字母 |
| int isDigit() | 判断是否为数字 |
| int isLetinstr() | 判断字符串中的字符是否合法 |
| int isSYm() | 判断char类型字符是否合法 |
| void setinial() | 初始化一些数组，为了后面打印词法分析结果信息 |
| void retract() | 读字符指针后退一个字符 |
| void clearToken() | 清空保存单词的token字符数组 |
| void  clearString() | 清空保存字符串的string数组 |
| void  addtoken() | 把当前读到的字符与token数组连接 |
| void  addstring() | 把当前读到的字符与string数组连接（处理字符串） |
| int transNUM() | 将数字串转化为整数 |
| **语法分析** |  |
| void program() | 处理程序 |
| void conststate() | 处理常量说明递归子程序 |
| void varstate() | 处理变量说明递归子程序 |
| void constdef() | 处理常量定义递归子程序 |
| void vardef() | 处理变量定义递归子程序 |
| void funcstate() | 处理函数说明，将函数名填入符号表 |
| void parameterlist() | 处理函数说明的形参表，将形参有关的信息登录到符号表 |
| void statement() | 处理语句列 |
| void cpstate() | 处理复合语句 |
| void whilestate() | 处理while循环语句 |
| void ifstate() | 处理if条件语句 |
| void switch() | 处理switch情况语句 |
| void casetab() | 处理情况表递归子程序 |
| void onecase() | 处理情况表子语句 |
| void defaultcase() | 处理缺省语句 |
| void call() | 处理非标准函数调用 |
| void assignment() | 处理赋值语句 |
| void scanfment() | 处理读语句 |
| void printment() | 处理写语句 |
| void retument() | 处理返回语句 |
| void expression() | 处理表达式 |
| void term() | 处理项 |
| void factor() | 处理因子 |
| void entertab() | 将标识符等信息登录到符号表 |
| void condiexpr() | 处理条件表达式 |
| **中间代码生成** |  |
| codemid() | 生成中间代码 |
| emit1(op,src1,src2,result) | 插入中间代码 |
| emit2(op,,src2,result) |  |
| emit3(op,,,result) |  |
| emit4(op,src1,src2,) |  |
| emit5(op,src1,,result) |  |
| **汇编代码生成** |  |
| void midasm() | 将中间代码转化为汇编 |
| void assasm() | 赋值语句转汇编 |
| void aassasm() | 赋值给数组元素 |
| void  assaasm() | 用数组元素赋值 |
| void  callasm() | 调用函数 |
| void  arithasm() | 处理加减乘除运算 |
| void compaasm() | 处理条件运算 |
| void  scprtasm() | 处理读写语句 |
| void  fuparasm() | 函数调用前处理参数 |
| void  retasm() | 处理返回语句 |
| void  constasm() | 处理常量定义四元式 |
| void  varasm() | 处理变量定义四元式 |
| void  varaasm() | 处理数组类型的变量声明 |
| void  jneasm() | 处理否定跳转 |
| void  jmp() | 处理无条件跳转 |
| void  parasm() | 处理形参定义 |
| void  fbegasm() | 处理函数开始四元式 |
| void  fendasm() | 处理函数结束四元式 |
| **错误处理** |  |
| void errordeal(int errcode) | errcode是出错编码,打印出错信息和位置 |
| void skip() | 跳过出错部分 |
| void test() | 测试当前符号是否合法 |
| **优化** |  |
| void dagbet() | 利用DAG图实现对基本快优化 |
| void regbet() | 寄存器分配 |
| void  streambet() | 数据流分析 |

### 调用依赖关系

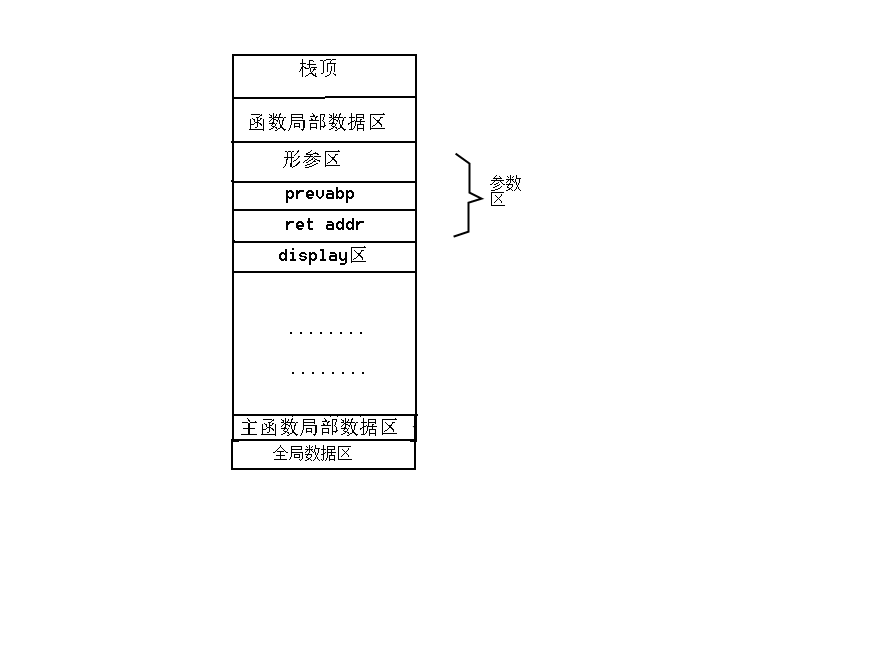


### 4.符号表管理方案



### 5.存储分配方案

采用动态存储分配：



prevabp要指向前一个活动记录的display区，所有的display区都应该指向主函数调用函数的地址；

### 6. 解释执行程序\*

没有解释执行，目标代码由Mars运行；

### 7.四元式设计\*

结构：op,src1,src2,result;

op: 操作符；

src1：操作数1;

src2：操作数2;

result：结果;

尽量按照这种结构来设计：

|  |  |
| --- | --- |
| 四元式 | 含义 |
| =，10，，a | a=10; |
| []=,k,i,a | a[i]=k; |
| const,int,5,a | const int a=5; |
| const,char,’c’,b | const int b=’c’; |
| char,,,a | char a; |
| int,,10,a | int a[10]; |
| char,,5,c | char c[5]; |
| set,,5,a | a是函数的实参，set a=5; |
| set,,’c’,b | set b=’c’; |
| call,fu1,,a | a=fu1() |
| call,fu1,, | fu1() |
| lab:,,,lname | 定义名为lname的lable |
| =,a,k,b | b=a[k]; |
| func,int,,fu1 | 声明一个int函数fu1 |
| func,void,,fu1 | 声明一个void函数fu1 |
| par,int,,a | 定义形参int a |
| par,char,,b | 定义形参char b |
| endf,,,fu1 | 函数定义结束 |
| >=,a,b, | a>=b，把结果压入栈顶 |
| !=,a,b, |  |
| ==,a,b, |  |
| <=,a,b, |  |
| >,a,b, |  |
| <,a,b, |  |
| jne,,,lable | 如果栈顶不等于0则跳转到lable |
| jmp,,,lable | 跳转到lable |
| ret,,,a | return a; |
| ret,,, | return; |
| scaf,,a, | scanf(a); |
| prtf,,a, | printf(a); |
| prtf,a,,strsym | printf(“a”); |
| prtf,a,b,strsym | printf(“a”,b); |

对于表达式，生成四元式就简单的多：op是运算符号，表达的意思：result=src1 op src2

表达式就是由多个四元式构成

### 目标代码生成方案\*

需要的汇编指令：

|  |  |  |
| --- | --- | --- |
| 指令名 | 格式 | 功能 |
| 系统调用 | syscall | 读写都需要系统调用 |
| 取立即数 | li,$t1,5 |  |
| move | move $s1,$zero | s1=0 |
| jal | jal Fun1 | 调用函数 |
| jr | jr $ra | 返回 |
| 加 | add $s1,$s2,$s3 | s1=s2+s3 |
| 减 | sub $s1,$s2,$s3 | s1=s2-s3 |
| 乘 | mult $s1,$s2 | hi,lo=s1\*s2 |
| 除 | div $s1,$s2 | hi=s1 mod s2,lo=s1/s2 |
| 加立即数 | addi $s1,$0,3 |  |
| 减立即数 | subi $s1,$0,3 |  |
| j | j lable | 无条件跳转 |
| beq | beq $s1,$s2,lable | 相等时跳转 |
| bne | bne $s1,$s2,lable | 不等时跳转 |
| bgez | bgez $s1,lable | 大于等于0时跳转 |
| bgtz | bgtz $s1,lable | >0 |
| blez | blez $s1,lable | <=0 |
| bltz | bltz $s1,lable | <0 |
| 取字 | lw $t0,0($t1) | $t0=[$t1+0] |
| 存字 | sw $t0,0($t1) | [$t1+0]=$t0 |
| 取地址 | la $t0,array | 将数组array首地址存入t0 |
| mflo | mflo $t0 | $t0=[lo] |
| mfhi | mfhi $t0 | $t0=[hi] |

### 利用一系列函数将中间代码转化为汇编代码；

### 优化方案\*

1.基本块内部的公共子表达式删除（DAG图）：

通过构建DAG图消除局部公共子表达式，要注意表达式中出现数组，函数调用情况(该文法不涉及指针)

采用从DAG导出中间代码的启发式算法

2.全局寄存器分配（引用计数或着色算法）；

运用计数算法扫描函数，查找变量使用的频率，按从大到小排列，分配寄存器

3.数据流分析（通过活跃变量分析，或利用定义-使用链建网等方法建立冲突图）；

### 10.出错处理

错误信息：

|  |  |  |
| --- | --- | --- |
| 错误名 | 错误编码 | 错误信息 |
| undef id | 0 | 没有定义过的标识符 |
| undef func | 1 | 没有定义过函数 |
| undef array | 2 | 没有定义过数组 |
| multi def | 3 | 定义冗余 |
| wrongdef id | 4 | 标识符错误定义 |
| charsize wrong | 5 | 字符长度不为1 |
| strsize wrong | 6 | 字符串长度过长 |
| can’t distinguish char | 7 | 不能识别的字符 |
| defect ] | 8 | 没有匹配到’]’ |
| defect } | 9 | 没有匹配到’}’ |
| defect ) | 10 | 没有匹配到’)’ |
| defect [ | 11 | 没有’[‘ |
| defect { | 12 | 没有‘{‘ |
| defect ( | 13 | 没有‘(‘ |
| const nonval | 14 | 常量定义没有赋值 |
| var assigval | 15 | 变量定义时赋值 |
| defect returwords | 16 | 缺失return语句 |
| defect ; | 17 | 缺失’;’ |
| scanf wrong | 18 | 读语句错误（读的值数目过多） |
| printf wrong | 19 | 写语句错误 |
| fileopenfailed | 20 | 文件打开失败 |
| rightofeqlerror | 21 | 等号右边产生错误，例如将整数值复制给字符类型标识符 |
| defectmain | 22 | 缺失main函数 |
| expresserror | 23 | 表达式错误 |
| paranummatchError | 24 | 形参与实参个数不匹配 |
| paratypeerror | 25 | 形参与实参类型不匹配 |
| idconflict | 26 | 标识符名称冲突 |
| doubquoerror | 27 | 缺失’ ” ’ |
| quotaerror | 28 | 缺失’ ’ ’ |

错误处理方案：

|  |  |
| --- | --- |
| 处理方式 | 说明 |
| 出错时立即调用error() | 直接报错，打印错误信息和位置； |
| 调用skip() | 跳过错误部分，继续编译，直到编译结束； |
| 调用Quit() | 退出程序执行；例如无法打开文件 |
| 报错但帮助修改 | 例如while后有’(’，如果缺失可以帮助补齐 |