Init.c

```
struct entry keywords[] = {

"div", DIV,

"mod", MOD,

"if",IF,

"while",WHILE,

"do",DO,

"begin",BEGIN,

"end",END,

"then",THEN,

"function",FUNCTION,

"main",MAIN,

"else",ELSE,

0, 0

};
```

Global:

```
#define ELSE 269
```

Parse:

```
#ifndef PARSEPHASE4_H_

#define PARSEPHASE4_H_

#include "emitterPHASE4.h"

#include "lexerPHASE4.h"

#include "errorPHASE4.H"

#include "globalPHASE4.h"

int tok;

void parse(){

 lookahead = lexan();

 Dec();

 Mainfun();

 match(DONE);
```

```
}

void Dec() {

        while(lookahead==FUNCTION){

        funcdec();

        match(';');}

}

void funcdec(){


match(FUNCTION);fprintf(output,"%s:\n",symtable[tokenval].lexptr); match(ID); match('(');
match(')');

match(BEGIN); CS(); match(END); fprintf(output,"ret\n");


}


void Mainfun(){

match(MAIN); fprintf(output,"main:\n");  match('('); match(')');


match(BEGIN); CS(); match(END); fprintf(output,"exit\n");

}




void stmt(){

 int t;

 switch(lookahead){

 case ID:

     tok=tokenval;

        match(ID);

        rest();


 break;
 case IF:
 match(IF);
 match('('); expr(); match(')');
 fprintf(output,"pop r2\ncmp r2,0\nbe else\n");
 match(THEN);
 stmt();

 fprintf(output,"b endif\n");
 Y();
 fprintf(output,"endif:\n");
 break;
 case WHILE:
 fprintf(output,"while:\n");
 match(WHILE);
 match('('); expr(); match(')');
 fprintf(output,"pop r2\ncmp r2,0\nbe endwhile\n");
 match(DO);
 stmt();
 fprintf(output,"b while\nendwhile:\n");
 break;
 case BEGIN:
 match(BEGIN);
 CS();
 match(END);
 break;
 default:
 return;
 }
}

void Y(){
```

```
if(lookahead==ELSE){
   match(ELSE);
   fprintf(output,"else:\n");
   stmt();
}

}

void rest(){


   switch(lookahead){
   case '=':match('=');expr(); fprintf(output,"pop %s\n",symtable[tok].lexptr); break;
   case '(':match('('); match(')');fprintf(output,"call %s:\n",symtable[tok].lexptr);break;
   default:error("rest error");



   }


}
void CS(){
 while(lookahead != END){
 stmt();match(';');
 }
}
void expr(){
 int t;
term();
while(1){
 switch (lookahead) {
 case '+': case '-':
```

```
 t = lookahead;
 match(lookahead);
 term(); emit(t, NONE);
 continue;
 default:
 return;
}
 }
}
void term(){
 int t;
factor();
while(1)
 switch (lookahead) {
 case '*': case '/': case DIV: case MOD:
 t = lookahead;
 match(lookahead);
 factor();
 emit(t,NONE);
 continue;
 default:
 return;
}
}
void factor(){
switch (lookahead) {
case '(':
 match('(');
 expr();
 match(')');
 break;
```

```
case NUM:
 emit(NUM, tokenval);
 match(NUM);
 break;
case ID:
 emit(ID, tokenval);
 match(ID);
 break;
default:
 error("syntax error");
}
}

void match(int t){
if (lookahead == t)
 lookahead = lexan();
else error("syntax error");
}


#endif // PARSE_H_
```