CS-323
Project 3
Due Wednesday 11/12 by 11:55 pm

Modify your parser from Project #2 so that it will produce a series of assembly language instructions that describe the actions in the program that is being compiled. The grammar that needs to be handled is the same that was used as input to the Parser program, the Parser will now just produce a different output.

The assembly language commands are listed below. The storage for the assembly language will be on the stack, so a postfix order is used for expressions.

The program should produce output of an assembly language program and output of the symbol table. The assembly code should appear as shown in the sample with line numbers (starting at 1), commands, and operands (if they exist). The output of the symbol table should include the name and address of all variables used in the program. The starting address for the symbol table should be 300 (as shown in the sample).

Compiler F14 Assembly instructions

PUSHV Val	Push constant (integer) value onto the top of the stack
PUSHM Addr	Push value from memory location onto the top of the stack
PUSHI	Push value from standard input onto the top of the stack
POPM Addr	Pop value from top of stack and store to memory location
POPO	Pop value from top of stack and send to the standard output device
ADD	Pop the top 2 values from the stack and place their sum on the top of the stack
SUB	Pop the top 2 values from the stack and place their difference (second_from_top_value - top_value) on the top of the stack.
MUL	Pop the top 2 values from the stack and place their product on the top of the stack.
DIV	Pop the top 2 values from the stack and place their quotient (second_from_top_value / top_value) on the top of the stack.
GRTR	Pop the top 2 values from the stack. Push 1 onto the stack if second_from_top_value > top_value, otherwise push 0 onto the stack.
LESS	Pop the top 2 values from the stack. Push 1 onto the stack if second_from_top_value < top_value, otherwise push 0 onto the stack.
EQL	Pop the top 2 values from the stack. Push 1 onto the stack if second_from_top_value = top_value, otherwise push 0 onto the stack.
JUMP Addr	Unconditionally jump to address in program
JMPZ Addr	Pop the top value from the stack. If the value is 0, jump to address in the program.
NOT	Pop the top value from the stack. If the value is 0, push 1 onto the stack. If the value is not zero, push 0 onto the stack.
NOP	No OPeration.

Sample code

```
program
  int i, val, fact;
begin
    read( val );
    fact <– 1;
    i <– 0;
    while( i < val + 1 )
      begin
        fact <– fact * i;
        i <– i + 1;
      end
    write( i - 1 );
    write( fact );
end
```

Symbol table

i	300
val	301
fact	302

Assembly code

```
1     PUSHI
2     POPM    301
3     PUSHV   1
4     POPM    302
5     PUSHV   0
6     POPM    300
7     PUSHM   300
8     PUSHM   301
9     PUSHV   1
10    ADD
11    LESS
12    JMPZ    22
13    PUSHM   302
14    PUSHM   300
15    MUL
16    POPM    302
17    PUSHM   300
18    PUSHV   1
19    ADD
20    POPM    300
21    JUMP    7
22    PUSHM   300
23    PUSHV   1
24    SUB
25    POPO
26    PUSHM   302
27    POPO
```