Project: Employ the *Unix* utility ***yacc*** to create a parser for P-Code programs. Add the parser to the scanner so that your P-Code Assembler consists of two phases, the scanner and the parser.

Program Files:

File	Description
pasmlex.h	File **pasmlex.h** contains the interface to the lexer and supporting functions defined in file **pasmlex.l**. Modify pasmlex.l to include file **pasmtkn.h** created by the makefile for this project.
pasmlex.l	File **pasmlex.l** contains specifications acceptable to the *Unix* utility ***lex*** for Subset Pascal tokens.
pasmpar.h	File **pasmpar.h** contains the interface to the parser and supporting functions defined in file **pasmpar.y**.
pasmpar.y	File **pasmpar.y** contains specifications acceptable to the *unix* utility ***yacc*** for the Subset Pascal grammar. Specifications for the grammar of the Pascal Subset are given in Table 1.
pasmtkn.h	File **pasmtkn.h** is produced by the *unix* utility ***yacc*** as a result of processing file **pasmpar.y**. File **pasmtkn.h** contains instructions that define the integer codes to tokens. File **pasmtkn.h** is file **y.tab.h** that has been renamed to file **pasmtkn.h**.
pasm.cpp	File **pasm.cpp** contains function main and processes command line arguments.
makep-assembler	File **makep-assembler** contains instructions for program **pam**. Instructions are written for the *Unix* utility ***make***. Program **pasm** is contained in file **pasm**.
makepasm	File **makepasm** is a Unix script file that removes old file created in the last creation of executable file **pas** and invokes file **makep-assembler** to create a new executable file **pasm**. File makepasm is given below. rm *.o rm pasmlex.cpp rm pasm make -f makep-assembler

Command Line: Project **2** can be invoked with zero or one program parameters. The first program parameter is the input file name. Sample command lines together with corresponding actions by program **pasm** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

$ **pasm**
Enter the input file name: **p00.pasm**

$ **pasm p00.pasm**

Input File: The input file contains a P-Code Assembler program. The input file name must have the suffix "**.pasm**" Please refer to figure 1 for an example of the format of an input file.

Output Files: There are two output files have suffixes, "**.pex**" and "**.trc**". The "**.pex**" output file contains the binary executable form of the input "**.pasm**" file. The "**.trc**" file contains a trace of the translation steps to help the designed locate and remove errors in the assembler.

The "**.pex**", "**.atrc**" and "**.alst**" files have the same prefix as the input "**.pasm**" file. For example, if the command below is issued

$ **pasm p00.pasm**

Files **p00.pex, p00.atrc,** and **p00.alst** are produced as shown in Figure 2.

```
L00001      ent     sp      L00002
            ent     ep      L00003
            rtn     p
#define     L00002  4
#define     L00003  4
            mst     0
            cup     0       L00001
            stp
```

Figure 1. Example input file **p00.pasm.**

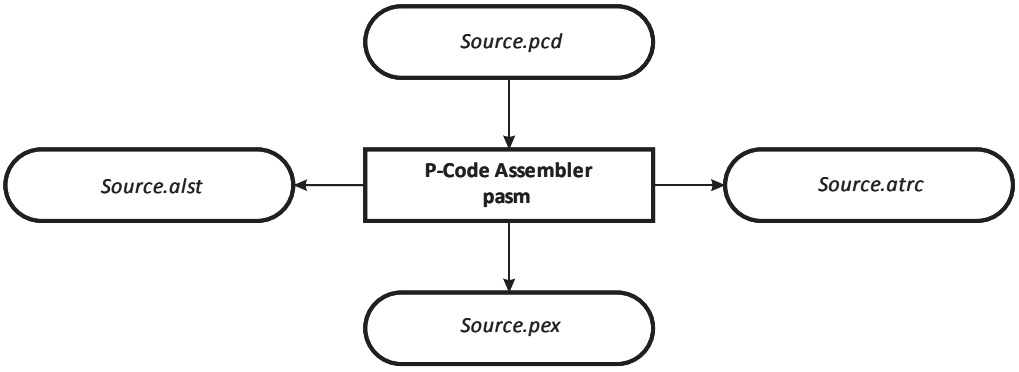

Figure 2. P-Code Assembler Block Diagram

Rule	LHS		RHS
1	*program*	→	*list*
2	*list*	→	*item*
3	*list*	→	*list item*
4	*item*	→	*statement*
5	*item*	→	*definition*
6	*definition*	→	**define label intlit**
7	*statement*	→	*label-list operation*
8	*statement*	→	*operation*
9	*label-list*	→	**label**
10	*label-list*	→	*label-list* **label**
11	*operation*	→	*class0-operation*
12	*operation*	→	*class1-operation*
13	*operation*	→	*class2-operation*
14	*operation*	→	*class3-operation*
15	*class0-operation*	→	**adi**
16	*class0-operation*	→	**sbi**
17	*class0-operation*	→	**ngi**
18	*class0-operation*	→	**mpi**
19	*class0-operation*	→	**dvi**
20	*class0-operation*	→	**abi**
21	*class0-operation*	→	**mod**
22	*class0-operation*	→	**adr**
23	*class0-operation*	→	**sbr**
24	*class0-operation*	→	**ngr**
25	*class0-operation*	→	**mpr**
26	*class0-operation*	→	**dvr**
27	*class0-operation*	→	**abr**
28	*class0-operation*	→	**ior**
29	*class0-operation*	→	**xor**
30	*class0-operation*	→	**and**
31	*class0-operation*	→	**not**
32	*class0-operation*	→	**inn**
33	*class0-operation*	→	**uni**
34	*class0-operation*	→	**ntr**
35	*class0-operation*	→	**dif**
36	*class0-operation*	→	**cmp**
37	*class0-operation*	→	**flt**
38	*class0-operation*	→	**flo**
39	*class0-operation*	→	**trc**
40	*class0-operation*	→	**stp**

Table 1 P-Code Assembler Grammar.

Rule	LHS		RHS
41	*class1-operation*	→	**mst intlit**
42	*class1-operation*	→	**rtn** *type*
43	*class1-operation*	→	**equ** *type*
44	*class1-operation*	→	**neq** *type*
45	*class1-operation*	→	**les** *type*
46	*class1-operation*	→	**leq** *type*
47	*class1-operation*	→	**grt** *type*
48	*class1-operation*	→	**geq** *type*
49	*class1-operation*	→	**sti** *type*
50	*class2-operation*	→	**csp** *stdfunction*
51	*class2-operation*	→	**ujp label**
52	*class2-operation*	→	**fjp label**
53	*class2-operation*	→	**tjp label**
54	*class2-operation*	→	**xjp label**
55	*class2-operation*	→	**ixa intlit**
56	*class3-operation*	→	**cup intlit label**
57	*class3-operation*	→	**ent** *register* **intlit**
58	*class3-operation*	→	**lda intlit intlit**
59	*class3-operation*	→	**ldc intlit intlit**
60	*class3-operation*	→	**ldi intlit intlit**
61	*class3-operation*	→	**lva intlit intlit**
62	*class3-operation*	→	**lvb intlit intlit**
63	*class3-operation*	→	**lvi intlit intlit**
64	*class3-operation*	→	**lvr intlit intlit**
65	*class3-operation*	→	**lvt intlit intlit**
66	*type*	→	**p**
67	*type*	→	**a**
68	*type*	→	**b**
69	*type*	→	**c**
70	*type*	→	**i**
71	*type*	→	**r**
72	*type*	→	**s**
73	*type*	→	**t**
74	*register*	→	**sp**
75	*register*	→	**ep**
76	*register*	→	**mp**
77	*register*	→	**pc**
78	*register*	→	**np**

Table 1 P-Code Assembler Grammar. (continued)

Rule	LHS		RHS
79	*stdfunction*	→	**rdb**
80	*stdfunction*	→	**rdc**
81	*stdfunction*	→	**rdi**
82	*stdfunction*	→	**rdr**
83	*stdfunction*	→	**rln**
84	*stdfunction*	→	**wrb**
85	*stdfunction*	→	**wrc**
86	*stdfunction*	→	**wri**
87	*stdfunction*	→	**wrr**
88	*stdfunction*	→	**wrs**
89	*stdfunction*	→	**wrt**
90	*stdfunction*	→	**sqt**
91	*stdfunction*	→	**ln**
92	*stdfunction*	→	**exp**

Table 1 P-Code Assembler Grammar. (continued)