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. Produce a trace file having the suffix ".atrc" and containing a description of every token encountered and every rule recognized in the source file. The contents of a sample trace file are shown in Figure 3. The grammar for P-Code programs is given in Table 1.	
Program Files:	**File**	**Description**
	makepasm	File **makepasm** contains instructions for program **pasm**. Instructions are written for the *Unix* utility ***make***. Program **pasm** is contained in file **pasm**.
	mkpasm	File **mkpasm** is a Unix script file that removes old file created in the last creation of executable file **pas** and invokes file **makepasm** to create a new executable file **pasm**.
Command Line:	Project **p04-2** 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:	A single output file is produced. The output file has the same prefix as the input **source.pcd** and the suffix **.pcd** is replaced by the suffix **.atrc**. For example, if the input file was named **hello.pcd**, the output file would be named **hello.atrc**. $ **pasm p00.pcd** File **p00.atrc** is 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

```
Token=(258,LABEL,L00001)
#009 label-list->LABEL
Token=(268,ENT_O,ent)
Token=(338,SP_R,sp)
#081 register->SP
Token=(258,LABEL,L00002)
060 class2-operation->ent register INTLIT
014 operation->class3-operation
#007 statement->label-list operation
#004 item->statement
#002 list->item
Token=(268,ENT_O,ent)
Token=(339,EP_R,ep)
#082 register->EP
Token=(258,LABEL,L00003)
060 class2-operation->ent register INTLIT
014 operation->class3-operation
#008 statement->operation
#004 item->statement
#003 list->list item

Token=(270,RTN_O,rtn)
Token=(335,P_T,p)
#072 type->P
042 class1-operation->rtn type
012 operation->class1-operation
#008 statement->operation
#004 item->statement
#003 list->list item

Token=(259,DEFINE,#define)
Token=(258,LABEL,L00002)
Token=(260,INTLIT,4)
#006 definition->DEFINE LABEL INTLIT
#005 item->definition
#003 list->list item

Token=(259,DEFINE,#define)
Token=(258,LABEL,L00003)
Token=(260,INTLIT,4)
#006 definition->DEFINE LABEL INTLIT
#005 item->definition
#003 list->list item
```

Figure 3. Example Output file p00.atrc.

```
Token=(269,MST_O,mst)
Token=(260,INTLIT,0)
041 class1-operation->mst intlit
012 operation->class1-operation
#008 statement->operation
#004 item->statement
#003 list->list item

Token=(266,CUP_O,cup)
Token=(260,INTLIT,0)
Token=(258,LABEL,L00001)
059 class2-operation->cup INTLIT LABEL
014 operation->class3-operation
#008 statement->operation
#004 item->statement
#003 list->list item

Token=(314,STP_O,stp)
040 class0-operation->stp
011 operation->class0-operation
#008 statement->operation
#004 item->statement
#003 list->list item

#001 program->list
```

Figure 3. Example Output file p00.atrc. (continued)

	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*	→	**mod**
21	*class0-operation*	→	**abi**
22	*class0-operation*		**sqi**
23	*class0-operation*	→	**adr**
24	*class0-operation*	→	**sbr**
25	*class0-operation*	→	**ngr**
26	*class0-operation*	→	**mpr**
27	*class0-operation*	→	**dvr**
28	*class0-operation*	→	**abr**
29	*class0-operation*	→	**sqr**
30	*class0-operation*	→	**ior**
31	*class0-operation*	→	**xor**
32	*class0-operation*	→	**and**
33	*class0-operation*	→	**not**
34	*class0-operation*	→	**inn**
35	*class0-operation*	→	**uni**
36	*class0-operation*	→	**ntr**
37	*class0-operation*	→	**dif**
38	*class0-operation*	→	**cmp**
39	*class0-operation*	→	**sgs**
40	*class0-operation*	→	**flt**
41	*class0-operation*	→	**flo**
42	*class0-operation*	→	**trc**
43	*class0-operation*	→	**rnd**
44	*class0-operation*	→	**chr**
45	*class0-operation*	→	**ord**

Table 1. P-Code Assembler Grammar.

	Rule		
	LHS		**RHS**
46	*class0-operation*	→	**stp**
47	*class1-operation*	→	**mst intlit**
48	*class1-operation*	→	**rtn** *type*
49	*class1-operation*	→	**equ** *type*
50	*class1-operation*	→	**neq** *type*
51	*class1-operation*	→	**les** *type*
52	*class1-operation*	→	**leq** *type*
53	*class1-operation*	→	**grt** *type*
54	*class1-operation*	→	**geq** *type*
55	*class1-operation*	→	**ldi** *type*
56	*class1-operation*	→	**sti** *type*
57	*class1-operation*	→	**inc** *type*
58	*class1-operation*	→	**dec** *type*
59	*class2-operation*	→	**csp** *stdfunction*
60	*class2-operation*	→	**ujp label**
61	*class2-operation*	→	**fjp label**
62	*class2-operation*	→	**tjp label**
63	*class2-operation*	→	**xjp label**
64	*class2-operation*	→	**ixa intlit**
65	*class3-operation*	→	**cup intlit label**
66	*class3-operation*	→	**ent** *register* **label**
67	*class3-operation*	→	**lda intlit intlit**
68	*class3-operation*	→	**ldc** *type* **intlit**
69	*class3-operation*	→	**ldc** *type* **realit**
70	*class3-operation*	→	**ldc** *type* **chrlit**
71	*class3-operation*	→	**ldc** *type* **strlit**
72	*class3-operation*	→	**lva intlit intlit**
73	*class3-operation*	→	**lvb intlit intlit**
74	*class3-operation*	→	**lvc intlit intlit**
75	*class3-operation*	→	**lvi intlit intlit**
76	*class3-operation*	→	**lvr intlit intlit**
77	*class3-operation*	→	**lvs intlit intlit**
78	*class3-operation*	→	**lvt intlit intlit**
79	*type*	→	**p**
80	*type*	→	**a**
81	*type*	→	**b**
82	*type*	→	**c**
83	*type*	→	**i**
84	*type*	→	**r**
85	*type*	→	**s**
86	*type*	→	**t**
87	*type*	→	**x**

Table 1. P-Code Assembler Grammar. (continued)

	Rule		
	LHS		**RHS**
88	*register*	→	**sp**
89	*register*	→	**ep**
90	*register*	→	**mp**
91	*register*	→	**pc**
92	*register*	→	**np**
93	*stdfunction*	→	**rdb**
94	*stdfunction*	→	**rdc**
95	*stdfunction*	→	**rdi**
96	*stdfunction*	→	**rdr**
97	*stdfunction*	→	**rln**
98	*stdfunction*	→	**wrb**
99	*stdfunction*	→	**wrc**
100	*stdfunction*	→	**wri**
101	*stdfunction*	→	**wre**
102	*stdfunction*	→	**wrf**
103	*stdfunction*	→	**wrs**
104	*stdfunction*	→	**wrt**
105	*stdfunction*	→	**wln**
106	*stdfunction*	→	**sqt**
107	*stdfunction*	→	**ln**
108	*stdfunction*	→	**exp**

Table 1. P-Code Assembler Grammar. (continued)