Project:	Employ the U*nix* utility ***lex*** to create a lexical analyzer for the P-Code Assembler. Reserve words, mnemonics, numeric constants, and other token descriptions can be extracted from the P-Machine Specification (http://cs2.uco.edu/~trt/cs4173/pspec.pdf).	
Program Files:	**File**	**Description**
	PasmToken.h	File **PasmToken.h** contains the list of positive integer codes that uniquely identify each token. #define macro directives are used to define each token. For example, #define PROGRAM 309. Alternatively, you may construct this list using an enumerated type. However, you must ensure that every token has a ***positive*** integer code.
	PasmScanner.h	File **PasmScanner.h** contains the implementation of the lexer and supporting functions defined in file **PasmScanner.l**.
	PasmScanner.l	File PasmScanner.l defines the interface to the lexer and supporting functions.
	Pasm.cpp	File **Pasm.cpp** contains function main and processes command line arguments.
	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**. File makepasm is given below. rm *.o rm pasmlex.cpp rm pasm make -f makepasm
Command Line:	Project **1** 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.pcd**	
Input File:	The input file contains a P-Code Assembler program. The input file name must have the suffix **.pcd**. 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.trc** 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.**

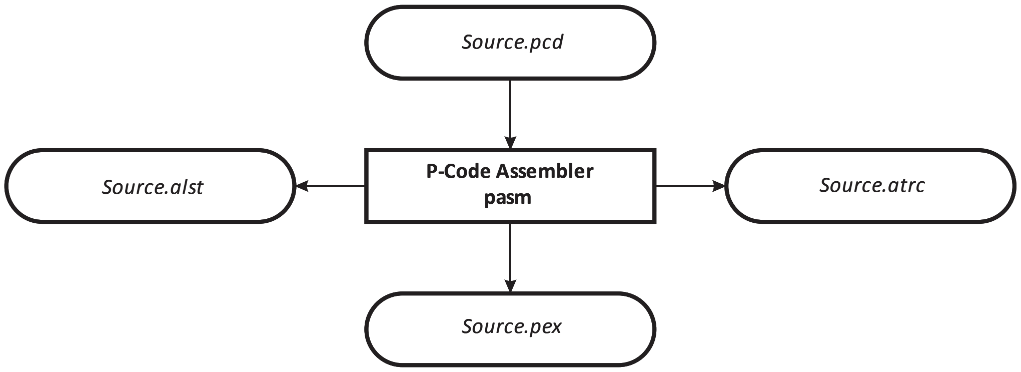

Figure 2. P-Code Assembler Block Diagram

Token Code	Token Name	Pattern	Token Code	Token Name	Pattern
1	CUP_O	cup	46	RND_O	rnd
2	CSP_O	csp	47	CHR_O	chr
3	ENT_O	ent	48	ORD_O	ord
4	MST_O	mst	49	STP_O	stp
5	RTN_O	rtn	50	LDA_O	lda
6	EQU_O	equ	51	LDC_O	ldc
7	NEQ_O	neq	52	LDI_O	ldi
8	GRT_O	grt	53	LVA_O	lva
9	GEQ_O	geq	54	LVB_O	lvb
10	LES_O	les	55	LVC_O	lvc
11	LEQ_O	leq	56	LVI_O	lvi
12	ADI_O	adi	57	LVR_O	lvr
13	SBI_O	sbi	58	LVT_O	lvt
14	NGI_O	ngi	59	STI_O	sti
15	MPI_O	mpi	60	IXA_O	ixa
16	DVI_O	dvi	61	RDB_F	rdb
17	MOD_O	mod	62	RDC_F	rdc
18	ABI_O	abi	63	RDI_F	rdi
19	SQI_O	sqi	64	RDR_F	rdr
20	INC_O	inc	65	RLN_F	rln
21	DEC_O	dec	66	WRB_F	wrb
22	ADR_O	adr	67	WRC_F	wrc
23	SBR_O	sbr	68	WRI_F	wri
24	NGR_O	ngr	69	WRE_F	wre
25	MPR_O	mrp	70	WRF_F	wrf
26	DVR_O	dvr	71	WRS_F	wrs
27	ABR_O	abr	72	WLN_F	wln
28	SQR_O	sqr	73	SQT_F	sqt
29	IOR_O	ior	74	LN_F	ln
30	AND_O	and	75	EXP_F	exp
31	XOR_O	xor	76	SP_R	sp
32	NOT_O	not	77	EP_R	pc
33	INN_O	inn	78	MP_R	mp
34	UNI_O	uni	79	PC_R	pc
35	INT_O	int	80	NP_R	np
36	DIF_O	dif	81	A_T	a
37	CMP_O	cmp	82	B_T	b
38	SGS_O	sgs	83	C_T	c
39	UJP_O	ujp	84	I_T	i
40	XJP_O	xjp	85	R_T	r
41	FJP_O	fjp	86	S_T	s
42	TJP_O	tjp	87	T_T	t
43	FLT_O	flt	88	P_T	p
44	FLO_O	flo	89	X_T	x
45	TRC_O	trc	90	DEFINE	#define

Token Code	**Token Name**	**Pattern**
91	LABEL	A label begins with the capital letter L and is followed by any number of integer digits. Example: L00001
92	INTLIT	An integer literal is an optional sign followed by one or more digits.
93	REALIT	A real literal – a real number constant – consists of one or more integer digits, a decimal point, one or more fractional digits, and an optional exponent. The exponent, if present, is the letter e, an optional sign, and one or more digits. Remember, P-Code is case-insensitive so the letter e may be capitalized.
93	REALIT	A real literal – a real number constant – consists of one or more integer digits and an exponent.
94	CHRLIT	A character constant – a single character enclosed between apostrophes. For example, the character constant, t, appears in the source as ‘t’. An apostrophe is double so that a single apostrophe appears as “”’.
95	STRLIT	A string constant – two or more characters enclosed between apostrophes. For example, the string constant banana appears as ‘banana’. Apostrophes in string constants are doubled so that the constant don’t appears as ‘don’’t’.
96	ID	Identifier – One or more lower case letters
97	ERROR	An unrecognized token