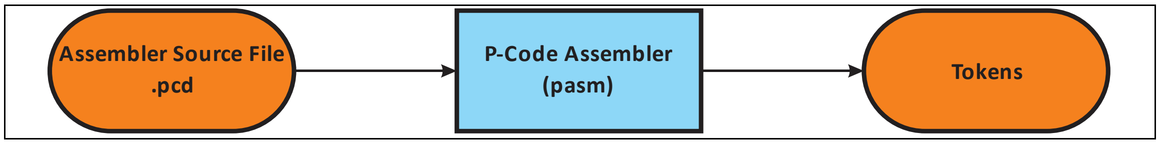

Primary goal: Recognize all tokens defined for the Pasm Scanner as given in Project P04.1, Pasm Scanner (http://cs2.uco.edu/~trt/cs5023/p04.1.pdf).

Term	Definition
Token	2-tuple • a unique positive integer code • recognized string

Example:
a:=2*pi*r*r;

Integer Code	Comment	String
1	Identifier	"a"
2	assignment operator	"="
3	integer constant	"2"
4	multiplication operator	"*"
1	identifier	"pi"
4	multiplication operator	"*"
1	identifier	"r"
1	identifier	"r"
5	semicolon	";"

Secondary goal: We want the positive integer codes associated with strings to be easily mapped their corresponding values in the binary pex file when translated to binary.

Example: Consider the P-Code instruction adi – add integer. The instruction mnemonic adi is translated to 0x0B, a one-byte value having the decimal equivalent of 11. We would like to assign the integer code returned when the string adi is recognized to be easily mapped 11.

As it turns out, we cannot achieve our desire and use the tool, *lex*, that we employ to create our scanner. However, we can make the integer code relative to a larger, fixed code such that when the integer code for adi is subtracted from our larger fixed code the difference is 11.

If we review the Token Code – Token Name table given in file p04-1.docx, the P-Code Assembler Scanner, we note that all the instruction mnemonics, standard function mnemonics, register mnemonics, and type mnemonics are all ordered as they are ordered in the P-Machine Specification (http://cs2.uco.edu/~trt/cs5023/pspec-2020-02-04.pdf).

Testing: Your instructor suggests that you create source files containing at least one instance of every kind of token recognized by the scanner.