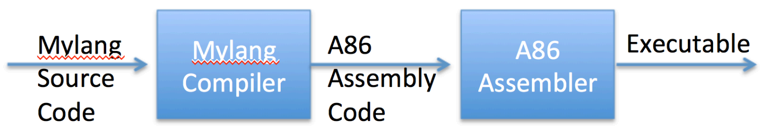
CMPE 230 Systems Programming

Homework 1 (due March. 23rd)

(This project can be implemented in C/C++ or Java)

In this project, you will implement a compiler for a language called Mylang that will compile Mylang code to A86 assembly language code. The assembly code can then be assembled by the A86 assembler to produce an executable file.



The grammar for Mylang will be as follows:

```
stm          →    id = expr
                | print expr
                | read id
                | if  expr then stm
                | while expr do stm
                | begin opt_stmts end

opt_stms     →    stmt_list
                | ε

stmt_list    →    stm ; stmt_list
                | stm

expr         →    term  moreterms

moreterms    →    +  term moreterms
                | - term moreterms
                | ε

term         →    factor  morefactors

morefactors  →    *   factor  morefactors
                | /   factor  morefactors
                | %   factor  morefactors
                | ε

factor       →    ( expr )
                | id
                | num
```

Your compiler should be able to parse codes given in Mylang following the grammar rules given above. Note that **id** is an identifier (variable) and **num** is an integer. You can assume only non-negative integers can be read in Mylang language

Consider the following operations. Your compiler will basically translate Mylang code by making use of the following operations.

+ - * / %	Binary operations: Pop two values from the stack, perform the binary operation and push the result onto the stack.
push-num n	push number n onto the stack
push-val-var v	push value of variable v onto the stack
push-addr-var v	push address of variable v onto the stack
pop	pop value on top of the stack
assign	the value on top of stack is placed in the address below it and both are popped from the stack.
print	Print the value on top and then pop the value.
label LABL	Target of jumps to label LABL, has no other effect (i.e. no operation)
jump LABL	Unconditional jump to label LABL
jump-if-false LABL	Pop the value on top of the stack and then jump to label LABL if it is zero
stop	Stop execution and return to the operating system

Here are some example of translations of small code fragments:

__Example 1__

The program **val = (461*y) div 4 + (200*m+2) div 5 + d** is translated to the following instructions:

	Abstract Instruction	A86 Instuctions
0	push-addr-var val	PUSH offset VAL
1	push-num 461	PUSH 461
2	push-val-var y	PUSH Y
3	*	POP CX POP AX MULT CX PUSH AX
4	push-num 4	PUSH 4
5	div	MOV DX,0 POP CX POP AX DIV CX PUSH AX
6	push-num 200	PUSH 200
7	push-val-var m	PUSH M

8	*	POP CX POP AX MULT CX PUSH AX
9	push-num 2	PUSH 2
10	+	POP CX POP AX ADD AX,CX PUSH AX
11	push-num 5	PUSH 5
12	div	MOV DX,0 POP CX POP AX DIV CX PUSH AX
13	+	POP CX POP AX ADD AX,CX PUSH AX
14	push-val-var d	PUSH D
15	+	POP CX POP AX ADD AX,CX PUSH AX
16	assign	POP AX POP BX MOV [BX],AX
17	stop	INT 20h

As seen in this example, infix expressions are converted into postfix expressions.

<u>Example 2</u>
Consider the following if statement

if *expr* then *stm*

It will be translated as:

code for *expr*
POP AX JZ OUTLABEL
code for *stm*
OUTLABEL NOP

<u>Example 3</u>
Consider a while loop:

while expression do stm

It will be translated as follows:

TESTLABEL NOP
code for *expr*
POP AX JZ OUTLABEL
code for *stm*
JMP TESTLABEL
OUTLABEL NOP

Your project will be graded according to the following criteria:

Documentation (written document describing how you implemented your project)	15%
Comments in your code	5%
Mylang Compiler implementation and tests	80%