# University of Central Florida
## Department of Computer Science
# COP 3402: System Software
# Montagne, Spring 2022

**Homework #4 (PL/0 Compiler)**

**Due by 11:59 p.m.**

**REQUIRMENT:**

**All assignments must compile and run on the Eustis3 server. Please see course website for details concerning use of Eustis3.**

**Objective:**

In this assignment, you must correct your previous homework and modify them to add AND, OR, and NOT functionality.

- VM Changes:
  - Add the following instructions to the ISA:

    2 0 12 AND Take the logical AND of the two values at the bottom of the register stack. If they are both 1, set the higher position to 1. Otherwise, set it to 0. RP must increase.

    2 0 13 ORR Take the logical OR of the two values at the bottom of the register stack. If either is equal to 1, set the higher position to 1. Otherwise, set it to 0. RP must increase.

    2 0 14. NOT Set the value at the bottom of the register stack to its logical reverse. (1 becomes 0, 0 becomes 1)
- Scanner Changes:
  - Add the special symbols andsym = 32 (&&), orsym = 33 (||), and notsym (!)
- Parser Changes:
  - Add a syntactic class/function called LOGIC and augment CONDITION

The modified PL/0 grammar is define as:

**EBNF of tiny PL/0:**

program ::= block "**.**" **.**
block ::= const-declaration var-declaration procedure-declaration statement**.**
const-declaration ::= ["**const**" ident ":=" number {"**,**" ident ":=" number} "**;**"]**.**
var-declaration ::= [ "**var** "ident {"**,**" ident} "**;**"]**.**
procedure-declaration ::= { "**procedure**" ident "**;**" block "**;**" }.
statement ::= [ ident "**:=**" expression
| "**call**" ident
| "**begin**" statement { "**;**" statement } "**end**"
| "**if**" logic "**then**" statement ["**else**" statement]
| "**while**" logic "**do**" statement
| "**read**" ident
| "**write**" expression
| **ε** ] **.**
**logic ::= "!" condition | condition {("&&"|"||") condition}.**
**condition ::= expression rel-op expression | "(" logic ")".**
rel-op ::= "="|"**!**="|"<"|"<="|">"|">=".
expression ::= [ "+"|"**-**"] term { ("+"|"**-**") term}**.**
term ::= factor {("*"|"/"|"") factor}**.**
factor ::= ident | number | "**(**" expression "**)**".
number ::= digit {digit}**.**
ident ::= letter {letter | digit}**.**
digit ;;= "**0**" | "**1**" | "**2**" | "**3**" | "**4**" | "**5**" | "**6**" | "**7**" | "**8**" | "**9**".
letter ::= "**a**" | "**b**" | … | "**y**" | "**z**" | "**A**" | "**B**" | ... | "**Y**" | "**Z**".

**Based on Wirth's definition for EBNF we have the following rule:**
**[ ] means an optional item.**
**{ } means repeat 0 or more times.**
**Terminal symbols are enclosed in quote marks.**
**A period is used to indicate the end of the definition of a syntactic class.**

**Changes in Error List**
8. if must be followed by then - found in **statement** in the if case when flow of control returns from **logic** and the current symbol is not then
9. while must be followed by do - found in **statement** in the while case when flow of control returns from **logic** and the current symbol is not do

12. ( must be followed by ) - found in **factor** in the parenthesis case when flow of control returns from **expression** without error, but a ) is not found OR in **condition** when flow of control returns from **logic** without error, but a ) is not found
20. Register Overflow Error - found when a load instruction (LOD, LIT, RED) should be emitted, but there is no more space on the register stack. Keep track of space on the register stack by using a counter variable. It should be incremented when a load instruction is emitted and decremented when STO, ADD, SUB, MUL, DIV, EQL, NEQ, LSS, LEQ, GTR, GEQ, JPC, AND, or ORR are emitted. The maximum register size is 10.

**Changes in Pseudocode**

PROGRAM, BLOCK, CONST-DECLARATION, VAR-DECLARATION, PROCEDURE-DECLARATION, STATEMENT, EXPRESSION, TERM, and FACTOR only need to call LOGIC instead of CONDITION; they need no other changes.

```
LOGIC
        if token == notsym
                CONDITION
                emit NOT
        else
                CONDITION
                while token == andsym || token == orsym
                        if token == andsym
                                CONDITION
                                emit AND
                        else
                                CONDITION
                                emit ORR

CONDITION
        if token == lparensym
                LOGIC
        else
                EXPRESSION
                if token == eqlsym
                        EXPRESSION
                        emit EQL
                else if token == neqsym
                        EXPRESSION
                        emit NEQ
                else if token == lsssym
                        EXPRESSION
                        emit LSS
                else if token == leqsym
                        EXPRESSION
                        emit LEQ
                else if token == gtrsym
                        EXPRESSION
                        emit GTR
                else if token == geqsym
                        EXPRESSION
```

emit GEQ

**Reminder:**

The compiler must read a program written in modified PL/0 and generate code for the Virtual Machine (VM) you implemented in HW1. Your compiler must neither parse nor generate code for programming constructs that are not in the grammar described above.

**Rubric**

15 – Compiles
05 – README.txt containing author names if the given files are augmented
20 - correctly implements lex changes
20 - produces correct code and error recognition for and and or
20 - produces correct code and error recognition for not
20 - correctly implements new instructions in vm

Except for the modifications described in this document, The rule of the game for this assignment (HW4) are the same used in HW3.