COS341 Semester Project 2025 : Code-Generation from"SPL"

General assumptions: Scope-Analysis and consistent re-naming of user-defined names have already been carried out, and the Symbol Table has already been filled with the relevant information.

VARIABLES ::= VAR VARIABLES
VAR ::= user-defined-name

Translation Advice:
Variable-Declarations do not get translated to target code. The Variable-Declarations were only needed for "filling" the Symbol-Table which the translator (code-generator) is going to consult.

PDEF ::= NAME **(** PARAM **) {** BODY **}**

Translation Advice:
The Sub-Tree "under" tree-node PDEF **will later be used for "inlining"**, as discussed in the Lecture of Thursday the 2nd of October.

FDEF ::= NAME **(** PARAM **) {** BODY **; return** ATOM **}**

Translation Advice:
The Sub-Tree "under" tree-node FDEF **will later be used for "inlining"**, as discussed in the Lecture of Thursday the 2nd of October.

BODY ::= **local {** MAXTHREE **}** ALGO
PARAM ::= MAXTHREE
MAXTHREE ::=
MAXTHREE ::= VAR
MAXTHREE ::= VAR VAR
MAXTHREE ::= VAR VAR VAR

Translation Advice:
Variable-Declarations do not get translated to target code. The Variable-Declarations were only needed for "filling" the Symbol-Table which the translator (code-generator) is going to consult. **Only the ALGO will get translated**.

MAINPROG ::= **var {** VARIABLES **}** ALGO

Translation Advice:
Variable-Declarations do not get translated to target code. The Variable-Declarations were only needed for "filling" the Symbol-Table which the translator (code-generator) is going to consult. **Only the ALGO will get translated**.

Turn the page ►

ATOM ::= VAR

Translation Advice:

Similar to Trans(Exp → id) in **Fig.6.3** of our blue textbook,
however with a normal **=** (instead of the textbook's **:=**) in generated target code.

ATOM ::= number

Translation Advice:

Similar to Trans(Exp → num) in **Fig.6.3** of our blue textbook,
however with a normal **=** (instead of the textbook's **:=**) in generated target code.

ALGO ::= INSTR **;** ALGO

Translation Advice:

Similar to Trans(Stat → Stat1 **;** Stat2) in **Fig.6.5** of our blue textbook.

INSTR ::= **halt**

Translation Advice:

Trans(**halt**) is { *target_code* = " STOP " ; return(*target_code*) ; }

INSTR ::= **print** OUTPUT
OUTPUT := string
OUTPUT ::= ATOM
ATOM ::= number
ATOM ::= VAR

Translation Advice:

```
Trans( print OUTPUT ) is { if child_node(OUTPUT) == string then code = " "string" " ;
                           if child_node(child_node(OUTPUT)) == number
                             then code = " number " ;
                           if child_node(child_node(OUTPUT)) == VAR
                             then { internal = lookup_symbol_table(VAR) ;
                                    code = " internal " ; }
                           target_code = " PRINT " ++ code ; // comment: ++ is string-append
                           return(target_code) ; }
```

INSTR ::= ASSIGN
ASSIGN ::= VAR **=** TERM

Translation Advice:

Similar to Trans(Stat → **id** := Exp) in **Fig.6.5** of our blue textbook,
however with a normal **=** (instead of the textbook's **:=**) in generated target code.

Turn the page ►

```
INSTR     ::=   BRANCH
BRANCH    ::=   if TERM { ALGO } else { ALGO }
```

Translation Advice:

- Our Target-Language does not have the keyword *ELSE* in its syntax !
- Our Target-Language does not have the keyword *LABEL* in its syntax !

Therefore, the **translation techniques from Fig.6.5 and Fig.6.6** of our textbook ***must get modified*** as follows:

- *Instead of* " LABEL " ++ label ++ *we produce:* " REM " ++ label ++ " \newline " ++
- *Instead of* " IF t1 op t2 THEN labelT ELSE labelF " *we produce:*

```
" IF t1 op t1 THEN labelT ++  \newline ++
  code_of_the_else_ALGO  ++  \newline ++
  GOTO ++ labelExit ++  \newline ++
  REM ++ labelT ++  \newline ++
  code_of_the_then_ALGO  ++ \newline ++
  REM ++ labelExit ++  \newline ++ "
```

```
BRANCH    ::= if TERM { ALGO }
```

Translation Advice:

For the same reasons as described above, *we modify* **Fig.6.5 and Fig.6.6** to produce:

```
" IF t1 op t1 THEN labelT ++  \newline ++
  GOTO ++ labelExit ++  \newline ++
  REM ++ labelT ++  \newline ++
  code_of_the_then_ALGO  ++ \newline ++
  REM ++ labelExit ++  \newline ++ "
```

```
INSTR     ::=   LOOP
LOOP      ::=   while TERM { ALGO }
LOOP      ::=   do { ALGO } until TERM
```

Translation Advice:

Similar to Figures **6.5** and **6.6** in the textbook though you must keep in mind (as mentioned above) that

- our Target-Language uses the keyword **REM** *instead of* the keyword LABEL
- our Target-Language does ***not*** have the keyword ELSE

```
INSTR     ::=   NAME ( INPUT )     // procedure call without return
ASSIGN    ::=   VAR = NAME ( INPUT ) // function call with return
```

Translation Advice:

The Expression NAME(INPUT) gets translated as in Textbook **Fig.6.3** whereby a CALL command will emerge. (In a ***later*** phase, the CALL will get ***replaced*** by way of ***inlining***.)

TERM ::= ATOM
TERM ::= (UNOP TERM)
TERM ::= (TERM BINOP TERM)

Translation Advice:
Very similar to Figure 6.3 in the textbook!
However, wherever the textbook generates **:=** we simply generate **=** (*without* the colon).

UNOP ::= **neg** Trans(**neg**) is simply the minus symbol **-**

UNOP ::= **not**

Translation Advice:
As our target language does *not* have the Boolean negation symbol in its syntax, we must *translate* **if** (**not**(TERM)) *as we translate* **if** (TERM) , however with *swapping* the **then**-Algo for the **else**-ALGO (and, vice versa, the else-ALGO for the then-ALGO).

BINOP ::= **eq**

Translation Advice:
Trans(**eq**) is simply the equality symbol **=** (which our target language "overloads" for both value assignments as well as also for the Boolean comparisons).

BINOP ::= **>** Trans(**>**) is simply itself: **>**

BINOP ::= **or**
BINOP ::= **and**

Translation Advice:
As our target language does *not* have any Boolean operators in its syntax, we use the "cascading" translation technique from **Fig.6.8** of our textbook whereby, however, we must *again* keep in mind that

- our Target-Language does **not** have any keyword ELSE
- our Target-Language uses **REM label** \newline , instead of LABEL **label**

BINOP ::= **plus** Trans(**plus**) is simply **+**
BINOP ::= **minus** Trans(**minus**) is simply **-**
BINOP ::= **mult** Trans(**mult**) is simply ***
BINOP ::= **div** Trans(**div**) is simply **/**

The generated target code must be stored in a *.txt (ASCII) output file

IF in doubt → then go to Tutor's Consultation Hour in the Orange Lab

And now: HAPPY CODING :)