PROJECT Specification

Task Booklet 05

COS**341** 2020

INTRODUCTION

- **BASIC** is a programming language that was designed in 1964. It will be the language you use as the “intermediate language” of your SPL compiler.*
- You must ensure that the BASIC code which your compiler generates is able to run (on an interpreter) and is semantically equivalent to your SPL source code. You may use any BASIC interpreter (from the internet) as long as it supports the original BASIC language from the 1970’s.
- It is your task to find a BASIC interpreter and learn the language as needed.

*** Footnote:** In this project we will not generate genuine machine code. Thus, in this project, your “intermediate code” is effectively your “target code”.

RULES

- The **BASIC** constructs you are allowed to use is defined in the next slide. The remainder of this slide will define how to read the rules in the next slide.
- **line number** refers to the line number of a BASIC statement.
- **variable name** refers only to variables by a name in BASIC.
- **+ - * < > =** are the operators you are allowed to use.
- **integer literal** is a hard coded literal number in BASIC code.
- **string literal** is a hard coded literal string in BASIC code.
- **[x | …]** is a set of one or more kinds of constructs listed above. Any of the constructs is allowed to be used in the position this is placed.
- *Anything else should be taken as verbatim.*

RULES

- You must generate BASIC code that use only the following simple statements from the *early* BASIC language. In other words: *you are NOT allowed to use the high-level-constructs* (e.g.: IF-THEN-**ELSE**, etc.) which newer editions of BASIC offer for the sake of convenience.
    - **GOTO** [line number]
    - **INPUT** [variable name]
    - **PRINT** [variable name]
    - **GOSUB** [line number]
    - **GOTO** [line number]
    - **RETURN**
    - **IF** [variable name] **THEN GOTO** [line number]
    - [variable name] = **NOT** [variable name]
    - [variable name] = [variable name | integer literal] [+-*><] [variable name | integer literal]
    - [variable name] = [variable name | integer literal | string literal] [=] [variable name | integer literal | string literal]
    - [variable name] = [variable name | integer literal | string literal]

Note: each of these statements will be line-numbered. Line numbers are symbolic addresses.

RULES

- You are **not allowed** to combine rules and make complex expressions. The following statement is **NOT ALLOWED**:
  - **V0 = V1 * V2 + V3 * V4**
- You must **instead** do the following:
  - V5 = V1 * V2
  - V6 = V3 * V4
  - V0 = V5 + V6

$\}$ **See book, chapter 6, on Code-Generation**

- Ensure that you stick to the rules given to you as strictly as possibly.

CONDITIONAL LOGIC

- You are **not allowed** to use any **AND**, **OR** constructs in BASIC. But, as SPL supports these, you must translate these features of SPL into BASIC **GOTO** jumps.
- In other words: Your **AND** and **OR** Logic must have short-circuiting behaviour in the BASIC code. Thus, for ***A*** **OR** ***B***, if ***A*** is true then ***B*** must be 'skipped'. Similarly, with ***A*** **AND** ***B***, if ***A*** is false then ***B*** must be 'skipped'. **See Fig.6.8** in your textbook.

ADDITIONAL ADVICE

- With HALT (in SPL) you may simply have a GOTO ***H*** where ***H*** is the line number for the final statement in your target program. This final line "does nothing".
- For **T**(rue) and **F**(alse) in SPL you may use the integers **0** and **1** in your generated BASIC code.

YOUR TASK

- Using the lessons from Chapter 6 of your book you must implement a code-generating ***trans*** function (which is <u>recursive</u>).
  - First you must generate an “abstract” source file without line numbers; instead of the line numbers you have abstract labels.
  - In a next transformation step you must generate the line numbers for proper BASIC code, out of the abstract labels, into another file.
- For **assessment** you will be asked run the generated BASIC code. Therefore you are required to have a working BASIC interpreter installed. Your generated BASIC code must be semantically equivalent to the SPL source code from which the BASIC code was generated.

ADDITIONAL NOTES

- **Plagiarism is not allowed!** You or your group may not use any code written by someone not within your own group.

And now:

Happy coding ☺☺☺