PROJECT – Part 6: Generator

Youssef Yousfi

Hajar El Boutahiri

Hiba Aqqaoui

CSC 3315 Languages and Compilers

Professor Violetta Cavalli-Sforza

PROJECT – Part 6: Generator

PROJECT – Part 6: Generator

Documentation:

To complete this phase, we worked for more than 15 hours, including weekends, and whole nights.

The teammates are:

- Youssef Yousfi
- Hajar El Boutahiri
- Hiba Aqqaoui

Changes/Improvement clearly noted:

- We added two predefined functions for input/output instructions: read and print.
- We handled some errors

Running instructions:

In order to run the code: you need to have first all files in one folder:

In other words:

- Start by unzipping the folder submitted on Canvas:
- Copy the address of the folder from your computer
- Open cmd to run the code
- Write cd + the address folder
- Copy the Command Python: Generator.py

PROJECT – Part 6: Generator

Generator Level 1:

Check Test Suite

Generator Level 2:

Step 1: Subprogram call & return design, global vars

This is the way our proposed design for the first step of level 2 would look like, we would implement a specialized stack memory of 1000 words instead of sharing it with the global data memory:

Sub data memory #1

Data of 1st function goes here

-9 999 999 999

Sub code memory #1

Code of 1st function goes here

-9 999 999 999

Sub data memory #2

Data of 2nd function goes here

-9 999 999 999

Sub code memory #2

Code of 2nd function goes here

…

+9 999 999 999

Data memory

+9 999 999 999

Code memory

+9 999 999 999

Where:

Sub data memory #k and sub code memory #k are the memory spaces for a certain declared function. We have as many sub data and sub code memories as there are functions. The separator in between the memories is -9999999999 instead of +9999999999, and this is to separate between the function memories and the main memory in the interpreter. The EP points to the current local context. The SP points to the top of the stack. We would implement a static scoping in which our global variables are accessible from everywhere whereas the local static variables are visible from the function defining them.

PROJECT – Part 6: Generator

Changes to the instruction set:

In order to execute a function, we can make a change in the instruction set, using ‘-9’ following this format:

-9 000 000 dst : to execute the code of the function at memory location #dst, and assign at the end using opcode ‘-0’ if there is a return.

For example: -9 000 000 001, would execute the function at sub data and sub code memories #1

In order to return, we can make a change in the instruction set, using ‘-0' following this format:

-0 src 000 dst : Assign SDM[src] to DM[dst]

For example: -0 002 000 003, would assign the value from sub data memory 002 of the function to the value of data memory 003.

Step 2: EC: Subprogram call & return implementation

Check Test Suite