Assessed Exercise 1: Uxntal and Uxn

Part 2: Uxn interpreter in Python

The default Uxn system consists of an assembler (`uxnasm` or `drifblim.rom`) and a VM (`uxnemu` and `uxncli` on Linux/Macos; `Uxn32.exe` om Windows). The aim of this exercise is to create an interpreter, a Python program that will interpret Uxntal code and run it. Revisit your notes on how an assembler works, as you will need to follow the same general approach:

- First you tokenise the code, i.e. split the program text into meaningful tokens.
- In a first pass over the tokenised code, you look for all symbol declarations (i.e. labels for variables and subroutines) and map them to addresses
- In a second pass, you resolve all symbols, i.e. replace label references by addresses as well. At this point, there are no more labels in the code.
- In a final pass, you interpret token by token.

Simplifying assumptions

To make this easier, your interpreter only needs to support a subset of Uxntal, as follows:

- No use of the zero page, so no `LDZ` or `STZ`
- No use of `LDR` or `STR`
- No padding except for the `|0100` to mark the start of the main program
- No `BRK` except at the end of the main program
- Relative labels are only used for jumps, and jumps only use relative child labels, so `,&` and `&` only.
- Absolute labels refer either to code or to initialised (pre-populated) arrays
- No explicit use of `LIT`, so only the `#` syntax
- You can skip `SFT`, `EOR`, `AND`, `ORA` and `DEI`; I provide the stack operations `DUP`, `OVR`, `SWP` and `NIP`
- No parentheses inside comments

Python reference implementation and Uxntal test cases

I provide the starting point code for the interpreter for this exercise, as well as a set of 16 test cases. Together, this serves as the functional specification.

Your task

Your task is to create a working interpreter that will ideally pass all 16 tests.

Marking scheme

- [2 marks] Identifying information (in the form of comments at the beginning of the program). The first comments identify the program, giving your name and student ID, and saying what the program does. These may be the easiest marks you'll ever get!

- [10 marks] Your status report. State clearly whether the program works. If parts are not working, say so.
- [5 marks] Your interpreter correctly runs a non-trivial Uxntal program of your choice, with at least 10 instructions that are not `LIT`.
- [32 marks] All tests pass. A test bench with 16 tests will be provided

What to submit

Please submit a zip file containing the Python script `uxntal-interpreter.py`, the "non-trivial Uxntal program of your choice" mentioned above, and a `README` file which contains your brief status report (aim for 500 words) as a text file.

The name of the zip file should be `CANS2024-AE1-partB-`*your-student-id*`*.zip`.

Due date

19 April 2024 via the GA Workbook submission link