Shab-bot x16® Instruction Set & Documentation

About the Processor:
The Shab-bot® Core-Hexa™ CPU and Shab-bot x16® Instruction Set has an **instruction length of 16 bits**, 8 of which are dedicated to the immediate number; therefore, any number can be **added** and **subtracted** between 0 and 255 (or 0x0 and 0xff). Furthermore, the CPU has **4 general-purpose registers,** each 8 bits in size. Referring to these registers is done by the assembler; specifying a register can be any single character followed by the ID of the register, which ranges from 0 to 3 (inclusive). The Shab-bot® Core-Hexa™ CPU also supports fundamental **loading (LDR)** from, and **storing (STR)** to a 256-byte memory. Further documentation on the general format of the instructions, as well as details regarding their binary encodings can be found below.

Binary Encoding & General Formatting:

Binary encoding:

00000000-00-0-0-00-00

Bits [0, 1]: Determine which register to read from (1, 2, 3, or 4).

- 2 Bits are necessary to address 4 unique registers.

Bits [2, 3]: Determine which register to write to (1, 2, 3, or 4).

- 2 Bits are necessary to address 4 unique registers.

Bits [4]: Determine if writing to the register is necessary (ADD, SUB, LDR).

- Only one bit is necessary to determine whether the instruction calls for writing to the register or not, as it is a boolean value (True/False => 1/0).
    - 0 to disable writing to registers
    - 1 to enable writing to registers

Bits [5]: Determine whether Add or Subtract will be used

- Only one bit is necessary to determine whether the instruction will use addition or subtraction, as it is a boolean value (True/False => 1/0).

- 0 for Add
- 1 for Subtract

Bits [6, 7]: Determine whether LDR or STR is being used.

- Two bits are necessary for determining whether the instruction is calling for Load or Store to be used. For the reason why, please look below at the binary encoding(s):
  - 00 = Nothing
  - 01 = Write to memory (STR)
  - 10 = Read from memory (LDR)
  - 11 = Invalid

Bits [8, 16]: Specify the immediate number (0 to 255 (0x0 to 0xff)

- 8 Bits are necessary to reference every address of the 256 byte memory, as well as being "left-over" bits that contribute to supporting a larger immediate number.
- Using the Immediate Number:
  - ADD/SUB Immediate number to be added/subtracted to/from the specified register.
  - LDR/STR Immediate number acts as the address of where to store to/load from the memory.

About the Assembler & How to Use it:

1) Syntax Notation:

INSTRUCTION Rw, Re, Imm

Rw = Register-Write – Specify the register that is being written to.

Rw ∈ {X0, X1, X2, X3}

Re = Register-Read – Specify the register that is being read from.

Re ∈ {X0, X1, X2, X3}

Imm = Immediate Number — Specify the immediate number to be added/subtracted, or to be used as the offset for memory.

Imm ∈ [0, 255] ∈ W

M = Byte Addressed Memory Array — M[Imm] specifies the address in memory being accessed.

2) Instructions & Syntax
   a) ADD Instruction:
      ADD Rw, Re, Imm Adds Imm and Re, and places the result in Rw.
   b) SUB Instruction:
      SUB Rw, Re, Imm Subtracts Imm and Re, and places the result in Rw.
   c) STR Instruction:
      STR Re, Imm Stores the value from the specified register to M[Imm].
   d) LDR Instruction
      LDR Rw, Imm Loads the value from the M[Imm] to the specified register.

3) How to Use It:
   Once it has been decided which instruction(s) the CPU should execute, place each instruction in a text file, one line per instruction, with no newline/extraneous characters at the bottom. Then, run the assembler and specify the name of the file/the file path as user input. If every instruction is valid at every level, an image file will be produced with the name "output.txt", containing the usable instructions for Shab-bot® Core-Hexa™.

4) Example Instructions in a text file:
   ADD X0 X0 16 // Performs the operation: Register 0 = Register 0 + 16
   SUB X0 X0 12// Performs the operation: Register 0 = Register 0 - 12
   STR X0 0// Stores the value in Register 0 to M[0].
   LDR X3 0 // Loads the value from M[0] to register 3.
5) Additional Assembler Notes:
   - Python 3.11 was the only version of Python 3 that was tested and QA'd. Other Python versions, (e.g., 3.10, 3.12, et cetera) are not guaranteed to work.
   - The Assembler file will only run if named "main.py"
   - No imports or external dependencies nor libraries are used in the Assembler.