Documentation - Sourcery

Sanjay Sankaran

Contents

Sourcery

Sourcery is an Assembly inspired Interpreted language, where the source code of the program is stored in program memory. for example,

```
01 << $00
```

This is the code to print out a value from address `00` (the first byte is ignored here). since the source code is stored in program memory, the value at `00` is `01`, since it is the `0`th byte.

Memory

While Sourcery is an Assembly inspired language, it also takes advantage of being an interpreter.

Each "byte" of a program can hold a value from `-0x3ff` to `0x3ff`. All of the source code is typed in hexadecimal notation, except for a few instructions which have aliases, such as `<<`, `++` and `0?`.

The `$` acts as a `"valueof"` operator, similar to unary `*` in C/C++. Thus, `$x` represents the value that is stored at the location `x`. In the example form the previous section, we can see that `$00` is printed to the screen, with a value of `1`.

Also, the program counter is stored in the `0`th byte. Upon loading a program, the program counter is set to the first byte of the program - for example, if the first byte of a program was `A7`, execution will start from address `0xA7`. The program counter is not read-only, so writing a value to `$00` will case a jump to the written value.

Since Sourcery does not use variables with variable names, specific memory addresses are used to store and access values in a program. for example, instead of declaring a variable `i`, you could just use `$05`

or any other address of your choosing.

Instructions

So far, the instructions are as follows:

`<< x` , Prints out the value at `$x` in hex.

`<- x` , Prints out the value at `$x` as an ascii character.

`++ x y` , Increments the value at `$x` by `y`.

`-- x y` , Decrements the value at `$x` by `y`.

`== x y` , Sets the value at `$x` to `y`.

`0? x y` , if the value at `$x` is `0`, set it to `y`. else set it to `0`.

Macros

The macro system Currently uses the following:

`{LABEL something}` sets a label wherever it is declared. the position of this label can be acessed using `{$something}`

Hello World

```
03 {$START} 00

<- $$01
++ 01 01

== 02 {$END}
```

```
-- 02 $01

0? 02 -4
++ 02 03
== 00 $02

{LABEL START} {TEXT Hello World} 0a {LABEL END}
```