SaHS Documentation

Table of Contents

Introduction

This documentation will cover the configuration and examples of what the application can do and how to change setting within the application to create custom scenarios, for lab or personal usage.

Main Screen

When you load up the application you will be shown the main screen, this will be the screen that you will add/delete instructions, add/delete registers and run the program. With also details of what is currently running and register information. As seen in figure 1.

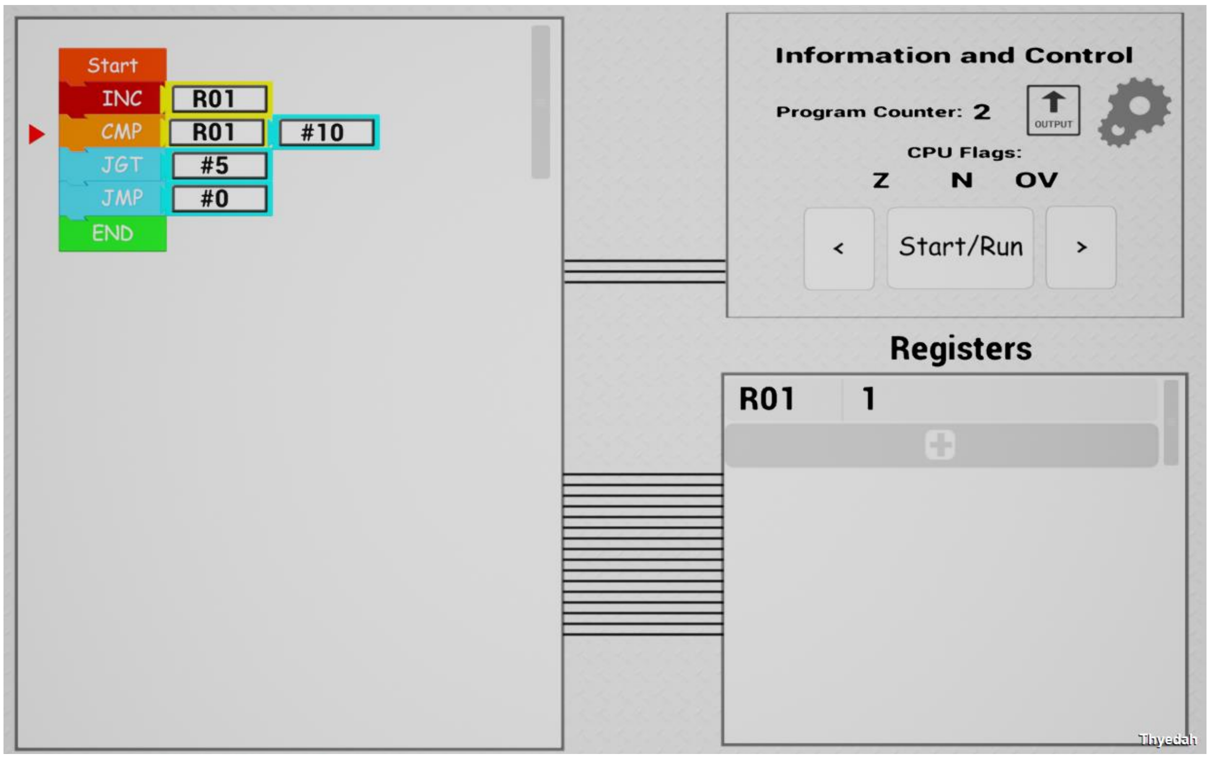


Figure 1 - Main Screen

In the example in figure 1 it currently has added a bunch of functions that are going to be executed. In this example it will increment register R01, then compare R01 with 10. If R01 is less than 10 the **N** will be set meaning negative and will loop until R01 is greater than 10. Every instruction will be explained in detail in this documentation.

On this screen you can also see the registers area and the Information and Control area, which will be explained later in this documentation.

Instructions Area

Here is the instruction area as in this example there are already 4 different types of instruction. Those with no attributes and those with 1 or more attributes, **the amount of maximum attributes is 3**.

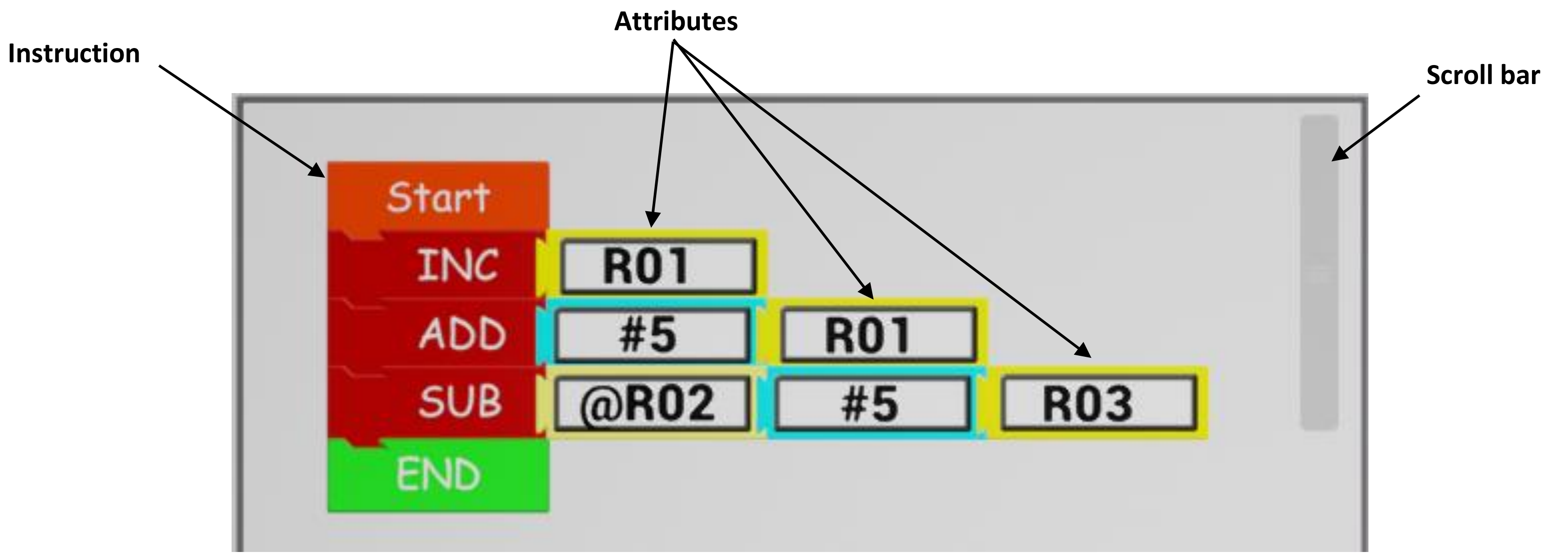


Figure 3

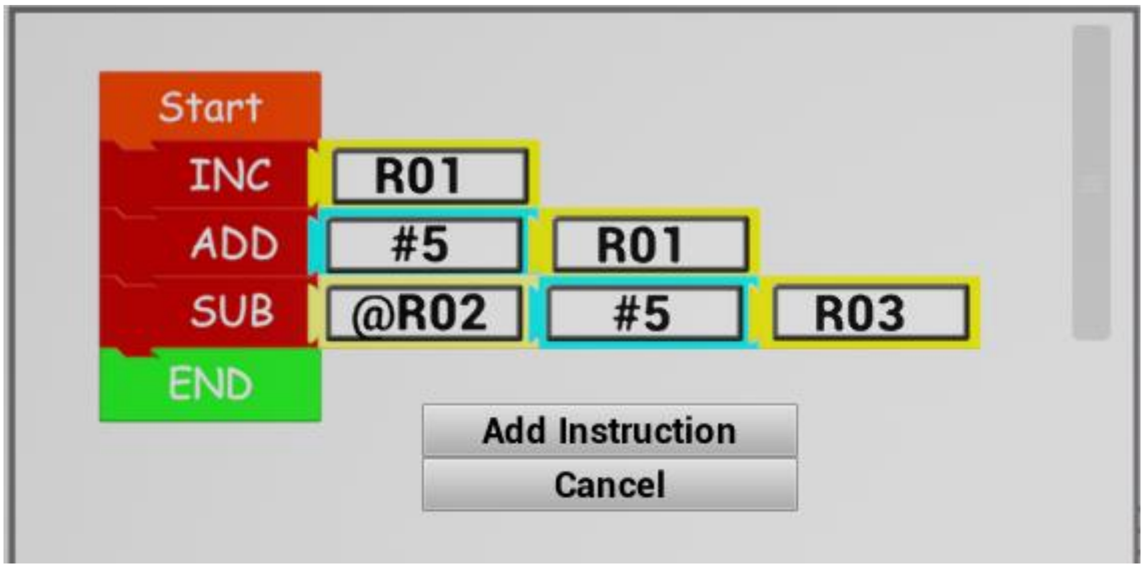


Figure 4

When you right click in the instructions area *not* the instructions themselves you will get the context menu to Add an instruction or to cancel.

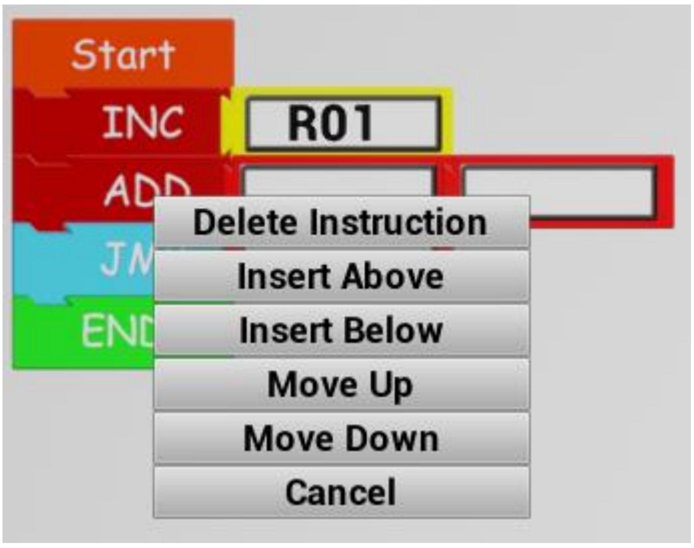


Figure 5

Here is the context menu for the instructions *not* the attributes, right clicking on the attributes will give you a different context menu. Here you are able to move, delete and insert above or below, the instruction.

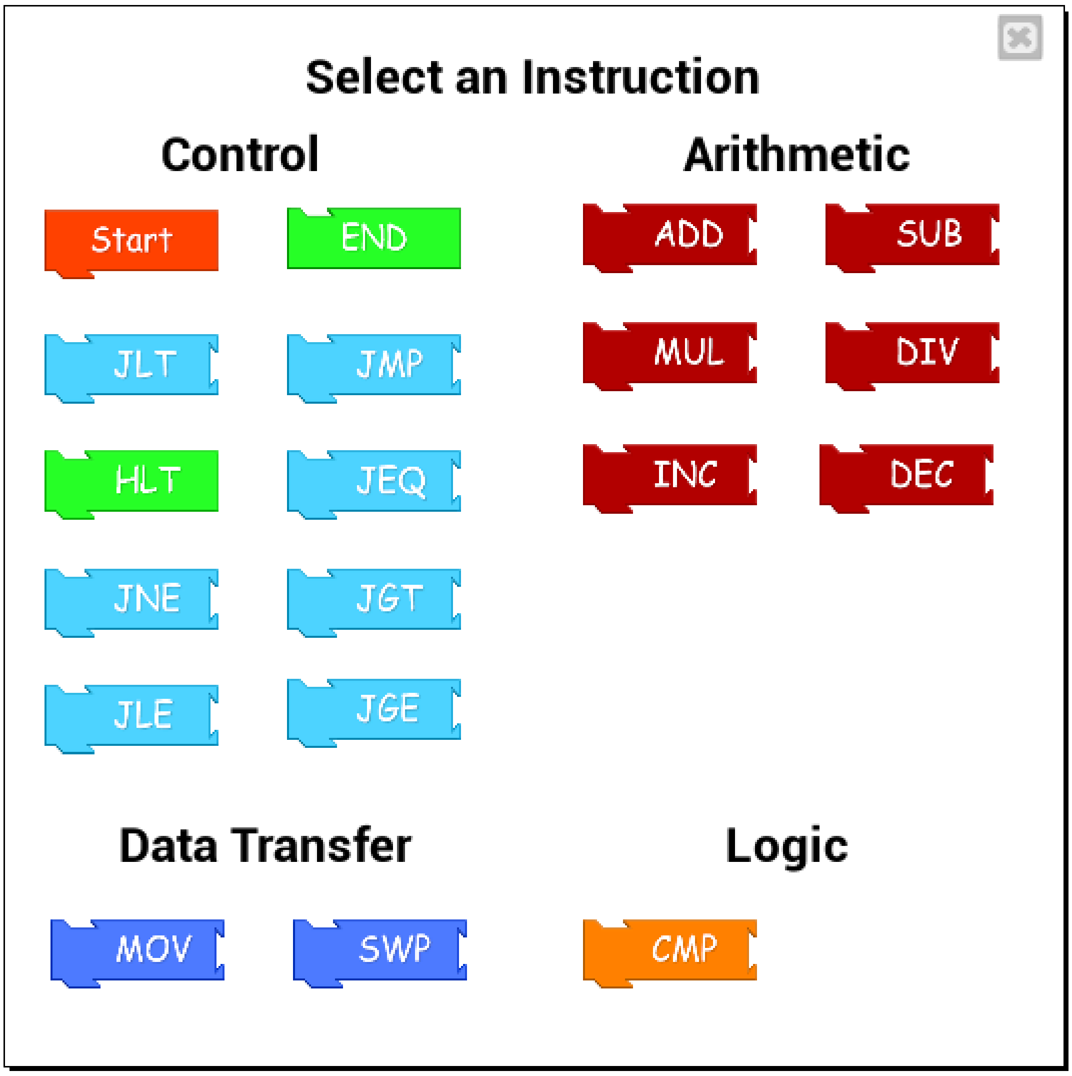


Figure 6

Here is the instructions menu, here you simple have to click on the instruction you want to add. If you do not know what each instruction does it is documented here. If you hover over the instruction you will be given a tip about the instruction.

Table 1

	No Attribute selected
	Register Value
	Indirect Register Value
	Numerical Value

Registers Area

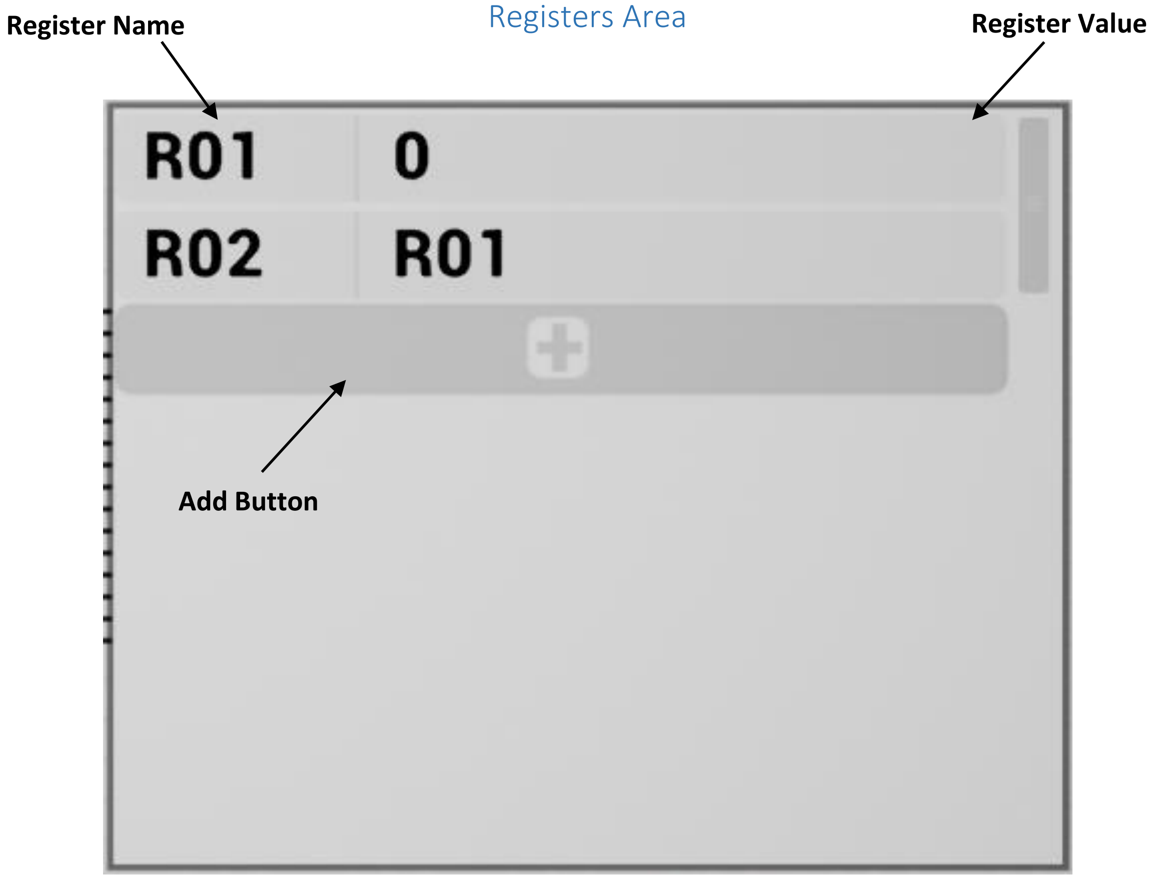


Figure 7

Above is the register's area. Here you can see the registers and their values. You can only have a maximum of 16 registers in SaHS, but you can also lower the max using the CPU settings.

Below is when you click and on the add button you will be give an option to make a register that holds a numerical value or a register that points to another register this is an indirect register.

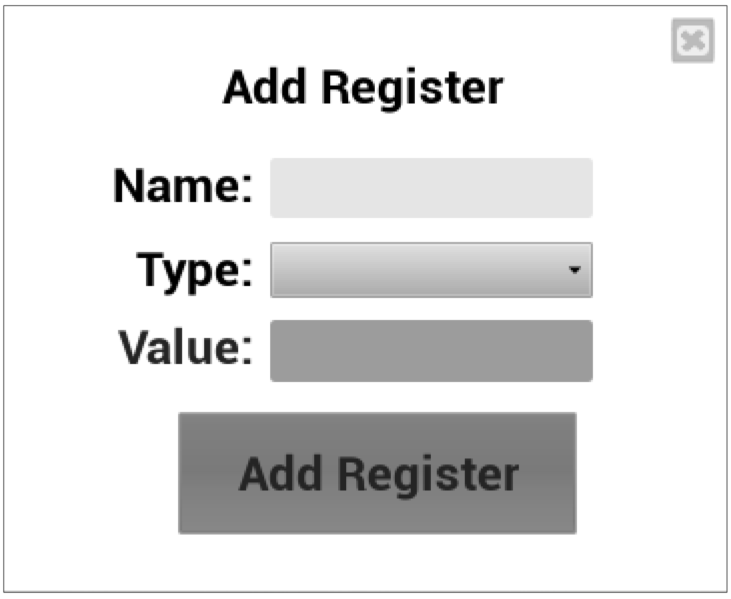


Figure 8

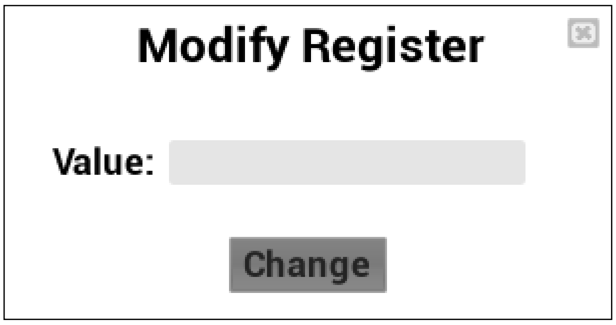


Figure 10

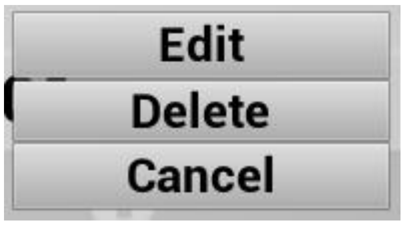


Figure 9

When right clicking on a register, you will be given the option to change the registers value, shown in figure 9. When you click on edit you will get the modify register menu this is figure 10.

Information and Control Area

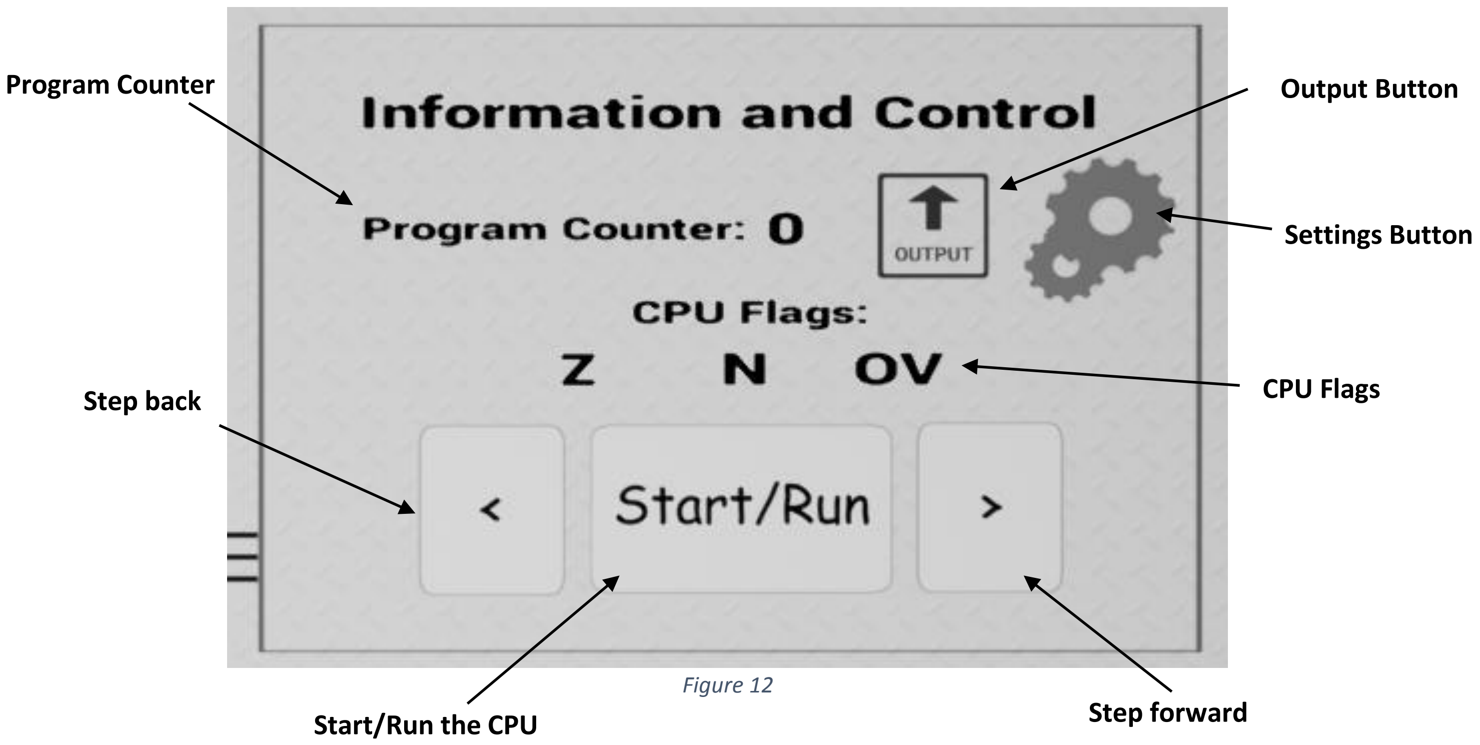


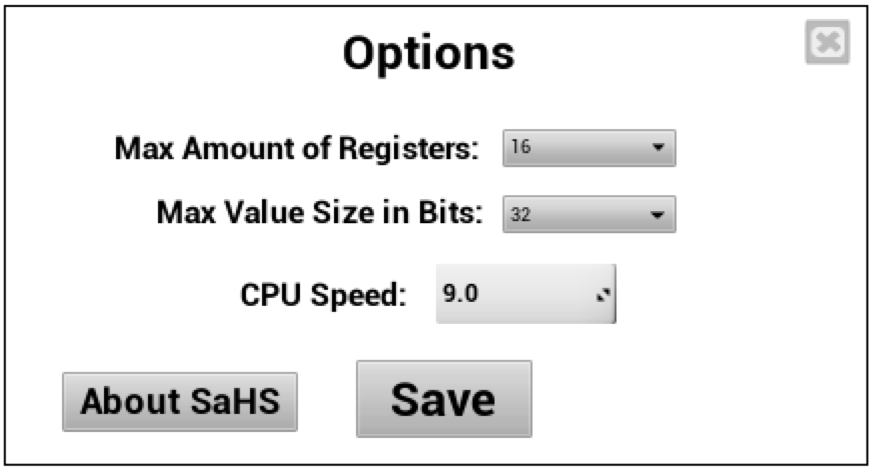
Figure 12

As shown above is the information and control area. Here you are able to start/run the CPU, step forward or backwards, show the output of all the code in CPU form and change the CPU settings.

You will also be able to see the current program counter and what flags have been set. You can also manually set flags by clicking on them.

Shown below is the CPU options, where you are able to customise the scenario to any need such as limit the max value size in bits, out of three options 32bits > 16bits > 8bits these are also signed numbers so you are able to use negative integers. You are also able to limit the max amount of registers and change the CPU speed.

There are also CPU flags Z (Zero), N (Negative) and OV (Overflow).



Options

Max Amount of Registers: 16

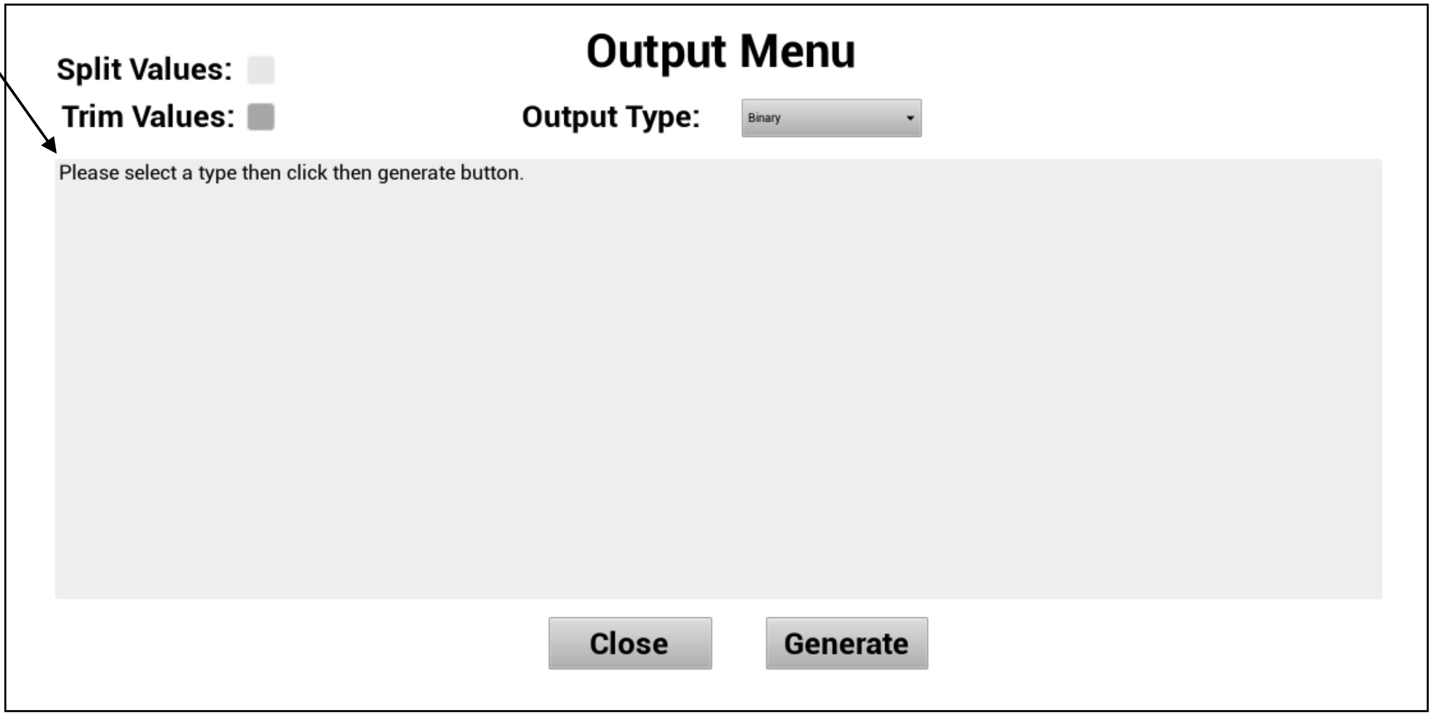
Max Value Size in Bits: 32

CPU Speed: 9.0

Figure 11

Output Menu

Output Text Area



Please select a type then click then generate button.

Figure 13

Above is the output menu, here the user is able to see the output of the code that they had created in the instructions area, into either Hex, Binary and Decimal values. There are also options to split the value into each instruction also each part of the instructions values and identifiers.

Trimming values is only available when split value is checked and the output type is binary as when outputting in binary there will be a lot of unused zeros that will quickly fill the textbox as it the CPU depending on the scenario uses 32bit values. So the option to trim makes it easier to read.

One selected you then click generate to create an output which you can copy to clipboard if needed.

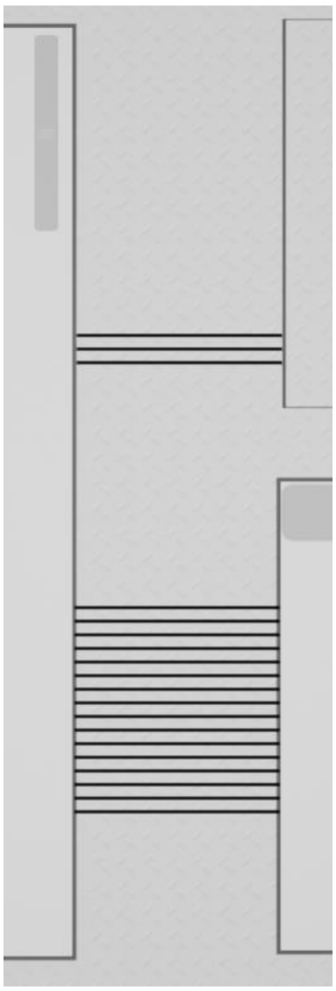


Figure 14

As shown if figure 14, there are connectors that connect the instructions are to the registers and information and control area.

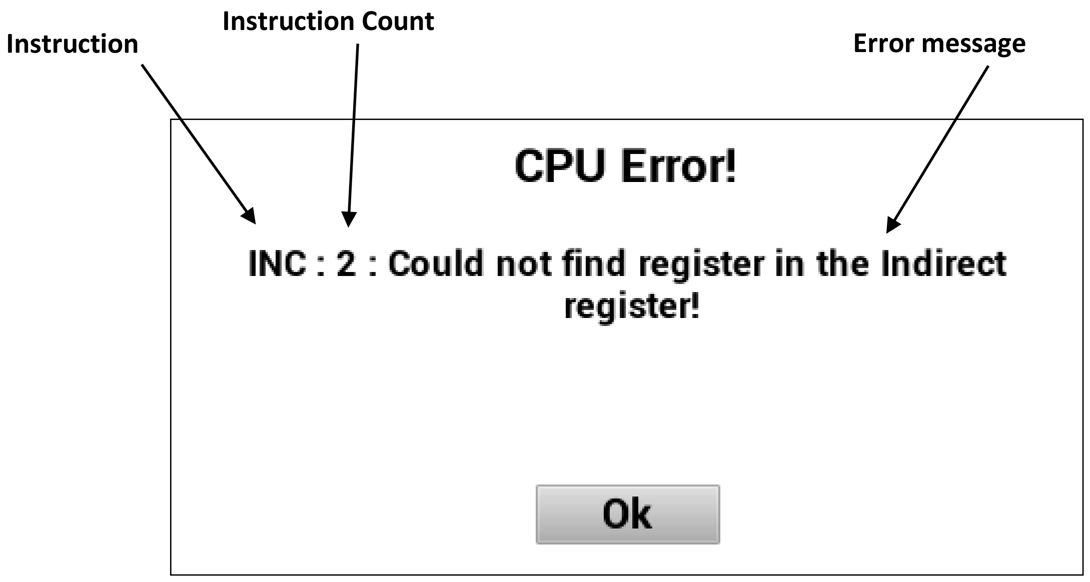
These indicate read and write for each instruction from top to bottom, so the top line connecting the registers area is the first register going down until the last register which will be 16.

The connectors connecting to the information area are read and writes for the CPU flags, starting at the top being the Z, N and OV flags.

CPU Errors

When adding instructions or registers, sometimes if you have not done it correctly or the way you have used an instruction is invalid then you will be thrown a CPU error when you go to the run the program. When an error occurs an error message will be shown telling you what went wrong and where to help you diagnose the problem.

As show in figure 15 is an example of an CPU Error explaining that an indirect register cannot find the register that it is pointed to.



CPU Error!

INC : 2 : Could not find register in the Indirect register!

Figure 15

Instructions

There are different types of instructions that do a range of logical, arithmetic and control functions in SaHS. There are 4 types of instructions; Data transfer, Logical, Arithmetic and Control. There is no limit to what instructions you can use apart from these rule when making a program:

- Programs must start with a Start Instruction
- Programs must end with an End Instruction
- You cannot add anything after the End Instruction only above it

Control Instructions

Control instructions are in charge of controlling how the program runs and what instruction is should execute next, for example the JMP (Jump) instruction will tell the program to jump to the entered address unconditionally.

Start	The Start instruction marks where the program should start and will be the first instruction to be executed and have Program counter of 0.
END	This marks the end of the program and will stop the CPU from running.
JLT	Jump Less Than. Only jump when the flags N is set. The attribute value must be the value of where to jump to in the program for example 0 would be the start.
JMP	Jump unconditionally
JEQ	Jump Equal. Jump if the Z flag is set.
JNE	Jump Not Equal. Jump if the Z flag is not set.
JGT	Jump Greater Than. Jump if the Z and N flag are not set.
JLE	Jump Less Than or Equal. Jump if the N flag is set or if the Z flag is set.
JGE	Jump Greater Than or Equal. Jump if the N flag is not set or the Z flag is set.
HLT	Halt. This will pause the currently running program, until the user presses the Start/Run button again.

Arithmetic

The arithmetic instruction is in charge of add, subtracting and other mathematic functions that can be then used in the program for example to add 10 to a register.

ADD	Add two values together this can be numerical to register/indirect register or register to register. It has two attributes, whatever the second address is that is where the calculate will be stored.
SUB	Subtract> This has 3 attributes, the first two are the value and the third is the address to where the answer should be stored.
MUL	Multiply. Same as add, again with two attributes.
DIV	Divide. This functions the same as subtract with three attributes.
INC	Increment. It only as one attribute, this will increment a register by 1 every time it is executed.
DEC	Decrement. Same as increment but will take away 1 every time it is executed.

Data Transfer

Data transfer instruction can for example swap two values in a register or indirect register.

MOV	The Move instruction for example can move one value to another location, an example would be to move the value 10 to register r01.
SWP	Swap will swap two register around with one and other you can also use indirect register to swap the values of the register of where they are point at.

Logic

The logic instructions are in charge or logic function such as comparison.

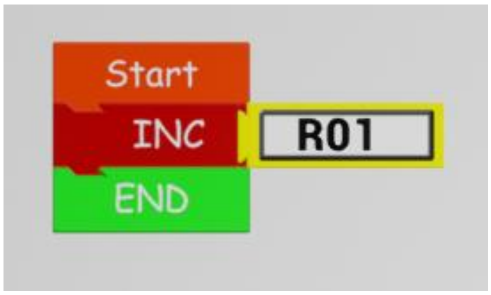
CMP	The compare instruction will compare the first value with the second value. **Take note of the orientation**. It will then set the CPU on the result of the comparison. If value is less then then the negative flag will be set.

The CPU

As the SaHS is a CPU simulator it is based on a simplified version of the Intel x86, instruction based CPU, with minor changes such as fixed length instructions instead of dynamic, no micro-processes, pipelining and also limited to 32bit size, instead of 64bit.

Understanding the Output Menu

As the output menu shows what the program will look like in binary, hex and decimal is it hard to understand if you do not know how the CPU decodes these random bits. So here is how to understand the output.



Here is a basic program, it is going to increment register R01 every time the program is run. R01 is the first register in this example.

In the output menu, if we generate binary that is Split and Trimmed, we get:

1|000||000||000|::1001|001|0|000||000|::10|000||000||000|

Which if we convert the second instruction it looks like this, also following the format.

5Bits	3Bits	32Bits	3Bits	2Bits	3Bits	2Bits
OPCODE	A-Mode	Address 1/ Value	A-Mode	Address 2/ Value	A-Mode	Address 3 / Value
9	1	0	0	0	0	0
INC	D	First Register	NULL	Ignored	NULL	Ignored

Table 2

Addressing Modes

Symbol	Function
IT	Intermediate Addressing
IN	Indirect Addressing
D	Direct Addressing

Table 3

Binary	Decimal	Hex	Address mode
000	0	0	None
001	1	1	Register Direct
010	2	2	Register Indirect
011	3	3	Memory Direct
100	4	4	Memory Indirect
101	5	5	Intermediate Addressing

Table 4

It is also the same format for Hexadecimal and Decimal. Following the same format as table 2.

CPU ID’s

For the CPU to understand what instruction to execute there are CPU ID’s for each instruction here is list of all ID’s to able be used to decode the output menu text.

Instruction	ID
START	1
END	2
MOV	3
CALL	4
ADD	5
SUB	6
MUL	7
DIV	8
INC	9
DEC	10
HLT	11
JMP	12
SWP	13
CMP	14
JEQ	15
JNE	16
JGT	17
JLT	18
JLE	19
JGE	20

Table 5

CPU Flags

There are three CPU flags that is used in the simulator, Z flag, N Flag and OV Flag. Zero Flag meaning the values compare are equal resulting in Zero. The Negative flag is set when the value compared is less than and when calculating results in the value being negative. The Overflow flag is set when the value in any of the register goes over the CPU maximum size, either resulting in a CPU Error or the value flipping to negative.

CPU Marker

The CPU marker shown in figure 16, shows that the instruction is going to be executed next *not* that the instruction has already been executed. So when stepping back and forward moving the marker, moving the mark on that instruction shows, that when running or resumed that instruction is next to be executed.

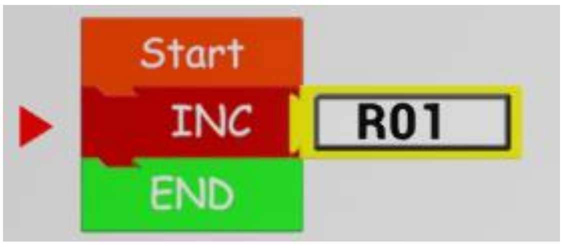


Figure 16