Samphire

Control Registers

- IP: The Instruction Pointer holds the address of the next instruction to be executed
- SP: The Stack Pointer
- SR: The Status Register indicates the status of the program. It's composed of a bunch of flags (each a single bit), which are set under different conditions, e.g. when there has been an interrupt.

Memory

Samphire has 256 RAM locations, each capable of holding 2 Hex digits (1 byte).

The red square represents the instruction pointer, and the blue square represents the stack pointer.

Addresses C0 to FF are mapped to the visual display unit, with the values interpreted as ASCII codes.

You write assemply code in the program, and then assemble it, which converts the code into hex and puts it into RAM, once it has performed the first checks of the code.

Sample Instructions

Move Instructions

```
mov al, 10
```

This code moves the value 10 (Hex) into the AL register. The Assembler converts this instruction into:

```
D0  00  10
```

In each case, the op code is first, followed by 2 operands.

This is an immediate move, because you are entering the value 10 by hand. The value is immediately available as part of the instruction.

There are other move instructions, distinguished by different op codes.

Arithmetic Instructions

```
add al, 01
```

This code adds the value 01 (Hex) to the contents of the AL register and leaves the result in the AL register.

There are other instructions for other arithmetic functions.

Control Flow Instructions

The unconditional jump instruction:

```
jmp label
```

"Label" refers to a label which is associated with another instruction in the program. The label defines where the jump goes to. This jump is unconditional because it does not check any conditions before jumping.

The Assembler translates this into:

```
C0  number
```

Where the number is the number of bytes between the jump instruction and hte label.

Writing a Value to Memory

A value can be placed into a memory location like this:

```
mov [addr], al
```

This will move the number pointed to by 'addr' into the location pointed to by AL.

In order to move a value into memory, the value must first be put into a register. The follow is not a legal instruction:

```
mov [addr], 10
```