Call-Return Mechanism

The call-return mechanism provides low-level implementation for subroutines[1].

Subroutines

We typically break problems down into smaller problems when solving them. If we have smaller problems that we solve many times, subroutines are useful to solve those smaller problems.

The *call* Instruction

This instruction does a number of things:

1. Transfers control to the called subroutine by placing the subroutine address into the instruction pointer.
2. Pushes the return address (the address of the instruction that follows the call instruction in memory) onto the stack.

Note: the stack is used because of the First In, Last Out nature it has, which allows us to call subroutines within other subroutines using the same mechanism.

The format of the call instruction in samphire is as follows:

```
call address
```

Where 'address' is the memory address of the start of the subroutine.

The *org* Instruction

In Samphire, the org instruction tells the assembler to place the following instructions into memory starting at a specific address.

We can also use the org instruction to put data into memory, which is a very useful application.

The format looks like this (in Samphire):

```
org address
```

Where address is the address at which the assembler should place the instructions.

This instruction can be used to place a subroutine into memory at a known address.

The *ret* Instruction

This instruction pops the return address from the stack into the Instruction Pointer.

We can use it to return to the main routine from a subroutine.

Example

```
mov al, 10
call 70             ; Subroutine call
mov [c0], al

org 70
add al, 1           ; The subroutine
ret

end
```

Subroutines with the Traffic Lights

Here's some code:

```
start:
mov al, 84
out 01              ; red, green
mov bl, ff
call 70

mov al, 48
out 01              ; orange, orange
mov bl, 10
call 70

mov al, 30
out 01              ; green, red
mov bl, ff
call 70

mov al 48
out 01              ; orange, orange
mov bl, 10
call 70

jmp start

org 70
in_delay:
dec bl
cmp bl, 00
jnz in_delay
ret

end
```

This code simulates a set of traffic lights, and the call function is used to add a delay at each point, with the length of delay specified by the value in bl.

---

1. A routine is basically an old-fashioned name for a program, making subroutines an old-fashioned name for functions or subprograms.↩