The myth of the Variable

What is a variable?

A variable is nothing more than an address. We use a logical name as a label to more comfortably refer to that location in memory containing data we are interested on.

The variable declaration in a program usually includes its type. This defines how many bytes are used starting from the address that has been associated with its name.

In the example for the DLX shown below, the Data segment in Memory is set to start at address 100. Then, the reference to “n” stands for 100, a data of type ***word*** and value 10. The address used for it goes from that address to 103, the 4 bytes contain a 32-bit integer that is the size associated with the "word" (a word is an integer and is also the size of the registers and many instructions deal with integers). The value 10 will be stored, in this case, in big-endian format. Note how this is shown in the first row of the following table:

Data Declaration	[Name]: Address : value	Shown in DLXVSim
`.data 100`	`n: 100 a 103 : 10`	`n: 10`
`n: .word 10`	`c: 104 a 107: 6`	`c: 6`
`c: .word 6,16,26,36`	`108 a 111: 16`	`c+0x4: 16`
`sum: .space 4`	`112 a 115: 26`	`c+0x8: 26`
`a: .double 10.4,2,3.7,8`	`116 a 119: 36`	`c+0xc: 36`
`cte: .float 3.1416, 1.6`	`sum: 120 a 123: 0`	`sum: 0`
	`a: 124 a 131: 10.4`	`a: 10.400000`
	`132 a 139: 2.0`	`a+0x8: 2.000000`
	`140 a 147: 3.7`	`a+0x10:3.700000`
	`148 a 155: 8.0`	`a+0x18:8.000000`
	`cte: 156 a 159: 3.14`	`cte: 3.141600`
	`160 a 163: 1.6`	`cte+0x4: 1.600000`

Following “n”, we have arranged a series of declarations of different sizes (4 bytes for integers (.word) and single precision floating point (.float), 8 bytes for double precision (.double) and “space” only reserve bytes). If there is more than one comma separated value, they are assigned in memory. For example, in the case of ***c***, each integer in the list is referred to as ***c+4***, ***c+8*** ... It is allowed to use ***n+4*** or ***c,*** it is the same address after all. Thus, ***n+8***, ***c+4*** and ***108*** reference the same memory location (108) and, therefore, may appear interchangeably in an assembler instruction.

For example:

```
lw r7, n+8(r0)    lw r7, c+4(r0)   y   lw r7, 108(r0)
```

they are the same and load the number 16 in r7 after accessing the memory position ***108+r0***. This is:

```
r7 <-- M(108)    then r0 always has 0 in the DLX
```

<u>**Note**</u>: Reserved words that begin with a period are assembly directives.
.data and **.text** are directives of the assembler to locate in that address the segments of data (variables) and text (program) in those directions. See the HELP tool.

Autor: Virginia Escuder. Dep Automática UAH