Como os dados e programas são armazenados

Podemos verificar o modo que um arquivo binário é armazenado usando a opção -l no montador. Exemplo: nasm -g -f elf64 exemplo1.asm ***-l exemplo1.lst***

Desse modo seria criado um arquivo de lista exemplo1.lst, contendo algumas informações de armazenamento e codigo binário. Mais detalhes sobre comando nasm, digite no terminal (shell): *nasm -h.*

Por exemplo, um fragmento da seção de dados do arquivo de lista, do programa de exemplo 1:

```
Linha    Endereço   Valor em hex
  36  00000009  40660301          dVar1     dd   17000000
  37  0000000D  40548900          dVar2     dd   9000000
  38  00000011  00000000          dResult   dd   0
```

Na primeira linha, o 36 é o número da linha. O próximo número, 0x00000009, é o endereço relativo na área de dados de onde essa variável será armazenada. Como *dVar1* é double word que requer quatro bytes, o endereço da próxima variável é 0x0000000D. A variável *dVar1* usa 4 bytes como endereços *0x00000009*, *0x0000000A*, *0x0000000B* e *0x0000000C*. O restante da linha é a declaração de dados digitada no arquivo de origem da linguagem de montagem original. O 0x40660301 é o valor, em hexadecimal, conforme colocado na memória. Os 17.000.000 são 0x01036640. Lembrando que **a arquitetura é little-endian**, o byte menos significativo (0x40) é colocado no endereço de memória mais baixo, com dois dígitos hexadecimais em cada byte. Como tal, o 0x40 é colocado no endereço 0x00000009, o próximo byte, 0x66, é colocado no endereço 0x0000000A e assim por diante.
Isso pode ser confuso, pois, à primeira vista, o número pode aparecer para trás ou distorcido (dependendo de como é visualizado). Separemos de dois em dois: 40.66.03.01, a seguir para interpretar o valor correto tomamos 01.03.66.40, ou seja, $0x01036640=17000000_d$.
De outro modo, vejamos a figura abaixo:

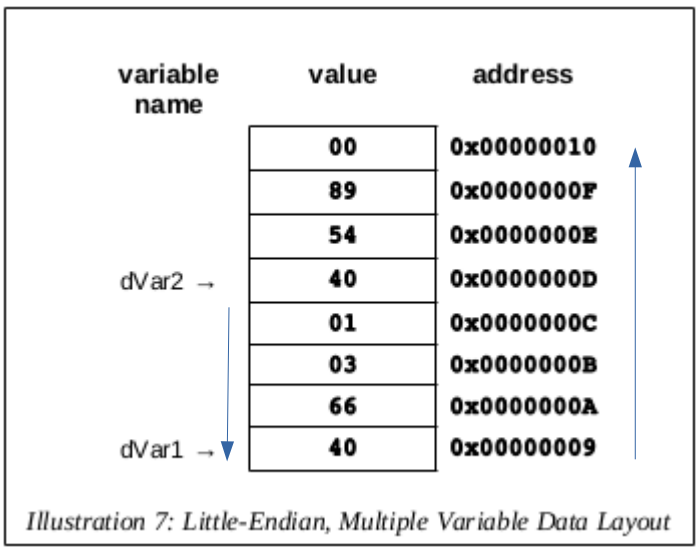

Illustration 7: Little-Endian, Multiple Variable Data Layout

Mais um exemplo, o fragmento da seção text do programa exemplo1.asm.

```
95                                  last:
96 0000005A 48C7C03C000000             mov     rax, SYS_exit
97 00000061 48C7C300000000             mov     rdi, EXIT_SUCCESS
98 00000068 0F05                       syscall
```

Novamente, os números à esquerda são os números das linhas. O próximo número, 0x0000005A, é o endereço relativo de onde a linha de código será colocada.
O próximo número, 0x48C7C03C000000, é a versão em linguagem de máquina da instrução, em hexadecimal, que a CPU lê e entende. O restante da linha é a instrução original da fonte de linguagem assembly.
O rótulo *last:*, não possui uma instrução em linguagem de máquina, pois o rótulo é usado para referenciar um endereço específico e não é uma instrução executável.