# 4º Relatório de Laboratório de Montagem

Bruno Cardoso Maciel RA: 141046279

Rafael Stefanini Carreira RA: 141040726

10/11/2015

---

## 1. Introdução

Podemos observar que as aplicações em Assembly muitas vezes se reutilizam sequências de instruções iguais, como por exemplo, escrever uma *string.* A partir desse fato, podemos utilizar um novo recurso conhecido como “macro”. Este recurso nos permite criar e utilizar trechos de código como funções de forma mais direta do que as convencionais chamadas “call”, isso se deve pela possibilidade de passarmos parâmetros para as macros. Um exemplo da criação de uma macro:

```
%macro NomeFunção NumParametros
;;corpo da função
%endmacro
```

Dentro da macro, nos referenciamos aos parâmetros pela sua ordem, da seguinte forma: “%NumParametro”. No corpo do código principal, a macro pode ser chamada através de seu nome e seus parâmetros separados por virgula. Um exemplo de uma chamada de uma macro, onde a variável “num” é uma cadeia de caracteres que representa um valor numérico e “num_L” é o tamanho desta variável:

```
imprime num, num_L
```

Através do uso de macros, os códigos podem ficar mais organizados e estruturados, pois evitamos a reescrita de estruturar iguais, o que torna a programação mais modular, ou seja, o código fica dividido em partes distintas, melhorando a legibilidade e facilitando sua manutenção.

## 2. Objetivo

Através do uso de macros e demais recursos aprendidos, escrever um programa que receba 2 matrizes 3x3 digitadas pelo usuário, realizar a soma das matrizes e retornar a matriz resultante para o usuário.

## 3. Metodologia

Primeiramente, notamos que algumas estruturas seriam utilizadas diversas vezes no código, como por exemplo, escrever mensagens na tela e ler os números da matriz. Dessa forma, implementamos duas estruturas de macro no início do código para tais sequências. Além das macros, também foram criadas funções para as demais funcionalidades do programa.

Foi proposto que cada valor da matriz ocuparia 4 *bytes* de memória, sendo necessário o armazenamento de 36 *bytes* para cada matriz.

Seguindo os macros e chamadas das funções obtemos a seguinte execução:

- Por meio de um macro, imprime a mensagem introdutória da aplicação.
    - Foram passados como parâmetro a variável e seu tamanho.

- Imprime uma mensagem solicitando a inserção de dos valores da primeira matriz.
    - Utilizamos a mesma macro anterior, mudando apenas os parâmetros.

- Armazena os valores da primeira matriz.
    - Para armazenarmos os valores da matriz, utilizamos o registrador ebx como a posição da matriz a qual o número digitado seria inserido.
    - Ao inserir um valor, o mesmo é lido por meio de um macro e é convertido de caractere para valor numérico através do processo de multiplicações por 10.
    - O registrador ebx é incrementado com o valor 4, pois cada valor ocupa 4 *bytes* de memória da matriz.
    - É realizado uma comparação para saber se a matriz já está completa, senão o processo é refeito até que a matriz esteja completa.

- Solicita ao usuário os valores da segunda matriz.
    - Novamente utilizando o macro de imprimir *strings,* é solicitado ao usuário que digite a segunda matriz.

- Armazena a segunda matriz.
    - Através do mesmo processo, cada valor da segunda matriz é lido e armazenado em sua posição correspondente.

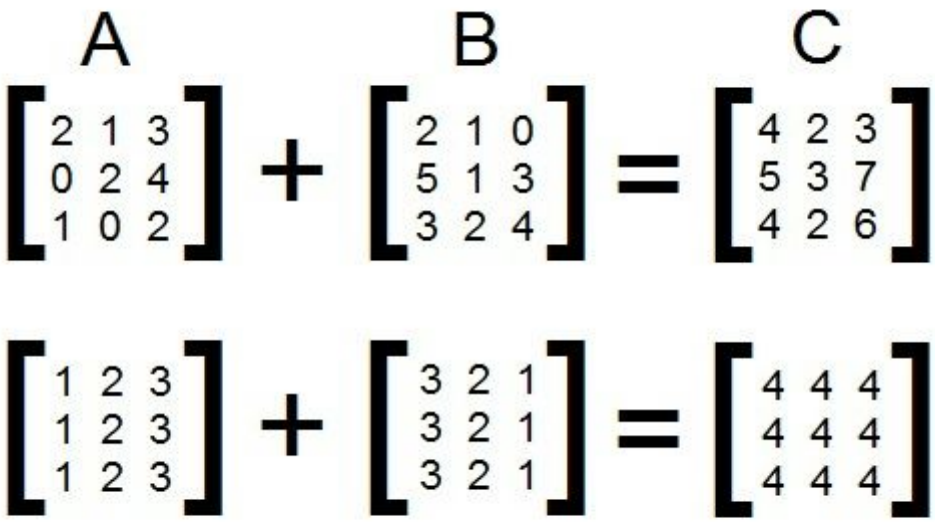
- Imprime na tela a mensagem sobre a matriz resultante da soma.
    - Através da macro de  impressão, a mensagem “A matriz resultante da soma é: “ é impressa na tela.

- Realiza a soma das matrizes e armazena em uma terceira matriz.
    - Para realizar a soma das matrizes digitadas pelo usuário, o registrador ebx foi usado para guardar a posição dos elementos a serem somados nas matrizes.
    - O valor marcado pela posição de ebx na primeira matriz é armazenado em eax.
    - Da mesma forma, o valor da segunda matriz é armazenado em edx.
    - Soma-se os valores de eax e edx, o valor resultante é armazenado na posição ebx da terceira matriz.
    - Esse mesmo valor é convertido para cadeia de caracteres e impresso na tela.
    - O contador ebx é incrementado em 4 *bytes* para a soma do proximo valor.
    - O processo continua até que todos os elementos da matriz sejam somados e impressos.
    - Para motivos estéticos, usou-se tambem o registrador ecx como um contador de tal forma que a cada 3 valores impressos, ocorre uma quebra de linha. Tal processo permite a melhor visualização da matriz resultante.

- Finaliza o programa.
    - Imprime uma linha indicando o fim da interação do usuário com o programa.
    - Finaliza a execução do programa através da *syscall exit.*

## 4. Resultados

Alguns resultados obtidos com o programa.




## 5. Discussão

Trabalhar com matrizes torna a aplicação mais complexa e extensa, seja pela quantidade de operações de leituras e pela complexidade das demais funções utilizadas. Dessa forma, o uso de macros pode nos ajudar quanto a organização do código, podendo tornar o código mais legível e facilitar futuras manutenções. Por meio da passagem de parâmetros, obtemos um funcionamento ainda mais parecido com as funções, presentes em linguagens de alto nível. Assim, podemos reutilizar um macro diversas vezes, o que evita repetição no código, tornando assim menos trabalhoso desenvolver o programa.

## 6. Conclusão

Conclui-se que uso de macros torna o desenvolvimento de uma aplicação em *Assembly* mais prático, principalmente quando é necessário o uso repetitivo de determinada sequência de instruções. Somado a isso, os recursos aprendidos até então, possibilitam que o código construído seja cada vez mais organizado e de fácil compreensão.

Diante de objetivo proposto, desenvolvemos a aplicação de forma que os resultados obtidos por meio dessas técnicas foram satisfatórios.