# Busca sequencial em assembly

**Guilherme Lage Albano, Guilherme Kaio Parreira Caldeira,**
**Kassio Rodrigues Ferreira, Jonas Elias Santos Kretli**

Universidade Federal de Ouro Preto (UFOP)

Rua 36, 115 - Loanda, João Monlevade - MG, 35931-008– João Monlevade – MG – Brazil

Departamento de Sistemas e Computação

**Abstract.** This meta-paper aims to explain the working of the sequential search algorithm built in assembly language, describing the code lines and the functions used.

**Resumo.** Este meta-artigo tem o objetivo de explicar o funcionamento do algoritmo de busca sequencial feito da linguagem Assembly, descrevendo as linhas de código e as funções utilizadas.

## 1. Introdução

O objetivo da implementação do algoritmo é o estudo da linguagem assembly para a disciplina de Fundamentos de arquitetura de dados do terceiro período do curso de sistemas de informação do campus ICEA, Universidade Federal de Ouro Preto. Foi utilizado a IDE Mars

## 2. Algoritmo

O algoritmo de busca sequencial realiza a busca de um valor inserido pelo usuário em um vetor analisando cada posição até encontrar a que está sendo buscada, a função retorna a posição do valor procurado ou caso não encontre em nenhuma posição o algoritmo retorna a mensagem ao usuário que o valor inserido não se encontra no vetor.

## 3. Implementação

A implementação do código foi dividida em quatro etapas: Leitura e armazenamento dos valores inseridos pelo usuário no vetor; leitura do valor a ser buscado no vetor; o algoritmo de busca sequencial; retorno para o usuário da posição em que o valor se encontra, caso ele exista no vetor, caso contrário retorna uma mensagem informando que o valor se encontra no vetor.

### 3.1 Cabeçalho

No cabeçalho (.data) do programa a label *array1* reserva 40 bytes para representar um vetor de 10 posições. As labels restantes são utilizadas para o feedback com o usuário.

```
.data
        array1: .space 40       #declara um array de 10 posições
        msg_inicial: .asciiz "\t ----> GRUPO 08 - Busca Sequencial <-----\n\n"
        msg_leitura: .asciiz " Insira os valores do vetor\n"
        msg_busca: .asciiz " Insira o valor a ser buscado no vetor: --- \n\n"
        msg_aux: .asciiz " valor: "
        msg_encontrado: .asciiz "\n\n Valor encontrado: "
        msg_posicao: .asciiz "\n Valor alocado na posição: "
        msg_naoEncontrado: .asciiz "\n\n Valor não encontrado... \n "


        msg_resultado: .asciiz "\n Resultado da busca:"
```

### 3.2 Main

A seção Main  contém a sequência de execução do programa. A princípio é exibida a mensagem inicial, e em seguida o procedimento leitura é chamado, uma vez que o procedimento termina  a execução os valores já estão armazenados em memória e o programa segue pedindo ao usuário para inserir o valor a ser buscado, o valor inserido é armazenado no registrador a0 e o procedimento busca Sequencial é chamado e ao fim de sua execução o programa é encerrado.

```
        li $v0, 4
        la $a0, msg_inicial  #imprime msg inicial
        syscall

        jal leitura

        li $v0, 4
        la $a0, msg_busca   #pede ao usuário para inserir o valor a ser buscado no vetor
        syscall

        li $v0, 5  #ler o valor inserido pelo usuário
        syscall
        move $a1, $v0

        jal buscaSequencial

        #encerra o programa
        li $v0, 10
        syscall
```

### 3.3 Subrotina  leitura

A sub rotina a seguir realiza a leitura dos dados inseridos pelo usuário e os armazena no vetor.
O primeiro passo é reservar as variáveis de controle t0 e t1 e então apontamos s0 para o início do vetor, após isso entramos no laço de repetição que ocorre 10 vezes e em cada repetição é pedido para que o usuário insira um valor a ser inserido na posição atual do array, posição que é indicada ao usuário a cada repetição. Após a inserção do valor pelo usuário, este é armazenado no vetor e então apontamos para a próxima posição.

```
leitura:
        #subrotina para ler os dados e armazenar no vetor

        li $v0, 4
        la $a0, msg_leitura   #imprime msg para o usuário
        syscall

        li $t0, 1    #variável de controle do while
        li $t1, 10   #variável de controle do while
        addi $s0, $zero, 0   #endereço base do vetor

        while1:
                bgt $t0, $t1, exit1  #enquanto t0 < t1

                # indica a posição do vetor para o usuário
                li $v0, 1
                move $a0, $t0
                syscall

                li $v0, 4
                la $a0, msg_aux
                syscall

                li $v0, 5  #lê um inteiro
                syscall

                move $s3, $v0  #guarda o valor armazenado

                sw $s3, array1($s0)   #armazena o inteiro no vetor
                addi $s0, $s0, 4   #aponta $s0 para a próxima posição do vetor
                addi $t0, $t0, 1  #atualiza variável de controle do while

                j while1

        exit1:
                jr $ra #retorna para onde a subrotina foi chamada
```

### 3.4 Subrotina  busca Sequencial

Após preenchido o vetor, a busca sequencial realiza a busca o valor requisitado pelo usuário, a busca sequencial é simples e é a forma de procura mais básica, o algoritmo analisa cada posição no array até encontrar o valor ou até analisar todos e não encontrar, no código a seguir realiza-se um laço que repete 10 vezes, que é o número de posições no vetor em questão e a condição de parada é se caso ele encontre o valor ou se após verificar todas as posições ele não encontrar o valor em nenhuma posição, nesse último caso imprime uma mensagem ao usuário dizendo que o valor procurado não existe no vetor.

```
buscaSequencial:
        #subrotina buscaSequencial
        li $v0, 4
        la $a0, msg_resultado
        syscall

        addi $t0, $zero, 0    #variável de controle do while
        addi $s0, $zero, 0   #endereço base do vetor

        while2:
                bgt $t0, $t1, exit2  #enquanto t0 < t1
                lw $s3, array1($s0)

                beq $a1, $s3, encontrado  #compara o valor buscado pelo usuário com o valor recuperado do vetor

                addi $s0, $s0, 4
                addi $t0, $t0, 1

                j while2

        exit2:
                li $v0, 4
                la $a0, msg_naoEncontrado
                syscall
                jr $ra
```

### 3.5 Subrotina encontrado

A Subrotina “encontrado” é responsável por retornar ao usuário a posição na qual se encontra o valor buscado. Como cada posição do vetor é referenciada por 4 bytes é necessário fazer a divisão da posição retornada por 4 e assim obter o índice do vetor. Por fim, a sub rotina imprime a posição na qual o valor se encontra e a execução retorna para a seção main.

```
encontrado:
        #retorna ao usuário que o valor foi encontrado
        li $v0, 4
        la $a0, msg_encontrado
        syscall

        li $v0, 1
        move $a0, $a1   #valor buscado
        syscall

        #retorna a posição em que o valor se encontra no vetor
        li $v0, 4
        la $a0, msg_posicao
        syscall

        #realiza a divisão do valor contido em $s0 por 4 para encontrar a posição do vetor
        li $s6, 4

        div $s0, $s6
        mflo $s0
        addi $s0, $s0, 1  #soma 1 ao valor posição para exibir de 1-10 posições ao invés de 0-9

        #imprime a posição do valor encontrado
        li $v0, 1
        move $a0, $s0
        syscall
        jr $ra #retorna para onde a subrotina foi chamada
```

## 4. Conclusão

Através do trabalho apresentado acima, podemos observar na prática como é trabalhar com a linguagem assembly e como ela pode ser usada para implementar uma busca sequencial. Dessa forma utilizamos do conhecimento adquirido através das aulas e elaboramos o código a fim de produzir um trabalho executando a busca sequencial desejada.