Aluno: Rômulo de Vasconcelos Feijão Filho - Matrícula: 14/0031260

Trabalho 3 de OAC - 2/2018

• *Objetivo*

Este trabalho objetiva o desenvolvimento de funções em assembler MIPS para realizar operações sobre árvores binárias de busca. As operações devem ser implementadas com algoritmos recursivos.

• *Descrição*

Árvores binárias de busca são árvores binárias onde os dados estão ordenados, de forma que todos os nós à esquerda de um nó raiz tem valores menores que este, e todos os nós à direita de um nó raiz tem valores maiores. Um exemplo de ABB é apresentado a seguir.
Um nó é representado no MARS por 3 palavras: (dado, esq, dir), onde dado é o valor contido no nó, esq e dir são os endereços dos nós à esquerda e à direita, respectivamente.
No MARS, uma árvore binária pode ser criada diretamente com labels indicando os endereços dos nós. Por exemplo, a árvore ilustrada acima poderia ser representada como segue.

```
.data
raiz: .word 8 a0 a1
a0: .word 3 a2 a3
a1: .word 10 0 a4
a2: .word 1 0 0
a3: .word 6 a5 a6
a4: .word 14 a7 0
a5: .word 4 0 0
a6: .word 7 0 0
a7: .word 13 0 0
```

Um novo nó pode ser alocado reservando-se 3 palavras adjacentes na memória. O registrador $gp do MIPS localiza o início da área de dados estáticos na memória, que pode ser utilizado neste trabalho como heap, ou seja, área para alocação dinâmica de nós da árvore. Alocar um nó utilizando o ponteiro $gp consiste em retornar o endereço atual de $gp e depois incrementá-lo de 12, para apontar para a próxima palavra livre na memória.

ATENÇÃO: No 4° passo na execução das funções funcao_exec_busca_recursiva e funcao_exec_insere_rec, deve-se inserir um valor no console.

OPERAÇÕES

inserção recursiva:

```
void insere_rec(nodo raiz, int valor) {
        if (raiz == NULL) {
        raiz = malloc(sizeof(nodo));
        raiz->dado = valor;
        raiz->esq = NULL;
        raiz->dir = NULL;
        return raiz;
 }
        if (valor < raiz->dado)
                return insere_rec(raiz->esq, valor);
        if (valor > raiz->dado)
                return insere_rec(raiz->dir, valor);
        return raiz;
}
```

busca recursiva:

```
        nodo busca_rec(nodo raiz, int valor) {
                if (raiz == NULL) return NULL;
                if (valor == raiz->dado)
                        return raiz;
                if (valor > raiz->dado)
                        return busca_rec(raiz->dir, valor);
                else
                        return busca_rec(raiz->esq, valor);
}
```

De forma similar, implementar também a função size_rec(), que retorna o número de nós da
árvore.

• *Funções*

Funções para execução do size_rec:

```
funcao_exec_size:                   #função usada para usar somente a função
                                    size, no final ela irá printar o tamanho da
                                    árvore
        addi $a0, $zero, 8192
        jal size_rec
        addi $v0, $zero, 1
        add $a0, $zero, $a1
        syscall
        addi $v0, $zero, 10
        syscall

 size_rec:
        lw $t0, 0($a0)              #busca valor do nó guardado no .data
        beqz $t0, retorna_endereco  #se valor for igual a 0, então já
                                    passamos por todos os nós, então
                                    retornamos ao $ra
        addi $a1, $a1, 1            #se for diferente de 0, incrementa 1 ao
                                    registrador a1, que será usado para
                                    guardar o tamanho
        addi $a0, $a0, 12           #busca próximo nó no .data
```

Funções para execução do insere_rec:

```
funcao_exec_insere_rec:             #caso for usar a função de inserção, deve-se chamar a
                                    funcao_exec_insere_rec
        addi $a0, $zero, 8192
        addi $v0, $zero, 5
        syscall                     #pede ao usuário valor a ser inserido
        jal size_rec
        addi $t0, $zero, 12
        mul $t8, $t0, $a1
        addi $t9, $t8, 8192
        sw $v0, 0($t9)
        addi $a0, $zero, 8192
        add $s1, $zero, $ra
        jal insere_rec
        addi $v0, $zero, 10
        syscall
```

```
insere_rec:
        addi $sp, $sp, -12
        sw $ra, 0($sp)
        add $a1, $zero, $a0
        lw $t0, 0($a0)
        addi $a0, $a0, 4
        lw $t1, 0($a0)
        beqz $t1, retorna_fim
        sw $t1, 4($sp)
        addi $a0, $a0, 4
        lw $t2, 0($a0)
        beqz $t2, retorna_fim
        sw $t2, 8($sp)
        bgt $v0, $t0, vai_direita
        blt $v0, $t0, vai_esquerda

retorna_fim:                    #insere o endereço do novo nó na árvore e encerra a
                                execução
        sw $t9, 0($a0)
        jr $s1

vai_direita:                    #vai para o nó da direita da árvore para a função
                                insere_rec
        lw $a0, 8($sp)
        addi $sp, $sp, 8
        j insere_rec

vai_esquerda:                   #vai para o nó da esquerda da árvore para a função
                                insere_rec
        lw $a0, 4($sp)
        addi $sp, $sp, 8
        j insere_rec
```

Funções para execução da busca_recursiva:

```
funcao_exec_busca_recursiva:        #caso for usar a função de busca recursiva, deve-se
                                    chamar a funcao_exec_busca_recursiva
        addi $a0, $zero, 8192
        addi $v0, $zero, 5
        syscall                     #pede ao usuário valor a ser procurado
        add $s0, $zero, $v0
        jal busca_recursiva
        addi $v0, $zero, 1
        add $a0, $zero, $a1
        syscall
        addi $v0, $zero, 10
        syscall

busca_recursiva:
        addi $sp, $sp, -12
        sw $ra, 0($sp)
        add $a1, $zero, $a0
        lw $t0, 0($a0)
        addi $a0, $a0, 4
        lw $t1, 0($a0)
        sw $t1, 4($sp)
        addi $a0, $a0, 4
        lw $t2, 0($a0)
        sw $t2, 8($sp)
        beq $s0, $t0, retorna_endereco
        bgt $s0, $t0, muda_direita
        blt $s0, $t0, muda_esquerda

retorna_endereco:
        jr $ra

muda_direita:                       #vai para o nó da direita da árvore para a
                                    função busca_recursiva
        lw $a0, 8($sp)
        addi $sp, $sp, 8
        j busca_recursiva

muda_esquerda:                      #vai para o nó da esquerda da árvore para a
                                    função busca_recursiva
        lw $a0, 4($sp)
        addi $sp, $sp, 8
        j busca_recursiva
```