CSCE 2610 Assembly Language/Computer Organization – Class Project #1

Date Assigned: 02-15-2012 (Wednesday)

Date Due: 03-28-2012 (Wednesday)

Percent of Final Grade: 20%

Important: There will be NO late submissions on the project.

Project Overview

You will be implementing a simple library database in MIPS that is structured as a binary search tree (BST). Each record will contain the following fields:

id – 4 bytes (1 word)
year – 4 bytes (1 word)
title – 64 byte string
description – 256 byte string

This implies that the size of each record is constant at 328 bytes. In C, the records could be implemented using the following structs:

```
struct book{
    int id, year;
    char title[64], description[256];
};
```

You may want to take a look at program9.txt in the sample problems to see how to address structures in MIPS.

You must be able to insert elements into the tree, search for a particular record (by id), print the data associated with a single record, and perform a traversal (inorder AND preorder). Note that deletion is NOT necessary for this project. Hopefully that simplifies this a bit.

The way that you organize your tree is up to you, but I offer a few ideas:

1) Use a dynamic tree structure (see dynamic.asm on BlackBoard) with all of the book members plus two "pointers" to a left/right child.
2) Have a static table that can be accessed by an index. Each row of this table will have the book members plus two index values. You can follow these to find the left and right children.

Project Deliverables

I would like all source code in a single .asm file. I will be assembling using MARS. Please submit using the BlackBoard system. If all else fails, e-mail me.

Grading

Eight Components (see below), each worth 10 points.

Project writeup (20 points)

I would like a description of how you've addressed the project – e.g. what procedures you've defined and why, what problems you encountered and how you solved them, what you've learned doing this project, etc. I'd like to see that you were actually challenged by this.

Project Components

Basic Working Program

No syntax errors, exits correctly, etc.

Valid user command loop

You should have some loop that interacts with the user (asks for input).
Please accept the following commands:

A – add a new record into the tree
F – find a record in the tree
P – perform a preorder traversal
I – perform an inorder traversal
Q – exit the loop and quit the program

Note that F and A will require additional input.

Handles user input on an "add" operation

Must correctly input id, year, title, and description from the user using the normal syscalls

Correctly handles find function:

Finds a record in the tree (and prints), if it exists.
Reports to the user if a record is NOT found in the tree

Prints the record data cleanly

This will need to be done on a "find" operation AND in both forms of the traversal.
It may be wise to have DIFFERENT print operations for these two cases, but I leave that to you.

Performs correct traversals

Inorder traversal (left, parent, right)
Preorder traversal (parent, left, right)

Correctly inserts into a tree

Note that this should not break any other operations (e.g. find, traversal)

Correctly uses recursion to perform tree operations

I will have to look at your code to determine this
The ability to use recursion is VERY helpful when dealing with trees.

Sample Code

Note that I have posted an example of this in C++ (project.cpp) to give an idea of what the finished product should look like. Feel free to compile and run this code – it's not too fancy. I use dynamic allocation, pointers, and C-style strings (with a limited and consistent size). I remind you that your title and description strings are limited to 64 bytes and 256 bytes respectively... INCLUDING the trailing NULL character, so be careful to handle this appropriately.

I have also included a file dynamic.asm which should indicate how to dynamically allocate space in MIPS.

As for the sample code already posted, you may be interested in the following:

program1 (basics)
program2 (looping)
program3 (recursion)
program5/7 (strings)
program9 (structures)