# ECE 30
# Introduction to Computer Engineering

## Programming Project – Towers of Hanoi Problem
## Due: Thursday, June 5, 2014 at 11:00
## Submit to: ecethirty@gmail.com

## Office Hours of the Project TAs

All office hours will be in WLH, 2213B. Times listed below are in 24-hour time to avoid any ambiguity.

- Week of April 28th:
  - Friday: 15:00 - 16:00 - Y
  - Friday: 16:00 - 17:00 - D
- Week of May 5th:
  - Monday: 10:00 - 12:00 - Y
  - Tuesday: 14:00 - 15:00 - Y
  - Wednesday: 12:00 - 14:00 - D
  - Thursday: 10:00 - 11:00 - D
- Week of May 12th:
  - Friday: 15:00 - 16:00 - Y
  - Friday: 16:00 - 17:00 - D
- Week of May 19th:
  - Monday: 11:00 - 12:00 - Y
  - Monday: 17:00 - 18:00 - D
  - Tuesday: 12:30 - 14:00 - D
  - Tuesday: 15:00 - 16:00 - Y
  - Wednesday: 11:30 - 12:30 - Y
  - Thursday: 13:00 - 14:30 - D
  - Friday: 15:00 - 16:00 - Y
- Week of May 26th:
  - Monday: 10:00 - 11:30 - D
  - Tuesday: 13:00 - 14:30 - D
  - Wednesday: 11:30 - 12:30 - Y
  - Wednesday: 14:30 - 15:30 - Y
  - Friday: 15:00 - 16:00 - Y

## 1 Important Links

You must use the template file provided at:
`http://cosmal.ucsd.edu/~gert/ece30/project_submission.s`

Again, your project **must use the provided template file**. While you are free to add any further helper functions, your code must implement the following functions according to specification given in this document. The grader should be able to remove any of your functions (and any non-essential helper functions it may use) from your code and use it in another person's programming assignment without issue; i.e. do not flip the order of variables passed to functions via registers $a0 and $a1 or returned via $v0 and $v1, use the conventions given. The main and print functions are provided, along with all the text strings one needs to complete this project. These should not, in any form, be modified, unless explicitly instructed by the project TAs.

You should use MIPS assembly code best practices as far as register storing and stack management. See these quick guides for more info:

`http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm`
`http://people.cs.pitt.edu/~xujie/cs447/Mips/sub.html`

## 2 Project Description

The goal of the project is to write a MIPS program that implements a recursive solution to the Towers of Hanoi problem. The puzzle consists of three pegs/columns (1, 2, and 3) and $n$ discs (the number $n$ can vary: typical values range from 1 to 8). disc 1 is smaller than disc 2, which is, in turn, smaller than disc 3, and so forth, with the $n^{th}$ disc being the largest. Initially, all the discs are on peg 1 (assuming the start position is 1), stacked in monotonically increasing size: disc 1 at the top and disc $n$ at the bottom. The goal is to move all the discs to peg 3 (assuming the end position is the third peg). One may only move one disc at a time, which means one can only move the top disc from any of the three pegs to the top of one of the other two pegs. At no point may a larger disc be stacked on top of a smaller disc.

The overarching steps for this program are the following:

1. Get column height (number of discs), start column and end column from user.
2. Recursively determine next move.
3. Print move out and the respective game state after each move.
4. Repeat the latter 2 steps until stack is moved to end column.
5. Restart sequence unless column height is set to zero.
6. Say "goodbye" and exit.

## 3 Towers of Hanoi Move Algorithm

The overview of the optimal move algorithm is actually conceptually simple. After getting the input values, determine the intermediate/helper peg, then follow these steps:

1. Move $height - 1$ disks from start column to helper column by calling solve_hanoi on these elements.
2. Move bottom disk to end column.
3. Move the $height - 1$ disks held in the helper column to the end column by calling solve_hanoi on these elements.

In pseudocode, this looks like:

```
• solve_hanoi (col_height, start_col, end_col, helper_col)
    – solve_hanoi(col_height-1, start_col, helper_col, end_col)
    – move_disk (start_col, end_col)
    – solve_hanoi(col_height-1, helper_col, end_col, start_col)
• move_disk (start_col, end_col)
    – start_col_height -= 1
    – end_col_height += 1
```

The recursive nature of this solution will keep dividing the problem into smaller and smaller Towers of Hanoi game, until the sub-game holds a single element (col_height == 1). A single element sub-column is the base case, and need only itself be moved from start-to-end. It is highly recommended that you read over the description of the `http://en.wikipedia.org/wiki/Tower_of_Hanoi`Towers of Hanoi problem on Wikipedia (it's a very good overview).

We will be containing the number of discs (i.e. current height, not the recursion col_height) of each column/peg in one, combined, state 32-bit value. Each column will be represented by 10 bits. It must follow this convention:

| **Bits** | **31-30** | **29-20** | **19-10** | **9-0** |
|---|---|---|---|---|
| **Status** | Unused | Column 3 | Column 2 | Column 1 |

Table 1: 32-bit state variable arrangement. Bits 0-9 hold the number of discs for column/peg 1. Bits 10-19 hold the number of discs for column/peg 2. Bits 20-29 hold the number of discs for column/peg 3. Bit numbers 9, 19, and 29 are the most-significant bit for columns/pegs 1, 2, and 3, respectively. Bits 30 and 31 remain unused.

## 4 Project structure and Function Pseudo-code

It bears repeating: your project **must use the provided template file**. While you are free to add any further helper functions, your code must implement the following functions according to specification given below. The grader should be able to remove any of these functions (and any non-essential helper functions it may use) from your code and use it in another person's programming assignment without issue (i.e. do not flip the variables passed to registers $a0 and $a1 or returned via $v0 and $v1, use the conventions given). Remember to manage your stack! Main, and both print functions are provided in the template file—do not modify these functions!

**Specific register assignments are also given in the the template file provided at: `http://cosmal.ucsd.edu/~gert/ece30/project_submission.s`. The following functions must be implemented or have been supplied:**

### 4.1 main:

The main program does the following. Already implemented—do not modify!

1. Save state of all used saved registers, frame pointer, and return address on stack.
2. "main_loop":
   (a) Call get_intial_height.
   (b) Go to "end_game" if initial_height == 0.
   (c) Call get_start_column.
   (d) Call get_end_column.
   (e) Call get_initial_state.
   (f) Call solve_hanoi.
   (g) Print line separator.
   (h) Loop back to start of "main_loop".
3. "end_game":
   (a) Print goodbye.
   (b) Restore saved registers, frame pointer, and return address.
   (c) Exit.

### 4.2 get_initial_height:

Prompts and gets initial height from user.

- $v0 returns the integer value of the entered height.

1. Prompt user for number of discs/intial height.
2. Get input from user.
3. Check that input in [0,1023]. Otherwise loop to 1.

### 4.3 get_start_column:

Prompts and gets starting column from user

- $v0 returns integer value of starting column.

1. Prompt user for start column.
2. Get input from user.
3. Check that input in [1,3]. Otherwise loop to 1.

### 4.4 get_end_column:

Prompts and gets end column from user.

- $a0 contains the value of the starting column.
- $v0 returns integer value of the end column.

1. Prompt user for end column.
2. Get input from user.
3. Check that input in [1,3] and that input != start column. Otherwise loop to 1.

### 4.5 get_initial_state:

Initializes state variable with the user-given column height and start column.

- $a0 contains initial height
- $a1 contains the start column
- $v0 returns the initialized state variable

1. Shift initial height to appropriate location in state variable

### 4.6 solve_hanoi:

Move the discs in the start column to the end column

- $a0 contains the height of the tower to move
- $a1 contains the column to move the discs from
- $a2 contains the destination column number
- $a3 contains the present state variable
- $v0 returns the updated state variable

1. Check for base case (one disc to move)
2. If base case:
   (a) Print the move
   (b) Reduce height of start column by one (by modifying state variable)
   (c) Increase height of end column by one (by modifying state variable)
   (d) Print current state
3. Otherwise:
   (a) Call solve_hanoi (height-1, start, helper, state)
   (b) Move remaining disc to end. Hint use solve_hanoi and its base case
   (c) Call solve_hanoi (height-1, helper, end, state)
4. Clean up and return to calling function

### 4.7 print_move:

Prints the to/from of the disc move. Already implemented—do not modify!

- $a1 contains the column the disc is moved from
- $a2 contains the column the disc is moved to

1. Load and print disc_text
2. Load and print start column
3. Load and print str_to
4. Load and print end column
5. Load and print newline
6. No return value, just exit

### 4.8 print_current_state:

Prints the number of discs in each column. Already implemented—do not modify!

- $a0 contains the disc state

1. Load and print state_str
2. Load and print column 1
3. Load and print a comma
4. Load and print column 2
5. Load and print a comma
6. Load and print column 3
7. Load and print a newline
8. No return value, just exit

## 5 MIPS Program Requirements

- You must submit your solution by emailing a completed file *2014project_ABC_XYZ.s*, where ABC and XYZ are the initials of the two partners, respectively.
- Make sure to fill in the full name and student ID for both partners in the submitted file. There is a spot reserved for you to do so at the very top of the template file.
- We expect your code to be well-commented. Each instruction should be commented with a **meaningful** description of the operation. Ask yourself what kinds of comments would be useful if you didn't touch this code for a year or two and have completely forgotten how to code in MIPS. For example, this comment is bad as it tells you nothing:
  ```
  addi $s0, $t0, -1                                                # $s0 gets $t0-1
  ```
  A good comment should explain the meaning behind the instruction. A better example would be:
  ```
  addi $s0, $t0, -1                          # Initialize loop counter $s0 to "n-1"
  ```
- Each project team contributes their own unique code - no copying, cheating, or hiring help. We have found out before, and will find out again. Even if only one of the two partners is culpable, both students will be reported to Academic Affairs.

## 6 Sample Output

### 6.1 Example 1

Please enter height of the tower from 1 to 1023 (or zero to quit):
**0**
==========
Goodbye!

### 6.2 Example 2

```
Please enter height of the tower from 1 to 1023 (or zero to quit):
3
Please enter the initial column of the tower (1, 2, or 3):
1
Please enter the final column of the tower (1, 2, or 3):
2
Current tower heights are: 3, 0, 0
Move disc from 1 to 2
Current tower heights are: 2, 1, 0
Move disc from 1 to 3
Current tower heights are: 1, 1, 1
Move disc from 2 to 3
Current tower heights are: 1, 0, 2
Move disc from 1 to 2
Current tower heights are: 0, 1, 2
Move disc from 3 to 1
Current tower heights are: 1, 1, 1
Move disc from 3 to 2
Current tower heights are: 1, 2, 0
Move disc from 1 to 2
Current tower heights are: 0, 3, 0
==============================

Please enter height of the tower from 1 to 1023 (or zero to quit):
0
==========
Goodbye!
```

### 6.3 Example 3

```
Please enter height of the tower from 1 to 1023 (or zero to quit):
-3
Please enter height of the tower from 1 to 1023 (or zero to quit):
10000
Please enter height of the tower from 1 to 1023 (or zero to quit):
2
Please enter the initial column of the tower (1, 2, or 3):
-1
Please enter the initial column of the tower (1, 2, or 3):
5
Please enter the initial column of the tower (1, 2, or 3):
2
Please enter the final column of the tower (1, 2, or 3):
2
Please enter the final column of the tower (1, 2, or 3):
1
Current tower heights are: 0, 2, 0
Move disc from 2 to 3
Current tower heights are: 0, 1, 1
Move disc from 2 to 1
Current tower heights are: 1, 0, 1
Move disc from 3 to 1
```

```
Current tower heights are: 2, 0, 0
==============================

Please enter height of the tower from 1 to 1023 (or zero to quit):
0
==========
Goodbye!
```