CO200 - Computer Organization and Architecture

A0 - Simulation of MIPS Assembly in SPIM

Points to Note

- Write the MIPS-I assembly code for all the questions below. Simulate your code on `spim` (text based simulator) or `xspim` or `qtspim` (GUI based simulators).
- In the comments section of each MIPS assembly code, include a short description of the program, your name, roll number, date of writing the program and other information you deem relevant.
- Your submission is an archive containing the following: One README file + One directory per assignment question. This assignment (A0) has 12 questions (1,. . .,12) - your submission will be a zip file containing 12 directories and a README file. For each question (in each directory) include the following: MIPS code, snapshots of the simulator at key execution stages, and other relevant information.
- This is an team assignment. Teams of max 2. One submission per team.
- (a) Submission is due on August, 12, 2016, 1159PM. Pack your report, code, screenshots and other files in an archive and mail to `co200.nitk@gmail.com`.

Write the MIPS assembly code for the following.

1. Load two 32b constants on to $t0 and $t1. Add them and store the result in $t2.
2. Load a 32b constant in the memory location pointed to by the global pointer. Load another 32b constant in the first memory location of the data segment. Add them both and save the sum in the 100^{th} word of the data segment.
3. **Hello World Program**. Print "Hello World" on the output screen.
4. Initialize a contiguous chunk of memory to contain 10 two digit decimal numbers. Your program should calculate the sum of these numbers and put the result into $v0.
5. Write a program to reverse a string (choose your favorite string as a global parameter).
6. Implement a function that calculates the sum of the elements of the array. The function accepts the length of the array and the address of the first element of the array. The function returns the sum to the caller. The main procedure calculates the average of the elements of the array. The main procedure uses the mentioned the function. Use the MIPS compiler subroutine conventions for this code.
7. Along the same lines as the previous question, implement a matrix multiplication program using functions.
8. Implement a matrix multiplication program using functions. Two 3×3 matrices may be initialized in the data segment.
9. **Factorial Program**. Load a random number in $t0. Calculate its factorial using (a) loops, (b) recursion.
10. Read a positive integer and print its equivalent in words. Eg. If input is 123, output will be One Two Three. Stop the program when a negative number is entered.
11. Compute and print the first 100 prime numbers. Implement the following two procedures:
    - test_prime(n) - Return 1 if n is prime, else 0.
    - main - Iterates over integers, testing if each is prime. Print the first 100 numbers that are prime.
12. A recursive program for solving the classic mathematical recreation, the Towers of Hanoi puzzle. The puzzle consists of three pegs (1, 2, and 3) and n disks (the number n can vary; typical values might be in the range from 1 to 8). Disk 1 is smaller than disk 2, which is in turn smaller than disk 3, and so forth, with disk n being the largest. Initially, all the disks are on peg 1, starting with disk n on the bottom, disk n-1 on top of that, and so forth, up to disk 1 on the top. The goal is to move all the disks to peg 2. You may only move one disk at a time, that is, the top disk from any of the three pegs onto the top of either of the other two pegs. Moreover, there is a constraint: You must not place a larger disk on top of a smaller disk.

    The C program below can be used to help write your assembly language program.

```
/* move n smallest disks from start to finish using extra */
void hanoi(int n, int start, int finish, int extra){
  if(n != 0){
    hanoi(n-1, start, extra, finish);
    print_string("Move disk");
    print_int(n);
    print_string("from peg");
    print_int(start);
    print_string("to peg");
    print_int(finish);
    print_string(".\n");
    hanoi(n-1, extra, finish, start);
  }
}

main(){
  int n;
  print_string(Enter number of disks>);
  n = read_int();
  hanoi(n, 1, 2, 3);
  return 0;
}
```

References

A few references to help you kickstart your assignment follow. Search for SPIM tutorials on Google!

1. MIPS ISA Reference Sheet.
2. Patterson and Hennessy. "Assemblers, Linkers, and the SPIM Simulator". Book: Computer Organization and Design. Appendix A. 5e. MK. 2015.
3. Wikibooks x86 Assembly Language
4. Tutorials Point x86 Assembly Language Tutorial