Spring 2018 CDA3101

Programming Assignment 2

Date assigned : Feb. 6th, 2018

Due date : Feb. 14th, 2018, 11:59 pm

A 24-hour grace period with 20% penalty is given. No submission is accepted after the grace period.

Instructions

Transform the following code into MIPS instructions. Your programs should run correctly on the QtSPIM simulator. Submit your assembly solution (project2.s) containing the neatly written/organized MIPS code in e-Learning (Canvas) website before the deadline.

Important

- You should use comments ('#' followed by text) in order to make your programs more readable.
- The name of the file submitted **MUST** be "**project2.s**"
- You **MUST** verify that your submission in Canvas is successful by downloading your submission from Canvas and successfully testing it again using SPIM simulator. This will ensure that you uploaded the right file in eLearning and the upload is successful.

Problem Statement

Given three positive integers x, k and n (1 < n < 1000), write a program to compute [(x^k) mod n] using fast modular exponentiation. More references about fast modular exponentiation can be found in the following websites.

https://discuss.codechef.com/questions/20451/a-tutorial-on-fast-modulo-multiplication-exponential-squaring

https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/fast-modular-exponentiation

Your assembly implementation **should exactly follow** the pseudo code sequence given below. **Please do not perform any optimization at pseudo code level or at assembly level.**

Note:

You can assume the three input values are 32-bit integers. In other words, the program should work with three inputs which are less than or equal to the number that can be represented by 32 bits. However, x to the power of k (x^k) may exceed 32 bits (e.g., 2^100).

Tips:

For printing to/reading from console, you should first load **correct** value to register $v0, and then call "syscall" method. If there is an input, the value would be returned in $v0.

```c
// Fast modular exponentiation
int fme(int x, int k, int n)
{
    int temp;
    int result = 1;
    if (k > 0){
        temp = fme(x, k / 2, n);
        if (k % 2 == 1){
            result = x % n;
        }
        // As n < 1000, the result of multiplication
        // is less than 2^30
        result = (result * temp * temp) % n;
    }
    return result;
}

int main()
{
    int x, k, n;

    printf("Enter the first integer x: ");
    scanf("%d", &x);
    printf("Enter the second integer k: ");
    scanf("%d", &k);
    printf("Enter the third integer n: ");
    scanf("%d", &n);

    // For example, if you use 987^654 mod 321 as a testcase
    // The output should be 57
    printf("The result of x^k mod n = %d\n", fme(x, k, n));
    return 0;
}
```