EIE

Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0321: Estructuras de Computadoras Digitales I

Tarea 3

Profesor: Ariel Fallas Pizarro

Asistente: Jose E. Flores

Instrucciones

1. La tarea es individual. Debe resolver e investigar por su propia cuenta. Cualquier intento de plagio se procesará de acuerdo al reglamento de la Universidad de Costa Rica.
2. La fecha de entrega es: **<DIA> de <MES> del 2024, antes de la media noche.**
3. Debe entregar en el sitio virtual del curso un único archivo con extensión **.s** o **.asm**. El archivo debe llamarse **<carnet>_tarea<número de tarea>_grupo<número de grupo>**. El archivo debe contener al inicio un encabezado con sus datos (nombre, carnet, etc.) y una explicación breve de la idea detrás del código implementado.
4. Es sumamente necesario que el código contenga comentarios que expliquen el porqué de lo realizado.
5. Su código debe ser ejecutable en el simulador MARS. Cualquier programa que no ensambla correctamente recibirá automáticamente una nota de 0.

Enunciado

Dos números enteros *M* y *N*, se consideran **PAR TORTUGA** sí y solo sí:

- *M* es un número primo.
- *M* + *N* es un número primo.

Realice una función en lenguaje ensamblador MIPS `tortoise_pair(M,N)` la cual determine si un par de números enteros, *M* y *N*, proporcionados por el usuario, son par tortuga o no. La interfaz de la función debe ser la siguiente:

- Entradas:
  - *$a0* → **M**
  - *$a1* → **N**
- Salida:
  - *$v0 *→ **1**, si M y N son par tortuga; **0** si no.

Para saber si un número *N* es primo o no, implemente la siguiente función, escrita en C, en lenguaje ensamblador MIPS:

```
int is_prime(int N){
    int SQRT = sqrt(N);

    for (int i=2; i<SQRT; i++){
        if (mod(N,i) == 0){
            return 0;
        }
    }
    return 1;
}
```

Como puede observar, la función anterior depende de las funciones `sqrt(x)` y `mod(a,b)`.

La función `mod(a,b)` funciona igual que el operador `%` en C y calcula el valor de **a modulo b**, es decir, el residuo de la división de *a* por *b*.

Por el otro lado, la función `sqrt(x)` debe calcular la raíz cuadrada de *x*, y retornar la parte entera de esta (por ejemplo: *sqrt(3) = 1*). Para calcular la raíz cuadrada de un número *X*, se deber implementar el algoritmo de búsqueda binaria, el cual consiste en lo siguiente:

Una implementación de dicho algoritmo en C se vería de la siguiente manera:

```
int sqrt(int x){

    x = (float) x;
    float hi = x;
    float lo = 0.0;
    float epsilon = 0.001;
    float guess, guess_sqrd;

    while (hi-lo > epsilon){
         guess = (hi + lo);
         guess /= 2;
         guess_sqrd = guess*guess;

         if (x < guess_sqrd){
             hi = guess;
         }
         else {
             lo = guess;
         }
    }

    return (int) guess;
}
```

Nota: Existen diferentes posibles implementaciones de las funciones utilizadas para la tarea. Sin embargo, uno de los propósitos principales de esta tarea consiste en entender y dominar el uso del punto flotante. Por lo tanto, se calificará que los algoritmos implementados correspondan a los utilizados en el enunciado.

Rúbrica de Evaluación

Ejecutable

Criterio	**Puntos**
Main	20
`is_tortoise(M, N)`	5
`is_prime(N)`	15
`sqrt(x)`	30
Comentarios	10
Convención de Registros	10
Uso Adecuado de Pila	5
Orden	5
Total	**100**