EIE

Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Estructuras de Computadores Digitales I
IE-0321
III Ciclo 2023

Tarea 2

Profesor: Julián Morúa Vindas
Asistente: Ana Sofía Barrantes Mena

Instrucciones

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.

La fecha de entrega es: 5 de febrero del 2024, antes de la media noche.

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.

Es sumamente necesario que el código contenga comentarios que expliquen el porqué de lo realizado.

Su código debe ser ejecutable en el simulador MARS. Cualquier programa que no ensambla correctamente recibirá automáticamente una nota de 0.

El enunciado del problema a resolver se encuentra en la siguiente página.

EIE

Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Estructuras de Computadores Digitales I
IE-0321
III Ciclo 2023

Enunciado

Escriba una función en ensamblador de MIPS que determine si los paréntesis en un string de caracteres son válidos.

Son considerados como paréntesis los siguientes seis caracteres: `(`, `)`, `[`, `]`, `{`, `}`.

Un string de caracteres con paréntesis válidos cumple lo siguiente:

- Todo paréntesis que se abre se debe cerrar con el mismo tipo de paréntesis.
- Los paréntesis que se abren se deben cerrar en el orden correcto.
- Todo paréntesis que cierra tiene un correspondiente paréntesis que abre.

Especificación de la función

La interfaz de la función tiene estas entradas y salidas:

- **Entrada**: `$a0` → la dirección en memoria de un arreglo de caracteres ASCIIZ.
- **Salida**: `$v0` → el resultado de la validación; 1 si los paréntesis son válidos, 0 si no.

Ejemplos:

- `“”` → **`$v0 = 1`**
- `“()”` → **`$v0 = 1`**
- `“() [] {} {{{()}[]}()}”` → **`$v0 = 1`**
- `“(hay [varios {niveles} de] parentesis)”` → **`$v0 = 1`**
- `“(se cierra con un tipo {diferente}]”` → **`$v0 = 0`**
- `“{este deja (un nivel abierto)!”` → **`$v0 = 0`**
- `“este, por otro lado, cierra 2 veces sin abrir])”` → **`$v0 = 0`**
- `“[{y este último cierra en un orden incorrecto]}”` → **`$v0 = 0`**

Sugerencia. Recuerde que la pila es una estructura de tipo LIFO (“Last In, First Out”). Piense de qué forma puede aprovechar esta propiedad para simplificar las validaciones que debe hacer con cada pareja de paréntesis que se abre y se cierra.

Programa Principal

Escriba un programa principal en el que imprime el string de caracteres, luego haga el llamado a la función implementada, y por último imprima “`parentesis validos`” o “`parentesis invalidos`” dependiendo del valor de retorno de la función.