Escuela Politécnica Superior

Práctica 2:

Estructura de Computadores

Saltos

UNIVERSIDAD NEBRIJA

Índice/Tabla de contenidos

1. METODOLOGÍA Y FORMATO DE LA ACTIVIDAD

Esta actividad está basada en los contenidos estudiados en los temas 1, 2 y 3 de la asignatura.

Se proponen varios ejercicios que deberán ser completados por equipos de hasta tres alumnos en la sesión 2 de prácticas de laboratorio.

- La entrega se realizará a través del campus virtual mediante un único archivo ZIP, que albergue todos los ejercicios, antes del 29 de marzo de 2023.
- La evaluación se realizará a través del campus virtual, teniendo en cuenta:
    - La corrección del programa en base a las especificaciones descritas.
    - La calidad del código.
    - El cumplimiento de las normas de estilo.

2. DESARROLLO DE LA ACTIVIDAD PRÁCTICA

2.1. Objetivo

El objetivo de esta práctica es comprender los **saltos en lenguaje ensamblador** y utilizarlos para modificar el flujo de control de un programa.

Las presentes prácticas están diseñadas para utilizar el lenguaje ensamblador de RISC-V, sobre un entorno de simulación RIPES.

2.2. Saltos

En programación, tanto las sentencias condicionales (*if-then-else*) como los bucles (*for, while*) se implementan mediante saltos. Un salto es una modificación del flujo de control de un programa, mediante el cual la próxima instrucción que se ejecuta no es la siguiente en orden secuencial, sino que el contador del programa (PC) realiza un salto a otra posición.

Los saltos más habituales son los condicionales, que se producen cuando se cumple una condición que relaciona dos registros.

INSTRUCCION	CONDICIÓN
`beq a1, a2, label`	a1 = a2
`bne a1, a2, label`	a1 ≠ a2
`bge a1, a2, label`	a1 ≥ a2
`blt a1, a2, label`	a1 < a2

En todos los casos, cuando se cumple la condición, el programa saltará a la instrucción marcada con la etiqueta "***label***".

Los saltos incondicionales son aquellos que se producen siempre, sin tener en cuenta ninguna condición, como, por ejemplo: **beq a0, a0, label**

En este caso, da igual el valor de registro que se use, ya que la condición de igualdad se cumplirá siempre.

También se puede usar la pseudoinstrucción de salto: **j label**

3. ENTREGABLES

Se entregará un programa en ensamblador para cada uno de los ejercicios siguientes:

3.1. Conversión simple de minúsculas en mayúsculas

Definir una variable de tipo "**string**" en memoria e inicializarla con cualquier palabra de 6 caracteres, utilizando únicamente letras minúsculas.

Realizar un programa que cambie todas las letras de la palabra a mayúsculas. Para este caso, no utilizar ningún salto. Como resultado, mostrar en pantalla la palabra inicial en minúsculas, y en la siguiente línea la misma palabra convertida a mayúsculas.

Para realizar el cambio, aprovechar la propiedad de que los códigos ASCII de una letra en mayúsculas y la misma letra en minúsculas están siempre a una distancia fija.

Recordar que cada letra almacenada en memoria ocupa un byte y que por lo tanto hay que utilizar "**lb**" y "**sb**".

3.2. Conversión avanzada de minúsculas en mayúsculas

Definir una variable de cadena en memoria, de longitud indeterminada, e inicializarla con una frase.

La frase puede contener cualquier cantidad de letras minúsculas y espacios en blanco, pero debe acabar en punto.

Repetir el apartado anterior, pero esta vez utilizando un bucle.
La condición de parada del bucle se producirá al encontrar el punto final de la frase.