95.57 Organización del Computador
Guía de Ejercicios ARM

v1.1.1

Parte 1

Práctica 1. Hola mundo

Escribir el código ARM que ejecutado bajo ARMSim# imprima el mensaje “`Hola Mundo`”

Práctica 2. Mostrar cadenas de caracteres

Escribir el código ARM que ejecutado bajo ARMSim# imprima dos cadenas de caracteres predefinidas en memoria incluyendo salto de línea “`Hola`” y ”`Chau`” utilizando una subrutina que imprima un string cuya dirección esté en el R3.

Práctica 3. Cálculos aritméticos y lógicos

Escribir el código ARM que ejecutado bajo ARMSim# realice las siguientes operaciones aritméticas y lógicas sobre dos números cargados en memoria: Suma, Resta, Multiplicación, AND, OR, XOR, Shift Izquierda, Shift Derecha, Shift Derecha Aritmética. Dejar el resultado de las operaciones en los registros del R2 al R10.

Práctica 4. Entrada y Salida de entero

Escribir el código ARM que ejecutado bajo ARMSim# que lea un entero desde un archivo e imprima el mismo entero por pantalla.

Práctica 5. Negar enteros desde archivo

Escribir el código ARM que ejecutado bajo ARMSim# lea dos enteros desde un archivo e imprima:

El primer entero en su propia línea.
El resultado de aplicar NOT al primer entero en su propia línea.
El segundo entero en su propia línea.
El resultado de aplicar NOT al segundo entero en su propia línea.

Práctica 6. Mostrar cálculos aritméticos y lógicos

Escribir el código ARM que ejecutado bajo ARMSim# realice las siguientes operaciones aritméticas y lógicas sobre dos enteros almacenados en un archivo: Suma, Resta, Multiplicación, AND, OR, XOR, Shift Izquierda, Shift Derecha, Shift Derecha Aritmética. Imprimir por pantalla los resultados de las operaciones en sus propias líneas.

Parte 2

Práctica 7: Cálculo de valor absoluto con instrucciones condicionales

Escribir el código ARM que ejecutado bajo ARMSim# lea un entero desde un archivo e imprima el valor absoluto del entero. Utilizar instrucciones ejecutadas condicionalmente y no utilizar bifurcaciones condicionales.

Práctica 8: Cálculo de valor absoluto con bifurcación

Escribir el código ARM que ejecutado bajo ARMSim# lea un entero desde un archivo e imprima el valor absoluto del entero. Utilizar bifurcaciones condicionales.

Práctica 9: Cálculo de mínimo y máximo

Escribir el código ARM que ejecutado bajo ARMSim# lea dos enteros desde un archivo e imprima el mínimo y el máximo respectivamente de la siguiente manera:

```
Min: <mínimo>
Max: <máximo>
```

Práctica 10. Cálculo de mediana

Escribir el código ARM que ejecutado bajo ARMSim# lea tres enteros desde un archivo e imprima la mediana, siendo la mediana el valor de la variable de posición central en un conjunto de datos ordenados. Por ejemplo, si los valores fueran 5, 8 y 9, la mediana sería 8.

Práctica 11: Codificación de While

Escribir el código ARM que ejecutado bajo ARMSim# imprima los números del 0 al 9.
Pseudocódigo:

```
x = 0
while (x < 10) {
   print x
   x++
}
```

Práctica 12: Cálculo de factorial

Escribir el código ARM que ejecutado bajo ARMSim# lea un entero desde un archivo, calcule el factorial de ese entero y muestre los valores intermedios del proceso. El algoritmo podría resumirse como:

```
n = <<entero leído desde archivo>>
accum = 1
while (n != 0) {
  accum = accum * n
  print accum
  print "\n"
  n = n - 1
```

```
}
print accum
print "\n"
```

Una salida aceptable del programa sería, para el caso que el valor de entrada fuera 5:

```
5
20
60
120
120
120
```

Puede asumirse que el archivo no contendrá un entero negativo.

Práctica 13: Cálculo recursivo de factorial

Escribir el código ARM que ejecutado bajo ARMSim# lea un entero desde un archivo, calcule el factorial de ese entero haciendo llamadas recursivas a la misma subrutina y muestre los valores intermedios del proceso. El algoritmo debería calcularse como:

$$x! \begin{cases} x * (x-1)! & \text{si } x > 1 \\ 1 & \text{si } x = 1 \end{cases}$$

Una salida aceptable del programa sería, para el caso que el valor de entrada fuera 5:

1
2
6
24
120
120

Práctica 14: Cálculo de fibonacci

Escribir el código ARM que ejecutado bajo ARMSim# lea un entero desde un archivo, calcule el valor de la posición que corresponde a ese entero en la sucesión de Fibonacci.

$$fib(x) \begin{cases} fib(x-1) + fib(x-2) & \text{si } x > 2 \\ 1 & \text{si } x < 2 \end{cases}$$

Una salida aceptable del programa sería, para el caso que el valor de entrada fuera 8:

21

Parte 3

Práctica 15: Imprimir y reemplazar enteros almacenados en memoria

Escribir el código ARM que ejecutado bajo ARMSim# imprima dos valores enteros definidos en memoria, los reemplace por otros dos valores e imprima los dos nuevos valores.

Práctica 16: Mostrar elementos de un vector utilizando direccionamiento por registro indirecto

Escribir el código ARM que ejecutado bajo ARMSim# imprima los valores de un vector de cuatro enteros definidos en memoria, recorriendo el vector mediante una subrutina que utilice direccionamiento por registro indirecto.

Práctica 17: Mostrar elementos de un vector utilizando direccionamiento por registro indirecto con post-incremento

Modificar el ejercicio para utilizar direccionamiento por registro indirecto con post-incremento.

Práctica 18: Mostrar elementos de un vector utilizando direccionamiento por registro indirecto con registro indexado

Modificar el ejercicio para utilizar direccionamiento por registro indirecto con registro indexado.

Práctica 19: Mostrar elementos de un vector utilizando direccionamiento por registro indirecto con registro indexado escalado

Modificar el ejercicio para utilizar direccionamiento por registro indirecto con registro indexado escalado.

Práctica 20: Encontrar el menor elemento de un vector

Escribir el código ARM que ejecutado bajo ARMSim# encuentre e imprima el menor elemento de un vector, donde el vector está especificado con el label `vector` y la longitud del vector con el label `long_vector`.

Práctica 21: Calcular y almacenar suma de una constante a un vector

Escribir el código ARM que ejecutado bajo ARMSim# lea los valores de un vector (`vector`) de longitud `long_vector`, sume un valor específico (`valor`) y guarde el resultado en otro vector (`vector_suma`).