Universidad Técnica Federico Santa María
Campus Santiago San Joaquín
Juan Pablo Jorquera 201573533-6
Cristian Navarrete 201573549-2

Informe Tarea 3
Arquitectura y Organización de Computadores

MCD

En primer lugar se imprime para pedir los valores que se desean calcular. El problema en sí se resolvió usando el Algoritmo de Euclides, ya habiendo guardado los valores en a_0 y a_1 se llama a MCD como función, pasándolos a los registros s_0 y s_1, trabajando sobre ellos e intercambiándolos de ser necesario, de modo que el menor de ellos quede en s_0.

Luego se creó un loop *mcd* que va buscando el resto de la división de ambos para aplicar el Algoritmo, en caso de ser 0 el resto significa que terminó el proceso; de no ser así, se vuelve a iterar usando como nuevos valores el menor de los antes divididos y el resto calculado.

No es necesario hacer caso base, ya que en caso de no encontrar, simplemente el resto llegará a 1, el cual debe entregar como resto 0.

Al ser 0 el resto, el *beq* dentro del loop lo dirige hacia el final, donde se imprime el valor encontrado y se almacena en la salida creada al principio.

Lucas

Para comenzar se recibe el término enésimo como input, el cual almacenamos en a_0 para llamar a la función.

Para continuar, definimos los dos primeros términos en s_0 y s_1, respectivamente y tanmbien se define s_3 el cual almacena la suma de los numeros vistos. A continuación se realizan saltos para entregar el resultado en caso de que los terminos s_0 o s_1 sean los términos que se busquen.

Luego se utilizó t_0 como un contador, el cual usaremos para verificar si se llegó al término enésimo y t_1 como auxiliar para ir moviendo los terminos de la suma inmediata (L_{s-1} y L_{s-2})

A continuación, en el loop se guarda en variable auxiliar el termino s0 y se mueve s_1 a s_0 para finalmente mover el valor de t_1 a s_1 y realizar la suma de s_2 con s_1.

Al finalizar el loop se imprime el valor y se guarda en la salida definida al principio.

Universidad Técnica Federico Santa María
Campus Santiago San Joaquín
Juan Pablo Jorquera 201573533-6
Cristian Navarrete 201573549-2

Fibonacci

Para comenzar se pide el input por consola para guardar el valor en a_0 y se llama a Fibonacci como función.

Para continuar, definimos los dos primeros términos en s_0 y s_1, respectivamente y se realizan saltos (j) para entregar el resultado en caso de que esos sean los términos que se busquen.

Luego se utilizó t_0 como un contador, el cual usaremos para verificar si se llegó al término enésimo y t_1 como auxiliar para ir calculando la suma de los términos actuales, para poder así actualizar s_0 y s_1 a los siguientes elementos de la serie.

Finalmente al finalizar el loop se imprime el valor y se guarda en la salida definida al principio.

Cabe destacar que como el MIPS trabaja en 32 bits y para enteros, utiliza el primer bit para el signo, el máximo término que puede calcular es cuando $n = 46$, ya que es el último que cabe en $2^{31} - 1$.

Factorial

Para comenzar se recibe como input el número al que se desea calcular su factorial, el cual almacenamos en a_0 para luego llamar a la función.

Para continuar, definimos el primer termino s_0 se realizan saltos para entregar el resultado en caso de que los terminos que se busquen sean 0 o 1.

Luego se utilizó t_0 como un contador, el cual usaremos para verificar si se llegó al término enésimo y ademas para realizar la multiplicacion por el siguiente termino.

Finalmente al finalizar el loop se imprime el valor y se guarda en la salida definida al principio.

Extra

Se dejo espacio en .data para ingresar la entrada y facilitar la lectura de ellos, utilizando hexadecimal para tener una mejor representación de los bits que se ingresan (comparado con decimal).

Se dividió el trabajo en tres funciones distintas: calcular largo de bits, calcular potencia de 2 y realizar el intercambio de bits. Desde el main se llama a la función del rotor, que utiliza las otras dos para realizar el proceso.

En primer lugar se calcula el largo de bits del primer registro, el cual almacenamos en s_0, el cual necesitamos para formar una constante que se usará para extraer los bits del primer registro. No es

necesario hacerlo para el segundo registro, ya que los bits que se extraen en este caso se calculan desde la derecha.

Luego se busca armar una potencia de 2 de el largo necesario para extraer los bits de cada registro: el primero se arma utilizando el largo encontrado menos la cantidad de bits que se desean intercambiar, por otro lado, la segunda potencia se arma simplemente utilizando la cantidad de bits a intercambiar, almacenándolos en s_1 y s_2 respectivamente.

Con esta potencias guardadas, se dividirán los números, para que así en $\$LO$ y en $\$HI$ (cuociente y resto) queden: la parte que se quiere incluir en el otro número y la parte que se desea guardar del número actual, dependiendo de cuál término estoy dividiendo. Estos resultados se van guardando entre t_0 y t_4.

A continuación la parte que corresponde dejar a la izquierda se amplifica por cada una de las potencias para los resultados y se les suma la parte que se desea dejar a la derecha que corresponda, quedando así los resultados en s_3 y s_4 respectivamente. Específicamente, el cuociente obtenido del primer término es lo que se va sumar al segundo término y el resto se mantiene tal cual. Por otro lado, el cuociente obtenido del segundo término se guarda, se amplifica por la potencia y se le suma lo antes calculado. Luego, el resto de éste se amplifica y luego es sumado al resto guardado del primero, para obtener así ambos términos requeridos. Dichos valores quedan en s_3 y s_4 y también se almacenan en las salidas creadas al inicio.

Es importante ver que al realizar este proceso se pueden producir problemas con los ceros a la izquierda del número, ya que son ignorados cuando se encuentran al principio del resultado, es por esto que analizaremos los casos posibles. Primero vemos que para los LSB no se generan problemas, ya que el número amplificado llena los dígitos para incluir los ceros que falten (como si se estuviera haciendo extensión de 0 hasta tener los dígitos deseados). Por otro lado cuando haya ceros al comienzo del número en los MSB, la rutina simplemente los ignorará y guardara el resultado con menos dígitos de lo deseado. Es por esto que por ejemplo, si el resultado esperado de un número es 0b0000, en cambio se entregará 0b0, lo que corresponde hacer es obviar los 0 al momento de leerlos.

Además, se decidió para esta ocasión no imprimir el resultado, ya que QtSpim no cuenta con impresión binaria y en su lugar lo muestra como decimal, ésto no ayuda a ver los bits que se intercambian. Entonces se recomienda verificar usando los registros s_3 y s_4 (Ya que v_0 se usa para salir).