Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Laboratorio Organización de computadores
Laboratorio n°2

Catalina Isidora Riquelme Zamora

Profesor: Carlos González
Ayudante: Ricardo Hasbun
Sección: L-4

Diciembre 2021
Santiago - Chile

Contenido

Introducción

A lo largo del laboratorio realizarán diversos programas en MIPS con la finalidad de escribir, ensamblar y depurar programas MIPS de manera correcta, a través del desarrollo de tres partes, donde el primero se relaciona con el uso de syscall, en la segunda parte se asocia con subrutinas para multiplicación y división de números y finalmente la tercera parte se relaciona con la aproximación de funciones matemáticas. Para dar solución a estas problemáticas, se escribirán programas incluyendo instrucciones aritméticas, de salto y memoria, además de diversas llamadas mediante syscall. El objetivo de esta actividad es usar MIPS para escribir, ensamblar y depurar programas, por otro lado, se busca implementar algoritmos en MIPS para resolver problemas matemáticos de baja complejidad, logrando comprender el uso de subrutinas en MIPS e incluyendo el manejo del stack.

Marco teórico

Para poder desarrollar esta experiencia es necesario contar con antecedentes sobre syscall, procedimiento de multiplicación y división. Los programas de lenguaje ensamblador solicitan servicios del sistema operativo mediante la instrucción, para esto la instrucción transfiere el control al sistema operativo que luego realiza el servicio solicitado. Por otra parte, al encontrarse restringido el uso de instrucciones de multiplicación, división y desplazamiento, se debe pensar esta como la suma repetitiva de n números. Finalmente, la división debe ser pensada de manera similar a una multiplicación como un conjunto de sumas repetitivas de n números y trabajo con restos.

Desarrollo de solución

Parte 1

Para lograr un programa que lea enteros ingresados por el usuario, determinando si la diferencia entre estos dos valores es par o impar es necesario determinar en el apartado .data cinco variables del tipo. asciiz las cuales ayudaran al usuario a interactuar con el programa. Posteriormente, en el apartado .text se declara la etiqueta main, en donde en primera instancia busca calcular los dos números enteros ingresados por el usuario, para esto se imprime en pantalla solicitando el primer número entero, donde luego de obtenerlo se mueve al registro temporal $t0, luego se imprime por pantalla solicitando el segundo número, una vez obtenido se mueve al registro temporal $t1, una vez almacenados los datos en los registros temporales se procede a restar estos dos valores a través de sub y almacenando en este resultado en la variable temporal $t2, posteriormente se imprime su valor por pantalla junto al mensaje correspondiente. En segunda instancia se busca calcular la paridad del resultado de esta sustracción, para conseguir esto se almacena el número 2 en la variable temporal $t2, para luego realizar una división entre el resultado de la sustracción y el número 2, donde el resto se almacena en H1 y posteriormente se mueve a la variable temporal $t3 en el caso que el resto almacenado en $t3 sea igual a 0 ($zero) se dirige a la etiqueta par, por el contrario, se muestra por pantalla impar. Finalmente se realiza un salto al término de la ejecución. Supuesto: se asume que la diferencia entre el primer entero y el segundo entero será siempre positiva

Parte 2

Parte 2.a

Para conseguir un programa el cual calcule la multiplicación de dos enteros mediante la implementación de subrutinas es necesario asignar el valor de la primera y segunda variable, en las variables temporales $t0 y $t1, respectivamente, posteriormente se salta a la subrutina mul_1, la cual asigna un contador en la variable temporal $t2, la cual comienza en 0, para luego realizar un loop el cual saltará a back únicamente cuando la variable temporal $t0 tome el valor 0, de lo contrario, se le suma al contador almacenado en $t2 el valor de la variable almacenada en $t1 el número de veces de las segunda variable, esto se consigue a través de la sustracción de una unidad cada vez que es ejecutado el loop, una vez finalizado y el valor de la variable temporal $t0 sea igual 0 se salta a back, donde se salta a la instrucción almacenada en ra, por consiguiente se regresa a la línea 16 (li $v0,1) donde se imprime el resultado y finalmente se termina la ejecución.

Parte 2.b

Para conseguir un programa el cual calcule el factorial de un número entero, utilizando multiplicación mediante la implementación de subrutinas es necesario determinar en el apartado .data dos variables del tipo .word, donde la primera tendrá el número del cual se desea calcular el factorial, por otra parte, en la segunda variable se determina la dirección. Posteriormente en el apartado .text se indica la dirección en la variable temporal $t0 y la dirección donde se encuentra el número en la variable temporal $t1, además de asignar el número, luego se asigna a la variable temporal $t2 el valor 1, esta servirá como contador. Es necesario tomar en cuenta los casos posibles 0! Y 1! para esto se determina la variable temporal $t4 igual a 1, en el caso de que el

número sea 0! y 1! se salta a la subrutina special_case, de lo contrario, se salta a la subrutina mul_1, donde se realiza un proceso similar al explicado en la parte 2.a, una vez completado este proceso se regresa a main donde se almacena el resultado de esta multiplicación, se le resta un número a la variable temporal $t1, donde se encuentra almacenado el número, luego en el caso de este ser diferente a 0 se repite el proceso, de lo contrario, se imprime por pantalla el resultado y finalmente se termina la ejecución.

Parte 2.c

Para conseguir un programa el cual calcule la división entre dos enteros mediante la implementación de subrutinas es necesario asignar el valor del dividiendo, divisor, primer decimal y segundo decimal,estos serán almacenados en las variables temporales $t0, $t1, $t9 y $t8, respectivamente, posteriormente en el caso que el divisor sea igual a 0 se muestra por pantalla no error, de lo contrario se salta a la subrutina div_1, la cual a través de un loop y un contador determina si el resto es igual a 0 se salta a decimal_2, de lo contrario se salta a decimal_1, donde decimal_1 se encarga de calcular el primer decimal y decimal_2 se encarga de calcular el segundo decimal, una vez realizado esto se imprime en la pantalla el resultado y termina la ejecución. Supuesto: se asume los dos enteros a dividir no son negativos.

Resultados

Parte 1

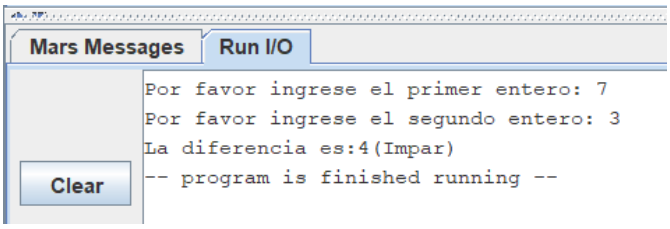

Imagen 1: parte1.s

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	268501085
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	7
$t1	9	3
$t2	10	2
$t3	11	1
$t4	12	0

Imagen2: registros parte1.s

Parte 2

Parte 2.a

Valor primera variable: 7. Valor segunda variable: 3

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	21
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	3
$t2	10	21

Imagen 3: registros parte2a.s

Parte 2.b

Numero de entrada: 4

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	24
$a1	5	0

Imagen 4: registros parte2b.s

Parte 2.c

Dividiendo: 7. Divisor: 3

```
2.33
-- program is finished running --
```

Imagen 5: parte2c.s

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	3
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	3
$t2	10	2
$t3	11	1
$t4	12	3
$t5	13	1
$t6	14	3
$t7	15	0
$s0	16	0

Imagen 6: parte2c.s

Conclusiones

Tras la realización de laboratorio se puede concluir que se lograron abordar dos de las tres partes del laboratorio, en el caso de la parte dos se logró realizar las tres partes de este, logrando escribir, ensamblar y depurar en MIPS, realizando un uso correcto de syscall, subrutinas con fines de multiplicación y división, sin lograr la aproximación correcta de funciones matemáticas en la parte tres. Para dar solución a las diversas problemáticas se escribieron programas con instrucciones aritméticas, de salto y memoria, además de diversas llamadas en syscall. Se logro el objetivo de usar MIPS con la finalidad de escribir, ensamblar y depurar la primera y segunda parte, además de cumplir el objetivo de resolver problemas de baja complejidad como lo son multiplicación, factorial y división, logrando comprender el uso de subrutinas en MIPS. Para una próxima entrega se requiere mejorar la compresión matemática para lograr realizar problemas de mayor complejidad matemática.

Referencias bibliográficas

Gómez, J. (1998) "Ensamblador MIPS". Departamento de electrónica UTFSM. http://www2.elo.utfsm.cl/~elo312/aplicaciones/spim/Tutorial_MIPS.pdf

Serrano, Ángel (s.f.) *"12. Programación en ensamblador MIPS".* Cartagena99. https://www.cartagena99.com/recursos/electronica/apuntes/tema12_ensamblador_MIPS.pdf

Kjell, B. (s.f.). "*Syscall".* Recuperado el 17 de diciembre de 2021 de: https://chortle.ccsu.edu/AssemblyTutorial/Chapter-22/ass22_2.html