Informe de Arquitectura sobre Fibonacci Recursivo

Elaborado por: José Díaz y María Moreno

2 de Abril de 2023

Introducción:

Se llama recursividad a un proceso mediante el cual, una función se llama a sí misma de forma repetida, hasta que se satisface alguna determinada condición. Nuestro informe se basa en la secuencia de Fibonacci en forma recursiva empleando MIPS. A continuación, presentamos nuestro informe.

Desarrollo:

Las directivas del ensamblador MIPS que sirven para delimitar, dentro del código fuente de un programa, sus segmentos de datos y de código (este último también denominado "Segmento de texto") son, respectivamente, ".data" y ".text". Entre ellos encontramos ascii y asciiz, que nos permiten almacenar la cadena string en memoria. En este espacio se escriben los mensajes que se desean imprimir. En nuestro caso, el resultado de la secuencia de Fibonacci.

(.word), Almacena las N cantidades de 32 bits en palabras de Memoria, sucesivas. Además, define las variables globales empleadas para insertar N y la respuesta, de acuerdo al valor ingresado.

(.globl main). Nos indica que el algoritmo es global.

(main) Indica el inicio del algoritmo, aquí se va a leer el valor que el usuario ingresa.

li $v0, 4. Nos muestra la cadena de mensaje de impresión

la $a0, prompt. Carga la dirección de inmediato

syscall. Se utiliza después de cargar un código numérico en el registro $v0 y en algunos casos poner un valor en $a0 como argumento.

Realiza un llamado al sistema, que sirve para pedirle alguna operación o servicio al sistema operativo. En el transcurso del algoritmo esta instrucción es utilizada en varias ocasiones.

Viene el ensamblaje de la función:

li $v0, 5. Aquí se carga el valor de inmediato y lee N como entero
syscall . Volvemos a llamar al sistema.
sw $v0, N. Almacena en dirección la palabra contenida en el registro $v0, que es el valor de N.
bltz $v0, Menor_Cero. Esta instrucción compara si el número $v0, que es el número ingresado por él usuario, es menor a cero.

En la siguiente parte del código, llamamos a la función Fibonacci:
lw $a0, N. Esta instrucción retorna $a0=N lee y desde la memoria las variables.
jal fib. Invoca a la función Fibonacci por su dirección.
sw $v0, answer. Almacena el resultado retornado en N=$v0.

Aquí, mostramos el resultado del código:
li $v0, 4. Este código llama al sistema para imprimir la cadena.
la $a0, results. Nos indica la dirección del resultado a imprimir.
syscall. Es la llamada al sistema.
li $v0, 1. Este código llama al sistema para imprimir un entero.
lw $a0, answer. Esta instrucción carga la respuesta.
Syscall. Nuevamente, se llama al sistema.

bgez $v0, Final. Nos indica que, si el contenido de $v0 es mayor o igual a cero, el flujo del programa salta a la etiqueta Final.

Esta parte del algoritmo nos indica, qué sucede cuando el usuario ingresa un número menor a cero:
li $a0, 0. Esta cargando inmediatamente el valor que se está ingresando como argumento del Fibonacci.
li $v0, 4. Es el código de llamada al sistema para imprimir el valor del resultado.
la $a0, results2. Es la dirección del resultado a imprimir.
syscall. Es la llamada al sistema.

Esta parte del código indica que está listo, termina el programa. Final:

li $v0, 10. Es el código de llamada al sistema para imprimir el valor del resultado y terminar.
Syscall. Es la llamada al sistema.
.end main. Indica el fin del algoritmo principal.

En esta parte se encuentra el algoritmo recursivo de Fibonacci.

fib:
subu $sp, $sp, 8. Aquí la diferencia se almacena en el primer registro sp.
sw $ra, ($sp). Esta instrucción salva la dirección de retorno.
sw $s0, 4($sp). Esta almacenando temporalmente el valor que está en $s0.

move $v0, $a0. Copia el contenido del registro $a0 al registro $v0.

ble $a0, 1, fibDone. Indica que salta a la instrucción por fibDone, si $a0 menor o igual a 1.

move $s0, $a0. Copia el contenido del registro $a0 al registro $s0.

sub $a0, $a0, 1. Coloca en el primer $a0 la diferencia de los enteros.(n-1)

jal fib. Aquí se indica que, salta incondicionalmente a la instrucción fib, y salva la dirección de la siguiente instrucción en el registro $ra.

move $a0, $s0. Mueve el contenido del registro $s0 a $a0

sub $a0, $a0, 2. Coloca en el primer $a0 la diferencia de los enteros.(n-2)

move $s0, $v0. Mueve el contenido del registro $v0 a $s0. jal fib. Indica que, salta a la instrucción fib, y salva la dirección de la siguiente instrucción en el registro $ra.(fib(n-2))

add $v0, $s0, $v0. Indica que se suma $s0 con el segundo $v0, y el resultado se coloca en el primer $v0. (fib(n-1)+fib(n-2))

fibDone: Nos dice que el Fibonacci ha finalizado.

lw $ra, ($sp). Carga en $ra la palabra de memoria direccionada por $sp.

lw $s0, 4($sp). Carga en $s0 la palabra de memoria direccionada por $sp.

addu $sp, $sp, 8. Se suma el segundo valor $sp con 8, y el resultado se almacena en el primer valor de $sp.

jr $ra. Salta a la instrucción cuya dirección esta contenida en el registro $ra. (Retorna al invocador)

.end f. Este es el fin de la secuencia del Fibonacci Recursivo.

Conclusión:

Utilizando Java y la herramienta Mars, hemos podido acceder al ensamblador del procesador MIPS. Esta programación es bastante distinta a la programación habitual. Además, hemos logrado realizar el algoritmo de la secuencia de Fibonacci y en nuestro ejemplo, recursivo.

Nuestro algoritmo, imprime el número de la secuencia del Fibonacci que el usuario desea conocer. Se le indica al usuario que debe ingresar enteros positivos para obtener el resultado, sin embargo, si comete un error al ingresar un número negativo, el algoritmo arroja un mensaje que indica que no es posible. Por lo que, debe ingresar un numero entero positivo mayor o igual a cero.