I Trabajo Extra Clase: Cálculo de Paridad

Yenira Chacón Molina
2015075331
e-mail:reiracm@gmail.com
Ian Coberly Jimenez
2016078185
e-mail:cobjim28@gmail.com
Eduardo Moya Bello
2015096664
e-mail:e.d.mobe@hotmail.com
Erick Obregón Fonseca
2016123157
e-mail:erickobregonf@gmail.com
Angélica Peréz Arias
2016110539
e-mail:angelicacr@hotmail.es

PALABRAS CLAVE: Algoritmo, Bit, Ensamblador, MIPS, Paridad.

1 INTRODUCCIÓN

En el siguiente informe se estudiará la implementación de algunas operaciones básica en pila a la hora de calcular la paridad de un número ingresado por un usuario de 32-bits. El siguiente programa será programado con el core instruction set de los MIPS definidas en su Green Card.

Primero se atacará el problema por medio de un mapa de memoria, luego se creará un diagrama de flujo y por último se programará en ensamblador un código el cual sea capaz de realizar dicha tarea.

2 MAPA DE MEMORIA

	Dirección	Instrucción
	40 0000	addi $t0, $zero, 3
	40 0004	jal parity
	40 0008	li $v0 10
parity:	40 000C	add $t2, $t0, $zero
	40 0010	add $t1, $zero, $zero
	40 0014	add $t3, $zero, $zero
	40 0018	j while
while:	40 001C	beq $t2, 0, END
	40 0020	andi $t3, $t2, 1
	40 0024	bne $t3, $zero, changeBitParity
	40 0028	j endLoop
changeBitParity:	40 002C	sub $t1, $t3, $t1
endLoop:	40 0030	srl $t2, $t2, 1
END:	40 0034	jr $31

Tabla 1: Mapa de memoria del los operandos.

3 DIAGRAMA DE FLUJO

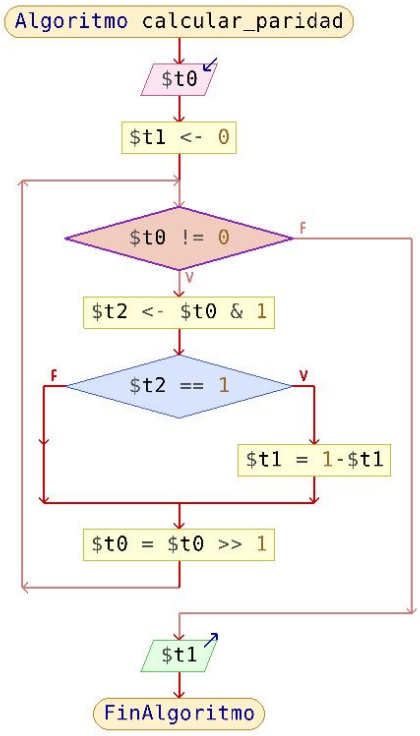

Figura 1. Diagrama de flujo para ilustrar el funcionamiento del código.

4 CÓDIGO EN ENSAMBLADOR

En el procedimiento principal primero el usuario le ingresa un número de 32-bits. Luego se realiza un subprocedimiento llamado parity. Luego termina y se sale del programa.

En parity se realizan los siguientes procesos. Primero se copia el número ingresado por el usuario con el fin de manipularlo. Luego se inicia una variable de bit de paridad en 0 y de la misma manera se inicia el bit que se va a leer. Luego inicia otro subproceso llamado while.

En el while se realiza lo siguiente. Primero se verifica que la copia del número ingresado por el usuario no sea igual a 0. En caso de serlo, se salta al subprocedimiento END. En caso de no serlo, se continúa con el resto del while. Después se obtiene el último bit del manipulable si este bit es un 1, se cambia la variable del bit de paridad. Luego se llama al subprocedimiento endLoop.

Para poder cambiar la variable del bit de paridad se utiliza otro subprocedimiento llamado changeBitParity. Lo que se hace es que se niega la variable del bit de paridad. Luego se llama al subproceso endLoop.

Para el subprocedimiento de endLoop, primero se obtiene el siguiente bit del número ingresado por el usuario y se devuelve al subprocedimiento while.

El subprocedimiento END solo realiza una función la cual es devolverse al procedimiento principal.

5 CONCLUSIONES

Por medio del código implementado se logró aprender acerca de algunas operaciones de pila básicas por mientras que este calculaba la paridad de un número aleatorio de 32-bits ingresado por un usuario. Así mismo también se logró adquirir un mejor manejo y comprensión del core instruction set de las MIPS.