Lenguaje Ensamblador y la arquitectura MIPS

David Ricardo Martínez Hernández Código: 261931
Juan Sebastian Roncancio Arevalo Código: 261585

***Resumen*— En esta práctica se realizara la implementación del algoritmo de la división en Asembler. Se procedió con el desarrollo explicado en la clase teórica, implementandolo en el lenguaje asembler para obtener la división de dos números enteros positivos.**

***Palabras clave*— Asembler, Condicional, División, Lenguaje, Mips, Resta, Suma.**

I. Procedimiento

Para cargar un archivo *.s* es necesario tener instalado *QtSpim* y se debe entrar a la interfaz gráfica del programa y hacer lo siguiente:

1) Ingresar a File
2) Entrar a Load File
3) Cargar el archivo .s que se va a ejecutar
4) Para ejecutar dicho archivo se ingresa a Simulator
5) Se hace click en Run/Continue o se oprime la tecla F5

EL menú principal de QtSpim tiene las siguientes opciones de menú:

- File
- Simulator
- Register
- Text Segment
- Data Segment
- Window
- Help

A. Definiciones

El **proceso de compilación** recibe el lenguaje de más alto nivel lo pasa a un compilador, el cual quita todas las etiquetas que tenga (una etiqueta es una referencia que establece el programador, con un nombre y no con una dirección especifica), luego de pasar al compilador es traducido al lenguaje ensamblador para finalmente enviarlo a un linker, el cual entregara el archivo ejecutable del proceso.
Un **linker** toma todos los objetos junto a las librerías, reservando las direcciones y variables, generando un ejecutable *.s* almacenándolo en memoria.
Un **archivo objeto** es un código máquina que contiene información binaria e información de mantenimiento que ayuda a combinar archivos de un programa.
El **lenguaje ensamblador** es el lenguaje que entienden las máquinas compuesto por 0 y 1.
La diferencia radica en que el **lenguaje de alto nivel** esta diseñado para ser comprendido por humanos mientras que el **lenguaje ensamblador** esta diseñado para las máquinas.
Una **subrutina** es una operación que detiene las instrucciones principales, realiza dicha la operación y continua las operaciones habituales.
Una **interrupción** es una función en la cual detiene la síntesis del programa por completo, hasta que se genere una función que reanude el proceso.

Arithmetic	add
	subtract
	add immediate
Data transfer	load word
	store word
	load half
	load half unsigned
	store half
	load byte
	load byte unsigned
	store byte
	load linked word
	store condition. word
	load upper immed.
Logical	and
	or
	nor
	and immediate
	or immediate
	shift left logical
	shift right logical
Conditional branch	branch on equal
	branch on not equal
	set on less than
	set on less than unsigned
	set less than immediate
	set less than immediate unsigned
Unconditional jump	jump
	jump register
	jump and link

Fig. 1: Set de instrucciones en lenguaje ensamblador para la arquitectura MIPS32. Tomado de [2], página 78

II. Aplicación

La operación división solo funciona para números positivos enteros, se realiza mediante una serie de restas sucesivas,

representado en la Fig. 2

III. Conclusiones

- Los problemas para realizar este informe fue que nunca se había trabajado en lenguaje ensamblador y como solo se puede trabajar con dos variables se hizo muy complicado realizar la rutina para la división, se empezó a realizar al suma para recibir los datos hacer la suma e imprimir el resultado en pantalla, pero generaba un error al realizar al suma, después de solucionar el problema se procedió a hacer la división por medio de las restas sucesivas.
- Las operaciones realizadas en el registro aun son desconocidas para nosotros, solo sabemos que se operan pero no sabemos como lo hace, es por eso que no incluimos el signo de los operandos.
- Es más sencillo realizar la división por medio de las restas sucesivas, porque solo se tiene que hacer restas hasta que el dividendo sea menor que el divisor.

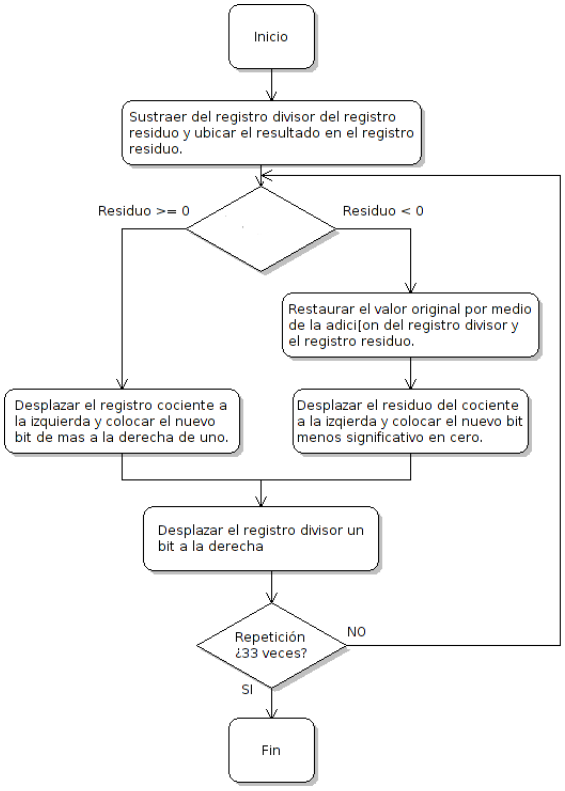

Fig. 2: Diagrama que se utilizo para realizar la división. Basados en [2], página 238.

Referencias

[1] Harris, David & Harris, Sarah. *"'Digital desing and computer architecture"'*. Pretince Hall, 2003.

[2] Patterson, David & Hennessy John *"'Computer Organization And Design - The Hardware-Software Interface"'*. Kindle Edition, Fourth Edition, 2006.