Comparación de aplicaciones en Hardware y Software

José Alejandro Logreira Ávila Código: 261722
David Ricardo Martínez Hernández Código: 261931
Edwin Fernando Pineda Vargas Código: 262100

***Palabras clave*— ALU, Ciclos de reloj, Hardware, Operandos, Multiplicación, Software, Sumas Sucesivas.**

***Resumen*— En esta práctica se busca encontrar las diferencias en implementación de una solución tanto en hardware como en software.**

I. Introducción

Existen varias formas de implementar algoritmos en MIPS. Durante esta práctica, se implementará el algoritmo mediante sumas sucesivas y mediante la instrucción mul o muli. Para verificar esto, se harán simulaciones mediante el GTK wave, con ello buscamos el tiempo de respuesta de cada casoy así podremos realizar la correspondiente comparación.

II. Tareas desarrolladas por software en el LM32

El código siguiente realiza la tarea de multiplicación, usando el algoritmo de sumas sucesivas. Este tipo de implementación de la multiplicación supone que no existe un módulo aritmético dedicado a la multiplicación en el procesador, y por tanto la forma de desarrollar una operación de multiplicación es mediante el uso recursivo de los recursos que sí tiene, como la unidad aritmética lógica.

Listing 1: Codigo inicial

```
#include "soc-hw.h"
int main(void){
    volatile uint32_t a,b,c,i;
    a = 10;
    b = 13;
    c = 0;
    while (i<b) {
        c = c + a;
        i = i + 1;
                }
    return 0;
}
```

Este algoritmo representa la forma de realizar tareas por software ya que ejecuta instrucciones sencillas, que hacen uso de los recursos simples disponibles en el procesador (suponiendo que no dispone del módulo multiplicador), para realizar tareas más complejas y elaboradas.
En la multiplicación de los enteros 10 y 13, se está sumando sucesivamente 10, 13 veces, por lo que estas sumas son las que debemos evidenciar como resultados en la salida de la ALU. Se declaran las variables de tipo volátil para poderlas observar en el simulador GTKwave.
La siguiente imagen muestra el comienzo de la operación, donde se inicializa uno de los registros con el sumando 10:

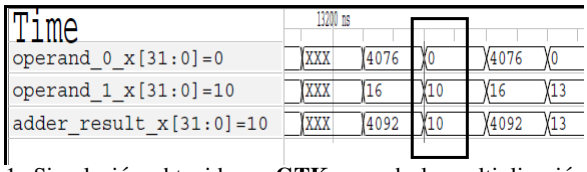


Fig. 1: Simulación obtenida en **GTKwave** de la multiplicación por sumas sucesivas.

Las siguientes imágenes muestran la primera iteración de la suma, y algunas de las iteraciones posteriores. Finalmente se obtiene el resultado en la iteración final, mostrada en la última imagen Fig. 2.
La TABLA I a continuación muestra los tiempos de ejecución del programa según se observó en el simulador GTKwave.

Operación	Tiempo de la OP	Tiempo de la OP Anterior	Tiempo Acumulado
Carga 10 en registro	13230	0	0
$0+10$	14430	1200	1200
$10+10$	15770	1340	2540
$10+20$	16560	760	3300
$10+30$	17290	760	4060
$10+40$	18050	760	4820
$10+50$	18810	760	5580
$10+60$	19570	760	6340
$10+70$	20330	760	7100
$10+80$	21090	760	7860
$10+90$	21850	760	8620
$10+100$	22610	760	9380
$10+110$	23370	760	10140
$10+120$	24130	760	10900

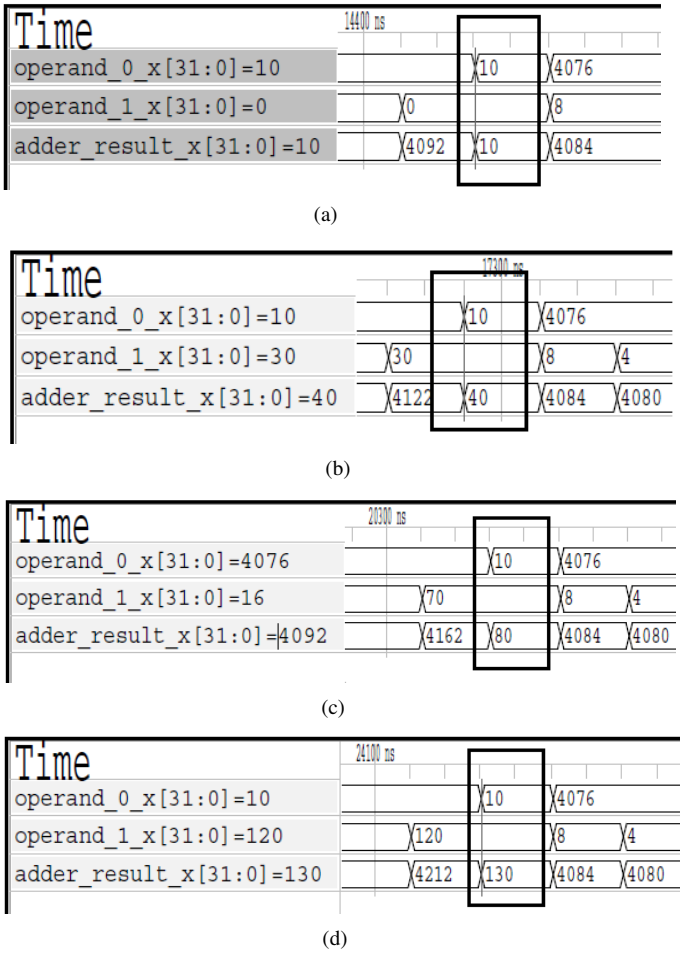
TABLA I: Tiempo de la multiplicación por sumas sucesivas para el procesador $LM32$.

Como es evidente en la tabla, existe una periodicidad en el tiempo que demora cada iteración de suma. El tiempo total de ejecución de la suma sucesiva es de 10900 nanosegundo, $10,9$ microsegundos. Este tiempo crecerá linealmente conforme crece la magnitud de los operandos.

III. Tareas implementadas en hardware en el LM32

Se utilizó el siguiente código en C para implementar la multiplicación

Listing 2: Codigo inicial



(a)

(b)

(c)

(d)

Fig. 2: Simulación obtenida en **GTKwave** de la multiplicación por sumas sucesivas.

```
#include "soc-hw.h"
int main(int argc, char **argv){
    volatile unsigned int multiplicador =10;
    volatile unsigned int multiplicando =13;
    volatile unsigned int producto = 0;
    producto = multiplicador*multiplicando;
    return 0;
}
```

El resultado de la simulación se observa en la figura 3, en dicha figura los datos ingresan a los registros en 14150 nanosegundos ($14.150\mu s$) y el resultado se obtiene a los 14170 nanosegundos ($14.170\mu s$), esto quiere decir que solo se demora un ciclo para dar la respuesta.

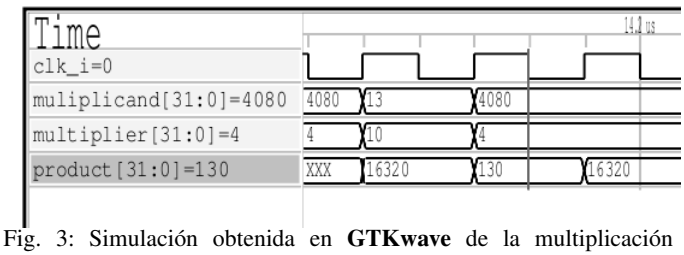


Fig. 3: Simulación obtenida en **GTKwave** de la multiplicación directa.

Comparado con el proceso de la sección pasada (ver II) el tiempo de ejecución es menor porque en el *main.s* llama a la función __ *mulsi3* por medio de la instrucción *calli*, esto hace que se reduzca el tiempo de ejecución de 13 ciclos a un solo ciclo (para este caso en particular).

IV. Conclusiones

- No cabe duda que el rendimiento en tiempo de realizar tareas por hardware es mucho más eficiente que por software.
- El lograr eficiencia implementando hardware, también significa un aumento de recursos en el mismo. Lo cual en el ámbito comercial se traduce en costos más elevados.
- Esta idea de desempeño versus recursos de hardware, es también evidente en la llamada Arquitectura del Set de instrucciones, en donde cada instrucción realizará una tarea sencilla o compleja conforme se dispongan de recursos de Hardware.

Referencias

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

[2] http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/index.cfm

[3] http://www.linuxencaja.net/wiki/Arquitectura_LM32_JPRR_%28261744%29