Desarrollo de una solución a la asignación de horarios de universitarios en lenguaje MIPS

Camila Gárate (6 horas) , Benjamín Sandoval (6 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
camila.garate@usach.cl, benjamin.sandoval.b@usach.cl

***Resumen*—Este documento aborda una problemática ficticia creada con fines educativos. Dicha problemática tiene lugar en el departamento de ingeniería informática de la Universidad de Santiago de Chile y consiste en la imposibilidad de designar un horario para los estudiantes, debido a problemas en la plataforma LOA, la cual normalmente cumple esa función. Los estudiantes de Arquitectura de Computadores crearon un programa que facilita la asignación de cursos al detectar similitudes y diferencias automáticamente.**

***Palabras claves*—MIPS, sistema binario, registros de memoria**

I. Introducción

En la búsqueda de una solución se decide utilizar un sistema de tablas que permitan representar los bloques de clases. Estas tablas contienen valores binarios. El 0 representa que el bloque no está ocupado mientras que el 1 indica que si lo está. El objetivo principal de este trabajo consiste en fundamentar la creación de un programa en lenguaje ensamblador MIPS que permita comparar los bloques de clase entre dos horarios para saber qué bloques son iguales y cuáles no, con el fin de agilizar el proceso de asignación de cursos. Los números ingresados deben ser de 8 o menos bits. En caso de que sean menos de 8 bits de debe completar con ceros los bits restantes. Como objetivo secundario es lograr interiorizarnos con la herramienta Latex para la realización de este informe.

II. Antecedentes

Los conceptos necesarios para realizar este laboratorio son los siguientes:

II-A. MARS

"MARS es un entorno de desarrollo interactivo (IDE por sus siglas en inglés) para programación en lenguaje ensamblador MIPS, destinado a uso de nivel educativo".[1] "El lenguaje ensamblador depende directamente de la arquitectura del computador. Por tanto cada arquitectura tiene su propio lenguaje ensamblador".[2] Este IDE permite programar en el lenguaje asociado a las arquitecturas MIPS.

II-B. MIPS

Es un lenguaje ensamblador el cual es la representación simbólica de la codificación binaria del computador (lenguaje de máquina). [2]

II-C. Registros de memoria

MIPS dispone , entre otros, de los siguientes registros: 32 registros en la CPU, cada uno de 32 bits. Contiene 32 registros en la unidad de coma flotante, cada uno de 32 bits. Un contador de programa (PC) de 32 bits, que indica, al principio de cada ciclo, la dirección de memoria de la instrucción del programa que se va a ejecutar.[3]

II-D. Sistema Binario

"Es el sistema numérico de las computadoras o aparatos electrónicos en su mayoría. Dicho sistema numérico sólo cuenta con dos elementos : 1 y 0. Al igual que el sistema decimal se denomina como base 10, este sistema binario se fundamenta en tener como base el número 2".[4]

"La unidad más elemental de información en el interior de un ordenador es un valor binario (0 ó 1). Esta unidad elemental de información se denomina BIT (BInary uniT). Un bit representa la información correspondiente a la ocurrencia de un suceso de dos posibilidades (opciones) distintas".[5]

II-D1. Operador XOR: El operador xor opera 2 binarios bit a bit y genera como resultado un nuevo binario modificado. "El resultado de aplicar el operador xor en cada posición es 1 si el par de bits son diferentes y 0 si el par de bits son iguales".[6]

II-E. RAM

"La memoria es un dispositivo electrónico en el que se almacenan datos. Permite operaciones de escritura en las que los datos se almacenan en un lugar dado. La memoria también permite operaciones de lectura en las que dado un lugar, se obtiene el contenido previamente almacenado en dicho lugar. A este tipo de memoria se le conoce como memoria RAM (Random Access Memory)".[7] Ese lugar al que se hace referencia corresponde a las direcciones de memoria que son valores en sistema hexadecimal (base 16) que representan los distintos espacios en la RAM.

III. Materiales y métodos

III-A. Materiales

1. Sistema operativo: Windows 8.1 y 10
2. Procesador: Intel Pentium (4 nucleos) y Intel Core i5-4310u
3. MARS (ensamblador de MIPS)
4. Software LaTeX, editor Overleaf

III-B. Métodos

El programa lee 2 binarios que son ingresados por el usuario bit a bit, son alojados en las direcciones 0x100100a0 y 0x100100c0, y muestra el resultado alojado en 0x100100e0 por consola.

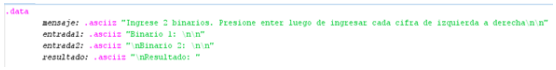

Figura 1: (Data segment + mensajes mostrados al usuario)

```
#Instrucciones de entrada
li $v0, 4
la $a0, mensaje
syscall
```

Figura 2: (Muestra en consola una instrucción)

Este bloque de código se repite cada vez que se requiera imprimir un texto en pantalla.

```
#pedido
li $v0, 5
syscall
```

Figura 3: (Se usa para que el usuario ingrese datos por consola)

Como este código se utiliza repetidamente se crea un procedimiento que consiste en una etiqueta.

```
#Guarda el bit ingresado en $t0
jal pedirValor
jal BitApropiado
```

Figura 4: (Instrucciones de salto)

```
pedirValor:

    #pedido
    li $v0, 5
    syscall

    jr $ra
```

Figura 5: (Instrucción jr que ejecuta el regreso)

```
#Valores posibles 0 o 1
addi $t1, $zero, 0
addi $t2, $zero, 1
```

Figura 6: (Se cargan los valores 0 y 1 en los registros $t0 y $t1, con el fin de que no haya otro valor)

Luego en el procedimiento llamado ”BitApropiado” comprueba si el bit ingresado es igual a 1 o a 0. Si no se cumple, se imprime un mensaje de error y el programa termina al darle un valor de 10 al registro $v0. Si se cumple se accede a un nuevo procedimiento llamado ”CargarEnTemporal” el cual mueve el valor desde el registro $v0 al registro $t0.

Una vez el dato se encuentra en $t0, se carga la dirección de memoria del primer número binario que corresponde a 0x100100a0 en el registro $s0 y luego se carga el contenido de $t0 en 0($s0) usando la instrucción ”sb”.

Nota : el valor que acompaña al registro corresponde a un desplazamiento a través de la memoria RAM. Este valor se debe ir aumentando en 4 para avanzar a la siguiente posición de memoria.

Luego, el procedimiento es repetir el mismo flujo para cada bit:

Recibir en $v0. Comprobar si es 0, 1 o distinto. Mover a $t0. Cargar en data segment

Para el segundo binario se usa la dirección 0x100100c0 como referencia. Una vez que se encuentran ambos binarios en el data segment, se llevan nuevamente a registros del procesador mediante la instrucción ”lb”que funciona de forma semejante a ”sb”. Ahora cada bit del binario1 se encuentra en los registros $t0 - $t7 mientras que los del binario2 están en los registros $s0 - $s7. Previo a ello, se cambia de posición las direcciones de memoria cargadas en $s0 y $s1 hacia $a0 y $a1 para no perderlas.

Lo siguiente es operar bit a bit con el operador ”xor”, el cual viene incluido como instrucción en el lenguaje MIPS. Se guarda el resultado en los registros $t0 - $t7 sobreescribiendo el binario 1 y luego se utiliza la instrucción ”sb”nuevamente para guardar el resultado bit a bit en la dirección 0x100100e0.

Estando el resultado alojado en los registros $t0 - $t7 se procede a enseñarlo por consola. Para ello, se crea el procedimiento ”ValorResultado” el cual le da el valor 1 a $v0 indicando al sistema que imprima un valor entero.

Finalmente el valor se muestra en pantalla.

IV. Resultados

Esta sección presenta los resultados obtenidos, siguiendo el mismo orden de las etapas descritas en la sección III.

Se ha llegado a crear un programa que realiza una operación xor entre dos binarios ingresados por el usuario.

Indicaciones de uso: Al ingresar cada binario se requiere que el usuario oprima la tecla enter cada vez que ingresa el valor de un bit. Teoricamente se aceptan únicamente los valores 1 y 0.

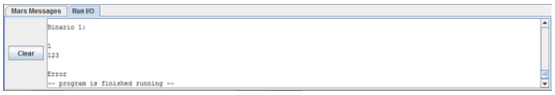

Figura 7: (Manejo de errores, programa finaliza si hay un dato que no es 1 o 0)

Error al ingresar una .ª”

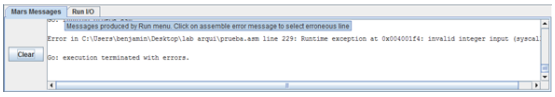

Figura 8: (Restricción, la ejecución finaliza con errores)

Otra restricción es que el programa no es capaz de identificar un binario de menos de 8 bits dejando al usuario la responsabilidad de rellenar con ceros cuando sea necesario.

Al ingresar como binario 1: 01010000 y como binario 2: 10100000 da como resultado: 11110000

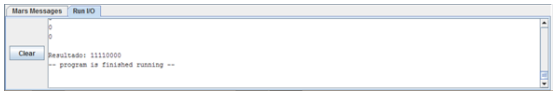

Figura 9: (Ejemplo)

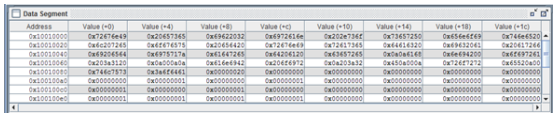

Figura 10: (Operandos y resultado en las direcciones de memoria indicadas)

V. Conclusiones

Se ha logrado cumplir con el objetivo principal de crear un programa funcional que lee 2 binarios, aplica el operador xor y devuelve el resultado por consola.

Por otro lado, hay objetivos específicos que no se han logrado como el hecho de no completar con ceros un binario de entrada de menos de 8 bits o no permitir como entrada datos que no sean enteros.

Además, se ha logrado dar a conocer los fundamentos detrás del desarrollo de un programa en el lenguaje MIPS, se ha aprendido a grandes rasgos el uso del entorno de desarrollo MARS, el uso de registros del procesador y el uso de memoria RAM.

También, se han aprendido conceptos básicos como son el bit, los sistemas numéricos, menmoria, entre otros.

Se podría mejorar buscando una forma más efectiva de aprender los elementos de MARS y la variedad de instrucciones con las que cuenta el lenguaje ensamblador MIPS.

Referencias

[1] M. university, “An ide for mips assembly language programming),” in *MARS (MIPS Assembler and Runtime Simulator*, Recuperado 20 de Abril de 2024. [Online]. Available: http://courses.missouristate.edu/kenvollmar/mars/

[2] A.Serrano and L.Rincon, “Programador ensamblador mips,” in *Estructura y tecnologia de computadores*, Recuperado el 21 de Abril de 2024, pp. 1–24.

[3] J.Gomez, “ensablador mips,” in *Estructura de computadores*, 1998, pp. 1–31.

[4] M. T. J. RAMÍREZ, “Sistema numérico binario,” in *Sistemas Numéricos*, 2020, pp. 1–31. [Online]. Available: https://repositorio.utn.ac.cr/bitstreams/8e8c99f5-ff4c-40f6-8f8a-2b5b38ddb300/download

[5] L. A. . T. J. C. Prieto A., “Introducción a la informática (vol. 20),” in *Introducción a la Informática (Vol. 20)*, 1989.

[6] H. R. . R. D. G. González, “UtilizaciÓn de operadores a nivel de bit en aplicaciones web de domÓtica con raspberry pi,” in *UTILIZACIÓN DE OPERADORES a NIVEL DE BIT EN APLICACIONES WEB DE DOMÓTICA CON RASPBERRY PI*, 2016. [Online]. Available: https://www.researchgate.net/profile/Henry-Gonzalez-Brito/publication/301359298_USING_OF_OPERATORS_BITWISE_IN_WEB_APPLICATIONS_OF_HOME_AUTOMATION_WITH_RASPBERRY_PI/links/571598ad08ae8ab56695b1eb/USING-OF-OPERATORS-BITWISE-IN-WEB-APPLICATIONS-OF-pdf

[7] A. Pardo, “Representación de datos en memoria,” in *Representación de Datos en Memoria*, Recuperado el 21 de Abril de 2024, pp. 1–12. [Online]. Available: http://www.it.uc3m.es/luis/fo2/Memory-1x2.pdf