Informe 1 - Laboratorio de Arquitectura de Computadores: Aritmética de antaño.

Sebastián Cassone (20 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
sebastian.gonzalez.cass@usach.cl

***Resumen*—En este informe se presenta el desarrollo de un programa en MIPS que permite realizar las operaciones aritméticas básicas (suma, resta, multiplicación y división) utilizando un formato especial utilizado por una antigua civilización. El programa fue desarrollado cumpliendo con el enunciado del laboratorio 2 de Arquitectura de Computadores, como parte de un estudio arqueológico para comprender mejor la forma en que esta civilización realizaba cálculos matemáticos en su vida diaria. Se detallan las restricciones impuestas en el desarrollo del código, así como el formato de números utilizado y el flujo de proceso del programa. Los resultados obtenidos demuestran la viabilidad de realizar operaciones aritméticas en el formato antiguo utilizando técnicas alternativas a las instrucciones tradicionales de multiplicación, división, suma, y resta.**

***Palabras claves*—aritmética, lenguaje ensamblador MIPS, memoria MIPS, y MARS.**

I. Introducción

En el marco de un estudio arqueológico sobre una antigua civilización, se ha desarrollado un programa en MIPS que permite realizar operaciones aritméticas utilizando un formato especial utilizado por los habitantes de esta civilización. El objetivo principal de este programa es comprender mejor cómo se realizaban los cálculos matemáticos en la vida diaria de esta cultura antigua. El desarrollo del código se basa en restricciones impuestas para simular las limitaciones técnicas del equipo obsoleto utilizado por los arqueólogos. Estas restricciones incluyen la exclusión de instrucciones de multiplicación, división y desplazamiento, así como la limitación del número de dígitos en los operandos. En este informe, se detalla el formato de números utilizado, las operaciones aritméticas implementadas y el flujo de proceso del programa.[1]

II. Antecedentes

Durante el desarrollo del laboratorio 1, se tuvieron en cuenta los siguientes elementos claves para llevar a cabo la solución al problema planteado:

II-A. Multiplicación por Sumas

La multiplicación por sumas es un método para calcular el producto de dos números enteros sin utilizar la operación de multiplicación directa. En su lugar, se basa en la idea de realizar sumas sucesivas de un número para obtener el resultado final. La multiplicación por sumas se puede representar mediante la siguiente ecuación con sumatoria:

$$\text{Resultado} = \sum_{i=1}^{n} \text{Número}_1 \quad \text{(sumado } n \text{ veces)}$$

Donde:

- Resultado es el producto de los números.
- Número_1 es el número que se va a sumar.
- n es la cantidad de veces que se realiza la suma.

Este método es útil cuando no se dispone de instrucciones de multiplicación[2]

II-B. División por Restas

La división por restas es un método para calcular el cociente y el residuo de una división entre dos números enteros utilizando únicamente operaciones de resta repetidas. Este método se basa en la idea de restar repetidamente el divisor del dividendo hasta que no sea posible seguir restando sin obtener un número negativo. El cociente se obtiene contando el número de restas realizadas, mientras que el residuo es el valor final obtenido.

La división por restas se puede representar mediante la siguiente ecuación:

$$\text{Dividendo} = \text{Divisor} \times \text{Cociente} + \text{Residuo}$$

Donde:

- Dividendo es el número que se va a dividir.
- Divisor es el número por el cual se divide el dividendo.
- Cociente es el resultado de la división.
- Residuo es el valor restante después de la división.

Es importante destacar que la división por restas puede ser un método ineficiente en comparación con los algoritmos de división más avanzados, especialmente para números grandes. Sin embargo, puede ser útil en situaciones donde no se dispone de instrucciones de división o se busca una implementación simple y comprensible.[3]

II-C. Conversión de números hexadecimales a decimales

La conversión de números hexadecimales a decimales es un proceso mediante el cual se convierte un número representado en base hexadecimal (base 16) a su equivalente en base decimal (base 10). Esto implica asignar valores a los

dígitos hexadecimales (0-9, A-F) y realizar cálculos para obtener el valor decimal correspondiente.

En el sistema hexadecimal, se utilizan 16 símbolos para representar los valores del 0 al 15. Los primeros diez símbolos representan los valores del 0 al 9, mientras que los seis símbolos restantes (A, B, C, D, E, F) representan los valores del 10 al 15.

La conversión de un número hexadecimal a decimal se realiza multiplicando cada dígito hexadecimal por la potencia de 16 correspondiente a su posición y sumando los resultados. [4]

La conversión de un número hexadecimal a decimal se realiza utilizando la siguiente ecuación:

$$\text{decimal} = \sum_{i=0}^{n-1} \text{dígito}_i \times 16^{n-1-i} \tag{1}$$

Donde:

- decimal es el número decimal resultante.
- dígito_i es el dígito hexadecimal en la posición i (comenzando desde el dígito más significativo).
- n es la cantidad de dígitos en el número hexadecimal.

II-D. MIPS

MIPS (Microprocessor without Interlocked Pipeline Stages) es una arquitectura de conjunto de instrucciones (ISA) utilizada en procesadores de tipo RISC (Reduced Instruction Set Computer). Proporciona un conjunto de instrucciones simples y eficientes que permiten realizar operaciones básicas de manera rápida y eficiente. MIPS se utiliza ampliamente en aplicaciones embebidas, dispositivos móviles y sistemas integrados. [5] [6]

II-E. IDE MARS

MARS (MIPS Assembler and Runtime Simulator) es un entorno de desarrollo integrado (IDE) diseñado específicamente para la programación en lenguaje ensamblador MIPS. Proporciona un editor de código, un ensamblador, un depurador y una simulación del entorno de ejecución de un procesador MIPS. MARS es ampliamente utilizado para aprender y enseñar la programación en MIPS, así como para desarrollar y depurar programas MIPS. [7]

II-F. Subrutinas

Las subrutinas, también conocidas como funciones o procedimientos, son bloques de código reutilizables que realizan una tarea específica dentro de un programa. En MIPS, las subrutinas se implementan utilizando la pila y las instrucciones de salto y retorno. Una subrutina se llama utilizando la instrucción ”jal”(jump and link), que guarda la dirección de retorno en el registro de enlace ($ra) y salta a la dirección de inicio de la subrutina. Una vez que se completa la subrutina, se utiliza la instrucción ”jr”(jump register) para volver a la dirección de retorno guardada en $ra. [8] [9]

II-G. CPU

La CPU (Unidad Central de Procesamiento) es el componente principal de un sistema informático que ejecuta las instrucciones de un programa y realiza las operaciones aritméticas y lógicas. En MIPS, la CPU está basada en una arquitectura RISC y consta de varios componentes, incluyendo el registro de instrucciones (PC), la unidad de control, la unidad aritmético-lógica (ALU) y los registros. [8] [10]

II-H. Segmentacion de Data de Mips

La segmentación de datos en MIPS es un enfoque utilizado para organizar y gestionar la memoria de datos en un programa MIPS. En este enfoque, la memoria de datos se divide en diferentes segmentos o secciones, cada uno destinado a un tipo específico de datos, como variables globales, variables locales, arreglos, pilas, entre otros. Cada segmento tiene un propósito y un tamaño definido, lo que permite una gestión eficiente de la memoria.

Es importante destacar que MIPS es una arquitectura de procesador y no define explícitamente una segmentación de datos. Sin embargo, el concepto de segmentación de datos se utiliza en la programación de MIPS para organizar y acceder a los datos de manera estructurada.[11] [12]

II-I. Registros

Los registros son ubicaciones de almacenamiento de alta velocidad que se encuentran dentro de la CPU y se utilizan para almacenar datos temporales y resultados de operaciones. [5][9] En MIPS, hay varios tipos de registros, incluyendo:

- Registros de propósito general (como $t0, $t1, $s0, etc.) que se utilizan para almacenar valores temporales y variables.
- Registro de enlace ($ra), que almacena la dirección de retorno al llamar a una subrutina.
- Registro de estado (flags), que almacena información sobre el resultado de operaciones aritméticas y lógicas.
- Registros de coma flotante (como $f0, $f1, $f2, etc.) que se utilizan para realizar operaciones con números de punto flotante.

III. Materiales y métodos

Como su nombre lo indica, esta sección se descompone en dos subsecciones: materiales y métodos.

III-A. Materiales

Para el desarrollo de la actividad, se utilizaron los siguientes recursos computacionales y herramientas:

- Un entorno de desarrollo o editor de código compatible con el lenguaje de programación ensamblador MIPS.
- Un compilador o ensamblador MIPS para traducir el código ensamblador a instrucciones de máquina ejecutables en un procesador MIPS.

III-B. Métodos

El procedimiento desarrollado consta de varias secciones en el código fuente. A continuación, se describen brevemente los métodos empleados en cada sección:

III-B1. Declaración de datos (`.data`): En esta sección, se definen los mensajes utilizados para el menú que solicita al usuario que ingrese la operación deseada, así como el mensaje para mostrar el resultado.

III-B2. Código principal (`.text`): El código principal del programa se encuentra en la sección `.text`. Aquí se definen las subrutinas y se invocan según la opción seleccionada por el usuario.

III-B2a. Función principal (`main`):

1. Se invoca la subrutina `entrada` para solicitar al usuario que elija una operación.
2. Se invocan las subrutinas `cargarEnMemoriaOperandoUno` y `cargarEnMemoriaOperandoDos` para obtener los operandos ingresados por el usuario y almacenarlos en la memoria.
3. Se limpian los registros temporales llamando a la subrutina `limpiarRegistrosTemporales`.
4. Se mueven los valores de los registros $s4 y $s5 a los registros $t1 y $t2 para realizar las operaciones.
5. Se realizan las operaciones de suma, resta, multiplicación y división llamando a las subrutinas correspondientes: `sumaOperacion`, `restaOperacion`, `multiplicacionOperacion` y `divisionOperacion`.
6. Se invoca la subrutina `cambiarSignoMultiplicacion` para cambiar el signo del resultado en caso de haber realizado una multiplicación.
7. Si la operación no es una división, se muestra el resultado llamando a la subrutina `imprimirResultado`.
8. Se guarda el resultado entero en la memoria llamando a la subrutina `guardarResultadoEnMemoria`.
9. Se guardan los decimales en la memoria llamando a la subrutina `guardarDecimalesEnMemoria`.
10. Finalmente, se invoca la subrutina `exit` para finalizar el programa.

III-B3. Subrutina `guardarResultadoEnMemoria`: Esta subrutina se encarga de guardar el resultado entero en la memoria en formato decimal. Utiliza un enfoque de división repetitiva para obtener los dígitos individuales del resultado y los almacena en posiciones de memoria consecutivas.

III-B4. Subrutina `guardarDecimalesEnMemoria`: Esta subrutina se encarga de guardar los decimales del resultado en la memoria en formato decimal. Utiliza una división repetitiva para obtener los dígitos decimales y los almacena en posiciones de memoria consecutivas.

III-B5. Subrutina `imprimirResultado`: Esta subrutina muestra el resultado en la consola utilizando la llamada al sistema syscall con el código de servicio 1 para imprimir un entero. El resultado se encuentra en el registro $t5.

III-B6. Subrutina `cambiarSignoMultiplicacion`: Esta subrutina cambia el signo del resultado en caso de que la operación realizada sea una multiplicación. Utiliza una operación de resta para cambiar el signo del resultado almacenado en el registro $t5.

III-B7. Finalización del programa (`exit`): Esta subrutina se utiliza para finalizar el programa. Limpia la pila y utiliza la llamada al sistema syscall con el código de servicio 10 para terminar la ejecución del programa.

Este programa en MIPS es capaz de realizar operaciones aritméticas básicas y mostrar el resultado obtenido en la consola. También guarda el resultado en memoria y almacena los decimales en otra ubicación de memoria.

IV. Resultados

En esta sección se presentan los resultados obtenidos durante la ejecución del programa. Se realizaron cálculos pertinentes a la suma, resta, multiplicación y división de dos números enteros. El usuario al iniciar el programa se le solicita una operación en consola, luego que el usuario selecciona la operación que desea se le muestra por consola. Además, el programa tiene los siguientes resultados para asegurar un buen resultado el programa:

1. La subrutina `guardarResultadoEnMemoria` parece ser responsable de guardar el resultado entero en memoria en una ubicación específica.
2. La subrutina `guardarDecimalesEnMemoria` parece ser responsable de guardar los decimales en memoria en otra ubicación específica.
3. La subrutina `imprimirResultado` parece ser responsable de imprimir el resultado (entero) por pantalla.
4. Las subrutinas `restaOperacion`, `sumaOperacion`, `multiplicacionOperacion` y `divisionOperacion` realizan las operaciones correspondientes según la opción seleccionada.
5. La subrutina `cambiarSignoMultiplicacion` parece cambiar el signo del resultado en caso de que sea necesario después de una multiplicación.
6. La subrutina `absoluto` calcula el valor absoluto de un número.
7. La subrutina `division` realiza la división entre dos números.

V. Conclusiones

El desarrollo del programa en MIPS para realizar operaciones aritméticas en el formato utilizado por una antigua civilización ha sido un paso significativo en el estudio arqueológico de esta cultura. A pesar de las restricciones impuestas en el código, se ha logrado implementar con éxito las operaciones de suma, resta, multiplicación y división utilizando técnicas alternativas a las instrucciones tradicionales de multiplicación, división y desplazamiento. Esto ha permitido comprender mejor cómo se llevaban a cabo los cálculos matemáticos en la vida diaria de esta civilización antigua. El programa desarrollado ha demostrado la viabilidad de realizar operaciones aritméticas

en el formato antiguo y ha brindado información valiosa sobre las técnicas y métodos utilizados por esta cultura para realizar cálculos matemáticos. Este estudio combinado de arqueología y programación ha enriquecido nuestro conocimiento sobre la aritmética antigua y ha contribuido a preservar y comprender mejor el legado de esta civilización.

REFERENCIAS

[1] U. de Santiago de Chile, "Arquitectura de computadores: Laboratorio 1," s.f. [Online]. Available: https://uvirtual.usach.cl/moodle/mod/resource/view.php?id=856143

[2] GCFGlobal.org, "Multiplicación con sumas y restas," https://edu.gcfglobal.org/es/multiplicar/multiplicacion-con-sumas-y-restas/1/.

[3] ——, "División con suma y resta," https://edu.gcfglobal.org/es/dividir/division-con-suma-y-resta/1/, recuperado el Fecha de acceso.

[4] "Binary, decimal, octal, hexadecimal conversion chart," Online, n.d. [Online]. Available: https://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html

[5] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2011.

[6] R. Britton, *MIPS Assembly Language Programming*. Rampant TechPress, 2003.

[7] M. Team, *MARS MIPS Simulator*, Missouri State University, 2020. [Online]. Available: http://courses.missouristate.edu/KenVollmar/MARS/

[8] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, 2017.

[9] A. S. Tanenbaum and A. S. Woodhull, *Operating Systems: Design and Implementation*. Prentice Hall, 2006.

[10] V. C. Hamacher, Z. G. Vranesic, and S. A. Zaky, *Computer Organization and Embedded Systems*. McGraw-Hill Education, 2011.

[11] D. Harris and S. L. Harris, *Digital Design and Computer Architecture*. Morgan Kaufmann, 2013.

[12] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2011.