Práctica de Laboratorio 2

Wendell Polo C.I 31648447 and Santiago Blanca C.I 33148530

Universidad de Carabobo

Facultad de Ciencias y Tecnología
Departamento de Computación

24 de febrero de 2026

Objetivo

Familiarizar al estudiante con la sintaxis del lenguaje ensamblador MIPS32 y su ejecución en el entorno MARS, resolviendo problemas clásicos de programación, prestando especial atención a la gestión de registros, llamadas a funciones, estructuras de control, camino de datos, manejo de la pila y configuración/utilización del ambiente de trabajo.

1 ¿Qué diferencias existen entre registros temporales ($t0–$t9) y registros guardados ($s0– $s7) y cómo se aplicó esta distinción en la práctica?

La principal diferencia entre estos dos tipos de registros nace a partir de su persistencia durante llamadas a subrutinas, los del tipo $t0-$t9 son utilizados para operaciones aritmeticas rapidas internas del algoritmo o para guardar valores temporales o auxiliares, no son conservados en el llamado a subrutinas, y no se necesita de ellos para la salida.

Por otra parte, los del tipo $s0-$s7 son usados para cumplir el rol de almacenar valores que deben conservarse a traves de las diferentes llamadas a subrutinas, son seguros para mantener datos que recibimos al inicio del algoritmo, y en caso de necesitar operarlos, guardamos su valor inicial de la pila para no perderlo.

2 ¿Qué diferencias existen entre los registros $a0–$a3, $v0–$v1, $ra y cómo se aplicó esta distinción en la práctica?

Primeramente, los registros $a0–$a3 son utilizados como variables de entrada para las subrutinas llamadas, los datos que ingresas y seran operados, en cambio los registros $v0–$v1 son utilizados para almacenar valores de retorno luego de terminar la función. En cambio el registro $ra(Return

Adress) como su nombre nos dice se ocupa de guardar la dirección de retorno y así al terminar la función o procedimiento, volver al punto de llamada original, lo cual nos ayuda en funciones anidadas o recursivas, fundamental para regresar de las llamadas luego de terminar de operar.

Dentro de esta práctica , al trabajar con algoritmos de ordenamiento, hicimos uso de los registros $a para guardar la direccion de memoria de los vectores a ordenar, y de esta manera tratarlos dentro de la función, además como directamente modificamos el vector y no necesitamos de una variable de salida, los registros del tipo $v fueron usados principalmente para hacer uso del syscall, y que dependiendo del valor asignado solicita un servicio del sistema operativo. Por otra parte las instrucciones $ra fueron fundamentales para no crear recursiones infinitas en el caso del QuickSort, al guardar los valores de este registro dentro de la pila, nos permite regresarnos luego de ordenar el vector a la dirección que tenga el registro durante esa llamada.

3 ¿Cómo afecta el uso de registros frente a memoria en el rendimiento de los algoritmos de ordenamiento implementados?

El acceso a banco de registros es más rapido que el acceso al banco de memoria, por lo que los que hacen menos uso de la memoria, tendran un mejor rendimiento ya que no dependen tanto de esa etapa en el camino de datos. En estos algoritmos de ordenamiento, el uso de acceso a memoria se basa principalmente para realizar el swap entre dos indices del vector para ordenarlo, para guardar los valores de los indices en la pila, haciendo mas uso de la memoria en este tipo de algoritmos que uso de registros.

4 ¿Qué impacto tiene el uso de estructuras de control (bucles anidados, saltos) en la eficiencia de los algoritmos en MIPS32?

Al realizar un algortimo en MIPS32 se debe hacer un uso correcto o mínimo de saltos y bucles debido a que al repetir subrutinas se duplica la cantidad de instrucciones generadas, afectando el rendimiento del algoritmo. Siempre debemos utilizar la menor cantidad de saltos y bucles anidados para evitar generar instrucciones de más.

5 ¿Cuáles son las diferencias de complejidad computacional entre el algoritmo Quicksort y el algoritmo alternativo? ¿Qué implicaciones tiene esto para la implementación en un entorno MIPS32?

Primeramente la complejidad en tiempo del QuickSort es de O(n log n) y del BubbleSort es $O(n^2)$, al ingresarle vectores pequeños a estos dos algoritmos nos damos cuenta que el Bubblesort genera menor cantidad de instrucciones, pero al someterlos a vectores grandes, se nota la diferencia del QuickSort, generando una menor cantidad notable con respeto al otro. Al ser un entorno MIPS32 es más complejo implementar un algoritmo recursivo como el QuickSort debido a la manera en que se debe estructurar la recursión, en cambio el BubbleSort es un algoritmo mucho más simple de implementar.

6 ¿Cuáles son las fases del ciclo de ejecución de instrucciones en la arquitectura MIPS32 (camino de datos)? ¿En qué consisten?

6.1 IF

Búsqueda de instrucción: Se obtiene la instrucción de la memoria utilizando la dirección del **PC**, el cual se incrementa en 4 para apuntar a la siguiente instrucción.

6.2 ID

Descodificación: Se interpreta el tipo de instrucción y se leen simultáneamente los operandos del banco de registros.

6.3 EX

Ejecución: La ALU realiza la operación aritmética/lógica propiamente dicha o calcula la dirección efectiva en caso de accesos a memoria .

6.4 MEM

Acceso a memoria: Solo si la instrucción lo requiere (`lw` o `sw`), se lee un dato o se escribe en la memoria de datos.

6.5 WB

Escritura: El valor resultante (ya sea de la ALU o de la memoria) se escribe de vuelta en el banco de registros.

7 ¿Qué tipo de instrucciones se usaron predominantemente en la práctica (R, I, J) y por qué?

Podemos observar en ambos algoritmos el tipo de instrucción que predomina es el tipo I, ya que trabajamos con vectores, en ambos se utiliza mucho las instrucciones sw y lw para acceder a la dirección de memoria del vector y poder operarlo, en el caso del Quicksort también se utiliza para acceder a la pila y reservar su espacio, esto nos permite la recursión. En las figuras a continuación se utilizaron vectores desordenados de 6 elementos para ambos

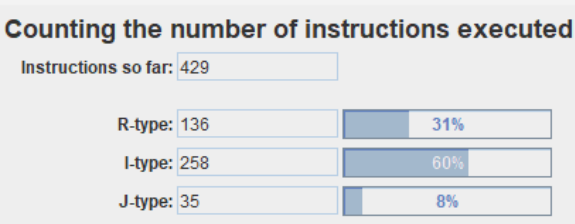

Figure 1: QuickSort, Numero de Instrucciones

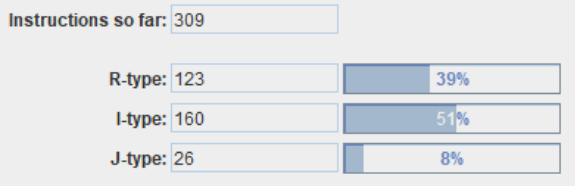

Figure 2: BubbleSort, Numero de Instrucciones

8 ¿Cómo se ve afectado el rendimiento si se abusa del uso de instrucciones de salto (j, beq, bne) en lugar de usar estructuras lineales?

Se nota principalmente el bajo rendimiento cuando se utiliza excesivamente los saltos ya que estos nos llevan a repetir grupos de instrucciones, generando un mayor tiempo de ejecución en el algoritmo, además el procesador trabaja de mejor manera al hacer uso de estructuras lineales ya que el procesador no tiene que decidir si cambiar o no el flujo del programa.

9 ¿Qué ventajas ofrece el modelo RISC de MIPS en la implementación de algoritmos básicos como los de ordenamiento?

El modelo RISC de MIPS prioriza instrucciones simples, con un tamaño fijo de 32 bits, y en este tipo de algoritmos basicos, se utiliza mucho las operaciones entre registros, ya que debemos comparar constantemente los valores del vector ,además el uso de memoria solo se centra en lw y sw. Al ser algoritmos con pocas variables de entrada , los 32 registros de MIPS son mas que suficientes para implementar estos algoritmos de ordenamiento.

10 ¿Cómo se usó el modo de ejecución paso a paso (Step, Step Into) en MARS para verificar la correcta ejecución del algoritmo?

Fue utilizado para comprobar que valores tomaban los registros durante la ejecución del algoritmo, este nos permite saber que instrucción se esta operando en el momento y que valores va tomando cada registro durante esa instrucción, de esta manera verificamos que el algoritmo estaba operando de manera correcta.

11 ¿Qué herramienta de MARS fue más útil para observar el contenido de los registros y detectar errores lógicos?

La herramienta del Step, Step into fue la más util ya que podiamos ejecutar línea por línea y en el modulo que nos muestra los registros podíamos consultar los valores mientras se iba ejecutando el algoritmo. También en el data segment nos permite ver como va cambiando el vector que estamos ordenando durante la ejecución.

12 ¿Cómo puede visualizarse en MARS el camino de datos para una instrucción tipo R? (por ejemplo: add)

En MARS, el camino de datos para una instrucción tipo R como `add` se visualiza mediante la herramienta "Data Path and Control", ejecutando la instrucción paso a paso. Se observa el flujo desde el PC hasta la ALU y finalmente el banco de registros, sin intervención de memoria de datos, lo que confirma el comportamiento típico de las instrucciones tipo R en la arquitectura MIPS32.

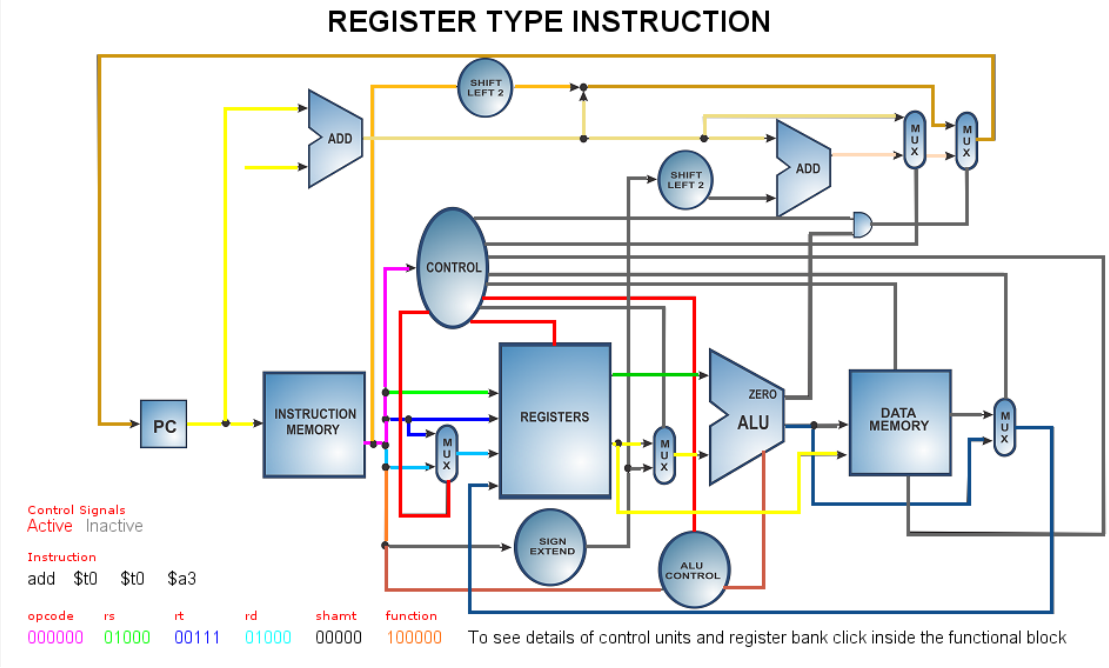

Figure 3: Camino de datos para la instruccion add en MARS

13 ¿Cómo puede visualizarse en MARS el camino de datos para una instrucción tipo I? (por ejemplo: lw)

En MARS, el camino de datos para una instrucción tipo I como `lw` se visualiza mediante la herramienta "Data Path and Control", ejecutando la instrucción paso a paso. Se observa el flujo desde el PC hasta la ALU luego al banco de registros, finalmente termina en la memoria de datos, y aumenta 4 en el PC para la siguiente instrucción. Lo que confirma el comportamiento típico de las instrucciones tipo I en la arquitectura MIPS32.

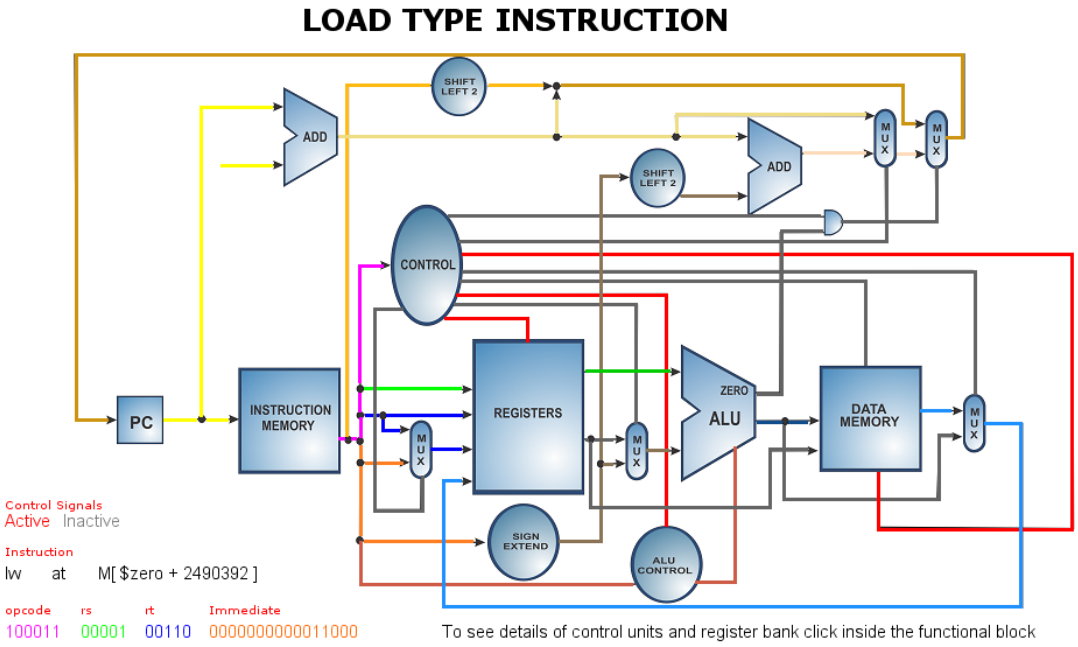

Figure 4: Camino de datos para la instrucción lw en MARS

14 Justificar la elección del algoritmo alternativo

Principalmente por su simplicidad al implementarlo en MIPS32, es un algortimo facil de leer en este entorno y es un algoritmo familiarizado con el hemos practicado anteriormente, y es un ejemplo esencial para entender algoritmos de ordenamiento.

15 Análisis y Discusión de los Resultados

Probamos diferentes entradas para ambos algoritmos, para ver el comportamiento de los dos con entradas pequeñas o grandes, dandonos cuenta que mientras mas grande la entrada la eficiencia del QuickSort se hace notar en comparación al BubbleSort. Además, en caso de usar como entrada un vector ordenado de manera creciente o decreciente con el QuickSort, este aumenta mucho su tiempo de ejecución cambiando de O(nlogn) a $O(n^2)$. Por otra parte utilizamos el enfoque iterativo del BubbleSort que al igual que su versión recursiva es de $O(n^2)$ pero al no tener que reservar tanto espacio en pila, es mas eficiente.