Practica de Laboratorio 2 de Jesús Rachadell 31421494

1. ¿Qué diferencias existen entre registros temporales ($t0-$t9) y registros guardados ($s0-$s7) y comó se aplicó está distinción en la práctica?

Técnicamente no poseen diferencias en términos de hadware, ya que ambos cumplen y pueden cumplir la función del otro, la distinción se hace respecto a la convención por la cual se ha decidido optar para establecer un estándar en el lenguaje, los $t0-$t9 son responsabilidad del llamador, una vez llamado a una función estos registros pueden modificarse de cualquier forma y es algo que el que llama debe tener en cuenta, en cambio los $s0-$s7 son responsabilidad del llamado, la función que se llama debe preservar estos registros, normalmente se usan la pila $sp para este propósito

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

Los registros $a0-$a3 son registros técnicamente iguales a $t y $s, pero en la convección poseen una función distintiva, se usan comúnmente para guardar los parámetros de una función, $a0 tiene la cualidad en MARS de imprimirse cuando se hace un *syscall* cuando $v0 está en modo imprimir. los registros $v0-$v1 se usan para retornar valores de las funciones hasta 64bits, $v0 como se mencionó antes posee la función especial de que cuando se ejecuta syscall, el sistema busca en $v0 como un ID para saber que función debe ejecutar. $ra es un registro especial el cual al usar *jal* este guarda la dirección de la instrucción siguiente, esto se usa para poder volver al punto de la llamada.

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

En la mayoría de algoritmos de ordenamientos se requiere iterar una gran cantidad de veces, esto implica chequear datos muy seguido, es por esto que el uso de registros que son internos al CPU pueden ayudar a reducir la carga en el runtime, ya que no debe chequear una dirección en memoria que es más lento, es por eso que usar registros para cosas como indices, pivotes entre otros ayuda al rendimiento de los algoritmos.

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

Como en un algoritmo ejecutado en C o cualquier otro lenguaje, el usar bucles anidados se incrementa exponencialmente el número de instrucciones ejecutadas, haciendo que en grandes n el programa se haga muy lento. Los saltos representan una categoría aparte, al hacer un salto el programa para un momento para cambiar el program counter (PC) lo cual ralentiza un poco.

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?

Yo opté por realizar el algoritmo InsertionSort como el segundo método de ordenamiento. Quicksort en la notación Big O representa un $O(n^2)$ en su peor caso, por otro lado el InsertionSort representa a su vez un $O(n^2)$ en el peor caso, la diferencia recae en el caso promedio donde Quicksort posee un $O(nlog(n))$ pero InsertionSort posee un $O(n^2)$. Esto implica que en la mayoría de casos InsertionSort sea más lento que usar QuickSort, sin embargo QuickSort requiere una mayor gestión del $sp, además que necesitará ocupar mucha mayor memoria que InsertionSort.

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

Consta de 5 fases, la primera es IF, en esta fase se consulta al Program Counter(PC) sobre la dirección de memoria de la próxima instrucción, luego sigue ID, en esta fase se consulta el banco de registros y se decodifica la instrucción, luego va EX, en esta fase se ejecuta la instrucción si se usa *lw* o *sw* se calcula el offset, luego sigue MEM si la instrucción no requiere el uso de memoria esta fase se salta, en esta fase se consulta a la memoria usando *lw* o *sw*, por ultimo va WB una vez realizados los cálculos se guardan.

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

Se usaron más el tipo I sin embargo está casi empatado con el uso del tipo R, diría que la principal razón es que las instrucciones tipo I son más frecuentes en las estructuras de control, y a su vez los algoritmos de la practica necesitan el uso de dichas estructuras para funcionar.

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?

Cada salto requiere un paso extra en el pipeline lo que hace que cada vez que se use una instrucción que requiera cambiar el PC, hace que el programa más lento, si se anidan muchas de este tipo instrucciones se puede empezar a notar en el rendimiento. en una estructura lineal al no necesitar salto siempre irá de punto A a punto B haciéndolo más rápido

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

La mayor ventaja diría que es la abundancia de registros ya que estos permiten reducir el tiempo que tardaría consultar a la memoria y permiten mantener datos como los indices y pivotes todos guardados en registros de fácil y rápido acceso. Otra ventaja es que gracias al formato de instrucción, casi toda instrucción puede realizarse en un mismo ciclo de reloj, lo cuál aumenta la velocidad del programa.

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

Principalmente en el Debugging es bastante útil ya que se puede visualizar que caminos toma, que variables cambian, sí una variable toma la dirección o por el contrario toma otro valor, etc. La herramienta es muy útil para saber en que punto un programa roto empieza a fallar, haciendo posible el cambio en especifico.

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

Diría que la mejor herramienta de MARS es la ejecución paso a paso, ya que como mencioné en la respuesta anterior es muy útil para el debugging permitiendo ubicar un fallo de manera precisa.

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

En MARS existe la herramienta MIPS X-ray que ayuda en la visualización del camino de datos, para la instrucción add, lo primero que hace es leer el opcode y lo manda para para el nodo de control, de ahí activa las señales necesaria para la ejecución, de ahí toma el camino común de hacer la operación y escribir en el registro de llegada.

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

Para las instrucciones del Tipo I, primero se recoge el registro de la instrucción, luego se pasa por la extensión de signo al número inmediato para mandarlo junto al registro a la ALU, dependiendo de la instrucción se toman diferentes caminos, para lw, la ALU calcula el offset y lo envia a la memoria, y el nodo de control ya habiendo activado MemToReg, guarda el resultado provieniente de la memoria en el registro objetivo.

14. Justificación de elección del algoritmo alternativo

Mi principal motivo de elegir InsertionSort fue simplemente que era uno de los algoritmos de ordenamientos que me faltaba por hacer, además me pareció que su diseño simple y fácil de entender, contrarrestaba el Quicksort al ser este recursivo, tomando diferentes enfoques.

15. Análisis y Discusión de los resultados

Ambos algoritmos cumplen su función, como se mencionó antes el Quicksort será más rápido que el Insertionsort en la mayoría de casos, y aunque con arreglos pequeños no se note casi la diferencia, con arreglos grandes la brecha de eficiencia será cada vez más grande.