Tecnológico de Costa Rica

Ingeniería en Computadores

Arquitectura de Computadores I

Tarea 1: ISA

Estudiante:
Malcolm Davis

Profesor:
Fabián Zamora

13 de septiembre de 2017

Resumen

Este trabajo se desarrolló para reforzar los conceptos vistos en clases sobre un set de instrucciones. Se resolvieron 2 preguntas dónde la primera consta de una comparación de implementación de diferentes arquitecturas con tablas, y la segunda es la implementación del algoritmo de ordenamiento bubble sort en ensamblador ARM(ARMv4).

1. Pregunta 1

Elementos de un set de instrucciones (ISA) [40 puntos]. En clase se estudiaron los diversos elementos que se toman en cuenta a la hora de diseñar un set de instrucciones y cómo las diferentes opciones en cada uno tienen sus ventajas y desventajas. Estos elementos se muestran en la Cuadro 1 para la arquitectura MIPS64. En este ejercicio debe desarrollar dos tablas, una para la arquitectura x86 y una para la arquitectura ARMv4. Para ambas debe tomar como referencia la Cuadro 1. Cada tabla de incluir por cada elemento del ISA un resumen de las decisiones que se tomaron para cada arquitectura.
El resultado de esta pregunta se puede ver en los Cuadros 2 y 3.

2. Pregunta 2

Ordenamiento de burbuja ("Bubble Sort") [60 puntos]. "Bubble Sort" es un algoritmo de ordenamiento que funciona revisando cada elemento de la lista que va a ser ordenado con el siguiente, intercambiándolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada. En este ejercicio debe desarrollar una implementación del algoritmo "Bubble Sort" utilizando el lenguaje ensamblador ARM (ARMv4) en el programa VisUAL.

(a) **Consideraciones generales:**

(i) El tamaño de la lista (N) es mayor o igual que 1.

(ii) La lista contiene únicamente enteros.

Elementos del ISA	MIPS64
Clase de ISA	• RISC • Load/Store
Registros	• 32 registros de 64-bits para enteros • 32 registros de 64-bits para puntos flotantes
Tipos de datos	• Bytes de 8 bits • Media palabra de 16 bits • Palabra de 32 bits • Doble palabra de 64 bits
Modos de direccionamiento	• Inmediato • Desplazamiento • Registro indirecto (desplazamiento con 0)
Organización de memoria	• Address space: 2^64 • Addressability: 8 bits • Todos los accesos a memoria deben estar alineados. • Un bit para indicar si Big-Endian o Little-Endian. • Los accesos a memoria pueden ser a nivel de byte, media palabra, palabra y doble palabra.
Formato de Instrucciones	• Tres tipos: I-type, R-type, J-type
Operaciones	• Load/Store • Operaciones de ALU • Branch & Jump • Operaciones de punto flotante
Encodificación	• Instrucciones de tamaño fijo: 32 bits • El modo de direccionamiento se guarda en el opcode.

Cuadro 1: Resumen de elementos de MIPS64

(III) La lista por ordenar se encuentra en memoria. La dirección de inicio de la lista se debe obtener de una etiqueta ("label") llamada "list". Ejemplo:
list DCD 9,8,7,6,5,4,3,2,1

(IV) El tamaño de la lista por ordenar (N) se encuentra en memoria. Este se debe obtener de una etiqueta ("label") llamada "list_count". Ejemplo:
list_count DCD 9

(v) El resultado (la lista ordenada) se debe almacenar en memoria, reemplazando la lista desordenada original.

(b) **Entregables:**

(i) Código fuente generado por VisUAL de la implementación del algoritmo "Bubble Sort".

(ii) El código debe ser funcional (no debe contener errores) y debe incluir un ejemplo (list y list_count) con el cual ser probado. El código fuente debe contener un encabezado (comentarios dentro del código) con:

(1) Nombre y número de carné del estudiante.

(2) Breve explicación de cómo implementó el algoritmo.

(3) La cantidad de registros y el uso que se le dio a cada uno.

(4) La cantidad de instrucciones ejecutadas para ordenar la lista: [9,8,7,6,5,4,3,2,1].

Los entregables para este problema se adjuntan en un archivo llamado bubble.S.

Elementos del ISA	X86	Justificación
Clase de ISA	• CISC • Registro/Memoria	La arquitectura x86 normalmente utiliza esta clase de ISA.
Registros	• 64 registros de 64-bits de propósito general	Al elegir la arquitectura CISC se quiere poder hacer accesos a memoria y que se pueda operar con un registro de propósito general.
Tipos de datos	• Bytes de 8 bits • Media palabra de 16 bits • Palabra de 32 bits • Doble palabra de 64 bits	Estos tipos de datos se eligieron para poder representar diferentes tipos de datos (int, float...)
Modos de direccionamiento	• Inmediato • Desplazamiento • Indexado	Se eligieron estos por la facilidad de desplazamiento a una posición en memoria a partir de un punto(para arreglos).
Organización de memoria	• Address space: 2^{64} • Addressability: 8 bits • Todos los accesos a memoria deben estar alineados. • Solo funcionará con little endian por estándar. • Los accesos a memoria pueden ser a nivel de byte, media palabra, palabra y doble palabra.	Al representar tipos de datos de byte, media, doble y palabra entera se debe de poder accesar con esos tamaños a memoria. El espacio y la addressabilidad se Eligen por el tamaño de datos también. Y los accesos se alinean para evitar accesos indeseados.
Formato de Instrucciones	• CISC con instrucciones variables	Aumenta la velocidad de decodificación pero disminuye el tamaño del código.
Operaciones	• Operaciones de ALU • Branch & Jump • Operaciones de punto flotante	Se eligen las operaciones necesarias para el flujo de un programa.
Encodificación	•CISC con instrucciones variables • El modo de direccionamiento se guarda en el opcode.	Aumenta la velocidad de decodificación pero disminuye el tamaño del código.

Cuadro 2: Resumen de elementos de x86

Elementos del ISA	ARMv4	Justificación
Clase de ISA	• RISC • Load/Store	La arquitectura ARM normalmente utiliza esta clase de ISA.
Registros	• 32 registros de 64-bits para enteros • 32 registros de 64-bits para puntos flotantes	Las arquitecturas RISC buscan tener abuntantes registros para aumentar la velocidad de ejecución(menos accesos a memoria).
Tipos de datos	• Bytes de 8 bits • Media palabra de 16 bits • Palabra de 32 bits	Se elimina la doble palabra para simplificar las instrucciones.
Modos de direccionamiento	• Inmediato • Desplazamiento • Registro indirecto (desplazamiento con 0)	Se utilizan estos para poder hacer operaciones sobre registros(registro registro).
Organización de memoria	• Address space: 2^64 • Addressability: 8 bits • Todos los accesos a memoria deben estar alineados. • Un bit para indicar si Big-Endian o Little-Endian. • Los accesos a memoria pueden ser a nivel de byte, media palabra, palabra.	Se elimina el acceso de doble palabra para poder simplificar las instrucciones y delimitarlas a 64bits.
Formato de Instrucciones	• Tres tipos: I-type, R-type	Se definen los necesarios para el flujo del programa, se eliminan los tipo j porque en ARM no se usan jumps solo branches.
Operaciones	• Load/Store • Operaciones de ALU • Branch • Operaciones de punto flotante	Se definen los necesarios para el flujo del programa. se eliminan los jumps porque en ARM no se usan.
Encodificación	• Instrucciones de tamaño fijo: 64 bits • El modo de direccionamiento se guarda en el opcode.	Se elije este tamaño por el tipo de datos.

Cuadro 3: Resumen de elementos de ARMv4