

Preguntas de Examen

1. Parcial 1 - 2024

1.1. Pregunta 1

Explique y describa los elementos que constituyen el conjunto de instrucciones de la arquitectura. De acuerdo con la cantidad de operandos, ¿Cómo se clasifican las instrucciones? ¿Cuáles son las ventajas de las instrucciones con tres operandos y registros de propósitos generales comparadas con de dos operandos en registro y uno en memoria? Justifique

Los elementos que constituyen el conjunto de instrucciones de la arquitectura (ISA) son:

- **Conjunto de Instrucciones:** son las instrucciones que puede realizar el procesador, las operaciones de control, aritméticas, lógicas y de movimiento de datos.
- **Tipos de Datos:** se refiere al tipo de datos con los que puede operar (bit, byte, word, nibble) y a su formato (enteros, flotantes, etc).
- **Organización de la Memoria:** es la forma en que está dividida la memoria, los lugares donde se encuentra el programa, la información de configuración, dirección de reset y además el modelo de entrada y salida de periféricos.
- **Gestión de Eventos:** es la forma de resolver interrupciones y excepciones (eventos ajenos al programa) durante la ejecución del programa. Las interrupciones se dan en forma sincrónica con la ejecución, mientras que las excepciones se dan de forma asincrónica, debido por ejemplo a un dispositivo que requiere la atención del procesador.
- **Operandos y Modos de Direccionamiento:** son los operandos que pueden usar las instrucciones (registros, inmediatos, direcciones de memoria, puertos) y las formas de recuperarlos. En este sentido, los operandos pueden estar contenidos en la misma instrucción, pueden estar en una dirección de memoria o registro indicado en la instrucción, estar implícitos según la arquitectura (por pila, por acumulador) o una combinación de estos mecanismos.
- **Modos de Operación:** son las formas en las que se puede interactuar con la máquina, los modos de operación del procesador limitan los recursos a los que puede acceder un programa. Un programa en modo usuario podrá acceder a recursos limitados que no pongan en riesgo la integridad del sistema, en modo supervisor podrá acceder directamente a la memoria de configuración del sistema y en modo hipervisor podría administrar máquinas virtuales.

Las instrucciones se puede clasificar según la cantidad de operandos en:

- **Implícito:** la memoria se organiza en pilas y se opera sobre el top de estas sin la necesidad de operandos. Ejemplo: las instrucciones `push 1`; `push 2`; `+` dan como resultado `1+2` cuyo resultado también es guardado en una pila.
- **Explícito:** en estas las operaciones son entre UN operando y un acumulador, guardando el resultado en el mismo acumulador.
- **Registros de propósito general:** en este diseño todos los registros pueden utilizarse para todas las instrucciones, y las instrucciones tienen dos campos para operandos y uno para el resultado. Estos operandos y resultado pueden tener referir a registros con el valor (direccionamiento directo) o a registros con la dirección en memoria donde se encuentra el valor (direccionamiento indirecto).

Las instrucciones con tres operandos y registros de propósitos generales tienen dos operandos de entrada y uno de salida (resultado), donde se direcciona a los mismos usando la direcciones de los registros que los almacenan.

Las instrucciones de dos operandos en registro y uno en memoria poseen un operando de entrada y el resultado por

registro, mientras que otro de los operandos de entrada está en memoria.

Por esto, las primeras requieren menos tiempo a la hora de recuperar los operandos, ya que el movimiento de datos (operando en memoria) es más costoso.

1.2. Pregunta 2

Explique cómo funciona una microarquitectura segmentada. Explique sus ventajas y problemas, ¿Cuál es la arquitectura de computadora óptima para su implementación? Fundamente. ¿Qué son los riesgos de datos? Explique cómo se resuelven. ¿En qué se diferencia de una microarquitectura superescalar? ¿Qué nuevos bloques se agregan en esta microarquitectura?

La microarquitectura más simple de implementar es la monociclo, en la cual una instrucción se ejecuta al finalizarse la anterior siguiendo el *orden de programa*.

La microarquitectura segmentada plantea la separación de la ejecución de la instrucción en etapas (fetch, decode, op fetch, execute, memory read/write, write-back) que puedan funcionar de forma separada. De esta forma, una vez una instrucción termine una etapa, esos recursos quedan disponibles para iniciar la ejecución de la instrucción siguiente, y el ciclo de reloj será igual a la *etapa* más lenta.

La arquitectura más óptima para implementarla es la arquitectura RISC (Reduced Instruction Set Computer) ya que tienen instrucciones con longitud fija, son más fáciles de decodificar.

Los riesgos de datos son situaciones en las que una instrucción requiere un dato que aún no está disponible. Se suelen clasificar en:

- Read After Write (Lectura después de Escritura): cuando una instrucción quiere leer un dato que está siendo escrito por otra.
- Write After Read (Escritura después de Lectura): una instrucción requiere escribir un dato que otra está leyendo
- Write After Write (Escritura después de Escritura): una instrucción requiere escribir un dato que otra está escribiendo.

Estos Riesgos se pueden resolver de distintas maneras, entre ellas están:

- Adelanto de Datos: cuando se requiere un dato que ya ha sido calculado (execute) pero aún no se ha escrito (write-back), se puede "adelantar" el dato, haciendo que esté disponible para su lectura en una instrucción que lo necesite para continuar su ejecución.
- Detención: cuando no se puede adelantar el dato, es necesario que se espere a que la instrucción devuelva el dato requerido. Para esto existen dos alternativas.
 - Bubble: se introducen instrucciones nop entre la instrucción que tiene el dato y las siguientes.
 - Flush: se vacía el pipeline de instrucciones posteriores a las que tiene el dato.

La microarquitectura superescalar lo que logra es aumentar el paralelismo a nivel de máquina, aumentando la cantidad de recursos (Unidades Funcionales y registros). Esto permite al procesador leer múltiples instrucciones para ejecutar en simultaneo grupos de instrucciones independientes, en unidades funcionales donde a su vez múltiples instrucciones pueden ejecutarse en diferentes etapas, donde a su vez se puede implementar una microarquitectura segmentada. La microarquitectura segmentada requiere añadir registros entre cada etapa del pipeline para almacenar los valores devueltos por cada una. En el caso de la superescalar, se incrementan el número de unidad de ejecución y registros

1.3. Pregunta 3

Explique cómo funciona una microarquitectura multihilo. Explique cuáles son sus ventajas y problemas. ¿Es lo mismo que una microarquitectura multinúcleo? ¿En qué se diferencian? Fundamente.

La microarquitectura superescalar aumenta los recursos del procesador (unidades de ejecución y registros) para leer múltiples instrucciones y encontrar aquellas que puedan ejecutarse en simultáneo. El problema de esto es que frecuentemente las unidades de ejecución se verán detenidas por instrucciones que necesitan datos aún no calculados, por lo que no se estarán aprovechando los recursos provistos por la arquitectura.

La microarquitectura multihilo viene a solucionar este problema ejecutando múltiples hilos (programas) en los distintos pipelines que provee la arquitectura superescalar. De esta forma nos aseguramos la independencia de las instrucciones y el aprovechamiento de los recursos.

Para lograr esto, se deben agregar algunos elementos:

- Mecanismo de permutación de hilos para los pipelines.
- Identificación de hilo en las tablas de saltos, para evitar predicciones fantasmas entre hilos.
- Aumento de recursos como caché y registros para evitar colisiones y errores.

Además de esto, existen distintas formas de gestionar los recursos entre los hilos

- Multihilo grueso: cada datapath es asignado a un hilo por un tiempo (variable o fijo) hasta que este se encuentra con un evento extenso, y es conmutado por otro hilo listo para ejecutarse. Aunque es el más sencillo, es poco eficiente a la hora de gestionar conmutaciones por bloqueos más cortos que los que se pierde por el vaciado del pipeline.
- Multihilo entrelazado: en este, el datapath va rotando entre los hilos listos para ejecutarse.
- Multihilo simultáneo: en este, cualquier etapa de cualquier pipeline puede asignarse a cualquier instrucción sin importar el hilo al que corresponda. Esto aumenta considerablemente el aprovechamiento de los recursos, aunque requiere un aumento del hardware y una complejidad en la gestión.

2. Parcial 1 - 2023

2.1. Pregunta 1

Ver [pregunta 1 de parcial 1 2024](#)

2.2. Pregunta 2

Ver [pregunta 2 de parcial 1 2024](#)

3. Parcial 1 R - 2023

3.1. Pregunta 1

¿Qué son los riesgos de control? Explique cuáles son las posibles soluciones. Explique cómo funciona un predictor estático. Explique cuáles son sus ventajas y problemas. ¿Cuáles son los lugares donde se pueden implementar los predictores? ¿En qué se diferencia una predicción estática de una dinámica?

Cuando se implementa una arquitectura segmentada (separación del datapath en etapas independientes), hay que lidiar con *situaciones en las que la instrucción a iniciar en el pipeline depende de una instrucción que aún no se ha resuelto*. Es decir, situaciones en las que dependiendo del resultado de una instrucción puede darse un salto que cambie las instrucciones que se van a ejecutar. Estos son los riesgos de control.

Para evitar o resolver estas situaciones existen las siguientes alternativas:

- Detención: esperar que se resuelva el salto para continuar con la ejecución de las instrucciones correctas.
- Predicción: intentar predecir si el salto será tomado o no y, en caso de predecir mal, vaciar el pipeline y continuar con el orden correcto.

A la hora de predecir los saltos del programa (para evitar riesgos de control), se puede hacer uso de un predictor estático. Esta es la forma más simple de predictores, donde se determina de antemano qué saltos va a tomar y cuáles no, es decir, no toma decisiones en tiempo de ejecución. Se clasifican en:

- Predictor Fijo: los saltos se toman siempre o no.
- Predictor por opcode: los saltos se toman dependiendo del tipo de instrucción. Esto se basa en que hay ciertos tipos de saltos que son más probables de tomar, como pueden ser los relacionados a bucles while/for.
- Predictor dirigido por compilador: el compilador hace un análisis del código y genera *sugerencias* de predicción basadas en el código analizado.

Estos predictores podemos implementarlos en diferentes etapas del pipeline, para ahorrar el mayor tiempo posible a la hora de detectarlo y continuar con la ejecución de otras instrucciones.

- Durante la Fetch: se detecta el salto al tiempo que se lee la memoria de instrucciones.
- Después de Fetch: se detectan las instrucciones de salto en el buffer de instrucciones antes de ser decodificadas.
- Durante el Decode: se introduce un decodificador dedicado a detectar saltos antes del final de la etapa de decodificación.

Estos predictores son simples y fáciles de implementar, pero con una baja tasa de predicciones correctas, a diferencia de los predictores dinámicos, que requieren más hardware pero son más fiables.

3.2. Pregunta 2

Explique qué es y qué elementos constituyen el conjunto de instrucciones de la arquitectura (ISA). Describa los elementos más importantes del mismo. De acuerdo con la cantidad de operandos ¿Cómo se clasifican las instrucciones? ¿Cuáles son las ventajas de las instrucciones con tres operandos y registros de propósitos generales (arquitectura Register-Register) comparadas con las de dos operandos en registro y uno en memoria (arquitectura Register-Memory)?

Ver [pregunta 1 de parcial 1 2024](#)