Preguntas de Examen

1. Parcial 1

1.1. Parcial 1 - 2024

1.1.1. 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 asicrónica, debido por ejemplo a un dispositiva 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 tambien 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.

1.1.2. 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 un 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 despues 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 dónde 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

1.1.3. 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 simultaneo. 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 simultaneo: en este, cualquier etapa de cualquier pipeline puede asignarse a cualquier intrucción sin importar el hilo al que corresponda. Esto aumenta considerablemente el aprovechamiento de los recursos, aunque requiere un aumento del hardware y un complejidad en la gestión.

1.2. Parcial 1 - 2023

1.2.1. Pregunta 1

Ver pregunta 1 de parcial 1 2024

1.2.2. Pregunta 2

Ver pregunta 2 de parcial 1 2024

1.3. Parcial 1 R - 2023

1.3.1. 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.
- Despues 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 un baja tasa de predicciones correctas, a diferencia de los predictores dinámicos, que requieren más hardware pero son más fiables.

1.3.2. 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

1.4. Parcial 1 - 2022

No guiarse por este parcial, fue especialmente imposible.

1.4.1. Pregunta 3

Explique cómo funciona un procesador con ejecución multihilo. Realice un diagrama de bloque básico que ilustre la microarquitectura de un procesador de este tipo (jaja no) ¿De dónde obtiene la mejora en la ejecución de instrucciones? ¿Cuáles son los tipos de ejecución multihilo que existen? ¿Cómo funcionan? ¿Cuáles son sus ventajas e inconvenientes?

Un procesador con ejecución multihilo implementa múltiples flujos de ejecución (datapaths) para ejecutar en simultaneo múltiples hilos (programas). A diferencia de la microarquitectura superescalar, no ejecuta grupos de instrucciones (streams) del mismo programa en datapaths distintos, ya que las detenciones por espera de datos hacen que no se aprovechen los recursos provistos. La microarquitectura multihilo gestiona los recursos entre distintos hilos (programas independientes) y así aprovecha los recursos. Existen distintas formas de gestionar los recursos entre los distintos hilos.

- Multihilo Temporal: el datapath permuta entre hilos cada cierto tiempo (fijo o variable) de acuerdo a las dependencias y recursos presentes.

    - Multihilo Grueso: ejecuta un hilo hasta encontrarse con evento de latencia alta (detencion, flush, lectura/escritura en memoria, etc) y lo permuta por un hilo listo para ejecutarse. El problema de este enfoque es que las permutaciones por bloqueos cortos son más costosas que el esperar a que se resuelva el problema, ya que se debe vaciar el datapath.
    - Multihilo Entrelazado: ejecuta cada hilo por un tiempo determinado y sigue una rotación entre los hilos listos para ejecutar.
- Multihilo Simultaneo: en esta implementación cualquier *etapa* del datapath puede ser asignada a cualquier hilo, aprovechando así al máximo los recursos. Es muy compleja de implementar, ya que es dificil gestionar las etapas de las instrucciones de cada hilo, además de necesitar un identificador de hilo por cada etapa y mayores recursos (como memoria) para evitar colisiones de datos y errores entre hilos.

En todas estas implementaciones es necesario implementar un identificador de hilo a cada entrada del predictor de saltos, para evitar predicciones fantasmas.

2. Parcial 2

2.1. Parcial 2 - 2023

2.1.1. Pregunta 1

Explique cómo funciona la memoria cache de una computadora y para qué se la utiliza. ¿Cuántos niveles de cache hay en una computadora? ¿Están organizados de la misma manera? Explique. Explique los mecanismos de mapeo de datos en una cache. ¿Cuáles son sus ventajas y desventajas? Explique cuáles son las fuentes de error en una caché

La memoria caché es una memoria volátil de gran velocidad aunque capacidad de almacenamiento limitada, muy cercana al núcleo del procesador. Se utiliza para reducir el costo en tiempo y/o energía de acceder a la información en la memoria principal, aprovechando la localidad temporal y espacial. Esta memoria almacena copias de los datos a los que se accede frecuentemente. De esta forma, reduce la brecha entre el procesador y la memoria, aumentando el desempeño general de la computadora.

La caché se implementa como un bloque de memoria para almacenamiento temporal de datos. Consiste en un conjunto de *entradas* asociadas a datos a través de *etiquetas*. Se organiza en S *conjuntos*, que mapean uno o más bloques de memoria. El número de bloques B en un conjunto determina el *grado de asociatividad* $N = B/S$. A su vez la cantidad de bloque en el conjunto está determinada por la capacidad de almacenamiento C del conjunto y el tamaño de bloque b: $B = C/b$.
Para direccionar la caché se utilizan direcciones parciales de memoria: los bits superiores determinan el bloque de memoria, mientras que los otros bits determinan el desplazamiento en ese bloque. Además, la *traducción* de la dirección puede incluir el cálculo de la dirección física, en donde es importante tener en cuenta:

- Latencia: la dirección física estará disponible por la MMU ciclos después de estar disponible la dirección virtual por el generador de direcciones.
- Aliasing: varias direcciones virtuales pueden hacer referencia (estar asignadas a) la misma dirección física, por lo que el procesador debe garantizar que exista una única copia de la dirección en la caché, para que todas las actualizaciones en esa dirección se realicen en el orden del programa.
- Granularidad: el espacio de direcciones virtuales se divide en páginas (no se que tendra que ver, pero ok).
  La función de mapeo es la que determinará la asignación de los bloques de la memoria principal a las entradas de la caché. Para esto existen tres tipos:
- Mapeo directo: se asigna una entrada de caché para cada bloque. Para esto necesita mantener tres datos: los datos de caché, la etiqueta de caché y el estado del bloque (contiene un bloque válido o no). El problema de este es

que siempre habrá más bloques de memoria que entradas de caché, por lo es propenso a conflicto, situaciones donde dos o más datos comparte el mismo conjunto.

- Mapeo asociativo: asigna un *conjunto* de líneas de caché a cada bloque de memoria.
- Mapeo asociativo completo: todos los bloque de la caché están en un solo conjunto.

Las *operaciones* en caché consisten en:

- Lectura: no importa el origen del dato. Si se encuentra en caché, se lee desde esta, en caso contrario se lee desde la memoria.
- Escritura: esta operación requiere que escribamos tanto en la entrada de la caché como en el dato en memoria.

Existen distintas cachés, clasificadas en niveles según la cercanía al núcleo del procesador, y donde cada nivel funciona como repositorio común para el nivel superior. A medida que nos alejamos del núcleo, la capacidad de la caché aumenta, asi como disminuye su velocidad:

- Nivel 1 (L1): la más cercana al núcleo, dividida en un bloque para *datos* y otro para *instrucciones*. Es la más pequeña.
- Nivel 2 (L2): se ubica dentro del procesador y no se divide como la L1. Cada núcleo posee su propia caché L2, que no comparte datos con las demás.
- Nivel 3 (L3): se ubica dentro del procesador, y es compartida entre núcleos.
- Nivel 4 (L4): se ubica fuera del procesador y en la memoria principal, es la de mayor capacidad.

Los errores en caché pueden deberse a una lectura o escritura fallidas, resultando en un acceso a la memoria principal, de latencia alta. Se clasifican en:

- Compulsory: al acceder por primera vez a un bloque, debe ser llevado a la cache desde la memoria principal. Se llama fallo de *arranque en frio* o de *primera referencia*.
- Capacity: el conjunto de trabajo del programa es mayor que la capacidad de la caché, por lo que los bloque se descartan.
- Conflict: en mapos asociativos o directos, si varios bloque se asignan al mismo conjunto se produce un fallo de *colisión* o de *interferencia*. Si se utiliza mapeo directo, la entrada simplemente es reemplaza, pero para mapeo asociativo es necesario una política especial de reemplazo. Estas pueden ser:
    - First in - First Out: la caché se comporta como una cola, se desaloja el bloque al que se accedió primero, como para mapeo directo.
    - Last in - First Out: al contrario que la FIFO, el bloque desalojado es el último al que se accedió.
    - Least Recently Used (LRU): se descarta el bloque menos usado recientemente. Obviamente requiere de un seguimiento de la vez que se usó por última vez cada bloque.
    - Time aware Least Recently Used (TLRU): un caso especial de la LRU, que tiene en cuenta el tiempo de vida de los datos almacenados. Se introduce una marca de tiempo que indica el tiempo de uso del dato.
    - Most Recently Used (MRU): descarta el bloque más recientemente usado.
    - Pseudo-LRU (PLRU): versión menos costosa de LRU.
    - Random Replacement (RR): se selecciona un bloque al azar para descartar.

2.1.2. Pregunta 2

Explique cómo funciona la gestión de periféricos por interrupciones. Explique cuáles son sus ventajas y problemas. ¿Cómo se gestionan las interrupciones anidadas? Explique el mecanismo utilizado. ¿Es lo mismo una interrupción que una excepción? Explique en qué se diferencian. ¿Para qué se usan las excepciones?

2.1.3. Pregunta 3

Explique qué es y cómo funciona un puerto serie. Explique cuáles son sus ventajas y desventajas. Explique la principal diferencia entre un puerto serie síncrono y asíncrono. ¿Cuáles son los mecanismos de un puerto asíncrono para lograr la sincronización entre el transmisor y el receptor?

2.2. Parcial 2 R - 2023

2.2.1. Pregunta 1

Explique cómo funciona la memoria cache de una computadora y para qué se la utiliza. ¿Cuáles niveles de cache hay en una computadora? ¿Están organizados de la misma manera? Explique. Explique los mecanismos de mapeo de datos en una cache. ¿Cuáles son sus ventajas y desventajas? Explique cuáles son las fuentes de error en una cache.

Ver pregunta 1 de parcial 2 2023

2.2.2. Pregunta 2

Explique cómo funciona la gestión de periféricos por consulta. Explique cuáles son sus ventajas y problemas. Explique la diferencia entre la gestión por consulta y la gestión por interrupciones. ¿En qué casos utilizaría una o la otra? Explique

2.2.3. Pregunta 3

Explque qué es y cómo funciona un puerto serie. Explique cuáles son sus ventajas y desventajas. Explique la principal diferencia entre un puerto serie síncrono y asíncrono. ¿Cuáles son los mecanismos de un puerto asíncrono para lograr la sincronización entre el transmisor y el receptor?

Ver pregunta 3 de parcial 2 2023

3. Resumen

Temas en los que enfocarse y dónde encontrarlos

3.1. Parcial 1

3.2. Parcial 2

3.2.1. Memoria Cache

- Cómo funciona.
- Para qué se la utiliza.
- Niveles de Cache.
- Mecanismos de mapeo. Ventajas y desventajas.
- Fuentes de error en la cache.

3.2.2. Interrupciones y Excepciones

- Interrupciones anidadas. Mecanismo utilizado.
- Para qué se usan las excepciones.

3.2.3. Gestión de Periféricos

- Gestión por interrupciones. Ventajas y problemas.
- Gestión por consulta. Ventajas y problemas.
- Diferencia entre gestión por consulta y por interrupciones. Casos de aplicación para cada una.

3.2.4. Puerto Serie

- Qué es y cómo funciona.
- Principal diferencia entre síncrono y asíncrono.
- Mecanismos en puerto serie asíncrono para sincronizar transmisor y receptor.