Arquitectura de un procesador

Set de instrucciones de la arquitectura

Definir un set de instrucciones incluye definir diferentes componentes de su estado, establecer las convenciones, y el manejo de excepciones.

Diseño lógico

Hay 3 elementos fundamentales requeridos para implementar un sistema digital:

Lógica combinacional: Llevar a cabo funciones con los bits.

Elementos de memoria: Almacenar bits.

Señales del reloj: Regular la actualizando de los elementos de la memoria.

Circuitos combinacionales: Conjunto de compuertas lógica unidas en una red.

Toda compuerta lógica debe estar conectada con alguno de los siguientes: (1) Una de las entradas del sistema. (2) La conexión de salida de algún elemento de memoria. (3) La salida de alguna compuerta lógica.

Las salidas de dos o más compuertas lógicas no pueden ser conectadas juntas.

La red tiene que ser acíclica, no puede haber un loop en la red.

Un circuito combinacional tiene la propiedad de que las salidas continuamente corresponden a los cambios en las entradas. Si alguna entrada del circuito cambia, la salida también cambiara.

Podemos construir circuitos combinacionales que calculen funciones más complejas. Un ejemplo es la ALU, arithmetic/logic unit.

Memoria y Reloj:

Los circuitos combinacionales por sí solos no almacenan información (simplemente reaccionan ante las señales de las entradas). Para poder crear circuitos secuenciales el sistema necesita almacenar el estado, y hacer operaciones según este, por eso se deben introducir dispositivos que almacenen información en bits.

Hay dos tipos de dispositivos de memoria: (1) Registros cronometrados (clocked registers) guarda bits individuales o words, la señal del reloj controla la carga del registro con el valor de su entrada. (2) Register files, guardan múltiples words, los identificadores de registros sirven de direcciones, por ejemplo, el id 0x15 implica que no se lleva a cabo una lectura ni escritura.

Cabe notar que la palabra “registro” tiene dos significados diferentes cuando hablamos de hardware versus programación de lenguaje máquina. Respecto a hardware, un registro está directamente conectado con el resto del circuito por sus cables de entrada y salida. En cuanto a programación de lenguaje máquina, los registros representan una pequeña colección de palabras direccionables en la CPU, donde las direcciones consisten en ID de registro.

El banco de registros tiene dos puertos de lectura, llamados A (srcA) y B (srcB), y un puerto de escritura llamado W (dstW). Estos puertos permiten múltiples operaciones de escritura y lectura simultaneas. No es un circuito combinacional, ya que tiene almacenamiento interno.

Secuencial:

Procesamiento organizado en etapas, el procesar una instrucción requiere un número de operaciones, entre ellas están:

Fetch (Búsqueda): Se leen las instrucciones de la memoria.

Bloques predefinidos: **Split:** Divide el byte instrucción en icode y ifun. **Aling:** Se encarga de obtener los campos para rA, rB, y valC. **PC:** Registro que contiene el PC. **Memoria de instrucciones.**

Lógica de control: **Instr valid:** Señaliza si la instrucción es válida o no. **Icode – ifun:** Generan no-op si la dirección es invalida. **Need regids:** Señaliza si la instrucción posee un byte de registros. **Need valC:** Señaliza si la instrucción posee un byte constante.

Los valores calculados en esta etapa son: icode (código de instrucción), ifun (función de la instrucción), rA (registro A), rB (registro B), valC (constante) y valP (PC incrementado).

Decode (Decodificación): Se leen los registros del programa.

Banco de registros: Puertos de lectura A y B. Puertos de escritura E y M. Las direcciones son los IDs de los registros 0 15 (0xF).

Lógica de control: srcA, srcB son las direcciones de los puertos de lectura. dstE, dstM son las direcciones de los puertos de escritura.

Señales: Cnd indica si se debe realizar un movimiento condicional o no (Se calcula en la etapa de ejecución).

Los valores calculados en esta etapa son: valE (resultado de la ALU) y Cnd (flag de salto/ movimiento condicional)

Execute (Ejecución): Se calcula el valor o dirección necesaria. Y establece los códigos de condición.

Unidades: **ALU:** Modifica los condition codes. Implementa las 4 operaciones (add, sub, and, xor). **CC:** Registro de 3 bits para las 3 condiciones. **Cond:** Calcula la bandera de saltos/movimientos condicionales.

Lógica de control: **Set CC:** Indica si se debe cargar el registro de código de condiciones. **ALU A:** Selector de entrada A de la ALU. **ALU B:** Selector de entrada B en la ALU. **ALU fun:** Indica la operación que debe realizar la ALU.

Los valores calculados en esta etapa son: srcA (id del registro A), srcB (id del registro B), dstE (registro destino E), dstM (registro destino M), valA (valor del registro A) y valB (valor del registro B).

Memoria (Acceso a memoria): Se leen o escriben datos.

Logica de control: **stat:** Computa el estado de la instrucción. **Mem. Read:** Indica si hay que leer. **Mem. Write:** Indica si hay que escribir. **Mem. Addr:**  Selecciona la dirección. **Mem. Data:** Selecciona la palabra.

Los valores calculados en esta etapa son: valM (valor leído de memoria).

Write-Back: Se actualizan los registros del programa.

PC: Actualiza el Program Counter. Aumenta la cantidad de bytes leídos en la etapa fetch.

Segmentado

Comparado con el diseño secuencial, el segmentado incremento el rendimiento del sistema, pero a coste de una pequeña latencia. La idea del segmentado está en dividir el proceso en etapas independientes. Mover objetos a través de las etapas. En todo momento, múltiples objetos se están procesando.

Cálculos paralelos: Consiste en que ciertos circuitos lleven a cabo cálculos, y se guarden los resultados en registros. Una señal del reloj controla la carga del registro sobre un cierto intervalo de tiempo. El delay se mide en pico segundos, es decir, 10^-12 segundos. El rendimiento se mide en GIPS (giga-instructions per second), el tiempo total requerido para hacer una instrucción la denominamos latencia.

No ocurre ningún cambio en el estado de los registros hasta que el reloj se levante, entonces si el reloj va muy rápido puede tener efectos desastrosos, los valores no tendrían tiempo de pasar por la lógica combinacional, entonces las entradas de los registros no serían validas cuando se levante el reloj.

Lógica de control: Detecta la condición de demora. Establece las señales indicando a los registros como se deben actualizar.

Limitaciones del segmentado: **Particionamiento no uniforme:** El rendimiento es limitado por la etapa más lenta, el desafío está en segmentar en etapas balanceadas. Cuando más se segmenta más pesa la carga de los registros.

Riesgos de segmentación (Hazards): **Estructurales:** Surgen de conflictos de los recursos, cuando el hardware no puede soportar todas las combinaciones posibles de instrucciones en ejecuciones solapadas simultáneamente. **Dependencia de datos:** Surgen cuando una instrucción depende de los resultados de una instrucción anterior, de forma que, ambas, podrían llegar a ejecutarse de forma solapada. **Control (Saltos):** Surgen de la segmentación de los saltos y otras instrucciones que cambian el PC, por ejemplo, el ret.

Estancamiento (**Stalling**): Se trata de que el procesador estanca una o más instrucciones mientras que otro grupo de instrucciones siguen ejecutándose, hasta que la dependencia haya desaparecido, se insertan dinámicamente una instrucción nop (**burbujas**).

Realimentación (**Data Forwarding**): Se puede evitar el uso de stalling simplemente pasando el valor que está a punto de ser escrito en el registro E, esta técnica requiere añadir más conexiones y control lógico en el hardware básico. Fuentes de reenvió: Execute = valE, Memory = valE;valM, Write-Back = valE;valM. La prioridad de envió se establece usando el de la etapa más cercana.

Dependencia de datos: **Load/Use Hazard:** Ocurre cuando la lógica de control detecta que una instrucción que está en la etapa decode requiere el resultado de haber leído de memoria. *Load interlock:* Se usa estancamiento combinado con la realimentación para solucionar el Hazard. **Read After Write (RAW):** Cuando el resultado de una operación se usa como operando en la siguiente.

Etapas segmentadas

**Fetch:** Seleccionar el PC actual. Leer la instrucción. Calcular el PC incrementado.

**Decode:** Leer registros.

**Execute:** Usar la ALU, establecer condition codes.

**Memory:** Leer o escribir datos de/en la memoria.

**Write-back:** Actualizar registros.

Paralelismo

La idea es que se refiere a ejecutar más de una instrucción a la vez, en paralelo. Pero segmentado también se refiere a ejecutar instrucciones en paralelo, pero la diferencia que el segmentado tiene limitado el CPI a 1 como el más mínimo.

Emisión múltiple: Con el CPI limitado a 1, si queremos recudirlo más necesitamos que por cada ciclo de reloj se termine mas de una instrucción, y estos procesadores se llaman de **emisión múltiple**. Si la cantidad máxima de instrucciones que se pueden emitir en un ciclo de reloj es m, entonces se dice que el procesador es de m-vías (**m-issue wide**).

Hay dos categorías de procesador de emisión múltiple, difieren mucho en el hardware, pero ambas parten de una idea fundamental en la división del trabajo, esta dada por como se toma la decisión para la emisión de las instrucciones y define las categorías de emisión múltiple. Si la decisión se toma en tiempo de compilación, es estática, y el procesador es de emisión múltiple estática (**static multiple issue**). Si la decisión se hace mientras se ejecuta el programa, es dinámica, y el procesador es de emisión dinámica (**Dynamic multiple issue**)

Emisión estática: El compilador organiza las instrucciones para ser emitidas juntas, se agrupan las instrucciones en espacios de emisión (**issue slots**), es el compilador quien detecta y evita los riesgos del pipeline (todo lo que puede).

Emisión dinámica: La CPU examina el flujo de instrucciones y decide que instrucciones se emiten en cada ciclo (el compilador puede haber ayudado organizando las instrucciones), la CPU resuelve los riesgos usando técnicas más avanzadas en tiempo ejecución.

Especulación: Basado en la idea de predicción, le permite al procesador o al compilador, dependiendo del tipo de la emisión, “adivinar” sobre las propiedades de una instrucción, también permite que comiencen instrucciones, que en caso contrario tendrían que esperar los resultados certeros, de instrucciones previas. En caso de que la especulación no fuese correcta, se debe deshacer o descartar lo procesado y hacerlo correcto.

**Ambos** tipos de procesadores hacen **uso de la especulación**, y la diferencia entre estos esta en donde se hace y como se resuelve. Ambas estrategias **pueden fallar**, en el caso estático, en el proceso de compilación, se agregan instrucciones adicionales para corroborar los resultados y para generar una rutina que repare el error producido. En el caso dinámico, en el hardware, el procesador usualmente almacena los resultados de la especulación, si esta fue correcta se usan estos resultados, si no, descarta los resultados y ejecuta la secuencia de instrucciones correcta, generalmente los errores dinámicos requieren que se vacié el pipeline. En ambos casos los fallos **degradan el rendimiento**. En estos fallos pueden llegar a **causar excepciones**.

Emisión múltiple estática: El **compilador (responsable)** es quien empaqueta, y reconoce los riesgos. Se puede llegar a pensar a estos paquetes como una única gran instrucción, estos paquetes se llaman **paquete de emisión (issue packet),** este conjunto es limitado. A estos procesadores también se los conoce como como procesadores de paralelismo explicito.

**Scheluding:** Proceso en el cual, el compilador se encarga de reordenar las instrucciones, y las agrupa en paquetes.

En estos paquetes **no pueden hacer dependencias internas, pero si puede haber entre paquetes**, se les debió haber ordenado de tal forma que no existan, esto puede ser con la inserción de nops. Sin embargo, el PROCESADOR si puede tener (o sea, que hay casos en los que no lo implementa, entonces esta responsabilidad cae completamente en el COMPILADOR) la capacidad de detectar riesgos, y detener paquetes, solo paquetes.

Emisión múltiple dinámica: Estos procesadores también son llamados **procesadores super escalares.** En cada ciclo de reloj es el **procesador (responsable)** quien decide cuantas instrucciones se emiten. Si bien estos procesadores no requieren planificación estática, pero podrían beneficiarse mucho por el uso de esta. Se utiliza lo que se llama **planificación dinámica del pipeline (Dynamic pipeline scheduling)**, no solo se decide cuantas instrucciones emitir, sino cuales.

Planificación dinámica del pipeline: Se lleva a cabo reordenando su emisión mientras el código se ejecuta, se divide en 3 grandes unidades:

* Una unidad para hacer el fetch y la emisión de instrucciones: Lee, decodifica y emite las instrucciones a las distintas unidades funcionales. Estas unidades tienen un buffer (**reservation station**) que las almacena.
* Un grupo de múltiples unidades funcionales: Una vez que se obtiene el resultado, se envía a las unidades funcionales que lo necesiten, y a la unidad de commits, y lo guarda en un buffer llamado **reordenamiento (reorder buffer)**.
* Una unidad de commit (**commit unit**): Actualiza los registros, y graba a memoria en el orden de lectura de las instrucciones (**in order commit**).

Se puede pensar que esta planificación organiza el flujo, y el procesador lo ejecuta manteniéndolo, a este modo se le llama **ejecución fuera de orden (out of order execution)**, el caso contrario se llama **ejecución en orden (in order execution)**.

Sumándole la ejecución especulativa, junto a la predicción de saltos, permite continuar con la lectura y ejecución de instrucciones, mientras que la unidad de commit garantiza que no se escribirán datos erróneos, puede especular incluso carga de datos, y cálculo de direcciones.

Eficiencia energética: Mientras mas complejidad, mas transistores, ya que se implementa en hardware, porque el software es mas lento, mas transistores significa mas consumo de energía, entonces se vuelve ineficientes (energéticamente). Por eso, se paso de un procesador complejo, a implementar múltiples procesadores por chip, con pipelines más chatos, y especulaciones menos agresivas.