Arquitectura de Ordenadores

Capítulo 1. Abstracciones, Tecnología y Rendimiento de los Procesadores:

Progresos en tecnología de computadoras, La ley de moore, el número de transistores que se pueden ajustar en un chip de computadora se duplicará cada 18 meses, dando resultado a incrementos periódicos.

**La abstracción** nos ayuda a lidiar con la complejidad y así ocultar detalles de nivel inferior. **Arquitectura del conjunto de instrucciones (ISA):** un conjunto de instrucciones implementadas en hardware, el nombre simbólico y el formato de código binario de cada instrucción. La estructura de una computadora es lo que un programador de lenguaje de máquina (o un compilador) debe entender para escribir un correcto programa para esa máquina. La interfaz de hardware/software.

Organización: Estructuras como rutas de datos, unidades de control, memorias y los buses que interconectan entre ellos.

Tabla

Descripción generada automáticamenteHardware: La lógica, la tecnología electrónica empleada, los diversos diseños físicos y aspectos de la computadora

Algoritmo: determina el número de operaciones ejecutadas

Lenguaje de programación, compilador, arquitectura: determinar el número de instrucciones de máquina ejecutadas por operación

Procesador y sistema de memoria: determinar qué tan rápido se ejecutan las instrucciones

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteSistema de E/S (incluido el sistema operativo): determina qué tan rápido se ejecutan las operaciones de E/S.

Tabla

Descripción generada automáticamenteEl CPU clocking es una operación de hardware digital gobernado por un reloj de tasa constante.

Clock period: duration of a clock cycle n e.g., 250ps = 0.25ns = 250×10–12s Clock frequency (rate): cycles per second n e.g., 4.0GHz = 4000MHz = 4.0×109Hz

Texto

Descripción generada automáticamenteCPU Time = Tcpu:

Rendimiento mejorado por:

Reducción del número de ciclos de reloj. Aumento de la frecuencia del reloj El diseñador de hardware a menudo debe compensar la frecuencia del reloj con Contador de Cíclos.

Texto

Descripción generada automáticamente

Recuento de instrucciones para un programa: determinado por programa, ISA y compilador.

Promedio de ciclos por instrucción: determinado por el hardware de la CPU, si diferentes instrucciones tienen diferentes CPI, CPI promedio afectado por la combinación de instrucciones.

Texto

Descripción generada automáticamenteEjemplo:

Si diferentes clases de instrucción toman diferentes números de ciclos:

Un reloj de aguja

Descripción generada automáticamente con confianza baja  
IPC medio ponderado:

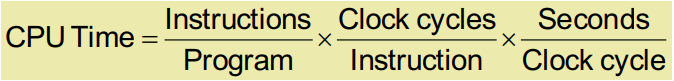
Imagen que contiene Diagrama

Descripción generada automáticamente

Ejemplo:

Tabla

Descripción generada automáticamente



El rendimiento depende de: Algoritmo: afecta IC, posiblemente CPI. Lenguaje de programación: afecta IC, CPI. Compilador: afecta IC, CPI. Arquitectura del conjunto de instrucciones: afecta a IC, CPI, Tc.

SpeedUp y la Ley de Amdahl:

Texto, Carta

Descripción generada automáticamenteSpeedup es la mejora conseguida en el tiempo de ejecución:

Texto

Descripción generada automáticamente con confianza mediaLa Ley de Amdahl se usa para encontrar la aceleración de un sistema cuando solo se mejora una parte del sistema:

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente  
El rendimiento está limitado a la parte sin aceleración del programa. Tiempo de ejecución después de la mejora = Tiempo de ejecución de la parte no afectada + (Tiempo de ejecución de la parte afectada / Cantidad de mejora).

Tabla

Descripción generada automáticamente

Dificultad: MIPS como medida de rendimiento

MIPS: millones de instrucciones por segundo, no tiene en cuenta diferencias en ISA entre computadoras ni diferencias de complejidad entre instrucciones.

Texto

Descripción generada automáticamente

El CPI varía entre los programas en una CPU determinada.

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza bajaMFLOPS: millones de operaciones de punto flotante por segundo

La medida MFLOPS, Millón de Operaciones de coma Flotante por Segundo, se emplea ampliamente debido a su eficiencia y eficacia en términos de simplicidad y sus capacidades indicativas.

Tabla

Descripción generada automáticamenteEjemplo: compare el rendimiento a 100 MHz de M1-CISC y M2-RISC. (También usando MIPS y métricas de MFLOP)

El rendimiento de M1-CISC =1,5 x M2-RISC (50% mejor)

• M1 MIPS < M2 MIPS

• M1 MFLOP > M2 MFLOP

SPEC Punto de referencia de CPU de especificaciones

Programas utilizados para medir el desempeño

Supuestamente típico de la carga de trabajo real

Forma, Rectángulo

Descripción generada automáticamenteCorporación de Evaluación de Desempeño Estándar (SPEC)

Desarrolla puntos de referencia para CPU, E/S, Web,...

ESPEC. CPU2006

Tiempo transcurrido para ejecutar una selección de programas

E/S insignificante, por lo que se centra en el rendimiento de la CPU

Normalizar en relación con la máquina de referencia

Resumir como media geométrica de los índices de rendimiento

CINT2006 (entero) y CFP2006 (coma flotante)

Gráfico, Diagrama

Descripción generada automáticamente

Suponga que una nueva CPU tiene:

85% de la carga capacitiva de la CPU antigua

15% de tensión y 15% de reducción de frecuencia

Imagen que contiene Texto

Descripción generada automáticamente

El poder de la barrera

No podemos reducir más el voltaje

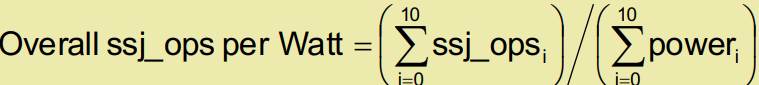
No podemos quitar más calor

¿De qué otra manera podemos mejorar el desempeño?

Punto de referencia de potencia SPEC

Consumo de energía del servidor en diferentes niveles de carga de trabajo n Rendimiento: ssj\_ops/seg n

Potencia: vatios (julios/seg)



Falacia: baja potencia en reposo

Mirar hacia atrás en el punto de referencia de potencia i7

Al 100% de carga: 258W

Al 50 % de carga: 170 W (66 %)

Al 10 % de carga: 121 W (47 %)

Centro de datos de Google

Funciona principalmente con una carga del 10 % al 50 %

Al 100 % de carga menos del 1 % del tiempo

Considere diseñar procesadores para generar energía proporcional a la carga

Multiprocesadores:

Microprocesadores multinúcleo

Más de un procesador por chip

Requiere programación paralela explícita

Comparar con el paralelismo de nivel de instrucción

El hardware ejecuta varias instrucciones a la vez

Oculto para el programador

Difícil de hacer

Programación para el desempeño

Equilibrio de carga

Optimización de la comunicación y la sincronización

Diagrama

Descripción generada automáticamente

Observaciones finales

El costo/rendimiento está mejorando

Debido al desarrollo tecnológico subyacente

Capas jerárquicas de abstracción

Tanto en hardware como en software

Arquitectura del conjunto de instrucciones

La interfaz de hardware/software

Tiempo de ejecución: “la mejor” medida de desempeño

La potencia es un factor limitante

Utilice el paralelismo para mejorar el rendimiento

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ejemplo de tiempo de CPU:

Computadora A: reloj de 2 GHz, tiempo de CPU de 10 s

Computadora de diseño B

Apunte a 6 s de tiempo de CPU

Puede hacer un reloj más rápido, pero causa 1,2 × ciclos de reloj

Texto

Descripción generada automáticamente¿Qué tan rápido debe ser el reloj de la computadora B?

Ley de Amdahl: ejemplo 1

Supongamos que mejoramos una máquina haciendo que todas las instrucciones de punto flotante se ejecuten cinco veces más rápido. Si el tiempo de ejecución de algún benchmark antes del realce de coma flotante es de 12 segundos, ¿cuál será la aceleración será si se gasta la mitad de los 12 segundos ejecutando instrucciones de coma flotante?

Diagrama

Descripción generada automáticamente con confianza baja

Estamos buscando un punto de referencia para mostrar lA nuevo unidad de coma flotante descrita en el ejemplo anterior, y queremos que el punto de referencia general muestre una aceleración de 3. Un punto de referencia que estamos considerando es de 100 segundos con el antiguo hardware de punto flotante. ¿Cómo gran parte del tiempo de ejecución sería de punto flotante instrucciones deben tener en cuenta en este programa para para producir nuestra aceleración deseada en este punto de referencia?

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Mejorar un aspecto de una computadora y esperando una mejora proporcional en rendimiento global

Imagen que contiene Diagrama

Descripción generada automáticamente

Ejemplo: multiplicar cuentas por 80/100 n ¿Cuánta mejora en el rendimiento multiplicado para obtener 5 × en general?

Icono

Descripción generada automáticamente

Corolario: hacer el caso común rápido.

**Capítulo 2. Procesadores segmentados**

Ejecución de instrucciones

Memoria de instrucciones PC®, obtener instrucciones

Números de registro ® archivo de registro, leer registros

Según la clase de instrucción

Use ALU para calcular

Resultado aritmético

Dirección de memoria para cargar/almacenar

Dirección de destino de la sucursal

Acceder a la memoria de datos para cargar/almacenar

PC ¬ dirección de destino o PC + 4

Estudiaremos dos procesadores RISC simples: MIPS vs RISC-V.

Procesador de 32 bits (datos, memoria)

32 registros de propósito general

Memoria separada de datos y códigos (arquitectura Harvard)

´

Diseño de ruta de datos y control

Datapath: Elementos que procesan datos y direcciones en la CPU

Registros, ALUs, mux’s, memorias,…

Construiremos una ruta de datos de forma incremental

Unidad de Control: La información proviene de los 32 bits de la instrucción y las líneas de control seleccionan:

Registros a leer (siempre leer dos).

La operación a realizar por ALU

Si la memoria de datos se va a leer o escribir

Qué está escrito y dónde en el archivo de registro

Qué pasa en la PC

Implementación Combinacional de Ciclo Único

RISC-V está diseñado para canalización

• Instrucciones de la misma longitud

• 32 bits, fácil de buscar y luego decodificar

• 6 tipos de formatos de instrucciones. Para un primer análisis veremos los 4 primeros

• 32 registros de 64 bits. En las prácticas usaremos registros de 32 bits

• Acceso a memoria solo a través de lw (palabra de carga) y sw (palabra de almacenamiento)

• Acceso a memoria después de ALU

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamenteTabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Implementación de Jumps MIPS

Jump usa dirección de palabra

Actualizar PC con concatenación de

Los 4 bits principales de la PC antigua

Dirección de salto de 26 bits

00

Necesita una señal de control adicional decodificada del código de operación

Forma

Descripción generada automáticamente con confianza baja

Saltos en RISC-V: jal y jalr

RISC-V no implementa saltos (J) como instrucción.

De hecho, J existe, pero es una pseudoinstrucción que se implementa a través de JAL

Por otro lado, implementa dos saltos incondicionales (jal y jalr) que son relativos a los registros.

**Saltar y enlazar (JAL).** Saltos incondicionales relativos a PC (programaencimera)

**Registro de salto y enlace (JALR).** Similar a JAL, pero Saltos incondicional en relación con cualquier registro

Saltos en RISC-V: jal

**Saltar y enlazar (JAL).** La instrucción JAL se puede utilizar para ambos llamando a funciones o simplemente haciendo un simple salto incondicional.

JAL hace un salto relativo (relativo a PC) al igual que el condicional rama. Sin embargo, el argumento de registro proporcionado no se utiliza para comparaciones sino para almacenar la dirección del remitente.

Si no necesita la dirección del remitente, simplemente puede proporcionar la registro cero x0.

JAL rd, offset # rd ← PC + 4, PC ← PC + offset

La convención utilizada con RISC-V es que la dirección de retorno debe almacenarse en el registro de dirección de retorno ra (que es x1)

Tabla

Descripción generada automáticamente

JAL guarda PC+4 en el registro rd (la dirección de retorno)

Set PC = PC + offset (salto relativo PC: offset = inmediato firmado \* 2)

Objetivo en algún lugar dentro de ±219 ubicaciones, 2 bytes de distancia (es decir, ±218 de 32 bits instrucciones, ±220 bytes)

Comentarios:

El salto “j” del ensamblador es una pseudoinstrucción, usa JAL pero establece rd=x0 para descartar la dirección de retorno. El valor inmediato está codificado de forma optimizada, de forma similar a la instrucción de bifurcación, para reducir costo de hardware

Saltos en soporte RISC-V jal

1. Decodificar el código de operación y generar señales para:

1. Ruta Pc+inm\*2 a PC a través de mux M1

2. Guardar PC+4 (memtoReg puede tener dos bits)

2. Modificar mux M2 para enrutar PC+4 (necesita una entrada más)

3. Imm\_gen debería generar el inmediato en la derecha forma

Problemas de desempeño

El retraso más largo determina el período del reloj

Camino crítico: instrucción de carga

Memoria de instrucciones ® archivo de registro ® ALU ® memoria de datos ® archivo de registro

No es factible variar el período para diferentes instrucciones

Viola el principio de diseño

Haciendo el caso común rápido

Mejoraremos el rendimiento canalizando

TUBERÍAS

El procesador — 25

Nuevas Arquitecturas: Ocho Grandes Ideas

Diseño para la Ley de Moore

Usar la abstracción para simplificar el diseño

Haga que el caso común sea rápido (Amdahl)

Rendimiento a través del paralelismo

Rendimiento a través de canalización

Rendimiento mediante predicción

Jerarquía de memorias

Fiabilidad mediante redundancia

Analogía de canalización

Lavadero canalizado: ejecución superpuesta

El paralelismo mejora el rendimiento

Cuatro cargas:

Aceleración = 8/3,5 = 2,3

Sin escalas: aceleración = 2n/0,5n + 1,5 ≈ 4 = número de etapas

Canalización del procesador

n Cinco etapas, un paso por etapa

1. SI: Obtener instrucciones de la memoria

2. ID: decodificación de instrucciones y lectura de registro

3. EJ: Ejecutar operación o calcular dirección

4. MEM: Operando de memoria de acceso

5. WB: escribe el resultado de nuevo en el registro

Desempeño de la tubería

Suponga que el tiempo para las etapas es

100ps para registro de lectura o escritura

200ps para otras etapas

Comparar la ruta de datos segmentada con la ruta de datos de ciclo único

Tabla

Descripción generada automáticamente

Tabla, Excel

Descripción generada automáticamente

Aceleración de la tubería

Si todas las etapas están equilibradas

es decir, todos toman el mismo tiempo

Tiempo entre instrucciones canalizadas = Tiempo entre instrucciones no canalizadas / Número de etapas

Si no está equilibrado, la aceleración es menor

Aceleración debido a un mayor rendimiento

Latencia (tiempo para cada instrucción) no disminuye

Tubería y Diseño ISA

MIPS o RISC-V ISA diseñado para canalización

Todas las instrucciones son de 32 bits

Más fácil de obtener y decodificar en un ciclo

cf x86: instrucciones de 1 a 17 bytes

Formatos de instrucción pocos y regulares

Puede decodificar y leer registros en un solo paso

Cargar/almacenar direccionamiento

Puede calcular la dirección en la 3ra etapa, acceder a la memoria en la 4ta etapa

Alineación de operandos de memoria

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamenteEl acceso a la memoria toma solo un ciclo

Tabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Riesgos**

Situaciones que impiden iniciar la siguiente instrucción en el siguiente ciclo

Estructura de peligros

Un recurso requerido está ocupado

Riesgo de datos

Necesita esperar a que la instrucción anterior complete sus datos mleer escribir

Peligro de control

La decisión sobre la acción de control depende de las instrucciones previas

**Riesgos Estructurales**

Conflicto por el uso de un recurso

En tubería MIPS o RISC-V con una sola memoria

Cargar/almacenar requiere acceso a datos

La búsqueda de instrucciones tendría que detenerse para ese ciclo

Causaría una “burbuja” en la tubería

Por lo tanto, las rutas de datos segmentadas requieren Memorias de instrucción/datos (Arquitectura de Harvard)

O cachés de instrucciones/datos separados

Una instrucción depende de la finalización del acceso a los datos por una instrucción previa

Diagrama

Descripción generada automáticamente

**Reenvío (también conocido como derivación)**

Usar el resultado cuando se calcula

No espere a que se almacene en un registro

Requiere conexiones adicionales en la ruta de datos

Gráfico

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

**Detección de la necesidad de reenviar**

Pasar números de registro a lo largo de la tubería

ej., ID/EX.RegisterRs1 = número de registro para Rs1 sentado en Registro de tubería ID/EX

Se dan números de registro de operandos ALU en la etapa EX por

ID/EX.RegistroRs1, ID/EX.RegistroRs2

Riesgos de datos cuando

1a. EX/MEM.RegistroRd = ID/EX.RegistroRs1

1b. EX/MEM.RegistroRd = ID/EX.RegistroRs2

2a. MEM/WB.RegistroRd = ID/EX.RegistroRs1

2b. MEM/WB.RegistroRd = ID/EX.RegistroRs2

¡Pero solo si la instrucción de reenvío escribirá en un registro!

EX/MEM.RegWrite, MEM/WB.RegWrite

Y solo si Rd para esa instrucción no es x0

EX/MEM.RegistroRd ≠ 0,

MEM/WB.RegistrarRd ≠ 0

**Peligro EX (mover datos de MEM a EX)**

\* si (EX/MEM.RegWrite y (EX/MEM.RegisterRd ≠ 0) y (EX/MEM.RegistroRd = ID/EX.RegistroRs1))

AdelanteA = 10

\* si (EX/MEM.RegWrite y (EX/MEM.RegisterRd ≠ 0) y (EX/MEM.RegistroRd = ID/EX.RegistroRs2))

AdelanteB = 10

Peligro MEM (mover datos de WB a EX)

\* si (MEM/WB.RegWrite y (MEM/WB.RegisterRd ≠ 0) y (MEM/WB.RegisterRd = ID/EX.RegisterRs1))

AdelanteA = 01

\* si (MEM/WB.RegWrite y (MEM/WB.RegisterRd ≠ 0) y (MEM/WB.RegistroRd = ID/EX.RegistroRs2)

Reenviar B = 01

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Riesgo de datos dobles**

Considere la secuencia:

add x1,x1,x2

add x1,x1,x3

add x1,x1,x4

Ambos peligros ocurren

Quiere usar la versión más reciente

Revisar la condición de peligro del MEM

Solo adelante si la condición de peligro EX no es verdadera

**Condición de reenvío revisada**

peligro MEM

if (MEM/WB.RegWrite

and (MEM/WB.RegistroRd ≠ 0)

and not (EX/MEM.RegWrite y (EX/MEM.RegisterRd ≠ 0)

and (EX/MEM.RegistroRd ≠ ID/EX.RegistroRs1))

and (MEM/WB.RegisterRd = ID/EX.RegisterRs1)) ForwardA = 01

si (MEM/WB.RegWrite

and (MEM/WB.RegistroRd ≠ 0)

and not (EX/MEM.RegWrite y (EX/MEM.RegisterRd ≠ 0)

and (EX/MEM.RegistroRd ≠ ID/EX.RegistroRs2))

and (MEM/WB.RegisterRd = ID/EX.RegisterRs2)) ForwardB = 01

**Peligro de datos de uso de carga**

No siempre se pueden evitar las paradas reenviando

Si el valor no se calcula cuando es necesario

¡No se puede avanzar hacia atrás en el tiempo!

**Detección de riesgos de carga y uso**

Compruebe cuándo se decodifica la instrucción de uso en la etapa de ID

Los números de registro de operandos ALU en la etapa ID están dados por

IF/ID.RegistroRs1, IF/ID.RegistroRs2

Peligro de uso de la carga cuando

ID/EX.MemRead y ((ID/EX.RegisterRd = IF/ID.RegisterRs1) o (ID/EX.RegistroRd = IF/ID.RegistroRs1))

Si se detecta, detenga e inserte la burbuja

**Cómo detener la tubería**

Forzar los valores de control en el registro ID/EX a 0

EX, MEM y WB nop (sin operación)

Impedir la actualización del registro de PC e IF/ID

La instrucción de uso se decodifica de nuevo

La siguiente instrucción se obtiene de nuevo

La parada de 1 ciclo permite que MEM lea datos para ld

Puede pasar posteriormente a la etapa EX

**Puestos y rendimiento**

Las paradas reducen el rendimiento

Pero son necesarios para obtener resultados correctos

El compilador puede organizar el código para evitar peligros y paradas

Requiere conocimiento de la estructura de la tubería

Programación de códigos para evitar paradas

Reordenar el código para evitar el uso del resultado de carga en el próximo

instrucción

código n C para A = B + E; C = B + F;

Diagrama

Descripción generada automáticamente

**Controlar los peligros**

Rama determina el flujo de control

Obtener la siguiente instrucción depende del resultado de la bifurcación

Pipeline no siempre puede obtener la instrucción correcta

Todavía trabajando en la etapa de identificación de la sucursal

En el pipeline cuando se determina el resultado de la sucursal en MEM, aparece

retardo de rama cuando se toma la rama.

Ejemplo: rama tomada

Texto, Carta

Descripción generada automáticamente

**Riesgos de rama**

Cuando el resultado de la bifurcación se determina en MEM, el retraso de la bifurcación es de 3 ciclos cuando se toma la rama.

El bloqueo no ocurre hasta después de la etapa de identificación (donde la instrucción se decodifica como una rama

Tabla, Calendario

Descripción generada automáticamente

En proceso de agregar hardware para hacerlo en la etapa de ID

Necesidad de comparar registros y calcular el objetivo al principio de la tubería

Mueva el hardware para determinar el resultado a la etapa de identificación

sumador de direcciones de destino

Comparador de registros

Espere hasta que se determine el resultado de la bifurcación antes de obtener el siguiente

instrucción

Tabla

Descripción generada automáticamente

**Predicción de rama**

Las canalizaciones más largas no pueden determinar fácilmente el resultado de la ramificación

temprano

La penalización de pérdida se vuelve inaceptable

Predecir el resultado de la rama

Solo se detiene si la predicción es incorrecta

En proyecto

Puede predecir ramas no tomadas

Obtener instrucción después de la bifurcación, sin demora

Tabla

Descripción generada automáticamente

**Predicción de ramas más realista**

Predicción de rama estática

Predecir siempre lo mismo:

Funcionamiento especulativo hasta que se resuelva la condición. Si hay error, eliminar

resultados especulativos

• Predicción Efectiva (E): se produce bifurcación.

• No Predicción Efectiva (NE): NO se produce bifurcación.

Basado en el comportamiento típico de una rama

Ejemplo: bifurcaciones de bucle y sentencia if

Predecir las ramas hacia atrás tomadas (E)

Predecir ramas hacia adelante no tomadas (NE)

Predicción NE si el ramal es hacia delante y E si es hacia atrás

**Peligro de rama y predicción**

• Predicción de rama dinámica

El hardware mide el comportamiento real de las sucursales

por ejemplo, registrar el historial reciente de cada sucursal

Suponga que el comportamiento futuro continuará la tendencia.

Cuando sea incorrecto, deténgase mientras vuelve a buscar y actualice el historial.

• Cambiar la predicción según el historial de sucursales.

Use una memoria pequeña para cada dirección de sucursal (BHT, tabla de historial de sucursales)

Diagrama

Descripción generada automáticamente

**Predicción de bifurcación dinámica**

En tuberías más profundas y superescalares, la penalización de rama es más importante

Usar predicción dinámica

Búfer de predicción de sucursales (también conocido como tabla de historial de sucursales)

Indexado por direcciones de instrucciones de bifurcación recientes

Almacena el resultado (tomado/no tomado)

Para ejecutar una rama

Verifique la tabla, espere el mismo resultado

Comenzar a buscar desde la caída o el destino

Si es incorrecto, vacía la canalización y cambia la predicción

**Predicción de bifurcación: 2 o N niveles**

Tener en cuenta la correlación entre ramas.

Predictor (1,1):

• Almacenar la predicción de la rama actual basada en la el último.

• Se necesitan 2 BHT.

BHR (Branch History Register) contiene 1 bit, que indica el estado de la última rama (0=NE, 1=E).

Predictor (m, n):

• Utiliza la información de las m últimas sucursales.

• Se necesitan 2 millones de BHT de n bits.

BHR (Branch History Register) es un registro de desplazamiento de m bits, y contiene el estado de las m últimas ramas.

• Se utiliza como índice de los que utiliza BHT.

**Cálculo del destino de la rama**

Incluso con predictor, aún es necesario calcular el objetivo Dirección

1 ciclo de penalización por ramal tomado

Búfer de destino de bifurcación

Caché de direcciones de destino

Indexado por PC cuando se obtiene la instrucción

Si el golpe y la instrucción se predicen en la rama, se puede buscar el objetivo

inmediatamente

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Ç

**Peligros de datos para sucursales**

Si un registro de comparación es un destino de un registro

Instrucción ALU o segunda instrucción de carga anterior

Necesita 1 ciclo de parada

Diagrama

Descripción generada automáticamente  
**Peligros de datos para sucursales**

Si un registro de comparación es un destino de

inmediatamente anterior a la instrucción de carga

Necesita 2 ciclos de parada

Diagrama

Descripción generada automáticamente

**Resumen de canalización**

Pipelines mejora el rendimiento al aumentar rendimiento de instrucción

Ejecuta múltiples instrucciones en paralelo

Cada instrucción tiene la misma latencia

Sujeto a peligros

Estructura, datos, control

El diseño del conjunto de instrucciones afecta la complejidad de la canalización implementación

**falacias**

Canalizar es fácil (!)

La idea básica es fácil

El diablo está en los detalles

p. ej., detección de riesgos de datos

La canalización es independiente de la tecnología

Entonces, ¿por qué no hemos hecho canalizaciones siempre?

Más transistores hacen factibles técnicas más avanzadas

El diseño de ISA relacionado con tuberías debe tener en cuenta la tecnología

tendencias

p. ej., instrucciones predicadas

**Trampas**

Un mal diseño de ISA puede dificultar la canalización

Por ejemplo, conjuntos de instrucciones complejas (VAX, IA-32)

Gastos generales significativos para hacer que la canalización funcione

Enfoque microoperativo IA-32 , por ejemplo, modos de direccionamiento complejos

Registro de efectos secundarios de actualización, direccionamiento indirecto de la memoria

p. ej., ramas retrasadas

Los oleoductos avanzados tienen intervalos de demora prolongados

**Observaciones finales**

ISA influye en el diseño de la ruta de datos y el control

Diseño de influencia de control y ruta de datos de ISA

La segmentación mejora el rendimiento de las instrucciones usando el paralelismo

Más instrucciones completadas por segundo

Latencia para cada instrucción no reducida

Peligros: estructurales, de datos, de control



Texto

Descripción generada automáticamente

**Saltos en RISC-V: jalr**

Registro de salto y enlace (JALR). esto es realmente lo mismo

instrucción pero con la diferencia de que usamos un desplazamiento de

¿Registrarse? ¿Cuál es el punto de esto?

En JAL simplemente no hay suficiente espacio para codificar una dirección completa de 32 bits.

Eso significa que no puede saltar a ninguna parte del código si está en un espacio más grande.

programa. Pero si usa una dirección contenida en un registro, puede saltar a

cualquier dirección.

n JALR funciona casi igual que JAL. Almacena la dirección de retorno en rd (x1).



La gran diferencia es que los saltos de JALR no son relativos a la PC, sino que son relativos a rs1.

Tabla

Descripción generada automáticamente

**JALR guarda PC+4 en el registro rd (la dirección de retorno**). Igual que JAL

Establecer PC = rs + offset (12 bits, extensión de signo). Offset tiene menos rango que JAL.

Usa los mismos inmediatos que la aritmética y carga

A diferencia de las ramas, no hay multiplicación por dos antes de sumar a rs para formar la nueva PC

Desplazamiento de byte NO desplazamiento de media palabra como en sucursales y JAL

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Saltos en soporte RISC-V jalr**

Decodificar el código de operación y generar señales a:

1. Enrutar el registro rs1+inm a la PC a través de mux M1

2. Guardar PC+4 (memtoReg puede tiene dos bits)

Calcular registro rs1+inm

Modificar mux M1 (necesita una entrada más)

Modificar mux M2 para enrutar PC+4 (necesita una entrada más). Similar a JAL

Imm\_gen debería generar el inmediato en la forma correcta

Diagrama

Descripción generada automáticamente