Procesadores superescalares

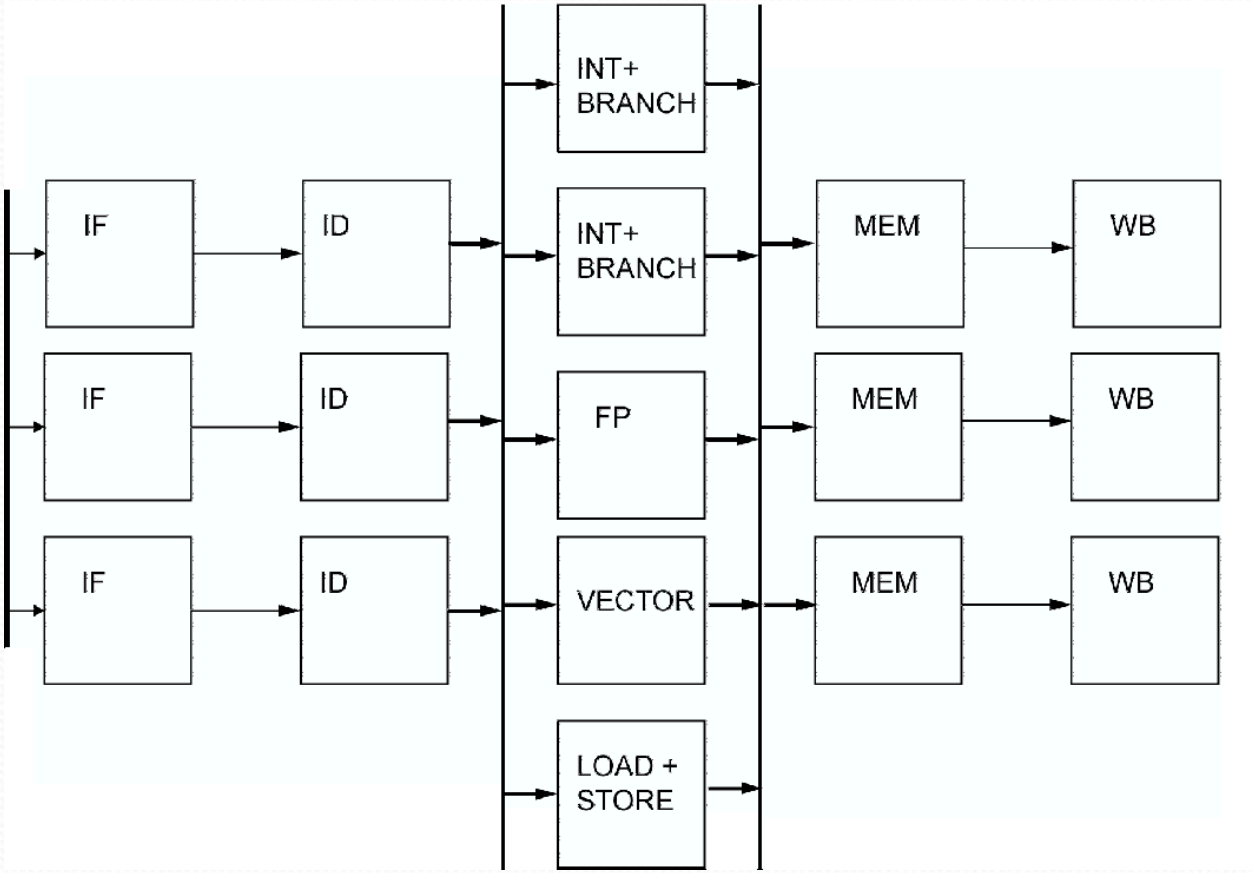
Introducción

Introducción

- El término “superescalar” (superscalar) fue acuñado a fines de los 80s.
- Todas las CPUs modernas son superescalares.
- Es un desarrollo de la arquitectura con pipeline.
- El pipeline se replica y en cada ciclo de reloj se emiten varias instrucciones.

Introducción

- Superescalar de grado 3.



Introducción

- En los superescalares dinámicos las instrucciones pueden correr y terminar en desorden.
- En los superescalares estáticos las instrucciones corren en orden y pueden terminar en desorden.
- Se generan nuevos peligros de datos por dependencias entre instrucciones.

Instrucciones por ciclo (IPC)

- Al emitir varias instrucciones por ciclo el CPI (ciclos por instrucción) es menor a 1.
- Una CPU de 6 GHz de grado 4 puede emitir hasta 4 instrucciones a la vez y ejecutar hasta 24 mil millones de instrucciones por segundo para un CPI de 0.25.
- En vez de CPI se usa IPC (instrucciones por ciclo de reloj) = 1 / CPI.
- Un CPI de 0.25 es igual a un IPC de 4.

Definiciones

- Grado. Número de instrucciones que el CPU *intenta* ejecutar (no hay garantía que lo pueda lograr).
- Scheduling. En que orden se ejecutan las instrucciones. Hay dos opciones:
  - Estático: la CPU no las puede reordenar.
  - Dinámico: la CPU si las puede reordenar.
- Emisión (issue). Una instrucción se emite al obtenerse (fetched) de la memoria.
- Paquete de emisión. Número de instrucciones que se emiten en cada ciclo.

Definiciones

- Hay dos estrategias de emisión para una superescalar de grado *n*:
  - Dinámica: el paquete de emisión puede contener 0, 1 o hasta *n* instrucciones.
  - Estática: el paquete de emisión siempre contiene *n* instrucciones. Si es necesario, el paquete se completa con instrucciones *nop*.

Definiciones

- Ejecución. Una instrucción se ejecuta, después de ser emitida, cuando sus operandos están listos. Hay dos opciones:
  - En orden del programa.
  - Fuera de orden.
- Especulación. Se intenta adivinar los brincos y el resultado de algunas otras instrucciones como por ejemplo que un lw/sw consecutivos no se refieren a la misma dirección.

Definiciones

- Compromiso (commit). Una instrucción se compromete cuando la CPU le permite actualizar el banco de registros o la memoria.
- Escribir (write back). Después de ejecutarse, la instrucción escribe el resultado.
- Retiro. Después de escribir, la instrucción es retirada del pipeline. También se dice que la instrucción ha sido *completada*.

Clasificación

- Superescalares estáticos.
  - Scheduling estático. Las instrucciones se emiten y ejecutan en orden, pero pueden terminar en desorden.
  - Emisión dinámica. El paquete de emisión no necesariamente tiene *n* instrucciones.

Clasificación

- Superescalares dinámicos.
  - Scheduling dinámico. Las instrucciones se emiten en orden, pero pueden correr y terminar en desorden.
  - Emisión dinámica. El paquete de emisión no necesariamente tiene *n* instrucciones.
  - Capacidad para especular brincos.

Clasificación

Common name	Issue structure	Hazard detection	Scheduling	Distinguishing characteristic	Examples
Superscalar (static)	Dynamic	Hardware	Static	In-order execution	Mostly in the embedded space: MIPS and ARM, including the ARM Cortex-A8
Superscalar (dynamic)	Dynamic	Hardware	Dynamic	Some out-of-order execution, but no speculation	None at the present
Superscalar (speculative)	Dynamic	Hardware	Dynamic with speculation	Out-of-order execution with speculation	Intel Core i3, i5, i7; AMD Phenom; IBM Power 7
VLIW/LIW	Static	Primarily software	Static	All hazards determined and indicated by compiler (often implicitly)	Most examples are in signal processing, such as the TI C6x
EPIC	Primarily static	Primarily software	Mostly static	All hazards determined and indicated explicitly by the compiler	Itanium

Peligros (hazards)

- Las CPUs superescalares tienen tres clases de peligros: estructurales, de datos y de control.
- Los peligros estructurales son los mismos que en el pipeline y se resuelven de la misma forma.
- Los peligros de control son los mismos que en el pipeline y se resuelven de la misma forma.

Peligros (hazards)

- Las CPUs superescalares tienen tres peligros de datos: RAW (read-after-write), WAW (write-after-write) y WAR (write-after-read).
- Los peligros de datos se generan por dependencias entre instrucciones.
- Además de la dependencia de datos, también conocida como dependencia de datos verdadera que puede producir un peligro RAW, hay dos dependencias de *nombre*.

Peligros (hazards)

- Las dependencias de datos por nombre se pueden eliminar.
- Las dependencias de datos verdaderas **no** se pueden eliminar.

Dependencias verdaderas

- La primera instrucción guarda su resultado en el mismo registro que una segunda instrucción va a leer.
- Ejemplo:

```
1. add $t1, $t2, $t3        # t1 = t2 + t3
2. sub $t4, $t5, $t1        # t4 = t5 – t1
```

- Si corren en desorden, sub lee el valor anterior de $t1.
- Puede producir un peligro RAW (read-after-write).

Dependencias de nombre

1. Dependencias de salida.
   - Dos instrucciones guardan sus resultados en la misma parte.
   - Ejemplo:

     1. add $t0, $t1, $t2 **# t0 = t1 + t2**

     2. add $t0, $t3, $t4 **# t0 = t3 + t4**
   - Si son ejecutadas fuera de orden, $t0 se queda con un valor incorrecto.
   - Puede producir un peligro WAW (write-after-write).

Dependencias de nombre

2. Antidependencias.
   - La primera instrucción necesita un valor que la segunda instrucción actualiza.
   - Ejemplo:

     1. add $t4, $t0, $t1 **# t4 = t0 + t1**

     2. sub $t0, $a0, $t2 **# t0 = a0 + t2**

   - Si son ejecutadas fuera de orden, add lee el valor actualizado de $t0.
   - Puede producir un peligro WAR (write-after-read).

Dependencias de nombre

- Dos métodos básicos para eliminar las dependencias de nombre:

1. Renombrar los registros.
2. Usar un buffer para reordenar (reorder buffer o ROB).

Renombrar registros

- El hardware mantiene una lista grande de registros virtuales.
- Los registros virtuales son invisibles al programador.
- Se asignan dinámicamente cuando un registro físico aparece como un registro destino.
- La CPU mantiene un mapeo entre registros virtuales y registros físicos.
- Varios registros virtuales pueden estar mapeados al mismo registro físico.

Renombrar registros

- Este mapeo funciona como una historia del registro físico.

Renombrar registros

- Ejemplo:

```
1. mul $r2, $r2, $r3    # r2 = r2 * r3
2. addi $r4, $r2, 1     # r4 = r2 + 1
3. addi $r2, $r3, 1     # r2 = r3 + 1
4. div $r5, $r2, $r4    # r5 = r2 / r4
```

- Dependencias de salida: 1 y 3 por $r2.
- Antidependencias: 2 y 3 por $r2.
- Dependencias verdaderas: 1 y 2 por $r2, 2 y 4 por $r4, 3 y 4 por $r2.
- El programa debe correr en orden.

Renombrar registros

- Los registros se pueden etiquetar:

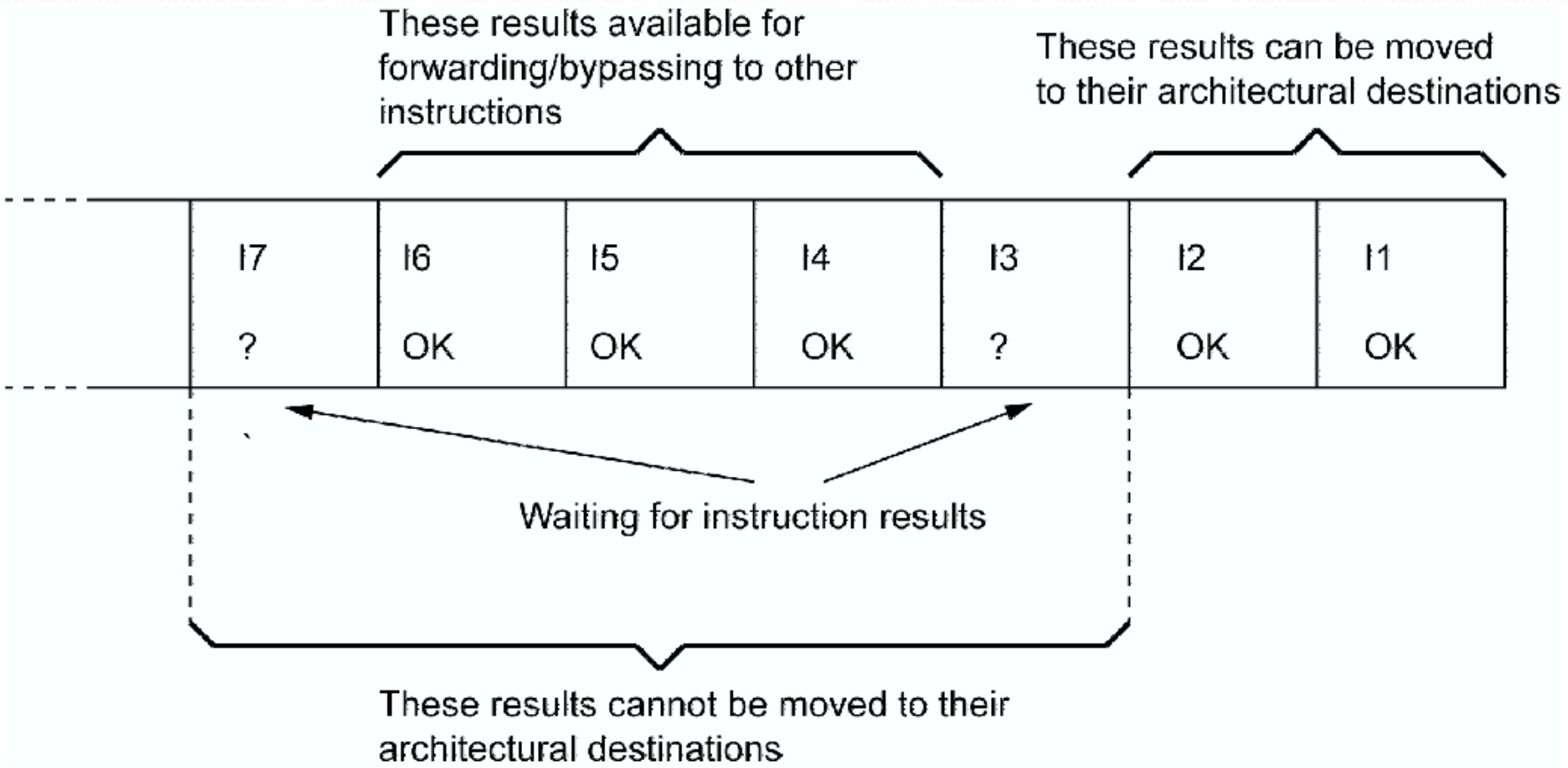
```
1. mul $r2_b, $r2_a, $r3_a      # r2 = r2 * r3
2. addi $r4_b, $r2_b, 1         # r4 = r2 + 1
3. addi $r2_c, $r3_a, 1         # r2 = r3 + 1
4. div $r5_b, $r2_c, $r4_b      # r5 = r2 / r4
```

- Ahora la instrucción 3 puede comenzar porque usa un registro $r2 "diferente" que 1 y 2.
- Las etiquetas funcionan como historia de cada registro: $r2_c es la versión más nueva de $r2, mientras que $r2_a es la versión más antigua.

ROB

- Provee registros adicionales.
- Mantiene el resultado de una instrucción desde que termina hasta que se compromete.
- Cada entrada en el ROB tiene 4 campos:
  - Tipo de instrucción: brinco, memoria o tipo R.
  - Registro destino si es carga o tipo R.
  - Valor de la instrucción.
  - Indicador si la instrucción terminó y el valor es válido.
- Las entradas están ordenados en cola (FIFO).

ROB


These results available for forwarding/bypassing to other instructions
These results can be moved to their architectural destinations
I7
?
I6
OK
I5
OK
I4
OK
I3
?
I2
OK
I1
OK
Waiting for instruction results
These results cannot be moved to their architectural destinations

ROB

- Si una instrucción necesita el valor de un registro, primero debe buscarlo en el ROB, de atrás hacia delante, y si no lo encuentra entonces lo toma del registro físico.

Conclusión

- Los procesadores superescalares son una extensión de la arquitectura pipeline simple que emiten y pueden ejecutar varias instrucciones a la vez.
- Producen más peligros y requieren soluciones más sofisticadas que el pipeline simple.
- Pueden alcanzar CPIs menores a 1.