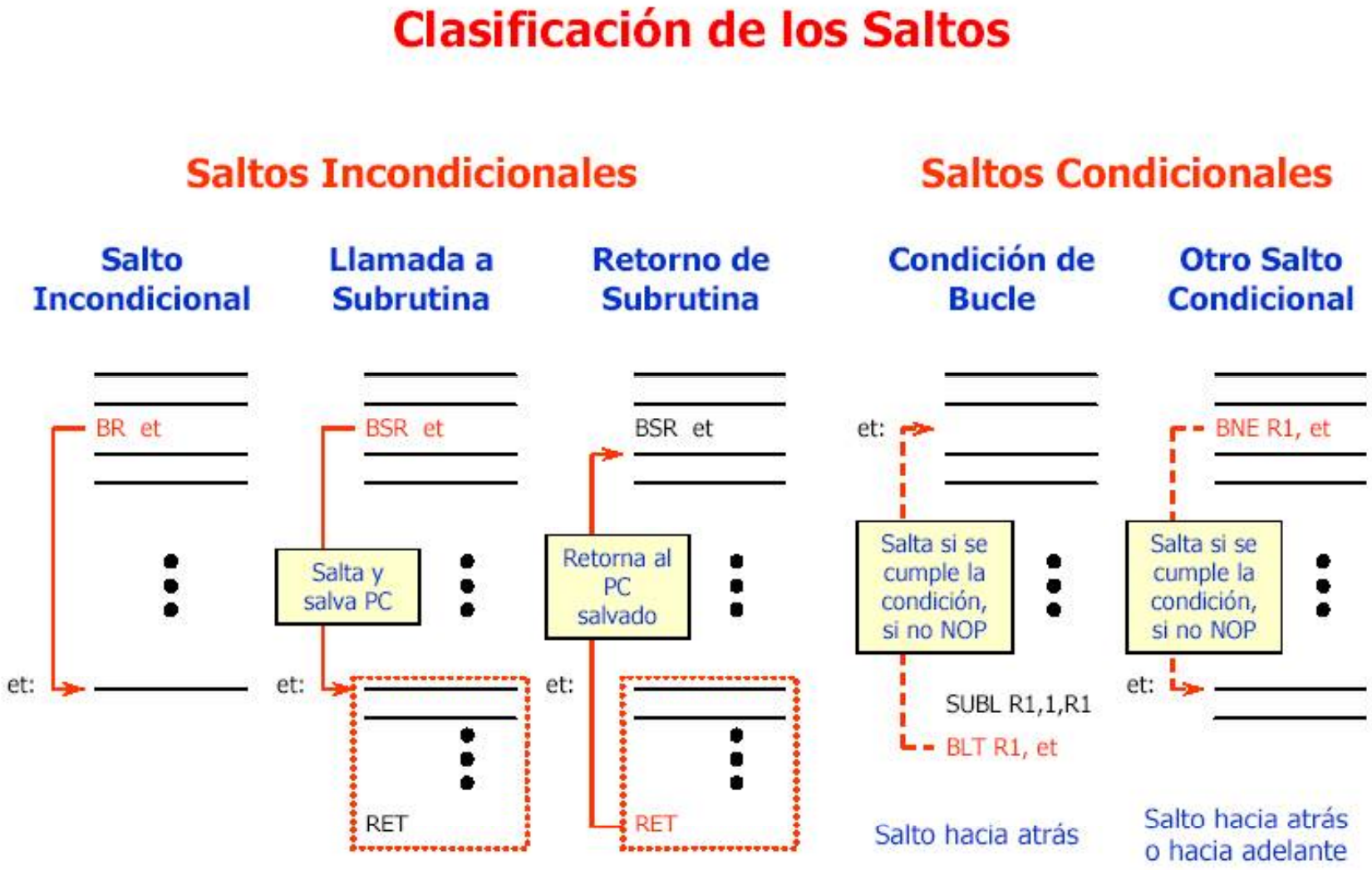
Arquitectura e Ingeniería de Computadores

Tema 2 – Segmentación y superescalares clase 6

Ingeniería en Informática

Departamento de Tecnología Informática y Computación

- **Tratamiento de las instrucciones de salto**



Aspectos específicos del tratamiento superescalar de los saltos

El **efecto de los saltos en los procesadores superescalares es más pernicioso** ya que, al emitirse varias instrucciones por ciclo, *prácticamente en cada ciclo* puede haber una instrucción de salto.

El **salto retardado no tiene mucho interés** porque la unidad de emisión decide las instrucciones que pasan a ejecutarse teniendo en cuenta las dependencias.

- **Detección de la Instrucción de Salto**

  Cuanto antes se detecte que una instrucción es de salto menor será la posible penalización. Los saltos se detectan usualmente en la fase de decodificación.

- **Gestión de los Saltos Condicionales no Resueltos**

  Si en el momento en que la instrucción de salto evalúa la condición de salto ésta no se haya disponible se dice que *el salto o la condición no se ha resuelto*. Para resolver este problema se suele utilizar el ***procesamiento especulativo del salto***.

- **Acceso a las Instrucciones destino del Salto**

  Hay que determinar la forma de acceder a la secuencia a la que se produce el salto

- **La implementación física de las Unidades de Salto**

1. Detección de las Instrucciones de Salto (I)

- **En la Etapa de Decodificación común a todas las instrucciones**
  - Usual en los primeros procesadores con segmentación de cauce **(80386, 68030).**
  - Acarrea una mayor penalización en el procesamiento de saltos.

- **Detección rápida (o temprana) del salto, *'Early branch detection'***
  - **Detección Paralela:** Hay una etapa específica para detectar la instrucción de salto que opera en paralelo con la etapa común de decodificación **(Procesadores Alpha)**

  - **Detección anticipada (*'Look-ahead branch detection'*):** Además de utilizar detección paralela se analizan las últimas líneas de los buffers previos a la etapa de decodificación donde se almacenan las instrucciones que van a pasar a decodificarse **(PowerPC 603)**

  - **Detección integrada con la Captación:** En el proceso de captación de instrucciones también se detecta si una instrucción es de salto (para ello se puede ayudar de los bits de predecodificación). En el caso que se detecte una instrucción de salto, junto con la instrucción siguiente a la instrucción de salto (o en lugar de, si así se predice) se capta la instrucción donde se produce el salto **(PowerPC 640)**

2. Gestion de saltos condicionales no resueltos (I)

Alternativas para la condición del salto

Estado del Resultado

Existen bits de Estado que se modifican al realizar operaciones o mediante operaciones que comprueban específicamente el valor de los registros

```
add r1,r2,r3
beq cero
div r5,r4,r1
.....
cero:
```

Dependencia que limita las Prestaciones en VLIW y superescalares

Ejemplos:

IBM/360, PDP-11, VAX, X-86, Pentium, PowerPC, Sparc

Comprobación Directa

Los resultados de las operaciones se comprueban directamente respecto a las condiciones específicas mediante instrucciones específicas

Dos Instrucciones

```
add r1,r2,r3
cmpeq r7, r1, 0
bt r7, cero
div r5,r4,r1
.....
cero:
```

Las dependencias se explicitan

Ejemplos:

Am 29000

Una Instrucción

```
add r1,r2,r3
bz r1, cero
div r5,r4,r1
.....
cero:
```

Ejemplos:

Cray, MIPS, MC881X0, HP PA, DEC Alpha

Tendencia

2. Gestion de saltos condicionales no resueltos (II)

Uso de los ciclos que siguen a la inst. de salto condicional	Salto Retardado	Se utilizan los ciclos que siguen a la captación de una instrucción de salto para insertar instrucciones que deben ejecutarse independientemente del resultado del salto (Primeras arquitecturas RISC y posteriores)
Gestión de Saltos Condicionales no Resueltos (Una condición de salto no se puede comprobar si no se ha terminado de evaluar)	Bloqueo del Procesamiento del Salto	Se bloquea la instrucción de salto hasta que la condición esté disponible (68020, 68030, 80386)
	Procesamiento Especulativo de los Saltos	La ejecución prosigue por el camino más probable (se especula sobre las instrucciones que se ejecutarán). Si se ha errado en la predicción hay que recuperar el camino correcto. (Típica en los procesadores superescalares actuales)
	Múltiples Caminos	Se ejecutan los dos caminos posibles después de un salto hasta que la condición de salto se evalúa. En ese momento se cancela el camino incorrecto. (Máquinas VLIW experimentales: Trace/500 , URPR-2)
Evitar saltos condicionales	Ejecución Vigilada (*Guarded Exec.*)	Se evitan los saltos condicionales incluyendo en la arquitectura instrucciones con operaciones condicionales (IBM VLIW, Cydra-5, Pentium, HP PA, Dec Alpha)

2.1 Procesamiento especulativo

Esquemas de Predicción de Salto

Predicción Fija

Se toma siempre la misma decisión: el salto siempre se realiza, *'taken'*, o no, *'not taken'*

Predicción Verdadera

La decisión de si se realiza o no se realiza el salto se toma mediante:

- **Predicción Estática**:
  Según los atributos de la instrucción de salto (el código de operación, el desplazamiento, la decisión del compilador)
- **Predicción Dinámica:**
  Según el resultado de ejecuciones pasadas de la instrucción (historia de la instrucción de salto)

Prestaciones

Otro esquema

2.1.1 Predicción fija

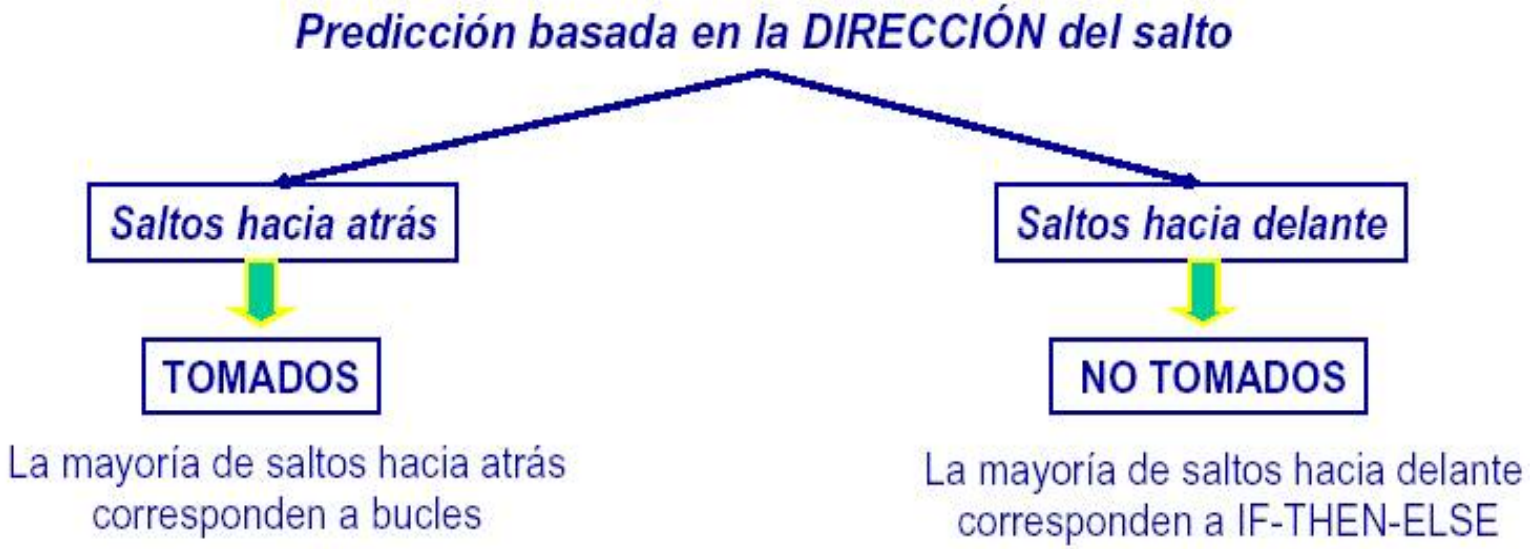
Aproximación 'Siempre No Tomado' ALWAYS NOT TAKEN - Toda condición de salto no resuelta se predice que no da lugar a un salto - Se continúa la ejecución por donde iba aunque se puede adelantar algo el procesamiento de la secuencia de salto (calculo de la dirección de salto, BTA) - Cuando se evalúa la condición se comprueba si la predicción era buena. - Si la predicción era buena el procesamiento continúa y se borra la BTA, y si era mala se abandona el procesamiento de la secuencia predicha (no se considera su efecto) y se captan instrucciones a partir de la BTA * Es más fácil de implementar que la aproximación de 'Siempre Tomado'	SuperSparc (1992) (TP: 1; NTP: 0) Alpha21064 Power I (1990) (TP: 3; NTP: 0) Power 2 (1993) (TP: 1; NTP: 0)
Aproximación 'Siempre Tomado' ALWAYS TAKEN - Toda condición de salto no resuelta se predice que da lugar a un salto - En previsión de error de predicción se salva el estado de procesamiento actual (PC) y se empieza la ejecución a partir de la dirección de salto. - Cuando se evalúa la condición de salto se comprueba si la predicción era buena - Si la predicción es correcta se continúa, y si es errónea se recupera el estado almacenado y no se considera el procesamiento de la secuencia errónea * Necesita una implementación más compleja que la aproximación anterior aunque suele proporcionar mejores prestaciones	MC68040 (1999) (TP: 1; NTP: 2)

2.1.2 Predicción estática

Predicción basada en el Código de Operación Para ciertos códigos de operación (ciertos saltos condicionales específicos) se predice que el salto se toma, y para otros que el salto no se toma	MC88110 (93) PowerPC 603(93)

Ejemplo: Predicción Estática en el MC88110

Formato	Instrucción		Predicción
	Condición Especificada	Bit 21 de la Instr.	
bcnd (*Branch Conditional*)	≠ 0	1	Tomado
	= 0	0	No Tomado
	> 0	1	Tomado
	< 0	0	No Tomado
	≥ 0	1	Tomado
	≤ 0	0	No Tomado
	bb1 (*Branch on Bit Set*)		Tomado
	bb0 (*Branch on Bit Clear*)		No Tomado



Mal comportamiento en programas con pocos bucles y muchos IF-THEN-ELSE

EJEMPLOS

- Alpha 21064 (1992) (Opción seleccionable)
- PowerPC 601/603 (1993)

- Se añade un ***Bit de Predicción*** al opcode de la instrucción
- El compilador activa o desactiva este bit para indicar su predicción

EJEMPLOS
- AT&T 9210 Hobbit (1993)
- PowerPC 601/603 (1993)
- PA 8000 (1996)

2.1.3 Predicción dinámica

- La predicción para cada instrucción de salto puede cambiar cada vez que se va a ejecutar ésta según la historia previa de saltos tomados/no-tomados para dicha instrucción.
- El presupuesto básico de la predicción dinámica es que es más probable que el resultado de una instrucción de salto sea similar al que se tuvo en la última (o en las *n* últimas ejecuciones)
- Presenta mejores prestaciones de predicción, aunque su implementación es más costosa

- **Predicción Dinámica Implícita**

  No hay bits de historia propiamente dichos sino que se almacena la dirección de la instrucción que se ejecutó después de la instrucción de salto en cuestión

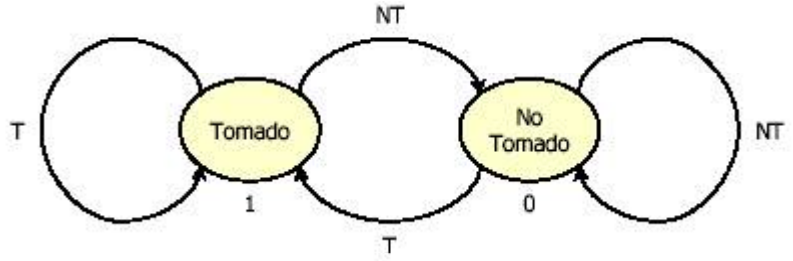
- **Predicción Dinámica Explícita**

  Para cada instrucción de salto existen unos bits específicos que codifican la información de historia de dicha instrucción de salto

Predicción dinámica explícita

Predicción con 1 bit de historia

La designación del estado, Tomado (1) o No Tomado (0), indica lo que se predice, y las flechas indican las transiciones de estado según lo que se produce al ejecutarse la instrucción (T o NT)



EJEMPLOS

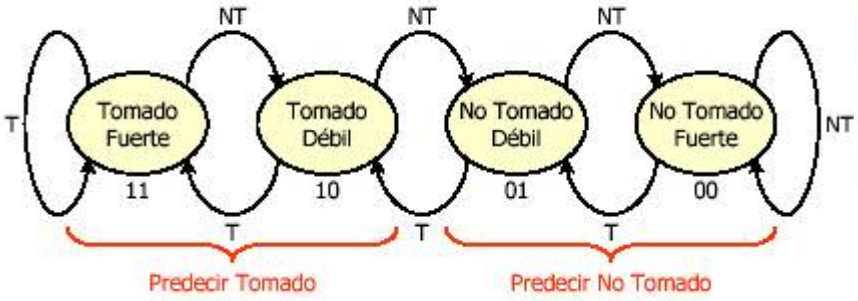
- Alpha 21064* (1992)
- R8000 (1994)

Predicción con 2 bits de historia

Existen cuatro posibles estados. Dos para predecir Tomado y otros dos para No Tomado

La primera vez que se ejecuta un salto se inicializa el estado con predicción estática, por ejemplo 11

Las flechas indican las transiciones de estado según lo que se produce al ejecutarse la instrucción (T o NT)



EJEMPLOS

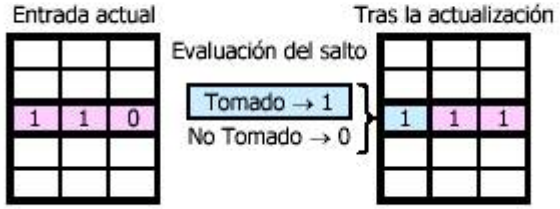
- MC68060 (199
- Pentium (199
- Alpha 21064A* (199
- Alpha 21164 (199
- UltraSparc (1995
- PowerPC 604 (1995
- PowerPC 620 (1996
- R10000 (1996

Predicción con 3 bits de historia

Cada entrada guarda las últimas ejecuciones del salto

Se predice según el bit mayoritario (por ejemplo, si hay mayoría de unos en una entrada se predice salto)

La actualización se realiza en modo FIFO, los bits se desplazan, introduciéndose un 0 o un 1 según el resultado final de la instrucción de salto



EJEMPLOS

- PA 8000 (1996)

Implementación de los Bits de Historia

• **En la Cache de Instrucciones**	Alpha 21064 (92) (2k x 1 bit) Alpha 21064A (94) (4k x 2 bits) UltraSparc (92) (2k x 2 bits)
• **En una Tabla de Historia de Salto (BHT)**	PA8000(96) (253 x 3bits) PowerPC 620(95) (2K x 2bits) R10000 (96) (512 x 2bits)
• **En una Cache para las Instrucciones a las que se produce el Salto (BTIC) o,** • **En una Cache para las Direcciones a las que se produce el Salto (BTAC)**	Pentium (94) (256 x 2 bits) (BTAC) MC 68069 (93) (256 x 2bits) (BTAC)

1) Branch Target Buffer (BTB): bits acoplados

La BTB almacena

- La dirección destino de los últimos saltos tomados
- Los bits de predicción de ese salto

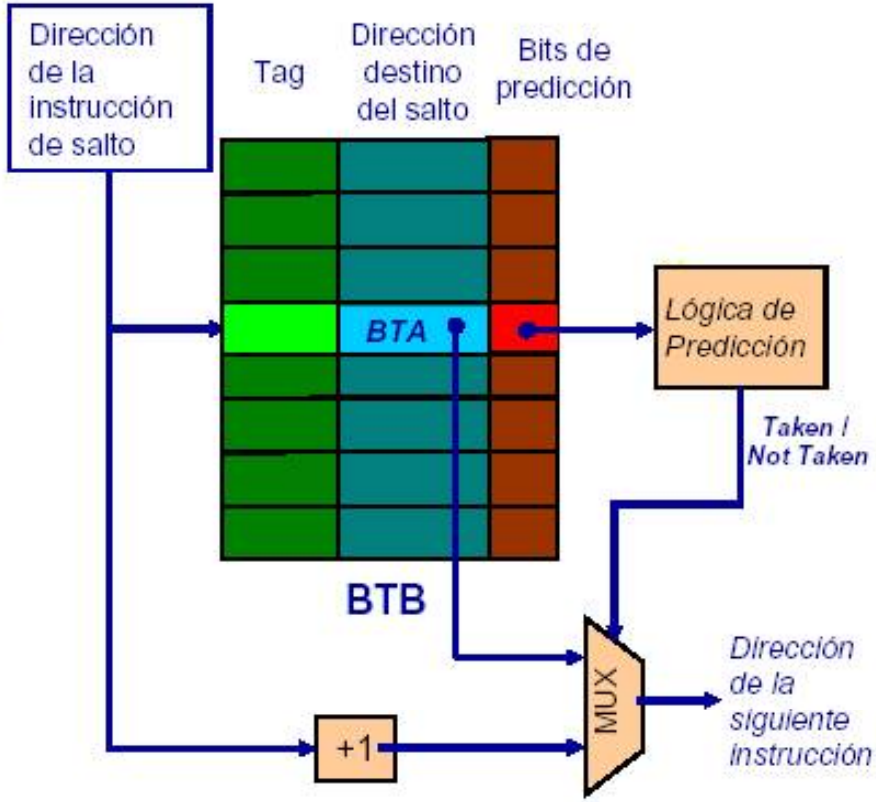
Actualización de la BTB

Los campos de la BTB se actualizan después de ejecutar el salto, cuando se conoce:

- Si el salto fue tomado o no ⇒ Actualizar bits de predicción
- La dirección destino del salto ⇒ Actualizar BTA

Predicción Implícita *(sin bits de predicción)*

Aplicable con un sólo bit de predicción
- Sí la instrucción de salto está en la BTB
  ⇒ El salto se predice como tomado
- Sí la instrucción de salto no está en la BTB
  ⇒ El salto se predice como no tomado



DESVENTAJA: Sólo se pueden predecir aquellas instrucciones de salto que están en la BTB

2) Tabla de historia de saltos (BHT): bits desacoplados

Existen dos tablas distintas:

- La BTAC, que almacena la dirección destino de los últimos saltos tomados
- La BHT, que almacena los bits de predicción de todos las instrucciones de salto condicional

Ventaja

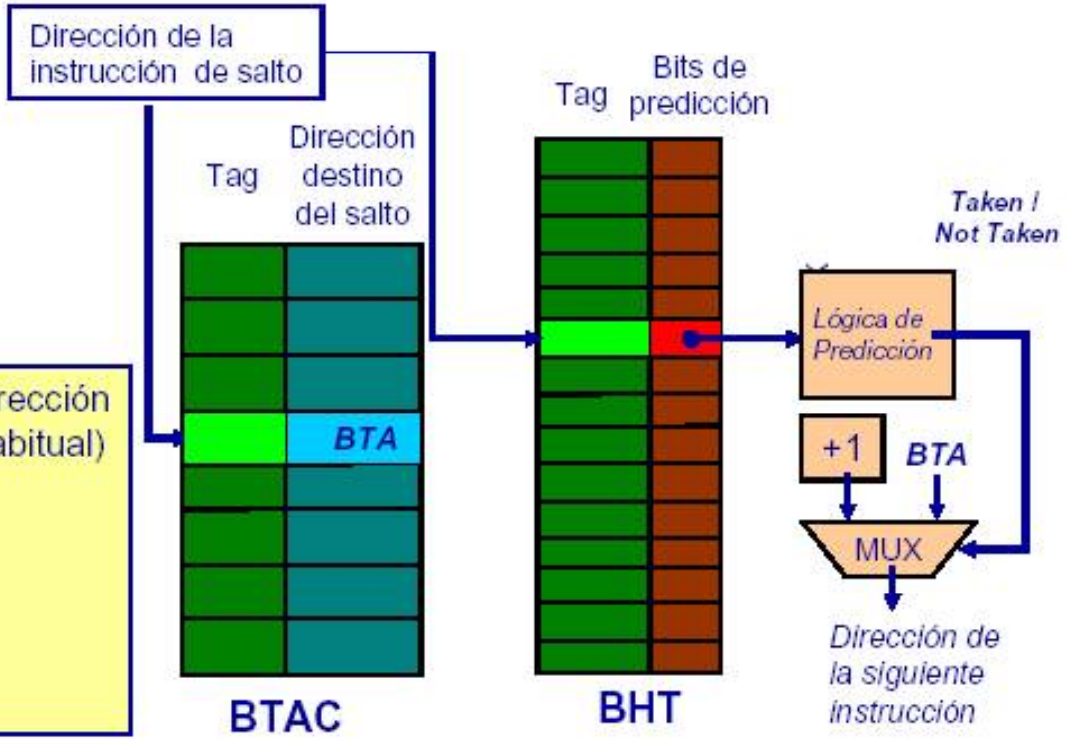
Puede predecir instruc.que no están en la BTAC (más entradas en BHT que en BTAC)

Desventaja

Aumenta el hardware necesario
⇒ 2 tablas asociativas

Acceso a la BHT

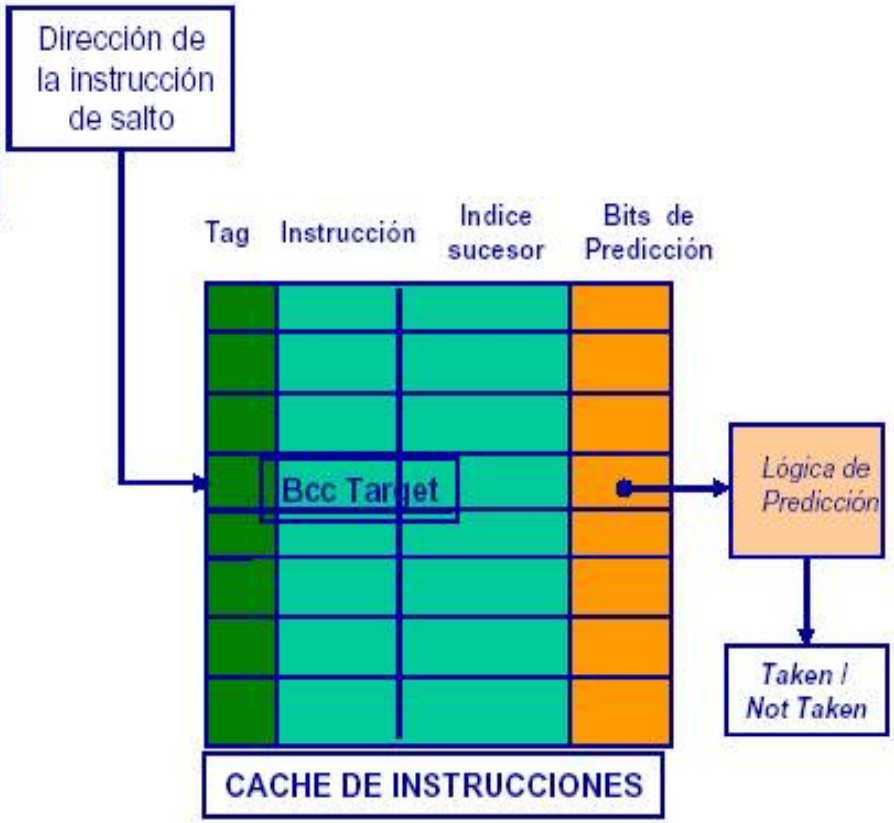
- Usando los bits menos significativos de la dirección
  - Sin TAGs ⇒ Menor coste (opción + habitual)
  - Compartición de entradas
    ⇒ Se degrada el rendimiento
- Asociativa por conjuntos
  - Mayor coste ⇒ Tablas pequeñas
  - Para un mismo coste hardware
    ⇒ Peor comportamiento



3) Bits de predicción en la I-cache

Funcionamiento

Cuando se capta la instrucción de la cache
Si se trata de una instrucción de salto condicional
- Se accede en paralelo a los bits de predicción
- Si el salto se predice como tomado se accede a la instrucción destino del salto



Acceso a la instrucción destino del salto

- BTB independiente
- Añadir *índice sucesor* a la I-cache

Alternativas de diseño

- Bits de predicción por cada instrucción de la cache
- Bits de predicción por cada línea de cache

Ventajas

- Puede predecir instrucciones que no están en la BTB
- No añade una cantidad extra de hardware excesiva

Extensión del Procesamiento Especulativo

- Tras la predicción, **el procesador continúa ejecutando instrucciones especulativamente hasta que se resuelve la condición**.
- El intervalo de **tiempo entre el comienzo de la ejecución especulativa y la resolución** de la condición **puede variar considerablemente y ser bastante largo**.
- En los procesadores superescalares, que pueden emitir varias instrucciones por ciclo, **pueden aparecer más instrucciones de salto condicional no resueltas durante la ejecución especulativa**.
- Si el número de instrucciones que se ejecutan especulativamente es muy elevado y la predicción es incorrecta, la penalización es mayor.

Así, **cuanto mejor es el esquema de predicción mayor puede ser el número de instrucciones ejecutadas especulativamente.**

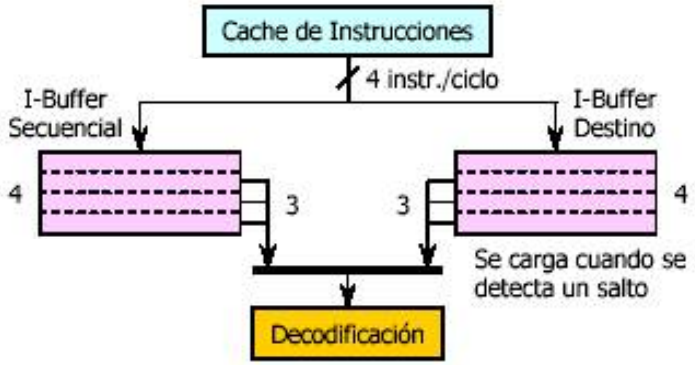
- **Nivel de Especulación:** Número de Instrucciones de Salto Condicional sucesivas que pueden ejecutarse especulativamente (si se permiten varias, hay que guardar varios estados de ejecución). **Ejemplos: Alpha21064, PowerPC 603 (1); Power 2 (2); PowerPC 620 (4); Alpha 21164 (6)**
- **Grado de Especulación:** Hasta qué etapa se ejecutan las instrucciones que siguen en un camino especulativo después de un salto. Ejemplos: Power 1 (Captación); PowerPC 601 (Captación, Decodificación, Envío); PowerPC 603 (Todas menos la finalización)

Recuperación de una Predicción Incorrecta

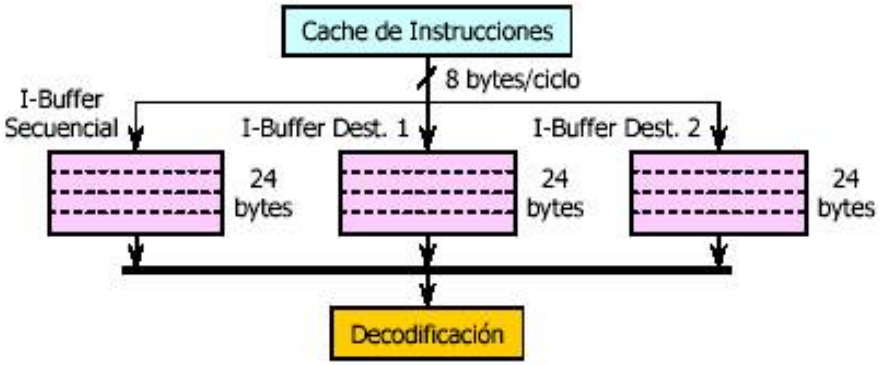
La recuperación de una predicción incorrecta comprende:
- **Descartar los resultados de la ejecución especulativa**
- **Continuar la ejecución de la secuencia de instrucciones alternativa (la correcta)**

Recuperación desde un salto efectuado:
- El procesador debe guardar la dirección de la instrucción siguiente a la de salto para utilizarla si la predicción es incorrecta.
- La recuperación es más rápida si no se descartan las instrucciones que se habían precaptado junto con la de salto

Recuperación cuando no se ha saltado:
- Pre-calcular la dirección de salto y almacenarse para permitir la recuperación.
- La recuperación es más rápida si se precaptan instrucciones de la secuencia que empieza a partir de la dirección a la que se salta.



Sistema de 2 buffers para recuperación
(SuperSparc, Power2, Pentium)



Sistema de 3 buffers para recuperación
(Nx586)

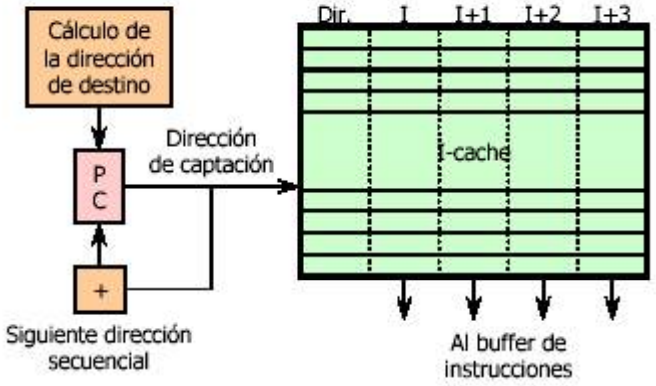
Acceso a la secuencia del salto

Los saltos condicionales efectuados ('taken') son más frecuentes que los no efectuados ('not taken'). Por ello, sería interesante **reducir al máximo el tiempo de acceso a la secuencia de instrucciones a partir de la dirección de salto y reducir la penalización para las predicciones incorrectas de los saltos efectuados.**

La rapidez de acceso a la secuencia de instrucciones que empieza en la dirección a donde se salta es fundamental para mejorar las prestaciones del esquema de gestión de los saltos condicionales.

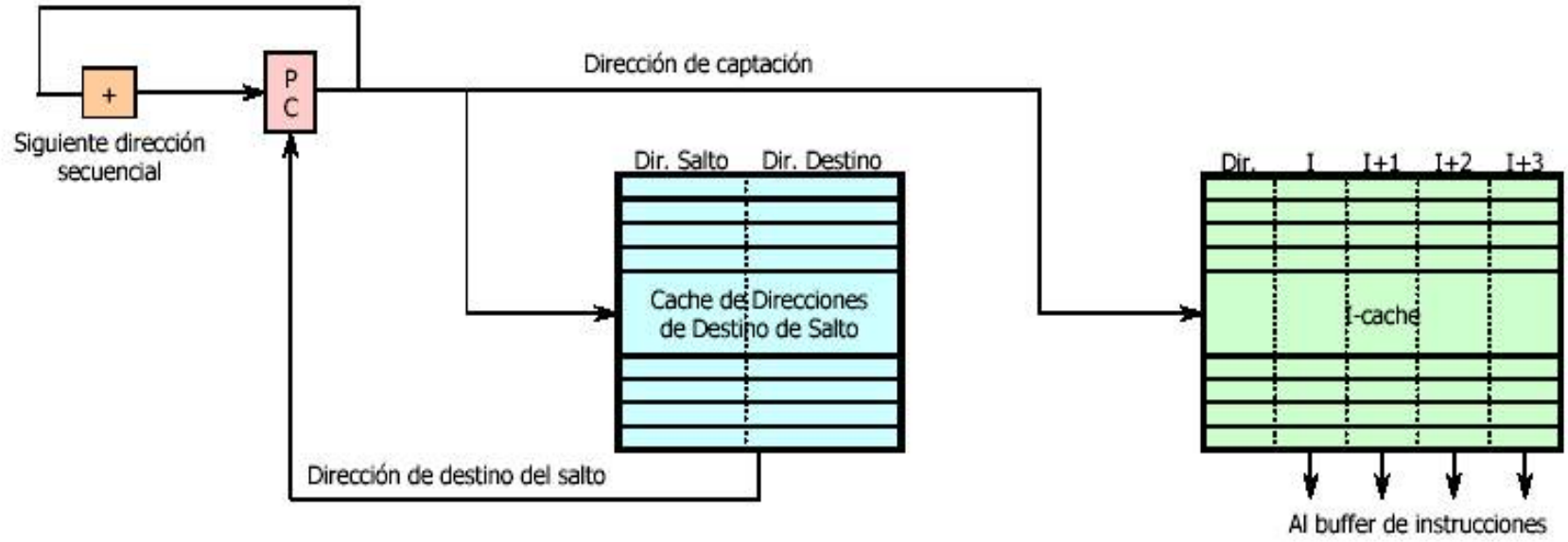
1 **Esquema Calcular/Captar**
(Procedimiento más sencillo de acceso)

- Si se detecta una instrucción de salto, se calcula su dirección de destino y se accede a la posición de memoria correspondiente
- Procedimiento utilizado en los primeros procesadores segmentados y en procesadores superescalares tales como alpha 21164 (95), y R10000 (96)



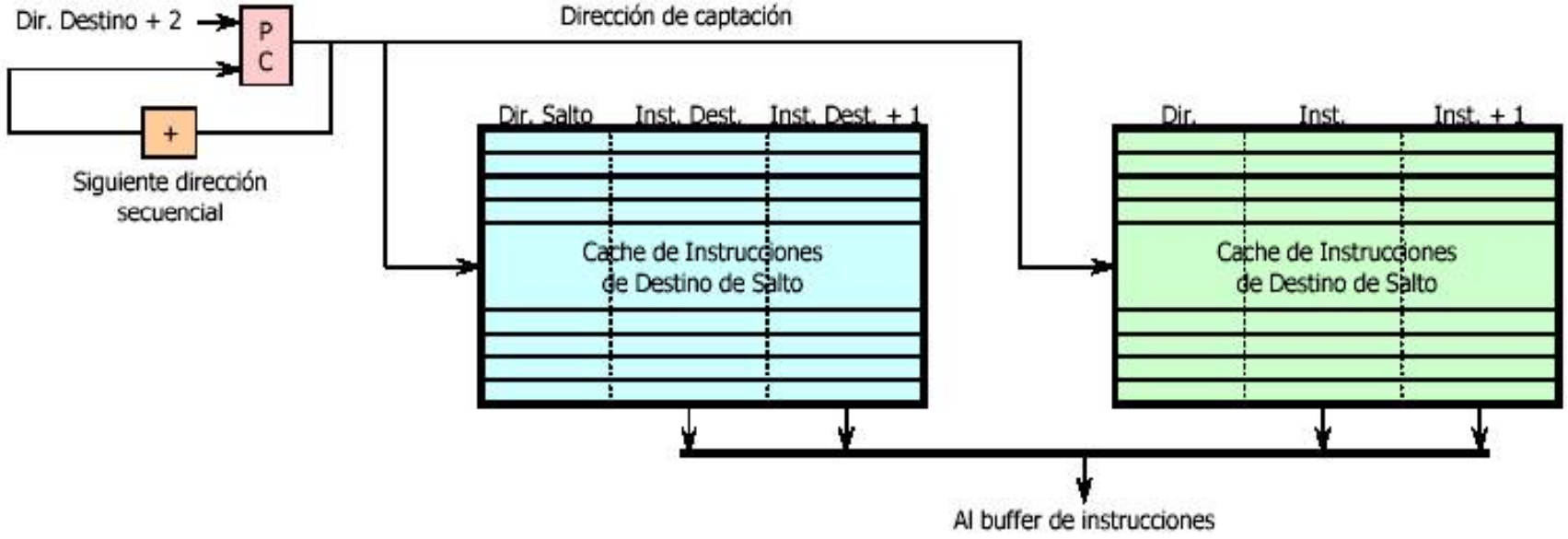
2. Esquema de cache de direcciones de destino de salto (BTAC)

- Se añade una cache que contiene las direcciones de las instrucciones destino de los saltos, junto con las direcciones de las instrucciones de salto.
- Se leen las direcciones al mismo tiempo que se captan las instrucciones de salto.



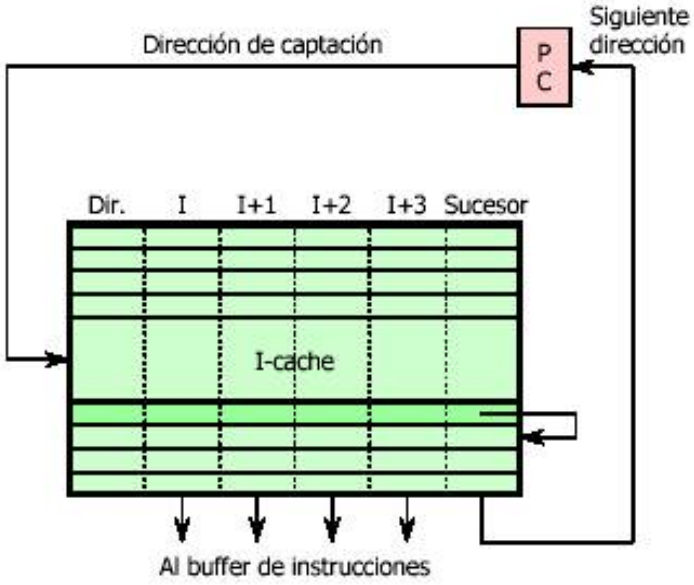
3. Esquema de cache de instrucciones de destino de salto (BTIC)

- Se añade una cache que contiene las instrucciones siguientes a la dirección de destino de los saltos, junto con las direcciones de las instrucciones de salto.
- Sólo tiene sentido si la cache de instrucciones tiene una latencia muy alta.
- Mientras se procesan estas instrucciones se calcula la dirección de las siguientes.



4. Esquema de índice sucesor en la cache de instrucciones

- La cache de instrucciones contiene un índice sucesor que apunta a la siguiente línea de la cache de instrucciones que hay que captar (la siguiente, o la que se predice que se debe captar si hay una instrucción de salto condicional en esa línea).

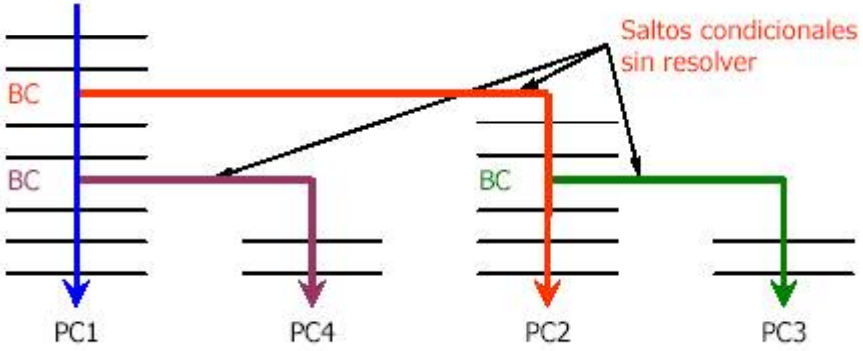


Ejemplos y evolución de los esquemas de Acceso

Calcular/captar	BTIC	BTAC	Índice sucesor
i486 (1989)	→	Pentium (1993)	
MC68040 (1990)	→	MC 68060 (1993)	
	Am 29000 (1988)	→	Am 29000 superscalar (1995)
Sparc CYC 600 (1992) SuperSparc (1992)	→	→	UltraSparc (1995)
R4000 (1992) R10000 (19996)	→	→	R8000 (1994)
PowerPC 601 (1993) PowerPC 603 (1993)	→	PowerPC 604 (1995) PowerPC 620 (1996)	

3. Ramificación Multicamino

- Se **siguen las dos secuencias de instrucciones que aparecen a partir de una instrucción de salto** (la correspondiente al salto efectuado y al salto no efectuado). Una vez resuelto el salto la secuencia incorrecta se abandona
- Para implementar esta técnica **hacen falta varios contadores de programa**
- Se necesitan **gran cantidad de recursos hardware** y el proceso para descartar las instrucciones que se han ejecutado incorrectamente es complejo



4. Instrucciones de Ejecución Condicional (*Guarded Execution*)

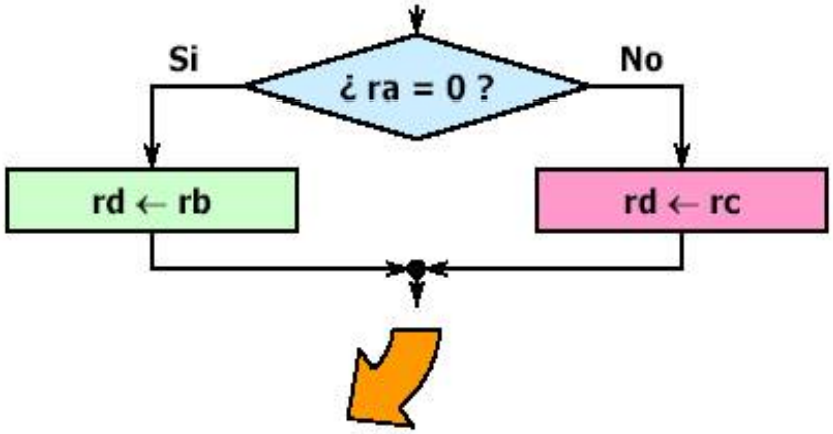
- Se pretende **reducir el número de instrucciones de salto** incluyendo en el **repertorio máquina instrucciones con operaciones condicionales** (*'conditional operate instructions'* o *'guarded instructions'*)
- Estas instrucciones tienen dos partes: la **condición** (denominada *guardia*) y la parte de **operación**

Ejemplo: cmovxx de Alpha

cmovxx ra.rq, rb.rq, rc.wq

- **xx** es una condición
- **ra.rq, rb.rq** enteros de 64 bits en registros ra y rb
- **rc.wq** entero de 64 bits en rc para escritura
- El registro ra se comprueba en relación a la condición xx y si se verifica la condición rb se copia en rc.

Sparc V9, HP PA, y Pentium ofrecen también estas instrucciones.




```
        beq   ra, cero
        or    rc, rc, rd
        br    fin
cero:   or    rb, rb, rd
fin:    ...
```

```
or      rb, rb, rd
cmovne  ra, rc, rd
...
```

Instrucciones de Ejecución Condicional (II)

Reducción en el Número de Saltos

Programa	% saltos de bucle	% saltos eliminados			
		Guardia completa		Guardia parcial	
		Cond.	Incond.	Cond.	Incond.
Compress	26.48	24.86	84.29	18.24	0.00
Eqntott	29.07	44.55	54.98	40.04	1.02
Espresso	38.08	16.76	29.03	10.17	1.17
Gcc-ccl	24.84	31.92	17.04	9.64	0.37
Sc	24.63	43.07	17.74	9.83	0.18
Sunbench	15.79	35.65	47.10	11.35	0.03
Supermips	5.03	50.69	19.36	17.15	0.60
Tektronix	16.83	37.53	41.60	17.08	7.48
TeX	25.09	12.80	24.03	5.99	1.00
Thissim	11.52	62.31	33.70	23.26	1.43
Tycho	18.28	15.64	33.84	7.10	1.31
Xlisp	27.03	13.64	14.33	13.87	14.14
Yacc	38.64	19.53	38.95	8.18	1.71
Media	23.17	31.15	35.07	14.76	2.34