PRÁCTICA 3. ANÁLISIS DE PRESTACIONES EN SISTEMAS REALES UTILIZANDO CONTADORES DE PRESTACIONES

Objetivos:

- Observar cómo el desperdicio vertical en la etapa de *issue* afecta a las prestaciones de los procesadores superescalares.
- Cuantificar el desperdicio vertical en una máquina comercial utilizando contadores de prestaciones.
- Identificar la necesidad de un buen tratamiento de las instrucciones de acceso a memoria para reducir el desperdicio vertical y analizar como los mecanismos de prebúsqueda contribuyen a lograrlo.

1. Conceptos teóricos

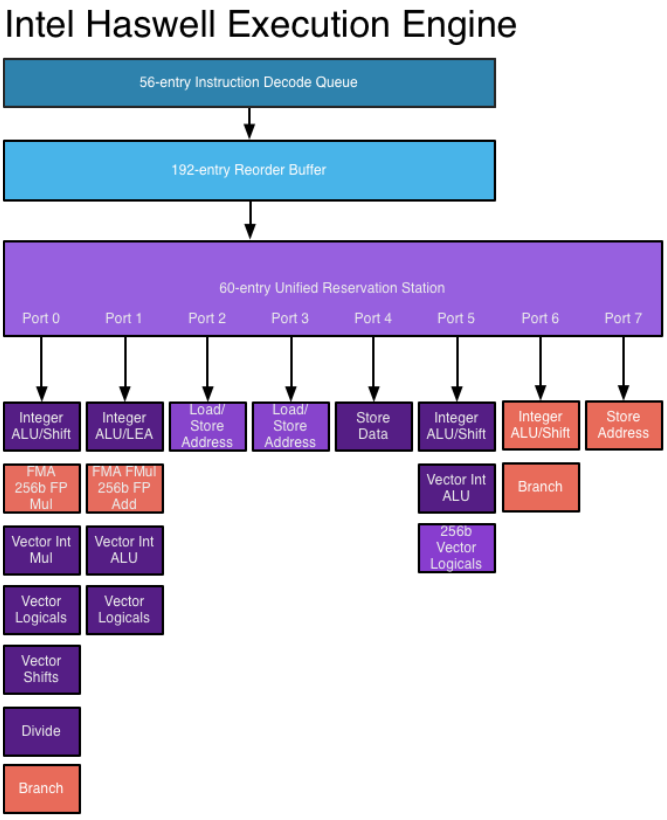


Figura 1. Diagrama de funcionalidad del *pipeline* de los procesadores con microarquitectura Haswell.

Procesadores superescalares y desperdicio vertical

Los procesadores escalares implementan varios puertos de lanzamiento (*issue slots* o *issue ports*) para incrementar el uso de los operadores y las prestaciones de la máquina. Los procesadores actuales suelen implementar más de 6 puertos. En concreto, en la Figura 1 se aprecia como el procesador Intel i5-4590, disponible en los equipos del laboratorio,

implementa 8 puertos de lanzamiento: 4 puertos pueden realizar operaciones aritméticas, 2 puertos pueden realizar el cálculo de la dirección de *loads* o *stores*, y solo a través de 1 puerto se puede transmitir el dato a almacenar (*store data*).
En un ciclo se pueden lanzar hasta cuatro instrucciones ajustándose a los puertos disponibles. Sin embargo, en la práctica aparecen muchos ciclos en los que se lanzan instrucciones, pero no se utilizan los cuatro *slots* de *issue* disponibles (desperdicio horizontal), y otros ciclos en los que no se lanza ninguna instrucción, lo que se conoce con el nombre de desperdicio vertical. En la siguiente figura se muestra un desperdicio vertical de 3 ciclos (ciclos 3, 4 y 8). En el desarrollo de esta práctica se cuantificará el desperdicio vertical en la máquina del laboratorio.

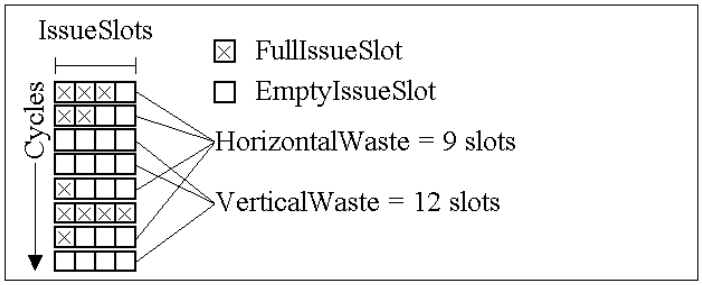


Figura 2. Diagrama del desperdicio de *issue slots*. El desperdicio vertical se produce cuando no se puede lanzar ninguna instrucción en un ciclo.

Contadores de prestaciones

Los contadores de prestaciones hardware son un conjunto de registros especiales que incorporan los procesadores para contabilizar actividades y eventos de bajo nivel que ocurren en el sistema. Cada procesador es capaz de monitorizar un conjunto diferente de eventos, aunque muchos de ellos son similares. Entre estos eventos encontramos el número de instrucciones ejecutadas, los saltos predichos correcta o incorrectamente, o los accesos y fallos en cada una de las caches del procesador, entre otros. Los procesadores más recientes permiten contabilizar cientos de eventos relacionados con estructuras muy concretas del procesador.
La configuración y el acceso a los contadores de prestaciones tiene una sobrecarga baja. Estas características, unidas al hecho de ofrecer información muy específica de eventos del sistema, hacen que el principal uso que se les da a los contadores de prestaciones actualmente sea el de obtener información acerca de la ejecución de las aplicaciones en el sistema para realizar análisis y optimizaciones tanto hardware como software, así como planificar la ejecución de las aplicaciones en el sistema de forma inteligente.

Ciclos de parada (*stalls*)

Los ciclos de parada son ciclos en los que el procesador no puede avanzar la ejecución de alguna instrucción, provocando un retardo en todas las instrucciones siguientes. En los procesadores con ejecución en orden se producen ciclos de parada cuando una instrucción cualquiera no puede avanzar debido a una dependencia de datos o riesgo estructural.

Sin embargo, en los procesadores con ejecución fuera de orden los ciclos de parada se contabilizan de manera distinta. Los procesadores comerciales lo realizan de tres maneras distintas, éstas son:

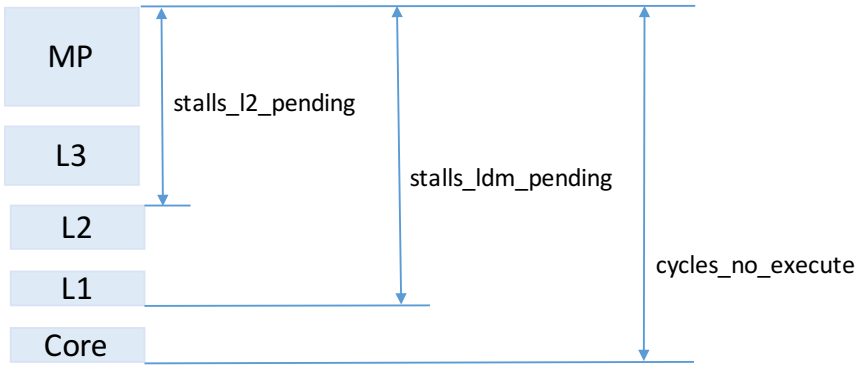
- Ciclos en los que el procesador no es capaz de completar ninguna instrucción (ciclos de parada asociados a la etapa *commit*). Es el caso de los procesadores IBM.
- Ciclos en los que el procesador no es capaz de lanzar ninguna instrucción a ejecución (asociados a la etapa *issue*), o ciclos de parada asociados a la etapa *dispatch* en los que el procesador no puede preparar alguna de las estructuras necesarias para su ejecución (por ej. el ROB). Los procesadores Intel permiten cuantificar ambos, tanto los ciclos de paradas asociados a la etapa *dispatch* como los asociados a la etapa *issue*.

De lo expuesto se deduce que los índices de prestaciones a estudiar dependen de la información que los contadores de prestaciones del sistema experimental nos pueden proporcionar. El procesador disponible en los equipos del laboratorio ofrece contadores de prestaciones que permiten contabilizar los ciclos de parada asociados a la etapa de *issue*. En concreto, permite contabilizar los siguientes eventos:

Evento	Descripción
cycle_activity:stalls_l2_pending	Ciclos en los que no se lanza ninguna instrucción debido a un fallo de L2.
cycle_activity:stalls_ldm_pending	Ciclos en los no se lanza ninguna instrucción debido a una load que accede al subsistema de memoria.
cycle_activity:cycles_no_execute	Ciclos en los que no se lanza ninguna instrucción.

Tabla 1. Eventos del Intel i5 relacionados con los ciclos de parada de la etapa *issue*.

La siguiente figura muestra gráficamente la relación entre los contadores utilizados y los distintos elementos del sistema:



El evento *cycle_activity:cycles_no_execute* incluye al *cycle_activity:stalls_ldm_pending*, que a su vez incluye al evento *cycle_activity:stalls_l2_pending*. Utilizando los eventos disponibles se deben clasificar los ciclos de parada en la etapa *issue* en tres categorías excluyentes:

- Debidos a *loads* que fallan en la cache L1 y aciertan en la L2.
- Debidos a *loads* que, tras fallar en la L2, aciertan en la LLC o en la memoria principal.
- Debidos a otras causas.

2. Desarrollo de la práctica

Configuración y conceptos básicos del entorno de trabajo

Descarga el fichero prac_3.tgz de poliformaT, el cual incluye:

- El programa `monitoriza_aplicacion`, que contabiliza los eventos configurados en la ejecución de una aplicación de la suite SPEC CPU2006. Para ello accede a los contadores de prestaciones a través de la librería `libpfm`.
- Los ejecutables y ficheros de entrada de un subconjunto de benchmarks de la suite SPEC CPU2006.

Para configurar el entorno de trabajo, descomprime el fichero prac_3.tgz y accede al directorio *working_dir*.

```
$ tar xzvf prac_3.tgz
$ cd working_dir
```

Actividad 1. Monitorización IPC y ciclos de parada en la jerarquía de memoria

Monitoriza la ejecución de los benchmarks de la suite SPEC CPU2006 y obtén para cada benchmark: el número de *ciclos de ejecución*, instrucciones retiradas (*committed*), ciclos de parada totales (evento *cycle_activity:cycles_no_execute*), ciclos de parada por acceso al subsistema de memoria (evento *cycle_activity:stalls_ldm_pending*) y ciclos de parada por fallo en la cache L2 (evento *cycle_activity:stalls_l2_pending*).

Para facilitar esta tarea se proporciona el fichero **`script_obtener_ciclos_parada.sh`**, que debe editarse para indicar los eventos a monitorizar en la variable EVENTS, **separándolos con comas y sin espacio tras la coma.** Por defecto, **`monitoriza_aplicacion`** ya contabiliza los ciclos en ejecución e instrucciones retiradas, por lo que no hay que añadir estos dos eventos en la variable EVENTS. Para reducir el tiempo de ejecución, la caracterización se realiza solo sobre un subconjunto de los benchmarks.

Con los datos obtenidos, realiza las siguientes 2 figuras:

- Diagrama de barras que muestre el IPC de los benchmarks estudiados en el eje vertical. A modo de ejemplo se proporciona la Figura 3.

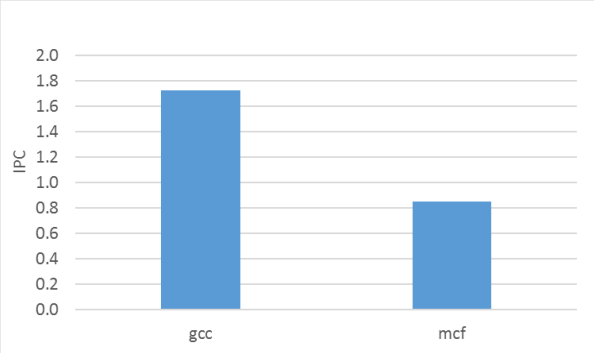


Figura 3. IPC de los benchmarks estudiados.

- Diagrama de barras apiladas que muestre el porcentaje de ciclos de parada respecto a los ciclos de ejecución de cada aplicación para los benchmarks estudiados. El porcentaje debe distinguir entre ciclos de parada provocados por accesos a la cache L2, por fallos en esta cache y accesos a la LLC (*last level cache*) y a la memoria principal, y ciclos de parada provocados por otros motivos. A modo de ejemplo se proporciona la siguiente figura.

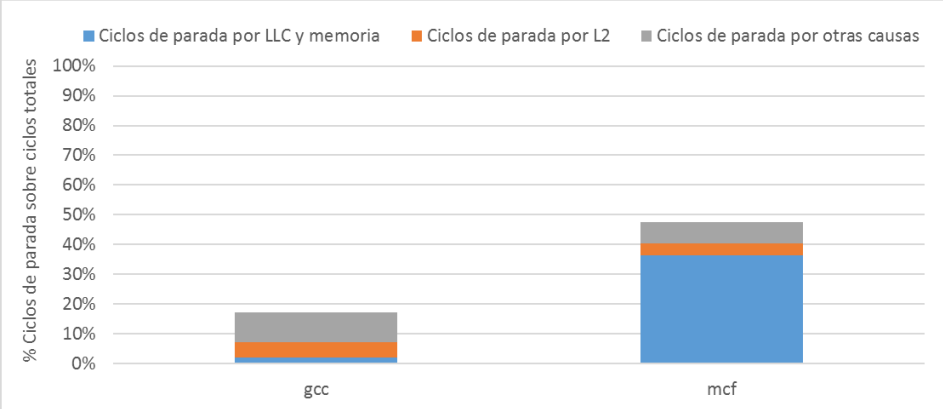


Figura 4. Ciclos de parada de los benchmarks estudiados.

¿Existe alguna relación entre el IPC y los ciclos de parada de las aplicaciones?

Activación y desactivación de la prebúsqueda en tiempo de ejecución

Los registros MSR (Model Specific Register) son un conjunto de registros de control del procesador que se utilizan, principalmente, para configurar ciertas características del procesador. Entre las características que permiten configurar, está la activación y desactivación de los diferentes mecanismos de prebúsqueda.

El control de la prebúsqueda se realiza con los cuatro bits de menor peso del registro MSR ubicado en la dirección 0x1A4. La Tabla 2 muestra las opciones de configuración. Un bit a 0 indica el mecanismo de prebúsqueda está habilitado. Para deshabilitarlo, basta con poner el bit a 1.

La lectura y escritura de los registros MSR se realiza con las instrucciones `rdmsr` y `wrmsr`, respectivamente, indicando los núcleos a los que se aplica, la dirección del registro, y en el caso de la escritura el valor a almacenar.

Mecanismo de prebúsqueda	Bit en el MSR 0x1A4	Descripción
L2 hardware prefetcher	0	Busca líneas adicionales de código o datos en la cache L2
L2 adjacent cache line prefetcher	1	Busca las líneas que comprenden un par de 128 bytes en la cache L2
DCU prefetcher	2	Busca la siguiente línea en la cache L1 de datos
DCU IP prefetcher	3	Clasifica las loads en base a su PC y, si detecta patrones en los accesos de una misma load, inicia la prebúsqueda

Tabla 2. Bits de configuración de la prebúsqueda en el MSR 0x1A4.

Para evitar errores en la configuración de los registros MSR, se proporcionan los scripts `activa_prefetch.sh` y `desactiva_prefetch.sh`. Al ejecutar los scripts, se solicitará la contraseña de usuario ya que las instrucciones `rdmsr` y `wrmsr` requieren permisos de administrador para poder ejecutase. Basta con introducir la contraseña de usuario, pues las cuentas se han configurado para poder ejecutar estas instrucciones con permisos de superusuario.
Para comprobar la configuración de la prebúsqueda se puede ejecutar el siguiente comando y observar el valor del registro. Si los cuatros bits de menor peso están a 0, los cuatro mecanismos de prebúsqueda están activados, mientras que, si están a 1, están desactivados.

```
$ sudo rdmsr 0x1a4
```

Actividad 2

Analiza como varía el IPC y los ciclos de parada de los benchmarks estudiados en función de si la prebúsqueda está activada o desactivada.
Para ello, **desactiva la prebúsqueda** del procesador y repite los experimentos del ejercicio anterior para obtener el IPC y los ciclos de parada.
A continuación, representa las mismas figuras que en el apartado anterior, pero con dos barras para cada aplicación, una con la prebúsqueda desactivada y otra con la prebúsqueda activada.
A modo de ejemplo, se presentan las siguientes dos figuras que permiten visualizar fácilmente las diferencias, en IPC y ciclos de parada, entre ambas configuraciones.

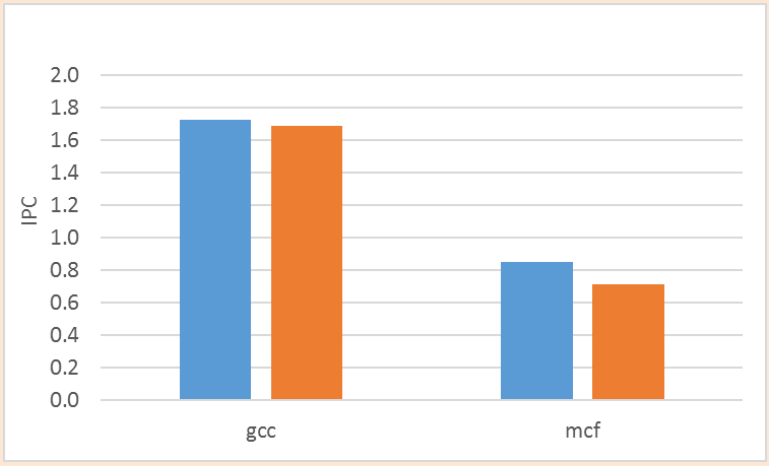


Figura 5. IPC para los benchmarks estudiados con la prebúsqueda activada y desactivada.

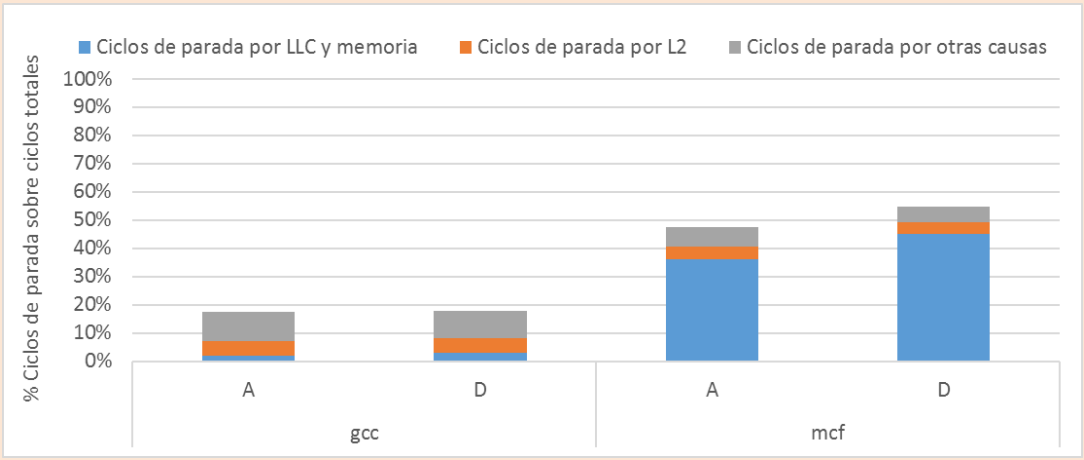


Figura 6. Ciclos de parada para los benchmarks estudiados con la prebúsqueda activada y desactivada.

¿Cuál es el componente principal de los ciclos de parada que se reduce? ¿Cómo contribuye la prebúsqueda a la reducción de este componente?

Actividad 3. Análisis de las características de las aplicaciones que más se benefician de los mecanismos de prebúsqueda

Observa que los ciclos de parada que se reducen en mayor medida gracias a la prebúsqueda son los que aciertan en la cache L2. Este hecho es indicativo de que la prebúsqueda permite anticipar a L1 o a L2 muchos de los datos que, sin prebúsqueda, estarían en la LLC o en memoria principal, reduciendo de esta forma el tiempo de acceso a los mismos.

A partir de esta observación, *¿qué comportamiento, desde el punto de vista de la jerarquía de cache, crees qué debe presentar una aplicación para beneficiarse de la prebúsqueda?*

Confírmalo experimentalmente, configurando los eventos apropiados en el script script_obtener_ciclos_parada.sh para poder calcular el número de fallos por cada mil instrucciones (MPKI) en las caches L1, L2 y LLC. En primer lugar, realiza la evaluación con los mecanismos de prebúsqueda desactivados y, posteriormente, activados. A continuación, se proporciona la lista de eventos a evaluar. ¿Qué características presentan los MPKIs (misses per kilo instructions) de las aplicaciones que se ven más y menos beneficiadas por los mecanismos de prebúsqueda?

Evento	Descripción
mem_load_uops_retired:l1_miss	Número de *loads* que fallan en L1
mem_load_uops_retired:l2_miss	Número de *loads* que fallan en L2
mem_load_uops_retired:l3_miss	Número de *loads* que fallan en la LLC

Nota: el MPKI representa los fallos por cada 1000 instrucciones ejecutadas. Para una determinada cache (L1 datos, L2 o LLC) se calcula como:
número total de fallos de dicho nivel dividido por número instrucciones ejecutadas dividido por 1000
Se aconseja calcular el $MPKI_{L1}$, $MPKI_{L2}$ y $MPKI_{L3}$.