Ingeniería Informática
Facultad de informática
Universidad Complutense de Madrid

Sistemas Informáticos

Proyecto fin de carrera

Modificación de la herramienta SimpleScalar para el estudio de las arquitecturas asíncronas.

Natalia García Casado
Enrique Gutiérrez Sancho

Introducción

El enorme coste en complejidad, tiempo y dinero que supone diseñar físicamente componentes hardware, obliga a depender de la simulación para desarrollar nuevas ideas en arquitectura de computadores.

El objetivo de nuestro proyecto consiste en desarrollar una herramienta que simule el comportamiento de una arquitectura asíncrona, para posteriormente realizar un estudio que demuestre que se puede un obtener un mayor rendimiento con este tipo de sistemas y que, por tanto, seria viable trabajar en el diseño de la misma.

Para crear este simulador, partiremos de la herramienta simplescalar. SimpleScalar es un software de simulación de arquitecturas avanzadas, utilizado para el análisis de funcionamiento de programas y el modelado de microarquitecturas.

Las herramientas de SimpleScalar se utilizan extensamente para la docencia e investigación, en el año 2000 más de un tercio de los investigadores en arquitectura de computadores utilizó SimpleScalar para evaluar sus diseños.
Las herramientas se distribuyen con todo el código de fuente, haciendo posible que los usuarios amplíen SimpleScalar, y adapten modelos existentes a sus propias ideas.

Además de distintos simuladores, SimpleScalar incluye herramientas de visualización del funcionamiento, análisis estadístico e infraestructura para su depuración.

SimpleScalar fue creado por Todd Austin, auque las primeras versiones del sistema incluyeron contribuciones de Doug Burger y Guri Sohi, hoy, SimpleScalar es desarrollado y apoyado por SimpleScalar LLC.

De los cinco simuladores incluidos en Simplescalar, nos centraremos en *Sim-Outorder*. Este simulador soporta ejecución especulativa y fuera de orden basada en el renombramiento de registros. Partiendo del módulo sim-outorder, obtendremos uno nuevo que devuelva como salida tiempos correspondientes a una ejecución asíncrona basada en la distinción tanto de instrucciones, como de operandos utilizados.

Una vez terminado este módulo, realizaremos una serie de pruebas que nos permitan llegar a una conclusión en nuestro estudio: por una parte ejecutaremos los benchmarks SPEC2000 para obtener resultados reales referentes a las estadísticas de instrucciones ejecutadas y por otra, ejecutaremos distintos tests que serán procesados por el temporizador para comparar su rendimiento en ambas arquitecturas.

Base teórica

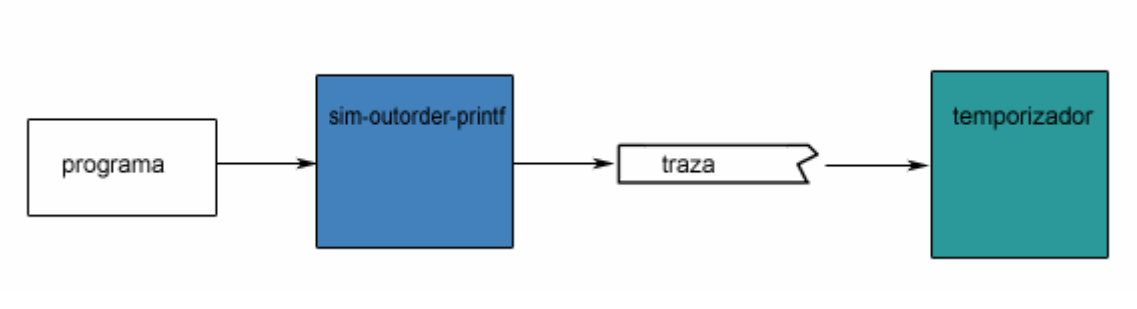
Desarrollo del simulador para arquitectura asíncrona.

Partiendo del simulador sim-outorder de SimpleScalar, desarrollamos un nuevo módulo, llamado *sim-outorder-printf* en el cual se recogen todas las modificaciones efectuadas en *sim-outorder* además de generar una traza de los tiempos que pasa cada instrucción en cada etapa.

Nuestro simulador basado en traza, se desarrolla como un módulo independiente: *temporizador*, el cual procesa la traza generada por *sim-outorder-printf* y devuelve como salida los tiempos estimados para una ejecución asíncrona del programa.

Toda la información sobre el simplescalar se puede encontrar en:

www.simplescalar.com



Notas sobre la arquitectura de los procesadores de propósito general.

Para el desarrollo de sistemas de procesamiento con mayor capacidad, los diseñadores han basado sus estrategias en las técnicas de integración de transistores en un chip, iniciando con bajas escalas de integración *LSI*, pasando por escalas de alta integración (*0.35)* hasta llegar a una muy alta capacidad de integración (*VLSI – 0.25)*.

Hoy se habla incluso de niveles superiores de integración, como la escala (*ULSI – 0.12)* con capacidad de almacenar miles de millones de transistores en una placa de silicio. Sin embargo, se ha demostrado que la capacidad de integrar transistores en una placa de silicio tiene un límite físico y al parecer dicho límite no se encuentra muy distante de los niveles que se alcanzan en la actualidad.

Una vez alcanzado el límite físico impuesto por el material, será mucho más complicado o casi imposible seguir aumentando la capacidad de procesamiento de sistemas por medio de la integración. Estas razones han obligado a los diseñadores de procesadores a pensar en nuevas alternativas para mejorar sus diseños.

Surge de esta manera la idea de desarrollar nuevas arquitecturas de procesadores como propuesta para mejorar el rendimiento de un sistema.

Remontándonos a los primeros diseños del MIPS, podemos hacer un recorrido por las nuevas ideas en el diseño de arquitectura de computadores que impulsaron el gran aumento del rendimiento de los procesadores hasta la actualidad.

La primera idea fue dividir la ejecución de las instrucciones en distinta etapas, esto es, pasar de la ruta de datos monociclo a la multiciclo, lo cual aumenta de forma sorprendente el rendimiento del computador, no solo en cuanto al tiempo de ciclo sino también en los costes ya que esto permite la posibilidad de reutilizar hardware. Ya no es necesario que el tiempo de ciclo sea igual que el tiempo de ejecución de la instrucción más lenta, sino que el periodo del reloj debe ser igual al de la etapa de ejecución más lenta, reduciendo los desperdicios a un nivel mínimo. Además si en una instrucción se necesitan realizar varias sumas, se puede realizar cada suma en una etapa con el consiguiente ahorro de hardware.

Más adelante se avanzaría todavía aún más introduciendo la segmentación en la ruta de datos. Mediante la segmentación se consigue solapar la ejecución de instrucciones, de manera que haya tantas instrucciones ejecutándose simultáneamente como etapas se tengan en la ruta de datos. Ahora el principal objetivo pasa a ser conseguir un rendimiento de una instrucción por ciclo.

Este rendimiento ideal, nunca llega a conseguirse totalmente debido a que una de la desventajas de la segmentación es la aparición de conflictos estructurales, de datos y de control.

Algunas de estas dependencias consiguen solventarse por medio de la anticipación de operandos, reordenamiento de código y detención del pipeline, pero el rendimiento real sigue estando algo lejos de una instrucción por ciclo.

Se sigue avanzando con nuevas ideas: la ejecución fuera de orden: si una instrucción es independiente (no presenta dependencias) y se dispone de unidades funcionales desocupadas ¿Por qué no ejecutarla fuera de orden? En vez de parar el pipe, cuando se presentan conflictos y dependencias que no se pueden resolver, se puede seguir con la ejecución de otras instrucciones que estén preparadas y no presenten dependencias.

Surgen entonces distintos algoritmos para realizar planificación dinámica:

- SCOREBOARD: permite ejecución fuera de orden si hay recursos libres y los datos están disponibles.
- TOMASULO: mediante la técnica de renombramiento de registros se consiguen resolver más dependencias.

En este escenario aparece también la ejecución especulativa, cuando hay un salto condicional en vez de esperar a resolver si se toma o no, se continúan ejecutando instrucciones (especuladas) y una vez que se resuelve las instrucciones se completan o se descartan. Esto dio lugar a los predictores de saltos que presentan muy buen comportamiento en bucles.

Bien, pues siguiendo con la idea de conseguir un diseño que se pueda acercar al rendimiento ideal de un procesador, aparece la idea de una arquitectura asíncrona.

En un procesador síncrono, cada operación esta sincronizada por la señal de reloj. Un procesador asíncrono no tiene reloj, cada parte del hardware se sincroniza mediante señales. La principal ventaja de este tipo de arquitecturas es que cada unidad puede ejecutar operaciones a su propia velocidad, mientras que en un procesador

asíncrono la velocidad esta limitada por el tiempo de ciclo, el cual se corresponde con el tiempo de ejecución de la etapa más lenta.

Un sistema asíncrono también consume menos energía, ya que en uno síncrono gran parte de la energía se destina en generar la señal de reloj. Además un sistema asíncrono produce muchas menos interferencias electromagnéticas porque las distintas actividades no se realizan en los mismos instantes de tiempo.

Debido a estas ventajas, la investigación en este campo sigue avanzando y actualmente se dispone ya de varios microprocesadores asíncronos aunque las grandes compañías como IBM, AMD e Intel no los han implantado, en parte por el coste que supondría.

En resumen, la principal característica de la ejecución asíncrona es que permite que se ejecute una instrucción sin necesidad de esperar a que se complete la ejecución asíncrona anterior. Por tanto, la secuencia de ejecución de las operaciones asíncronas se puede modificar siempre que las dependencias de los datos permanezcan invariables.

Hay que tener en cuenta, que al introducir este factor de ejecución fuera de orden aparecen nuevos conflictos y dependencias por lo que no siempre es posible aplicar la ejecución asíncrona.

Es necesario, por la tanto realizar un estudio exhaustivo que produzca en primer lugar resultados referentes a los tipos de instrucciones que serian candidatas a ejecutarse asincronamente, y hacer después una estimación sobre el rendimiento que se puede obtener al aplicar la ejecución asíncrona en dichas instrucciones.

Enlaces con información de las arquitecturas asíncronas:
http://www.async.caltech.edu
http://www.cs.man.ac.uk/async/
http://www.win.tue.nl/async-bib
http://www.hanssummers.com/computers/ttlcpu/intro.htm
http://www.async04.gr

Proceso de desarrollo

FASE 1: modificación de sim-outorder

El primer objetivo de la fase de modificaciones en Simplescalar es tomar tiempos de la ejecución de las instrucciones como el tiempo que pasa en el pipe cada una y tiempo entre las distintas etapas.

También es necesario añadir variables estadísticas para contabilizar los tipos de instrucciones ejecutadas en un programa y los operandos que utilizan, estas estadísticas servirán más adelante para diseñar las unidades funcionales asíncronas.

FASE 2: generación de la traza de ejecución

Una vez modificado sim-outorder, lo siguiente es generar una traza de ejecución para su posterior tratamiento. La traza debe describir no solo la secuencia de ejecución sino también los tiempos de una instrucción en cada etapa.

FASE 3: ejecución de SPEC

Antes de simular una ejecución asíncrona, es necesario conocer la frecuencia de las instrucciones para poder dar una estimación de los tiempos de cada instrucción. Para ello, ejecutamos en el simplescalar, ya modificado, los SPEC2000 (Integer y Float) y obtenemos una serie de estadísticas.

FASE 3: simulación de la ejecución asíncrona:

Para realizar una simulación de la ejecución asíncrona de los programas, programamos un temporizador. El temporizador procesa la traza generada por el Simplescalar y a partir de ella calcula los nuevos tiempos de ejecución y da como salida el tiempo total asíncrono, teniendo en cuenta el tipo de cada instrucción y las unidades funcionales.

FASE 4: pruebas y conclusión

Por último, debemos de realizar unas pruebas con trazas de tests ejecutados a fin de obtener las correspondientes simulaciones asíncronas para comprobar si realmente se consigue el rendimiento esperado con una arquitectura asíncrona.

Trabajo realizado

Modificaciones del Código

1. Las primeras modificaciones del sim-outorder se hicieron para tomar contacto con el código y familiarizarse con él.
Así pues, la primera modificación fue contar el número de instrucciones de salto condicional e incondicional y mostrar los porcentajes respecto a los saltos en total.
Para ello, se añadieron cuatro variables para la estadística: Dos para los saltos que se ejecutan y dos para los saltos que llegan a la fase commit. Los nombres de las variables se muestran a continuación:

```
sim_num_branch_cond        <- saltos condicionales que llegan a commit
sim_total_branch_cond      <- saltos condicionales que se ejecutan
sim_num_branch_uncond      <- saltos incondicionales que llegan a commit
sim_total_branch_uncond    <- saltos incondicionales que se ejecutan
```

Todas estas variables son de tipo *counter_t*, tipo definido en la librería *host.h*.

Estas variables se van actualizando en la fase de *dispatch* por la razón que en el propio código muchas variables de la estadística las actualiza en esta fase.
Para comprobar los tipos de instrucciones usamos los siguientes flags:

```
F_CTRL     <- activo si la instrucción es de control.
F_COND     <- activo si la instrucción es un salto condicional.
F_UNCOND   <- activo si la instrucción es un salto incondicional.
```

Y para comprobar si la instrucción llegará o no a la fase de *commit*, comprobando el flag *spec_mode*, podemos saber si la instrucción está siendo especulada o no.

2. Otra modificación fue comprobar los ciclos que pasa una instrucción en el pipe, controlando la entrada y salida en cada etapa. Para ello, se creó la siguiente estructura, la que nos servirá para guardar los ciclos de inicio en los que una instrucción ha entrado por cada etapa:

```
struct time_ptrace
{
  tick_t c_fetch;     /* ciclo inicio fetch*/
  tick_t c_dispatch;  /* ciclo inicio dispatch*/
  tick_t c_exe;       /* ciclo inicio issue*/
  tick_t c_writeback; /* ciclo inicio writeback*/
  tick_t c_commit;    /* ciclo inicio commit*/
  struct traza_c_exe datos_exe; /* datos adicionales en issue*/
};
```

(tick_t también definido en host.h)

Como en momento de la traza puede haber hasta 64 instrucciones en el pipe, la variable que guardase estos datos debía ser una tabla (un array) con tantas entradas como instrucciones fueran posibles que hubiese en un momento

determinado en el pipe.

Esta variable es *time_ptrace_data*, que se inicializa con la función *time_ptrace_init* a partir de una constante *TAM_TIME_PTRACE* donde daremos el tamaño de esta tabla. Esta función es llamada desde el procedimiento *sim_load_prog*.

La actualización de esta variable vendrá dada por el ciclo de inicio en cada etapa y el número de secuencia de la instrucción. Como este número de secuencia puede ser muy grande (del orden de miles de millones) y no hay memoria capaz de guardar tablas tan grandes, se decidió que puesto en cada momento del pipe no puede haber más de 64 instrucciones, se hiciese un módulo al valor del número de secuencia para tener una posición en el array para asignárselo a una instrucción.

Por tanto en cada etapa del pipe dadas por las funciones: *ruu_fetch(), ruu_dispatch(), ruu_issue(), ruu_writeback()* y *ruu_commit()*, actualizamos *time_ptrace_data* que nos servirá para la siguiente modificación realizada en el *sim-outorder*.

3. La siguiente modificación fue un poco más compleja. Queríamos mostrar en la traza de salida del pipe, además de la información que ya se proporciona, los datos referentes a la entrada y salida en cada etapa por las instrucciones y los ciclos que se han pasado en el pipe desde *fetch* hasta *commit*.

   Para ello tuvimos que modificar los archivos *ptrace.h* y *ptrace.c*, pues ellos son los que escribían la traza de salida. Se añadieron funciones para escribir en el archivo de traza que mostraran el ciclo de entrada, de salida y la diferencia por cada etapa y para el pipe entero. Estas funciones son:

   ```
   ptrace_diffcycle(…)
   ```

   Para mostrar el nº de secuencia de la instrucción, el ciclo de entrada en el pipe, el ciclo de salida y la diferencia.

   ```
   ptrace_stage_fetch(…)
   ptrace_stage_dispatch(…)
   ptrace_stage_issue(…)
   ptrace_stage_writeback(…)
   ptrace_stage_commit(…)
   ```

   Lo mismo pero para cada etapa.

4. Ahora viene la parte en la que se añaden estadísticas al simulador. Queremos contabilizar:

   Instrucciones aritméticas:

   - Operaciones de suma enteras
   - Operaciones de suma enteras con inmediato
   - Operaciones de suma enteras con algún operando 0

- Operaciones de suma en punto flotante
- Operaciones de suma en punto flotante con algún operando 0

- Operaciones de resta enteras
- Operaciones de resta enteras con inmediato
- Operaciones de resta enteras con algún operando 0
- Operaciones de resta en punto flotante
- Operaciones de resta en punto flotante con algún operando0

- Operaciones de multiplicaciones enteras
- Operaciones de multiplicaciones en punto flotante
- Operaciones de multiplicaciones enteras por 0
- Operaciones de multiplicaciones enteras por 1
- Operaciones de multiplicaciones enteras por 2
- Operaciones de multiplicaciones de punto flotante por 0
- Operaciones de multiplicaciones de punto flotante por 1
- Operaciones de multiplicaciones de punto flotante por 2

- Operaciones de divisiones enteras
- Operaciones de divisiones en punto flotante
- Operaciones de divisiones enteras por 1
- Operaciones de divisiones enteras por 2
- Operaciones de divisiones en punto flotante por 1
- Operaciones de divisiones en punto flotante por 2

Registros utilizados (enteros y punto flotante)

- Numero de registros menores de 2^8
- Numero de registros menores de 2^16
- Numero de registros menores de 2^24
- Numero de registros menores de 2^32

Acarreos en operaciones aritméticas

- Variables que cuentan donde se rompe el acarreo en la suma con intervalos desde 2^8 a 2^32 de 8 en 8 bytes.

Para cada una, se añadió una variable para la estadística y se mostraron además porcentajes que ayudasen a leer los datos mostrados. Todas estas modificaciones se realizaron en la función de *ruu_issue()* que es donde se han resuelto ya los operandos y se ejecutan las instrucciones.

Se añadieron dos métodos: uno para transformar una secuencia de números a binario y otro para contabilizar el número de veces que se rompe el acarreo en la suma y el dónde se rompió el último acarreo:

```
rompeAcarreo(…)
DecToBin(…)
```

Notas sobre la implementación de esta parte:

- En el caso de las sumas y las restas, se actualizaron en el caso que las instrucciones fueran de suma y resta. Pues hay otras operaciones que no son aritméticas que también operan con sumas y restas.
- En el caso del producto y la división, la actualización se llevó a cabo con los flags disponibles.
- La cuenta del acarreo se hizo solo en el caso de sumas enteras.

5. Ultimas modificaciones:

   - Calcular tiempos medios en ciclos que se pasan las instrucciones en cada etapa:
     Usando dos variables por etapa, una para el número de instrucciones que pasa por una etapa y otra que vaya acumulando los ciclos que ha pasado en esa etapa. Después mostramos la media de ciclos por etapa.

   - Contabilizar uso de unidades funcionales:
     Una variable que se irá actualizando según vayan pasando las instrucciones por cada unidad funcional.

   - Añadidos en modificaciones anteriores:
     Información adicional en la variable *time_ptrace_data* que nos permita tener información de la etapa *issue* para posteriormente mostrarla en la traza. La estructura usada fue:

```
struct traza_c_exe
{
  int tipo;  /* 0 suma, 1 resta, 2 multiplicacion , 3
division . -1 otra*/
  int pf;  /* si es de punto flotante o no  .-1 otra*/
  int valor;  /* si tiene operandos 0,1 o 2 . -1 otra*/
};
```

     Se amplió la función *ptrace_stage_issue()* para permitir mostrar estos datos adicionales.

6. Mejora de tiempos en la generación de trazas:

   Al hacer pruebas en la generación de trazas, se observó que con las funcionalidades añadidas al programa, este tardaba mucho más que el programa original. Así que se buscaron alternativas para mostrar las trazas y que no durase tanto tiempo.

   La solución propuesta fue sustituir las funciones de traza por funciones que imprimiesen en la salida estándar (printf) la información de la traza.

   Después de varias pruebas se comprobó que esta solución mejoraba los tiempos de los programas anteriores. Pero como se necesitaba un redireccionamiento de la salida a un fichero para que no inundase la pantalla de los datos de la traza, además de una observable bajada del

rendimiento del programa. Se optó por crear una versión del *sim-outorder* paralela a la que habíamos modificado. Esta versión se llamó *sim-outorder-printf*, que es la que se usó para generar las trazas de las pruebas.

Temporizador.

Introducción

El temporizador es la herramienta que vamos a usar para comparar los resultados de una ejecución síncrona a una asíncrona. Con ello, queremos comprobar si merece la pena hacer transformaciones asíncronas en el pipe para aumentar el rendimiento de un procesador.

El programa interpreta el fichero de traza generado por *sim-outorder* o *sim-outorder-printf*. Y muestra los valores de tiempo generados a partir de los valores de configuración que se le hayan introducido. Estos ficheros nos permitirán dar valores a cada etapa del pipe e incluso de cada tipo de instrucción.

Los ficheros de configuración del temporizador nos permiten dar modificar los valores en ciclos por unidad funcional, dar a cada uno de esos valores una constante multiplicativa que se traducirá en tiempo.

En este proyecto nos interesaba hacer un estudio exhaustivo de las instrucciones aritméticas a la hora de entrar en la etapa de *issue*, pues es en dónde se pueden intentar reducir los tiempos en el pipe. Por ello, en el fichero de configuración hay variables por cada tipo de instrucción y el valor que tengan sus operandos. Así, podemos dar valores a todas estas combinaciones:

Para etapa *Issue*:
- Sumas y restas enteras diferenciado valores de operando con cero.
- Sumas y restas de punto flotante y de operandos cero.
- Productos enteros y de punto flotante con valor de operando cero, uno y dos.
- Divisiones enteras y de punto flotante con valor de operando uno y dos.
- Para cada una de estas operaciones también, pero con operando distinto a los antes indicados.

Resto de etapas
- Constantes multiplicativas para *Fetch*, *Dispatch*, *Writeback* y *Commit.*

Compilar el programa

Desde la carpeta donde tenemos situados los programas fuentes: make

Escribir: make clean, si se desea limpiar la carpeta de ficheros auxiliares usados en la compilación.

Modo de uso

Desde la carpeta donde tenemos situado el programa:
./tempor [-a fichero] fichero-traza

- Si no se pone la opción –a y a continuación un fichero, el programa lee default.cfg por defecto.
- El fichero traza debe ser el generado por los programas *sim-outorder* o *sim-outorder-printf*.

Salida del programa

El programa nos mostrará de salida:
- Ciclos de ejecución y los ciclos totales acumulados.
- El tiempo de ejecución y el tiempo total acumulado.
- Por cada etapa, ciclos máximo, mínimo y media de paso por la etapa.
- Igual que antes, pero en tiempos.

Detalles de implementación del temporizador

El fichero de traza contiene información sobre los ciclos de entrada y salida de cada etapa (por las instrucciones) y en el caso especial de la etapa *issue*, unos códigos que indicarán al programa el tipo (suma, resta, etc..), si es entera o punto flotante y el valor de los operandos.

El temporizador según va leyendo línea a línea el fichero, va actualizando sus variables contador y sus estadísticas. Consideramos que la diferencia de tiempo entre dos instrucciones en la diferencia de valores entre sus etapas de *commit*, que es cuando terminan. Así calculamos los ciclos que tarda la simulación en realizarse, que será igual a las mostradas por las estadísticas del *sim-outorder*. (A partir de aquí, me referiré al *sim-outorder* o al *sim-outorder-printf* como sim-outorder indistintamente para abreviar).

También vamos acumulando en una variable los tiempos que pasan en el pipe cada instrucción para que podamos ver la diferencia de en las estadísticas usando unos valores de configuración u otros. Aunque este medida no tiene mucho uso a efectos prácticos, siendo más útil los ciclos de ejecución.

Por cada etapa, se leen los ciclos de inicio y fin que una instrucción pasa en ella. Este valor, es el que usamos para calcular los ciclos de ejecución. En el caso especial que la instrucción esté entre las habladas anteriormente, se sumará los ciclos indicados en los ficheros de configuración. Estos ciclos, después se operarán por las constantes multiplicativas para hallar una equivalencia en tiempo.

Notas del programa:
Se ha comprobado en las pruebas que con ficheros de traza muy grandes (mayores a 800 megas) se pueden datos de salida erróneos.

Pruebas realizadas

Ejecución de tests y conclusiones

Introducción

Los programas que se ejecutaron con el *Simplescalar*, para probar los tiempos, asíncronos fueron los tests que vienen con el código fuente del simulador. Por falta de recursos no se pudieron hacer pruebas más reales con los programas SPEC, que nos hubiesen sido más útiles para nuestros propósitos.

Los programas ejecutados fueron:

- test-math: Realiza varias operaciones matemáticas con datos en punto flotante.
- test-printf: Muestra en la salida resultados de operaciones sencillas en distintos formatos.
- test-dirent: Muestra el contenido del directorio que le pasemos como argumento.
- test-fmath: Realiza operaciones en punto flotante.
- test-llong: Muestra por pantalla datos de tipo long long de C

Se ejecutaron estos programas con el programa *sim-outorder* generando el archivo de estadísticas del simulador, la salida del propio programa y la traza. Un ejemplo de cómo se ejecutaron:

./sim-outorder-printf –redir:sim sim.out –redir:prog prog.out test > traza.trc

Seguidamente se analizaron las trazas con el programa *tempor* con la configuración que aparece en el apéndice A.

Programas usados y resultados

Se modificaron las constantes multiplicativas en las instrucciones aritméticas que caen dentro de la etapa *Issue*, con el objetivo que al hacer el análisis de las trazas.
Primero vamos a mostrar los resultados de los análisis para cada programa:

- **test-dirent:**

  Podemos observar que este programa no tiene casi operaciones de punto flotante y el uso de los registros de punto flotante es mínimo. En cuanto a las operaciones enteras, observamos:
  - 41% del total de sumas , tiene algún operando cero
  - 25% de las restas, tiene algún operando cero también.
  - 21% y 16 % de las multiplicaciones son con operando cero o uno.

  En la salida generada por el programa *tempor*, se observa cierta mejor en los

tiempos medios de la etapa *issue*. Eso se ve reflejado también en los tiempos de ejecución y los tiempos totales acumulados.

- **test-fmath:**

  Programa pequeño que solo ha generado 20503 instrucciones con poco uso de las unidades funcionales y de los registros de punto flotante. Datos significativos:
  - 37% de sumas enteras con operando cero
  - 22% de las restas enteras con operando cero
  - 28% y 21% de las multiplicaciones enteras con operando cero y uno respectivamente.

  En la salida generada del programa *tempor*, no observamos una mejoría importante en cuanto a los resultados síncronos. Los tiempos medios en la etapa de *Issue* no son tan importantes como para decir que en "modo asíncrono" funciona mejor este programa.

- **test-llong:**

  Programa también cortito, de 11796 instrucciones. Con un uso mínimo de los registros de punto flotante. Las operaciones matemáticas de punto flotante son casi inexistentes.

  Es significativo que el 34% de las operaciones enteras sean con operado cero. El resto de datos tiene poca relevancia por las pocas operaciones que se realizan con él.

  En los resultados del temporizador en "modo asíncrono" no se observan mejoras significativas por las pocas operaciones en las que podíamos reducir tiempos.

- **test-math:**

  En este programa es en el que se realizan más operaciones matemáticas y en el que esperamos observar las mejoras del "modo asíncrono". Ya se observa un uso más intensivo de los registros y de las operaciones en punto flotante. Datos significativos:

  - 34% de las sumas enteras son con operando cero
  - 23% y 25% de las restas con operando cero enteras y punto flotante respectivamente
  - 28% de las multiplicaciones de punto flotante son con operando ceo
  - 16% de las multiplicaciones enteras son con operando 1.

  Los resultados generados por el temporizador nos dicen que en "modo asíncrono" hay una reducción de tiempo, no notable, pero si a tener en cuenta. Aunque la media de tiempo en la etapa *issue* no se ha reducido mucho.

- **test-printf:**
  Es el programa que más instrucciones ha generado: 1038348 de instrucciones. Inexistentes las operaciones de punto flotante y su uso con los registros. Solo operaciones enteras y no muchas.

  Datos significativos:
  - 35% de las sumas enteras son con operando cero
  - 28% de las restas enteras son con operando cero
  - 50% de las multiplicaciones enteras son con operando uno.

  Los resultados del temporizador nos dicen que se han reducido los tiempos globales y en la etapa de *issue*, pero que no son tan significativos como para optar por el "modo asíncrono"

Las tablas con las estadísticas del simulador y la salida del temporizador las podemos encontrar en el apéndice A.

Conclusiones

En pronto para concluir que las arquitecturas asíncronas sean mejores que las síncronas a partir de estos primeros resultados. Hay que tener en cuenta que el programa del temporizador no es una versión final que tenga en cuenta los detalles de funcionamiento del pipe (dependencia de datos, algoritmos dinámicos, etc.). Tampoco sabemos si la reducción de tiempos en una etapa de la arquitectura supone un mayor coste en tiempo en otras etapas.

Además, por falta de recursos técnicos, no se pudieron generar las trazas de los programas spec que son los que nos darían estadísticas reales sobre el funcionamiento del pipe en programas comunes para su mejor aprovechamiento.

Aún con lo dicho anteriormente, si en la ruta de datos se hiciesen cambios, dependiendo de un tipo de instrucción y el valor de sus operandos, para que tardase menos en ciertas circunstancias, el rendimiento de la arquitectura mejoraría bastante. Se ha comprobado que un gran número de operaciones de la ALU (sumas, restas… mirar en Apéndice B), son con uno de los operandos cero. Añadiendo elementos en el pipe para que detecten estos operandos ceros y dar el resultado directo sin tener que hacer los cálculos, reduciría bastante los tiempos de ejecución.

En resumen, el estudio de la viabilidad de la arquitectura asíncrona, requiere aún más estudio y pruebas con simuladores como el *Simplescalar*. Y conseguir una serie de herramientas para poder hacer también estudios de consumo, potencia, costes de componentes tan importantes a la hora de diseñar procesadores de propósito general. Esperamos que con las modificaciones del simulador *sim-outorder* y el programa *tempor* ayuden a continuar este estudio.

Apéndices

Apéndice A: Ejecución y pruebas con los Tests.

Datos del fichero de configuración: *config.cfg*

```
#
# Fichero de configuración
#
# Formato:
# OPCION    valor
#
#
# Valores en ciclos por unidad funcional
#

# Sumas enteras
SUM_ENT           1

# Suma enteras con operando cero
SUM_ENT_0         1

# Sumas en punto flotante
SUM_FP            2

# Sumas punto flotante con operando cero
SUM_FP_0          2

# Restas enteras
RESTA_ENT         1

# Restas enteras con operando cero
RESTA_ENT_0       1

# Restas en punto flotante
RESTA_FP          2

# Restas punto flotante con operando cero
RESTA_FP_0        2

# Multiplicaciones enteras
MULT_ENT          3

# Multiplicaciones enteras con operandos cero, uno o dos
MULT_ENT_0        3

MULT_ENT_1        3

MULT_ENT_2        3

# Multiplicaciones en punto flotante
MULT_FP           4

# Multiplicaciones en punto flotante con operandos cero, uno o dos
MULT_FP_0         4

MULT_FP_1         4

MULT_FP_2         4

# Divisiones enteras
DIV_ENT            20

# Divisiones enteras con denominador 1 o 2
DIV_ENT_1         20

DIV_ENT_2         20

# Divisiones en punto flotante
DIV_FP            12

# Divisiones en punto flotante con denominador 1 o 2
DIV_FP_1          12
```

```
DIV_FP_2          12

#
# Valores de tiempos en nanosegundos
#

# Sumas enteras
T_SUM_ENT         1.0

# Suma enteras con operando cero
T_SUM_ENT_0       0.4

# Sumas en punto flotante
T_SUM_FP           1.0

# Sumas punto flotante con operando cero
T_SUM_FP_0        0.5

# Restas enteras
T_RESTA_ENT       1.0

# Restas enteras con operando cero
T_RESTA_ENT_0     0.3

# Restas en punto flotante
T_RESTA_FP        1.0

# Restas punto flotante con operando cero
T_RESTA_FP_0      0.5

# Multiplicaciones enteras
T_MULT_ENT        1.0

# Multiplicaciones enteras con operandos cero, uno o dos
T_MULT_ENT_0      0.3

T_MULT_ENT_1      0.3

T_MULT_ENT_2      0.7

# Multiplicaciones en punto flotante
T_MULT_FP         1.0

# Multiplicaciones en punto flotante con operandos cero, uno o dos
T_MULT_FP_0       0.5

T_MULT_FP_1       0.5

T_MULT_FP_2       0.8

# Divisiones enteras
T_DIV_ENT         1.0

# Divisiones enteras con denominador 1 o 2
T_DIV_ENT_1       0.3

T_DIV_ENT_2       0.7

# Divisiones en punto flotante
T_DIV_FP          1.0

# Divisiones en punto flotante con denominador 1 o 2
T_DIV_FP_1        0.3

T_DIV_FP_2        0.7

# Etapas
T_FETCH           1.0

T_DISPATCH        1.0

T_ISSUE           1.0

T_WRITEBACK       1.0

T_COMMIT          1.0
```

Pruebas realizadas

test-dirent

Salida generada por el temporizador:

```
Ciclos en ejecucion: 198581
Ciclos totales : 2781533


Estadísticas en ciclos de etapas
est_fetch.max = 65
est_fetch.min = 1
est_fetch.media = 1.301
est_dispatch.max = 68
est_dispatch.min = 1
est_dispatch.media = 2.226
est_issue.max = 4
est_issue.min = 1
est_issue.media = 1.006
est_writeback.max = 66
est_writeback.min = 1
est_writeback.media = 2.241
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 72
est_total.min = 5
est_total.media = 7.811

Tiempo : 185957.60
Tiempo total: 2768088.30


Estadísticas en nanosegundos de etapas
est_fetch_tmp.max = 65.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.301
est_dispatch_tmp.max = 68.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.226
est_issue_tmp.max = 4.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 0.968
est_writeback_tmp.max = 66.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 2.241
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 72.000
est_total_tmp.min = 4.300
est_total_tmp.media = 7.774
```

Resultados tests-dirent

Instrucciones

Instrucciones ejecutadas	386.856
Instrucciones a commit	344.451

Ciclos

Número total de ciclos	198.580
Instrucciones por ciclo	1,7346

Media ciclos entre etapas

Fetch - Dispatch	1.3014
Dispatch - Issue	2.3741
Issue - Writeback	1.0217
Writeback - Commit	2.9889

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	59.077	15,27%
Stores	29.834	7,71%
Total:	**88.911**	**22,98%**
Saltos		
Saltos condicionales	48.517	12,54%
Saltos incondicionales	14.045	3,63%
Total:	**62.562**	**16,17%**
Instrucciones enteras		
Add	33.297	8,61%
Sub	24.705	6,39%
Mult	941	0,24%
Div	0	0,00%
Total:	**58.943**	**15,24%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	1	0,00%
Div	0	0,00%
Total:	**1**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	13.923	41,81%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	6.210	25,14%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	200	21,25%
Operando 1 (int)	152	16,15%
Operando 2 (int)	3	0,32%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	252.320	35,10%
Enteros entre 2^8 y 2^16	17.823	2,48%
Enteros entre 2^16 y 2^24	7.006	0,97%
Enteros entre 2^24 y 2^32	441.659	61,44%
PF menores de 2^8	718.808	100,00%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	0	0,00%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	9.297	95,77%
Entre 2^8 y 2^16	408	4,20%
Entre 2^16 y 2^24	3	0,03%
Entre 2^24 y 2^32	0	0,00%

test-fmath

Salida generada por el temporizador:

```
Ciclos en ejecucion: 23748
Ciclos totales : 165419

Estadisticas en ciclos de etapas
est_fetch.max = 65
est_fetch.min = 1
est_fetch.media = 1.417
est_dispatch.max = 68
est_dispatch.min = 1
est_dispatch.media = 2.343
est_issue.max = 4
est_issue.min = 1
est_issue.media = 1.010
est_writeback.max = 62
est_writeback.min = 1
est_writeback.media = 2.951
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 72
est_total.min = 5
est_total.media = 8.753

Tiempo : 23302.60
Tiempo total: 164846.90

Estadisticas en nanosegundos de etapas
est_fetch_tmp.max = 65.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.417
est_dispatch_tmp.max = 68.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.343
est_issue_tmp.max = 4.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 0.980
est_writeback_tmp.max = 62.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 2.951
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 72.000
est_total_tmp.min = 4.300
est_total_tmp.media = 8.729
```

Resultados test-fmath

Instrucciones

Instrucciones ejecutadas	20.503
Instrucciones a commit	18.204

Ciclos

Número total de ciclos	23.747
Instrucciones por ciclo	0,7666

Media ciclos entre etapas

Fetch - Dispatch	1.4169
Dispatch - Issue	2.5033
Issue - Writeback	1.2237
Writeback - Commit	3.5173

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	3.613	17,62%
Stores	2.453	11,96%
Total:	**6.066**	**29,59%**
Saltos		
Saltos condicionales	2.227	10,86%
Saltos incondicionales	931	4,54%
Total:	**3.158**	**15,40%**
Instrucciones enteras		
Add	1.672	8,15%
Sub	868	4,23%
Mult	52	0,25%
Div	0	0,00%
Total:	**2.592**	**12,64%**
Instrucciones PF		
Add	4	0,02%
Sub	0	0,00%
Mult	1	0,00%
Div	0	0,00%
Total:	**5**	**0,02%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	624	37,32%
Operando 0 (pf)	0	0,00%
SUB		
Operando 0 (int)	199	22,93%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	15	28,85%
Operando 1 (int)	11	21,15%
Operando 2 (int)	2	3,85%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	13.841	36,03%
Enteros entre 2^8 y 2^16	489	1,27%
Enteros entre 2^16 y 2^24	141	0,37%
Enteros entre 2^24 y 2^32	23.943	62,33%
PF menores de 2^8	32.547	84,73%
PF entre 2^8 y 2^16	2.994	7,79%
PF entre 2^16 y 2^24	1.349	3,51%
PF entre 2^24 y 2^32	1.524	3,97%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	447	91,98%
Entre 2^8 y 2^16	34	7,00%
Entre 2^16 y 2^24	2	0,41%
Entre 2^24 y 2^32	3	0,62%

test-llong

```
Ciclos en ejecucion: 17536
Ciclos totales : 97649

Estadisticas en ciclos de etapas
est_fetch.max = 34
est_fetch.min = 1
est_fetch.media = 1.475
est_dispatch.max = 61
est_dispatch.min = 1
est_dispatch.media = 2.341
est_issue.max = 4
est_issue.min = 1
est_issue.media = 1.003
est_writeback.max = 66
est_writeback.min = 1
est_writeback.media = 3.345
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 71
est_total.min = 5
est_total.media = 9.219

Tiempo : 17376.90
Tiempo total: 97388.70

Estadisticas en nanosegundos de etapas
est_fetch_tmp.max = 34.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.475
est_dispatch_tmp.max = 61.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.341
est_issue_tmp.max = 4.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 0.979
est_writeback_tmp.max = 66.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 3.345
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 71.000
est_total_tmp.min = 4.300
est_total_tmp.media = 9.204
```

Resultados test-llong

Instrucciones

Instrucciones ejecutadas	11.796
Instrucciones a commit	10.108

Ciclos

Número total de ciclos	17.535
Instrucciones por ciclo	0,5764

Media ciclos entre etapas

Fetch - Dispatch	1.4747
Dispatch - Issue	2.5097
Issue - Writeback	1.3290
Writeback - Commit	3.8098

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	2.093	17,74%
Stores	1.613	13,67%
Total:	**3.706**	**31,42%**
Saltos		
Saltos condicionales	1.399	11,86%
Saltos incondicionales	647	5,48%
Total:	**2.046**	**17,34%**
Instrucciones enteras		
Add	883	7,49%
Sub	499	4,23%
Mult	9	0,08%
Div	0	0,00%
Total:	**1.391**	**11,79%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	1	0,01%
Div	0	0,00%
Total:	**1**	**0,01%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	301	34,09%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	93	18,64%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	1	11,11%
Operando 1 (int)	5	55,56%
Operando 2 (int)	0	0,00%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	7.365	33,92%
Enteros entre 2^8 y 2^16	624	2,87%
Enteros entre 2^16 y 2^24	384	1,77%
Enteros entre 2^24 y 2^32	13.343	61,44%
PF menores de 2^8	21.716	100,00%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	0	0,00%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	245	89,42%
Entre 2^8 y 2^16	26	9,49%
Entre 2^16 y 2^24	3	1,09%
Entre 2^24 y 2^32	0	0,00%

Test-math

```
Ciclos en ejecucion: 50230
Ciclos totales : 390846

Estadisticas en ciclos de etapas
est_fetch.max = 59
est_fetch.min = 1
est_fetch.media = 1.307
est_dispatch.max = 100
est_dispatch.min = 1
est_dispatch.media = 2.219
est_issue.max = 12
est_issue.min = 1
est_issue.media = 1.039
est_writeback.max = 67
est_writeback.min = 1
est_writeback.media = 2.529
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 104
est_total.min = 5
est_total.media = 8.110

Tiempo : 48770.10
Tiempo total: 389208.80

Estadisticas en nanosegundos de etapas
est_fetch_tmp.max = 59.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.307
est_dispatch_tmp.max = 100.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.219
est_issue_tmp.max = 12.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 1.005
est_writeback_tmp.max = 67.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 2.529
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 104.000
est_total_tmp.min = 4.300
est_total_tmp.media = 8.078
```

Resultados test-math

Instrucciones

Instrucciones ejecutadas	51.880
Instrucciones a commit	46.541

Ciclos

Número total de ciclos	50.229
Instrucciones por ciclo	0,9266

Media ciclos entre etapas

Fetch - Dispatch	1.3075
Dispatch - Issue	2.3628
Issue - Writeback	1.1654
Writeback - Commit	3.1775

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	8.920	17,19%
Stores	5.187	10,00%
Total:	**14.107**	**27,19%**
Saltos		
Saltos condicionales	5.655	10,90%
Saltos incondicionales	1.926	3,71%
Total:	**7.581**	**14,61%**
Instrucciones enteras		
Add	4.610	8,89%
Sub	2.502	4,82%
Mult	153	0,29%
Div	0	0,00%
Total:	**7.265**	**14,00%**
Instrucciones PF		
Add	190	0,37%
Sub	103	0,20%
Mult	308	0,59%
Div	2	0,00%
Total:	**603**	**1,16%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	1.569	34,03%
Operando 0 (pf)	16	8,42%
SUB		
Operando 0 (int)	582	23,26%
Operando 0 (pf)	26	25,24%
MULT		
Operando 0 (int)	4	2,61%
Operando 1 (int)	26	16,99%
Operando 2 (int)	2	1,31%
Operando 0 (pf)	87	28,25%
Operando 1 (pf)	1	0,32%
Operando 2 (pf)	7	2,27%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	32.088	32,96%
Enteros entre 2^8 y 2^16	1.376	1,41%
Enteros entre 2^16 y 2^24	611	0,63%
Enteros entre 2^24 y 2^32	63.289	65,00%
PF menores de 2^8	81.710	83,92%
PF entre 2^8 y 2^16	5.251	5,39%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	10.403	10,68%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	1.205	89,00%
Entre 2^8 y 2^16	63	4,65%
Entre 2^16 y 2^24	19	1,40%
Entre 2^24 y 2^32	67	4,95%

test-printf

```
Ciclos en ejecucion: 589676
Ciclos totales : 7328959

Estadisticas en ciclos de etapas
est_fetch.max = 65
est_fetch.min = 1
est_fetch.media = 1.275
est_dispatch.max = 68
est_dispatch.min = 1
est_dispatch.media = 2.181
est_issue.max = 4
est_issue.min = 1
est_issue.media = 1.005
est_writeback.max = 68
est_writeback.min = 1
est_writeback.media = 2.149
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 72
est_total.min = 5
est_total.media = 7.635

Tiempo : 565133.40
Tiempo total: 7302569.10

Estadisticas en nanosegundos de etapas
est_fetch_tmp.max = 65.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.275
est_dispatch_tmp.max = 68.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.181
est_issue_tmp.max = 4.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 0.978
est_writeback_tmp.max = 68.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 2.149
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 72.000
est_total_tmp.min = 4.300
```

Resultados test-printf

Instrucciones

Instrucciones ejecutadas	1.038.348
Instrucciones a commit	919.845

Ciclos

Número total de ciclos	589.675
Instrucciones por ciclo	1,5599

Media ciclos entre etapas

Fetch - Dispatch	1.2748
Dispatch - Issue	2.3380
Issue - Writeback	1.0137
Writeback - Commit	2.8318

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	191.153	18,41%
Stores	106.383	10,25%
Total:	**297.536**	**28,65%**
Saltos		
Saltos condicionales	116.075	11,18%
Saltos incondicionales	45.403	4,37%
Total:	**161.478**	**15,55%**
Instrucciones enteras		
Add	66.161	6,37%
Sub	46.678	4,50%
Mult	2.020	0,19%
Div	0	0,00%
Total:	**114.859**	**11,06%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	1	0,00%
Div	0	0,00%
Total:	**1**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	23.760	35,91%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	13.466	28,85%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	287	14,21%
Operando 1 (int)	1.001	49,55%
Operando 2 (int)	1	0,05%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	658.351	33,76%
Enteros entre 2^8 y 2^16	39.336	2,02%
Enteros entre 2^16 y 2^24	11.292	0,58%
Enteros entre 2^24 y 2^32	1.240.947	63,64%
PF menores de 2^8	1.350.542	69,26%
PF entre 2^8 y 2^16	76.838	3,94%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	522.546	26,80%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	15.730	96,07%
Entre 2^8 y 2^16	564	3,44%
Entre 2^16 y 2^24	21	0,13%
Entre 2^24 y 2^32	59	0,36%

test-lswlr

Salida generada por el temporizador:

```
Ciclos en ejecucion: 13619
Ciclos totales : 55835

Estadisticas en ciclos de etapas
est_fetch.max = 59
est_fetch.min = 1
est_fetch.media = 1.669
est_dispatch.max = 67
est_dispatch.min = 1
est_dispatch.media = 2.444
est_issue.max = 4
est_issue.min = 1
est_issue.media = 1.004
est_writeback.max = 58
est_writeback.min = 1
est_writeback.media = 4.391
est_commit.max = 1
est_commit.min = 1
est_commit.media = 1.000
est_total.max = 71
est_total.min = 5
est_total.media = 10.584

Tiempo : 13568.90
Tiempo total: 55683.00

Estadisticas en nanosegundos de etapas
est_fetch_tmp.max = 59.000
est_fetch_tmp.min = 1.000
est_fetch_tmp.media = 1.669
est_dispatch_tmp.max = 67.000
est_dispatch_tmp.min = 1.000
est_dispatch_tmp.media = 2.444
est_issue_tmp.max = 4.000
est_issue_tmp.min = 1.000
est_issue_tmp.media = 0.976
est_writeback_tmp.max = 58.000
est_writeback_tmp.min = 1.000
est_writeback_tmp.media = 4.391
est_commit_tmp.max = 1.000
est_commit_tmp.min = 1.000
est_commit_tmp.media = 1.000
est_total_tmp.max = 71.000
est_total_tmp.min = 4.300
est_total_tmp.media = 10.573
```

Resultados test-lswlr

Instrucciones

Instrucciones ejecutadas	6.026
Instrucciones a commit	4.974

Ciclos

Número total de ciclos	13.618
Instrucciones por ciclo	0,3653

Media ciclos entre etapas

Fetch - Dispatch	1.6694
Dispatch - Issue	2.5974
Issue - Writeback	1.6198
Writeback - Commit	4.6064

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	1.109	18,40%
Stores	1.063	17,64%
Total:	**2.172**	**36,04%**
Saltos		
Saltos condicionales	651	10,80%
Saltos incondicionales	400	6,64%
Total:	**1.051**	**17,44%**
Instrucciones enteras		
Add	483	8,02%
Sub	193	3,20%
Mult	4	0,07%
Div	0	0,00%
Total:	**680**	**11,28%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	1	0,02%
Div	0	0,00%
Total:	**1**	**0,02%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	208	43,06%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	30	15,54%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	1	25,00%
Operando 1 (int)	1	25,00%
Operando 2 (int)	0	0,00%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	4.252	38,70%
Enteros entre 2^8 y 2^16	116	1,06%
Enteros entre 2^16 y 2^24	97	0,88%
Enteros entre 2^24 y 2^32	6.521	59,36%
PF menores de 2^8	10.986	100,00%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	0	0,00%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	137	84,05%
Entre 2^8 y 2^16	23	14,11%
Entre 2^16 y 2^24	2	1,23%
Entre 2^24 y 2^32	1	0,61%

Apéndice B: Ejecución de los Benchmarks SPEC2000

Introducción

La finalidad de ejecutar estos programas en el simulador es obtener estadísticas reales acerca de la frecuencia de los distintos tipos de instrucciones y sus operandos, para aplicarlas en el diseño de la arquitectura asíncrona.

Para optimizar el rendimiento, lo ideal es diseñar las unidades funcionales adaptadas a la carga de las operaciones que realizan, por ejemplo si se detecta que un alto porcentaje de instrucciones ADD tienen algún operando 0, tratando estas operaciones de forma independiente se podrían conseguir unos tiempos significativamente menores

Se ejecutan los siguientes programas incluidos en Spec2000 con entradas reducidas (mini-spec) en sim-outorder target alpha.

Benchmark Name	
bzip200	apsi00
crafty00	art00
eon00	equake00
gap00	facerec00
gcc00	fma3d00
gzip00	galgel00
mcf00	lucas00
parser00	mesa00
perlbmk00	mgrid00
twolf00*	sixtrack00*
vortex00	swim00
vpr00*	wupwise00
ammp00	
applu00	

*(*) **twolf00, vpr00** y **sixtrack00** no pudieron completar su ejecución debido a errores internos.*

Resultados pruebas

Resultados SPEC2000

Programa:	164.gzip	Entrada:	test	Tipo:	Int

Instrucciones

Instrucciones ejecutadas	3.686.557.555
Instrucciones a commit	3.367.274.746

Ciclos

Número total de ciclos	1.861.577.972
Instrucciones por ciclo	1,8088

Media ciclos entre etapas

Fetch - Dispatch	1.5175
Dispatch - Issue	2.8491
Issue - Writeback	1.0478
Writeback - Commit	3.3536

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	913.704.735	24,78%
Stores	264.814.490	7,18%
Total:	**1.178.519.225**	**31,97%**
Saltos		
Saltos condicionales	278.076.033	7,54%
Saltos incondicionales	102.507.416	2,78%
Total:	**380.583.449**	**10,32%**
Instrucciones enteras		
Add	395.639.318	10,73%
Sub	174.433.202	4,73%
Mult	233.315	0,01%
Div	0	0,00%
Total:	**570.305.835**	**15,47%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	4	0,00%
Div	0	0,00%
Total:	**4**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	30.233.537	7,64%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	40.227.570	23,06%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	34.482	14,78%
Operando 1 (int)	8.903	3,82%
Operando 2 (int)	2.238	0,96%
Operando 0 (pf)	1	25,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	2.028.721.096	28,67%
Enteros entre 2^8 y 2^16	121.130.952	1,71%
Enteros entre 2^16 y 2^24	16.789.667	0,24%
Enteros entre 2^24 y 2^32	4.908.479.735	69,38%
PF menores de 2^8	6.108.301.791	86,33%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	966.819.659	13,67%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	73.612.661	57,89%
Entre 2^8 y 2^16	30.994.441	24,37%
Entre 2^16 y 2^24	22.557.685	17,74%
Entre 2^24 y 2^32	453	0,00%

Resultados SPEC2000

Programa:	wupwise	Entrada:	test	Tipo:	Float

Instrucciones

Instrucciones ejecutadas	10.645.927.598
Instrucciones a commit	10.196.493.668

Ciclos

Número total de ciclos	6.003.632.648
Instrucciones por ciclo	1,6984

Media ciclos entre etapas

Fetch - Dispatch	2.0607
Dispatch - Issue	2.5758
Issue - Writeback	1.5487
Writeback - Commit	5.0477

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	2.241.442.276	21,05%
Stores	787.912.139	7,40%
Total:	**3.029.354.415**	**28,46%**
Saltos		
Saltos condicionales	805.296.100	7,56%
Saltos incondicionales	209.691.037	1,97%
Total:	**1.014.987.137**	**9,53%**
Instrucciones enteras		
Add	463.029.296	4,35%
Sub	437.980.191	4,11%
Mult	38.224.971	0,36%
Div	0	0,00%
Total:	**939.234.458**	**8,82%**
Instrucciones PF		
Add	828.417.052	7,78%
Sub	281.088.009	2,64%
Mult	1.149.900.813	10,80%
Div	13.485.110	0,13%
Total:	**2.272.890.984**	**21,35%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	13.026.703	2,81%
Operando 0 (pf)	298.776.954	36,07%
SUB		
Operando 0 (int)	48.378.875	11,05%
Operando 0 (pf)	137.981.738	49,09%
MULT		
Operando 0 (int)	6.094	0,02%
Operando 1 (int)	1.048.605	2,74%
Operando 2 (int)	18.321	0,05%
Operando 0 (pf)	467.546.005	40,66%
Operando 1 (pf)	137.086.801	11,92%
Operando 2 (pf)	9.811.371	0,85%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	6.746.938.035	32,56%
Enteros entre 2^8 y 2^16	200.247.589	0,97%
Enteros entre 2^16 y 2^24	84.733.069	0,41%
Enteros entre 2^24 y 2^32	13.686.795.359	66,06%
PF menores de 2^8	20.328.928.322	98,12%
PF entre 2^8 y 2^16	20.739.598	0,10%
PF entre 2^16 y 2^24	1.407.768	0,01%
PF entre 2^24 y 2^32	367.638.364	1,77%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	106.046.831	27,88%
Entre 2^8 y 2^16	180.922.189	47,56%
Entre 2^16 y 2^24	80.230.420	21,09%
Entre 2^24 y 2^32	13.180.760	3,47%

Resultados SPEC2000

Programa: 172.mgrid	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	115.256.118
Instrucciones a commit	114.894.795

Ciclos

Número total de ciclos	62.280.635
Instrucciones por ciclo	1,8448

Media ciclos entre etapas

Fetch - Dispatch	2.1082
Dispatch - Issue	2.1418
Issue - Writeback	1.7288
Writeback - Commit	5.0516

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	34.974.431	30,34%
Stores	6.041.892	5,24%
Total:	**41.016.323**	**35,59%**
Saltos		
Saltos condicionales	922.875	0,80%
Saltos incondicionales	38.616	0,03%
Total:	**961.491**	**0,83%**
Instrucciones enteras		
Add	3.255.442	2,82%
Sub	1.119.487	0,97%
Mult	298.680	0,26%
Div	0	0,00%
Total:	**4.673.609**	**4,05%**
Instrucciones PF		
Add	46.672.896	40,49%
Sub	4.624.384	4,01%
Mult	7.910.449	6,86%
Div	14	0,00%
Total:	**59.207.743**	**51,37%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	41.921	1,29%
Operando 0 (pf)	1.336.457	2,86%
SUB		
Operando 0 (int)	7.399	0,66%
Operando 0 (pf)	394.599	8,53%
MULT		
Operando 0 (int)	690	0,23%
Operando 1 (int)	894	0,30%
Operando 2 (int)	127	0,04%
Operando 0 (pf)	215.388	2,72%
Operando 1 (pf)	8	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	39.500.659	17,20%
Enteros entre 2^8 y 2^16	4.258.978	1,86%
Enteros entre 2^16 y 2^24	2.423.929	1,06%
Enteros entre 2^24 y 2^32	183.409.184	79,88%
PF menores de 2^8	229.542.439	99,98%
PF entre 2^8 y 2^16	309	0,00%
PF entre 2^16 y 2^24	120	0,00%
PF entre 2^24 y 2^32	49.882	0,02%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	295.036	19,41%
Entre 2^8 y 2^16	624.465	41,08%
Entre 2^16 y 2^24	600.384	39,50%
Entre 2^24 y 2^32	214	0,01%

Resultados SPEC2000

Programa: 171.swim	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	432.049.015
Instrucciones a commit	431.881.663

Ciclos

Número total de ciclos	333.044.950
Instrucciones por ciclo	1,2968

Media ciclos entre etapas

Fetch - Dispatch	3.0734
Dispatch - Issue	2.1047
Issue - Writeback	2.1219
Writeback - Commit	8.9256

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	101.660.918	23,53%
Stores	36.543.662	8,46%
Total:	**138.204.580**	**31,99%**
Saltos		
Saltos condicionales	4.525.245	1,05%
Saltos incondicionales	1.116.767	0,26%
Total:	**5.642.012**	**1,31%**
Instrucciones enteras		
Add	2.322.281	0,54%
Sub	1.103.874	0,26%
Mult	26.936	0,01%
Div	0	0,00%
Total:	**3.453.091**	**0,80%**
Instrucciones PF		
Add	68.252.281	15,80%
Sub	43.891.310	10,16%
Mult	65.080.056	15,06%
Div	2.622.508	0,61%
Total:	**179.846.155**	**41,63%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	6.328	0,27%
Operando 0 (pf)	60.263	0,09%
SUB		
Operando 0 (int)	10.629	0,96%
Operando 0 (pf)	418.866	0,95%
MULT		
Operando 0 (int)	104	0,39%
Operando 1 (int)	24	0,09%
Operando 2 (int)	20	0,07%
Operando 0 (pf)	274.071	0,42%
Operando 1 (pf)	540.276	0,83%
Operando 2 (pf)	1.050	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	105.888.406	12,41%
Enteros entre 2^8 y 2^16	2.719.209	0,32%
Enteros entre 2^16 y 2^24	1.737.025	0,20%
Enteros entre 2^24 y 2^32	743.013.272	87,07%
PF menores de 2^8	620.233.664	72,68%
PF entre 2^8 y 2^16	2.041.771	0,88%
PF entre 2^16 y 2^24	370.914	0,04%
PF entre 2^24 y 2^32	230.711.563	27,04%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	21.616	24,95%
Entre 2^8 y 2^16	33.558	38,73%
Entre 2^16 y 2^24	8.093	9,34%
Entre 2^24 y 2^32	23.379	26,98%

Resultados SPEC2000

Programa: 173.applu	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	183.732.740
Instrucciones a commit	182.849.459

Ciclos

Número total de ciclos	138.306.631
Instrucciones por ciclo	1,3221

Media ciclos entre etapas

Fetch - Dispatch	2.9310
Dispatch - Issue	3.4732
Issue - Writeback	2.3276
Writeback - Commit	6.2758

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	48.641.531	26,47%
Stores	19.398.445	10,56%
Total:	**68.039.976**	**37,03%**
Saltos		
Saltos condicionales	2.630.141	1,43%
Saltos incondicionales	146.157	0,08%
Total:	**2.776.298**	**1,51%**
Instrucciones enteras		
Add	1.831.972	1,00%
Sub	209.257	0,11%
Mult	103.376	0,06%
Div	0	0,00%
Total:	**2.144.605**	**1,17%**
Instrucciones PF		
Add	16.323.469	8,88%
Sub	18.146.142	9,88%
Mult	52.051.185	28,33%
Div	1.388.136	0,76%
Total:	**87.908.932**	**47,85%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	250.298	13,66%
Operando 0 (pf)	1.951.483	11,96%
SUB		
Operando 0 (int)	19.376	9,26%
Operando 0 (pf)	3.806.801	20,98%
MULT		
Operando 0 (int)	1.188	1,15%
Operando 1 (int)	13	0,01%
Operando 2 (int)	2.460	2,38%
Operando 0 (pf)	7.903.620	15,18%
Operando 1 (pf)	783.076	1,50%
Operando 2 (pf)	512	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	148	0,01%
Operando 2 (pf)	148	0,01%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	64.500.967	17,75%
Enteros entre 2^8 y 2^16	34.014.524	9,36%
Enteros entre 2^16 y 2^24	95.619	0,03%
Enteros entre 2^24 y 2^32	264.785.256	72,86%
PF menores de 2^8	314.721.523	86,61%
PF entre 2^8 y 2^16	11.342.509	3,12%
PF entre 2^16 y 2^24	3.681.863	1,01%
PF entre 2^24 y 2^32	33.650.471	9,26%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	290.627	27,68%
Entre 2^8 y 2^16	336.738	32,07%
Entre 2^16 y 2^24	412.301	39,27%
Entre 2^24 y 2^32	10.213	0,97%

Resultados SPEC2000

Programa: 176.gcc	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	2.297.654.706
Instrucciones a commit	2.016.190.465

Ciclos

Número total de ciclos	1.551.186.444
Instrucciones por ciclo	1,2998

Media ciclos entre etapas

Fetch - Dispatch	1.5154
Dispatch - Issue	2.5347
Issue - Writeback	1.0588
Writeback - Commit	3.1736

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	785.470.741	34,19%
Stores	233.597.540	10,17%
Total:	**1.019.068.281**	**44,35%**
Saltos		
Saltos condicionales	279.655.029	12,17%
Saltos incondicionales	79.441.256	3,46%
Total:	**359.096.285**	**15,63%**
Instrucciones enteras		
Add	158.529.967	6,90%
Sub	34.287.941	1,49%
Mult	1.901.152	0,08%
Div	0	0,00%
Total:	**194.719.060**	**8,47%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	11.570	0,00%
Div	36.131	0,00%
Total:	**47.701**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	29.105.665	18,36%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	5.052.342	14,74%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	285.586	15,02%
Operando 1 (int)	837.389	44,05%
Operando 2 (int)	49.464	2,60%
Operando 0 (pf)	177	1,53%
Operando 1 (pf)	631	5,45%
Operando 2 (pf)	319	2,76%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	869	2,41%
Operando 2 (pf)	1.460	4,04%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	1.481.303.637	34,26%
Enteros entre 2^8 y 2^16	70.899.321	1,64%
Enteros entre 2^16 y 2^24	33.738.218	0,78%
Enteros entre 2^24 y 2^32	2.738.203.162	63,32%
PF menores de 2^8	3.894.634.454	90,07%
PF entre 2^8 y 2^16	28.010.025	0,65%
PF entre 2^16 y 2^24	9.955.044	0,23%
PF entre 2^24 y 2^32	391.544.815	9,05%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	35.189.676	71,39%
Entre 2^8 y 2^16	10.765.100	21,84%
Entre 2^16 y 2^24	1.695.140	3,44%
Entre 2^24 y 2^32	1.638.952	3,33%

Resultados SPEC 2000

Programa: 177.mesa	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	1.907.063.374
Instrucciones a commit	1.743.972.649

Ciclos

Número total de ciclos	824.326.905
Instrucciones por ciclo	2,1156

Media ciclos entre etapas

Fetch - Dispatch	1.3813
Dispatch - Issue	2.3795
Issue - Writeback	1.0278
Writeback - Commit	3.0101

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	404.052.276	21,19%
Stores	193.609.122	10,15%
Total:	**597.661.398**	**31,34%**
Saltos		
Saltos condicionales	222.672.176	11,68%
Saltos incondicionales	89.940.514	4,72%
Total:	**312.612.690**	**16,39%**
Instrucciones enteras		
Add	136.883.933	7,18%
Sub	87.801.714	4,60%
Mult	14.440.016	0,76%
Div	0	0,00%
Total:	**239.125.663**	**12,54%**
Instrucciones PF		
Add	2.236.599	0,12%
Sub	1.831.665	0,10%
Mult	2.257.210	0,12%
Div	358.993	0,02%
Total:	**6.684.467**	**0,35%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	44.697.264	32,65%
Operando 0 (pf)	1.672.604	74,78%
SUB		
Operando 0 (int)	19.020.514	21,66%
Operando 0 (pf)	17.046	0,93%
MULT		
Operando 0 (int)	5.731.415	39,69%
Operando 1 (int)	1.399.733	9,69%
Operando 2 (int)	916	0,01%
Operando 0 (pf)	1.477.520	65,46%
Operando 1 (pf)	367.194	16,27%
Operando 2 (pf)	15	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	4	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	1.565.508.895	43,71%
Enteros entre 2^8 y 2^16	20.719.616	0,58%
Enteros entre 2^16 y 2^24	24.198.353	0,68%
Enteros entre 2^24 y 2^32	1.971.448.046	55,04%
PF menores de 2^8	3.374.304.576	94,20%
PF entre 2^8 y 2^16	6.646	0,00%
PF entre 2^16 y 2^24	321	0,00%
PF entre 2^24 y 2^32	207.563.367	5,79%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	26.089.247	93,89%
Entre 2^8 y 2^16	1.689.552	6,08%
Entre 2^16 y 2^24	8.586	0,03%
Entre 2^24 y 2^32	53	0,00%

Resultados SPEC2000

Programa: 178.galgel	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	355.914.731
Instrucciones a commit	347.588.697

Ciclos

Número total de ciclos	185.340.523
Instrucciones por ciclo	1,8754

Media ciclos entre etapas

Fetch - Dispatch	1.8642
Dispatch - Issue	2.4619
Issue - Writeback	1.4548
Writeback - Commit	4.2369

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	121.556.508	34,15%
Stores	24.813.132	6,97%
Total:	**146.369.640**	**41,12%**
Saltos		
Saltos condicionales	22.259.535	6,25%
Saltos incondicionales	674.909	0,19%
Total:	**22.934.444**	**6,44%**
Instrucciones enteras		
Add	9.150.084	2,57%
Sub	881.798	0,25%
Mult	143.933	0,04%
Div	0	0,00%
Total:	**10.175.815**	**2,86%**
Instrucciones PF		
Add	33.707.952	9,47%
Sub	4.378.980	1,23%
Mult	42.435.430	11,92%
Div	141.733	0,04%
Total:	**80.664.095**	**22,66%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	227.169	2,48%
Operando 0 (pf)	10.262.270	30,44%
SUB		
Operando 0 (int)	297.001	33,68%
Operando 0 (pf)	280.109	6,40%
MULT		
Operando 0 (int)	5.838	4,06%
Operando 1 (int)	22.177	15,41%
Operando 2 (int)	10.460	7,27%
Operando 0 (pf)	15.118.819	35,63%
Operando 1 (pf)	521.242	1,23%
Operando 2 (pf)	52.443	0,12%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	6.701	4,73%
Operando 2 (pf)	4.330	3,06%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	176.438.037	25,40%
Enteros entre 2^8 y 2^16	15.070.150	2,17%
Enteros entre 2^16 y 2^24	14.962.324	2,15%
Enteros entre 2^24 y 2^32	488.089.183	70,27%
PF menores de 2^8	607.226.473	87,43%
PF entre 2^8 y 2^16	7.019.316	1,01%
PF entre 2^16 y 2^24	4.219.789	0,61%
PF entre 2^24 y 2^32	76.094.116	10,96%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	1.355.399	44,52%
Entre 2^8 y 2^16	1.187.137	39,00%
Entre 2^16 y 2^24	499.405	16,40%
Entre 2^24 y 2^32	2.309	0,08%

Resultados SPEC2000		
Programa: 179.art	Entrada: Test	Tipo: Float

Instrucciones	
Instrucciones ejecutadas	1.531.082.784
Instrucciones a commit	1.478.590.373

Ciclos	
Número total de ciclos	2.749.593.313
Instrucciones por ciclo	0,5377

Media ciclos entre etapas	
Fetch - Dispatch	6.7132
Dispatch - Issue	4.5514
Issue - Writeback	4.1381
Writeback - Commit	18.8169

Desglose de instrucciones		
TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	474.897.899	31,02%
Stores	129.059.326	8,43%
Total:	**603.957.225**	**39,45%**
Saltos		
Saltos condicionales	124.120.202	8,11%
Saltos incondicionales	12.328.649	0,81%
Total:	**136.448.851**	**8,91%**
Instrucciones enteras		
Add	129.772.547	8,48%
Sub	34.118	0,00%
Mult	307.227	0,02%
Div	0	0,00%
Total:	**130.113.892**	**8,50%**
Instrucciones PF		
Add	120.599.596	7,88%
Sub	10.200.126	0,67%
Mult	139.810.848	9,13%
Div	42.662	0,00%
Total:	**270.653.232**	**17,68%**

Operandos en operaciones aritméticas		
Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)		
ADD		
Operando 0 (int)	48.828.600	37,63%
Operando 0 (pf)	62.915.890	52,17%
SUB		
Operando 0 (int)	15.815	46,35%
Operando 0 (pf)	7.201.462	70,60%
MULT		
Operando 0 (int)	659	0,21%
Operando 1 (int)	641	0,21%
Operando 2 (int)	640	0,21%
Operando 0 (pf)	72.815.523	52,08%
Operando 1 (pf)	19	0,00%
Operando 2 (pf)	1.479.269	1,06%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros		
TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	1.046.843.712	34,56%
Enteros entre 2^8 y 2^16	3.463.611	0,11%
Enteros entre 2^16 y 2^24	137.690	0,00%
Enteros entre 2^24 y 2^32	1.978.747.645	65,32%
PF menores de 2^8	2.580.957.393	85,20%
PF entre 2^8 y 2^16	63.400.781	2,09%
PF entre 2^16 y 2^24	33.422.552	1,10%
PF entre 2^24 y 2^32	351.411.932	11,60%

Acarreos en enteros (posición donde se rompe)		
TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	43.574.448	73,19%
Entre 2^8 y 2^16	6.702.084	11,26%
Entre 2^16 y 2^24	9.262.449	15,56%
Entre 2^24 y 2^32	0	0,00%

Resultados SPEC2000		
Programa: 181.mcf	Entrada: Test	Tipo: Float

Instrucciones	
Instrucciones ejecutadas	297.811.190
Instrucciones a commit	259.643.405

Ciclos	
Número total de ciclos	242.952.160
Instrucciones por ciclo	1,0687

Media ciclos entre etapas	
Fetch - Dispatch	2.7669
Dispatch - Issue	4.3026
Issue - Writeback	1.8363
Writeback - Commit	5.1144

Desglose de instrucciones		
TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	105.623.191	35,47%
Stores	36.083.944	12,12%
Total:	**141.707.135**	**47,58%**
Saltos		
Saltos condicionales	36.778.961	12,35%
Saltos incondicionales	15.079.434	5,06%
Total:	**51.858.395**	**17,41%**
Instrucciones enteras		
Add	19.473.252	6,54%
Sub	7.437.813	2,50%
Mult	61.431	0,02%
Div	0	0,00%
Total:	**26.972.496**	**9,06%**
Instrucciones PF		
Add	2	0,00%
Sub	0	0,00%
Mult	7	0,00%
Div	0	0,00%
Total:	**9**	**0,00%**

Operandos en operaciones aritméticas		
Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)		
ADD		
Operando 0 (int)	1.951.022	10,02%
Operando 0 (pf)	0	0,00%
SUB		
Operando 0 (int)	2.357.050	31,69%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	9.948	16,19%
Operando 1 (int)	12.601	20,51%
Operando 2 (int)	1.744	2,84%
Operando 0 (pf)	2	28,57%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros		
TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	154.384.897	27,79%
Enteros entre 2^8 y 2^16	6.594.194	1,19%
Enteros entre 2^16 y 2^24	5.722.946	1,03%
Enteros entre 2^24 y 2^32	388.832.281	69,99%
PF menores de 2^8	463.066.289	83,36%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	92.468.029	16,64%

Acarreos en enteros (posición donde se rompe)		
TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	2.327.829	53,09%
Entre 2^8 y 2^16	1.839.603	41,96%
Entre 2^16 y 2^24	98.550	2,25%
Entre 2^24 y 2^32	118.327	2,70%

Resultados SPEC2000

Programa: 183.equake	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	1.515.237.750
Instrucciones a commit	1.443.348.058

Ciclos

Número total de ciclos	1.007.150.481
Instrucciones por ciclo	1,4331

Media ciclos entre etapas

Fetch - Dispatch	2.1889
Dispatch - Issue	2.9698
Issue - Writeback	1.5176
Writeback - Commit	4.9905

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	459.364.335	30,32%
Stores	114.956.381	7,59%
Total:	**574.320.716**	**37,90%**
Saltos		
Saltos condicionales	113.657.539	7,50%
Saltos incondicionales	59.469.752	3,92%
Total:	**173.127.291**	**11,43%**
Instrucciones enteras		
Add	83.659.840	5,52%
Sub	15.132.118	1,00%
Mult	2.081.501	0,14%
Div	0	0,00%
Total:	**100.873.459**	**6,66%**
Instrucciones PF		
Add	60.193.020	3,97%
Sub	26.637.775	1,76%
Mult	102.141.591	6,74%
Div	4.709.103	0,31%
Total:	**193.681.489**	**12,78%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	19.702.697	23,55%
Operando 0 (pf)	20.531.561	34,11%
SUB		
Operando 0 (int)	423.899	2,80%
Operando 0 (pf)	2.814.199	10,56%
MULT		
Operando 0 (int)	286.938	13,79%
Operando 1 (int)	336.258	16,15%
Operando 2 (int)	51.660	2,48%
Operando 0 (pf)	25.494.112	24,96%
Operando 1 (pf)	4.622.334	4,53%
Operando 2 (pf)	47.152	0,05%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	870.882.200	29,49%
Enteros entre 2^8 y 2^16	42.273.688	1,43%
Enteros entre 2^16 y 2^24	12.001.355	0,41%
Enteros entre 2^24 y 2^32	2.027.922.875	68,67%
PF menores de 2^8	2.693.154.939	91,20%
PF entre 2^8 y 2^16	32.241.236	1,09%
PF entre 2^16 y 2^24	11.761.256	0,40%
PF entre 2^24 y 2^32	215.922.687	7,31%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	15.687.917	59,47%
Entre 2^8 y 2^16	6.695.958	25,39%
Entre 2^16 y 2^24	3.991.541	15,13%
Entre 2^24 y 2^32	2.069	0,01%

Resultados SPEC2000

Programa: 186.crafty	Entrada: test	Tipo: Int

Instrucciones

Instrucciones ejecutadas	4.764.640.709
Instrucciones a commit	4.264.781.553

Ciclos

Número total de ciclos	3.271.492.757
Instrucciones por ciclo	1,3036

Media ciclos entre etapas

Fetch - Dispatch	1.5660
Dispatch - Issue	2.5464
Issue - Writeback	1.1254
Writeback - Commit	3.7819

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	1.482.713.381	31,12%
Stores	229.390.540	4,81%
Total:	**1.712.103.921**	**35,93%**
Saltos		
Saltos condicionales	435.662.390	9,14%
Saltos incondicionales	133.985.900	2,81%
Total:	**569.648.290**	**11,96%**
Instrucciones enteras		
Add	472.874.836	9,92%
Sub	180.602.125	3,79%
Mult	17.157.039	0,36%
Div	0	0,00%
Total:	**670.634.000**	**14,08%**
Instrucciones PF		
Add	4	0,00%
Sub	0	0,00%
Mult	109	0,00%
Div	24	0,00%
Total:	**137**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB...)

ADD		
Operando 0 (int)	68.004.221	14,38%
Operando 0 (pf)	0	0,00%
SUB		
Operando 0 (int)	76.986.068	42,63%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	2.557.621	14,91%
Operando 1 (int)	2.272.291	13,24%
Operando 2 (int)	1.806.354	10,53%
Operando 0 (pf)	21	19,27%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	6	5,50%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	2.489.151.359	27,70%
Enteros entre 2^8 y 2^16	206.520.727	2,30%
Enteros entre 2^16 y 2^24	126.321.724	1,41%
Enteros entre 2^24 y 2^32	6.164.105.880	68,60%
PF menores de 2^8	8.581.207.519	95,49%
PF entre 2^8 y 2^16	217	0,00%
PF entre 2^16 y 2^24	87	0,00%
PF entre 2^24 y 2^32	404.891.867	4,51%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	102.758.060	76,56%
Entre 2^8 y 2^16	18.299.700	13,63%
Entre 2^16 y 2^24	12.229.435	9,11%
Entre 2^24 y 2^32	926.643	0,69%

Resultados SPEC2000

Programa: 187.facerec	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	267.253.668
Instrucciones a commit	252.292.548

Ciclos

Número total de ciclos	132.093.072
Instrucciones por ciclo	1,9100

Media ciclos entre etapas

Fetch - Dispatch	1.7533
Dispatch - Issue	2.4039
Issue - Writeback	1.3851
Writeback - Commit	3.9581

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	60.276.816	22,55%
Stores	28.197.519	10,55%
Total:	**88.474.335**	**33,11%**
Saltos		
Saltos condicionales	25.338.162	9,48%
Saltos incondicionales	7.329.754	2,74%
Total:	**32.667.916**	**12,22%**
Instrucciones enteras		
Add	23.374.839	8,75%
Sub	5.850.891	2,19%
Mult	527.321	0,20%
Div	0	0,00%
Total:	**29.753.051**	**11,13%**
Instrucciones PF		
Add	9.334.615	3,49%
Sub	8.619.001	3,23%
Mult	10.423.724	3,90%
Div	209.682	0,08%
Total:	**28.587.022**	**10,70%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	2.160.021	9,24%
Operando 0 (pf)	718.247	7,69%
SUB		
Operando 0 (int)	1.632.285	27,90%
Operando 0 (pf)	583.257	6,77%
MULT		
Operando 0 (int)	72.998	13,84%
Operando 1 (int)	11.081	2,10%
Operando 2 (int)	8.865	1,68%
Operando 0 (pf)	598.882	5,75%
Operando 1 (pf)	550.271	5,28%
Operando 2 (pf)	3.759	0,04%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	9	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	176.646.799	34,38%
Enteros entre 2^8 y 2^16	6.846.107	1,33%
Enteros entre 2^16 y 2^24	3.279.591	0,64%
Enteros entre 2^24 y 2^32	327.024.945	63,65%
PF menores de 2^8	375.498.664	73,08%
PF entre 2^8 y 2^16	2.468.507	0,48%
PF entre 2^16 y 2^24	1.930.772	0,38%
PF entre 2^24 y 2^32	133.899.499	26,06%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	7.317.774	82,54%
Entre 2^8 y 2^16	1.393.159	15,71%
Entre 2^16 y 2^24	129.242	1,46%
Entre 2^24 y 2^32	25.465	0,29%

Resultados SPEC2000

Programa: 188.ammp	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	5.630.623.025
Instrucciones a commit	5.491.072.422

Ciclos

Número total de ciclos	7.532.522.073
Instrucciones por ciclo	0,7290

Media ciclos entre etapas

Fetch - Dispatch	5.1637
Dispatch - Issue	9.4489
Issue - Writeback	2.8034
Writeback - Commit	8.7820

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	1.637.259.058	29,08%
Stores	392.955.341	6,98%
Total:	**2.030.214.399**	**36,06%**
Saltos		
Saltos condicionales	620.880.165	11,03%
Saltos incondicionales	42.100.865	0,75%
Total:	**662.981.030**	**11,77%**
Instrucciones enteras		
Add	278.268.694	4,94%
Sub	38.842.424	0,69%
Mult	804.559	0,01%
Div	0	0,00%
Total:	**317.915.677**	**5,65%**
Instrucciones PF		
Add	330.717.460	5,87%
Sub	234.115.426	4,16%
Mult	722.317.524	12,83%
Div	13.761.854	0,24%
Total:	**1.300.912.264**	**23,10%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	11.575.743	4,16%
Operando 0 (pf)	3.728.421	1,13%
SUB		
Operando 0 (int)	12.052.203	31,03%
Operando 0 (pf)	3.480.332	1,49%
MULT		
Operando 0 (int)	376	0,05%
Operando 1 (int)	66	0,01%
Operando 2 (int)	35	0,00%
Operando 0 (pf)	8.608.920	1,19%
Operando 1 (pf)	357.249	0,05%
Operando 2 (pf)	213.432	0,03%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	39.909	0,29%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	2.022.644.957	18,26%
Enteros entre 2^8 y 2^16	157.064.353	1,42%
Enteros entre 2^16 y 2^24	110.738.197	1,00%
Enteros entre 2^24 y 2^32	8.785.295.987	79,32%
PF menores de 2^8	9.719.166.181	87,75%
PF entre 2^8 y 2^16	202.096.292	1,82%
PF entre 2^16 y 2^24	106.098.161	0,96%
PF entre 2^24 y 2^32	1.048.382.860	9,47%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	63.464.261	81,31%
Entre 2^8 y 2^16	12.466.904	15,97%
Entre 2^16 y 2^24	1.174.720	1,51%
Entre 2^24 y 2^32	943.270	1,21%

Resultados SPEC2000

Programa: 189.lucas	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	3.716.238.871
Instrucciones a commit	3.712.193.264

Ciclos

Número total de ciclos	2.170.467.874
Instrucciones por ciclo	1,7103

Media ciclos entre etapas

Fetch - Dispatch	2.3200
Dispatch - Issue	3.4408
Issue - Writeback	1.6956
Writeback - Commit	4.7998

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	675.386.986	18,17%
Stores	185.091.112	4,98%
Total:	**860.478.098**	**23,15%**
Saltos		
Saltos condicionales	60.515.441	1,63%
Saltos incondicionales	284.812	0,01%
Total:	**60.800.253**	**1,64%**
Instrucciones enteras		
Add	388.902.833	10,46%
Sub	59.658.236	1,61%
Mult	60.022.192	1,62%
Div	0	0,00%
Total:	**508.583.261**	**13,69%**
Instrucciones PF		
Add	433.910.999	11,68%
Sub	372.542.282	10,02%
Mult	444.785.665	11,97%
Div	19.945	0,00%
Total:	**1.251.258.891**	**33,67%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	56.780.508	14,60%
Operando 0 (pf)	2.155.682	0,50%
SUB		
Operando 0 (int)	1.081.114	1,81%
Operando 0 (pf)	5.460.332	1,47%
MULT		
Operando 0 (int)	7.075.312	11,79%
Operando 1 (int)	2.631.391	4,38%
Operando 2 (int)	2.631.293	4,38%
Operando 0 (pf)	12.937.177	2,91%
Operando 1 (pf)	8.014.953	1,80%
Operando 2 (pf)	41.195	0,01%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	1.271.206.429	17,12%
Enteros entre 2^8 y 2^16	130.066.810	1,75%
Enteros entre 2^16 y 2^24	56.688.132	0,76%
Enteros entre 2^24 y 2^32	5.967.950.685	80,37%
PF menores de 2^8	4.924.013.025	66,31%
PF entre 2^8 y 2^16	19.061.367	0,26%
PF entre 2^16 y 2^24	3.420.861	0,05%
PF entre 2^24 y 2^32	2.479.416.803	33,39%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	68.460.061	48,27%
Entre 2^8 y 2^16	73.361.862	51,73%
Entre 2^16 y 2^24	14	0,00%
Entre 2^24 y 2^32	6	0,00%

Resultados SPEC2000

Programa: 191.fma3d	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	728.262.477
Instrucciones a commit	668.503.621

Ciclos

Número total de ciclos	543.902.551
Instrucciones por ciclo	1,2291

Media ciclos entre etapas

Fetch - Dispatch	1.2777
Dispatch - Issue	2.3357
Issue - Writeback	1.0230
Writeback - Commit	2.6842

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	175.383.795	24,08%
Stores	99.218.638	13,62%
Total:	**274.602.433**	**37,71%**
Saltos		
Saltos condicionales	76.821.778	10,55%
Saltos incondicionales	24.298.559	3,34%
Total:	**101.120.337**	**13,89%**
Instrucciones enteras		
Add	48.053.476	6,60%
Sub	25.391.336	3,49%
Mult	1.007.398	0,14%
Div	0	0,00%
Total:	**74.452.210**	**10,22%**
Instrucciones PF		
Add	1.273.384	0,17%
Sub	388.948	0,05%
Mult	2.115.147	0,29%
Div	106.082	0,01%
Total:	**3.883.561**	**0,53%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	11.270.292	23,45%
Operando 0 (pf)	601.108	47,21%
SUB		
Operando 0 (int)	5.990.423	23,59%
Operando 0 (pf)	320.271	82,34%
MULT		
Operando 0 (int)	445.430	44,22%
Operando 1 (int)	37.982	3,77%
Operando 2 (int)	35.522	3,53%
Operando 0 (pf)	2.115.147	100,00%
Operando 1 (pf)	1.098.160	51,92%
Operando 2 (pf)	48.305	2,28%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	15.118	14,25%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	527.566.726	37,87%
Enteros entre 2^8 y 2^16	42.621.080	3,06%
Enteros entre 2^16 y 2^24	5.105.211	0,37%
Enteros entre 2^24 y 2^32	817.866.611	58,71%
PF menores de 2^8	1.117.905.790	80,24%
PF entre 2^8 y 2^16	29.688.564	2,13%
PF entre 2^16 y 2^24	750.834	0,05%
PF entre 2^24 y 2^32	244.814.440	17,57%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	10.167.976	82,26%
Entre 2^8 y 2^16	599.643	4,85%
Entre 2^16 y 2^24	1.256.744	10,17%
Entre 2^24 y 2^32	336.254	2,72%

Resultados SPEC2000

Programa: 200.sixtrack	Entrada: test	Tipo: Float

Instrucciones

Instrucciones ejecutadas	13.198.496
Instrucciones a commit	12.045.468

Ciclos

Número total de ciclos	5.378.633
Instrucciones por ciclo	2,2395

Media ciclos entre etapas

Fetch - Dispatch	1.2173
Dispatch - Issue	2.3513
Issue - Writeback	1.0295
Writeback - Commit	2.2416

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	562.504	4,26%
Stores	2.818.183	21,35%
Total:	**3.380.687**	**25,61%**
Saltos		
Saltos condicionales	1.268.761	9,61%
Saltos incondicionales	30.928	0,23%
Total:	**1.299.689**	**9,85%**
Instrucciones enteras		
Add	35.107	0,27%
Sub	13.220	0,10%
Mult	19	0,00%
Div	3	0,00%
Total:	**48.349**	**0,37%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	0	0,00%
Div	0	0,00%
Total:	**0**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	2.607	7,43%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	423	3,20%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	4	21,05%
Operando 1 (int)	0	0,00%
Operando 2 (int)	0	0,00%
Operando 0 (pf)	1	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!
DIV		
Operando 1 (int)	0	0,00%
Operando 2 (int)	0	0,00%
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	7.188.323	31,75%
Enteros entre 2^8 y 2^16	97.040	0,43%
Enteros entre 2^16 y 2^24	57.895	0,26%
Enteros entre 2^24 y 2^32	15.299.626	67,57%
PF menores de 2^8	20.080.680	88,68%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	2.562.204	11,32%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	8.518	70,31%
Entre 2^8 y 2^16	2.436	20,11%
Entre 2^16 y 2^24	225	1,86%
Entre 2^24 y 2^32	936	7,73%

Resultados SPEC2000

Programa: 252.eon	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	104.512.521
Instrucciones a commit	94.039.228

Ciclos

Número total de ciclos	65.320.448
Instrucciones por ciclo	1,4397

Media ciclos entre etapas

Fetch - Dispatch	1.6452
Dispatch - Issue	2.4637
Issue - Writeback	1.2249
Writeback - Commit	3.4995

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	31.180.542	29,83%
Stores	18.300.193	17,51%
Total:	**49.480.735**	**47,34%**
Saltos		
Saltos condicionales	6.564.255	6,28%
Saltos incondicionales	5.036.561	4,82%
Total:	**11.600.816**	**11,10%**
Instrucciones enteras		
Add	3.159.875	3,02%
Sub	1.106.395	1,06%
Mult	486.259	0,47%
Div	0	0,00%
Total:	**4.752.529**	**4,55%**
Instrucciones PF		
Add	2.828.532	2,71%
Sub	889.240	0,85%
Mult	4.000.183	3,83%
Div	361.379	0,35%
Total:	**8.079.334**	**7,73%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	273.904	8,67%
Operando 0 (pf)	889.868	31,46%
SUB		
Operando 0 (int)	308.364	27,87%
Operando 0 (pf)	141.226	15,88%
MULT		
Operando 0 (int)	4.923	1,01%
Operando 1 (int)	164	0,03%
Operando 2 (int)	135	0,03%
Operando 0 (pf)	1.286.196	32,15%
Operando 1 (pf)	105.176	2,63%
Operando 2 (pf)	47.271	1,18%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	15.048	4,16%
Operando 2 (pf)	15.302	4,23%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	48.505.311	24,70%
Enteros entre 2^8 y 2^16	836.286	0,43%
Enteros entre 2^16 y 2^24	641.609	0,33%
Enteros entre 2^24 y 2^32	146.424.954	74,55%
PF menores de 2^8	192.288.302	97,90%
PF entre 2^8 y 2^16	613.556	0,31%
PF entre 2^16 y 2^24	216.591	0,11%
PF entre 2^24 y 2^32	3.289.711	1,67%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	1.057.427	79,34%
Entre 2^8 y 2^16	232.330	17,43%
Entre 2^16 y 2^24	2.310	0,17%
Entre 2^24 y 2^32	40.670	3,05%

Resultados SPEC2000

Programa: perlbmk	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	5.923.980
Instrucciones a commit	5.250.203

Ciclos

Número total de ciclos	4.515.317
Instrucciones por ciclo	1,1628

Media ciclos entre etapas

Fetch - Dispatch	1.3720
Dispatch - Issue	2.4768
Issue - Writeback	1.0630
Writeback - Commit	3.0558

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	1.676.463	28,30%
Stores	735.926	12,42%
Total:	**2.412.389**	**40,72%**
Saltos		
Saltos condicionales	673.570	11,37%
Saltos incondicionales	273.125	4,61%
Total:	**946.695**	**15,98%**
Instrucciones enteras		
Add	336.166	5,67%
Sub	130.779	2,21%
Mult	152	0,00%
Div	0	0,00%
Total:	**467.097**	**7,88%**
Instrucciones PF		
Add	1	0,00%
Sub	0	0,00%
Mult	7	0,00%
Div	0	0,00%
Total:	**8**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	61.746	18,37%
Operando 0 (pf)	0	0,00%
SUB		
Operando 0 (int)	21.178	16,19%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	29	19,08%
Operando 1 (int)	69	45,39%
Operando 2 (int)	8	5,26%
Operando 0 (pf)	1	14,29%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	3.239.913	28,94%
Enteros entre 2^8 y 2^16	185.194	1,65%
Enteros entre 2^16 y 2^24	51.424	0,46%
Enteros entre 2^24 y 2^32	7.720.663	68,95%
PF menores de 2^8	10.714.726	95,69%
PF entre 2^8 y 2^16	4.878	0,04%
PF entre 2^16 y 2^24	5.181	0,05%
PF entre 2^24 y 2^32	472.409	4,22%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	80.708	55,80%
Entre 2^8 y 2^16	58.774	40,63%
Entre 2^16 y 2^24	1.241	0,86%
Entre 2^24 y 2^32	3.923	2,71%

Resultados SPEC2000

Programa: 254.gap	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	1.374.270.958
Instrucciones a commit	1.223.888.818

Ciclos

Número total de ciclos	879.662.812
Instrucciones por ciclo	1,3913

Media ciclos entre etapas

Fetch - Dispatch	1.3849
Dispatch - Issue	2.6204
Issue - Writeback	1.0090
Writeback - Commit	2.9811

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	420.993.507	30,63%
Stores	129.921.653	9,45%
Total:	**550.915.160**	**40,09%**
Saltos		
Saltos condicionales	147.615.930	10,74%
Saltos incondicionales	54.684.156	3,98%
Total:	**202.300.086**	**14,72%**
Instrucciones enteras		
Add	58.461.053	4,25%
Sub	12.090.614	0,88%
Mult	691.675	0,05%
Div	0	0,00%
Total:	**71.243.342**	**5,18%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	100.542	0,01%
Div	0	0,00%
Total:	**100.542**	**0,01%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	12.937.299	22,13%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	26.288.620	217,43%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	68.392	9,89%
Operando 1 (int)	75.755	10,95%
Operando 2 (int)	34.092	4,93%
Operando 0 (pf)	1	0,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	16.047	#¡DIV/0!
Operando 2 (pf)	38.647	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	644.528.287	24,87%
Enteros entre 2^8 y 2^16	40.875.231	1,58%
Enteros entre 2^16 y 2^24	18.140.629	0,70%
Enteros entre 2^24 y 2^32	1.888.096.809	72,85%
PF menores de 2^8	2.323.044.722	89,64%
PF entre 2^8 y 2^16	2.023.483	0,08%
PF entre 2^16 y 2^24	291.764	0,01%
PF entre 2^24 y 2^32	266.280.987	10,27%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	8.688.298	54,17%
Entre 2^8 y 2^16	4.871.058	30,37%
Entre 2^16 y 2^24	1.329.815	8,29%
Entre 2^24 y 2^32	1.149.319	7,17%

Resultados SPEC2000

Programa: 255.vortex	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	481.609
Instrucciones a commit	454.650

Ciclos

Número total de ciclos	283.478
Instrucciones por ciclo	1,6038

Media ciclos entre etapas

Fetch - Dispatch	1.4554
Dispatch - Issue	2.1875
Issue - Writeback	1.0396
Writeback - Commit	2.9712

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	140.669	29,21%
Stores	63.116	13,11%
Total:	**203.785**	**42,31%**
Saltos		
Saltos condicionales	53.321	11,07%
Saltos incondicionales	19.286	4,00%
Total:	**72.607**	**15,08%**
Instrucciones enteras		
Add	25.065	5,20%
Sub	16.350	3,39%
Mult	318	0,07%
Div	0	0,00%
Total:	**41.733**	**8,67%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	5	0,00%
Div	10	0,00%
Total:	**15**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	5.615	22,40%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	2.028	12,40%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	155	48,74%
Operando 1 (int)	49	15,41%
Operando 2 (int)	6	1,89%
Operando 0 (pf)	1	20,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	10	100,00%
Operando 2 (pf)	0	0,00%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	280.552	30,40%
Enteros entre 2^8 y 2^16	13.673	1,48%
Enteros entre 2^16 y 2^24	4.820	0,52%
Enteros entre 2^24 y 2^32	623.707	67,59%
PF menores de 2^8	906.364	98,22%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	16.388	1,78%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	7.133	80,81%
Entre 2^8 y 2^16	1.653	18,73%
Entre 2^16 y 2^24	33	0,37%
Entre 2^24 y 2^32	8	0,09%

Resultados SPEC2000

Programa: 256.bzip	Entrada: test	Tipo: int

Instrucciones

Instrucciones ejecutadas	9.072.729.625
Instrucciones a commit	8.822.143.316

Ciclos

Número total de ciclos	4.215.601.782
Instrucciones por ciclo	2,0927

Media ciclos entre etapas

Fetch - Dispatch	1.7087
Dispatch - Issue	2.8122
Issue - Writeback	1.0685
Writeback - Commit	3.3298

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	2.348.735.081	25,89%
Stores	1.479.442.630	16,31%
Total:	**3.828.177.711**	**42,19%**
Saltos		
Saltos condicionales	922.411.317	10,17%
Saltos incondicionales	109.285.758	1,20%
Total:	**1.031.697.075**	**11,37%**
Instrucciones enteras		
Add	1.232.689.159	13,59%
Sub	128.594.866	1,42%
Mult	87	0,00%
Div	0	0,00%
Total:	**1.361.284.112**	**15,00%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	11	0,00%
Div	36	0,00%
Total:	**47**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	91.242.188	7,40%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	20.107.956	15,64%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	11	12,64%
Operando 1 (int)	28	32,18%
Operando 2 (int)	0	0,00%
Operando 0 (pf)	2	18,18%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	6	16,67%
Operando 2 (pf)	6	16,67%

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	3.368.574.123	18,91%
Enteros entre 2^8 y 2^16	277.837.653	1,56%
Enteros entre 2^16 y 2^24	223.235.920	1,25%
Enteros entre 2^24 y 2^32	13.942.427.614	78,28%
PF menores de 2^8	15.704.922.399	88,17%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	2.107.152.911	11,83%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	219.661.526	64,33%
Entre 2^8 y 2^16	78.291.652	22,93%
Entre 2^16 y 2^24	41.340.776	12,11%
Entre 2^24 y 2^32	2.163.898	0,63%

Resultados SPEC2000

Programa: 301.apsi	Entrada: test	Tipo: float

Instrucciones

Instrucciones ejecutadas	50.206.637
Instrucciones a commit	50.195.159

Ciclos

Número total de ciclos	18.908.872
Instrucciones por ciclo	2,6546

Media ciclos entre etapas

Fetch - Dispatch	1.4383
Dispatch - Issue	3.9122
Issue - Writeback	1.0007
Writeback - Commit	1.8194

Desglose de instrucciones

TIPO	NÚMERO	% SOBRE EL TOTAL EXEC
Accesos a Memoria		
Loads	56.888	0,11%
Stores	25.021.111	49,84%
Total:	**25.077.999**	**49,95%**
Saltos		
Saltos condicionales	3.148.263	6,27%
Saltos incondicionales	10.699	0,02%
Total:	**3.158.962**	**6,29%**
Instrucciones enteras		
Add	12.968	0,03%
Sub	3.760	0,01%
Mult	19	0,00%
Div	0	0,00%
Total:	**16.747**	**0,03%**
Instrucciones PF		
Add	0	0,00%
Sub	0	0,00%
Mult	1	0,00%
Div	0	0,00%
Total:	**1**	**0,00%**

Operandos en operaciones aritméticas

Nota: los porcentajes calculados son sobre el tipo de operación (ADD, SUB…)

ADD		
Operando 0 (int)	1.468	11,32%
Operando 0 (pf)	0	#¡DIV/0!
SUB		
Operando 0 (int)	456	12,13%
Operando 0 (pf)	0	#¡DIV/0!
MULT		
Operando 0 (int)	4	21,05%
Operando 1 (int)	6	31,58%
Operando 2 (int)	0	0,00%
Operando 0 (pf)	1	100,00%
Operando 1 (pf)	0	0,00%
Operando 2 (pf)	0	0,00%
DIV		
Operando 1 (int)	0	#¡DIV/0!
Operando 2 (int)	0	#¡DIV/0!
Operando 1 (pf)	0	#¡DIV/0!
Operando 2 (pf)	0	#¡DIV/0!

Uso de registros

TIPO	NÚMERO	% DEL TOTAL USO
Enteros menores de 2^8	29.801.403	31,65%
Enteros entre 2^8 y 2^16	4.850	0,01%
Enteros entre 2^16 y 2^24	2.498	0,00%
Enteros entre 2^24 y 2^32	64.341.159	68,34%
PF menores de 2^8	94.149.910	100,00%
PF entre 2^8 y 2^16	0	0,00%
PF entre 2^16 y 2^24	0	0,00%
PF entre 2^24 y 2^32	0	0,00%

Acarreos en enteros (posición donde se rompe)

TIPO	NÚMERO	% DEL TOTAL
Menores de 2^8	4.258	73,86%
Entre 2^8 y 2^16	1.429	24,79%
Entre 2^16 y 2^24	26	0,45%
Entre 2^24 y 2^32	52	0,90%