Aspectos Avanzados de Arquitectura de Computadoras

2018

Obligatorio 1

Objetivo

Apoyar el aprendizaje de la técnica de pipeline y la jerarquía de memoria, así como estudiar diferentes técnicas de explotación de ILP utilizadas en procesadores superescalares. Familiarizarse en el uso de simulaciones y benchmarks.

Herramientas

Este obligatorio se desarrollará en un entorno linux. Se entregará una máquina virtual con linux y un conjunto de herramientas para instalar en la misma.

Trabajaremos con el simulador SimpleScalar. El simulador original y su documentación están disponibles en [1]. En ese sitio podrá encontrar además varios archivos con programas de ejemplo. Nosotros utilizaremos una versión modificada del mismo disponible en la página del curso.

Específicamente utilizaremos **sim-outorder**, una de las herramientas mas potentes y detalladas de simulación en la versión actual de SimpleScalar.

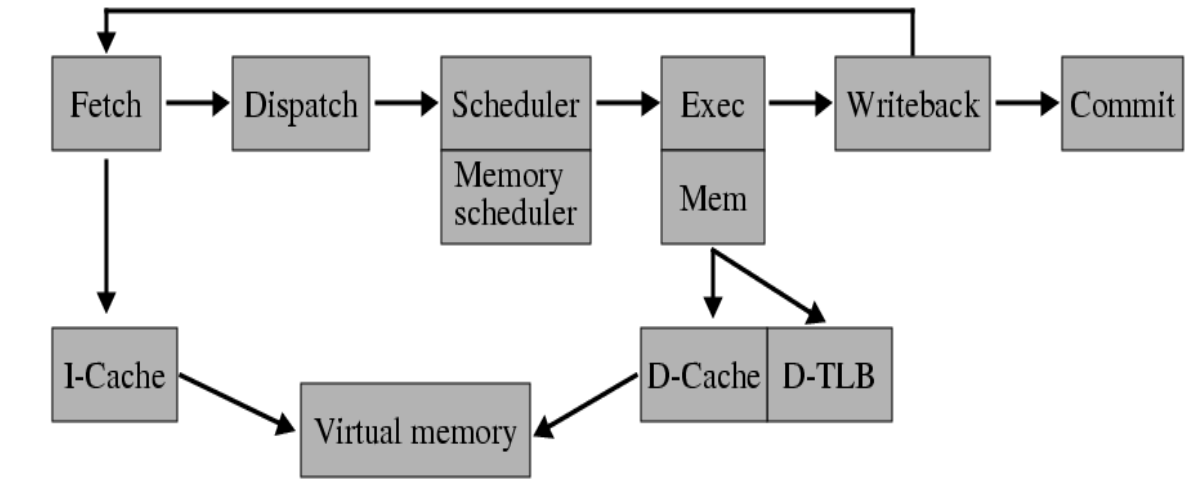

Figure 1. Pipeline for sim-outorder

La herramienta sim-outorder puede ser usada para simular pipelines avanzados. Las etapas del pipeline se muestran en la Figura 1. A diferencia del pipeline de Hennesy y Patterson, la herramienta sim-outorder puede simular un pipeline que permite emisión y ejecución fuera de orden de las instrucciones. En la Figura 1, las instrucciones pasan por seis etapas, *Fetch, Dispatch, Scheduler(Memory scheduler), Exec(Mem), Writeback, y Commit*.

La herramienta sim-outorder provee gran flexibilidad para explorar otros problemas de diseño como son el impacto del número de unidades funcionales, diferentes estrategias de predicción de saltos, etc. Fácilmente se pueden cambiar las características de la organización modificando el archivo de configuración. También es posible escribir la traza de ejecución de un programa en un archivo en el que se puede consultar el estado de la ejecución ciclo a ciclo. Para mas detalles, ver la guía de usuario [2] y otros documentos que se encuentran en [1].

Descripción general

En este obligatorio, utilizaremos el simulador SimpleScalar para estudiar la performance de la técnica de pipelining y sus variaciones para la explotación del paralelismo a nivel de instrucción. En la segunda parte analizaremos las distintas opciones de cache y se estudiarán ventajas y desventajas de cada una. Finalmente, en la última parte se investigará el impacto de diferentes técnicas estáticas.

Se pide

El obligatorio consta de resolver los problemas planteados a continuación, documentando tanto la solución como los obstáculos encontrados, utilizando las herramientas dadas para trabajar con la arquitectura. Recomendamos inspeccionar el formato solicitado para la entrega para evitar confusiones.

Parte A

Pipelining

1. **Introducción**

   (a) Describa los procesadores correspondientes a los archivos de configuración config_A1.cfg y config_A2.cfg entregados, explicando las características del pipeline, el tipo de predictor de saltos, la cantidad de unidades funcionales y los tipos y características de caches utilizadas.

   (b) Ejecute el programa de prueba *test-math* que se encuentra en tests-pisa/bin en sim-outorder utilizando ambas configuraciones y responda las siguientes preguntas:

   i. ¿Cuál es el CPI del programa ejecutado para cada configuración?

   ii. Comente los resultados justificando la variación de rendimiento.

   iii. ¿Cuántas instrucciones se ejecutaron? ¿Cuántas de estas fueron saltos?

   iv. ¿Qué tipo de predicción de saltos se utilizó? ¿Cuál fue la tasa de aciertos de direcciones del predictor?

2. **Out-of-order**

   (a) Compile y ejecute el programa *test-outorder* entregado y encuentre instrucciones ejecutadas fuera de orden. ¿Por qué se elige ejecutar fuera de orden?

   (b) Encuentre, para la configuración config_A2.cfg y el programa *test-math*, la primer instrucción ejecutada out-of-order. ¿Por qué se eligió ejecutar fuera de orden?.

3. **Superescalares**

   (a) Describa el procesador correspondiente a la configuración config_A3.cfg y ejecute *test-math* con esta configuración. Indique el valor de CPI

   (b) ¿Por qué, para la mayoría de los ciclos, solo una instrucción es ejecutada en la etapa EX? ¿Dónde esta el cuello de botella? Para mejorar el rendimiento, modifique los parámetros hasta aumentar el rendimiento en un 15%. Justifique.

4. **Predicción de saltos**

   Describa el funcionamiento de los predictores de saltos 'Always not taken' y 'Bimod'. Ejecute el benchmark go.ss en sim-simoutorder (y basándose en config_A4.cfg) con las siguientes estrategias de predicción de saltos:

   i. Always Not Taken.

   ii. Bimod con 1 entrada

   iii. Bimod con 64 entradas

   ¿Con cuál se obtiene mejor CPI? (Como entrada del go utilize 2 8.). ¿Es esto esperable? ¿Por qué?

Aclaraciones:

- Para cada parte donde se entregue código se deberá entregar el archivo de configuración correspondiente.
- En sim-outorder es posible generar trazas del pipeline y visualizarlas luego de la ejecución con la herramienta pipeview.pl (ver [2]).

Parte B

Jerarquía de Memoria

1. **Rendimiento**

   a) Para el procesador dado en config_Bcache.cfg, calcule el AMAT (en ciclos) para el programa *cache-perf* entregado.

   b) Ejecute el programa *cache-perf* entregado en sim-outorder y estudie el rendimiento de la cache de datos. Compare con el valor teórico calculado y si existe desviación de los resultados explique las posibles causas.

   c) Escriba un programa en assembler pisa que le permita determinar si el simulador implementa 'Critical Word First' o 'Early Restart'. Explique si se utiliza alguna y cómo se determina a partir de la prueba.

2. **Configuraciones de Cache**

   El tamaño de cache, tamaño de bloque y la asociatividad inciden directamente en el miss rate obtenido en la ejecución de un programa. En esta parte se realizará un estudio de la variación del *miss-rate* en función de estos parámetros para el programa *test-math*. En todos los casos se utilizará únicamente una cache de datos de primer nivel. Para cada test realizado, indique la línea utilizada en la configuración para definir la cache y grafique los resultados:

a) Para una cache con correspondencia directa, con tamaño de bloque de 8 bytes, halle el *miss_rate* cuando el tamaño de la misma es 512B, 1KB, 2 KB, 4 KB y 8KB. Grafique e interprete los resultados obtenidos.

b) Obtenga el *miss_rate* para una cache con correspondencia directa de 1 KB, 2 KB y 4 KB, y tamaño de bloque de 8, 16, 32, 64, 128 y 256 bytes. Grafique e interprete los resultados obtenidos.

c) Utilizando una cache de 1KB y 32 bytes de tamaño de bloque, halle el *miss_rate* para todos los niveles de asociatividad posibles en dicha configuración. Grafique e interprete los resultados obtenidos. ¿Por qué se utilizan caches asociativas por n-vías en lugar de caches totalmente asociativas?

3. Políticas de remplazo

a) ¿En qué situación se utilizan las políticas de remplazo en la cache?

b) Diseñe una configuración de cache para sim-outorder (y config_B2.cfg) y escriba un programa en assembler PISA donde la política FIFO tenga mejor rendimiento que la política LRU. Muestre los resultados en una simulación.

Parte C

Mejoras: Loop Unrolling & Otras Técnicas Estáticas

a) Dado el programa parteC.s, calcule aproximadamente los ciclos de espera necesarios por hazard de control (para el pipeline del simulador utilizando config_C.cfg y asuma CPI=1).

b) Explique la técnica de loop unrolling.

c) Aplique la técnica de loop unrolling en el programa parteC.s para aumentar el rendimiento en CPI en al menos un 15%. Justifique cuantitativamente y valide los resultados en sim-outorder con config_C.cfg.

Aclaraciones:

- Para cada parte donde se entregue código se deberá entregar el archivo de configuración correspondiente.
- Todos los archivos que son parte de la letra se encuentran en el eva del curso.

Descripción de la entrega

Formato de la entrega

Se debe entregar un único archivo llamado obligatorio1.tar.gz, en el formato utilizado por los comandos tar y gzip de Unix, con el siguiente contenido:

1. La documentación del obligatorio en donde se responden todas las preguntas en formato PDF con denominación **obligatorio1.pdf**. La documentación debe ser clara y concisa, con información relevante respecto a la solución realizada como pudiesen ser los objetivos, restricciones, pseudo-código, conclusiones y posibles mejoras. ***Una buena documentación es clave en la evaluación del obligatorio.***
2. **El código de la parte A4 como parteA4.s**

3. **Los códigos de la parte B1C y B3 como: parteB1C.s, parteB3.s.**
4. Los códigos de la parte C como:**, parteC_unrolled.s**
5. **(Opcional)** Una carpeta denominada “otros” con material adicional que se considere pertinente entregar (programas de prueba desarrollados, resultados de ejecución, etc...).

Es muy imporante respetar:

1. Los archivos correspondientes a los tres primeros puntos deben estar en el directorio raíz y su nombre debe ser **exactamente** el dado.
2. El archivo entregado debe tener el formato gzip-tarball.
3. El formato del archivo de documentación debe ser pdf.

Fecha de entrega

La fecha límite para realizar la entrega es el **10 de mayo de 2018 a las 23:30 horas**. La entrega se realizará a través del Moodle del curso en http://eva.fing.edu.uy . El sistema soporta múltiples entregas. Se recomienda que se realice una entrega con tiempo para verificar que el sistema le permite entregar correctamente. **Se recomienda especialmente que no deje para último momento su única entrega y que realice una con al menos contenidos parciales de forma de asegurar que no quede sin entregar.**

Observaciones.

No se aceptará bajo ninguna circunstancia una entrega realizada pasada la fecha límite estipulada. Esto incluye (pero no se restringe) a : entregas por correo electrónico y entregas en formatos físicos. En caso de que una entrega no cumpla los criterios pautados en cuanto a los formatos y mecanismos establecidos, la misma puede no ser considerada como tal y descartada.

Observación sobre instancias de no individualidad.

Está prohibido utilizar código de otros grupos, de otros años, de cualquier índole, o hacer público código, corto o largo, general o específico, a través de cualquier medio (news, correo, papeles sobre la mesa, etc.). Los grupos que a juicio de los docentes no hayan cumplido con esta norma serán eliminados, con la consiguiente pérdida del curso para todos los involucrados. Además todos los casos serán enviados a los órganos competentes de la Facultad, lo cual puede acarrear sanciones de otro carácter y gravedad para los estudiantes involucrados.

Apéndice A

SimpleScalar

- **Manual**

Antes de comenzar recomendamos leer por completo la documentación [2] y [3], y fundamentalmente lo relacionado a la herramienta sim-outorder y al Assembler SimpleScalar PISA (sección "A instruction set definition" de [2]).

- **Instalación**

Seguir los pasos dentro de Tutorial Virtual Box descargable de la página del curso [4]. La versión entregada esta especialemnte compilada para ejecutar dentro de la máquina backtrack, si se desea usar el simulador en otra plataforma deberá descargarse de [1] la versión original.

- **Ejecución**

Para ejecutar un programa dentro del simulador (por ejemplo el programa de prueba test-math que se encuentra en tests-pisa/bin):

/opt/simplescalar/simplesim/sim-outorder –config config_a.cfg –ptrace config_a.trc : –redir:sim sim_configa.out ./test-math

Con este comando ejecutamos test-math (que debe estar en el directorio donde estemos ubicados) utilizando el archivo de configuración config_a.cfg. Además se almacena la traza del pipeline en config_a.trc y el log de la simulación en sim_configa.out.

- **Compilación de assembler PISA**

Para compilar un programa escrito en assembler PISA debe ejecutar:

/opt/simplescalar/bin/sslittle-na-sstrix-gcc -nostartfiles -nostdlib -nodefaultlibs -o programa programa.s

Esto genera un binario con nombre "programa" a partir del archivo programa.s listo para ejecutar como se menciona en la parte anterior.

- **Sugerencias**
  - El tamaño de una palabra de memoria es de 4 bytes.
  - Las direcciones de memoria son de 31 bits y la disposición de los segmentos en memoria se puede ver en [3]. Cada dirección referencia a un byte de memoria.
  - Existen dos maneras de direccionar en memoria
    - offset(registro)
    - (registro1 + registro2)

Referencias

[1] http://www.simplescalar.com

[2] http://eva.fing.edu.uy/mod/resource/view.php?id=15806

[3] http://eva.fing.edu.uy/mod/resource/view.php?id=15808

[4] https://eva.fing.edu.uy/pluginfile.php/89970/mod_resource/content/1/tutorialVBox.pdf