Laboratorio 2 Organización de Computadores: Pipeline y simulación

Profesores: Felipe Garay, Erika Rosas, Nicolás Hidalgo
Ayudante: Francisco López, Pablo Ulloa
Email: {nombre}.{apellido}@usach.cl

29 de agosto de 2016

El profesor y los ayudantes son sus clientes. Si algo no queda claro es su deber como ingeniero/a hacer las consultas correspondientes para desarrollar el proyecto de forma correcta.

Enunciado

Hasta ahora durante todo el ramo se ha buscando maneras de hacer más rápidos los procesadores y programas que se ejecutan sobre ellos, comenzando sobre técnicas para comparar procesadores y ahora el pipeline, una forma de paralelizar la ejecución de las instrucciones en el procesador.

Al momento de agregar las etapas del pipeline al procesador, inmediatamente surgen problemas con las dependencias de los datos o con las mismas estructuras del procesador. Estos problemas son llamados hazards.

En este laboratorio vamos a construir un pequeño simulador de MIPS para detectar hazards en programas simples. Puede implementar el simulador en el lenguaje de los que se listan en este enunciado solamente.

El simulador debe leer un programa en MIPS (código fuente) e indicar en que puntos hay hazards (de datos y de control) a lo largo de su ejecución.

Asuma que existe una label implícita (no se requiere definir en el programa) llamada ARREGLO con un máximo de 5000 elementos de 4 bytes definida en memoria.

En usachvirtual va a poder encontrar disponibles dos programas de prueba.

Las instrucciones que usted debe implementar son las siguientes:

- add
- addi
- mul
- j
- beq
- blt
- bgt
- nop
- div
- mflo
- mfhi

- lw
- sw
- la

Estas deben comportarse de igual forma que en MARS.
Los registros a implementar son los de los siguientes rangos:

- $t0-$t9
- $a0-$a3
- $zero
- $v0
- $v1
- lo
- hi

Al final de la simulación debe escribir en un archivo llamado "END_STATE.txt" los valores finales de los registros y en "HAZARDS.txt" la ubicación en el código y el tipo de hazard encontrado.

Implemente además labels en su simulador para permitir el funcionamiento de las instrucciones de branch.

No es necesario que implemente comentarios en su simulador.

Procedimiento

1. Construya su simulador de pipeline para descubrir hazards de datos y de control.
2. Pruebe los programas proporcionados como ejemplo para encontrar los hazards.
3. Proponga al menos dos mejoras por programa, justificando con un dibujo del pipeline, que puedan reducir los hazards sin utilizar instrucciones nop.
4. Pruebe nuevamente sus soluciones para mostrar que se obtuvieron menos hazards, si no es así entonces explique las razones y vuelta a intentar dar una solución en el paso anterior.

Pistas e información adicional

Etapas del pipeline

Las etapas del pipeline pueden ser vistas como funciones, cada una afectando una cierta parte del estado del programa (o del procesador). No es necesario que se ejecuten de manera paralela.

Lenguajes permitidos para construir el simulador

- C/C++ (gcc, debe correr en GNU/Linux).
- Python.
- Ruby.
- Bash.
- Node/Javascript.

- Java/Scala.
- OCaml.
- Haskell.

Si quiere utilizar algún otro lenguaje consulte primero.

Limitaciones de entrada

La entrada puede ser un programa en MIPS compuesto de las instrucciones mencionadas en este enunciado y solamente utilizando los registros solicitados. El archivo puede contener espacios

Informe, análisis y bibiliografías

El informe contiene una introducción que consiste en los siguientes elementos:

- Motivación: ¿Por que es importante el problema?
- Objetivo general: Es uno solo. Su conclusión se debe hacer a partir de este objetivo.
- Objetivos específicos: Cada objetivo específico es lo que se debe hacer en el trabajo para poder cumplir con el objetivo general. Son numerados.
- Problema: ¿Qué es lo que se intenta resolver?. Se plantea generalmente como una pregunta.
- Herramientas: ¿Qué programas o, en general, herramientas utilizó para el trabajo?. Indique las versiones.
- Organización del documento: ¿Cuáles son los capítulos de su informe y que contienen?

Un análisis es explicar cada uno de los fenómenos que se observan en sus experimentos. Si usted obtiene por ejemplo en una ejecución E_1 0.2 segundos y en una ejecución E_2 0.3, indicar estos resultados sin justificar no es un análisis. Un análisis podría ser: "Dado que E_1 utiliza las instrucciones X en vez de las Y como en el caso de E_2, se obtienen mejores tiempos ya que el CPI de las instrucciones X es Z". Claramente este es solamente un ejemplo y se pueden hacer muchos análisis más.

Otro punto a tomar en cuenta es que puede que obtenga resultados erróneos a simple vista, trate de investigar fuera de lo visto en clases para tratar de encontrar una explicación.

Comentarios

Una buena forma de documentar un programa es comentando las entradas y salidas de las funciones. Por ejemplo en C:

```
/**
* Suma dos números.
* @param x El primer número a sumar.
* @param y El segundo número a sumar.
* @return La suma de x e y.
*/
int sumar(int x, int y){
  return x + y;
}
```

O en MIPS:

```
# Suma de dos números
# Argumentos:
# $a0: primer número
# $a1: segundo número
# Retorna la suma de $a0 con $a1
add $v0, $a0, $a1
jr $ra
```

Comentar todas las líneas puede resultar excesivo, incluso en ensamblador, pero queda a libertar suya si quiere hacerlo o no.

Referencias del informe

Existen muchos estilos de referencias que se pueden utilizar en un informe pero las más utilizadas en los papers y en el departamento de informática son: APA e IEEE. Si está usando LaTeXpuede utilizar bibtex para autogenerar estas referencias a partir de un archivo .bib.

Si usted está diciendo por ejemplo: "MIPS es una arquitectura de procesadores RISC que se ha utilizado para...", eso fue sacado de alguna parte, entonces en el texto debe incluir de donde lo sacó. "MIPS es una arquitectura de procesadores RISC que se ha utilizado para... [1]"(en caso de utilizar IEEE). De esta forma el lector puede ir al final de su informe y revisar la referencia número 1 donde puede obtener más información o saber que lo que usted puso está justificado por algún experto en el área.

Imágenes y tablas sacadas de libros o Internet también deben ser referenciadas en el subtítulo de cada una de ellas. Por ejemplo: "Figura 1: Tipos de instrucciones en la arquitectura MIPS [2]."

MoSCoW del laboratorio

MoSCoW es un método para priorizar tareas en un proyecto. Consiste en asignar un nivel de necesidad de entre los cuatro disponibles. Nosotros vamos a utilizar solamente dos: "must have" y "should have".

Todo lo que se indique como "must have" debe ser incluido para recién comenzar a evaluar el laboratorio mientras que todo lo que quede fuera de esta categoría se considera como "should have" y afecta solamente la nota. En otras palabras, si falta un "must have" se va a calificar con la nota mínima tanto programa como informe.

Los siguientes puntos son los "must have":

- Cada análisis debe tener su correspondiente experimento.
- Cada experimento debe tener su correspondiente programa.
- Tanto programa como informe deben ser entregados primero por usachvirtual y luego el informe de forma impresa.
- El análisis del informe debe poder jusficiar los resultados obtenidos.
- El simulador debe funcionar correctamente con los archivos de prueba, esto incluye los reportes tanto de estado final de los registros como cantidad de hazards encontrados.
- Un Makefile debe ser incluido para construir sus programas en caso de ser necesario.
- Sus programas deben funcionar en GNU/Linux incluyendo la lectura de archivos con el separador de líneas de UNIX.
- Referencias al final del informe deben ser utilizadas en el texto.
- Las mejoras de programas deben entregar la misma respuesta que los programas originales.
- No se pueden utilizar librerías externas al lenguaje (solo librerías estándar) con la excepción de generadores de parsers

Consideraciones

Generales

- Las copias entre compañeros e Internet serán calificadas con la nota mínima.
- El laboratorio es individual.
- En caso de no cumplir con un "must have" se va a calificar con la nota mínima.
- Si el laboratorio es entregado con 4 o más días de atraso se considerará que no se ha presentado ningún trabajo.
- Si no entrega uno de los 3 laboratorios del semestre entonces se reprueba todo el laboratorio.
- El simulador y programas (mejoras) corresponden a un 40 % de la nota y el informe un 60 % de la nota de entrega del laboratorio. Note que no puede hacer el informe sin hacer el o los programas ya que debe basarse en ellos.
- Por cada día de atraso se descuenta 1 punto a la nota general del laboratorio (esto incluye la entrega atrasada de informes). Ej: si el laboratorio se debe entregar a las 23:50 y se entrega a las 23:55 hay un punto de descuento. Si se entrega a las 23:55 del siguiente día hay dos puntos de descuento. Si tiene un 7 como nota de laboratorio (promedio entre informe y programa) entonces tendría un 6.
- Debe entregar en el espacio habilitado en usachvirtual una carpeta comprimida (.zip o .tar.gz o .tar.bz2) con el código fuente del programa en una carpeta llamada "src" y el informe y manual de usuario en pdf. Este archivo debe llamarse: Apellido1_Apellido2.(zip—tar.bz2)
- La fecha de entrega es el jueves 3 de noviembre a las 23:55 por usachvirtual. No se corregirán laboratorios entregados por otros medios.
- El laboratorio consiste en informe más programa. Si no entrega uno de estos elementos se calificará con la nota mínima.

Programa

- Recuerde poner comentarios en su programa e indicar los argumentos que reciben las funciones y que es lo que retornan (también indique aquellas funciones que no tienen un valor de retorno).
- El programa debe funcionar en sistemas GNU/Linux. Para que el programa se califique de forma correcta debe poder ejecutarse sin problemas en este sistema operativo, de lo contrario no se va a poder evaluar el programa.
- Los programas consisten en las siguientes ponderaciones:
  1. Simulador (80 %).
  2. Mejoras (20 %) (promedio entre todas las mejoras).

Informe

- El informe debe ser entregado impreso en secretaría con el nombre del profesor, ayudantes y asignatura a más tardar 12 hrs desde la fecha de entrega por usachvirtual. Además debe subir el informe junto con el programa a usachvirtual. Ej: Si el laboratorio se debe entregar a las 23:50 por usachvirtual, entonces el informe debe estar a las 12:00 en secretaría. Se aplicará el mismo criterio que el que aparece en las consideraciones generales para los atrasos.

- El informe debe contener las siguientes secciones (consulte el formato de memoria para más información sobre lo que deben llevar estas secciones):

    1. Introducción: Objetivo generales, objetivos específicos, organización del documento, motivación, problema, herramientas. (10 %)
    2. Marco teórico: Lo que un lector debería saber para entender su trabajo (**NO ES UN GLOSARIO**). (15 %)
    3. Desarrollo: Explicar como se construyó el simulador y presentar resultados tanto de programas como de las pruebas.(35 %)
    4. Análisis: Explicar los resultados utilizando conceptos vistos en clases justificando las mejoras. (20 %)
    5. Conclusión: Sobre los objetivos planteados en la introducción. (15 %)
    6. Referencias (5 %)

- Debe utilizar el formato de presentación de memoria disponible en el archivo "Propuesta de normas para presentación del trabajo de titulación de pregrado, Departamento de Ingeniería Informática" que puede encontrar en la página del curso. Se descontará una décima por cada error en el formato.

- Cuide la ortografía. Se descontará una décima por cada falta.

- Incluya referencias en la bibliografía indicando claramente el texto citado. Utilice el formato APA (parte del formato del informe) o IEEE. En caso de encontrar textos que no hayan sido citados se considerará como copia.