Laboratorio 1 Organización de Computadores: Benchmarks

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

22 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

En este laboratorio usted debe construir un programa para medir el desempeño de un computador. Puede elegir ya sea escribirlo directamente en C pero debe ser ejecutado en un emulador de MIPS.

Una vez tenga el programa para hacer el benchmark de la CPU de su computador junto con el emulador utilizado debe escribir un programa en C que permita ordenar un arreglo utilizando un algoritmo de ordenamiento iterativo y otro recursivo (ambos a su elección) los cuales deben leer un archivo de entrada de números (uno por cada línea en el rango de un entero con signo) y generar un archivo de salida ordenado. Los nombres de ambos archivos son pasados como argumento de la forma: "./PROGRAMA ARCHIVO_ENTRADA.txt ARCHIVO_SALIDA.txt" ambos con líneas terminadas en "\n" como en UNIX. Además cree un Makefile para poder compilar todos sus programas de manera inmediata SOLO las veces necesarias.

Trate de estimar el tiempo que le va a tomar a su programa ejecutarse a partir de los benchmarks realizados anteriormente y luego verifique con los resultados obtenidos realmente.

En total son 3 programas que usted debe entregar.

Procedimiento

1. Escriba sus dos programas de ordenamiento.
2. Escriba su programa de benchmark para obtener los MIPS (Million instructions per second) tomando en cuenta los distintos tipos de instrucciones utilizadas en sus programas anteriores.
3. Estime el tiempo que le va a tomar ejecutarse a ambos programas.
4. Modifique el programa de benchmark para que ahora muestre además de los MIPS los tiempos estimados de sus programas de manera tal que se puedan calcular fácilmente en cualquier computador.
5. Obtenga los tiempos reales de ejecución para los parámetros de optimización: -O0 (O cero), -O1, -O2, -O3 y -Os.
6. Analice los resultados obtenidos en el informe.

Pistas e información adicional

Docker

Dado que estamos trabajando en procesadores x86_64 es necesario utilizar emuladores para poder ejecutar código de MIPS. Uno de estos emuladores es qemu que puede ser instalado para ejecutar binarios de otras arquitecturas en vez de un sistema operativo completo.

Para facilitar la instalación de este software podemos utilizar Docker, un sistema que permite crear contendores de software a modo de una máquina virtual pero en vez de virtualizar todo un computador solamente lo hace con algunos recursos y otros se comparten (como el kernel).

En sistemas Debian/Ubuntu se puede obtener Docker por medio de:

```
sudo apt-get install docker.io
```

Luego puede bajar el contendor que incluye el software necesario para ejecutar el emulador por medio de:

```
sudo docker pull felipegaray/orga
```

Una vez se obtiene el contenedor se puede ejecutar bash dentro de este:

```
sudo docker run -it felipegaray/orga bash
```

Note que al ser un entorno virtualizado no puede acceder desde dentro del contedor hacia el sistema operativo host. Puede utilizar volúmenes para montar directorios externos dentro del contendor (`https://docs.docker.com/engine/tutorials/dockervolumes/`).

Puede obtener información en: `https://docs.docker.com/engine/understanding-docker/`

Qemu y compiladores para MIPS

Si no quiere utilizar Docker puede instalar directamente el emulador y compiladores con:

```
sudo apt-get install qemu-user-static qemu gcc-5-mipsel-linux-gnu
```

Ahora se pueden ejecutar binarios de MIPS mediante:

```
qemu-mipsel-static -L /usr/mipsel-linux-gnu/ ./PROGRAMA
```

Para compilar un programa en C a MIPS se debe hacer de la siguiente forma:

```
mipsel-linux-gnu-gcc-5 CODIGO.c -o PROGRAMA
```

Aunque se recomienda agregar la flag "-Wall" para obtener advertencias que puedan resultar útiles para crear programas con menos errores.

Benchmarks

Para calcular los MIPS recuerde la fórmula:

$$MIPS = \frac{NI}{T * 10^6} \tag{1}$$

Dónde NI es el número de instrucciones y T es el tiempo utilizado para ejecutar esas instrucciones.

Dado que estamos ejecutando un programa en C: ¿Cómo podemos saber el número de instrucciones generadas?

Es posible obtener el código ensamblador mediante el siguiente programa:

```
mipsel-linux-gnu-objdump -d ./a.out
```

Si quiere puede redireccionar la salida a un archivo para facilitar la lectura. El compilador de C genera labels con los nombres de las funciones del programa, así que puede buscar por "main" para encontrar el punto de entrada de su programa.

Mezclando C con ensamblador

Es posible incluir código ensamblador en C mediante "Inline Assembly" (`https://gcc.gnu.org/onlinedocs/gcc/Using-Assembly-Language-with-C.html#Using-Assembly-Language-with-C`).

Con esto puede ser más fácil controlar que instrucciones son las que se están midiendo en el benchmark.

Ejemplo de salida del programa de benchmark

Su programa de benchmark debe tener al menos 3 salidas las cuales pueden ser las siguientes:

```
MIPS calculados: XXXX
Tiempo de ejecución del programa iterativo (-O0): AAAAAs (Estimado)
Tiempo de ejecución del programa iterativo (-O1): BBBBs (Estimado)
Tiempo de ejecución del programa iterativo (-O2): CCCCs (Estimado)
Tiempo de ejecución del programa iterativo (-O3): DDDDs (Estimado)
Tiempo de ejecución del programa iterativo (-Os): DDDDs (Estimado)
Tiempo de ejecución del programa recursivo (-O0): EEEEs (Estimado)
Tiempo de ejecución del programa recursivo (-O1): FFFFs (Estimado)
Tiempo de ejecución del programa recursivo (-O2): GGGGs (Estimado)
Tiempo de ejecución del programa recursivo (-O3): HHHHs (Estimado)
Tiempo de ejecución del programa recursivo (-Os): IIIIs (Estimado)
```

Midiendo tiempos

Tiene tres alternativas para medir los tiempos de su programa: el comando "time", obtener los tiempos mediante las funciones de "sys/time.h" o mediante gprof.

Cada una tiene sus ventajas y desventajas, por ejemplo time mide el tiempo total de ejecución de su programa sin discriminar entre la lectura/escritura de archivo y su algoritmo (que es lo que hay que medir). Las funciones de "sys/time.h" tienen una buena resolución (de entre los milisegundos) y se pueden poner donde usted quiera dentro del programa. Finalmente gprof permite hacer un profile de todo su programa por función incluyendo la cantidad de llamadas a cada una de ellas pero debe tener en cuenta que esto agrega algo de overhead-

Sus pruebas, tanto del cálculo de MIPS como de los cálculos del tiempo, pueden verse afectada por otros programas que se encuentran en ejecución: ¿Cómo podría asegurar que los tiempos que está obteniendo son cercanos a la realidad? Aplique estadísticas para justiciar el número de muestras y resultados obtenidos.

Cuando obtenga sus pruebas analice si se parecen con sus estimaciones o no. Trate de justificar los resultados.

Iterativo vs recursivo

Sabemos que para poder hacer una función recursiva es necesario utilizar el stack, pero si se utiliza recursión por cola el compilador puede transformar la llamada a la función por una iteración de instrucciones de salto normales para compilaciones con la opción de optimización. Tome en cuenta esto si es que su programa cumple con la condición de optimización.

Otro punto a tomar en cuenta es el acceso al stack ya que debe considera mayor cantidad de instrucciones que acceden a la memoria

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?

El desarrollo del informe debe indicar como usted hizo el laboratorio. Paso a paso tratar de explicar el proceso lógico que siguió para lograr obtener sus resultados o las decisiones tomadas que lo llevaron a completar el desarrollo.

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.

Sobre los niveles de optimización

Cada uno de los niveles de optimización entrega cantidad y tipo de instrucciones distintas lo que puede afectar los tiempos de ejecución.

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;
}
```

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.
- Cada programa debe entregar la respuesta correcta (archivo con los números ordenados de menor a mayor).
- Los programas deben ser realizados en C.
- Los programas deben obtener los nombres de archivo mediante los argumentos de la línea de comandos.
- Los programas deben obtener la entrada desde el archivo de texto dado como argumento.
- Los programas deben obtener la salida desde el archivo de texto dado como argumento.
- Tanto programa como informe deben ser entregados primero por usachvirtual y luego el informe de forma impresa.
- El análisis del informe debe poder justificar los resultados obtenidos.
- El Makefile debe ser incluido para construir sus programas.
- Sus programas deben funcionar en GNU/Linux (específicamente el contenedor de Docker entregado) incluyendo la lectura de archivos con el separador de líneas de UNIX.
- El programa de benchmark debe entregar una salida con al menos la información mostrada en el ejemplo.
- Referencias al final del documento deben ser utilizadas en el texto.
- En el desarrollo debe estar justificada la construcción del benchmark.

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.

- Los programas corresponden a un 20 % de la nota y el informe un 80 % 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 29 de de septiembre 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 valen todos lo mismo y son los siguientes
    1. Programa de benchmark completo.
    2. Programa de ordenamiento iterativo.
    3. Programa de ordenamiento recursivo.

Informe

- El informe debe ser entregado impreso en secretaría con el nombre de 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 programa y presentar resultados (25 %)
    4. Análisis: Explicar los resultados utilizando conceptos vistos en clases (30 %)
    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.