DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico II

subtitulo del trabajo

Organización del Computador II

Primer Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Nombre	XXX/XX	`mail`
Nombre	XXX/XX	`mail`

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

En el presente trabajo se describe la problemática de ...

Índice

Figura 1: Descripcion de la figura

1. Objetivos generales

El objetivo de este Trabajo Práctico es ...

2. Contexto

Titulo del parrafo Bla bla bla bla. Esto se muestra en la figura 1.

```
struct Pepe {

    ...

};
```

3. Enunciado y solucion

Filtro *tiles*

Programar el filtro *tiles* en lenguaje C y luego en ASM haciendo uso de las instrucciones vectoriales (**SSE**).

Experimento 1 - análisis el código generado

Utilizar la herramienta `objdump` para verificar como el compilador de C deja ensamblado el código C. Como es el código generado, ¿cómo se manipulan las variables locales?¿le parece que ese código generado podría optimizarse?

Experimento 2 - optimizaciones del compilador

Compile el código de C con optimizaciones del compilador, por ejemplo, pasando el flag `-O1`[1]. ¿Qué optimizaciones realizó el compilador? ¿Qué otros flags de optimización brinda el compilador? ¿Para qué sirven?

Experimento 3 - secuencial vs. vectorial

Realice una medición de las diferencias de performance entre las versiones de C y ASM (el primero con -O1, -O2 y -O3).
¿Como realizó la medición?¿Cómo sabe que su medición es una buena medida?¿Cómo afecta a la medición la existencia de *outliers*[2]?¿De qué manera puede minimizar su impacto?¿Qué resultados obtiene si mientras corre los tests ejecuta otras aplicaciones que utilicen al máximo la CPU? Realizar un análisis **riguroso** de los resultados y acompañar con un gráfico que presente estas diferencias.

Experimento 4 - cpu vs. bus de memoria

Se desea conocer cual es el mayor limitante a la performance de este filtro en su versión ASM.

¿Cuál es el factor que limita la performance en este caso? En caso de que el limitante fuera la intensidad de cómputo, entonces podrían agregarse instrucciones que realicen accesos a memoria y la performance casi no debería sufrir. La inversa puede aplicarse si el limitante es la cantidad de accesos a memoria.

Realizar un experimento, agregando múltiples instrucciones de un mismo tipo y realizar un análisis del resultado. Acompañar con un gráfico.

Experimento 5 (*opcional*) - secuencial vs. vectorial (parte II)

Si vemos a los pixeles como una tira muy larga de bytes, este filtro en realidad no requiere ningún procesamiento de datos en paralelo. Esto podría significar que la velocidad del filtro de C puede aumentarse hasta casi alcanzar la del de ASM. ¿ocurre esto?

Modificar el filtro para que en vez de acceder a los bytes de a uno a la vez se accedan como tiras de 64 bits y analizar la performance.

Filtro *Popart*

Programar el filtro *Popart* en lenguaje C y en en ASM haciendo uso de las instrucciones vectoriales (**SSE**).

Experimento 1 - saltos condicionales

Se desea conocer que tanto impactan los saltos condicionales en el código del ejercicio anterior con `-O1`.
Para poder medir esto, una posibilidad es quitar las comparaciones al procesar cada pixel. Por más que la imagen resultante no sea correcta, será posible tomar una medida del impacto de los saltos condicionales. Analizar como varía la performance.

Si se le ocurren, mencionar otras posibles formas de medir el impacto de los saltos condicionales.

[1] agregando este flag a `CCFLAGS64` en el makefile

[2] en español, valor atípico: `http://es.wikipedia.org/wiki/Valor_atpico`

Experimento 2 - cpu vs. bus de memoria

¿Cuál es el factor que limita la performance en este caso?

Realizar un experimento, agregando múltiples instrucciones de un mismo tipo y realizar un análisis del resultado. Acompañar con un gráfico.

Experimento 3 - prefetch

La técnica de *prefetch* es otra forma de optimización que puede realizarse. Su sustento teórico es el siguiente:

Suponga un algoritmo que en cada iteración tarda n ciclos en obtener un dato y una cantidad similar en procesarlo. Si el algoritmo lee el dato i y luego lo procesa, desperdiciará siempre n ciclos esperando entre que el dato llega y que se comienza a procesar efectivamente. Un algoritmo más inteligente podría pedir el dato $i+1$ al comienzo del ciclo de proceso del dato i (siempre suponiendo que el dato i pidió en la iteración $i-1$. De esta manera, a la vez que el procesador computa todas las instrucciones de la iteración i, se estáran trayendo los datos de la siguiente iteración, y cuando esta última comience, los datos ya habrán llegado.

Estudiar esta técnica y proponer una aplicación al código del filtro en la versión ASM. Programarla y analizar el resultado. ¿Vale la pena hacer prefetching?

Experimento 3 - secuencial vs. vectorial

Analizar cuales son las diferencias de performace entre las versiones de C y ASM. Realizar gráficos que representen estas diferencias.

Filtro *Temperature*

Programar el filtro *Temperature* en lenguaje C y en en ASM haciendo uso de las instrucciones vectoriales (**SSE**).

Experimento 1

Analizar cuales son las diferencias de performace entre las versiones de C y ASM. Realizar gráficos que representen estas diferencias.

Filtro *LDR*

Programar el filtro *LDR* en lenguaje C y en ASM haciendo uso de las instrucciones **SSE**.

Experimento 1

Analizar cuales son las diferencias de performace entre las versiones de C y ASM. Realizar gráficos que representen estas diferencias.

4. Conclusiones y trabajo futuro