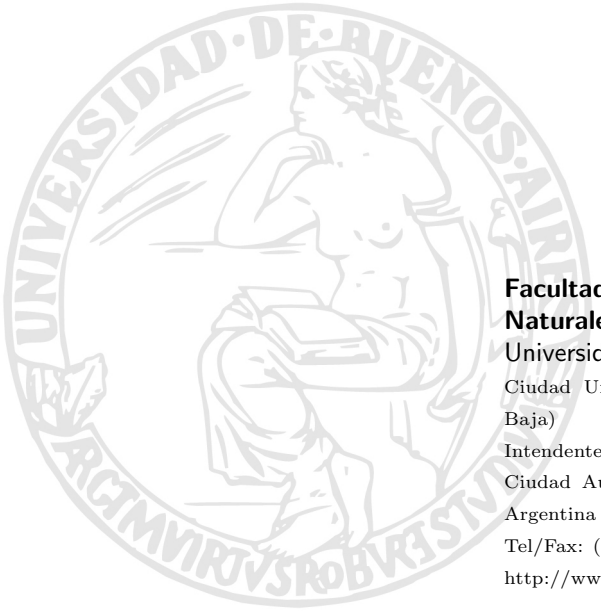
Trabajo Práctico

September 28, 2014 — Organización Del Computador 2

Integrante	LU	Correo electrónico
Fosco, Martin Esteban	449/13	mfosco2005@yahoo.com.ar
Palladino, Julián	231/13	julianpalladino@hotmail.com
De Carli, Nicolás	164/13	nikodecarli@gmail.com



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) 4576-3300
http://www.exactas.uba.ar

1 Introducción

Este Trabajo Práctico se ha centrado en explorar el modelo de programación **SIMD**, usándolo para una aplicación popular del set de instrucciones SIMD de intel (SSE), procesamiento de imágenes y videos.
En particular, se implementaron 4 filtros de imágenes: Cropflip, Sierpinski, Bandas y Motion Blur.

Se ha buscado aprovechar los beneficios de SSE:

- Procesar conjuntos de datos de manera eficiente, ejecutando de manera paralela (al mismo tiempo) la misma instrucción sobre distintos datos.
- El uso de los registros XMM, los cuales proveen una gran versatilidad con la opción de ejecutar operaciones de punto flotante y enteras con distintas precisiones (sobre datos empaquetados o escalares).
- Minimizar los accesos a memoria.
- No se me ocurrió nada mas, pero llenen o borren, como les parezca mejor :)

Luego de implementar en C y asm (procesando de manera escalar y vectorial los datos, respectivamente) los filtros de imágenes se ha comparado la performance de ambos modelos de programación para determinar de manera aproximada la ventaja que se gana al trabajar sobre muchos datos de manera simultánea.

2 Desarrollo

1.1)

a) Al hacer el "objdump" no solo se imprime la función "cropflip_c.c", sino que también se imprimen varias funciones que utiliza GDB para hacer el debugging, tales como ".debug_info", ".debug_abbrev", ".debug_aranges", etc. También imprime ".comment" y "eh_header", que contienen información sobre el Linker y el compilador.

b) Las variables locales las almacena en memoria, haciendo movs manualmente en el stack frame. (Por ej, poniendo las variables en [rbp-0x58], [rbp-0x60], [rbp-0x64]).

c) Se podría optimizar el almacenamiento de las variables locales. Como el acceso a memoria es mucho más ineficiente que el acceso a los registros, sería ms óptimo almacenar las variables en ellos.

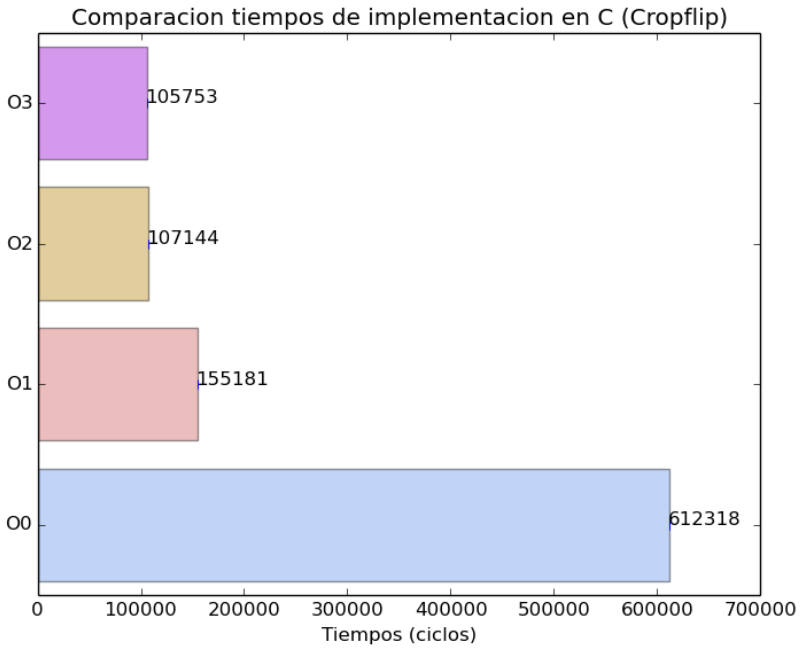
1.2)

a) -O1 no reduce mucho el tiempo de compilación, y ejecuta una optimización "moderada". Se ve claramente que se reducen los accesos a memoria para las variables locales, y se reducen mucho las líneas de código.

b) Usando otros parámetros como -O2 u -O3 se hace la compilación con ms optimizaciones cuanto ms grande sea el número. También están -Os que optimiza el tamaño del código y -Og que optimiza el debugging.

c)

- **FDCE:** Hace eliminación de "dead code" (Código muerto), es decir, borra el código que consume recursos pero sus resultados nunca son utilizados.
- **FDSE:** Hace eliminación de "dead store" (Almacenamiento muerto), o sea, ignora aquellas variables que después no llegan a ser utilizadas.
- **-fif-conversion y -fif-conversion2:** Reemplazan condicionales por equivalentes aritméticos. Esto incluye funciones como movimientos condicionales, mínimos, máximos, función "abs", entre otros. Luego de ver los resultados del experimento 3.1 (Saltos condicionales en el filtro Bandas) queda claro que esta optimización es extremadamente útil.



En el gráfico se nota la clara diferencia entre las corridas con optimizaciones y sin ellas. El criterio fue el mismo que aplicamos en los experimentos de más adelante (Haciendo 1000 iteraciones).

1.3)

En este experimento consideraremos diferentes criterios realizando siempre 10 mediciones, con outliers, sin ellos, y agregando programas en C++ que trabajen en simultáneo.

a) Resultados:

422861, 242136, 186857,
194709, 234097, 221979,
247912, 227976, 223245,
200770.

b) Resultados:

289209, 302654, 258757,
254428, 248527, 254606,
260391, 338981, 373612,
280937

c)
10 mediciones
Esperanza: 240254.2 Desvío standard: 67250.78758 Varianza: 4522668429.51111
10 mediciones con el programa en simultáneo:
Esperanza: 286210.2 Desvío standard: 41607.01932 Varianza: 1731144056.62222

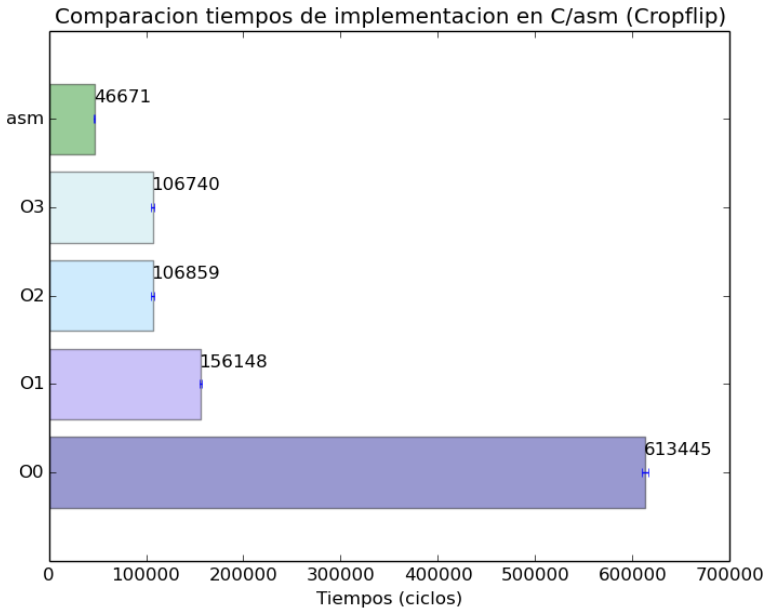
d) 10 mediciones sin 2 outliers:
Esperanza: 224103 Desvío standard: 18595.78063 Varianza: 345803057.14286

e)
INSERT GRAFICOU HERE

1.4)

C O0- 613445 ciclos
C O1- 156148 ciclos
C O2- 106859 ciclos
C O3- 106740 ciclos

asm- 46671



3 Conclusión