

Taller #3

- 1- Indique según la teoría ¿cómo se aplican las optimizaciones y para qué son? Ejecútelas y contraste la diferencia entre tres de ellas. Sugerencia: busque el término optimización gcc.

La optimización en compiladores, como GCC (GNU Compiler Collection), se refiere al proceso de mejorar el rendimiento y/o la eficiencia del código generado, sin cambiar su comportamiento externo. Las optimizaciones pueden mejorar la velocidad de ejecución, reducir el tamaño del código o ambos. GCC ofrece una amplia variedad de opciones de optimización que pueden ser utilizadas para adaptar el comportamiento del compilador a las necesidades específicas de una aplicación.

-O0 vs. -O2:

-O0: No realiza ninguna optimización. Es el nivel predeterminado y es útil durante la depuración, ya que no altera el flujo del código.

-O2: Realiza optimizaciones que no involucren un compromiso de espacio-tiempo. Puede mejorar significativamente el rendimiento sin aumentar mucho el tamaño del código.

-O2 vs. -O3:

-O2: Como se mencionó anteriormente, realiza optimizaciones sin comprometer el espacio.

-O3: Realiza todas las optimizaciones de -O2 y añade otras que pueden aumentar el tamaño del código. Es el nivel más agresivo de optimización.

-O2 vs. -Os:

-O2: Optimiza el código para el rendimiento sin comprometer el espacio.

-Os: Optimiza el código para reducir su tamaño. Puede comprometer algunas optimizaciones de rendimiento para lograr un código más pequeño.

- 2- ¿Qué información le proporciona el simulador sim-fast sobre la aplicación?

La información proporcionada ofrece detalles sobre cómo se ejecutó una aplicación en un simulador, incluyendo estadísticas sobre la ejecución de instrucciones, la segmentación de memoria y las estadísticas relacionadas con el acceso y asignación de memoria.

- 3- Utilice el simulador sim-outorder por medio del uso de la opción -issue:inorder/ outoforder (defecto) , determine si la ejecución fuera de orden proporciona algún eneficio para la aplicación y explique según la teoría. Proporcione los resultados necesarios para sustentar su respuesta.

```
johnnyzaet@Kali: ~/Desktop/ez-install-simplescalar-op1
ld_stack_size          16384 # program initial stack size
ld_prog_entry          0x00400140 # program entry point (initial PC)
ld_environ_base        0x7fff8000 # program environment base address address
ld_target_big_endian    0 # target executable endian-ness, non-zero if big endian
mem.page_count         26 # total number of pages allocated
mem.page_mem           104k # total size of memory pages allocated
mem.ptab_misses        26 # total first level page table misses
mem.ptab_accesses      575864 # total page table accesses
mem.ptab_miss_rate     0.0000 # first level page table miss rate

root@b43ee73d84ec:~/build/ejercicio# ls
sim_fast.txt      sim_inorder2.txt  sim_outorder1.txt  sim_profile.txt  test
sim_inorder1.txt  sim_outorder.txt  sim_outorder2.txt  sim_safe.txt     test-llong.c
root@b43ee73d84ec:~/build/ejercicio# diff -u sim_outorder1.txt sim_outorder2.txt &> diffOutorder.txt
root@b43ee73d84ec:~/build/ejercicio# diff -u sim_inorder1.txt sim_inorder2.txt &> diffInorder.txt
root@b43ee73d84ec:~/build/ejercicio# grep -e 'sim_IPC' -e 'sim_cycle' -e 'sim_CPI' diffOutorder.txt
-sim_cycle          33339 # total simulation time in cycles
-sim_IPC            0.5590 # instructions per cycle
-sim_CPI            1.7888 # cycles per instruction
+sim_cycle          25256 # total simulation time in cycles
+sim_IPC            0.7380 # instructions per cycle
+sim_CPI            1.3551 # cycles per instruction
root@b43ee73d84ec:~/build/ejercicio# grep -e 'sim_IPC' -e 'sim_cycle' -e 'sim_CPI' diffInorder.txt
-sim_cycle          40063 # total simulation time in cycles
-sim_IPC            0.4652 # instructions per cycle
-sim_CPI            2.1495 # cycles per instruction
+sim_cycle          36535 # total simulation time in cycles
+sim_IPC            0.5101 # instructions per cycle
+sim_CPI            1.9602 # cycles per instruction
root@b43ee73d84ec:~/build/ejercicio#
```

Como se puede observar en la imagen anterior, al utilizar el issue:inorder hace que el programa se ejecute más lento que fuera de orden.

- 4- El simulador sim-outorder le permite modificar el número de unidades funcionales, Explique sus resultados con base en el código fuente de la aplicación que se está analizando. Contraste el funcionamiento con respecto al archivo sin modificaciones.

```
root@b43ee73d84ec:~/build/ejercicio# /root/build/simplesim-3.0/sim-outorder -res:fpu 1 -res:imult 1 -res:ialu 1 -res:f
pmult 1 -res:memport 2 test &> sim_outorder1.txt
root@b43ee73d84ec:~/build/ejercicio# /root/build/simplesim-3.0/sim-outorder -res:fpu 4 -res:imult 1 -res:ialu 4 -res:f
pmult 1 -res:memport 2 test &> sim_outorder2.txt
root@b43ee73d84ec:~/build/ejercicio#
```