Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE4301– Arquitectura de Computadores I
Johnny Zaet Agüero Sandí

Taller #4

1- Analice los siguientes aspectos con el comando grep (tome en cuenta que - es para 1 unidad y + es para 8 unidades

```
johnnyzaet@Kali: ~/Desktop/ez-install-simplescalar-op1
root@dd1dd574f1dc:~/build/prueba/wattchg7/pruebas# cat diff.txt | grep -E "Total Power Consumption|Int ALU Power|FP ALU Power
|avg_alu_power|avg_falu_power|total_power"
-Total Power Consumption: 57.7939
+Total Power Consumption: 113.283
-Int ALU Power: 1.16503  (2.09%)
-FP ALU Power: 3.57026  (6.4%)
+Int ALU Power: 9.32026  (8.38%)
+FP ALU Power: 28.5621  (25.7%)
-avg_alu_power                4.7353 # avg power usage of alu
-avg_falu_power               3.5703 # avg power usage of falu
+avg_alu_power               37.8824 # avg power usage of alu
+avg_falu_power              28.5621 # avg power usage of falu
-total_power            3976431.5994 # total power per cycle
-avg_total_power_cycle       55.7939 # average total power per cycle
-avg_total_power_cycle_nofp_nod2      48.0145 # average total power per cycle
-avg_total_power_insn        81.5227 # average total power per insn
-avg_total_power_insn_nofp_nod2      70.1560 # average total power per insn
-avg_alu_power_cc1            0.6673 # avg power usage of alu_cc1
-total_power_cycle_cc1   832806.2881 # total power per cycle_cc1
-avg_total_power_cycle_cc1      11.6852 # average total power per cycle_cc1
-avg_total_power_insn_cc1      17.0737 # average total power per insn_cc1
+total_power            6254996.0864 # total power per cycle
+avg_total_power_cycle      111.2830 # average total power per cycle
+avg_total_power_cycle_nofp_nod2      78.5118 # average total power per cycle
+avg_total_power_insn       132.9690 # average total power per insn
+avg_total_power_insn_nofp_nod2      93.8116 # average total power per insn
+avg_alu_power_cc1            3.2405 # avg power usage of alu_cc1
+total_power_cycle_cc1   845576.9120 # total power per cycle_cc1
+avg_total_power_cycle_cc1      15.0437 # average total power per cycle_cc1
+avg_total_power_insn_cc1      17.9753 # average total power per insn_cc1
-avg_alu_power_cc2            0.6791 # avg power usage of alu_cc2
-total_power_cycle_cc2   361055.8499 # total power per cycle_cc2
-avg_total_power_cycle_cc2       5.0660 # average total power per cycle_cc2
-avg_total_power_insn_cc2       7.4022 # average total power per insn_cc2
-avg_alu_power_cc3            1.0859 # avg power usage of alu_cc3
-total_power_cycle_cc3   669624.3320 # total power per cycle_cc3
-avg_total_power_cycle_cc3       9.3956 # average total power per cycle_cc3
-avg_total_power_insn_cc3      13.7283 # average total power per insn_cc3
+avg_alu_power_cc2            0.8898 # avg power usage of alu_cc2
+total_power_cycle_cc2   369278.4902 # total power per cycle_cc2
+avg_total_power_cycle_cc2       6.5699 # average total power per cycle_cc2
+avg_total_power_insn_cc2       7.8501 # average total power per insn_cc2
+avg_alu_power_cc3            4.3540 # avg power usage of alu_cc3
+total_power_cycle_cc3   906165.1815 # total power per cycle_cc3
+avg_total_power_cycle_cc3      16.1216 # average total power per cycle_cc3
+avg_total_power_insn_cc3      19.2633 # average total power per insn_cc3
root@dd1dd574f1dc:~/build/prueba/wattchg7/pruebas#
```

Estas diferencias muestran un aumento significativo en el consumo de energía y la potencia de las ALUs cuando se incrementa el número de recursos de 1 a 8. Específicamente, el consumo total de energía se duplica, y la potencia de las ALUs aumenta en varios órdenes de magnitud. Estos datos sugieren que, al aumentar el número de recursos disponibles para cada tipo de unidad funcional, el consumo de energía y la potencia de las ALUs también aumentan de manera significativa, es evidente que hay un costo energético asociado con el aumento de recursos.

2- Vuelva a la carpeta de trabajo del taller anterior y ejecute el sim-profile en test con la bandera -iprof

```
johnnyzaet@Kali: ~/Desktop/ez-install-simplescalar-op1
root@dd1dd574f1dc:~/build/prueba# ../simplesim-3.0/sim-profile -iprof test
sim-profile: SimpleScalar/PISA Tool Set version 3.0 of August, 2003.
Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
All Rights Reserved. This version of SimpleScalar is licensed for academic
non-commercial use.  No portion of this work may be used by any commercial
entity, or for any commercial purpose, without the prior written permission
of SimpleScalar, LLC (info@simplescalar.com).

sim: command line: ../simplesim-3.0/sim-profile -iprof test

sim: simulation started @ Sun Oct 15 20:17:12 2023, options follow:

sim-profile: This simulator implements a functional simulator with
profiling support.  Run with the `-h' flag to see profiling options
available.

# -config                     # load configuration from a file
# -dumpconfig                 # dump configuration to a file
# -h                    false # print help message
# -v                    false # verbose operation
# -d                    false # enable debug message
# -i                    false # start in Dlite debugger
-seed                       1 # random number generator seed (0 for timer seed)
# -q                    false # initialize and terminate immediately
# -chkpt               <null> # restore EIO trace execution from <fname>
# -redir:sim           <null> # redirect simulator output to file (non-interactive only)
# -redir:prog          <null> # redirect simulated program output to file
-nice                       0 # simulator scheduling priority
-max:inst                   0 # maximum number of inst's to execute
-all                  false # enable all profile options
-iclass               false # enable instruction class profiling
-iprof                 true # enable instruction profiling
-brprof               false # enable branch instruction profiling
-amprof               false # enable address mode profiling
-segprof              false # enable load/store address segment profiling
-tsymprof             false # enable text symbol profiling
-taddrprof            false # enable text address profiling
-dsymprof             false # enable data symbol profiling
-internal             false # include compiler-internal symbols during symbol profiling
# -pcstat              <null> # profile stat(s) against text addr's (mult uses ok)

sim: ** starting functional simulation **
q=4 (int)x=12 (int)y=29
z=144
z=841
z=13
z=13
l=6
l=36
*lp=216
z=144.000000
q=4 x=12.000000 (int)x=12 y=29.000000 (int)y=29
q = 16  x = 11.700001  y = 23.400000

sim: ** simulation statistics **
sim_num_insn                52365 # total number of instructions executed
sim_num_refs                15894 # total number of loads and stores executed
sim_elapsed_time                1 # total simulation time in seconds
sim_inst_rate            52365.0000 # simulation speed (in insts/sec)

sim_inst_prof          # instruction profile
sim_inst_prof.array_size = 119
sim_inst_prof.bucket_size = 1
```

Al combinar el perfil de instrucciones con un análisis de potencia, se obtiene una visión completa de cómo el código afecta el consumo de energía. Esta información es invaluable para optimizar tanto el software como el hardware en términos de rendimiento y eficiencia energética.

3- Utilizando el simulador Wattch, realice los casos que ejecutó en el taller anterior y contraste los resultados con la teoría.

```
johnnyzaet@Kali: ~/Desktop/ez-install-simplescalar-op1

5b324396f:~/build/prueba# clear
5b324396f:~/build/prueba# ./wattchg7/sim-outorder -issue:inorder -res:fpalu 1 -res:imult 1 -res:ialu 1 -res:fpmult 1
mport 2 test1 &> sim_inorder_wattch_1.txt
5b324396f:~/build/prueba# ./wattchg7/sim-outorder -issue:inorder -res:fpalu 4 -res:imult 1 -res:ialu 4 -res:fpmult 1
mport 2 test1 &> sim_inorder_wattch_2.txt
5b324396f:~/build/prueba# ./wattchg7/sim-outorder -res:fpalu 1 -res:imult 1 -res:ialu 1 -res:fpmult 1 -res:memport 2
> sim_outorder_wattch_1.txt
5b324396f:~/build/prueba# ./wattchg7/sim-outorder -res:fpalu 4 -res:imult 1 -res:ialu 4 -res:fpmult 1 -res:memport 2
> sim_outorder_wattch_2.txt
5b324396f:~/build/prueba# diff -u sim_inorder_wattch_1.txt sim_inorder_wattch_2.txt &> diff_inorder_wattch.txt
5b324396f:~/build/prueba# diff -u sim_outorder_wattch_1.txt sim_outorder_wattch_2.txt &> diff_outorder_wattch.txt
5b324396f:~/build/prueba#
```

A pesar de que teóricamente la ejecución fuera de orden debería ofrecer un mejor rendimiento que la ejecución en orden, en los resultados que analizamos, las diferencias son mínimas. Esto puede deberse a que el programa de prueba no tiene suficientes instrucciones que se beneficien significativamente de la ejecución fuera de orden.
Aumentar las unidades de fpalu y ialu tuvo un impacto mínimo en el rendimiento y la potencia, lo que indica que el programa de prueba puede no estar utilizando intensivamente estas unidades o que ya estaba saturado con una unidad.

4- Utilice el simulador sim-cache

```
sim: command line: ../simplesim-3.0/sim-cache test

sim: simulation started @ Mon Oct 16 04:49:15 2023, options follow:

sim-cache: This simulator implements a functional cache simulator.  Cache
statistics are generated for a user-selected cache and TLB configuration,
which may include up to two levels of instruction and data cache (with any
levels unified), and one level of instruction and data TLBs.  No timing
information is generated.

# -config                     # load configuration from a file
# -dumpconfig                 # dump configuration to a file
# -h                    false # print help message
# -v                    false # verbose operation
# -d                    false # enable debug message
# -i                    false # start in Dlite debugger
-seed                       1 # random number generator seed (0 for timer seed)
# -q                    false # initialize and terminate immediately
# -chkpt                <null> # restore EIO trace execution from <fname>
# -redir:sim            <null> # redirect simulator output to file (non-interactive only)
# -redir:prog           <null> # redirect simulated program output to file
-nice                       0 # simulator scheduling priority
-max:inst                   0 # maximum number of inst's to execute
-cache:dl1       dl1:256:32:1:l # l1 data cache config, i.e., {<config>|none}
-cache:dl2       ul2:1024:64:4:l # l2 data cache config, i.e., {<config>|none}
-cache:il1       il1:256:32:1:l # l1 inst cache config, i.e., {<config>|dl1|dl2|none}
-cache:il2               dl2 # l2 instruction cache config, i.e., {<config>|dl2|none}
-tlb:itlb        itlb:16:4096:4:l # instruction TLB config, i.e., {<config>|none}
-tlb:dtlb        dtlb:32:4096:4:l # data TLB config, i.e., {<config>|none}
-flush                 false # flush caches on system calls
```

¿Cuál es el numeró de desaciertos en las caches L1 y L2?

Cache L1 de datos (dl1): 663 desaciertos
Cache L2 unificada (ul2): 837 desaciertos

5- Modifique los parámetros de la jerarquía de memoria (dimensiones de cache, políticas de reemplazo, etc) y observe como afectan los resultados obtenidos con respecto a la configuración por defecto.

```
root@6105b324396f:~/build/prueba# ../simplesim-3.0/sim-cache -cache:dl1 dl1:512:32:1:l test &> sim_cache_dl1_size.txt

root@6105b324396f:~/build/prueba# ../simplesim-3.0/sim-cache -cache:dl1 dl1:256:32:2:l test &> sim_cache_dl1_assoc.tx
t
root@6105b324396f:~/build/prueba# ../simplesim-3.0/sim-cache -cache:dl1 dl1:256:32:1:f test &> sim_cache_dl1_fifo.txt
root@6105b324396f:~/build/prueba# ../simplesim-3.0/sim-cache -cache:dl2 ul2:2048:64:4:l test &> sim_cache_ul2_size.tx
t
root@6105b324396f:~/build/prueba# ../simplesim-3.0/sim-cache -cache:dl2 ul2:1024:64:8:l test &> sim_cache_ul2_assoc.t
xt
root@6105b324396f:~/build/prueba# ls
diff_inorder_wattch.txt   sim_cache_dl1_assoc.txt  sim_cache_ul2_size.txt     sim_outorder_wattch_2.txt  test1
diff_outorder_wattch.txt  sim_cache_dl1_fifo.txt   sim_inorder_wattch_1.txt   test                       wattchg7
dumpfile.txt              sim_cache_dl1_size.txt   sim_inorder_wattch_2.txt   test-fmath.c
sim_cache.txt             sim_cache_ul2_assoc.txt  sim_outorder_wattch_1.txt  test-llong.c
root@6105b324396f:~/build/prueba#
```

En todos los archivos, los resultados de desaciertos y tasas de desacierto para las caches L1 y L2 parecen ser consistentes. Esto es sorprendente, ya que se esperaría que al modificar diferentes características de las memorias, habría algún cambio en el rendimiento. Es importante recordar que el rendimiento de la cache puede variar dependiendo de la naturaleza del programa que se está ejecutando. Por lo tanto, aunque estas modificaciones no mostraron cambios en este caso particular, podrían tener un impacto en otros escenarios o programas

Tamaño de Cache L1 (sim_cache_dl1_size.txt): Al aumentar el tamaño de la cache L1, se esperaría que se redujera el número de desaciertos, ya que hay más espacio para almacenar datos. Sin embargo, si el programa accede a un conjunto de datos relativamente pequeño que ya cabe en la cache original, el aumento del tamaño podría no tener un impacto significativo.

Asociatividad de Cache L1 (sim_cache_dl1_assoc.txt): Al aumentar la asociatividad, se espera mejorar la flexibilidad en la ubicación de los bloques en la cache. Esto puede reducir los desaciertos en ciertos escenarios, pero nuevamente, si el patrón de acceso del programa no se beneficia de la mayor asociatividad, es posible que no se observe un cambio significativo.

Política de Reemplazo FIFO (sim_cache_dl1_fifo.txt): Cambiar la política de reemplazo puede afectar cómo se seleccionan los bloques para ser reemplazados en la cache. FIFO puede no ser la política óptima para todos los patrones de acceso, y esto podría explicar por qué no se observan mejoras significativas.

Tamaño y Asociatividad de Cache L2: Las modificaciones en la cache L2 también pueden no mostrar cambios significativos si la mayoría de los accesos son satisfechos por la cache L1 o si el patrón de acceso no se beneficia de las modificaciones realizadas.