ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2014-2015, 8ο εξάμηνο, Σχολή ΗΜ&ΜΥ

1η ΕΡΓΑΣΙΑ

Τελική Ημερομηνία Παράδοσης: 26/04/2015

1. Σκοπός της άσκησης

Σκοπό της άσκησης αποτελεί η εξοικείωση με το περιβάλλον προσομοίωσης του "*Sniper Multicore Simulator*", καθώς και η μελέτη της επίδρασης διαφόρων παραμέτρων της ιεραρχίας της μνήμης στην απόδοση των διαφόρων εφαρμογών. Περισσότερο υλικό (παρουσιάσεις, κώδικα, manual κτλ) για τον προσομοιωτή μπορείτε να βρείτε εδώ:

http://snipersim.org/w/The_Sniper_Multi-Core_Simulator

2. Χρήση του προσομοιωτή στις εικονικές μηχανές του ~okeanos

2.1 Εγγραφή στην υπηρεσία του ~okeanos

Για να χρησιμοποιήσετε τον προσομοιωτή σας προτείνουμε να χρησιμοποιήσετε τις υπηρεσίες του ~okeanos. Μετά την εγγραφή σας μπορείτε να γίνετε μέλη του project **advcomparch15.cslab.ece.ntua.gr** *(Dashboard → Projects → join a project)*. Η συμμετοχή στο project δεν είναι υποχρεωτική αλλά σας εξασφαλίζει μέχρι 2 extra VMs για τους σκοπούς του μαθήματος.

Αντίστοιχα, δεν είναι υποχρεωτική και η εγγραφή στο ~okeanos, καθώς μπορείτε να κατεβάσετε και να εγκαταστήσετε τον προσομοιωτή στο μηχάνημα σας ακολουθώντας τις οδηγίες που βρίσκονται στο site του προσομοιωτή.

2.2 Δημιουργία κατάλληλου VM

Ο προσομοιωτής είναι προεγκατεστημένος στο public image "**Advcomparch@CSLab 2014-2015**". Για να δημιουργήσετε το κατάλληλο VM, πρέπει να ακολουθήσετε την παρακάτω διαδικασία:

1. New Machine +
2. Image type : Public
3. Επιλογή του image "Advcomparch@CSLab 2014-2015" (confirm → next)
4. Επιλογή των χαρακτηριστικών του μηχανήματος. Αν είστε μέλη του project έχετε πρόσβαση σε 8 CPUs, 8GB RAM και 20GB disk τα οποία μπορείτε να μοιράσετε το πολύ σε 2 VMs.

Μόλις ολοκληρωθεί η δημιουργία του VM μπορείτε να συνδεθείτε σε αυτό είτε με ssh είτε από γραφικό περιβάλλον. Περισσότερες λεπτομέρειες μπορείτε να βρείτε εδώ:

https://okeanos.grnet.gr/support/user-guide/cyclades-how-do-i-connect-to-a-vm/

2.3 Χρήση προσομοιωτή

Στο home directory του VM σας (/home/user) υπάρχει ο φάκελος **advcomparch**:

```
user@snf-xxx:~/advcomparch$ ll
total 20
drwxrwxr-x  5 user user 4096 May 27 14:44 ./
drwxr-xr-x 23 user user 4096 May 28 16:30 ../
drwxr-xr-x  9 user user 4096 May 27 14:53 benchmarks/
drwxr-xr-x  2 user user 4096 May 27 14:53 script_templates/
drwxr-xr-x 16 user user 4096 May 27 15:18 sniper/
```

Στο φάκελο *sniper* βρίσκεται ο προσομοιωτής (source, scripts, tools κτλ), στο φάκελο *script_templates* υπάρχουν παραδείγματα στα οποία μπορείτε να στηριχτείτε προκειμένου να δημιουργήσετε κατάλληλα scripts για τις ανάγκες της άσκησης, ενώ στο φάκελο *benchmarks* βρίσκονται τα μετροπρογράμματα που θα χρησιμοποιήσετε για τις ασκήσεις. Μαζί με αυτά υπάρχει και το script **run-sniper**, το οποίο χρησιμοποιείται για την προσομοίωση και την εκτέλεση των benchmarks.

```
user@snf-xxx:~/advcomparch/benchmarks$ ./run-sniper
Run benchmark under the Sniper simulator
Usage:
  ./run-sniper  -p <program>  -i <inputsize (test)> -n <ncores (1)>  -m <machines (1)>
-d <outputdir (.)>  -c <config-file>  -r <sniper-root-dir>  -g <options>
Benchmarks:
  splash2:
    splash2-barnes splash2-cholesky splash2-fft splash2-fft_O0 splash2-fft_O1 splash2-
fft_O2 splash2-fft_O3 splash2-fft_forever splash2-fft_rep2 splash2-fmm splash2-lu.cont
splash2-lu.ncont splash2-ocean.cont splash2-ocean.ncont splash2-radiosity splash2-radix
splash2-raytrace  splash2-raytrace_opt  splash2-volrend  splash2-water.nsq  splash2-
water.sp splash2-barnes-scale splash2-fft-scale splash2-fmm-scale splash2-lu.cont-scale
splash2-lu.ncont-scale  splash2-ocean.cont-scale  splash2-radix-scale  splash2-water.nsq-
scale
  parsec:
    parsec-blackscholes  parsec-bodytrack  parsec-canneal  parsec-dedup  parsec-facesim
parsec-ferret parsec-fluidanimate parsec-freqmine parsec-raytrace parsec-streamcluster
parsec-swaptions parsec-vips parsec-x264
  local:
    local-pi
```

Από τις παραμέτρους που δέχεται αυτό το script μας ενδιαφέρουν κυρίως οι εξής:

- -p <program> : το benchmark που προσομοιώνουμε
- -i <inputsize> : το μέγεθος του input για το benchmark
- -n <ncores> : ο αριθμός των cores που προσομοιώνουμε
- -d <outputdir> : ο φάκελος στον οποίο θα αποθηκευτούν τα στατιστικά της προσομοίωσης
- -c <config-file> : το αρχείο με τις παραμέτρους του συστήματος που προσομοιώνουμε
- -g <options> : ορισμός παραμέτρων της προσομοίωσης
- --viz : ενεργοποιεί τα visualizations για τα αποτελέσματα της προσομοίωσης

2.3.1 Παραμετροποίηση συστήματος

Η παραμετροποίηση των προσομοιωμένων συστημάτων γίνεται με τη χρήση των config files, τα οποία βρίσκονται στο φάκελο **advcomparch/sniper/config**. Εκεί βρίσκονται τα configurations για διάφορους επεξεργαστές, καθώς και αυτό που θα χρησιμοποιήσετε για τους σκοπούς της άσκησης, το **advcomp.cfg**.

Μέσα στο config file ορίζονται διάφορες παράμετροι του επεξεργαστή όπως η συχνότητα, το issue width, ο branch predictor και η ιεραρχία μνήμης. Για παράδειγμα, το παρακάτω κομμάτι ορίζει την L1 data cache:

```
[perf_model/l1_dcache]
perfect = false
cache_block_size = 64
cache_size = 32
associativity = 8
address_hash = mask
replacement_policy = lru
data_access_time = 4
tags_access_time = 1
perf_model_type = parallel
writethrough = 0
shared_cores = 1
```

Συγκεκριμένα, ορίζει μια L1 data cache μεγέθους 32KB, 8-way set associative, με block μεγέθους 64 bytes, η οποία είναι write-back και χρησιμοποιεί πολιτική αντικατάστασης LRU. Όσες παράμετροι δεν ορίζονται στο αρχείο αυτό, ορίζονται στο **base.cfg**, το οποίο χρησιμοποιείται σε κάθε προσομοίωση από το run-sniper. Επίσης, μπορείτε να ορίσετε παραμέτρους της προσομοίωσης μέσω της γραμμής εντολών με τη χρήση της επιλογής *–g* του run-sniper. Για παράδειγμα το παρακάτω ορίζει το μέγεθος του block size της L1 data cache:

```
./run-sniper <other options> -g --perf_model/l1_dcache/cache_block_size=32
```

Σε κάθε εκτέλεση μιας προσομοίωσης δημιουργείται στο φάκελο των αποτελεσμάτων το αρχείο **sim.cfg**, το οποίο περιέχει αναλυτικά όλες τις παραμέτρους και τις τιμές που χρησιμοποιήθηκαν.

2.3.2 Στατιστικά Προσομοίωσης

Κάθε προσομοίωση αποθηκεύει τα στατιστικά που παρακολουθεί στον φάκελο των αποτελεσμάτων. Συγκεκριμένα δημιουργεί 2 αρχεία τα **sim.out** και **sim.stats.sqlite3**. Το πρώτο περιέχει μια περίληψη των πιο σημαντικών στατιστικών και αφορά ολόκληρη την προσομοίωση. Για παράδειγμα:

```
user@snf-xxx:~/advcomparch/benchmarks/parsec-blackscholes$ cat sim.out
                            | Core 0
  Instructions              |  226192675
  Cycles                    |  188677065
  Time                      |   70931228
Branch predictor stats      |
  num correct               |   11209611
  num incorrect             |     398124
  misprediction rate        |       3.43%
  mpki                      |       1.76
Cache Summary               |
  Cache L1-I                |
    num cache accesses      |   25511080
    num cache misses        |        180
    miss rate               |      0.00%
    mpki                    |       0.00
  Cache L1-D                |
    num cache accesses      |   63675354
    num cache misses        |     236626
    miss rate               |       0.37%
    mpki                    |       1.05
  Cache L2                  |
    num cache accesses      |     236834
    num cache misses        |        707
    miss rate               |       0.30%
    mpki                    |       0.00
```

```
DRAM summary                    |
  num dram accesses             |        708
  average dram access latency   |      60.29
  average dram queueing delay   |       6.86
```

Το δεύτερο αρχείο περιλαμβάνει πιο αναλυτικά στατιστικά, τα οποία μπορείτε να προσπελάσετε χρησιμοποιώντας το script **dumpstats.py**.

```
user@snf-xxx:~/advcomparch/benchmarks/parsec-blackscholes$ dumpstats.py -h
Usage: /home/user/advcomparch/sniper/tools/dumpstats.py [-h (help)] [-l|--list | -t|--
topology | -m|--markers] [--partial <section-start>:<section-end> (default: roi-
begin:roi-end)]  [-d <resultsdir (default: .)>]
```

Ο sniper έχει τη δυνατότητα να αποθηκεύει τα στατιστικά όχι μόνο για ολόκληρη την προσομοίωση, αλλά και σε συγκεκριμένα διαστήματα (π.χ. κάθε 10000 εντολές ή κύκλους). Τα στατιστικά αυτά αποθηκεύονται επίσης στο sim.stats.sqlite3 και μπορείτε να τα προσπελάσετε χρησιμοποιώντας το script **advcomparch_dumpstats.py**.

```
user@snf-xxx:~/advcomparch/benchmarks/parsec_blackscholes$ advcomparch_dumpstats.py -h
Usage: /home/user/advcomparch/sniper/tools/advcomparch_dumpstats.py [-h (help) |-l
(list stat points)] [--statsWindow=<stat_interval> (print stats for intervals>=
stat_interval, 0 for whole run]
--core : stats for core
--L1I  : stats for L1-Icache
--L1D  : stats for L1-Dcache
--L2   : stats for L2 cache
```

2.3.3 Μετροπρογράμματα

Ο sniper μπορεί να χρησιμοποιηθεί για την εκτέλεση πολλών και διαφορετικών benchmarks. Στο image που χρησιμοποιήσατε για τη δημιουργία του VM σας έχουμε ήδη εγκαταστήσει τα **splash** και **parsec** benchmarks όπως αυτά δίνονται στο site του simulator. Περισσότερες πληροφορίες για τα benchmarks αυτά μπορείτε να βρείτε εδώ:

http://www-flash.stanford.edu/apps/SPLASH/
http://www.capsl.udel.edu/splash/index.html
http://parsec.cs.princeton.edu/

Τρέχοντας το script run-sniper μπορείτε να δείτε αναλυτικά τα benchmarks τα οποία μπορεί να εκτελέσει ο προσομοιωτής. Σε αυτή την άσκηση θα χρησιμοποιήσετε benchmarks από τα parsec. Ο κώδικας τους έχει τροποποιηθεί κατάλληλα, ώστε να μη χρειάζεται να προσομοιωθεί αναλυτικά ολόκληρη η εφαρμογή. Πιο συγκεκριμένα, έχουν οριστεί οι περιοχές του κώδικα που μας ενδιαφέρουν (Regions Of Interest, ROI) και προσομοιώνονται με ακρίβεια ενώ στις υπόλοιπες η προσομοίωση είναι μη ακριβής αλλά πολύ γρήγορη.

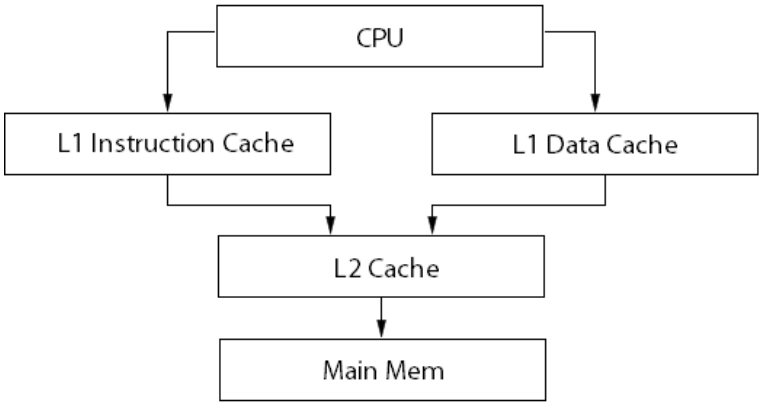
Πιο συγκεκριμένα, στην άσκηση αυτή θα χρησιμοποιήσετε τα παρακάτω benchmarks input size **small**:

1. parsec-blackscholes
2. parsec-canneal
3. parsec-dedup
4. parsec-fluidanimate
5. parsec-freqmine
6. parsec-streamcluster
7. parsec-swaptions
8. parsec-vips

3. Προσομοίωση Ιεραρχίας Μνήμης

3.1 Ορισμός της ιεραρχίας μνήμης

Η ιεραρχία κρυφής μνήμης που θα χρησιμοποιήσουμε στα πλαίσια αυτής της άσκησης απεικονίζεται στο παρακάτω σχήμα:



Για κάθε προσομοίωση θα πρέπει να μεταβάλετε ανάλογα τις παραμέτρους της ιεραρχίας μνήμης με βάση τα ζητούμενα της άσκησης.

3.2 Διαδικασία συλλογής στατιστικών

Στην άσκηση αυτή θα χρησιμοποιήσουμε τα parsec benchmarks με μέγεθος εισόδου small. Προκειμένου να αποκτήσετε τα απαραίτητα στατιστικά, θα πρέπει να ορίσετε στην προσομοίωση να τα αποθηκεύει κάθε 10M εντολές. Αυτό επιτυγχάνεται με τη χρήση του script periodicins-stats, το οποίο δίνεται σαν όρισμα στο run-sniper. Παράδειγμα χρήσης μπορείτε να βρείτε στο Παράρτημα Β, καθώς και στο φάκελο script_templates.

4. Πειραματική Αξιολόγηση

Στα πλαίσια της εργασίας αυτής, θα διερευνηθεί αρχικά η επίδραση των βασικότερων παραμέτρων ιεραρχίας κρυφής μνήμης στην απόδοση της εφαρμογής. Σε δεύτερη φάση, θα διερευνηθεί για μία συγκεκριμένη παραμετροποίηση της ιεραρχίας μνήμης ο τρόπος που μεταβάλλονται διάφορες μετρικές απόδοσης στο χρόνο. Τα πειράματα που θα χρειαστεί να εκτελέσετε παρουσιάζονται στη συνέχεια.

4.1 Μελέτη επίδρασης παραμέτρων ιεραρχίας μνήμης στην απόδοση της εφαρμογής

4.1.1 L1 cache

Για όλες τις περιπτώσεις που εξετάζονται στο πείραμα αυτό, το block size της L2 θα πρέπει να είναι ίσο με το block size της L1 ενώ οι υπόλοιπες παράμετροι της θα διατηρηθούν σταθερές και ίσες με τις default παραμέτρους (όπως δίνονται στο Παράρτημα Α).

- Για τις L1 instruction και data caches θα χρησιμοποιήσετε το configuration που δίνεται στο Παράρτημα Α, αλλάζοντας κατάλληλα το associativity και το μέγεθος ώστε να εκτελέσετε τα benchmarks για τις παρακάτω περιπτώσεις:

L1 size	L1 associativity
16K	4
32K	4
32K	8
64K	4
64K	8
128K	8

- Εκτελέστε τα benchmarks για τους παρακάτω συνδυασμούς size και associativity για cache_block_size 32 και 128.

L1 size	L1 associativity
16K	4
32K	4
64K	4

4.1.2 L2 cache

Για όλες τις περιπτώσεις που θα εξεταστούν εδώ θα χρησιμοποιήσετε L1 instruction και data caches όπως δίνονται στο Παράρτημα Α, ορίζοντας τις κατάλληλες παραμέτρους ώστε να είναι 8-way associative, 32KB. Το block size τους θα πρέπει να είναι ίσο με το block size της L2 cache.

- Για την L2 θα χρησιμοποιήσετε το configuration που δίνεται στο Παράρτημα Α, αλλάζοντας κατάλληλα το associativity και το μέγεθος ώστε να εκτελέστε τα benchmarks για τις παρακάτω περιπτώσεις:

L2 size	L2 associativity
256K	4
512K	4
512K	8
1024K	8
1024K	16
2048K	8
2048K	16

Το block size στις παραπάνω περιπτώσεις θα είναι ίσο με 128.

- Εκτελέστε τα benchmarks για τους παρακάτω συνδυασμούς size και associativity για cache_block_size 64 και 256.

L2 size	L2 associativity
512K	8
1024K	8
2048K	8

4.1.3 Ζητούμενο

Σαν βασική μετρική απόδοσης θα χρησιμοποιήσετε το IPC (Instructions Per Cycle). Με την προϋπόθεση ότι ο κύκλος μηχανής και ο εκτελούμενος αριθμός εντολών παραμένουν σταθεροί κάθε φορά, μεγαλύτερες τιμές στο IPC υποδεικνύουν καλύτερη απόδοση (*σημείωση: αυτό ισχύει μόνο στα πλαίσια της προσομοίωσης. Στην πράξη, οι διάφορες τροποποιήσεις στα μικροαρχιτεκτονικά χαρακτηριστικά του επεξεργαστή επιφέρουν συνήθως αλλαγές και στην διάρκεια του κύκλου ρολογιού*).

Για κάθε μια από τις παραπάνω περιπτώσεις μελετήστε τις μεταβολές στο IPC και στο miss rate της cache της οποίας τις παραμέτρους μεταβάλλετε. Παρουσιάστε σε γραφικές παραστάσεις τις μεταβολές αυτές για κάθε περίπτωση. Συνοψίστε τα συμπεράσματα των προηγούμενων ερωτημάτων. Ποιες από τις παραμέτρους που εξετάσατε έχουν τη μεγαλύτερη επίδραση στην απόδοση; ***Χρησιμοποιήστε τα στατιστικά για ολόκληρη την προσομοίωση (από την αρχή ως το τέλος).***

4.2 Μελέτη μεταβολής μετρικών απόδοσης στο χρόνο

Στα προηγούμενα ερωτήματα εξετάσατε τη συμπεριφορά της απόδοσης κάθε εφαρμογής για ολόκληρη την περιοχή ενδιαφέροντος όπως αυτή έχει οριστεί στον κώδικα της κάθε εφαρμογής. Στο ερώτημα αυτό καλείστε να μελετήσετε τη δυναμική συμπεριφορά των εφαρμογών, εξετάζοντας τον τρόπο που οι διάφορες μετρικές απόδοσης μεταβάλλονται στο χρόνο.

Οι προσομοιώσεις που εκτελέσατε για τα προηγούμενα ερωτήματα αποθήκευαν στατιστικά κάθε 10M εντολές. Διαλέξτε ένα configuration (π.χ. L2 8-way, 2048KB, block size 128) και μελετήστε τη δυναμική συμπεριφορά των μετρικών απόδοσης (IPC, miss rates) για τα benchmarks. Για κάθε μία μετρική απόδοσης χρησιμοποιήστε ένα διάγραμμα που θα έχει στον x άξονα το χρόνο (στην ουσία τον αριθμό των εντολών στο κάθε σημείο που αποθηκεύτηκαν αποτελέσματα) και στον y την τιμή της μετρικής, προκειμένου να δείξετε τη δυναμική μεταβολή της συγκεκριμένης μετρικής στο χρόνο.

Τι συμπεράσματα βγάζετε; Κατά πόσο τα αποτελέσματα που πήρατε στο ερώτημα 4.1 για την ίδια παραμετροποίηση της ιεραρχίας μνήμης, ανταποκρίνονται στην εκτέλεση των πρώτων 10M ή των πρώτων 100M εντολών; Αντιπροσωπεύουν οι πιο «σύντομες» αυτές προσομοιώσεις τη δυναμική συμπεριφορά κάθε εφαρμογής;

Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (pdf, doc ή odt). Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ). Η άσκηση να παραδοθεί ηλεκτρονικά στην ιστοσελίδα:
http://www.cslab.ece.ntua.gr/courses/advcomparch/submit

Δουλέψτε ατομικά. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην την αντιγράψετε από άλλους συμφοιτητές σας.

Μην αφήσετε την εργασία για το τελευταίο Σαββατοκύριακο, απαιτεί αρκετό χρόνο για την εκτέλεση όλων των προσοιμοιώσεων, ξεκινήστε αμέσως!

Παράρτημα Α: Αρχικοί παράμετροι ιεραρχίας μνήμης (~/advcomparch/sniper/config/advcomp.cfg)

```
# Κομμάτι του αρχείου advcomp.cfg

[perf_model/cache]
levels = 2

[perf_model/l1_icache]
perfect = false
cache_block_size = 64
cache_size = 32
associativity = 8
address_hash = mask
replacement_policy = lru
data_access_time = 4
tags_access_time = 1
perf_model_type = parallel
writethrough = 0
shared_cores = 1

[perf_model/l1_dcache]
perfect = false
cache_block_size = 64
cache_size = 32
associativity = 8
address_hash = mask
replacement_policy = lru
data_access_time = 4
tags_access_time = 1
perf_model_type = parallel
writethrough = 0
shared_cores = 1

[perf_model/l2_cache]
perfect = false
cache_block_size = 64
cache_size = 1024
associativity = 8
address_hash = mask
replacement_policy = lru
data_access_time = 8
tags_access_time = 3
writeback_time = 50 # L3 hit time will be added
perf_model_type = parallel
writethrough = 0
shared_cores = 1

[perf_model/dram_directory]
# total_entries = number of entries per directory controller.
total_entries = 1048576
associativity = 16
directory_type = full_map
home_lookup_param = 12
```

Παράρτημα Β: ~/advcomparch/script_templates/run_l1.sh

```
#!/bin/bash

BENCH="parsec-blackscholes"
INPUT_SIZE="test"
INS=10000000
CONF_FILE=${GRAPHITE_ROOT}/config/advcomp.cfg

for cache in 16_4_64 32_4_64; do

      ## Get parameters
      size=$(echo $cache | cut -d'_' -f1)
      assoc=$(echo $cache | cut -d'_' -f2)
      bsize=$(echo $cache | cut -d'_' -f3)

      ## Run benchmark.
      ## Note: the block size must be the same for L1-data, L1-instruction and L2
      OUTDIR="./${BENCH}-L1-${size}_${assoc}_${bsize}.sim"
      $BENCHMARKS_ROOT/run-sniper -p ${BENCH} -i ${INPUT_SIZE} -n 1 -d ${OUTDIR} \
        -c ${CONF_FILE} -s periodicins-stats:$INS \
        -g --perf_model/l1_dcache/cache_size=${size} \
        -g --perf_model/l1_dcache/associativity=${assoc} \
        -g --perf_model/l1_dcache/cache_block_size=${bsize} \
        -g --perf_model/l1_icache/cache_size=${size} \
        -g --perf_model/l1_icache/associativity=${assoc} \
        -g --perf_model/l1_icache/cache_block_size=${bsize} \
        -g --perf_model/l2_cache/cache_block_size=${bsize}

done
```