1η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2024-2025, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τελική Ημερομηνία Παράδοσης: **27/04/2025**

1. Εισαγωγή

Στα πλαίσια της παρούσας άσκησης θα χρησιμοποιήσετε το εργαλείο **PIN** για να μελετήσετε την επίδραση διαφορετικών συστημάτων πρόβλεψης εντολών άλματος καθώς και η αξιολόγηση τους με δεδομένο το διαθέσιμο χώρο πάνω στο τσιπ.

2. Το εργαλείο PIN

Το PIN είναι ένα εργαλείο το οποίο αναπτύσσεται και συντηρείται από την Intel και χρησιμοποιείται για την ανάλυση εφαρμογών. Χρησιμοποιείται για dynamic binary instrumentation, δηλαδή για την εισαγωγή κώδικα δυναμικά (την στιγμή που εκτελείται η εφαρμογή) ανάμεσα στις εντολές της εφαρμογής με σκοπό την συλλογή πληροφοριών σχετικά με την εκτέλεση (π.χ. cache misses, total instructions κλπ.).

Περισσότερες πληροφορίες σχετικά με το PIN καθώς και εγχειρίδια χρήσης μπορείτε να βρείτε εδώ:

https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool

2.1. Λήψη και εγκατάσταση του PIN

Για να εγκαταστήσετε το PIN στο σύστημα σας κατεβάστε το από το παρακάτω link:

https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-binary-instrumentation-tool-downloads.html

Το PIN εξαρτάται άμεσα από τον πυρήνα του λειτουργικού συστήματος, οπότε υπάρχει πιθανότητα κάποιες εκδόσεις του PIN να έχουν ασυμβατότητα με συγκεκριμένες εκδόσεις του πυρήνα Linux. Τα βήματα που παρουσιάζονται παρακάτω για την εκτέλεση του PIN έχουν δοκιμαστεί επιτυχώς χρησιμοποιώντας την πιο πρόσφατη του έκδοση (3.31 – Ιούνιος 2024) σε Ubuntu 22.04.5 LTS (με έκδοση πυρήνα 6.8).

Αφού ολοκληρωθεί η λήψη του αρχείου θα πρέπει να το αποσυμπιέσετε δίνοντας σε ένα τερματικό την παρακάτω εντολή:

```
$ tar xvfz pin-external-3.31-98869-gfa6f126a8-gcc-linux.tar.gz
```

Τώρα μπορείτε να περιηγηθείτε στα περιεχόμενα του PIN:

```
$ cd pin-external-3.31-98869-gfa6f126a8-gcc-linux
$ ls –aF

./ ../ README doc/ extras/ ia32/ intel64/ licensing/ pin pin.sig source/
```

2.2. Χρήση του PIN

Το **pin** είναι το εκτελέσιμο που θα χρησιμοποιήσετε για την εκτέλεση των εφαρμογών στα πειράματά σας. Για να δείτε τον τρόπο χρήσης του μπορείτε να το εκτελέσετε χωρίς ορίσματα:

```
$ ./pin

E: Missing application name
Pin: pin-3.31-98869-fa6f126a8
Copyright 2002-2024 Intel Corporation.

Usage: pin [OPTION] [-t <tool> [<toolargs>]] -- <command line>
Use -help for a description of options
```

Με το όρισμα **–t** λέτε στο PIN ποιο pintool να χρησιμοποιήσει, ενώ ως **<command line>** δίνετε το εκτελέσιμο το οποίο θα αναλυθεί από το PIN καθώς και τα ορίσματά του. Ένα παράδειγμα εκτέλεσης του pin δίνεται παρακάτω:

```
$ cd source/tools/ManualExamples/
$ make obj-intel64/inscount0.so
$ cd ../../../
$ ./pin –t ./source/tools/ManualExamples/obj-intel64/inscount0.so \
                                –o ls.inscount0.output -- /bin/ls –aF
./  ../  README*  doc/  extras/  ia32/  intel64/  licensing/  ls.inscount0.output
pin*  pin.log  pin.sig*  source/
$ cat ls.inscount0.output
Count 638514
```

Στο παραπάνω παράδειγμα χρησιμοποιήθηκε το pintool **inscount0.so** το οποίο αθροίζει το σύνολο των εντολών που εκτελούνται. Τα pintools είναι προγράμματα γραμμένα σε C++ που χρησιμοποιούνται από το PIN και επικοινωνούν με αυτό μέσω του API του για να κατευθύνουν την ανάλυση των εφαρμογών. Μπορείτε να γράψετε τα δικά σας pintools αλλά υπάρχουν και αρκετά που παρέχονται μαζί με το PIN. Θα τα βρείτε στον φάκελο **source/tools/**. Για να τα μεταγλωττίσετε μπορείτε να εκτελέσετε την εντολή make στον φάκελο που σας ενδιαφέρει.

3. PINTOOL

Στον βοηθητικό κώδικα της άσκησης θα βρείτε τα pintools **cslab_branch_stats.cpp** και **cslab_branch.cpp**. Αφού τροποποιήσετε το PIN_ROOT path στο αρχείο makefile, για την μεταγλώττισή τους δώστε:

```
$ cd advcomparch-ex1-helpcode/pintool
$ make clean; make
```

Το **cslab_branch_stats.cpp** χρησιμοποιείται για την εξαγωγή στατιστικών σχετικά με τις εντολές αλμάτων που εκτελούνται από την εφαρμογή. Ένα παράδειγμα χρήσης δίνεται παρακάτω:

```
$ cd /path/to/advcomparch-ex1-helpcode/spec_execs_train_inputs/429.mcf
$ /path/to/pin-external-3.31-98869-gfa6f126a8-gcc-linux/pin –t \
/path/to/advcomparch-ex1-helpcode/pintool/obj-intel64/cslab_branch_stats.so \
–o my_output.out -- ./mcf_base.gcc49-static inp.in
```

```
$ cat my_output.out
Total Instructions: 18199622978
Branch statistics:
  Total-Branches: 3862655828
  Conditional-Taken-Branches: 1332795713
  Conditional-NotTaken-Branches: 2377823068
  Unconditional-Branches: 97771833
  Calls: 27132609
  Returns: 27132605
```

Το **cslab_branch.cpp** χρησιμοποιείται για την αξιολόγηση τεχνικών πρόβλεψης άλματος ενώ για τις εντολές επιστροφής από διαδικασίες προσομοιώνει διαφορετικά μεγέθη στοίβας διεύθυνσης επιστροφής (RAS). Ένα παράδειγμα χρήσης δίνεται παρακάτω:

```
$ /path/to/pin-external-3.31-98869-gfa6f126a8-gcc-linux/pin \
–t /path/to/advcomparch-ex1-helpcode/pintool/obj-intel64/cslab_branch.so \
–o my_output.out -- ./mcf_base.gcc49-static inp.in
```

Στο αρχείο **branch_predictor.h** ορίζουμε τους διαφορετικούς branch predictors. Για την προσθήκη ενός branch predictor απαιτείται η δημιουργία μίας νέας υπο-κλάσης της κλάσης **BranchPredictor** και ο ορισμός τριών μεθόδων **predict()**, **update()** και **getName()**. Η πρώτη συνάρτηση δέχεται ως ορίσματα το PC της εντολής και τη διεύθυνση προορισμού και καλείται να προβλέψει αν το άλμα θα εκτελεστεί ή όχι (Taken / Not Taken). Η δεύτερη μέθοδος καλείται να αποθηκεύσει τις πληροφορίες εκείνες που απαιτούνται για τις μελλοντικές προβλέψεις. Τα ορίσματα της είναι η πρόβλεψη που έκανε ο predictor, το πραγματικό αποτέλεσμα της εντολής διακλάδωσης, το PC της εντολής και η διεύθυνση προορισμού. Τέλος, η μέθοδος **getName()** χρησιμοποιείται για την εκτύπωση των αποτελεσμάτων του branch predictor στο αρχείο εξόδου του pintool.

4. Μετροπρογράμματα

Το PIN μπορεί να χρησιμοποιηθεί για την εκτέλεση οποιασδήποτε εφαρμογής. Στα πλαίσια της παρούσας άσκησης θα χρησιμοποιήσετε τα μετροπρογράμματα (benchmarks) SPEC_CPU2006. Πιο συγκεκριμένα θα χρησιμοποιήσετε τα παρακάτω 16 benchmarks:

1. 401.bzip2
2. 403.gcc
3. 410.bwaves
4. 416.gamess
5. 429.mcf
6. 433.milc
7. 435.gromacs
8. 436.cactusADM
9. 437.leslie3d
10. 450.soplex
11. 456.hmmer
12. 459.GemsFDTD
13. 464.h264ref
14. 470.lbm
15. 471.omnetpp
16. 483.xalancbmk

Στον βοηθητικό κώδικα της άσκησης σας δίνονται οι φάκελοι **spec_execs_train_inputs** και **spec_execs_ref_inputs**, οι οποίοι περιέχουν τα εκτελέσιμα μαζί με τα απαραίτητα αρχεία εισόδου την εκτέλεση τους. Οι δύο διαφορετικοί φάκελοι (train και ref) περιέχουν διαφορετικές εισόδους για το κάθε μετροπρόγραμμα, με τις ref εισόδους να οδηγούν συνήθως σε μεγαλύτερες χρονικά και πιο αντιπροσωπευτικές εκτελέσεις.

5. Πειραματική Αξιολόγηση

5.1 Εκτέλεση πειραμάτων και μετρικές

Για όλες τις προσομοιώσεις των ερωτημάτων 5.3 – 5.6 θα χρησιμοποιήσετε τα **ref** inputs. Για την αξιολόγηση της επίδοσης ενός predictor θα χρησιμοποιήσετε τη μετρική **direction Mispredictions Per Thousand Instructions** (direction MPKI), υπολογίζοντας και τους κατάλληλους μέσους όρους. Το βιβλίο σας χρησιμοποιεί γενικά το γεωμετρικό μέσο όρο, παρόλα αυτά εδώ και χρόνια εξελίσσεται στο χώρο μια κουβέντα περί του ποιος είναι ο πιο κατάλληλος. Ένα πρόσφατο σχετικό άρθρο μπορείτε να διαβάσετε εδώ.

5.2 Ανάλυση εντολών άλματος

Στο πρώτο κομμάτι της πειραματικής αξιολόγησης ο στόχος είναι η συλλογή στατιστικών για τις εντολές άλματος που εκτελούνται από τα benchmarks. Χρησιμοποιήστε το cslab_branch_stats.cpp και για κάθε benchmark βρείτε τον αριθμό των **εντολών άλματος που εκτελέστηκαν** και το ποσοστό αυτών που ανήκουν σε **κάθε κατηγορία** (conditional-taken, conditional-not taken κλπ.) τόσο για τα **train** όσο και για τα **ref** inputs.

5.3 N-bit predictors

(i) Διατηρώντας σταθερό τον αριθμό των BHT entries και ίσο με 16K, προσομοιώστε τους n-bit predictors, για **N = 1, 2, 3, 4**. Τα n-bits υλοποιούν ένα saturating up-down counter (cslab_branch.cpp) όπως είδαμε στις διαλέξεις. Ποιον predictor θα επιλέγατε και γιατί;

(ii) Για N = 2 **υλοποιήστε** και προσομοιώστε και τα εναλλακτικά FSM που ορίζονται στον πίνακα VI του paper "Optimal 2-Bit Branch Predictors" (R. Nair, 1995). Το FSM στην 1η γραμμή του πίνακα είναι ο saturating counter predictor που είδαμε στο μάθημα, δίνεται υλοποιημένος και προσομοιώσατε προηγουμένως. Ποιον predictor θα επιλέγατε και γιατί;

(iii) Στα προηγούμενα ερωτήματα η αύξηση του αριθμού των bits ισοδυναμεί με αύξηση του απαιτούμενου hardware, αφού ο αριθμός των entries του BHT παραμένει σταθερός. Διατηρώντας τώρα **σταθερό το hardware** και ίσο με **32K bits**, εκτελέστε ξανά τις προσομοιώσεις για όλα τα benchmarks, θέτοντας **N = 1, 2, 4** και τον κατάλληλο αριθμό entries. Για N = 2 προσομοιώστε τόσο τον saturating counter predictor όσο και τα εναλλακτικά FSMs του ερωτήματος (ii). Δώστε το κατάλληλο διάγραμμα και εξηγείστε τις μεταβολές που παρατηρείτε. **Ποιον από τους 7 predictors θα διαλέγατε ως την βέλτιστη επιλογή;**

5.4 Μελέτη του BTB

Υλοποιήστε έναν BTB και μελετήστε την ακρίβεια πρόβλεψής του για τις ακόλουθες περιπτώσεις:

BTB entries	BTB associativity
512	1, 2
256	2, 4
128	2, 4
64	4, 8

Εκτελέστε τις προσομοιώσεις και δώστε όπως και πριν τα κατάλληλα διαγράμματα. Υπενθυμίζεται ότι για τον BTB υπάρχουν 2 περιπτώσεις misses. Η πρώτη είναι direction misprediction και η δεύτερη target misprediction στην περίπτωση ενός direction hit. Πώς θα εξηγούσατε τη διαφορά επίδοσης ανάμεσα στις διαφορετικές περιπτώσεις; **Διαλέξτε την καλύτερη οργάνωση για το BTB.**

5.5 Μελέτη του RAS

Χρησιμοποιώντας την υλοποίηση της RAS (**ras.h**) που σας δίνεται, μελετήστε το ποσοστό αστοχίας για τις ακόλουθες περιπτώσεις:

Αριθμός εγγραφών στη RAS
4
8
16

Αριθμός εγγραφών στη RAS
32
48
64

Προσομοιώστε και δώστε όπως και πριν τα κατάλληλα διαγράμματα εξηγώντας τις μεταβολές που παρατηρείτε. **Επιλέξτε το κατάλληλο μέγεθος για το RAS.**

5.6 Σύγκριση διαφορετικών predictors

Στο κομμάτι αυτό θα συγκρίνετε την επίδοση των παρακάτω predictors (οι predictors σε **bold *δε δίνονται*** και **πρέπει** να τους υλοποιήσετε **εσείς**):

- **Static AlwaysTaken**
- **Static BTFNT (BackwardTaken-ForwardNotTaken)**
- Ο n-bit predictor που επιλέξατε στο 5.2 (iii)
- Pentium-M predictor (δίνεται ότι το hardware overhead είναι περίπου 30K)
- **Local-History two-level predictors** (βλ. διαφάνειες μαθήματος) με τα εξής χαρακτηριστικά :
  - PHT entries = 8192
  - PHT n-bit counter length = 2
  - BHT entries = X
  - BHT entry length = Z

  Υπολογίστε το Z ώστε το απαιτούμενο hardware να είναι σταθερό και συνολικά ίσο με 32K, όταν X=2048, X=4096 και X=8192

- **Global History two-level predictors** με τα εξής χαρακτηριστικά:
  - PHT entries = Z
  - PHT n-bit counter length = X
  - BHR length = 2, 4

  Υπολογίστε το Z ώστε το απαιτούμενο hardware να είναι σταθερό και ίσο με 32K όταν X=2 και X = 4. Το κόστος του Branch History Register (2, 4 και 8 bits) θεωρείται αμελητέο

- **Alpha 21264 predictor** (βλ. διαφάνειες μαθήματος – hardware overhead 29K)
- **Tournament Hybrid predictors** (βλ. διαφάνειες μαθήματος) με τα εξής χαρακτηριστικά:

- Ο meta-predictor M είναι ένας 2-bit predictor με 1024 ή 2048 entries (το overhead του μπορείτε να το αγνοήσετε στην ανάλυση σας)
- Οι P_0, P_1 μπορούν να είναι n-bit, local-history, ή global-history predictors.
- Οι P_0, P_1 έχουν overhead 16K ο καθένας.
- Υλοποιήστε ***τουλάχιστον*** 4 διαφορετικούς tournament predictors.

Προσομοιώστε για τα benchmarks που παρέχονται και συγκρίνετε τους παραπάνω (***τουλάχιστον 16***) predictors. Δώστε τα κατάλληλα διαγράμματα. **Ποιον predictor θα διαλέγατε τελικά να υλοποιήσετε στο σύστημα σας;**

5.7 Ακρίβεια προσομοιώσεων

Στις παραπάνω προσομοιώσεις των predictors χρησιμοποιήσατε τα ref inputs. Αν είχατε χρησιμοποιήσει τα train inputs ο χρόνος των προσομοιώσεων θα είχε μειωθεί κατά πολύ. Θα είχατε όμως καταλήξει στα ίδια συμπεράσματα, τόσο για την επίδοση του κάθε predictor ξεχωριστά όσο και για τη μεταξύ τους σύγκριση;

Προσομοιώστε χρησιμοποιώντας τώρα τα train inputs και συγκρίνετε τα αποτελέσματα σας (μπορείτε να προσομοιώσετε όλους τους predictors του ερωτήματος 5.6 ή κάποιο υποσύνολο, π.χ. n-bit, Pentium-M, Alpha και τον καλύτερο global).

Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (**pdf**, **docx** ή **odt**). *Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ).*

Η άσκηση θα παραδοθεί ηλεκτρονικά στο moodle του μαθήματος:

https://helios.ntua.gr/course/view.php?id=1039

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

Καθώς τα ίδια εργαλεία και μετροπρογράμματα είχαν χρησιμοποιηθεί και τις προηγούμενες χρονιές, είναι εξαιρετικά πιθανό αρκετές (αν όχι όλες οι) απορίες που μπορεί να προκύψουν να έχουν ήδη απαντηθεί. Σας συμβουλεύουμε λοιπόν να ψάξετε για απαντήσεις/βοήθεια στα archives της mailing list του μαθήματος, τα οποία βρίσκονται εδώ:

http://lists.cslab.ece.ntua.gr/pipermail/advcomparch/