ΕΠΛ 605: ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2018
ΕΡΓΑΣΙΑ 2 (21/09/2018)
ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ 4/10/2018

Θα πρέπει να παραδώσετε τυπωμένη την αναφορά σας στο εργαστήριο.

Σας δίνονται 2 benchmarks από το SPEC CPU2017 (https://www.spec.org/cpu2017/) από τα οποία το ένα είναι το ίδιο για όλους και είναι το 502.gcc_r και το δεύτερο έχει ανατεθεί στον κάθε φοιτητή όπως περιγράφετε παρακάτω:

Οδηγίες για το πώς να τρέξετε τα benchmarks :

Gcc ('Ολοι):
cpugcc_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 gcc-pp.c -O3 -finline-limit=0 -fif-conversion -fif-conversion2 -o gcc-pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2.s > gcc-pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2.out

Roms (Andreas Gelasis):
roms_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 < ocean_benchmark2.in.x > ocean_benchmark2.log

Fotonik3d (Jiayi Fu):
fotonik3d_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 > fotonik3d_r.log

Mcf (Christos Othonos):
mcf_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 inp.in > inp.out

Bwaves (Giorgos Komodromos):
bwaves_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 bwaves_1 < bwaves_1.in > bwaves_1.out

Lbm (Zaharias Georgiou):
lbm_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 3000 reference.dat 0 0 100_100_130_ldc.of > lbm.out

Omnetpp (Giorgos Moleskis):
omnetpp_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 -c General -r 0 > omnetpp.General-0.out

Wrf (Marios Palechoros):
wrf_r_base.EPL221SingleCore-gcc4.8.5-linux3.10-bits-64 > rsl.out.0000

Τα benchmarks και τα δεδομένα εισόδου τους θα τα βρείτε στον πιο κάτω φάκελο: **/home/students/cs/CPU2017/CPU2017Source/benchspec/CPU.** Επίσης τα executables του κάθε benchmark θα τα βρείτε στο directory **exe**.

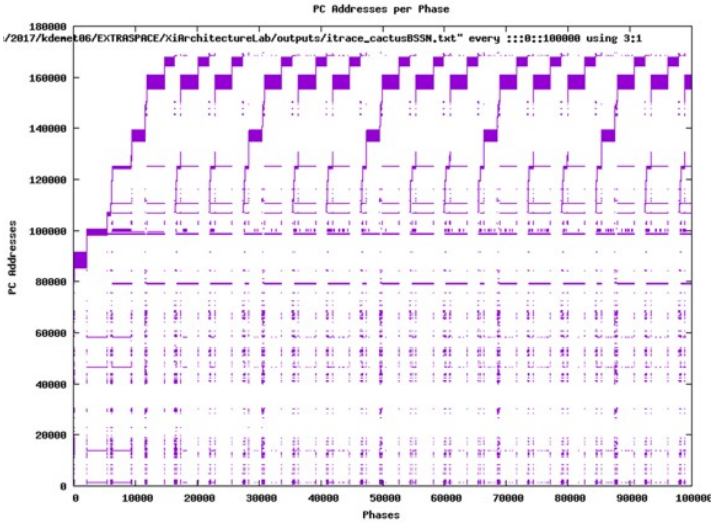
Τα inputs του κάθε benchmark θα τα βρείτε στο directory **data/refrate/input**. Προσοχή, υπάρχει περίπτωση κάποια από τα benchmarks να πρέπει να τρέξουν ήτε από το input directory ήτε από το exe directory.

Παρακαλώ όπως κάνετε copy τα directories των δυο αυτών benchmarks στο δικό σας προσωπικό directory και μετά να προχωρήσετε με τα παρακάτω.

1) Περιγράψετε με απλά λόγια το τι κάνει (τι υπολογίζει) το καθένα από τα Benchmarks που σας έχουν δοθεί και δώστε τον τρόπο και χρόνο (Wall Time) εκτέλεσης του (Χρησιμοποιήστε τις μηχανές του εργαστηρίου). Επίσης καταγράψτε την συχνότητα της μηχανής που χρησιμοποιήσατε για να τρέξετε το κάθε benchmark.

Αφού τρέξετε τα benchmarks με τη βοήθεια του εργαλείου PIN:

2) Να γίνει καταμέτρηση για το πόσες φορές εκτελέστηκαν οι εντολές branch κατά την εκτέλεση των πιο πάνω benchmarks. Η καταμέτρηση να γίνει για κάθε ένα εκατομμύριο εντολές. Δηλαδή κάθε εκατομμύριο εντολών θα αντικατοπτρίζει μια φάση. Για παράδειγμα να φτιάξετε μία γραφική παράσταση παρόμοια με την ακόλοθη:



3) Με τη βοήθεια του PINTool iTrace να δημιουργήσετε μια γραφική παράσταση με τη συχνότητα εμφάνισης των εντολών (cumulative) για κάθε φάση (1M εντολές).

4) Να βρείτε την συνάρτηση στην οποία ανήκει η εντολή που εκτελείτε τις περισσότερες φορές με την βοήθεια του objdump και να εξηγήσετε τι κάνει αυτή η συνάρτηση.

5) Χρησιμοποιώντας τις παραπάνω μετρήσεις σας, να υπολογίσετε το CPI του κάθε benchmark.

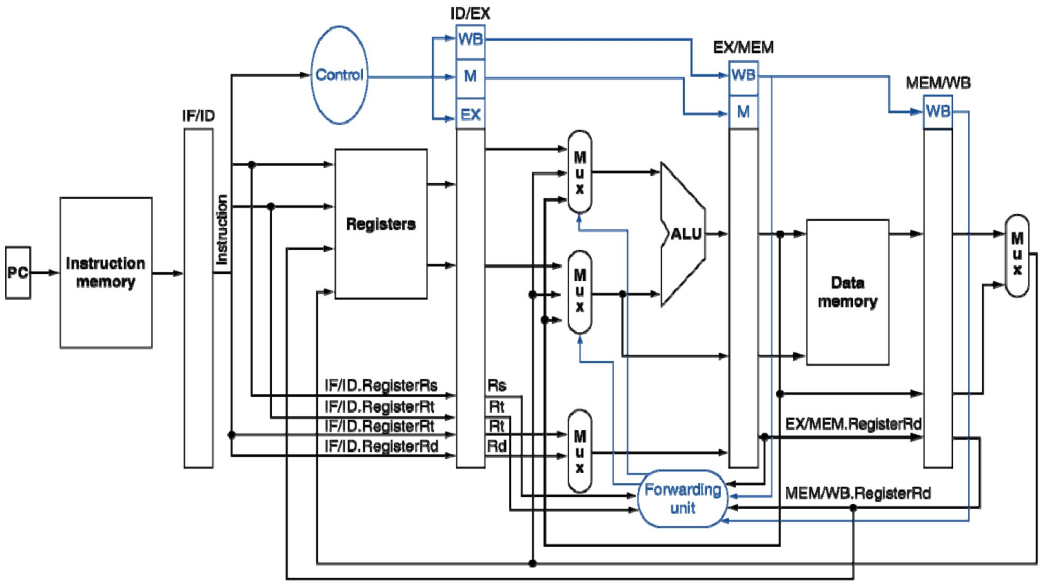
6) Σχολιάστε εκτενώς τα αποτελέσματα των πιο πάνω μετρήσεων σας καθώς και τη μεθοδολογία που ακολουθήσατε για να παράξετε το κάθε ένα από αυτά.

Θεωρητικές ασκήσεις.

1. Σχεδιάστε τον τρόπο εκτέλεσης με Gantt Chart των ποιο κάτω εντολών σε ένα 5-stage pipeline και να αναφέρετε τον συνολικό αριθμό κύκλων.

```
lw $2, 100($1)
add $3, $1, $2
add $5, $6, $2
sw $5, 200($3)
add $7, $8, $9
sub $10, $7, $2
add $5, $2, $1
```

2. Στο πιο κάτω σχήμα προσθέστε τα απαραίτητα συστατικά ώστε αν εντολές LW-SW εκτελούνται σε διαδοχικά (χρησιμοποιούν τον ίδιο καταχωρητή) να εκτελούνται πιο αποδοτικά (forwarding). (move from mem. Location A to mem. Location B)



3. **Άσκηση απο το βιβλίο (5th edition of Computer Architecture: A Quantitative Approach by Hennessy and Patterson) [15/15]** <B.1> For the purpose of this exercise, we assume that we have 512-byte cache with 64-byte blocks. We will also assume that the main memory is 2 KB large. We can regard the memory as an array of 64-byte blocks: M0, M1, ..., M31. Figure B.30 sketches the memory blocks that can reside in different cache blocks if the cache was fully associative.

a. [15] <B.1> Show the contents of the table if cache is organized as a direct- mapped cache.

b. [15] <B.1> Repeat part (a) with the cache organized as a four-way set associative cache.