Εργαστήριο #4 (Simulator GEM5)

Με την προϋπόθεση ότι έχετε κάνει compile τον προσομοιωτή για αρχιτεκτονική X86, όπως ακολουθεί:

scons build/X86/gem5.opt -j3

(ο αριθμός στο j είναι ο αριθμός πυρήνων+1)

θα τρέξουμε ένα παράδειγμα εφαρμογής στον gem5. Έστω το παρακάτω πρόγραμμα (main.c)

```
#include <stdio.h>
int main( int argc, char **argv ) {
  size_t I;
  for ( i=0; i<(size_t)argc; i++) {
     printf("%s\n", argv[i]);
  }
  return 0;
}
```

Κάνετε compile το πρόγραμμά σας ώστε να παράγετε το εκτελέσιμο για αρχιτεκτονική X86:

gcc -O0 -ggdb3 -std=c99 -static -o mainx86.out main.c

Τώρα βεβαιωθείτε ότι όταν το τρέξετε με τα default στον gem5, λειτουργεί σωστά,

```
build/X86/gem5.opt configs/example/se.py -c mainx86.out -o 'arg1 arg2'
```

Άρα θα πρέπει να δείτε να τυπώνει τα ορίσματα:

arg1

arg2

Μελετήστε το αρχείο με τα στατιστικά που γράφονται: m5out/stats.txt

Στα default θα δείτε να αναφέρεται η συχνότητα προσομοίωσης, στα 1000000000000 ticks per sec.

Δεν είναι ιδιαίτερα ενδιαφέρον το αποτέλεσμα της προσομοίωσης, επαληθεύετε όμως ότι λειτουργεί σωστά το πρόγραμμά σας στον προσομοιωτή. Όπως θα δείτε στις ρυθμίσεις, m5out/config.ini:

```
[system.cpu]
type=X86AtomicSimpleCPU
```

ο προσομοιωτής gem5 χρησιμοποιεί το default μοντέλο CPU που κάνει ατομικές λειτουργίες και ατομικές προσβάσεις στην μνήμη, οπότε δεν υπάρχουν πραγματικά δεδομένα χρόνου και επιδόσεων.

Βήμα 1

Τώρα θα κάνετε προσομοίωση με το μοντέλο επεξεργαστή με timing, άρα θα χρησιμοποιήσετε το μοντέλο TimingSimpleCPU και το μέγεθος που φαίνεται παρακάτω για data cache και instruction cache. (προσοχή στο case sensitivity των ορισμάτων που δίνετε).

```
build/X86/gem5.opt configs/example/se.py --cmd=mainx86.out -o ‘arg1 arg2’ --cpu-type=TimingSimpleCPU --l1d_size=64kB --l1i_size=16kB
```

έλεγξετε το αρχείο m5out/config.ini for “cache”, αλλά θα διαπιστώσετε ότι δεν δημιουργήθηκε κάποια cache!

Σημείωση: στην τελευταία γραμμή προσέξτε σε ποιο tick έχει ολοκληρωθεί το simulation και κάνει τερματισμό ο gem5:

```
Exiting @ tick 666458000 because exiting with last active thread context
```

Τώρα δώστε τη σωστή εντολή για εκτέλεση:

```
build/X86/gem5.opt configs/example/se.py --cmd=mainx86.out -o ‘arg1 arg2’ --cpu-type=TimingSimpleCPU --l1d_size=64kB --l1i_size=16kB --caches
```

Παρατηρήστε σε ποια χρονική στιγμή ολοκληρώθηκε τώρα η προσομοίωση και υπολογίστε το ποσοστό βελτίωσης. Βεβαίως ελέξτε και το αρχείο config.ini

Σε πόσο χρόνο έτρεξε το πρόγραμμα ? Ελέξτε το stats.txt ("sim_seconds": χρόνος εκτέλεσης του προγράμματος σε δευτερόλεπτα και προκύπτει από το sim_ticks*(10^{-12}) αφού τα tick του προσομοιωτή αντιστοιχούν σε picoseconds).

Πόσες εντολές εκτελέστηκαν ? ("sim_insts" δίνει τον αριθμό των εντολών που εκτελέστηκαν)

Ποιός είναι ο αριθμός

system.cpu_cluster.cpus.icache.demand_misses::total

system.cpu_cluster.cpus.dcache.demand_misses::total

Υπολογίστε το παρακάτω για το πρόγραμμα το συγκεκριμένο.

CPI = 1 + ((IL1.misses + DL1.misses) * mem_penalty)/sim_insts