**ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΠΡΟΗΓΜΕΝΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

Ερωτήματα Πρώτου Μέρους

1. starter\_se.py

Βασικές παράμετροι που πέρασε στον gem5 το σύστημα προς εξομείωση.

--cpu (τύπος επεξεργαστή):

“Atomic” : AtomicSimpleCPU (default)

“Minor” : MinorCPU

“Hpi” :HPI.HPI

--cpu-freq (συχνότητα-ρολόι επεξεργαστή):

1 GHz (default)

--num-cores ( αριθμός πυρήνων επεξεργαστή):

1 (default)

--mem-type (τύπος μνήμης που χρησιμοποιούμε):

DDR3\_1600\_8x8 (default)

--mem-channels (αριθμός καναλιών μνήμης):

2 (default)

--mem-ranks (κατάταξη μνήμης ανά κανάλι): ???

None (default)

--mem-size (μέγεθος μνήμης):

2GB (default)

Πως θα μπορούσαμε να αλλάξουμε τη συχνότητα λειτουργίας του συστήματος; ??? από parser.add.argument(“--cpu-freq” ….)

1. stats.txt

Παραγόμενα στατιστικά απο το simulation του gem5.

sim\_secons: 0.000035 (Number of seconds simulated)

sim\_insts: 5027 (Number of instructions simulated)

host\_inst\_rate: 66131 (Simulator instruction rate (inst/s) )

1. config.ini , config.json.

Έστω πως L1 = 6 cycles και L2 = 50 cycles και 1 cycle cache hit/instruction execution

Από τα αρχεία παρατηρούμε πως:

l2.overall\_misses::total = 474

Icache.overall\_misses::total = 327

dcache.overall\_misses::total = 177

Οπότε υπολογίζοντας την σχέση CPI που μας δίνεται, έχουμε: CPI = 6.31

1. gem5.org

In-order CPUs :

SimpleCPU:

Ο **SimpleCPU** είναι ένας πλήρως λειτουργικός τύπος επεξεργαστή, το οποίο είναι κατάλληλο για περιπτώσεις στις οποίες δεν είναι απαραίτητο ένα λεπτομερές μοντέλο.Τέτοιες περιπτώσεις είναι οι περίοδοι προθέρμανσης, η διαδικασία testing για την ομαλή λειτουργία ενός προγράμματος. Πρόσφατα επαναπρογραμματίστηκε έτσι ώστε να λειτουργεί σύμφωνα με το νέο σύστημα μνήμης και πλέον διακρίνετε σε τρεις κατηγορίες :BaseSimpleCPU , AtomicSimpleCPU, TimingSimpleCPU.

**BaseSimpleCPU**: Τον χρησιμοποιούμε για περιπτώσεις όπου, θέλουμε να καθορίζει λειτουργίες για τον έλεγχο για διακοπές, τη ρύθμιση ενός αιτήματος ανάκτησης, τον χειρισμό της ρύθμισης πριν από την εκτέλεση, τον χειρισμό ενεργειών μετά την εκτέλεση και την προώθηση του υπολογιστή στην επόμενη εντολή. Δεν μπορεί να εκτελεστεί από μόνος του, καθώς πρέπει να χρησιμοποιήσει κλάσεις που κληρονομούνται απο τα άλλα δύο μοντέλα

**AtomicSimpleCPU**: Είναι η έκδοση που χρησιμοποιεί πρόσβαση ατομικής μνήμης. Χρησιμοποιεί τις εκτιμήσεις καθυστέρησης για την ατομικές προσβάσεις για να υπολογίσει τον συνολικό χρόνο πρόσβασης της cache. Προέρχεται από την έκδοση BaseSimple και εφαρμόζει λειτουργίες ανάγνωσης και εγγραφής μνήμης, και για κάθε τικ που καθορίζει τι συμβαίνει σε κάθε κύκλο. Τέλος, καθορίζει τη θύρα που χρησιμοποιείται για την σύνδεση στη μνήμη και συνδέει τον CPU με την μνήμη cache.

**TimingSimpleCPU**: Είναι η έκδοση που χρησιμοποιεί προσβάσεις στη μνήμη χρονισμού. Σταματά τις προσβάσεις cache και περιμένει το σύστημα μνήμης να απαντήσει πριν προχωρήσει. Προέρχεται και αυτός από τον BaseSimple και ενσωματώνει το ίδιο σετ συναρτήσεων. Καθορίζει τη θύρα που χρησιμοποιείται για τη σύνδεση στη μνήμη και συνδέει τον CPU με την cache. Τέλος, ορίζει επίσης τις απαραίτητες λειτουργίες για το χειρισμό της απόκρισης από τη μνήμη στις προσβάσεις που αποστέλλονται.

**Minor CPU Model**: Είναι ένα μοντέλο επεξεργαστή με σταθερό pipeline αλλά διαμορφώσιμες δομές δεδομένων και συμπεριφορά εκτέλεσης. Προορίζεται να χρησιμοποιηθεί για τη μοντελοποίηση επεξεργαστών με αυστηρή συμπεριφορά εκτέλεσης κατά σειρά και επιτρέπει την οπτικοποίηση της θέσης μιας εντολής στη διοχέτευση μέσω της μορφής/εργαλείου MinorTrace/minorview.py. Η πρόθεση είναι να παράσχει ένα πλαίσιο για μικροαρχιτεκτονικό συσχετισμό του μοντέλου με έναν συγκεκριμένο, επιλεγμένο επεξεργαστή με παρόμοιες δυνατότητες. Δεν διαθέτει την δυνατότητα multithreading.

4a. Πρόγραμμα σε C και εκτέλεση στον gem5. TimingSimpleCPU και MinorCPU με se.py