ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Αρχιτεκτονική Προηγμένων Υπολογιστών

1η Εργαστηριακή Άσκηση

Παναγιώτης Σαββίδης

8094

ΘΕΣΣΑΛΟΝΙΚΗ 2020

1.

CPU types:

i) Atomic Simple CPU
ii) Minor CPU
iii) HPI CPU

Από τις παραπάνω CPU τελικά χρησιμοποιήσαμε την **Minor**

Voltage_Domain = 3.3V (by default)
Clock speed = 1GHz (by default)
CPU cluster Voltage = 1.2V (by default)
Number of Cores = 1 (by default)

Caches: Εφόσον επιλέξαμε την Minor CPU θα έχουμε **L1** και **L2** caches

Μνήμη : Στην εντολή που τρέξαμε δεν προσδιορίσαμε κάτι για την μνήμη (μέσω της εντολής –mem) συνεπώς θα ληφθούν οι default τιμές.
mem-type = DDR3_1600_8x8
mem-channels = 2
mem-ranks = None
mem-size = 2GB
membus = SystemXBar()

2. *A)*

Το **cpu type** βρίσκεται στη γραμμή 65

```
[system.cpu_cluster.cpus]
type=MinorCPU
```

Στη γραμμή 1652 φαίνεται το **Voltage**

```
[system.voltage_domain]
type=VoltageDomain
eventq_index=0
voltage=3.3
```

Domain.

Στη γραμμή 44 βρίσκουμε το **Clock**
Το 1000 αναφέρεται στα ticks, άρα έχουμε

```
[system.clk_domain]
type=SrcClockDomain
clock=1000
```

Domain.

$\frac{1}{1000}$ ή $\frac{1}{10^{-9}} = 10^9 = 1GHz$

Στη γραμμή 1339 υπάρχει το **Voltage.**

```
[system.cpu_cluster.voltage_domain]
type=VoltageDomain
eventq_index=0
voltage=1.2
```

Στη γραμμή 58 βρίσκουμε το **cpu Clock.** Με αντίστοιχο τρόπο όπως και πιο πάνω , έχουμε 4GHz

```
[system.cpu_cluster.clk_domain]
type=SrcClockDomain
clock=250
```

Στη γραμμή 1610 υπάρχει το **membus type**.

```
[system.membus]
type=CoherentXBar
```

Τέλος στη σειρά 113 βρίσκεται το **number of** `numThreads=1` **Cores**.

B)

Το συνολικό νούμερο των **committed Instructions** φαίνονται στις γραμμές 14 και 15.

```
system.cpu_cluster.cpus.committedInsts          5028                       # Number of instructions committed
system.cpu_cluster.cpus.committedOps            5834                       # Number of ops (including micro ops) committed
```

Αυτή η διαφορά εμφανίζεται γιατί στη δεύτερη σειρά υπολογίζονται και τα micro operations.

C)

Ο συνολικός αριθμός **που προσπελάστηκε η L2 cache** φαίνεται στη σειρά 493 στο **stats.txt**.

```
system.cpu_cluster.l2.demand_accesses::total          479                       # number of demand (read+write) accesses
```

Αν αυτό το νούμερο δεν μας δινόταν, θα μπορούσαμε να το υπολογίσουμε από το άθροισμα των σειρών 556,574 και 594, όπως φαίνεται παρακάτω.

```
system.cpu_cluster.l2.ReadCleanReq_accesses::total          332                       # number of ReadCleanReq accesses(hits+misses)
system.cpu_cluster.l2.ReadExReq_accesses::total           43                       # number of ReadExReq accesses(hits+misses)
system.cpu_cluster.l2.ReadSharedReq_accesses::total          104                       # number of ReadSharedReq accesses(hits+misses)
```

3.

MinorCPU:

Αυτό το μοντέλο CPU, όπως μας ζητήθηκε, είναι ένα in-order μοντέλο επεξεργαστή. Χρησιμοποιείται κυρίως διότι επιτρέπει την οπτικοποίηση της θέσης μιας εντολής μέσα στο pipeline διαμέσω του MinorTrace/minorview.py format/tool. Η MinorCPU χρησιμοποιείται για να προσομοιώνει μοντέλα επεξεργαστών με αυστηρή in-order εκτέλεση και να παρέχει μια δομή για μικρο-αρχιτεκτονική συσχέτιση του μοντέλου με μια συγκεκριμένη διαδικασία, όπως έχουμε επιλέξει, που έχει παρόμοιες δυνατότητες με αυτήν που χρησιμοποιούμε.

AtomicSimpleCPU:

Το μοντέλο αυτό αποτελεί μια version της SimpleCPU η οποία χρησιμοποιεί atomic memory accesses. Η AtomicSimpleCPU επιλέγει ποια θύρα θα χρησιμοποιήσει έτσι ώστε να έρθει σε

επαφή με τη μνήμη και να ενώσει την CPU με την μνήμη Cache. Για να εκτιμήσει τον συνολικό χρόνο πρόσβασης της Caches, η AtomicSimpleCPU χρησιμοποιεί τα latency estimates από τα atomic accesses.

TimingSimpleCPU:

Παρόμοια, αυτό το μοντέλο αποτελεί μια έκδοση της SimpleCPU, αλλά διαφέρει καθώς χρησιμοποιεί χρονικές προσβάσεις μνήμης. Επίσης, καθυστερεί τις προσβάσεις στη μνήμη Cache και περιμένει για το σύστημα μνήμης να αποκριθεί, πριν προχωρήσει στην αποστολή της επόμενης.

Default

	MinorCPU	**TimingSimpleCPU**
Number of Ticks	45767000	63736000
Sim_seconds	0.000046	0.000064
Number of CPU cycles	91534	127472
Number of Instructions Commited	23914	23759
CPI: cycles per instruction	3.827632	Δεν υπάρχει στο αρχείο, αλλά με μια διαίρεση προκύπτει 5.365209

--sys-clock=400000000

	MinorCPU	**TimingSimpleCPU**
Number of Ticks	57403000	74527500
Sim_seconds	0.000057	0.000075
Number of CPU cycles	114806	149055
Number of Instructions Commited	23914	23759
CPI: cycles per instruction	4.800786	Δεν υπάρχει στο αρχείο, αλλά με μια διαίρεση προκύπτει 6.273623

--mem-type=DDR4_2400_8x8

	MinorCPU	**TimingSimpleCPU**
Number of Ticks	44503000	63637000
Sim_seconds	0.000045	0.000064
Number of CPU cycles	89006	127274
Number of Instructions Commited	23914	23759
CPI: cycles per instruction	3.72192	Δεν υπάρχει στο αρχείο, αλλά με μια διαίρεση προκύπτει 5.356875

Όπως παρατηρούμε, σε όλες τις περιπτώσεις ο MinorCPU τρέχει πιο γρήγορα από τον TimingSimpleCPU (sim_seconds) και αυτό μπορούμε να το αποδώσουμε στο γεγονός ότι στον MinorCPU υπάρχει το pipeline, συνεπώς δεν χρειάζεται να περιμένει να τελειώσει η προσπέλαση της μνήμης πριν συνεχίσει στην επόμενη εντολή. Η ίδια παρατήρηση γίνεται και από τον αριθμό των ticks σε κάθε περίπτωση.

Ο αριθμός των instructions και στα δύο μοντέλα είναι σχεδόν ίδιος. Αντίθετα ο αριθμός των CPU cycles έχει μια αισθητή διαφορά και αυτό οφείλεται στο ότι ο TimingSimpleCPU χρησιμοποιεί timing memory access, άρα χρειάζονται περισσότεροι κύκλοι.

Όταν αλλάζουμε την συχνότητα από το 1GHz σε 0.4GHz παρατηρούμε ότι αυξάνονται ο αριθμός των ticks, ο χρόνος εκτέλεσης και ο αριθμός των CPU cycles. Τέτοιο αποτέλεσμα είναι απολύτως λογικό.

Τέλος με την αλλαγή της τεχνολογίας της μνήμης σε DDR4, αναμένουμε όλα τα νούμερα να είναι μικρότερα, κάτι το οποίο επαληθεύεται από τις πειραματικές μετρήσεις.

Ο αριθμός των Instructions παρατηρούμε ότι δεν αλλάζει, όταν αλλάζουμε τις ρυθμίσεις, καθώς το τι πρέπει να εκτελεστεί εξαρτάται μόνο από το πρόγραμμα και όχι από τη συχνότητα και τον τύπο της μνήμης.

Αξιολόγηση εργασίας

Για την εργασία χρειάστηκα αρκετό χρόνο, κυρίως επειδή δεν έχω μεγάλη εμπειρία από linux. Η διαδικασία του να φτιάξω και να ρυθμίσω το VM, τον gem5 και ότι επιπλέον χρειάζεται ήταν αρκετά επεξηγηματική στο pdf της εργασίας και όλα κύλησαν ομαλά. Στα ερωτήματα πρώτου μέρους, δυσκολεύτηκα και παρατήρησα ότι υπήρχαν κάποια λάθη (κυρίως τυπογραφικά) και ασάφειες στην εργασία που μου κόστισαν πολύ χρόνο, καθώς δεν μου ήταν κατευθείαν εμφανές το τι φταίει. Όταν όμως κατάφερα και δούλεψαν όλα, μου άρεσε η όλη διαδικασία και πιστεύω έμαθα καινούργια πράγματα για τα linux, για τα μοντέλα επεξεργαστών αλλά και για το github.

Μια συμβουλή που θα έδινα θα ήταν, να γίνει λίγο πιο επεξηγηματικό και το κομμάτι των ερωτημάτων και να μην θεωρούνται δεδομένα όλα τα προ απαιτούμενα. Σε πολλά ερωτήματα δεν καταλάβαινα τι ακριβώς πρέπει να επιλέξω να συγκρίνω από ένα αρχείο με πάρα πολλές γραμμές. Επίσης αν γίνεται, θα βοηθούσε πολύ να ανέβουν κάποιες «πρότυπες» απαντήσεις ή να μας πείτε κάποιο πλήρες github αρχείο

κάποιο συμφοιτητή μας, για να μπορέσουμε να δούμε ποιές ήταν οι πλήρεις απαντήσεις.

Συνολικά, ήταν μια ενδιαφέρον, ωραία αλλά λίγο χρονοβόρα άσκηση.

ΠΗΓΕΣ:

http://www.gem5.org/documentation/general_docs/cpu_models/SimpleCPU

http://www.gem5.org/documentation/general_docs/cpu_models/minor_cpu