ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ
1Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΔΕΛΗΖΩΝΑΣ ΑΠΟΣΤΟΛΟΣ: 8479
ΤΣΙΑΝΤΟΣ ΑΠΟΣΤΟΛΟΣ: 8256

ΕΙΣΑΓΩΓΗ
Η 1η εργαστηριακή άσκηση περιλαμβάνει το παρόν αρχείο README.md, καθώς και όλα τα αρχεία εξόδου όπως (stats.txt, config.ini κλπ) για κάθε διαφορετικό μοντέλο προσωμοιωσης.Τα αρχεία αυτά βρίσκονται σε ξεχωριστούς φακέλους για τα αντίστοιχα μοντέλα, στην ονομασία των οποίων αναδεικνύονται και οι βασικές παράμετροι του μοντέλου. Επιπλέον, υπάρχει σε ξεχωριστό φάκελο ο πηγαίος κώδικας σε .c αρχείο, καθώς και το compiled αρχείο για αρχιτεκτονική ARM. Ο πηγαίος κώδικας είναι εξαιρετικά απλός, καθώς σκοπός της εργασίας αυτής ήταν ο πειραματισμός με το gem5. Στο τέλος του αρχείου README.md, υπάρχουν και οι πηγές που χρησιμοποιήθηκαν για την εκπόνηση της εργασίας.

Ερωτήματα 1ου Μέρους

1.
Ανοίγοντας το αρχείο starterse.py, το οποίο χρησιμοποιήσαμε ως configuration για να τρέξουμε το παράδειγμα "Hello World" και πηγαίνοντας στην συνάρτηση main(), μπορούμε να δούμε τις βασικές παραμέτρους που έχουν περάσει στον gem5 για το σύστημα προς εξομοίωση. Αναλυτικότερα:

Μοντέλο CPU: Β Παρατηρούμε πως το default μοντέλο για το configuration file μας είναι το "atomic" CPU, ωστόσο όταν εκτελέσαμε το παράδειγμα χρησιμοποιήσαμε σαν argument το --cpu="minor", πράγμα που σημαίνει πως η εκτέλεση του προγράμματος έγινε με "minor" CPU.

Για όλα τα υπόλοιπα χαρακτηριστικά του μοντέλου, κι εφόσον δεν δώσαμε κάποιο άλλο argument, το configuration file χρησιμοποιεί τις default τιμές του. Συγκεκριμένα:
Συχνότητα λειτουργίας: --cpu-freq, default=4GHz
Αριθμός πυρήνων: --num-cores, default-1
Τύπος μνήμης και μέγεθος μνήμης: --mem-type, default=DDR3-1600-8x8 , --mem-size, default=2GB
Μέγεθος γραμμής στην cache: cache line size = 64 byte

2.
Από το αρχείο config.ini, μπορούμε εύκολα να διασταυρώσουμε τις παραμέτρους του configuration file, καθώς φαίνονται ξεκάθαρα στα πεδία

Cache line size = 64 byte
system.cpu_cluster.cpus --- type = MinorCPU

Ενδιαφέρον παρουσιάζει η συχνότητα λειτουργίας, η οποία δεν εμφανίζεται ως συχνότητα ακριβώς, αλλά ως clock=250. Αυτό μεταφράζεται ως 250picoseconds, δηλαδή η περίοδος ενός παλμού. Συνεπώς έχουμε: --cpu-freq = 1/250picoseconds = 4GHz, επιβεβαιώνοντας τη παράμετρο που είδαμε στο configuration file.

3.
Για τα επόμενα ερωτήματα της εργασίας, θα χρησιμοποιήσουμε τις CPUs: TimingSimpleCPU & MinorCPU: Ας δούμε λίγα παραπάνω για καθεμία από αυτές.

TimingSimpleCPU: Αυτή η CPU, αποτελεί μια πολύ απλή μοντελοποίηση κατά την οποία η cpu περιμένει την προσπέλαση της μνήμης πριν προχωρήσει, ωστόσο απλοποιεί πολύ τη λειτουργία της αφού δεν υπάρχει η λογική του pipelining, πράγμα που σημαίνει πως ότι σε κάθε στιγμή γίνεται επεξεργασία ενός και μόνο instruction. Η TimingSimpleCPU έχει CPI=1, που σημαίνει πως κάθε αριθμητικό instruction εκτελείται σε ένα κύκλο, ενώ η προσπέλαση της μνήμης απαιτεί πολλαπλούς κύκλους.

MinorCPU: Η MinorCPU αποτελεί ένα αρκετά πιο λεπτομερές είδος IN-ORDER CPU, το οποίο είναι σε θέση να προσομοιώσει ρεαλιστικά συστήματα. Σε αντίθεση με την οικογένεια των BasicCPUs, διαθέτει in-order execution pipeline λογική τεσσάρων σταδίων, τα οποία είναι (fetch1(),fetch2(),decode(),execute()).

Από τα παραπάνω συμπαιρένουμε πως η MinorCPU θα είναι αρκετά γρηγορότερη στην εκτέλεση του προγράμματός μας στον ίδιο περίπου αριθμό instructions(καθώς δεν αλλάζει το εκτελέσιμο πρόγραμμα), ενώ αντιθέτως η διαδικασία του simulation θα διαρκέσει λιγότερο σε πραγματικό χρόνο στον υπολογιστή μας κατα την TimingSimpleCPU, καθώς πρόκειται για αρκετά πιο απλή λογική.

3a,b.
Στη συνέχεια της εργασίας μας θα εκτελέσουμε το πρόγραμμα t_prog.c το οποίο έχουμε δημιουργήσει(Ο πηγαίος κώδικας βρίσκεται στο φάκελο source code). Κατά την εκτέλεση του προγράμματος θα χρησιμοποιήσουμε το configuration file se.py, και για το πρώτο ερώτημα θα αφήσουμε όλες τις παραμέτρους ίδιες αλλάζοντας μόνο το --cpu-type, όπου θα χρησιμοποιήσουμε τις δύο διαφορετικές που αναφέρθηκαν πιο πάνω.

Αναλυτικότερα για τα 2 διαφορετικά μοντέλα:

TimingSimple	Minor	
277983000	139155000	Number of ticks from beginning of simulation
219274	64226	Simulator instruction rate (inst/s)
674208	678560	Number of bytes of host memory used
221444	64900	Simulator op (including micro ops) rate (op/s)
0.76	2.58	Real time elapsed on the host
367215182	53837488	Simulator tick rate (ticks/s)
1000000000000	1000000000000	Frequency of simulated ticks
165926	165986	Number of instructions simulated
167619	167745	Number of ops (including micro ops) simulated
0.000278	0.000139	Number of seconds simulated
277983000	139155000	Number of ticks simulated

Πολύ εύκολα παρατηρούμε πως οι προβλέψεις μας με βάση τη λογική των εκάστοτε CPUs επιβεβαιώνονται, καθώς ο χρόνος εκτέλεσεις του προγράμματος είναι σαφώς μικρότερος στην MinorCPU (διπλάσιος), στον ίδιο περίπου αριθμό instructions. Ωστόσο ο gem5 χρειάζεται αρκετά περισσότερη ώρα για να προσομοιώσει τη MinorCPU, καθώς η λογική της είναι πιο περίπλοκη.

Τα αρχεία stats.txt στα οποία βρίσκονται όλες οι παραπάνω πληροφορίες βρίσκονται στους φακέλους /statistics_1lab/Minor & /statistics_1lab/TimingSimple αντίστοιχα για τα παραπάνω μοντέλα.

3c.

Για το τελευταίο ερώτημα της εργασίας, έχουμε δοκιμάσει να τρέξουμε το μοντέλο της MinorCPU αλλάζοντας τη συχνότητα σε 4GHz γεγονός που επιφέρει μια σαφή μείωση στο χρόνο εκτέλεσης του προγράμματος χωρίς να αλλάζει βέβαια τον τελικό αριθμό instructions που εκτελούνται. Παρακάτω παραθέτονται τα αποτελέσματα: (το stats.txt για τη προσομοίωση στα 4GHz βρίσκεται στο φάκελο /statistics_1lab/Minor4ghz)

Minor	Minor-4GHz	
139155000	83855000	Number of ticks simulated
64226	678816	Simulator instruction rate (inst/s)
165986	165986	Number of instructions
0.000139	0.000084	Number of seconds simulated

Στη συνέχεια, δοκιμάζουμε να τρέξουμε το μοντέλο της TimingSimpleCPU αλλάζοντας τη μνήμη cache σε επιπέδου 2 (l2cache) (Στα παραδείγματα που συγκρίνονται η TimingSimpleCPU έχει τρέξει στα 4GHz). Ως γνωστών, η l2cache έχει μεγαλύτερο μέγεθος από το πρώτο επίπεδο cache, ωστόσο είναι αρκετά πιο αργή. Το γεγονός αυτό φαίνεται ξεκάθαρα στα δεδομένα που παίρνουμε από την προσομοίωση, καθώς ο χρόνος εκτέλεσης με l2cache είναι σαφώς μεγαλύτερος. Παρακάτω παραθέτονται τα αποτελέσματα: (το stats.txt για τη προσομοίωση με 4GHZ/l2cache βρίσκεται στο φάκελο /statistics_1lab/TimingSimple4ghz & /s/TimingSimple4ghzl2cache)

TimingSimple-4GHz	TimingSimple-4GHz-l2cache	
152020000	1409974750	Number od ticks simulated
223533	104742	Simulator instruction rate (inst/s)
165926	165926	Number of instructions
0.000152	0.001410	Number of seconds simulated

Στο φάκελο /statistics_1lab, υπάρχουν και προσομοιώσεις για τύπους μνήμης ram DDR3 και DDR4, με τη διαφορά στους χρόνους εκτέλεσης ωστόσο να είναι ελάχιστη, καθώς μιλάμε για ένα εξαιρετικά απλοικό πρόγραμμα.

Βιβλιογραφία

Arm Research Starter Kit: System Modeling using gem5 (https://raw.githubusercontent.com/arm-university/arm-gem5-rsk/master/gem5_rsk.pdf)

A Tutorial on the Gem5 Minor CPU Model (https://nitish2112.github.io/post/gem5-minor-cpu/)

Learning gem5 (https://research.cs.wisc.edu/multifacet/papers/learning_gem5_tutorial.pdf)

The gem5 simulator wiki (http://gem5.org/Main_Page)