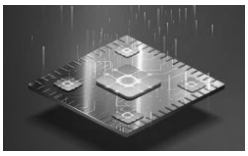
Εθνικο & Καποδιστριακο Πανεπιστημιο Αθηνων
Τμημα Πληροφορικης & Τηλεπικοινωνιων
Μαθημα: Αρχιτεκτονικη Υπολογιστων ΙΙ



Εργασία 1 (*υποχρεωτική*) – Διοχέτευση

Ακαδημαϊκο Ετος 2022 – 2023
(εκφωνηση) Τριτη 8 Νοεμβριου 2022
(παραδοση στο eclass μεχρι) Δευτερα 5 Δεκεμβριου 2022

Επώνυμο	Όνομα	Αριθμός Μητρώου	Email
Ζερμπίνος	Ιωάννης	1115202000053	sdi2000053@di.uoa.gr
Ρούσσος	Βασίλης	1115200700224	sdi0700224@di.uoa.gr
Λεωνίδας	Κουτσούκης	1115201500235	sdi1500235@di.uoa.gr

Πληροφορίες για τις Υποχρεωτικές Εργασίες του μαθήματος

- Οι υποχρεωτικές εργασίες του μαθήματος είναι **δύο**. Σκοπός τους είναι η κατανόηση των εννοιών του μαθήματος με χρήση αρχιτεκτονικών προσομοιωτών. Η πρώτη υποχρεωτική εργασία (αυτή) αφορά τη διοχέτευση (pipelining) και η δεύτερη θα αφορά τις κρυφές μνήμες (cache memories).
- Οι δύο εργασίες είναι υποχρεωτικές και η βαθμολογία του μαθήματος θα προκύπτει από το γραπτό (60%), την εργασία της διοχέτευσης (20%), και την εργασία των κρυφών μνημών (20%).
- Κάθε ομάδα μπορεί να αποτελείται **από 1 έως και 3 φοιτητές**. Συμπληρώστε τα στοιχεία όλων των μελών της ομάδας στον παραπάνω πίνακα. Όλα τα μέλη της ομάδας πρέπει να έχουν ισότιμη συμμετοχή και να γνωρίζουν τις λεπτομέρειες της υλοποίησης της ομάδας.
- Για την εξεταστική Σεπτεμβρίου δε θα δοθούν άλλες εργασίες. Το Σεπτέμβριο εξετάζεται μόνο το γραπτό.
- Σε περίπτωση αντιγραφής θα μηδενίζονται όλες οι ομάδες που μετέχουν σε αυτή.
- Η παράδοση της **Εργασίας Διοχέτευσης** πρέπει να γίνει μέχρι τα **μεσάνυχτα της προθεσμίας ηλεκτρονικά** και μόνο στο eclass (να ανεβάσετε ένα μόνο αρχείο zip ή rar με την τεκμηρίωσή σας σε PDF και τον κώδικά σας). **Μην περιμένετε μέχρι την τελευταία στιγμή. Η εκφώνηση της εργασίας 2 των κρυφών μνημών θα ανατεθεί αμέσως μετά.**

Ζητούμενο

Το ζητούμενο της εργασίας είναι να σχεδιάσετε έναν μικροεπεξεργαστή MIPS με διοχέτευση (pipeline) με την καλύτερη δυνατή **ενεργειακή αποδοτικότητα (energy efficiency)** δηλαδή την καλύτερη απόδοση (performance) ανά μονάδα ισχύος (power). Η εκτέλεση των προγραμμάτων και η αξιολόγηση των σχεδιάσεών σας θα γίνει στον προσομοιωτή QtMips που χρησιμοποιούμε στο εργαστήριο.

Το πρόγραμμα του οποίου την απόδοση θα αξιολογήσετε επεξεργάζεται μια λίστα N ακεραίων αριθμών [i1, i2, …, iN] η οποία δεν είναι ταξινομημένη και στη λίστα μπορεί να εμφανίζεται οποιοσδήποτε ακέραιος οσεσδήποτε φορές. Το πρόγραμμά σας πρέπει να βρίσκει: τον μικρότερο ακέραιο μέσα στη λίστα (MIN), τον μεγαλύτερο ακέραιο μέσα στη λίστα (MAX), το πλήθος των άρτιων αριθμών που περιέχει η λίστα (EVEN), το πλήθος των περιττών αριθμών που περιέχει η λίστα (ODD), το πλήθος των μηδενικών που περιέχει η λίστα (ZERO), το πλήθος των ακεραίων που η απόλυτη τιμή τους διαιρείται με το 3 (DIV3), το πλήθος των ακεραίων που η απόλυτη τιμή τους διαιρείται με το 5 (DIV5). Ο αρχικός πίνακας θα έχει ήδη περιεχόμενα στο τμήμα .data του προγράμματος και δεν θα δίδονται αριθμοί από τον χρήστη ούτε θα εκτυπώνονται αποτελέσματα στην κονσόλα[1]. Ο αρχικός πίνακας με τους N ακεραίους καθώς και τα αποτελέσματα των μετρήσεων θα πρέπει να βρίσκονται στην μνήμη και να διακρίνονται με κατάλληλες ετικέτες.

Υποθέστε ότι ο βασικός επεξεργαστής σας είναι ο **MIPS-single-cycle** σχεδιασμένος σε έναν μεγάλο κύκλο ρολογιού με ρυθμό 100 MHz. Η κατανάλωση ισχύος του είναι 20 watt.

Το επόμενο βήμα είναι να σχεδιάσετε τον επεξεργαστή **MIPS-pipeline-simple** ο οποίος διαθέτει διοχέτευση πέντε σταδίων και ρυθμό ρολογιού 300 MHz. Η κατανάλωση ισχύος του είναι 25 watt. Για τους κινδύνους δεδομένων διαθέτει μόνο ανίχνευση και προσθήκη καθυστερήσεων (stalls) αλλά όχι προώθηση (forwarding). Για τους κινδύνους ελέγχου διαθέτει επίσης μόνο ανίχνευση και προσθήκη καθυστερήσεων (δηλαδή stall-on-branch) και η επίλυση των διακλαδώσεων γίνεται στο στάδιο EX.

Το τελευταίο βήμα είναι η σχεδίαση ενός πιο επιθετικού επεξεργαστή **MIPS-pipeline-turbo** ο οποίος διαθέτει κι αυτός διοχέτευση πέντε σταδίων, ρυθμό ρολογιού 300 MHz και καταναλώνει 35 watt. Για τους κινδύνους δεδομένων διαθέτει ανίχνευση αλλά και προώθηση (forwarding). Για τους κινδύνους ελέγχου χρησιμοποιεί πρόβλεψη διακλάδωσης των 2-bit και έναν BHT των 5 bit. Η επίλυση των διακλαδώσεων γίνεται στο στάδιο ID.

Σε αυτή την εργασία θεωρούμε ότι η μνήμη είναι ιδανική και συνεπώς σε όλους τους επεξεργαστές οι κρυφές μνήμες πρέπει να είναι απενεργοποιημένες και η κύρια μνήμη να έχει χρόνο προσπέλασης 1 κύκλο ρολογιού.

[1] Κατά την ανάπτυξη του κώδικά σας και για τις δοκιμές σας μπορείτε να εκτυπώνετε ή να δίνετε εισόδους όπως το επιθυμείτε.

Για αρχικό μέγεθος πίνακα N=100 λάβετε τις παρακάτω μετρήσεις για το πρόγραμμά σας και για καθέναν από τους τρεις επεξεργαστές:

MIPS-single-cycle

Πλήθος κύκλων προγράμματος	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/ [χρόνος*ισχύς])
2371	2371	1	0.00002371	2108.815

MIPS-pipeline-simple

Πλήθος κύκλων προγράμματος	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/ [χρόνος*ισχύς])
3990	2534	1.57	0.00001326	3016.6

MIPS-pipeline-turbo

Πλήθος κύκλων προγράμματος	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/ [χρόνος*ισχύς])
2671	2534	1.054	0,0000089	3210.2

Επαναλάβετε το παραπάνω και καταγράψτε σε ξεχωριστούς πίνακες όμοιους με τον παραπάνω τις μετρήσεις σας για τιμές του N ίσες με 200 και 500. Τι παρατηρείτε;

Εκτός από τα προγράμματά σας σε συμβολική γλώσσα, που πρέπει να παραδώσετε ξεχωριστά να αναφέρετε και τα περιεχόμενα του αρχικού πίνακα ακεραίων με τα οποία πήρατε τις μετρήσεις σας. Αν πάρετε μετρήσεις με περισσότερα από ένα περιεχόμενα του πίνακα να δώσετε όλα τα δεδομένα.

Τεκμηρίωση

[Σύντομη τεκμηρίωση της λύσης σας μέχρι **10 σελίδες ξεκινώντας από την επόμενη σελίδα** - μην αλλάζετε τη μορφοποίηση του κειμένου (**και παραδώστε την τεκμηρίωση σε αρχείο PDF**). Η τεκμηρίωσή σας πρέπει να περιλαμβάνει παραδείγματα ορθής εκτέλεσης του προγράμματος και σχολιασμό για την επίλυση του προβλήματος και την επίτευξη του ζητούμενου. Μπορείτε να χρησιμοποιήσετε εικόνες, διαγράμματα και ό,τι άλλο μπορεί να βοηθήσει στην εξήγηση της δουλειάς σας.]

ΑΝΑΛΥΤΙΚΑ ΣΧΟΛΙΑ ΓΙΑ ΤΟΝ ΚΩΔΙΚΑ

(SINGLE CYCLE)

Αρχικά ξεκινάμε φορτώνοντας στον $s0 το size της λίστας (όπου ο $s0 λειτουργεί σαν counter της βασικής επανάληψης) καθώς και την διεύθυνση που ξεκινάει η λίστα στον $t0. Μετά κάνουμε αρχικοποίηση όλους τους counters($s1-$s7) για τα ζητούμενα και ξεκινάμε τη βασική επανάληψη. Αν δεν έχουμε προσπελάσει όλη τη λίστα τότε: φόρτωσε στον $t7 (ο αριθμός που θα χρησιμοποιήσουμε για όλες τις δουλειές) το στοιχείο όπου δείχνει ο $t0 (list[0]) και ξεκίνα.

- Αν ο αριθμός $s1 είναι μικρότερος του $t7 τότε ο $t5 θα πάρει την τιμή 1. Επομένως η από κάτω branch είναι ψευδής και ο $t7 είναι μεγαλύτερος οπότε εκχωρείται στον $s1 σαν τον νέο max μέχρι να βρεθεί ή όχι μεγαλύτερος.

- Ακριβώς ίδια δουλειά για τον min, μόνο που ελέγχεται στην περίπτωση που δεν είναι max, γιατί αν είναι max δεν υπάρχει περίπτωση να είναι min. Στην slt ο $t7 πρέπει να είναι μικρότερος αυτήν την φορά απο τον $s2 (καταχωρητής του min).

- Μετά κάνουμε andi (μάσκα) τον $t7 με το 1 (000…001) και αν το τελευταίο bit (που είναι αποθηκευμένο στον $t6) είναι 0, τότε αυξάνεται ο counter των άρτιων κατά 1 ($s4) και κάνουμε jump στο check_div3 αγνοώντας τελείως το is_odd, γιατί δεν είναι περιττός προφανώς.

- Μέσα εκεί ελέγχουμε αν έχουμε αριθμό μηδέν ή όχι, γιατί το μηδέν είναι άρτιος αριθμός άρα δεν χρειάζεται να κάνουμε αυτόν τον έλεγχο αν ο αριθμός που έχουμε στον $t7 είναι περιττός.

- Αν τώρα δεν είναι 0 το τελευταίο bit, τότε jump στο is_odd, και δεν χρειάζεται να κάνουμε κανέναν έλεγχο - αφού αν δεν είναι άρτιος, τότε θα είναι περιττός - και απλά αυξάνουμε το counter των περιττών $s5.

- Έπειτα γίνεται ο έλεγχος για το αν ο αριθμός δαιρείται με το 3 φορτώνοντας στον $t3 το 3 με li, διαιρούμε το $t7 με το 3, και αν το υπόλοιπο (mfhi που αποθηκεύτεται στον $t5) είναι 0, τότε ο αριθμός διαιρείται με το 3 και αυξάνουμε τον counter ($s6) κατά 1.

- Τώρα, το μόνο που μένει με τον ίδιο ακριβώς τρόπο όπως και πάνω να δούμε αν ο αριθμός διαιρείται με 5. Ο counter αποθηκεύεται στον καταχωρητή $s7.

- Για να πάμε σε επόμενο στοιχείο, αυξάνεται ο pointer $t0 που δείχνει στη λίστα κατά 4(1 byte, επόμενη λέξη) και ο counter $s0 που καθορίζει το main loop μειώνεται κατά 1 μέχρι να τελειώσει η επανάληψη. Φυσικά μετά jump στο main_loop.

- Τέλος, φορτώνεται στον $t7 η διεύθυνση του label max (αφού τελειώσαμε με την χρήση του) και κάνουμε store όλα τα αποτελέσματα των counters στην μνήμη ξεκινώντας απο το label max, μέχρι div5 σειριακά.

Παρένθεση: Έχουμε βάλει σε σχόλια έναν κώδικα που φτιάξαμε για να βρίσκουμε αν ένας αριθμός διαιρείται με το 3, χωρίς όμως να πληρώνουμε τους πολλούς κύκλους του div. Τον τρόπο τον βρήκαμε στο stack_overflow. Αλλά αφου ο QTMIPS μετράει το div σαν ένα κύκλο την παρατήσαμε την συγκεκριμένη τεχνική, αλλά το αφήσαμε στον κώδικα σαν κάτι έξτρα που σκεφτήκαμε να κάνουμε.

(PIPILINED STALL-FORWARDING)

Εδώ αναδιατάξαμε κάποιες εντολές για να γλιτώσουμε κύκλους από hazards. Θα αναφερόμαστε στο single cycle σαν πρότυπο κώδικα και μετά τι αλλαγή κάναμε. Εξήγηση:

- Αρχικά, αφού φορτώνουμε την διεύθυνση μνήμης του array θα υπάρχει καθυστέρηση 2 stalls με την επόμενη εντολή καθώς η επόμενη φορτώνει το περιεχόμενο που θα δείχνει ο $t0. Όμως θα πρέπει να περάσουν 2 κύκλοι ώστε να γίνει η εγγραφή στο $t0 στο στάδιο write back. Επομένως βάλαμε 2 εντολές από κάτω που δεν σχετίζονται για να εκμεταλλευτούμε αυτούς τους δύο κύκλους για να μην πάνε χαμένοι.

- Με την ίδια λογική κάνουμε και τις υπόλοιπες αναδιατάξεις. Η επόμενη στην γραμμή 26 είναι το li του $t3 που το κάνουμε για να μην δημιουργήσει μόνο 1 stall με την εντολή div που είναι 2 γραμμές πιο κάτω.

- Από κάτω κάνουμε slt για το max ώστε ο $t5 να πάρει την τιμή αρκετή ώρα πιο πριν απο τη branch που ελέγχει αυτόν τον καταχωρητή για να γλιτώσουμε stalls.

- Από κάτω κάνουμε το div που τώρα πλέον δεν δημιουργεί stall γιατί οι δύο καταχωρητές έχουν πάρει τις τιμές τους ήδη.

- Από κάτω ξανά, φορτώνουμε στον $t4 το 5 που θα είναι ο καταχωρητής που θα διαιρεθεί με τον $t7 για να δούμε αν διαιρείται με το 5 ο αριθμός. Ο λόγος είναι ακριβώς ο ίδιος με το li του $t3.

- Μετά κάνουμε το slt για το min για ακριβώς τον ίδιο λόγο με το slt του max.

- Και φορτώνουμε στον $t3 πλέον το υπόλοιπο της διαίρεσης που έγινε πιο πριν με το 3.

- Από κάτω, αφού πλέον έχουν περάσει τόσοι κύκλοι από το slt για το max, μπορούμε να κάνουμε το branch χωρίς κανένα stall.

- Και αν ο αριθμός δεν είναι μεγαλύτερος, προχωράει στην επόμενη ενέργεια που αυτή τη φορά δεν είναι να βρούμε το min αλλά να τσεκάρουμε αν ο αριθμός διαιρείται με το 3, αν δεν διαιρείται, jump στο skip και πάμε στο min. Ο λόγος που το κάναμε αυτό είναι για να ξεμπερδέψουμε από αυτή τη δουλειά γιατί χρειάζεται να κάνουμε τα ίδια για την διαίρεση με το 5 μόνο που δεν θα είχαμε ελεύθερο το mfhi καταχωρητή (δεν μπορούμε να έχουμε ταυτόχρονα 2 αποτελέσματα στον mfhi) **και** ταυτόχρονα χώρο για να μην φάμε stall. Ενώ τώρα ο mfhi είναι ελεύθερος για να τον χρησιμοποιήσουμε όποτε θέλουμε από δω και πέρα.

- Αφού λοιπόν τελειώσουμε με τον min, μετά στην γραμμή 45 ήρθε η ώρα να συνεχίσουμε με την διαίρεση του $t7 με το $t4 αφού έχουμε και τους δύο καταχωρητές ελεύθερους, και μετά mfhi στον $t5.

- Μετά από κάτω κοιτάμε αν είναι 0 ή όχι ο αριθμός χωρίς να μπορούμε να κερδίσουμε κάτι εδώ, αλλά έπειτα στην γραμμή 51 κάνουμε το and (μάσκα) και έπειτα ελέγχουμε το branch για το αν ο αριθμός διαιρείται με το 5, για να εκμεταλευτούμε τα 2 stall που θα δημιουργούσε η εντολή and(γραμμή 51) με το branch(γραμμή 56) στην περίπτωση που ήταν από κάτω γιατί θα stallαρε μέχρι ο $t6 να πάρει την τιμή από το and και να κάνει το branch. Στην συγκεκριμένη τεχνική κερδίζουμε 1000 κύκλους!

- Στο label skip_2 ελέγχουμε αν ο αριθμός είναι άρτιος μόνο που τώρα έχουμε το αποτέλεσμα στο $t6 λόγω της παραπάνω εξήγησης και δεν τρώμε stall. Αν είναι, τότε jump στο next_element, αλλιώς είναι περιττός.

- Έπειτα, στο label next_element απλά κάνουμε ότι κάναμε και πιο πριν, δηλαδή αύξηση pointer και μείωση counter.

- Τέλος, store πάλι τις τιμές στην μνήμη χωρίς να μπορούμε κάπως εδώ να γλιτώσουμε τα stalls των store.

Παρακάτω είναι και το πινακάκι με τις αναλυτικές μετρήσεις. Για single-cycle χρησιμοποιήθηκε ο πρώτος κώδικας, ενώ για τα άλλα δύο configurations ο δεύτερος κώδικας.

Επίσης πήραμε μετρήσεις με τα εξής σετ δεδομένων: -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5
Και αυτό που είναι και το μεγάλο των 500 αριθμών. Όταν θέλαμε 200 απλά αλλάζαμε το size του $s0 σε 200 που καθορίζει το loop.

-429,487,380,-967,691,250,217,53,853,431,-98,344,815,141,-925,-372,94,212,509,-644,449,767,-653,193,-104,909,957,140,543,200,-732,576,-609,412,228,-86,-548,207,327,-730,-921,-733,-38,300,640,-48,939,-302,-229,-44,947,-659,-15,802,-617,921,946,555,952,231,-290,719,-113,-756,955,-524,860,672,787,-421,-583,-541,-591,22,378,-323,-807,-549,-19,-720,-588,-813,61,-954,227,780,-475,-321,820,-80,-105,5,383,-538,-316,106,938,840,607,41,726,-811,962,897,-127,-503,884,-690,-451,-681,-483,-457,564,551,-907,74,328,-194,967,275,477,-969,-530,-907,-375,-176,29,258,367,987,37,-426,860,265,-703,-99,-218,891,431,-783,82,-834,-13,-699,-57,-446,-94,-938,-695,-313,937,-974,-988,63,252,-155,-633,705,-595,920,792,817,362,617,-251,654,18,-825,-968,929,342,613,19,108,-319,-307,-886,-783,-945,-221,-850,488,-650,-336,-239,-884,717,971,-345,-669,756,-212,963,151,535,372,-238,422,40,281,-637,239,-426,-196,-565,-760,393,656,597,366,489,-534,-187,856,675,-256,-700,0,-33,505,-630,483,-665,-61,2,-933,389,-657,-38,-5,355,530,-529,21,263,417,-351,-163,46,-257,-911,-173,876,-166,-31,-590,337,863,864,188,32,-600,-323,621,-609,813,-824,616,269,-277,-214,-183,815,434,106,-869,25,585,-625,346,856,-373,-178,302,578,-751,821,137,34,-534,532,-76,-962,-915,-627,305,-749,406,769,908,-481,284,606,791,804,433,173,-906,368,817,-310,541,-947,-688,723,-297,-287,-961,864,-91,-978,236,332,273,-687,-437,923,-744,-914,-704,-435,-597,-370,209,-913,85,422,647,-17,434,439,833,438,-503,264,-649,106,113,993,-85,444,-170,781,-448,646,17,-727,816,917,-688,-315,-781,-379,461,654,449,-879,-296,466,734,235,951,-263,71,201,-535,-741,346,665,313,387,-298,-659,-226,-866,509,-184,-296,94,388,-564,-269,310,-124,381,476,1,641,-968,209,308,-192,-325,287,541,-41,-891,506,-75,167,-765,583,339,837,831,-861,-782,776,-772,677,-900,881,-473,-532,16,949,-553,163,-59,68,773,-564,-612,-436,-557,-286,-883,-329,-236,-623,763,44,493,339,-161,-112,-333,-425,-792,339,-220,-391,-996,-482,451,-24,-134,790,54,519,-810,-457,937,52,-636,925,827,914,938,928,153,66,-911,585,655,828,334,0,191,336,-773

MIPS-single-cycle

Αριθμοί	Πλήθος κύκλων προγράμματος	Πλήθος εντολών προγράμματος	CPI	Χρόνος εκτέλεσης (Κύκλοι * Χρόνος κύκλου)	Ενεργειακή αποδοτικότητα (απόδοση/ισχύς ή 1/[χρόνος*ισχύς])
200	4713	4713	1	0.00004713	1060.9
500	11819	11819	1	0.00011819	423.047

MIPS-pipeline-simple

Αριθμοί	Πλήθος κύκλων προγράμματος	Πλήθος εντολών προγράμματος	CPI	Χρόνος εκτέλεσης (Κύκλοι * Χρόνος κύκλου)	Ενεργειακή αποδοτικότητα (απόδοση/ισχύς ή 1/[χρόνος*ισχύς])
200	7935	5037	1.57	0.00002636	1517.45
500	19842	12597	1.57	0.00006592	606.8

MIPS-pipeline-turbo

Αριθμοί	Πλήθος κύκλων προγράμματος	Πλήθος εντολών προγράμματος	CPI	Χρόνος εκτέλεσης (Κύκλοι * Χρόνος κύκλου)	Ενεργειακή αποδοτικότητα (απόδοση/ισχύς ή 1/ [χρόνος*ισχύς])
200	5573	5037	1.11	0.00001857	1538.57
500	13735	12597	1.09	0.00004577	624.24