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

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.

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

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

MIPS-single-cycle

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

MIPS-pipeline-simple

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

MIPS-pipeline-turbo

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

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.

MIPS-single-cycle

Πλήθος Αριθμών	**Πλήθος κύκλων προγράμματος**	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/[χρόνος*ισχύς])
100	1896	1896	1	$1,896* 10^{-5}$	2637,130802
200	3733	3733	1	$3,733* 10^{-5}$	1339,405304
500	9353	9353	1	$9,353* 10^{-5}$	534,5878328

MIPS-pipeline-simple

Πλήθος Αριθμών	**Πλήθος κύκλων προγράμματος**	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/[χρόνος*ισχύς])
100	3330	1908	1,74528	$1,11* 10^{-5}$	3603,603604
200	6551	3745	1,74927	$2,183* 10^{-5}$	1832,340815
500	16391	9361	1,75099	$5,463* 10^{-5}$	732,1984258

MIPS-pipeline-turbo

Πλήθος Αριθμών	**Πλήθος κύκλων προγράμματος**	**Πλήθος εντολών προγράμματος**	**CPI**	**Χρόνος εκτέλεσης** (Κύκλοι * Χρόνος κύκλου)	**Ενεργειακή αποδοτικότητα** (απόδοση/ισχύς ή 1/[χρόνος*ισχύς])
100	2033	1908	1,06551	$6,776 * 10^{-6}$	4216,562658
200	4001	3745	1,06836	$1,333*10^{-5}$	2143,392991
500	9980	9361	1,06613	$3,326*10^{-5}$	859,0327291

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.]

Σχετικά με το αρχείο SingleCycle.s

Το τμήμα δεδομένων .data περιέχει 3 πίνακες Array1 ,Array2 ,Array3 των 100,200 και 500 64-bit ποσοτήτων σε μία γραμμή η καθεμία, χώρο 8 byte στα μέγιστα και ελάχιστα στοιχεία και έναν πίνακα των 8 byte με ένα στοιχείο -το μηδέν- για τα labels των μετρητών για άρτιους, περιττούς, μηδενικούς,div3,div5 αριθμών. Τέλος περιέχει 3 πίνακες των 8 byte ο καθένας για το μέγεθος (Size)κάθε array.

Το πρόγραμμα αρχίζει με το πεδίο _start που συνδέουμε τις ετικέτες με τους καταχωρητές στην μνήμη. Σε σχόλια υπάρχουν οι κατάλληλες κλήσεις για τους πίνακες των 200 (Array2) και των 500(Array3) ενώ οι t0,t1,t3,t5 αρχικοποιούνται με addi με τις τιμές Size,1,3,5 καθώς θα μας χρειαστούν μέσα στο πρόγραμμα.

Στο πεδίο loop που ακολουθεί φορτώνουμε στη μεταβλητή t6 τον πρώτο αριθμό του πίνακα. Στη συνέχεια εξετάζουμε εάν είναι μικρότερος του Min(Το Min έχει αρχικοποιηθεί με το πρώτο στοιχείο του πίνακα) και εφόσον η συνθήκη είναι αληθής φορτώνουμε το αριθμό ως Min και πηγαίνουμε στο πεδίο Odd_Even, αλλιώς με μία branch εντολή μεταφερόμαστε στον έλεγχο εάν ο αριθμός είναι μεγαλύτερος του μέγιστου Max ο οποίος έχει αρχικοποιηθεί παρομοίως με το Min στο πρώτο τμήμα του προγράμματος _start.

Στο τμήμα Odd_Even βλέπουμε τη συμπεριφορά του αριθμού στην bitwise πρόσθεση κατά ένα .Το αποτέλεσμα είναι 1 αν πρόκειται για άρτιο και 0 αν πρόκειται για περιττό αριθμό. Για την ώρα μας ενδιαφέρει μόνο η περίπτωση του άρτιου αριθμού οπότε πηγαίνουμε στην επόμενη branch ισότητας με το μηδέν($zero).Αν ο αριθμός είναι ίσος με μηδέν τότε αυξάνουμε τους μετρητές διαίρεσης με το 3,διαίρεσης με το 5 και αυτού των μηδενικών. Προφανώς πριν την διακλάδωση έχουμε αυξήσει τον κατάλληλο μετρητή των άρτιων.

Έπειτα γίνονται οι διαιρέσεις με το 3 και το 5 με το υπόλοιπο της διαίρεσης να αποθηκεύεται στους καταχωρητές t7 και t6.Εαν ο αριθμός διαιρείται ακριβώς δηλαδή το υπόλοιπο της διαίρεσης είναι μηδέν τότε αυξάνεται ο μετρητής div3 ,ανάλογη branch ισότητας με μηδέν υπάρχει και για το div5.

Τέλος της επανάληψης στο πεδίο Go_next όπου μειώνεται κατά ένα ο μετρητής της επαναληπτικής διαδικασίας, προσθέτουμε κατά 4 τη διεύθυνση του πίνακα (μεταφερόμαστε στο επόμενο στοιχείο αφού κάθε word απαιτεί offset 4) και γίνεται ο έλεγχος εάν ο μετρητής επανάληψης είναι ίσος με μηδέν και εφόσον δεν ισχύει αυτή η συνθήκη μεταφερόμαστε στο πεδίο loop όπου η παραπάνω διαδικασία επαναλαμβάνεται.

Αφού βγούμε από την loop του πίνακα στον μετρητή των περιττών αποθηκεύω το αποτέλεσμα της αφαίρεσης του μεγέθους του πίνακα με το πλήθος των άρτιων αριθμών μιας και ένας αριθμός θα είναι είτε άρτιος είτε περιττός και καλώ την break.

Σχετικά με το αρχείο Pipeline.s:

Ο συγκεκριμένος κώδικας χρησιμοποιήθηκε στους επεξεργαστές MIPS-pipeline-simple και MIPS-pipeline-turbo.

Η λογική ως προς τη προσέγγιση του προβλήματος ήταν η ίδια ωστόσο υπήρξαν διαφοροποιήσεις προκειμένου να αξιοποιήσουμε όσο το δυνατόν καλύτερα το Pipeline των 5 σταδίων. Για αυτόν τον λόγο έγινε αναδιάταξη των εντολών(reordering).

Το τμήμα δεδομένων .data και η αρχή του προγράμματος _start παρέμειναν ίδια. Οι διαφορές αφορούν κυρίως το τμήμα loop όπου ξεκινάω με τις load word ώστε να μπουν νωρίς στα στάδια της διοχέτευσης. Ακολουθούν οι slt εντολές και τέλος οι andi για τη bitwise πρόσθεση και addi για μείωση του loop counter .Έτσι, εντολές που προηγουμένως ήταν ακριβώς πριν από πράξεις and,add κ branch και που δημιουργούσαν φυσαλίδες μετά τη μετάβαση από single cycle σε pipeline παίρνουν τιμές αρκετά νωρίς στη loop και μειώνουν κατά πολύ τους κύκλους και τα data hazard stalls .

Προσθέσαμε και μερικά nop μετά από τις jump και μετά από τη τελική διακλάδωση που μας στέλνει στην αρχή του loop,καθώς αν η εντολή διαβάζεται στο στάδιο IF η επόμενη εντολή sub $s5,$t9,$s4 έχει ήδη τοποθετηθεί στο στάδιο ID λόγω του pipeline. Εφόσον δεν ισχύει η συνθήκη του branch και μεταφερόμαστε στο επόμενο πεδίο του προγράμματος η addi εντολή διακόπτεται δημιουργώντας branch taken stall.

Παράδειγμα από τον αρχικό κώδικα

```
71  Min:
72
73      slt $t2, $t6, $s1       #t2 will be 1 if number<min,else 0
74      beq $t2, $zero,Max      #if t2==0 check if the number is greater than the max
75      lw $s1,0($s0)           #min=number
76      j Odd_Even
77
78  Max:
79
80      slt $t4, $s2,$t6        #t4 will be 1 if number>max,else 0
81      beq $t4, $zero,Odd_Even #if t4 is zero then number>max so we have a new max
82      lw $s2,0($s0)           #max=number
83
```

Οι εντολές slt στις γραμμές 73 και 80 δεν προλαβαίνουν να μπουν στο WB στάδιο της διοχέτευσης για να χρησιμοποιηθούν στις αποκάτω branch εντολές με αποτέλεσμα να υπάρχουν data hazard stalls λόγω φόρτωσης/χρήσης.

Η αναδιάταξη του κώδικα ώθησε προς τα 'πάνω' αρκετές εντολές που εκχωρούν τιμές σε καταχωρητές οι οποίοι είναι απαραίτητη στη loop ανεξαρτήτου των συνθηκών που έχουμε ορίσει. Συγκεκριμένα, για οποιοδήποτε αριθμό του πίνακα θα ελέγξουμε αν είναι μεγαλύτερος του Max ,μικρότερος του Min, εάν είναι άρτιος ή περιττός και θα μειώσουμε σίγουρα κατά ένα το μετρητή της επαναληπτικής διαδικασίας(αλλιώς θα έχουμε ένα infinite loop).Η μεταφορά τους μακριά από την εντολή που κρίνεται απαραίτητη η προσπέλαση στη μνήμη τους μας βοηθάει να πετύχουμε αρκετά καλή απόδοση CPI και ελάχιστους κινδύνους δεδομένων.

Παρόλα αυτά παρατηρούμε πως εξαιτίας των αρκετών (εφτά) branch εντολών τα control hazards είναι υψηλά. Αρκετοί επίσης είναι και οι κύκλοι ενώ το CPI είναι αυξημένο σε σχέση με το single cycle .

Αυτό αναμένουμε να βελτιωθεί με τη προσθήκη της πρόβλεψης διακλάδωσης για κινδύνους ελέγχου και της προώθησης για κινδύνους δεδομένων. Συγκεκριμένα με το επεξεργαστή MIPS-pipeline-turbo η πρόβλεψη για τις διακλαδώσεις δεν επιτρέπει την φόρτωση εντολών στο pipeline εφόσον αυτές πρόκειται να διαγραφούν.

Pipeline-Simple

Cycle Statistics

Total Cycles:	3325
Instructions:	1900
CPI:	1.75
Data Hazard Stalls:	114
Control Hazard Stalls:	1311
RAM Stalls:	0
L1 Data Stalls:	0
L1 Program Stalls:	0
L2 Unified Stalls:	0

Pipeline-Turbo

Cycle Statistics

Total Cycles:	2013
Instructions:	1900
CPI:	1.05947
Data Hazard Stalls:	0
Control Hazard Stalls:	113
RAM Stalls:	0
L1 Data Stalls:	0
L1 Program Stalls:	0
L2 Unified Stalls:	0

Οι κύκλοι ήρθαν πιο κοντά στα επίπεδα προ-διοχέτευσης ενώ και το CPI πλησιάζει σε αρκετά μεγάλο βαθμό τη μονάδα.