ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ (5^{ο} εξάμηνο)

ΚΑΝΟΝΙΚΗ ΕΞΕΤΑΣΗ (ΜΑΡΤΙΟΣ 2008)
ΔΙΑΡΚΕΙΑ ΕΞΕΤΑΣΗΣ: 2 ΩΡΕΣ 30 ΛΕΠΤΑ

Οι εξετάσεις θα πραγματοποιηθούν ΧΩΡΙΣ την παρουσία βιβλίων, βοηθημάτων ή άλλου είδους σημειώσεων. Το μόνο που επιτρέπεται να έχετε μαζί σας είναι ένα φύλλο Α4 στο οποίο μπορείτε να έχετε γράψει ό,τι έχετε κρίνει πιο σημαντικό για το μάθημα και θέλετε να το έχετε ως βοήθημά σας. Απαγορεύεται η ανταλλαγή οποιουδήποτε αντικειμένου κατά την ώρα της εξέτασης, ούτε και των φύλλων Α4 που είναι ατομικά.

Καλή Επιτυχία!

Θέμα 1^{ο} (30%):

Εξετάζουμε την εκτέλεση του εξής κώδικα:

```
        lw   $s5,0($s1)
        addi $s5,$s5,1
        lw   $s6,8($s1)
        slt  $s2, $s5, $s6        ; if($s5<$s6) $s2=1 else $s2=0
        bne  $s2,0,L1
        subi $s5,$s5,1
        lw   $s6,16($s1)
L1:     sw   $s6,0($s1)
```

Υποθέτουμε ότι έχουμε αρχιτεκτονική σωλήνωσης (pipelining) 5 σταδίων (IF ID EX MEM WB), και ότι όλες οι αναφορές στη μνήμη ικανοποιούνται από την κρυφή μνήμη σε 1 κύκλο (δεν υπάρχουν δηλαδή αστοχίες). Επίσης, η εγγραφή σε κάποιον καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιον καταχωρητή στο δεύτερο μισό του ίδιου κύκλου. Επιπλέον, η ενημέρωση του μετρητή προγράμματος κατά την εκτέλεση μιας εντολής διακλάδωσης γίνεται στο στάδιο MEM, και για να γίνει η διακλάδωση πρέπει να "εκκενωθεί" (flush) το pipeline. Υποθέτουμε ότι η εντολή διακλάδωσης `bne` είναι NOT TAKEN.

α) Αρχικά, υποθέτουμε ότι η αρχιτεκτονική σωλήνωσης δε διαθέτει σχήμα προώθησης (forwarding). Χρησιμοποιείστε ένα διάγραμμα χρονισμού για να δείξετε τα διάφορα στάδια του pipeline από τα οποία διέρχονται οι εντολές του παραπάνω κώδικα. Υποδείξτε και εξηγείστε τους πιθανούς κινδύνους (hazards) που μπορούν να προκύψουν κατά την εκτέλεση, καθώς και τον τρόπο με τον οποίον αυτοί αντιμετωπίζονται. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του κώδικα;

β) Για την ίδια ακολουθία εντολών, δείξτε και εξηγείστε όπως και πριν τον χρονισμό του pipeline, θεωρώντας όμως τώρα ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης. Πόσοι κύκλοι απαιτούνται

για την εκτέλεση του κώδικα; Είναι δυνατόν αναδιατάσσοντας τις εντολές -χωρίς να αλλάξει η σημασιολογία του προγράμματος- να επιτύχουμε λιγότερους κύκλους;

γ) Υποθέτουμε ότι διαθέτουμε αρχιτεκτονική σωλήνωσης 6 σταδίων (IF ID EX M1 M2 WB), όπου απαιτούνται 2 κύκλοι ανάγνωσης ή εγγραφής στη μνήμη, ενώ ισχύουν οι υπόλοιπες υποθέσεις του ερωτήματος β. Δείξτε και εξηγείστε όπως στα προηγούμενα ερωτήματα τον χρονισμό του pipeline. Θεωρείστε ότι η τελική απόφαση για μια διακλάδωση υπό συνθήκη λαμβάνεται στο στάδιο M1 (ενημέρωση PC).

Θέμα 2^{ο} (20%):

α) Έστω δύο διαδοχικές εντολές i και j. Λέμε ότι εμφανίζεται *write-after-write* (WAW) κίνδυνος μεταξύ των εντολών αυτών, όταν η j προσπαθεί να γράψει έναν τελεστέο (καταχωρητή ή θέση μνήμης) προτού η i γράψει σε αυτόν. Αυτό έχει σαν αποτέλεσμα η τελική τιμή του τελεστέου να είναι αυτή που γράφεται από την i, κάτι που παραβιάζει τη σωστή σειρά εκτέλεσης του προγράμματος. Εξηγείστε γιατί αυτό το είδος κινδύνων δεν είναι δυνατόν να εμφανιστεί στην αρχιτεκτονική αγωγού 5 σταδίων του MIPS. Αναφέρατε μια πιθανή περίπτωση σωλήνωσης όπου θα μπορούσε να εμφανιστεί ο κίνδυνος αυτός.

β) Έστω ότι οι εντολές αναφοράς στη μνήμη (load, store) δεν επιτρέπεται να περιέχουν το σταθερό όρισμα (offset). Δηλαδή επιτρέπεται να έχουν μόνο τη μορφή:

```
lw $s1,($s2)
```

για τη λειτουργία:

```
$s1 = Mem[$2]
```

- Πώς επηρεάζει η συγκεκριμένη τροποποίηση του συνόλου εντολών (ISA) την υλοποίηση της αρχιτεκτονικής αγωγού; Θα μπορούσε να μειωθεί το βάθος της σωλήνωσης; Αν ναι, πώς; Τι συνέπειες θα είχαν οι όποιες αλλαγές σε throughput και latency; Αν όχι, γιατί;
- Αναφέρατε δύο διαφορετικές περιπτώσεις (ακολουθίες εντολών) όπου ενδέχεται να εμφανισθούν κίνδυνοι δεδομένων καθώς και τις αντίστοιχες αναγκαίες προωθήσεις δεδομένων.

Θέμα 3^{ο} (30%):

α) Εξηγείστε τις έννοιες της *χωρικής τοπικότητας* και *χρονικής τοπικότητας* αναφορών στη μνήμη.

β) Περιγράψτε τα γενικά χαρακτηριστικά ενός προγράμματος που θα παρουσίαζε πολύ υψηλά ποσοστά χρονικής τοπικότητας αλλά πολύ μικρή χωρική τοπικότητα ως προς τις αναφορές σε δεδομένα στη μνήμη. Ομοίως, περιγράψτε τα γενικά χαρακτηριστικά ενός προγράμματος που θα παρουσίαζε πολύ μικρή χρονική τοπικότητα αλλά πολύ υψηλά ποσοστά χωρικής τοπικότητας ως προς τις αναφορές σε δεδομένα. Για κάθε περίπτωση, δώστε ένα παράδειγμα προγράμματος (με ψευδοκώδικα).

γ) Θεωρήστε ένα σύστημα μνήμης με μία 8-way set associative cache μεγέθους 512KB, και με cache line 8 λέξεων. Το μέγεθος της λέξης είναι 32 bits. Η μικρότερη μονάδα δεδομένων που μπορεί να διευθυνσιοδοτηθεί είναι το 1 byte, ενώ οι διευθύνσεις μνήμης έχουν εύρος 64 bit.

γ1) Για τα επιμέρους πεδία στα οποία χωρίζεται μία διεύθυνση μνήμης σε μία τέτοια οργάνωση cache, υπολογίστε τον αριθμό των bits του καθενός. Παρουσιάστε ένα διάγραμμα που να δείχνει πώς διαχωρίζεται η διεύθυνση στα πεδία αυτά, και εξηγείστε τη σημασία του καθενός. Σχεδιάστε το συνολικό σύστημα κρυφής μνήμης.
γ2) Σε ποιες θέσεις της cache μπορεί να απεικονιστεί το byte στη διεύθυνση μνήμης $300A21_{16}$;
γ3) Ποιες θέσεις μνήμης μπορούν να απεικονιστούν στο σύνολο 18 της cache;
γ4) Τι ποσοστό του συνολικού μεγέθους της cache αφιερώνεται για τα bits του tag;

Θέμα 4ο (20%):

Θεωρήστε το ακόλουθο κομμάτι κώδικα:

```
int i,j;
float tmp, x[11][8];

for(i=0; i<8; i++)
     for(j=0; j<10; j++)
          tmp += x[j][i]/x[j+1][i];
```

Ο πίνακας x περιέχει στοιχεία κινητής υποδιαστολής μονής ακρίβειας, μεγέθους 4 bytes. Κάνουμε τις εξής υποθέσεις:
- Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων.
- Η κρυφή μνήμη είναι πλήρως συσχετιστική (fully associative), αποτελείται από 10 cache lines δεδομένων, και έχει LRU πολιτική αντικατάστασης. Το μέγεθος του cache line είναι 32 bytes.
- Υποθέτουμε ότι όλες οι μεταβλητές, πλην των στοιχείων του πίνακα x, μπορούν να αποθηκευτούν σε καταχωρητές του επεξεργαστή, οπότε οποιαδήποτε αναφορά σε αυτές δεν συνεπάγεται προσπέλαση στην κρυφή μνήμη.
- Ο επεξεργαστής στέλνει προς εκτέλεση τα loads του προγράμματος, με τη σειρά που αυτά εμφανίζονται στο πρόγραμμα (δηλαδή, πρώτα εκτελεί το load για το στοιχείο x[j][i] και μετά για το x[j+1][i]).
- Ο πίνακας είναι αποθηκευμένος στην κύρια μνήμη κατά γραμμές. Επιπλέον, είναι "ευθυγραμμισμένος" ώστε το πρώτο στοιχείο του να απεικονίζεται στην αρχή μιας γραμμής της κρυφής μνήμης.
- Αρχικά, η κρυφή μνήμη δεδομένων είναι άδεια.

α) Βρείτε ποιες από τις αναφορές στα στοιχεία του πίνακα x για όλη την εκτέλεση του παραπάνω κώδικα καταλήγουν σε misses στην cache. Υποδείξτε ποια είναι compulsory, ποια είναι capacity και ποια conflict. Δώστε τον συνολικό αριθμό των misses.

β) Πώς θα ξαναγράφατε τον κώδικα ώστε να μειωθούν τα misses; Υπολογίστε εκ νέου τον αριθμό των misses.