ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Επαναληπτική Εξέταση Σεπτεμβρίου 2010
Διάρκεια 2,5 ώρες

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.

Θέμα 1ο (20%)

Α) Για κάθε μια από τις παρακάτω αλλαγές απαντήστε αν θα μειωθούν, αυξηθούν ή παραμείνουν αμετάβλητα τα Instructions/Program, Cycles/Instruction και Seconds/Cycle. Εξηγήστε σύντομα τις απαντήσεις σας.

(i) Ένωση 2 σταδίων της σωλήνωσης.
(ii) Αύξηση της συχνότητας του ρολογιού.
(iii) Χρήση καλύτερου compiler.
(iv) Μείωση των διαδρομών προώθησης.
(v) Αύξηση της ταχύτητας πρόσβασης στη μνήμη.
(vi) Μείωση του αριθμού των καταχωρητών.

Β) Μια από τις επεκτάσεις του σετ εντολών του MIPS ορίζει τις εντολές `movn` (move if not zero) και `movz` (move if zero). Έτσι, η εντολή

```
movn $8, $11, $4
```

μετακινεί τα περιεχόμενα του καταχωρητή 11 στον καταχωρητή 8 εφόσον η τιμή του καταχωρητή 4 είναι μη μηδενική (διαφορετικά δεν κάνει τίποτα). Αντίστοιχα η `movz` θα πραγματοποιήσει τη μεταφορά αν η τιμή του καταχωρητή 4 είναι μηδενική.

(i) Χρησιμοποιήστε τις νέες εντολές για να μεταφέρετε στον καταχωρητή 10 όποια τιμή είναι μεγαλύτερη μεταξύ των περιεχομένων των καταχωρητών 8 και 9. Αν οι τιμές είναι ίσες τότε μπορείτε να μεταφέρετε όποια τιμή από τις 2 θέλετε. Επιτρέπεται να χρησιμοποιήσετε μόνο τον καταχωρητή $1 ως έξτρα καταχωρητή για αποθήκευση προσωρινών δεδομένων. Επίσης απαγορεύεται η χρήση conditional branches.
(ii) Οι εντολές αυτές επιτρέπουν συνήθως την εκτέλεση λιγότερων εντολών σε ένα πρόγραμμα από ότι αν χρησιμοποιούνταν conditional branches. Ακόμα όμως και αν ο αριθμός των εντολών δεν μειωθεί, το πρόγραμμα εκτελείται ταχύτερα εφόσον ο επεξεργαστής χρησιμοποιεί αρχιτεκτονική σωλήνωσης. Εξηγήστε γιατί.

Θέμα 2ο (30%)

Δίνεται το ακόλουθο κομμάτι κώδικα :

```
1.    lw    r3, 100(r2)
2.    add   r1, r2, r3
3.    lw    r4, 0(r1)
4.    add   r5, r4, r6
5.    add   r7, r5, r8
6.    add   r1, r2, r3
7.    lw    r4, 0(r1)
8.    add   r5, r1, r6
```

Α) Βρείτε όλες τις εξαρτήσεις στο παραπάνω κομμάτι κώδικα καθώς και την κατηγορία στην οποία ανήκουν (true / output / anti / control dependence).

Β) Υποθέστε την κλασσική αρχιτεκτονική σωλήνωσης 5 σταδίων του MIPS χωρίς σχήμα προώθησης. Χρησιμοποιήστε ένα διάγραμμα χρονισμού για να δείξετε τα διάφορα στάδια της σωλήνωσης από τα οποία διέρχονται οι παραπάνω εντολές. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του κώδικα;

Γ) Υποθέστε την ίδια αρχιτεκτονική με το Β, αλλά με όλα τα δυνατά σχήματα προώθησης. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα; Δείξτε όπως και πριν το διάγραμμα χρονισμού, υποδεικνύοντας τις προωθήσεις που γίνονται.

Δ) Κάποιος αρχιτέκτονας σας προτείνει να προσθέσετε μια δεύτερη αριθμητική μονάδα ALU2 στη σωλήνωση και συγκεκριμένα στο στάδιο MEM, το οποίο μετατρέπεται σε MEM/EX2. Με αυτή την προσθήκη οι εντολές πρόσβασης στη μνήμη λειτουργούν όπως και πριν, δηλαδή υπολογίζουν στο στάδιο EX τη διεύθυνση τους χρησιμοποιώντας την ALU1 και πραγματοποιούν την προσπέλαση στο στάδιο MEM. Οι αριθμητικές εντολές μπορούν να χρησιμοποιήσουν είτε την ALU1 είτε την ALU2, ***ποτέ όμως και τις 2***. Πιο συγκεκριμένα, αν στο τέλος του σταδίου ID τα ορίσματα της εντολής είναι διαθέσιμα (είτε μέσω του register file είτε μέσω προώθησης), τότε η εντολή θα χρησιμοποιήσει την ALU1 στο στάδιο EX. Διαφορετικά, θα χρησιμοποιήσει υποχρεωτικά την ALU2 στο στάδιο MEM/EX2. Η σωλήνωση δηλαδή έχει την εξής λειτουργικότητα :

IF	ID	EX	MEM/EX2	WB
Instruction Fetch	Instruction Decode and register read	ALU1 execution or address calculation	ALU2 execution or memory access	Write back to register file

Τα σχήματα προώθησης παραμένουν ίδια με την κλασσική αρχιτεκτονική του MIPS. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα; Δείξτε όπως και πριν το διάγραμμα χρονισμού, υποδεικνύοντας τις προωθήσεις που γίνονται. Ποια αριθμητική μονάδα χρησιμοποιεί η κάθε εντολή πρόσθεσης;

Θέμα 3ο (25%)

Α) Δίνονται δύο fully associative caches, εκ των οποίων η πρώτη χρησιμοποιεί την LRU πολιτική αντικατάστασης ενώ η δεύτερη την FIFO. Βρείτε μια αλληλουχία προσβάσεων στη μνήμη όπου η FIFO είναι πιο αποδοτική (έχει λιγότερα misses) από την LRU. Υποθέστε ότι κάθε cache έχει 4 blocks, το κάθε block έχει μέγεθος 1 word και ότι η μνήμη είναι word-addressable.

Β) Δίνεται το παρακάτω κομμάτι κώδικα:

```
total = 0;
for(j = 0; j < k; j++) {
      sub_total = 0;
      for(i = 0; i < N; i++) {
            sub_total += A[j*N + i];
      }
      total += sub_total / N;
}
average = total / k;
```

Σαν αρχιτέκτονες καλείστε να επιλέξετε την κατάλληλη cache για την εκτέλεση αυτού του κώδικα. Για δεδομένη χωρητικότητα και associativity θα επιλέγατε μεγάλο ή μικρό block size; Γιατί;

Γ) Υποθέστε μια cache και ότι οι διάφοροι παράμετροι (μέγεθος, associativity, line size) παραμένουν σταθεροί εκτός από αυτή που αλλάζει σε κάθε υποπερίπτωση. Απαντήστε αν οι παρακάτω προτάσεις είναι αληθείς ή όχι. Εξηγήστε συνοπτικά τις απαντήσεις σας :

(i) Διπλασιασμός του line size θα μειώσει στο μισό τον αριθμό των tags της cache.
(ii) Διπλασιασμός του associativity θα διπλασιάσει τον αριθμό των tags της cache.
(iii) Διπλασιασμός της χωρητικότητας σε μια direct-mapped cache συνήθως μειώνει τα conflict misses.
(iv) Διπλασιασμός της χωρητικότητας σε μια direct-mapped cache συνήθως μειώνει τα compulsory misses.
(v) Διπλασιασμός του line size συνήθως μειώνει τα compulsory misses.

Θέμα 4ο (25%)

Δίνεται ο παρακάτω κώδικας γραμμένος σε C.

```
double A[4], B[32];

for(i = 0; i < 3; i++) {
      A[i] = 0;
      for(j = 0; j < 8; j++)
            A[i] = A[i] + B[i*8 + j];
}
```

Οι πίνακες περιέχουν στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 bytes το καθένα. Κάνουμε τις εξής υποθέσεις:

1. Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι direct-mapped, write-allocate και αποτελείται από 4 blocks δεδομένων. Το μέγεθος του block είναι 32 bytes, ενώ η μικρότερη μονάδα δεδομένων που μπορεί να διευθυνσιοδοτηθεί είναι το 1 byte.

2. Υποθέτουμε ότι όλες οι μεταβλητές, πλην των στοιχείων των πινάκων, μπορούν να αποθηκευτούν σε καταχωρητές του επεξεργαστή, οπότε οποιαδήποτε αναφορά σε αυτές δεν συνεπάγεται προσπέλαση στην κρυφή μνήμη. Επίσης, σε επίπεδο εντολών assembly οι αναγνώσεις γίνονται με τη σειρά που εμφανίζονται στον κώδικα.

3. Οι πίνακες είναι αποθηκευμένοι στην κύρια μνήμη κατά γραμμές. Το πρώτο στοιχείο του πίνακα A βρίσκεται στη διεύθυνση 0x000a07a0 και του πίνακα B στη διεύθυνση 0x000a3f20.

Α) Βρείτε ποιες από τις αναφορές στα στοιχεία των πινάκων, για όλη την εκτέλεση του παραπάνω κώδικα, καταλήγουν σε misses. Υποδείξτε ποια είναι compulsory, ποια είναι capacity και ποια conflict.

Β) Αν η κρυφή μνήμη γίνει write no allocate, διατηρώντας τα υπόλοιπα χαρακτηριστικά της ίδια, ποιός θα είναι ο νέος ρυθμός αστοχίας;

Γ) Αν η κρυφή μνήμη αντικατασταθεί με μια κρυφή μνήμη 2-way associative, write allocate, με ίδιο συνολικό αριθμό blocks δεδομένων και ίδιο μέγεθος blocks, ποιος θα είναι ο νέος ρυθμός αστοχίας; Υποθέστε ότι η cache χρησιμοποιεί LRU πολιτική αντικατάστασης.