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

19/10/07

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Επαναληπτική Εξέταση Οκτωβρίου 2007
Διάρκεια 3 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 (35%):

Έστω το ακόλουθο τμήμα C κώδικα:

```
for (i = 0; i < 20; i++)
     if (array[i] < j)
          array[i] = j;
     else
          array[i] = array[i]+4;
```

Θεωρούμε ότι οι τιμές των μεταβλητών `i,4·i,j` βρίσκονται στους καταχωρητές `r1,r2,r8` και η διεύθυνση του πίνακα `array` στον `r3`. Το περιεχόμενο των καταχωρητών `r1` και `r4` είναι αρχικά 0 και 20, αντίστοιχα. Ο αντίστοιχος κώδικας σε assembly είναι ο εξής:

```
  Loop:     sll   r2, r1, #2          ;r2 = r1 << 2
            add   r5, r2, r3
            lw    r6, 0(r5)
            slt   r7, r6, r8          ;if(r6<r8) r7=1 else r7=0
            beqz  r7, Else
            sw    r8, 0(r5)
            j     Next
  Else:     addi  r6, r6, #4
            sw    r6, 0(r5)
Next:       addi  r1, r1, #1
            bne   r1, r4, Loop
  Exit:
```

α) Δείξτε το διάγραμμα εκτέλεσης των επιμέρους φάσεων για κάθε εντολή μέσα στην αρχιτεκτονική αγωγού 5 σταδίων του MIPS, υποθέτοντας την ύπαρξη όλων των δυνατών σχημάτων προώθησης. Σημειώστε όλες τις αναγκαίες προωθήσεις δεδομένων. Υποθέστε επίσης ότι για την πρώτη εντολή διακλάδωσης υπό συνθήκη υπάρχει πρόβλεψη διακλάδωσης που είναι πάντα NOT TAKEN, ενώ σε περίπτωση εσφαλμένης πρόβλεψης το pipeline πρέπει να καθαριστεί ("flush"). Επιπλέον, στο 40% των περιπτώσεων ισχύει η ανισότητα `array[i]<j`. Πόσους κύκλους χρειάζεται για να εκτελεστεί ο παραπάνω βρόχος;

β) Θεωρείστε ότι επεκτείνουμε το σύνολο εντολών του MIPS προσθέτοντας δύο νέες εντολές υπό συνθήκη, τις: `swn` (store if not zero) και `swz` (store if zero), οι οποίες πραγματοποιούν τις εξής λειτουργίες:

```
swn r1, r2, r3        ;if (r1!=0) sw r2,0(r3)
swz r1, r2, r3        ;if (r1==0) sw r2,0(r3)
```

Δηλαδή, αν η τιμή του καταχωρητή `r1` είναι διάφορη του μηδενός (`swn`), αποθηκεύει στην διεύθυνση `(r3)` την τιμή του καταχωρητή `r2`. Διαφορετικά, λειτουργεί σαν εντολή `nop`, δηλαδή δεν κάνει τίποτα. Ομοίως για την εντολή `swz`.

Σχεδιάστε ξανά τον παραπάνω assembly κώδικα χρησιμοποιώντας τις νέες εντολές υπό συνθήκη, ώστε να αποφύγετε κατά το δυνατό τις υποχρεωτικές καθυστερήσεις που εισάγονται στη σωλήνωση στη περίπτωση α.

Στη συνέχεια, για την παραπάνω αρχιτεκτονική αγωγού με ύπαρξη σχήματος προώθησης, δείξτε το διάγραμμα εκτέλεσης των επιμέρους φάσεων για κάθε εντολή και σημειώστε όλες τις αναγκαίες προωθήσεις δεδομένων. (Ισχύουν οι υποθέσεις του ερωτήματος α σε ό,τι αφορά τις εντολές διακλάδωσης και τον έλεγχο της ανισότητας `array[i]<j`). Πόσους κύκλους χρειάζεται για να εκτελεστεί ο παραπάνω βρόχος; Συγκρίνετε τη διαφορά επίδοσης μεταξύ του αρχικού κώδικα (ερώτημα α) και του νέου. Η βελτίωση οφείλεται σε μείωση του αριθμού των εντολών του assembly κώδικα; Αν όχι, πώς εξηγείται;

γ) Στον κώδικα που προέκυψε από το προηγούμενο ερώτημα, εφαρμόστε την τεχνική του ξεδιπλώματος βρόχων (loop unrolling) μία φορά. Χρησιμοποιείστε όσο το δυνατόν λιγότερες εντολές. Συγκρίνετε τη διαφορά επίδοσης μεταξύ του νέου κώδικα και του κώδικα του προηγούμενου ερωτήματος, λαμβάνοντας υπόψη μόνο το συνολικό αριθμό εκτελούμενων εντολών.

δ) Έστω ότι επιθυμούμε την προσθήκη δύο νέων εντολών:

1. Μίας εντολής αποθήκευσης (`sta`), η οποία διαφέρει ως προς την υπάρχουσα `sw` στο ότι η διεύθυνση αποθήκευσης υπολογίζεται σαν το άθροισμα δύο καταχωρητών.
   Δηλαδή η εντολή: `sta r3, r4, r5`
   πραγματοποιεί την εξής λειτουργία: `Mem[r4 + r5] = r3`
2. και μίας εντολής φόρτωσης-ενημέρωσης (`lwu`), η οποία εκτός από τη συνηθισμένη ανάγνωση από τη μνήμη πραγματοποιεί και ενημέρωση ενός από τους καταχωρητές με τη υπολογιζόμενη διεύθυνση.
   Δηλαδή η: `lwu  r3, 8(r4)`
   πραγματοποιεί τις λειτουργίες:
   ```
   r3 = Mem[r4 + 8]
   r4 = r4 + 8;
   ```

Περιγράψτε ποιες είναι οι επιπρόσθετες δομικές μονάδες που απαιτούνται για την υλοποίηση των εντολών αυτών στην αρχιτεκτονική αγωγού 5 σταδίων του MIPS. Τι συνέπειες θα είχαν οι όποιες αλλαγές για την υλοποίηση των νέων εντολών (σε throughput, latency και hazards);

Θέμα 2ο (35%):

α) Εξηγείστε συνοπτικά τις πολιτικές *write through* - *write back* και *write allocate* - *no write allocate* που αφορούν τις εγγραφές στη μνήμη.

β) Αν ένας επεξεργαστής δε διέθετε κρυφή μνήμη με write allocate πολιτική, περιγράψτε ένα σενάριο όπου αυτό θα μπορούσε να οδηγήσει σε χαμηλή απόδοση. Αντίστοιχα, αν υποθέσουμε ότι ο επεξεργαστής διαθέτει κρυφή μνήμη με write allocate πολιτική, περιγράψτε ένα άλλο σενάριο όπου αυτό θα μπορούσε να οδηγήσει σε χαμηλή απόδοση. Σε κάθε περίπτωση, αναφέρετε τις υποθέσεις που κάνετε για το υποσύστημα μνήμης του επεξεργαστή.

γ) Θεωρήστε την εκτέλεση του ακόλουθου βρόχου:

```
for (i=0; i<128; i++)
     A[i] = A[i] + B[i]
```

Κάθε στοιχείο των πινάκων Α και Β είναι 8 bytes. Έστω ότι για τα στοιχεία Α[0] και Β[0], οι διευθύνσεις του πρώτου byte τους είναι οι (δίνονται σε 16-δική μορφή): 00001000 και 00008000, αντίστοιχα. Έστω ότι έχουμε ένα επίπεδο κρυφής μνήμης δεδομένων μεγέθους 128 bytes. Επιπλέον, οι εντολές σε επίπεδο assembly του παραπάνω βρόχου είναι διατεταγμένες με τέτοιο τρόπο ώστε γίνεται πρώτα ανάγνωση του Α[i] και έπειτα του Β[i].
Για κάθε μία από τις ακόλουθες περιπτώσεις οργάνωσης της cache, βρείτε τον αριθμό από *read hits*, *read misses*, *write hits* και *write misses* για τις διάφορες αναφορές που γίνονται στη μνήμη, εξηγώντας τις απαντήσεις σας.

1. Ευθείας αντιστοίχισης (direct mapped), write through, no write allocate, με μέγεθος block ίσο με 32 bytes.
2. Ευθείας αντιστοίχισης, write back, write allocate, με μέγεθος block ίσο με 32 bytes.
3. Συσχέτισης 2 δρόμων (2-way set associative), write through, no write allocate, με μέγεθος block ίσο με 32 bytes και πολιτική αντικατάστασης FIFO.
4. Πλήρως συσχετιστική (fully associative), write through, no write allocate, με μέγεθος block ίσο με 16 bytes και πολιτική αντικατάστασης FIFO.

Θέμα 3ο (30%):

Εξετάζουμε την εκτέλεση του ακόλουθου προγράμματος:

```
#define N 4
double a[N][N], b[N][N] ;
int i,j;

for(i=0; i<N; i++)
     for(j=0; j<N; j++)
          a[i][j] = a[i][j] + b[j][i];
```

Όλοι οι πίνακες περιέχουν στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 bytes το καθένα. 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) με LRU πολιτική αντικατάστασης, write-allocate, και αποτελείται από 4 blocks των 32 bytes το καθένα. Αρχικά, η κρυφή μνήμη δεδομένων είναι άδεια.

- 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.
- Σε επίπεδο εντολών assembly, η σειρά με την οποία γίνονται οι αναφορές στους πίνακες είναι η εξής: a, b, a.

α) Βρείτε το συνολικό ποσοστό αστοχίας (miss rate) για τις αναφορές που γίνονται στη μνήμη στον παραπάνω κώδικα.

β) Ποια/ες από τις ακόλουθες τεχνικές βελτιστοποίησης βρόχων θα μπορούσαν να εφαρμοστούν στον παραπάνω κώδικα προκειμένου να μειώσουμε το ποσοστό αστοχίας του;

1. *loop blocking*
2. *loop interchange*
3. *loop unrolling*

Εξηγείστε την απάντησή σας, παρουσιάζοντας το βελτιστοποιημένο κώδικα και υπολογίζοντας το νέο ποσοστό αστοχίας.