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

19 Φεβρουαρίου 2003

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΟΡΓΑΝΩΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Εξετάσεις Φεβρουαρίου 2003

Επώνυμο:	
Όνομα:	
Εξάμηνο:	

Θέμα	Βαθμός
1	
2	
3	
4	

Μην ξεχάσετε να αναγράψετε το ονοματεπώνυμο σας σε όλες τις κόλλες. Η εξέταση γίνεται με ανοικτά βιβλία και σημειώσεις. Διάρκεια εξέτασης 2 ½ ώρες. Ο βαθμός του διαγωνίσματος είναι το 70% του τελικού βαθμού προαγωγής στο μάθημα.

Θέμα 1ο (20%):

Έστω το τμήμα του C κώδικα:

```
while (save[i] == k)
     i = i + j;
```

Θεωρούμε ότι οι τιμές των μεταβλητών `i, j, k` βρίσκονται στους καταχωρητές `$s3, $s4, $s5` και η διεύθυνση του πίνακα `save` στον `$s6` Ο αντίστοιχος κώδικας σε assembly είναι ο εξής:

```
Loop:     add  $t1, $s3, $s3
          add  $t1, $t1, $t1
          add  $t1, $t1, $s6
          lw   $t0, 0($t1)
          bne  $t0, $s5, Exit
          add  $s3, $s3, $s4
          j    Loop
Exit:
```

Κάθε φορά που διατρέχει το loop χρησιμοποιούνται 2 εντολές άλματος (μία υπό συνθήκη και μία χωρίς συνθήκη). Ωστόσο μόνο μη βελτιστοποιημένοι compilers θα παρήγαγαν κώδικα με τέτοιο κόστος. Ξαναγράψτε το τμήμα αυτό του κώδικα έτσι ώστε να εκτελείται το πολύ μία εντολή άλματος (χωρίς ή με συνθήκη) εντός του βρόχου. Πόσες εντολές εκτελούνται πριν και μετά τη βελτιστοποίηση αν ο αριθμός των επαναλήψεων του βρόχου είναι 10. (Θεωρείστε ότι `save[i+10*j]` ≠ `k` και `save[i], … , save[i+9*j] = k`).

Θέμα 2ο (35%):

Η τεχνική loop unrolling (ξεδίπλωμα βρόχου) είναι μια από τις πιο διαδεδομένες τεχνικές βελτιστοποίησης κώδικα. Στην τεχνική αυτή, διαδοχικές επαναλήψεις του αρχικού βρόχου γράφονται σαν ξεχωριστές εντολές (π.χ. 2 διαδοχικές επαναλήψεις, οπότε ο unrolled βρόχος έχει πλέον βήμα 2).

```
for (i=0;i<12;i++) a(i) = a(i) + 8;
```

γίνεται:

```
for (i=0;i<12;i+=2) {
                a(i) = a(i) + 8;
                a(i+1) = a(i+1) + 8;
                }
```

Εδώ, θέλουμε να αξιολογήσουμε τα πλεονεκτήματα υλοποίησης της τεχνικής ξεδιπλώματος βρόχων σε κώδικα που εκτελείται σε αρχιτεκτονική MIPS με χρήση αγωγού (pipeline datapath) και ύπαρξη σχήματος προώθησης (forwarding).

Ο αρχικός μας κώδικας σε MIPS assembly είναι ο ακόλουθος (ο `$s1` έχει αρχικά την τιμή 48):

```
Loop:   lw   $t0, 0($s1)
        add  $t0, $t0, $s2
        sw   $t0, 0($s1)
        addi $s1, $s1, -4
        bne  $s1, $zero, Loop
```

A

α) Δείξτε το διάγραμμα εκτέλεσης των επιμέρους φάσεων για κάθε εντολή, μέσα στην αρχιτεκτονική αγωγού, υποθέτοντας την ύπαρξη σχήματος προώθησης (διάγραμμα εκτέλεσης όπως στις διαφάνειες του μαθήματος: IF - ID -…κλπ). Μπορείτε να το βελτιστοποιήσετε (αποφυγή τυχόν stalls) ?

β) Παρακάτω είναι το τμήμα του αρχικού κώδικα, όπου ο βρόχος είναι ξεδιπλωμένος μία φορά :

```
Loop:   lw   $t0, 0($s1)
        add  $t0, $t0, $s2
        sw   $t0, 0($s1)
        lw   $t1, -4($s1)
        add  $t1, $t1, $s2
        sw   $t1, -4($s1)
        addi $s1, $s1, -8
        bne  $s1, $zero, Loop
```

B

Σχεδιάστε ξανά τον κώδικα ώστε να αποφύγετε τα υποχρεωτικά memory stalls. Λαμβάνοντας υπόψη τα αναγκαία memory stalls συγκρίνετε τη διαφορά επίδοσης μεταξύ του αρχικού μη-ξεδιπλωμένου κώδικα (A) και του δικού σας (μετά το ξεδίπλωμα και τη βελτιστοποίηση του κώδικα B).

Θέμα 3ο (25%):

Το ακόλουθο πρόγραμμα εκτελείται σε μηχάνημα με κρυφή μνήμη οργανωμένη σε blocks με 4 λέξεις (words, με μήκος word = 4 bytes) χωρητικότητας 256 bytes δεδομένων:

```
int i, j, c, stride, array[256];
…
for (i=0; i<10000; i++)
   for (j=0; j<256; j+=stride)
      array[j] = c + 5;
```

Εξετάζουμε τη δραστηριότητα της κρυφής μνήμης που αφορά μόνο τις αναφορές στα στοιχεία του πίνακα `array`. Αν θεωρήσετε επιπλέον ότι κάθε ακέραιος καταλαμβάνει χώρο ίσο με 1 λέξη, ποιος είναι ο αναμενόμενος αριθμός αναφορών στη κρυφή μνήμη που είναι επιτυχείς (hits), όταν `stride=33`; Τι συμβαίνει όταν `stride=30`; Ποιες αλλαγές θα προκύψουν αν η κρυφή μνήμη είναι συσχέτισης 2-δρόμων (2-way set associative); Θεωρούμε ότι το στοιχείο `array[0]` αποθηκεύεται στο block 0 της κρυφής μνήμης και ότι η μνήμη είναι write allocate.

Θέμα 4ο (20%):

Δίνεται μια σειρά αναφορών σε διευθύνσεις λέξεων στη μνήμη ενός υπολογιστή: 2, 7, 11, 5, 25, 32, 15, 49, 11, 10, 12, 2, 13, 7, 48, 3. Υποθέτουμε ότι έχουμε κρυφή μνήμη με οργάνωση:

i) απευθείας απεικόνισης (direct mapped) με 8 blocks, όπου κάθε block έχει μέγεθος μια λέξη (word),

ii) απευθείας απεικόνισης (direct mapped), με 8 λέξεις (words) συνολικό μέγεθος cache, όπου κάθε block έχει μέγεθος δύο (2) λέξεις.

iii) συνόλου συσχέτισης 2-δρόμων (2-way set associative) με συνολικό μέγεθος 8 λέξεις (words) όπου κάθε block έχει μέγεθος μια λέξη (Υποθέστε LRU αλγόριθμο αντικατάστασης).

Δείξτε για τις παραπάνω περιπτώσεις οργάνωσης της κρυφής μνήμης, για κάθε αναφορά, αν είναι επιτυχής (hit) ή όχι (miss) καθώς και την τελικά περιεχόμενα της κρυφής μνήμης.