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

21 Σεπτεμβρίου 2006

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΟΡΓΑΝΩΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Εξετάσεις Σεπτεμβρίου 2006

Επώνυμο:	
Όνομα:	
Εξάμηνο:	

Θέμα	Βαθμός
1	
2	
3	

Μην ξεχάσετε να γράψετε το ονοματεπώνυμο σας σε όλες τις κόλλες. Η εξέταση γίνεται με ανοικτά βιβλία και σημειώσεις. Διάρκεια εξέτασης 2 ½ ώρες. Το διαγώνισμα βαθμολογείται με άριστα 10 μονάδες. Στον τελικό βαθμό προαγωγής του μαθήματος προστίθεται 1 μονάδα (bonus) αν έχετε παραδώσει μία ικανοποιητική λύση στις δύο σειρές που μοιράσθηκαν στο μάθημα (0,5 μονάδες για κάθε σειρά ασκήσεων).

Θέμα 1ο (35%):

Εξετάζουμε την εκτέλεση του παρακάτω κώδικα ο οποίος αντιγράφει ένα "null-terminated" string. Οι καταχωρητές r1 και r2 περιέχουν τις διευθύνσεις του πρώτου χαρακτήρα των source και target strings, αντίστοιχα (κάθε χαρακτήρας αποτελείται από 1 byte).

```
(1)   Repeat:   lb r3,0(r1)        ;load char (1byte)
(2)             sb r3,0(r2)        ;store char
(3)             beqz r3, Exit      ;was char NULL?
(4)             addi r1,r1,1       ;update r1 to point at next char of source string
(5)             addi r2,r2,1       ;update r2 to point at next char of target string
(6)             j Repeat
(7)   Exit:
```

Αρχικά, υποθέτουμε ότι έχουμε αρχιτεκτονική σωλήνωσης (pipelining) 5 σταδίων (IF ID EX MEM WB). Υποθέτουμε επίσης τα εξής:

- Η εγγραφή σε κάποιον καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιον καταχωρητή στο δεύτερο μισό του ίδιου κύκλου.
- Δεν υπάρχουν επιπλέον καθυστερήσεις λόγω ανάγνωσης από τη κύρια μνήμη.
- Για την εντολή διακλάδωσης χωρίς συνθήκη (6), η διεύθυνση-στόχος της διακλάδωσης ("Repeat") γίνεται άμεσα γνωστή στο στάδιο IF, ενώ ο μετρητής προγράμματος (PC) ανανεώνεται σύμφωνα με τη διεύθυνση-στόχο αυτή στο ίδιο στάδιο.
- Για την εντολή διακλάδωσης υπό συνθήκη (3), υπάρχει πρόβλεψη διακλάδωσης που είναι πάντα NOT TAKEN. Στην περίπτωση αυτή, δηλαδή, στο τέλος του σταδίου IF ο μετρητής προγράμματος θα ανανεώνεται ώστε να δείχνει στην επόμενη σειριακά εντολή.

  Ο έλεγχος για το αν η πρόβλεψη έγινε σωστά ή όχι γίνεται στο στάδιο MEM. Σε περίπτωση λάθος πρόβλεψης, στο ίδιο στάδιο υπολογίζεται και ο σωστός στόχος της διακλάδωσης, και ανανεώνεται κατάλληλα ο μετρητής προγράμματος. Σε αυτή την περίπτωση, το pipeline πρέπει να καθαριστεί ("flush") από τις εντολές που είχαν εισαχθεί στο pipeline εξαιτίας της εσφαλμένης πρόβλεψης.

α) Αρχικά, υποθέτουμε ότι η αρχιτεκτονική σωλήνωσης δε διαθέτει σχήμα προώθησης (forwarding). Για τις 2 πρώτες επαναλήψεις του παραπάνω βρόχου (και θεωρώντας ένα string με τουλάχιστον 2 μη μηδενικούς χαρακτήρες), χρησιμοποιείστε ένα διάγραμμα χρονισμού για να δείξετε τα διάφορα στάδια του pipeline από τα οποία διέρχονται οι εντολές σε αυτό το διάστημα εκτέλεσης. 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 κατά μέσο όρο για την αντιγραφή ενός χαρακτήρα; (μη λάβετε υπόψη σας τον πιθανώς διαφορετικό αριθμό κύκλων που απαιτούνται για την αντιγραφή του τελευταίου χαρακτήρα – θεωρείστε την μέση περίπτωση, δηλαδή την αντιγραφή κάποιου ενδιάμεσου χαρακτήρα).
β) Για την ίδια ακολουθία εντολών, δείξτε όπως και πριν τον χρονισμό του pipeline, θεωρώντας όμως τώρα ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης. Υποδείξτε σε κάθε περίπτωση τα σημεία όπου γίνεται προώθηση. Πόσοι κύκλοι απαιτούνται κατά μέσο όρο για την αντιγραφή ενός χαρακτήρα;
γ) Επεκτείνουμε την παραπάνω αρχιτεκτονική σωλήνωσης διπλασιάζοντας το εύρος της σωλήνωσης (εύρος 2 σωληνώσεων). Αναδιατάξτε την ακολουθία των εντολών των προηγούμενων ερωτημάτων ώστε να είναι δυνατόν να εκτελεστεί αποδοτικά σύμφωνα με τα νέα δεδομένα. Με άλλα λόγια, αναδιατάξτε τις εντολές ώστε, κατά το δυνατόν, ανά 2 να είναι ανεξάρτητες μεταξύ τους ώστε να μπορούν να εκτελεστούν παράλληλα, αλλά χωρίς ωστόσο να επηρεάζεται η σημασιολογία και η σωστή ροή εκτέλεσης του προγράμματος. Δείξτε τον χρονισμό του pipeline, θεωρώντας όλα τα δυνατά σχήματα προώθησης, και υποδείξτε τα σημεία όπου γίνεται προώθηση. Πόσοι κύκλοι απαιτούνται κατά μέσο όρο για την αντιγραφή ενός χαρακτήρα;
δ) Για να βελτιώσουμε κι άλλο την απόδοση, εφαρμόζουμε την τεχνική του ξεδιπλώματος βρόχων (loop unrolling). Αυτό επιτυγχάνεται ως εξής: πρέπει να επαναλάβουμε n φορές συνολικά τις εντολές (1) έως (3) του αρχικού κώδικα ώστε να αντιγραφούν n διαδοχικοί χαρακτήρες σε μία επανάληψη του loop, και παράλληλα, πρέπει να αυξήσουμε τους καταχωρητές r1 και r2 ώστε να δείχνουν, όχι στον επόμενο, αλλά στον χαρακτήρα που βρίσκεται n θέσεις μπροστά από τον τρέχοντα. Θεωρείστε ότι διατίθενται οι τρόποι διευθυνσιοδότησης *+k(r1)* και *-k(r1)* (k ακέραιος), ώστε να μπορούμε να αναφερθούμε στο byte που βρίσκεται k θέσεις μπροστά ή k θέσεις πίσω, αντίστοιχα, σε σχέση με το byte που είναι στη θέση μνήμης που δείχνει ο r1.
Για την ακολουθία εντολών των προηγούμενων ερωτημάτων (για την αντιγραφή των 2 πρώτων χαρακτήρων, δηλαδή), εφαρμόστε unrolling 2 φορές ακολουθώντας τις παραπάνω οδηγίες. Θεωρείστε ότι έχετε πάλι διπλή σωλήνωση όπως και πριν, οπότε φροντίστε να αναδιατάξετε όπου είναι δυνατόν την ακολουθία, επιτυγχάνοντας το μέγιστο δυνατό παραλληλισμό. Δείξτε τον χρονισμό του pipeline, θεωρώντας όλα τα δυνατά σχήματα προώθησης, και υποδείξτε τα σημεία όπου γίνεται προώθηση. Πόσοι κύκλοι απαιτούνται κατά μέσο όρο για την αντιγραφή ενός χαρακτήρα;

Θέμα 2ο (30%):

Θεωρήστε το ακόλουθο κομμάτι κώδικα:

```
int i,j;
double result, a[110][4];

for(i=0; i<4; i++)
      for(j=0; j<100; j++)
            result += a[j][i]*a[j+1][i] + 0.5;
```

Ο πίνακας a περιέχει στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 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 100 blocks δεδομένων, και έχει LRU πολιτική αντικατάστασης. Το μέγεθος του block είναι 32 bytes.
- Υποθέτουμε ότι όλες οι μεταβλητές, πλην των στοιχείων του πίνακα a, μπορούν να αποθηκευτούν σε καταχωρητές του επεξεργαστή, οπότε οποιαδήποτε αναφορά σε αυτές δεν συνεπάγεται προσπέλαση στην κρυφή μνήμη. Επίσης, 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 a[j][i] και μετά για το a[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 a 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) Το σύνολο εντολών της αρχιτεκτονικής του επεξεργαστή διαθέτει μία ειδική εντολή prf(*addr). Η εντολή αυτή προ-φορτώνει στην κρυφή μνήμη ολόκληρο το μπλοκ που περιέχει τη λέξη που βρίσκεται στη διεύθυνση μνήμης addr. Χωρίς να αλλάξετε τη σειρά των loads για τις αναφορές στα στοιχεία του πίνακα, εισάγετε κλήσεις στην prf (1 ή περισσότερες) στον παραπάνω κώδικα ώστε να μειωθούν τα misses. Δώστε τον συνολικό αριθμό των misses.
Υποθέστε ότι 7 επαναλήψεις του εσωτερικού loop (συμπεριλαμβανομένων των όποιων κλήσεων στην prf) είναι αρκετές ώστε να "καλυφθεί" ο χρόνος που απαιτείται για να έρθουν τα δεδομένα που ζητά η prf στην cache. Επιπλέον, μη λάβετε ειδική μέριμνα για την προφόρτωση δεδομένων στις αρχικές επαναλήψεις του loop, ούτε για τις έξτρα προφορτώσεις στις τελευταίες επαναλήψεις. Φροντίστε μόνο να εισάγετε τον ελάχιστο αριθμό εντολών prf, αποφεύγοντας τις όποιες περιττές κλήσεις στην εντολή.
γ) Υποθέστε τώρα ότι έχετε μια cache ίδιας οργάνωσης, αλλά "απείρου" μεγέθους (δηλαδή, δεν υπάρχουν capacity misses). Πώς θα ξαναγράφατε τον κώδικα του ερωτήματος 2, μειώνοντας περαιτέρω τον αριθμό των κλήσεων στην prf;
δ) Θεωρείστε πάλι τον αρχικό κώδικα (χωρίς τις προφορτώσεις, δηλαδή) και την αρχική cache της άσκησης. Εκτός από την προφόρτωση δεδομένων, ποια άλλη γνωστή τεχνική βελτιστοποίησης θα εφαρμόζατε στον κώδικα ώστε να μειωθούν τα misses; Ξαναγράψτε τον κώδικα εξηγώντας πώς επιτυγχάνεται η μείωση αυτή.

Θέμα 3ο (35%):

Θεωρήστε ένα σύστημα μνήμης με μία cache χωρητικότητας 1 MB δεδομένων, με cache line 4 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 32 bit. Για κάθε μία από τις ακόλουθες περιπτώσεις οργάνωσης της cache:

(i) ευθείας αντιστοίχισης (direct mapped)
(ii) συσχέτισης 2 δρόμων (2-way set associative)
(iii) πλήρως συσχετιστική (fully associative)

α) Υπολογίστε τον αριθμό των bits καθενός από τα επιμέρους πεδία στα οποία χωρίζεται μία διεύθυνση μνήμης σε μία τέτοια οργάνωση cache. Παρουσιάστε ένα διάγραμμα που να δείχνει πώς διαχωρίζεται η διεύθυνση στα πεδία αυτά, και εξηγείστε τη σημασία του καθενός.
β) Παρουσιάστε σε block διάγραμμα την cache, τα επιμέρους πεδία μιας διεύθυνσης, και τον τρόπο που διασυνδέονται μεταξύ τους ώστε να οδηγήσουν στην προσπέλαση (εγγραφή/ανάγνωση) δεδομένων της cache. Εξηγείστε εν συντομία τη σημασία της κάθε διασύνδεσης στη διαδικασία εύρεσης δεδομένων στην cache. Λάβετε υπόψη ότι κάθε cache line έχει ένα επιπλέον bit valid/invalid. Τι ποσοστό του συνολικού μεγέθους της cache αφιερώνεται για τα bits του tag σε κάθε μία από τις περιπτώσεις;
γ) Σε ποιες θέσεις της cache απεικονίζονται οι ακόλουθες διευθύνσεις bytes στη μνήμη (δίνονται σε 16-δική μορφή): 300A21, 20FC03, 1B30, 1B32, 51028, A2E, 20FC05, 29E000, 200A22, 151023, 1B33, 300A21, 251027, 1B33, 300A2D, 51025. Η αναζήτησή τους στη μνήμη καταλήγει σε miss ή hit;