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

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2005-2006, 8ο Εξάμηνο ΗΜ&ΜΥ

2η Σειρά Ασκήσεων

(Παράδοση στο e-mail: advanced-ca-2006@cslab.ece.ntua.gr μέχρι 21/9/06)

1. Θεωρήστε ένα σύστημα μνήμης με μία 4-way set associative cache μεγέθους 256 KB, 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.

1) 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.
2) Σε ποιες θέσεις της cache μπορεί να απεικονιστεί το byte στη διεύθυνση μνήμης 5000_{10} ;
3) Ποιες θέσεις μνήμης μπορούν να απεικονιστούν στο σύνολο (set) 244 της cache;
4) Τι ποσοστό του συνολικού μεγέθους της cache αφιερώνεται για τα bits του tag;

2. Έχουμε ένα σύστημα που αποτελείται από έναν επεξεργαστή με in-order εκτέλεση εντολών, συχνότητα λειτουργίας 1.5 GHz και CPI ίσο με 1 (στην περίπτωση όπου δεν υπάρχουν προσβάσεις στη μνήμη). Οι μόνες εντολές που διαβάζουν/γράφουν δεδομένα από/προς τη μνήμη, είναι loads (40% κατά μέσο όρο επί του συνόλου των εντολών) και stores (10% επί του συνόλου των εντολών).
Η ιεραρχία μνήμης αποτελείται από τα εξής στοιχεία:
- Κρυφή μνήμη εντολών επιπέδου 1 (L1-I): direct mapped, μεγέθους 32 KB, ποσοστό αστοχίας 5%, με blocks των 64 bytes. Ο χρόνος που η L1I ικανοποιεί ένα hit ισούται με 2 nsec.
- Κρυφή μνήμη δεδομένων επιπέδου 1 (L1-D): direct mapped, write-through, write-no-allocate, μεγέθους 32 KB, ποσοστό αστοχίας 8%, με blocks των 64 bytes. Ο χρόνος που η L1-D ικανοποιεί ένα hit ισούται με 2 nsec.
- Για να μειωθούν τα "write-stalls", δηλαδή οι καθυστερήσεις που οφείλονται στο ότι ο επεξεργαστής πρέπει να περιμένει να ολοκληρωθούν οι "write-through" εγγραφές της L1-D, το σύστημα μνήμης διαθέτει έναν ειδικό buffer εγγραφής, που επιτρέπει στον επεξεργαστή να συνεχίσει μόλις τα δεδομένα εγγραφούν σε αυτόν. Με αυτό τον τρόπο γίνεται επικάλυψη της εκτέλεσης και της εγγραφής στη μνήμη. Το 96% των εγγραφών της L1-D, βρίσκουν ελεύθερη θέση στον buffer άμεσα. Το υπόλοιπο 4%, πρέπει να περιμένουν μέχρι να ελευθερωθεί κάποια θέση στον buffer. Σε αυτή την περίπτωση, ο buffer κάνει στην ουσία μία αίτηση στην L2 για να στείλει εκεί τα δεδομένα κάποιας θέσης του, και η θέση αυτή αποδεσμεύεται μόλις η L2 είναι έτοιμη να δεχτεί τα δεδομένα αυτά. Αν δεν υπάρχουν ελεύθερες θέσεις στον buffer, ο επεξεργαστής θα πρέπει να περιμένει όποτε κάνει κάποια εγγραφή.
- Ενοποιημένη κρυφή μνήμη εντολών και δεδομένων επιπέδου 2 (L2): write-back, write-allocate, μεγέθους 1024 KB, με blocks των 64 bytes. Ο χρόνος που η L2 ικανοποιεί ένα hit είναι 22 nsec. Αυτός είναι και ο χρόνος εγγραφής μιας λέξης στην L2. Το ποσοστό επιτυχίας της είναι 84%. Επιπλέον, το 50% όλων των blocks της που αντικαθίστανται είναι "dirty".

Οι διευθύνσεις στο σύστημά μας έχουν εύρος 64 bits. Ο χρόνος πρόσβασης σε μία διεύθυνση της

κύριας μνήμης (είτε λόγω ανάγνωσης, είτε λόγω "write-back") είναι 50 nsec. Ο δίαυλος δεδομένων μεταξύ μνήμης-επεξεργαστή έχει εύρος 64 bits, λειτουργεί στα 100 MHz, και μπορεί να μεταφέρει 64 bits σε κάθε κύκλο διαύλου.

Σημείωση: στα ερωτήματα που ακολουθούν, όπου ζητούνται ποσοστά αστοχίας για κάποια κρυφή μνήμη, δώστε τα "τοπικά" ποσοστά (δηλαδή, από την οπτική γωνία της μνήμης αυτής).

Α) Υπολογισμός μέσου χρόνου πρόσβασης στη κύρια μνήμη για ανάγνωση εντολών:

i) Για την ανάγνωση εντολών, δώστε τις τιμές για τα ακόλουθα μεγέθη: χρόνος ικανοποίησης hit από την L1, ποσοστό αστοχίας στην L1, χρόνος ικανοποίησης hit από την L2, ποσοστό αστοχίας στην L2.
ii) Δώστε έναν τύπο που να υπολογίζει την ποινή αστοχίας στην L2, και υπολογίστε την τιμή της για την περίπτωση που εξετάζουμε. Ως ποινή αστοχίας στην L2, εννοούμε τον χρόνο από τη στιγμή που η L2 ζητήσει τα δεδομένα που δεν βρέθηκαν σε αυτήν, μέχρι τα δεδομένα να έρθουν σε αυτήν από την κύρια μνήμη. Για το υποερώτημα αυτό, λάβετε υπόψη σας το γεγονός ότι το 50% όλων των blocks της που αντικαθίστανται είναι "dirty".
iii) Χρησιμοποιώντας τα μεγέθη που ζητούνται στα i και ii, δώστε έναν τύπο που να υπολογίζει τον μέσο χρόνο πρόσβασης στη μνήμη. Υπολογίστε την τιμή του για το σύστημα μνήμης που εξετάζουμε.

Β) Υπολογισμός μέσου χρόνου πρόσβασης στη κύρια μνήμη για ανάγνωση δεδομένων:

i) Για την ανάγνωση δεδομένων, δώστε την τιμή του ποσοστού αστοχίας στην L1.
ii) Δώστε έναν τύπο που να υπολογίζει τον μέσο χρόνο πρόσβασης στην μνήμη. Υπολογίστε την τιμή του για το σύστημα μνήμης που εξετάζουμε.

Γ) Υπολογισμός μέσου χρόνου πρόσβασης στη κύρια μνήμη για εγγραφή δεδομένων:

i) Για την εγγραφή δεδομένων, υπολογίστε την τιμή της ποινής αστοχίας στην L2.
ii) Υπολογίστε τον μέσο χρόνο εγγραφής των δεδομένων μιας θέσης του buffer εγγραφής στην L2.
iii) Χρησιμοποιώντας τα προηγούμενα μεγέθη, δώστε έναν τύπο και υπολογίστε τον μέσο χρόνο πρόσβασης στη μνήμη για εγγραφές δεδομένων. Σημείωση: λάβετε υπόψη σας τις 2 περιπτώσεις για την κατάσταση του buffer εγγραφής (γεμάτος ή όχι).

3. Θεωρείστε το ακόλουθο κομμάτι κώδικα:

```
 1:    int x,y;
 2:    float tmp1, tmp2, arr1[64][64], arr2[64][64];
 3:    for ( x = 0; x < 64; x++ )
 4:    {
 5:          for ( y = 0; y < 64; y++ )
 6:                tmp1 -= arr1[x][y];
 7:
 8:          for ( y = 0; y < 32; y++ )
 9:                tmp2 -= arr2[x][2*y];
10:
11:    }
```

Κάνουμε τις ακόλουθες υποθέσεις:
- Οι τύποι δεδομένων int και float είναι μεγέθους 4 bytes.
- Όλες οι μεταβλητές, πλην των στοιχείων των 2 πινάκων, μπορούν να αποθηκευτούν σε καταχωρητές του επεξεργαστή, οπότε οποιαδήποτε αναφορά σε αυτές δεν συνεπάγεται προσπέλαση στην κρυφή μνήμη. Επιπλέον, δε λαμβάνονται υπόψη οι προσπελάσεις των εντολών στη μνήμη (υποθέτουμε ότι έχουμε τέλεια κρυφή μνήμη εντολών).
- Η κρυφή μνήμη δεδομένων είναι πλήρως συσχετιστική, με LRU πολιτική αντικατάστασης,

αποτελούμενη από 32 γραμμές, με κάθε γραμμή να αποτελείται από 16 bytes. Αρχικά, είναι εντελώς άδεια.
- Οι πίνακες είναι αποθηκευμένοι στην κύρια μνήμη κατά γραμμές ("row-major layout"). Επιπλέον, είναι "ευθυγραμμισμένοι" ώστε το πρώτο στοιχείο του καθενός να απεικονίζεται στην αρχή μιας γραμμής της κρυφής μνήμης.
- Η εκτέλεση των εντολών γίνεται σειριακά. Ο χρόνος που απαιτείται για να εκτελεστεί κάθε φορά κάποια από τις γραμμές 3,5 και 8 ισούται με 5 κύκλους. Οι γραμμές 6 και 9 χρειάζονται 12 κύκλους η κάθε μία, και 50 κύκλους επιπλέον αν σημειωθεί κάποια αστοχία στην κρυφή μνήμη δεδομένων.
- 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. Η εντολή εκτελείται σε 1 κύκλο από τον επεξεργαστή, ενώ από το σημείο αυτό και έπειτα, η προφόρτωση των δεδομένων γίνεται παράλληλα και ανεξάρτητα από την υπόλοιπη εκτέλεση του προγράμματος, χωρίς να προκαλείται κάποιο stall στο pipeline του επεξεργαστή. Με άλλα λόγια, ο επεξεργαστής μπορεί να συνεχίσει με την εκτέλεση των εντολών που ακολουθούν. Αν τα δεδομένα που ζητάει η εντολή prf δεν είναι στην κρυφή μνήμη, τότε θεωρούμε ότι απαιτούνται 50 κύκλοι μέχρι να μεταφερθούν σε αυτήν.

α. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του παραπάνω τμήματος κώδικα αν δεν χρησιμοποιήσουμε προφόρτωση δεδομένων;

β. Θεωρείστε την περίπτωση εισαγωγής εντολών prf στα δύο εσωτερικότερα loops του κώδικα. Εξηγείστε γιατί είναι απαραίτητο σε αυτή την περίπτωση να εφαρμόσουμε στα loops αυτά την τεχνική του ξεδιπλώματος βρόχων ("loop unrolling"). Ποιος είναι ο ελάχιστος αριθμός φορών που χρειάζεται να "ξεδιπλώσουμε" κάθε ένα από τα 2 loops για το σκοπό αυτό;

γ. Εφαρμόστε unrolling στα εσωτερικά loops, για τον αριθμό φορών που απαντήσατε στο προηγούμενο ερώτημα. Εισάγετε τον μικρότερο δυνατό αριθμό εντολών prf στα κατάλληλα σημεία του "ξεδιπλωμένου" κώδικα, ώστε να ελαχιστοποιηθεί ο χρόνος εκτέλεσης. (Σημείωση: μην λάβετε ειδική μέριμνα για τις αρχικές επαναλήψεις των loops)

δ. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα που προέκυψε από το προηγούμενο ερώτημα; Υπολογίστε το ποσοστό % της βελτίωσης του χρόνου εκτέλεσης ("speedup") σε σχέση με τον αρχικό, μη βελτιστοποιημένο κώδικα.

ε. Υπάρχει άλλος τρόπος, εκτός του loop unrolling, να γραφτεί το αρχικό πρόγραμμα, ώστε να επιτυγχάνεται μεν ο σκοπός που εξυπηρετεί το unrolling, αλλά χρησιμοποιώντας αυτή τη φορά λιγότερες εντολές σε σχέση με τον κώδικα του ερωτήματος "γ";