ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Κανονική Εξεταστική Απριλίου 2014
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.

ΤΜΗΜΑ 1ο (Α – Κ)

ΟΝΟΜΑ : _____ Α.Μ:_____ ΕΞΑΜΗΝΟ : ____

Θέμα 1ο (15%)

Α) Ποια η διαφορά μεταξύ μιας εξάρτησης και ενός κινδύνου; Δώστε τους ορισμούς των WAW, WAR και RAW εξαρτήσεων. Εξηγήστε ποιες από τις παραπάνω εξαρτήσεις οδηγούν σε κινδύνους και ποιες όχι στην κλασσική αρχιτεκτονική σωλήνωσης 5 σταδίων του MIPS.

Β) Απαντήστε αν οι παρακάτω προτάσεις είναι ψευδείς ή αληθείς. Δικαιολογήστε τις απαντήσεις σας.

i) Για το ίδιο datapath, το CPI ενός pipelined επεξεργαστή είναι μικρότερο από το CPI ενός single-cycle επεξεργαστή.
ii) Κατά την εκτέλεση ενός κώδικα σε ένα pipelined επεξεργαστή, η χρήση καλύτερου compiler δεν επηρεάζει το CPI.
iii) Η εκτέλεση ενός κώδικα που δεν επαναχρησιμοποιεί δεδομένα (π.χ. streaming εφαρμογές) δεν επηρεάζεται αν μειωθεί στο μισό το associativity της data cache (διατηρώντας τα υπόλοιπα χαρακτηριστικά της σταθερά).
iv) Ο υποδιπλασιασμός του συνολικού μεγέθους της cache, διατηρώντας σταθερό το associativity καθώς και τον αριθμό των sets, έχει σαν αποτέλεσμα την αύξηση των compulsory misses.
v) Η χρήση victim cache βοηθάει στη μείωση των compulsory misses.

Γ) Δίνεται ο παρακάτω κώδικας assembly MIPS, ο οποίος προσπαθεί να φορτώσει την σταθερά 0x0ABCFF0A στον καταχωρητή $t0.

```
lui  $t0, 0xABC
addi $t0, $t0, 0xFF0A
```

Εξηγήστε ποιο είναι το λάθος στον παραπάνω κώδικα. Δώστε μια αλληλουχία εντολών που να φορτώνει σωστά τη σταθερά στον καταχωρητή $t0.

Θέμα 2ο (35%)

Υποθέστε την κλασσική αρχιτεκτονική σωλήνωσης του MIPS στην οποία το στάδιο MEM έχει χωριστεί σε 2 στάδια (M1 και M2) σε μια προσπάθεια να μειωθεί η διάρκεια του κύκλου. Κατά τον εντοπισμό μιας εντολής άλματος υπό συνθήκη, ο επεξεργαστής κάνει stall τη σωλήνωση μέχρι την επίλυση η

οποία πραγματοποιείται στο στάδιο M2. Υποθέστε επίσης, ότι η εγγραφή σε ένα καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου. Δίνεται το ακόλουθο κομμάτι κώδικα :

```
1.    LOOP: LW   $t1, 100($t2)
2.          LW   $s1, 0($t1)
3.          ADDI $t3, $t3, -4
4.          ADD  $s2, $s1, $s1
5.          LW   $s3, 100($t4)
6.          ADDI $t2, $t2, 4
7.          ADD  $s3, $s3, $s2
8.          SW   $s3, 100($t4)
9.          ADDI $t4, $t4, 4
10.         BNEZ $t3, LOOP
```

Δίνεται ότι η αρχική τιμή του καταχωρητή $t3 (ακριβώς πριν το loop) είναι ίση με 100.

A) Υποθέστε ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης. Δώστε το διάγραμμα χρονισμού για την 1η επανάληψη του βρόχου, υποδεικνύοντας τις προωθήσεις που γίνονται. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του κώδικα;

B) Σας δίνονται οι εξής 2 επιλογές:

i) Η χρήση ενός compiler ο οποίος αναδιατάσσει τον κώδικα (με τις απαραίτητες βέβαια μετατροπές για να μην αλλάξει την σημασιολογία του προγράμματος) προκειμένου να επιτύχει βέλτιστη επίδοση.

ii) Η ενοποίηση των σταδίων M1 και M2 σε ένα στάδιο MEM.

Ποια επιλογή θα ακολουθούσατε προκειμένου να επιτύχετε την ταχύτερη εκτέλεση του κώδικα; Δικαιολογήστε την απάντηση σας δίνοντας τα κατάλληλα διαγράμματα χρονισμού υποδεικνύοντας πάντα τις προωθήσεις που γίνονται.

Θέμα 3ο (25%)

A) Δίνεται το παρακάτω πρόγραμμα σε C καθώς και η μετάφρασή του σε assembly MIPS. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής $0 (ή $zero) είναι πάντα μηδέν. Δίνεται επίσης ότι η διεύθυνση του πρώτου στοιχείου του πίνακα a βρίσκεται αποθηκευμένη στον καταχωρητή $t4.

```
int a[12], i;
int mask = 0xFFFAB;

for (i=0; i < 11; i++){
   a[i] = a[i+1] & mask;
}
```

```
                add  $t1, $zero, ____
                lui  $t2, ____
                ____ $t2, $t2, ____
loop:           lw   ___, ____($t4)
                and  ___, $t3, $t2
                sw   $t3, ____(____)
                addi $t4, $t4, ____
                addi $t1, $t1, ____
                addi ___, $t1, ____
                bne  $t8, ___, loop
```

B) Δίνεται το παρακάτω πρόγραμμα σε C καθώς και μια μετάφρασή του σε assembly MIPS, όπου οι μεταβλητές s και p είναι αποθηκευμένες στους καταχωρητές $s6 και $s9 αντίστοιχα, ενώ ο καταχωρητής $s0 περιέχει τη διεύθυνση του πρώτου στοιχείου του πίνακα v. Η μετάφραση αυτή περιέχει λάθη (ένα εκ των οποίων είναι μια εντολή που λείπει). Βρείτε τα λάθη και διορθώστε τα.

```
int v[12], s;
int *p;
s = 16;
for(p = &v[2]; *p!=0; p+=2)
   s = s + *p;
```

```
                or   $s6, $zero, $zero
                lw   $s9, 8($s0)
loop:           bne  $t8, $zero, finish
                add  $s6, $s9, $s6
                addi $s9, $s9 , 2
                jmp loop
finish:
```

Γ) Υποθέστε μια 2-way set associative cache με συνολικό μέγεθος tag array 48 bits και με LRU πολιτική αντικατάστασης. Η ελάχιστη μονάδα διευθυνσιοδότησης είναι το 1 byte ενώ το σύστημα χρησιμοποιεί διευθύνσεις μήκους 9 bits. Υποθέτοντας μια αρχικά άδεια cache, παρατηρούμε την εξής συμπεριφορά για τις ακόλουθες προσπελάσεις σε διευθύνσεις μνήμης:

011011000	m
011001000	m
011011100	h

i) Ποιο είναι το συνολικό μέγεθος της cache;
ii) Έστω ότι πραγματοποιούνται οι παρακάτω προσβάσεις (με τη σειρά που δίνονται). Αν η cache είναι αρχικά άδεια, υπολογίστε το αποτέλεσμα της κάθε πρόσβασης (Hit ή Miss). Οι διευθύνσεις δίνονται σε δεκαεξαδική μορφή, αλλά παραμένουν μήκους 9 bits (αγνοείστε δηλαδή τα 3 MSB).
• 0x151, 0x155, 0x020, 0x191, 0x1d4, 0x153, 0x123, 0x021

Θέμα 4ο (25%)

Εξετάζουμε την εκτέλεση του ακόλουθου κώδικα C.

```
double a[16][16], b[256];
k=0;
for(i=0; i<8; i++)
  for(j=0; j<16; j++) {
    a[i+4][j] = a[i][j] + a[i+1][j] + b[k+2];
    k++;
}
```

- Οι πίνακες περιέχουν στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 bytes το καθένα.
- Το πρόγραμμα εκτελείται σε επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι συσχέτισης δύο δρόμων, write-allocate, αποτελείται από 32 blocks δεδομένων και χρησιμοποιεί LRU πολιτική αντικατάστασης. Το μέγεθος του block είναι 32 bytes, ενώ η μικρότερη μονάδα δεδομένων που μπορεί να διευθυνσιοδοτηθεί είναι το 1 byte.
- Όλες οι μεταβλητές, πλην των στοιχείων των πινάκων, αποθηκεύονται 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.
- Οι πίνακες είναι αποθηκευμένοι στην κύρια μνήμη κατά γραμμές, και διαδοχικά, δηλαδή ο ένας αμέσως μετά τον άλλον. Το πρώτο στοιχείο του πίνακα a βρίσκεται στη διεύθυνση **0x00080000**.

Α) Βρείτε το συνολικό αριθμό hits και misses για όλη την εκτέλεση του παραπάνω κώδικα. Υποδείξτε ποια misses είναι compulsory, ποια είναι capacity και ποια conflict.

Β) Αν η κρυφή μνήμη αντικατασταθεί με μια ίδια κρυφή μνήμη αλλά write-no-allocate, πώς θα επηρεαστούν τα hits και misses; Δικαιολογήστε την απάντησή σας δίνοντας όπως και πριν το συνολικό τους αριθμό.