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

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ (5ο εξάμηνο)

ΕΠΑΝΑΛΗΠΤΙΚΗ ΕΞΕΤΑΣΗ (ΟΚΤΩΒΡΙΟΣ 2008)

ΔΙΑΡΚΕΙΑ ΕΞΕΤΑΣΗΣ: 2 ΩΡΕΣ 30 ΛΕΠΤΑ

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.

Καλή Επιτυχία!

Θέμα 1 (25%)

Α) Για κάθε μία από τις ακόλουθες προτάσεις απαντήστε αν είναι σωστή ή λάθος, δικαιολογώντας την απάντησή σας:

i) *Έστω μια δίοδος δεδομένων (datapath) που αποτελείται από διαδοχικά στάδια διαφορετικής διάρκειας το καθένα και Χ το στάδιο με τη μικρότερη διάρκεια. Σε μια pipelined υλοποίηση της διόδου, αν διαιρέσω περαιτέρω το Χ σε επιμέρους στάδια, θα προκύψει μεγαλύτερη συχνότητα ρολογιού.*

ii) *Στην κλασική αρχιτεκτονική αγωγού 5 σταδίων του MIPS, οι WAW κίνδυνοι ποτέ δεν προκαλούν stalls στο pipeline.*

iii) *Στην κλασική αρχιτεκτονική αγωγού 5 σταδίων του MIPS, οι RAW κίνδυνοι ποτέ δεν προκαλούν stalls στο pipeline.*

Β) Έστω ότι σε ένα πρόγραμμα εμφανίζονται οι κατηγορίες εντολών που φαίνονται στον ακόλουθο πίνακα. Στον ίδιο πίνακα παρουσιάζεται η κατανομή των εντολών, καθώς και ο χρόνος που κάθε τύπος εντολής απαιτεί σε κάθε στάδιο της διόδου δεδομένων (datapath). Όταν μια εντολή δε χρησιμοποιεί κάποιο στάδιο, αυτό υποδηλώνεται με "-".

Εντολή	Κατανομή	IF	ID	EX	MEM	WB
Load	25%	2ns	1ns	2ns	2ns	1ns
Store	10%			2ns	1ns	-
Arithmetic	45%			1ns	-	1ns
Branch	20%			2ns	-	-

i) Με βάση τους χρόνους σε nanoseconds που δίνονται στον πίνακα, αν επιλέγαμε να υλοποιήσουμε το datapath σαν *single-cycle datapath*, ποιο θα ήταν το throughput (σε εντολές ανά second) για το συγκεκριμένο πρόγραμμα;

ii) Αν επιλέγαμε να υλοποιήσουμε το datapath σαν *multi-cycle datapath*, ποιο θα ήταν το throughput για το πρόγραμμά μας; Είναι η multi-cycle υλοποίηση καλύτερη από την single-cycle για την περίπτωση που εξετάζουμε; Εξηγείστε πού οφείλεται η όποια παρατηρούμενη διαφορά στην απόδοση.
iii) Αν επιλέγαμε να υλοποιήσουμε το datapath σαν *pipelined datapath 5 σταδίων*, ποιο θα ήταν το throughput για το πρόγραμμά μας;

Θέμα 2 (30%)

Εξετάζουμε την εκτέλεση του εξής κώδικα:

```
Rep:   mul $1, $2, 20($6)
       or $4, $5, 100($1)
       sw $4, 20($6)
       and $2, $2, #10
       sub $6, $6, #8
       bnez $6, Rep
Exit:
```

Υποθέτουμε ότι έχουμε αρχιτεκτονική σωλήνωσης (pipelining) με τα εξής στάδια: IF ID ALU1 MEM ALU2 WB. Το στάδιο ALU1 χρησιμοποιείται για τον υπολογισμό των τελικών διευθύνσεων μνήμης (effective address calculation), όπως επίσης και για τον υπολογισμό των διευθύνσεων-στόχων (targets) των εντολών διακλάδωσης. Το στάδιο ALU2 χρησιμοποιείται για όλους τους υπόλοιπους υπολογισμούς, καθώς και για την επίλυση των διακλαδώσεων (εάν είναι υπό συνθήκη). Έστω ότι η αρχική τιμή του $6 είναι 400.

Α) Αρχικά, υποθέτουμε ότι η αρχιτεκτονική σωλήνωσης δε διαθέτει σχήμα προώθησης (forwarding). Η εγγραφή σε κάποιον καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιον καταχωρητή στο δεύτερο μισό του ίδιου κύκλου. Για την 1η επανάληψη του παραπάνω βρόχου, μέχρι και τη 1η εντολή της 2ης επανάληψης, 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 εν συντομία τους πιθανούς κινδύνους (hazards) που μπορούν να προκύψουν κατά την εκτέλεση, καθώς και τον τρόπο με τον οποίον αυτοί αντιμετωπίζονται. Πόσοι κύκλοι απαιτούνται συνολικά για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η);

Β) Για την ίδια ακολουθία εντολών, δείξτε και εξηγείστε όπως και πριν, τον χρονισμό του pipeline, θεωρώντας όμως τώρα ότι υπάρχει σχήμα προώθησης. Πόσοι κύκλοι απαιτούνται συνολικά για να ολοκληρωθεί ο βρόχος;

Θέμα 3 (20%)

Α) Τι είναι η χωρική τοπικότητα των αναφορών και πώς την εκμεταλλεύεται μια cache;
Β) Ποια είναι η διαφορά ανάμεσα σε μια write-back και σε μια write-through cache; Ποιο είναι το πλεονέκτημα της χρήσης της πρώτης σε σχέση με τη δεύτερη;
Γ) Έστω μια cache αποτελούμενη από 4 γραμμές, με κάθε γραμμή να χωρά μία λέξη. Υπάρχει ακολουθία αναφορών στη μνήμη για την οποία μια direct mapped οργάνωση της cache θα έδινε λιγότερα misses σε σχέση με μια fully associative οργάνωση με LRU πολιτική αντικατάστασης; Αν ναι, δώστε ένα παράδειγμα. Αν όχι, εξηγείστε γιατί.

Δ) Δίνεται μια 4-way 32KB cache με 128 sets. Η μικρότερη μονάδα δεδομένων που μπορεί να διευθυνσιοδοτηθεί είναι το 1 byte. Για μια αρχιτεκτονική των 48 bit, ποιο είναι το συνολικό μέγεθος (σε bits ή bytes) του tag array; Για την ίδια αρχιτεκτονική, αν αλλάξουμε την οργάνωση της cache σε fully-associative (διατηρώντας τη χωρητικότητα και το μέγεθος της cache line), ποιο θα είναι το νέο μέγεθος του tag array;

Θέμα 4 (25%)

Θεωρήστε την εκτέλεση του ακόλουθου βρόχου:

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

Κάθε στοιχείο των πινάκων A και B είναι 8 bytes. Έστω ότι για τα στοιχεία A[0] και B[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 A[i] και έπειτα του B[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.

Α) Ευθείας αντιστοίχισης (direct mapped), write through, no write allocate, με μέγεθος block ίσο με 32 bytes.

Β) Ευθείας αντιστοίχισης, write back, write allocate, με μέγεθος block ίσο με 32 bytes.

Γ) Συσχέτισης 2 δρόμων (2-way set associative), write through, no write allocate, με μέγεθος block ίσο με 32 bytes και πολιτική αντικατάστασης FIFO.