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

Εξετάσεις Σεπτεμβρίου 2008
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.

## Θέμα 1ο (2 μονάδες)

**Α.** Δίνεται επεξεργαστής με σωλήνωση Ν σταδίων ο οποίος χρησιμοποιεί Branch-Target Buffer (BTB) για την πρόβλεψη διακλαδώσεων. Ισχύουν τα εξής :

- 20% των εντολών είναι εντολές διακλάδωσης.
- Ο BTB έχει hit rate 90% ενώ η ακρίβεια πρόβλεψης του είναι 90%.
- Σε περίπτωση miss του BTB προβλέπεται ότι η εντολή διακλάδωσης θα είναι Not Taken. Η ακρίβεια πρόβλεψης αυτού του static predictor είναι 45%.
- Για κάθε εντολή διακλάδωσης η αποκωδικοποίηση και η πρόβλεψη γίνονται στο πρώτο στάδιο της σωλήνωσης. Ο έλεγχος ορθότητας της πρόβλεψης γίνεται στο τελευταίο στάδιο και σε περίπτωση λάθους η εκτέλεση της σωστής εντολής ξεκινά από τον επόμενο κύκλο.
- Οι εντολές που δεν προκαλούν stalls απαιτούν 1 κύκλο για να εκτελεστούν.

Υπολογίστε το μέγιστο βάθος της σωλήνωσης, ώστε ο επεξεργαστής αυτός να είναι πιο γρήγορος από έναν άλλο επεξεργαστή όπου δεν υπάρχει πρόβλεψη διακλάδωσης και κάθε εντολή διακλάδωσης κάνει stall την σωλήνωση για 2 κύκλους.

**Β**. Στις αρχιτεκτονικές σωλήνωσης υπάρχουν RAW, WAR και WAW κίνδυνοι (hazards).
**(i)** Για κάθε έναν από από τους παραπάνω hazards δώστε τον ορισμό του καθώς και ένα παράδειγμα κώδικα όπου αυτός δημιουργείται.
**(ii)** Με ποια τεχνική καταφέρνει ο Tomasulo να επιλύσει τους WAR και WAW hazards; Εξηγήστε γιατί η τεχνική αυτή δεν μπορεί να επιλύσει και τους RAW hazards. Με ποιο τρόπο επιλύει τους RAW hazards ο Tomasulo;
**(iii)** Ο Tomasulo βελτιώνει την επίδοση του συστήματος επιτρέποντας την εκτέλεση εντολών (στάδιο EX) σε διαφορετική σειρά από αυτή που έχει ορίσει ο προγραμματιστής. Αν μάλιστα δεν χρησιμοποιείται ο Reorder Buffer, τότε επιτρέπεται και η εκτός σειράς ολοκλήρωση των εντολών (στάδιο WB). Σε κάθε περίπτωση όμως, η ανάκληση και δρομολόγηση των εντολών (στάδια Fetch και ISSUE) γίνονται με τη σειρά του προγράμματος (in-order). Εξηγήστε γιατί.

## Θέμα 2ο (2 μονάδες)

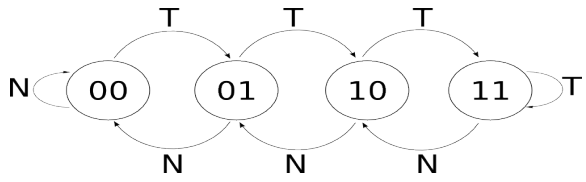
**Α.** Υποθέστε εντολές διακλάδωσης με τις παρακάτω συμπεριφορές :

**(i)** T, T, T, T, T, T, *N*, T, T, T, *N*, T, T, T, T, T, T, T, *N*, *N*, *N*, *N*, *N*, *N*, *N*, T, T, T, T, T, T, T, T, T, T, *N*
**(ii)** T, T, T, T, T, T, T, T, T, T, T, *N*, *N*, *N*, T, T, T, T, *N*, T, T, T, *N*, T, T, T, T, *N*, *N*, *N*, T, *N*

**(iii)** T, T, T, T, *N*, T, T, T, T, T, T, *N*, T, T, T, T, T, T, T, *N*, *N*, *N*, *N*, *N*, T, *N*, *N*, *N*, *N*, *N*, *N*, T, *N*, *N*, *N*

όπου T taken και N not-taken. Σας δίνονται επίσης οι επόμενοι μηχανισμοί πρόβλεψης :
1. static Taken
2. 1-bit, με αρχική τιμή 1 (κατάσταση T)
3. 2-bit, με αρχική τιμή 3 (κατάσταση T)

Ποιο μηχανισμό θα προτείνατε για κάθε μια από τις παραπάνω εντολές διακλάδωσης και γιατί; Ο 2-bit predictor χρησιμοποιεί το παρακάτω FSM (finite state machine) :




**B.** Δίνεται ο παρακάτω κώδικας σε C, ένας (2,1) global history predictor και ένας 2-bit predictor.

```
for (i=100; i>0; i--) {        // Branch1
     if (i mod 2 == 0) {       // Branch2
          .....
     }
     else {
          .....
     }
}
```

**(i)** Ποιος predictor είναι ο καταλληλότερος για το Branch1 και γιατί;
**(ii)** Ποιος predictor είναι ο καταλληλότερος για το Branch2 και γιατί;

## Θέμα 3ο (3 μονάδες)

**A.** Δίνεται αρχιτεκτονική η οποία υλοποιεί τον αλγόριθμο Tomasulo χρησιμοποιώντας ROB για in-order commit εντολών. Το pipeline του επεξεργαστή περιέχει τα στάδια Issue (IS), Execute (EX), Write Result (WR) και Commit (CMT), αγνοούμε δηλαδή το IF. Ισχύουν επίσης τα ακόλουθα :

1. Τα IS, WR, CMT απαιτούν 1 κύκλο.
2. Στο στάδιο EX εκτελείται και ο υπολογισμός της διεύθυνσης μιας προσπέλασης στη μνήμη καθώς και η ίδια η προσπέλαση.
3. Το σύστημα περιέχει άπειρα reservation stations καθώς και άπειρα execution units.
4. Οι εντολές αναφοράς στη μνήμη απαιτούν 4 κύκλους εκτέλεσης, οι SUB και ADD 2 κύκλους, η MULT 6 κύκλους και οι εντολές διακλάδωσης 2 κύκλους.
5. Ο ROB έχει 6 θέσεις.
6. Για τις εντολές διακλάδωσης χρησιμοποιείται ένας static Taken predictor.
7. Η πρόβλεψη για μια εντολή διακλάδωση γίνεται ταυτόχρονα με τη δρομολόγηση της εντολής.
8. Ο έλεγχος της πρόβλεψης για μια εντολή διακλάδωσης γίνεται αμέσως μόλις γίνει γνωστό το αποτέλεσμα της εντολής, δηλαδή στο στάδιο WR. Σε περίπτωση λανθασμένης πρόβλεψης, σταματά η εκτέλεση των λάθος εντολών και στον επόμενο κύκλο δρομολογείται η σωστή εντολή.

Δίνεται ο παρακάτω κώδικας :

```
1.    LOOP:     LD   R0, 0(R1)
2.              MULT R4, R0, R2
3.              SD   R4, 0(R1)
4.              SUB  R1, R1, #8
5.              BNEZ R1, LOOP
6.              LD   R5, 0(R7)
7.              ADD  R6, R8, R5
8.              SD   R6, 0(R2)
```

Αν η αρχική τιμή του καταχωρητή R1 είναι 16, εκτελέστε τον παραπάνω κώδικα και δώστε τους χρόνους δρομολόγησης, εκτέλεσης και ολοκλήρωσης των εντολών σε έναν πίνακα όπως ο παρακάτω :

| OP | IS | EX | WR | CMT | Σχόλιο |
|---|---|---|---|---|---|
| L.D R0, 0(R1) | 1 | 2-5 | 6 | 7 | |
| MULT R4, R0, R2 | ?? | ?? | ?? | ?? | ?? |

Στο πεδίο “Σχόλιο” δικαιολογήστε τυχόν καθυστερήσεις μεταξύ IS-EX, EX-WR και WR-CMT καθώς και ακυρώσεις εντολών.

**B.** Δίνεται αρχιτεκτονική η οποία υλοποιεί τον αλγόριθμο Tomasulo χρησιμοποιώντας ROB για in-order commit εντολών. Το pipeline του επεξεργαστή περιέχει τα στάδια Issue (IS), Execute (EX), Write Result (WR) και Commit (CMT), αγνοούμε δηλαδή το IF. Ισχύουν επίσης τα ακόλουθα :

1. Τα IS, WR, CMT απαιτούν 1 κύκλο.
2. Στο στάδιο EX εκτελείται και ο υπολογισμός της διεύθυνσης μιας προσπέλασης στη μνήμη καθώς και η ίδια η προσπέλαση.
3. Το σύστημα έχει άπειρα reservation stations καθώς και άπειρες θέσεις στον ROB.
4. Υπάρχει 1 Integer Functional Unit το οποίο χρησιμοποιείται για όλες τις πράξεις integer αριθμών καθώς και τις προσπελάσεις στη μνήμη.
5. Υπάρχουν 2 Floating Point Functional Units, 1 FP adder (για προσθέσεις/αφαιρέσεις) και 1 FP multiplier (για πολλαπλασιασμούς/διαιρέσεις). Ο FP adder είναι πλήρως pipelined (στη συχνότητα του εξεργαστή), ενώ ο FP multiplier όχι.
6. Δεν υπάρχει προώθηση μεταξύ των διαφόρων FUs. Τα αποτελέσματα μεταφέρονται σε κάθε reservation station μέσω του CDB.
7. Οι εντολές που χρησιμοποιούν το Integer FU απαιτούν 1 κύκλο εκτέλεσης, ενώ για αυτές που χρησιμοποιούν τα Floating Point FUs απαιτούνται 5 κύκλοι για ADD/SUB, 10 κύκλοι για MULT και 14 κύκλοι για DIV.
8. Ο επεξεργαστής είναι 2-wide superscalar, επομένως σε κάθε κύκλο 2 εντολές μπορούν να γίνονται issued, να γίνονται commit ή να βρίσκονται στο WR στάδιο (υπάρχουν 2 CDBs).

Το σύστημα εκτελεί τις παρακάτω εντολές :

```
1.    L.D       F2, 0(R1)
2.    ADD.D     F10,  F2, F2
3.    MUL.D     F6, F10, F2
4.    L.D       F4, 0(R2)
5.    ADD.D     F8, F4, F4
6.    ADD.D     F2, F4, F2
7.    DIV.D     F8, F8, F2
8.    S.D       F8, 0(R3)
9.    DADDUI    R1, R1, #8
10.   DADDUI    R2, R2, #8
```

Δώστε τους χρόνους δρομολόγησης, εκτέλεσης και ολοκλήρωσης των εντολών συμπληρώνοντας έναν πίνακα όπως ο παρακάτω :

| OP | IS | EX | WR | CMT | Σχόλιο |
|---|---|---|---|---|---|
| L.D F2, 0(R1) | 1 | 2-2 | 3 | 4 | |
| ADD.D F10,F2,F2 | ?? | ?? | ?? | ?? | ?? |

Στο πεδίο “Σχόλιο” δικαιολογήστε τυχόν καθυστερήσεις μεταξύ IS-EX, EX-WR και WR-CMT.

## Θέμα 4ο (2 μονάδες)

Θεωρούμε το ακόλουθο κομμάτι κώδικα:

```
#define N 1024
float A[N], B[N];

for(i=0; i<N; i+=1)
     B[i] += 2*A[i];
```

Κάνουμε τις εξής υποθέσεις:

1. Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι direct mapped, write-allocate, και έχει μέγεθος 2KB. Το μέγεθος του block είναι 16 bytes.
2. Το μέγεθος ενός float είναι 4 bytes.
3. Δήλωση διαδοχικών μεταβλητών (βαθμωτών και μη) στο πρόγραμμα συνεπάγεται αποθήκευσή τους σε διαδοχικές θέσεις στη μνήμη.

**Α**. Βρείτε το συνολικό ποσοστό αστοχίας (miss rate) για τις αναφορές που γίνονται στην μνήμη στον παραπάνω κώδικα.

**Β**. Ποιες από τις παρακάτω τεχνικές βελτιστοποίησης *επιπέδου λογισμικού* θα ακολουθούσατε προκειμένου να βελτιώσετε την απόδοση του παραπάνω κώδικα; Δικαιολογήστε την επιλογή ή μη-επιλογή κάθε τεχνικής. Για αυτές που επιλέξατε, *παρουσιάστε* τον αντίστοιχο κώδικα και υπολογίστε το νέο ποσοστό αστοχίας που αυτός συνεπάγεται.

1. loop unrolling
2. merging arrays
3. loop blocking
4. loop distribution

**Γ**. Ποιες από τις παρακάτω τεχνικές βελτιστοποίησης *επιπέδου υλικού* θα ακολουθούσατε προκειμένου να βελτιώσετε την απόδοση του παραπάνω κώδικα; Δικαιολογήστε όπως και πριν την απάντησή σας. Για τις τεχνικές που επιλέξατε, πώς διαμορφώνεται το νέο ποσοστό αστοχίας;

1. αύξηση block size σε 32 bytes (με διατήρηση της χωρητικότητας της cache)
2. αύξηση associativity σε 2-way (με διατήρηση της χωρητικότητας της cache)
3. προσθήκη victim cache
4. χρήση μηχανισμού hardware prefetching