

Θέμα 1**α) Τι υποδηλώνει το «τείχος ισχύος» στους σύγχρονους επεξεργαστές;**

Το "τείχος ισχύος" (power wall) αναφέρεται στους περιορισμούς που προκύπτουν από την αυξημένη κατανάλωση ενέργειας και την παραγόμενη θερμότητα όταν προσπαθούμε να αυξήσουμε την απόδοση των επεξεργαστών. Οι σύγχρονοι επεξεργαστές πρέπει να ισορροπούν την απόδοση με την αποδοτικότητα της ενέργειας για να ξεπεράσουν αυτά τα όρια.

β) Η εικονική μνήμη χρησιμοποιεί την μέθοδο της ταυτόχρονης εγγραφής (write through) για λόγους ακεραιότητας δεδομένων;

Η μέθοδος "write-through" αναφέρεται συνήθως στην πολιτική εγγραφής στη **μνήμη cache** και όχι στην εικονική μνήμη. Στην **εικονική μνήμη**, χρησιμοποιούνται άλλες τεχνικές όπως η σελιδοποίηση (paging) ή η κατάτμηση (segmentation) για την διαχείριση της μνήμης και την ακεραιότητα των δεδομένων.

Η μέθοδος "write-through" στην cache διασφαλίζει ότι κάθε εγγραφή δεδομένων ενημερώνει τόσο την cache όσο και την κύρια μνήμη άμεσα, διατηρώντας έτσι την ακεραιότητα των δεδομένων μεταξύ των δύο.

γ) Μπορούμε να αποφύγουμε όλες τις καθυστερήσεις με την «πρώθηση»;

Δεν μπορούμε. Για `lw $t0, 0($s0)` θα έχουμε 1 stall αν για 2, αλλά ακόμα
`addi $t0, $t0, 1` έχουμε καθυστέρηση

δ) Στην ένθετη (nested) κλήση διαδικασιών, τι θα πρέπει ο καλών να αποθηκεύσει στη στοίβα;

Στις nested διαδικασίες, συνήθως προτιμάται η μέθοδος "**callee-save**" για τη διαχείριση των καταχωρητών. Αυτό συμβαίνει επειδή στις nested διαδικασίες, η εκάστοτε επίπεδο κλήσης θα μπορούσε να επηρεάσει την κατάσταση των καταχωρητών του παραπάνω επιπέδου αν χρησιμοποιούνταν η μέθοδος "caller-save". Με τη μέθοδο "callee-save", η κληθείσα διαδικασία διατηρεί την ακεραιότητα των καταχωρητών που χρησιμοποιεί χωρίς να επηρεάζει το επίπεδο κλήσης.

Συνεπώς ο καλών θα πρέπει να σώσει όλους τους καταχωρητές στον πρόλογο και να τους επαναφέρει στον επίλογο.

ε) Η αύξηση των ψηφίων του εκθέτη αυξάνει την ακρίβεια του αριθμού κινητής υποδιατολής;

Αυτή η δήλωση είναι σωστή. Η αύξηση των ψηφίων του εκθέτη σε έναν αριθμό κινητής υποδιατολής συνήθως αυξάνει την ακρίβεια του αριθμού. Αυτό συμβαίνει διότι περισσότερα ψηφία στον εκθέτη επιτρέπουν την αποθήκευση περισσότερων δεκαδικών ψηφίων, παρέχοντας έτσι μεγαλύτερη ακρίβεια κατά τις πράξεις.

στ) Τι είναι η διαδικασία «εξισορρόπηση φθοράς» στις μνήμες Φλάς;

Η διαδικασία "εξισορρόπηση φθοράς" (wear leveling) στις μνήμες flash είναι ένας μηχανισμός που χρησιμοποιείται για να μειώσει τη φθορά των κυττάρων μνήμης flash, η οποία προκαλείται από την επανειλημμένη εγγραφή και διαγραφή δεδομένων. Η φθορά μπορεί να επηρεάσει την απόδοση και την αξιοπιστία της μνήμης.

Η διαδικασία εξισορρόπησης φθοράς λειτουργεί κατανομώντας τις εγγραφές και διαγραφές δεδομένων σε διάφορα τμήματα της μνήμης flash, αντί να επαναλαμβάνονται συνεχώς στα ίδια σημεία. Με αυτόν τον τρόπο, η φθορά κατανέμεται ομοιόμορφα σε όλη τη μνήμη, αυξάνοντας έτσι τη διάρκεια ζωής της μνήμης.

ζ) Με την μέθοδο της επικοινωνίας με «χειραψία» (hand-shaking) τί κερδίζουμε;

Με τη μέθοδο της επικοινωνίας με "χειραψία" στο πλαίσιο των λειτουργικών συστημάτων υπολογιστών κερδίζουμε:

1. Κοινοποίηση πόρων: Δυνατότητα να μοιραζόμαστε πόρους όπως εκτυπωτές, φακέλους και δίσκους.
2. Δικτύωση: Δημιουργία δικτύων για επικοινωνία και συνεργασία μεταξύ υπολογιστών.
3. Ανταλλαγή δεδομένων: Αποστολή και λήψη δεδομένων μεταξύ συστημάτων.
4. Ασφάλεια και αυθεντικοποίηση: Προστασία της επικοινωνίας και αυθεντικοποίηση των χρηστών και των συστημάτων.

η) Ποιο είναι το κύριο πλεονέκτημα της μεθόδου DMA (Direct Memory Access);

Το βασικό πλεονέκτημα του DMA (Direct Memory Access) είναι η ικανότητά του να επιτρέπει σε συσκευές εισόδου/εξόδου να μεταφέρουν δεδομένα απευθείας από και προς τη μνήμη του συστήματος, χωρίς την ανάμειξη του επεξεργαστή. Αυτό επιτρέπει στον επεξεργαστή να ασχολείται με άλλες εργασίες κατά τη διάρκεια της μεταφοράς δεδομένων, βελτιώνοντας έτσι την απόδοση και την αποδοτικότητα του συστήματος.

Θέμα 2

Ζητούμενο 1

α) (15%) Ένας επεξεργαστής (P1) εκτελεί εντολές φόρτωσης (load instructions) σε 10 κύκλους, εντολές αποθήκευσης (store instructions) σε 8 κύκλους, αριθμητικές εντολές (arithmetic instructions) σε 4 κύκλους και οδηγίες διακλάδωσης (branch instructions) σε 4 κύκλους. Άλλος επεξεργαστής (P2) λειτουργεί με διπλάσια συχνότητα, ωστόσο, οι λαμβάνοντες χρόνοι των οδηγιών φόρτωσης, αποθήκευσης, αριθμητικής και διακλάδωσης αυξάνονται κατά 2, 2, 2 και 1 κύκλους, αντίστοιχα. Σε μια εφαρμογή Α, όπου το 20% όλων των εντολών είναι οδηγίες φόρτωσης, το 20% όλων των εντολών είναι οδηγίες αποθήκευσης, το 50% όλων των εντολών είναι αριθμητικές οδηγίες και το 10% όλων των εντολών είναι οδηγίες διακλάδωσης, ποιος επεξεργαστής είναι πιο γρήγορος; (Ο μεταγωγιστής που χρησιμοποιείται για τη δημιουργία εντολών για τον P2 είναι ο ίδιος με τον P1. Έτσι, παράγει τον ίδιο αριθμό εντολών για το πρόγραμμα Α).

$$f_2 = 2f_1 \Rightarrow \\ \text{cycles}_1 = 2 \cdot \text{cycles}_2 \\ \text{instr}_1 = \text{instr}_2$$

	P1	P2	
load instruct :	20% 10cc	12cc	$CPI_1 = 20\% \cdot 10 + 20\% \cdot 8 + 40\% \cdot 4 + 10\% \cdot 4 = 4,6cc$
store instruct :	20% 8cc	10cc	$CPI_2 = 20\% \cdot 12 + 20\% \cdot 10 + 40\% \cdot 6 + 20\% \cdot 5 = 7,3cc$
arithmetic inst :	40% 4cc	6cc	$\text{tex}_1 = CPI_1 \cdot \text{instr}_1 \cdot \text{cycles}_1$
branch instruct :	10% 4cc	5cc	$\text{tex}_2 = CPI_2 \cdot \text{instr}_2 \cdot \text{cycles}_2$
			$\Rightarrow \text{tex}_1 > \text{tex}_2 \Rightarrow P2 \text{ γρηγορότερος}$

Ζητούμενο 2

\$s0 ← pow, \$s1 ← x

```
addi $s0, $0, 1
add $s1, $0, $0
addi $t0, $0, 128
while: beq $s0, $t0, done
      sll $s0, $s0, 1
      addi $s1, $s1, 1
      j while
done:
```

```
int pow = 1;
int x = 0;
while (pow != 128) {
    x = x * 2;
    pow = pow + 1;
}
```

Θέμα 3

Υποθέτουμε ότι η αρχική τιμή του \$s3 είναι ίση με τον δεκαεξαδικό αριθμό 13516. Υποθέστε την κλασική αρχιτεκτονική σωλήνωσης του MIPS αποτελούμενη από τα στάδια IF, ID, EX, MEM, WB. Όλα τα στάδια διαρκούν έναν κύκλο. Κατά τον εντοπισμό μιας εντολής άλματος υπό συνθήκη, ο επεξεργαστής εισάγει καθυστερήσεις (stalls) μέχρι την επίλυση η οποία πραγματοποιείται στο στάδιο EX. Τέλος, υποθέστε ότι η εγγραφή σε έναν καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου.

```
add $t5, $zero, $zero
add $t0, $zero, $zero
L:   sll $t1, $t0, 2
     add $t2, $t1, $s1
     lw  $t4, 0($t2)
     add $t3, $t1, $s2
     add $t5, $t5, $t3
     add $t4, $t4, $t5
     sw  $t4, 0($t2)
     addi $t0, $t0, 1
     slt $t6, $t0, $s3
     bne $t6, $zero, L
```

Ζητούμενο 1

(10%) Υποθέστε ότι δεν υπάρχουν σχήματα προώθησης. Εκτελέστε την 1η επανάληψη του βρόχου (μέχρι και την 1η εντολή της 2ης επανάληψης) και χρησιμοποιήστε ένα διάγραμμα χρονισμού, σχολιάζοντας τον λόγο οποιασδήποτε καθυστέρησης. Πόσοι κύκλοι απαιτούνται για την εκτέλεση ολόκληρου του κώδικα; Υποδείξτε και εξηγήστε τους πιθανούς κινδύνους (hazards) που μπορούν να προκύψουν κατά την εκτέλεση, καθώς και τον τρόπο με τον οποίον αυτοί αντιμετωπίζονται;

$$\$s3 = 13516 = 16^2 + 3 \cdot 16 + 5 = 309 \xrightarrow[\$t0]{\$t0=0} 309 \text{ loops}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
add	F	D	X	M	W																				
add		F	D	X	M	W																			
sll			F	D	-	-	X	M	W																
add				F	-	-	D	-	-	X	M	W													
lw							F	-	-	D	-	-	X	M	W										
add									F	-	-	D	X	M	W										
add											F	D	-	-	X	M	W								
add												F	-	-	D	-	-	X	M	W					
sw															F	-	-	D	-	-	X	M			
addi																		F	-	-	D	X			
slt																						F	D		
bne																								F	

	26	27	28	29	30	31	32	33
sw	W							
addi	M	W						
slt	-	-	X	M	W			
bne	-	-	D	-	-	X	M	W
sll						F	D	

total cycles:

2 → add, add εκτός loop

30 → 1st loop

28 → 307 loops (ox stalls 5,6)

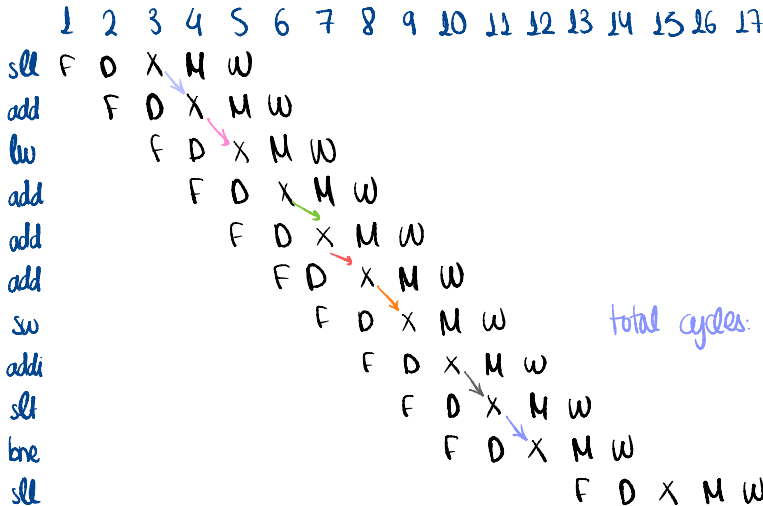
30 → last loop

$$\Rightarrow \text{total} = 2 + 30 + 307 \cdot 28 + 30 = 8658 \text{cc}$$

Ζητούμενο 2

(10%) Υποθέστε τώρα ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης. Χρησιμοποιήστε όπως και πριν το διάγραμμα χρονισμού, υποδεικνύοντας τις προωθήσεις που γίνονται. Πόσοι κύκλοι απαιτούνται για την εκτέλεση ολόκληρου του κώδικα;

```
add $t5, $zero, $zero
add $t0, $zero, $zero
L:   sll $t1, $t0, 2
      add $t2, $t1, $s1
      lw $t4, 0($t2)
      add $t3, $t1, $s2
      add $t5, $t5, $t3
      add $t4, $t4, $t5
      sw $t4, 0($t2)
      addi $t0, $t0, 1
      slt $t6, $t0, $s3
      bne $t6, $zero, L
```



total cycles: 2 + 308 + 12 + 14

Ζητούμενο 3

(10%) Μπορείτε να επιτύχετε καλύτερη επίδοση / μειώσετε τον αριθμό των προωθήσεων αναδιατάσσοντας τον κώδικα (με τις απαραίτητες βέβαια μετατροπές για να μην αλλάξετε τη σημασιολογία του προγράμματος); Να υποδείξετε ότι μετατροπές χρειάζονται σε εντολές προκειμένου να διατηρηθεί η σημασιολογία του προγράμματος. Δείξτε όπως και πριν το διάγραμμα χρονισμού, υποδεικνύοντας και τις προωθήσεις που γίνονται. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα;

Θεμα 4

Οι πίνακες περιέχουν στοιχεία κινητής υποδιαστολής διπλής ακρίβειας, μεγέθους 8 bytes το καθένα. Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων, χωρητικότητας 512 bytes, η οποία είναι αρχικά άδεια. Το μέγεθος του block είναι 32 bytes, η ελάχιστη ποσότητα δεδομένων που μπορεί να διευθυνοδοτηθεί είναι το 1 byte, και γίνεται χρήση 32-bit διευθύνσεων. Όλες οι μεταβλητές, πλην των στοιχείων των πινάκων, μπορούν να αποθηκευτούν 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.

Το πρώτο στοιχείο του πίνακα x βρίσκεται στη διεύθυνση 0xFD00A000, του y στη 0xA080C100 και του z στη 0xF1001000. Ζητείται να υπολογιστούν ο αριθμός των read hits, read misses, write hits και write misses για τις διάφορες αναφορές που γίνονται στη μνήμη, εξηγώντας σε κάθε περίπτωση τις απαντήσεις σας, για όλη την εκτέλεση του παραπάνω κώδικα για τις κάτωθι περιπτώσεις

```
#define M 32
int i;
double x[M], y[M], z[M];
for (i = 0; i < M; i++) {
    z[i] = x[i] * y[i];
}
```

Ζητούμενο 1

(1) Κρυφή μνήμη άμεσης απεικόνισης (direct-mapped), write-allocate, write-back

$$\begin{aligned} \text{element size} &= 8 \text{ bytes} \\ \text{block size} &= 32 \text{ bytes} \end{aligned} \quad \left. \begin{aligned} \text{elem} / \text{block} &= 32/8 = 4 \\ \text{cache size} &= 512 \text{ bytes} = 2^9 \\ \text{block size} &= 32 \text{ bytes} = 2^5 \end{aligned} \right\} \begin{aligned} \# \text{blocks} &= 2^9/2^5 = 2^4 = 16 \\ \# \text{sets} &= 16 \end{aligned}$$

$$\text{block size} = 2^5 \Rightarrow \text{offset} = 5 \text{ bits}$$

$$\# \text{sets} = 16 = 2^4 \Rightarrow \text{index} = 4 \text{ bits}$$

$$\text{address size} = 32 \text{ bits} \Rightarrow \text{tag} = 23 \text{ bits}$$

0

$$\begin{aligned} x: 0 \times \text{FD00A000}_{16} &= \dots 0000 \ 0000 \ 0000 \Rightarrow \text{index} = 0000_2 = 0 \quad \text{offset} = 00000_2 = 0 \\ y: 0 \times \text{A080C100}_{16} &= \dots 0001 \ 0000 \ 0000 \Rightarrow \text{index} = 1000_2 = 8 \quad \text{offset} = 00000_2 = 0 \\ z: 0 \times \text{F1001000}_{16} &= \dots 0000 \ 0000 \ 0000 \Rightarrow \text{index} = 0000_2 = 0 \quad \text{offset} = 00000_2 = 0 \end{aligned}$$

$$\text{set 0: } x[0-3] \quad z[0-3] \quad i=0-3: \sim \text{ (read hits, write hits, read misses, write misses)}$$

$$\text{set 1: } x[4-7] \quad z[4-7] \quad m \ m \ m \quad (3, 0, 5, 4)$$

$$\vdots \quad \vdots \quad m \ h \ m$$

$$\text{set 7: } x[28-31] \quad z[28-31] \quad m \ h \ m \quad \text{total} = 8(3, 0, 5, 4) = (24, 0, 40, 32)$$

$$\text{set 8: } y[0-3] \quad m \ h \ m$$

$$\text{set 9: } y[4-7] \quad m \ h \ m$$

$$\vdots \quad \vdots$$

$$\text{set 15: } y[28-31]$$

Ζητούμενο 2

(2) Κρυφή μνήμη συσχέτισης δύο δρόμων (2-way set associative), write-allocate, write-back, με LRU πολιτική

$$\begin{aligned} \text{block size} &= 2^5 \Rightarrow \text{block offset} = 5 \text{ bits} \\ \# \text{sets} &= 16/2 = 8 = 2^3 \Rightarrow \text{index} = 3 \text{ bits} \\ \text{address size} &= 32 \text{ bits} \Rightarrow \text{tag} = 24 \text{ bits} \end{aligned} \quad \begin{aligned} x: 0 \times \text{FD00A000}_{16} &= \dots 0000 \ 0000 \Rightarrow \text{offset} = 0 \quad \text{index} = 0 \\ y: 0 \times \text{A080C100}_{16} &= \dots 0000 \ 0000 \Rightarrow \text{offset} = 0 \quad \text{index} = 0 \\ z: 0 \times \text{F1001000}_{16} &= \dots 0000 \ 0000 \Rightarrow \text{offset} = 0 \quad \text{index} = 0 \end{aligned}$$

set 0:	x(0-3)	y(0-3)	z(0-3)	i=0-3:	~ (read hits, write hits, read misses, write misses)
set 1:	x(4-7)	y(4-7)	z(4-7)	m	m m (0,0,8,4)
:	:	:	:	m	m m
set 7:	x(28-31)	y(28-31)	z(28-31)	m	m m total = 8(0,0,8,4) = (0,0,64,32)
				m	m m