Πανεπιστήμιο Κρήτης	**Δημήτρης Νικολόπουλος, Χριστόφορος Κάχρης**
Τμήμα Επιστήμης Υπολογιστών	**18 Ιουνίου 2010, 09:00–12:00**
ΗΥ225: Οργάνωση Υπολογιστών, Άνοιξη 2010	**Η εξέταση γίνεται με κλειστές σημειώσεις**
Εξέταση Α' Περιόδου	**Τα θέματα επιστρέφονται, ΚΑΛΗ ΕΠΙΤΥΧΙΑ!**
Ονοματεπώνυμο:	
Αριθμός Μητρώου:	

Ερώτημα 1 (22 βαθμοί): Assembly και κρυφές μνήμες

1. Έστω επεξεργαστής με το σύνολο εντολών του MIPS. Υλοποιήστε τον διπλανό κώδικα C σε assembly (με ή χωρίς βρόχους), φροντίζοντας να ελαχιστοποιήσετε τις προσβάσεις στη μνήμη (`lw` | `sw`). Θεωρήστε ότι οι πίνακες a και b έχουν δηλωθεί όπως φαίνεται δίπλα και ότι ο πίνακας b έχει τυχαίες ακέραιες τιμές που έχουν διαβαστεί από την κονσόλα. **(10 βαθμοί)**

```
for (i=0; i<4; i++)
  for (j=0; j<4; j++)
    a[i] = a[i] + b[j];
```

```
.data
.align 2
a:      .word 0,0,0,0
b:      .size 16
...
```

Λύση, 0.66 βαθμοί ανά εντολή από main έως bne.Στη λύση που δίνουμε υποθέτουμε ότι σε κάθε επανάληψη του εξωτερικού βρόχου φορτώνουμε τα στοιχεία του b από τη μνήμη στον ίδιο καταχωρητή (σε διαδοχικές επαναλήψεις του εσωτερικού βρόχου). Η λύση αυτή δεν είναι η βέλτιστη ως προς τις προσπελάσεις στη μνήμη αλλά τη θεωρούμε σωστή. Η βέλτιστη λύση ξετυλίγει τον εσωτερικό βρόχο (unrolling) στην πρώτη επανάληψη του εξωτερικού βρόχου, αποθηκεύει κάθε στοιχείο του b σε διαφορετικό καταχωρητή, και στις υπόλοιπες επαναλήψεις του εξωτερικού βρόχου διαβάζει τα στοιχεία του b από καταχωρητές και όχι από τη μνήμη. Η δεύτερη λύση επιτυγχάνει περαιτέρω μείωση των προσπελάσεων στη μνήμη σε σχέση με την προτεινόμενη λύση (8 αντί για 20) και επιβραβεύεται με 2 επιπλέον βαθμούς (12/10) για όποιον την έδωσε. Στη λύση έχουμε δώσει σταθερές τιμές στα στοιχεία του πίνακα b και όχι τον κώδικα που διαβάζει από την κονσόλα, χάρην συντομίας.

```
        .data
a:      .word 0,0,0,0
b:      .word 3,5,7,9
        .text
        .globl main
main:   la $2, a
        addi $8, $0, 4          #$8 = 4, sets loop bound
        addi $4, $0, 0          #i=0
loop_i: addi $5, $0, 0          #j=0
        addi $6, $0, 0          #initialize a[i] = 0
        la $3, b                #load b[0]
loop_j: lw $7, 0($3)            #load  b[j]
        add $6, $6, $7          #a[i] = a[i] + b[j]
        addi $3, $3, 4          #point to b[j+1]
        addi $5, $5, 1          #j = j + 1
        bne $5, $8, loop_j      #j < 4
        sw $6, 0($2)            #store a[i]
        addi $2, $2, 4          #point to a[i+1]
        addi $4, $4, 1          #i = i + 1
        bne $4, $8, loop_i      #i < 4
        li $v0,10
        syscall
```

2. Έστω ότι ο παραπάνω επεξεργαστής έχει κρυφή μνήμη μονοσήμαντης απεικόνισης με χωρητικότητα 4 λέξεων (κάθε λέξη είναι 4 bytes). Χωρίς να σχεδιάσετε την κρυφή μνήμη υπολογίστε πόσες ευστοχίες και πόσες αστοχίες έχει ο παραπάνω κώδικας: (α) αν η κρυφή μνήμη χρησιμοποιεί πολιτική *write allocate* **(3 βαθμοί)**, (β) αν η κρυφή μνήμη χρησιμοποιεί πολιτική *write no allocate* **(3 βαθμοί)**. (γ) Ποια από τις δύο πολιτικές είναι καλύτερη και γιατί; **(3 βαθμοί)**

Λύση

(α) write-allocate: Παρατηρούμε ότι τα ζευγάρια (a[0], b[0]), (a[1], b[1]), κλπ. απεικονίζονται στο ίδιο block στην κρυφή μνήμη Η ακολουθία προσβάσεων στη μνήμη είναι: load b[0], load b[1], load b[2], load

b[3] store a[0] load b[0], load b[1], load b[2], load b[3] store a[1] load b[0], load b[1], load b[2], load b[3] store a[2] load b[0], load b[1], load b[2], load b[3] store a[3]. Η πρώτη επανάληψη του εσωτερικού βρόχου προκαλεί 5 αστοχίες, ενώ κάθε μία από τις υπόλοιπες επαναλήψεις *i* προκαλεί 2 αστοχίες λόγω της σύγκρουσης μεταξύ των στοιχείων a[i-1], b[i-1] και a[i], b[i]. Συνολικά ο κώδικας προκαλεί 11 αστοχίες σε 20 προσπελάσεις (9 ευστοχίες). (1 βαθμός για σωστή εφαρμογή της write-allocate, 2 βαθμοί για σωστό υπολογισμό ευστοχιών αστοχιών)

(β) write-noallocate: Σε αυτή την περίπτωση δεν δεσμεύεται χώρος στην κρυφή μνήμη για τα στοιχεία του πίνακα a[]. Κάθε write στον a[] μετράει ως αστοχία εφόσον πρέπει να γίνει στην κύρια μνήμη. Η πρώτη επανάληψη έχει 5 αστοχίες ενώ κάθε επόμενη επανάληψη έχει 1 αστοχία (αυτή για το write του στοιχείου του a[]). Ο κώδικας συνολικά έχει 8 αστοχίες και 12 ευστοχίες. (1 βαθμός για σωστή εφαρμογή της write-noallocate, 2 βαθμοί για σωστό υπολογισμό ευστοχιών αστοχιών)

(γ) Η πολιτική write-noallocate είναι καλύτερη γιατί έχει μικρότερο ποσοστό αστοχιών επί του συνόλου των προσπελάσεων.

3. Έστω ότι έχετε την επιλογή μεταξύ πολιτικής *write-back* και πολιτικής *write-through* για την κρυφή μνήμη του παραπάνω επεξεργαστή. Ποια πολιτική θα επιλέγατε, την *write-through*, την *write-back*, ή οποιαδήποτε από τις δύο γιατί δεν υπάρχει διαφορά; Εξηγήστε την απάντησή σας. **(3 βαθμοί)**

Σωστή απάντηση I: Θεωρώντας από το ερώτημα 2(γ) καλύτερη την πολιτική write-noallocate, τότε δεν υπάρχει διαφορά μεταξύ της write-through ή της write-back (ίδιος αριθμός αστοχιών και προσβάσεων στην κύρια μνήμη).

Εναλλακτικά:

Σωστή απάντηση II: write-through γιατί η πολιτική write-back έχει μεν ίδιο αριθμό αστοχιών και προσβάσεων στην κύρια μνήμη, αλλά είναι ουσιαστικά ανενεργή και δαπανά πόρους στο υλικό (πολυπλοκότητα και χώρος κρυφής μνήμης) χωρίς να χρησιμοποιείται ποτέ.

Εξήγηση: Εάν χρησιμοποιήσουμε πολιτική write-back σε συνδυασμό με πολιτική write-noallocate τότε εφόσον τα στοιχεία του b[] που γράφουμε δεν αποθηκεύονται ποτέ στην κρυφή μνήμη, όλες οι αστοχίες κοστίζουν μία πρόσβαση στην κύρια μνήμη (με άλλα λόγια η πολιτική write-back δεν έχει νόημα να χρησιμοποιηθεί και δεν θα βελτιώσει το ποσοστό αστοχίας εφόσον δεν υπάρχουν dirty blocks στην κρυφή μνήμη, κοστίζει δε σε πολυπλοκότητα και χώρο στην κρυφή μνήμη σε σχέση με την πολιτική write-through). Εάν χρησιμοποιήσουμε πολιτική write-through με write-noallocate τότε κάθε αστοχία κοστίζει μία πρόσβαση στην κύρια μνήμη.

Ερώτημα 2 (11 βαθμοί): Επίδοση επεξεργαστών

1. Έστω επεξεργαστής με υλοποίηση πολλαπλών κύκλων με τα χαρακτηριστικά εκτέλεσης εντολών που φαίνονται στον πίνακα αριστερά. Ο επεξεργαστής εκτελεί το μείγμα εντολών που φαίνεται στον πίνακα δεξιά:

Βήμα	R-type εντολές	Εντολές πρόσβασης στη μνήμη (load, store)	Διακλαδώσεις (branch)	Άλματα (jump)
IF	$IR <= Memory[PC], PC <= PC + 4$			
ID/RF	$A = REG[IR[25:21]]$ $B = REG[IR[20:16]]$ $ALUOut = PC + (\text{sign-extend}(IR[15:0]) << 2)$			
EX	$ALUOut <= AopB$	$ALUOut <= A +$ sign-extend$(IR[15:0])$	$if(A == B)$ $PC <= ALUOut$	$PC <= [PC[31:28], IR[25:0], 2'b00)]$
MEM	$Reg[IR[15:11]] <= ALUOut$	$lw : MDR <= Memory[ALUOut]$ $sw : Memory[ALUOut] <= B$		
WB		$lw : Reg[IR[20:16]] <= MDR$		

R-type	50%
Loads	20%
Stores	10%
Branches	10%
Jumps	10%

Ο επεξεργαστής έχει ρολόϊ στα 2 GHz. Ποιο είναι το CPI αυτού του επεξεργαστή; **(3 βαθμοί)**

Λύση

$$CPI = 0.5 \times 4 + 0.2 \times 5 + 0.1 \times 4 + 0.1 \times 3 + 0.1 \times 3 = 4.0 \quad (1)$$

2. Έστω επεξεργαστής με ρολόι 2.5 GHz, μνήμη δεδομένων (μόνο, όχι εντολών) που έχει χρόνο πρόσβασης 2 κύκλους και υπόλοιπα χαρακτηριστικά όπως αυτά του επεξεργαστή του ερωτήματος 1. (α) Ποιο είναι το CPI αυτού του επεξεργαστή για το παραπάνω μείγμα εντολών; **(5 βαθμοί)** (β) Ποιος επεξεργαστής (1 ή 2) είναι πιο γρήγορος με βάση το CPI, τη συχνότητα του ρολογιού και το παραπάνω μείγμα εντολών; **(3 βαθμοί)**

Λύση

(α)

$$CPI = 0.5 \times 4 + 0.2 \times 6 + 0.1 \times 5 + 0.1 \times 3 + 0.1 \times 3 = 4.3 \quad (2)$$

(β) Έστω I το πλήθος των εντολών που εκτελούνται

$$T_1 = CPI_1 \times I \times cycletime = 4.0 \times I \times 0.5ns = 2Ins \quad (3)$$

$$T_2 = CPI_2 \times I \times cycletime = 4.3 \times I \times 0.4ns = 1.72Ins \quad (4)$$

Άρα ο δεύτερος επεξεργαστής είναι ταχύτερος του πρώτου.

Ερώτημα 3 (14 βαθμοί): Ομοχειρία

1. Σε επεξεργαστή με ομοχειρία 5 βαθμίδων (IF, ID, EX, MEM, WB) εκτελούμε την ακολουθία εντολών που φαίνεται δίπλα. Έστω ότι ο επεξεργαστής *δεν υλοποιεί μηχανισμούς προώθησης (forwarding)*. Κατασκευάστε έναν πίνακα όπως ο παρακάτω που να δείχνει τους κύκλους στους οποίους εκτελείται κάθε βαθμίδα της κάθε εντολής. Εάν κάποια εντολή πρέπει να περιμένει (φυσαλίδα καθυστέρησης) σε κάποιο κύκλο σημειώστε τη λέξη *stall*. Υποθέστε ότι το register file υποστηρίζει ταυτόχρονη εγγραφή και ανάγνωση στην ίδια διεύθυνση (εγγραφή στο πρώτο μισό του κύκλου, ανάγνωση στο δεύτερο μισό του κύκλου). **(5 βαθμοί)**

```
add   $2,$1,$3
add   $10,$2,$5
add   $11,$6,$2
lw    $8,100($10)
add   $14,$2,$8
```

Παρακάτω υποθέτουμε ότι ο επεξεργαστής έχει μία μονάδα εκτέλεσης (ALU) η οποία σε κάθε κύκλο μπορεί να χρησιμοποιηθεί από μία εντολή. Εναλλακτικές σωστές λύσεις υπάρχουν αν υποθέσουμε ότι έχουμε πολλές μονάδες εκτέλεσης (στην οποία περίπτωση δεν υπάρχουν stalls σε εντολές που συμπίπτουν με άλλες εντολές στη βαθμίδα EX). (1 βαθμός ανά σωστή εντολή).

	Κύκλοι επεξεργαστή													
Εντολή	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`add $2,$1,$3`	IF	ID	EX	MEM	WB									
`add $10,$2,$5`		IF	ID	stall	stall	EX	MEM	WB						
`add $11,$6,$2`			IF	ID	stall	stall	EX	MEM	WB					
`lw $8,100($10)`				IF	ID	stall	stall	stall	EX	MEM	WB			
`add $14,$2.$8`					IF	ID	stall	stall	stall	stall	stall	EX	MEM	WB

2. Έστω ότι τροποποιούμε τον παραπάνω επεξεργαστή για να υποστηρίζει προώθηση από τον καταχωρητή EX/MEM προς τη βαθμίδα EX και από τον καταχωρητή MEM/WB προς τη βαθμίδα EX. Επαναλάβετε το προηγούμενο ερώτημα και στον πίνακα που θα κατασκευάσετε *δείξτε με βέλη* όλες τις περιπτώσεις στις οποίες γίνεται προώθηση από μία εντολή σε μία άλλη. **(9 βαθμοί)**

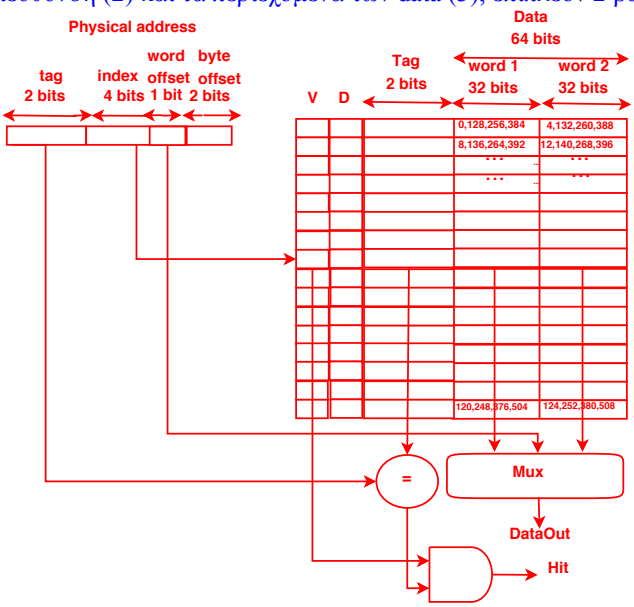
Παρακάτω υποθέτουμε ότι ο επεξεργαστής έχει μία μονάδα εκτέλεσης (ALU) η οποία σε κάθε κύκλο μπορεί να χρησιμοποιηθεί από μία εντολή. Εναλλακτικές σωστές λύσεις υπάρχουν αν υποθέσουμε ότι έχουμε πολλές μονάδες εκτέλεσης (στην οποία περίπτωση δεν υπάρχουν stalls σε εντολές που συμπίπτουν με άλλες εντολές στη βαθμίδα EX). (1.2 βαθμοί ανά σωστή εντολή και 3 βαθμοί για κάθε προώθηση).

	Κύκλοι επεξεργαστή											
Εντολή	1	2	3	4	5	6	7	8	9	10	11	12
`add $2,$1,$3`	IF	ID	EX	MEM	WB							
`add $10,$2,$5`		IF	ID	EX	MEM							
`add $11,$6,$2`			IF	ID	EX	MEM	WB					
`lw $8,100($10)`				IF	ID	EX	MEM	WB				
`add $14,$2.$8`					IF	ID	stall	EX	MEM	WB		

Ερώτημα 4 (22 βαθμοί): Κρυφές μνήμες

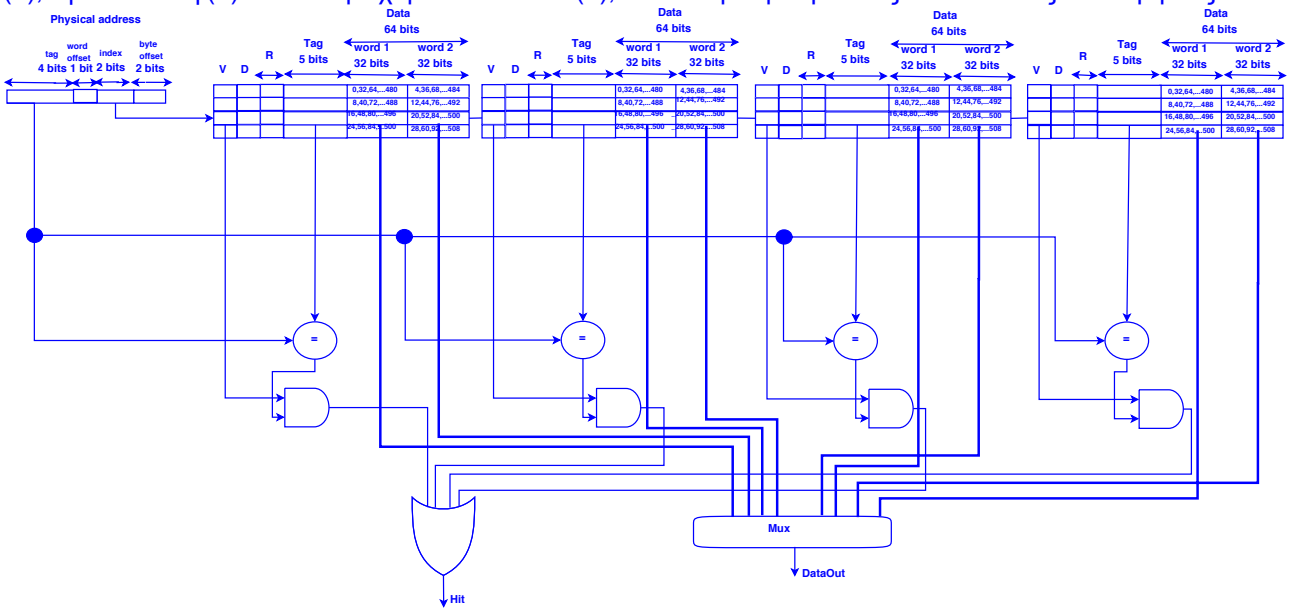
1. Σχεδιάστε μία κρυφή μνήμη με τα εξής χαρακτηριστικά: Μέγεθος λέξης 4 bytes, συνολικός χώρος δεδομένων 32 λέξεις, μέγεθος block (γραμμής) 2 λέξεις, μονοσήμαντη απεικόνιση, μέγεθος φυσικής μνήμης 128 λέξεις, πολιτική write-back. Δείξτε το πλάτος των δεδομένων και των ετικετών στην κρυφή μνήμη, πώς απεικονίζεται μία διεύθυνση φυσικής μνήμης στην κρυφή μνήμη (διαγραμματικά), ποιες και πόσες λέξεις της φυσικής μνήμης απεικονίζονται σε κάθε block της κρυφής μνήμης και τα bits ελέγχου που χρειάζεται κάθε block στην κρυφή μνήμη. **(7 βαθμοί)**

Λύση 7 βαθμοί για τους πίνακες (2) τη διεύθυνση (2) και τα περιεχόμενα των data (3), επιπλέον 2 βαθμοί για τους πολυπλέκτες και συγκριτές.



2. Επαναλάβετε το προηγούμενο ερώτημα για μία κρυφή μνήμη με την ίδια χωρητικότητα για δεδομένα (32 λέξεις) η οποία χρησιμοποιεί προσεταιριστική απεικόνιση με βαθμό διαφύλλωσης 4 (4-way set associative). Θεωρήστε ότι κάθε block διαθέτει 2 *bits χρήσης* που χρησιμοποιούνται από την πολιτική αντικατάστασης. **(8 βαθμοί)**

Λύση 8 βαθμοί για τους πίνακες (3), τη διεύθυνση (2) και τα περιεχόμενα των data (3), bonus 3 βαθμοί για τους πολυπλέκτες και συγκριτές.



3. Η κρυφή μνήμη προσεταιριστικής απεικόνισης μπορεί να επιτύχει χαμηλότερα ποσοστά αστοχίας από την κρυφή μνήμη μονοσήμαντης απεικόνισης, λόγω μείωσης των συγκρούσεων μεταξύ διαφορετικών διευθύνσεων που απεικονίζονται στο ίδιο block. Δώστε ένα παράδειγμα ακολουθίας προσπελάσεων στις δύο κρυφές μνήμες των ερωτημάτων 1 και 2, που να αποδεικνύει τον παραπάνω ισχυρισμό. Δώστε το παράδειγμα σαν μία ακολουθία διευθύνσεων φυσικής μνήμης στο δεκαδικό και δείξτε πώς η κρυφή μνήμη προσεταιριστικής απεικόνισης μειώνει τις αστοχίες. Μπορείτε να υποθέσετε ότι η κρυφή μνήμη προσεταιριστικής απεικόνισης χρησιμοποιεί πολιτική αντικατάστασης LRU. **(7 βαθμοί)**

Έστω η ακολουθία προσπελάσεων 0,128,0,128,0,128,... λόγω της σύγκρουσης μεταξύ των δύο διευθύνσεων (οποιαδήποτε ακολουθία με ζευγάρια διευθύνσεων που απέχουν 128 bytes μεταξύ τους θα

έδινε το ίδιο αποτέλεσμα). Κάθε προσπέλαση θα προκαλέσει αστοχία στην κρυφή μνήμη μονοσήμαντης απεικόνισης, αντίθετα η ίδια ακολουθία θα προκαλέσει μόνο δύο αστοχίες στη μνήμη προσεταιριστικής απεικόνισης (στην πρώτη επανάληψή της) εφόσον τα δύο ζευγάρια λέξεων μπορούν να αποθηκευθούν σε διαφορετικές banks (στήλες σχηματικά) της κρυφής μνήμης.

Ερώτημα 5 (11 βαθμοί): Εικονική μνήμη

1. Ποιο είναι το πλεονέκτημα ενός πολυεπίπεδου πίνακα σελίδων σε σχέση με έναν μονοεπίπεδο πίνακα σελίδων στα συστήματα εικονικής μνήμης; **(3 βαθμοί)**

   Οικονομία χώρου για τα προγράμματα που χρησιμοποιούν μόνο μικρά και αραιά τοποθετημένα κομμάτια του χώρου εικονικών διευθύνσεων

2. Έστω επεξεργαστής με μέγεθος διεύθυνσης 64 bits που υποστηρίζει εικονική μνήμη, με πολυεπίπεδο πίνακα σελίδων 4 επιπέδων, ίσο αριθμό γραμμών σε κάθε πίνακα κάθε επιπέδου και μέγεθος σελίδας 4 Kbytes. (α) Πόσες γραμμές έχει κάθε πίνακας (ανεξαρτήτως επιπέδου); **(3 βαθμοί)** (β) Σχεδιάστε την εικονική διεύθυνση δείχνοντας τα πεδία στα οποία διασπάται για την εύρεση της απεικόνισής της στη φυσική μνήμη (δεν χρειάζεται να δείξετε και τους πίνακες σελίδων, αρκούν τα πεδία της διεύθυνσης) **(5 βαθμοί)**.

   (α) $2^{13} = 8192$ γραμμές. (β)

Ερώτημα 6 (20 βαθμοί): DMA και συνέπεια κρυφής μνήμης

1. Μία κάρτα δικτύου με ενσωματωμένη μηχανή DMA έχει μέγιστο ρυθμό μετάδοσης δεδομένων 10 Gbit/s = 1.25 Gbytes/s και είναι συνδεδεμένη σε μία αρτηρία με ρολόϊ 100 MHz. Η αρτηρία έχει πλάτος 128 bits = 16 bytes = 4 λέξεις. Η μετάδοση δεδομένων με DMA μέσω της αρτηρίας απαιτεί 1 κύκλο αρτηρίας για διαιτησία και αρχικοποίηση της DMA και 1 κύκλο αρτηρίας ανά μεταφορά 128 bits δεδομένων. Η DMA γίνεται με χρήση bursts που μεταδίδουν 16 λέξεις = 64 bytes τη φορά. Στην αρτηρία είναι συνδεδεμένος επεξεργαστής ο οποίος απασχολεί την αρτηρία σε ποσοστό 20% του χρόνου. (α) Ποια είναι η μέγιστη παροχή δεδομένων σε Gbytes/s με DMA που επιτρέπει η αρτηρία; **(7 βαθμοί)** (β) Πόσες κάρτες δικτύου σαν την παραπάνω μπορούμε να συνδέσουμε στην αρτηρία χωρίς κίνδυνο να χάνουμε δεδομένα όταν αυτές εκτελούν DMA και αξιοποιούν πλήρως τον χρόνο της αρτηρίας που μένει ελεύθερος από τον επεξεργαστή; **(3 βαθμοί)**

   (α) Η αρτηρία μπορεί να μεταδίδει ένα burst 16 λέξεων (64 bytes) κάθε 5 κύκλους. Ο κάθε κύκλος έχει διάρκεια 10 ns. Άρα ο μέγιστος ρυθμός μετάδοσης δεδομένω που επιτρέπει η αρτηρία είναι:

   $$\text{max bus throughput} = \frac{64bytes}{50ns} = 1.28GB/s \quad (5)$$

   Θεωρώντας ότι η αρτηρία είναι απασχολημένη το 20% του χρόνου από τον επεξεργαστή ο μέγιστος ρυθμός μετάδοσης δεδομένων στο χρόνο που η αρτηρία είναι διαθέσιμη στη συσκευή DMA είναι:

   $$\text{max DMA throughput} = 0.8 \times \frac{64bytes}{50ns} = 1.02GB/s \quad (6)$$

   (β) Δεδομένου ότι κάθε κάρτα δικτύου έχει μέγιστο ρυθμό μετάδοσης δεδομένων 1.25 Gbytes/s δεν μπορούμε να συνδέσουμε καμία κάρτα δικτύου στην αρτηρία χωρίς να κινδυνεύουμε να χάνουμε δεδομένα.

2. Έστω ότι στο παραπάνω σύστημα επιτρέπουμε στις διευθύνσεις κύριας μνήμης προς/από τις οποίες γίνεται DMA να αποθηκεύονται σε αντίγραφα στην κρυφή μνήμη του επεξεργαστή και έστω ότι η παραπάνω κάρτα δικτύου γεμίζει (γράφει) με μία DMA έναν ενταμιευτή δεδομένων στην κύρια μνήμη. (α) Εάν ο επεξεργαστής δεν διαθέτει μηχανισμό που να παρακολουθεί τις διευθύνσεις προς/από τις οποίες γίνεται DMA πώς μπορεί να εξασφαλιστεί η συνέπεια των περιεχομένων της κρυφής μνήμης του επεξεργαστή με τα περιεχόμενα της κύριας μνήμης; Μπορεί ο επεξεργαστής να διαβάσει σωστά δεδομένα από τον ενταμιευτή κατά τη διάρκεια της DMA από τη κάρτα δικτύου, ναι, όχι, και γιατί; **(5 βαθμοί)** (β) Εάν ο επεξεργαστής διαθέτει μηχανισμό που να παρακολουθεί τις διευθύνσεις της κύριας μνήμης τις οποίες γράφει μία DMA πώς μπορεί να εξασφαλιστεί η συνέπεια των περιεχομένων της κρυφής μνήμης του επεξεργαστή με τα περιεχόμενα της κύριας μνήμης; Μπορεί ο επεξεργαστής να διαβάσει σωστά δεδομένα από τον ενταμιευτή κατά τη διάρκεια της DMA, ναι, όχι, και γιατί; **(5 βαθμοί)**

(α) Ο επεξεργαστής πρέπει να αδειάζει τα περιεχόμενα της κρυφής μνήμης (flush) πριν η συσκευή εκκινήσει τη λειτουργία DMA και δεν μπορεί να διαβάζει σωστά δεδομένα από τον ενταμιευτή κατά τη διάρκεια της λειτουργίας DMA εφόσον η συσκευή μπορεί να μεταβάλει περιεχόμενα διευθύνσεων μνήμης που έχει διαβάσει ήδη ο επεξεργαστής. Συγκεκριμένα, αν ο επεξεργαστής διατρέξει τον ενταμιευτή και επιχειρήσει να διαβάσει δεδομένα πριν η συσκευή γράψει τις νέες τιμές τους, ο επεξεργαστής θα έχει στην κρυφή μνήμη παλαιές (μη συνεπείς) τιμές δεδομένων. Κατά συνέπεια ο επεξεργαστής πρέπει να ειδοποιηθεί για να διαβάσει τα περιεχόμενα του ενταμιευτή αφού ολοκληρωθεί η λειτουργία DMA, εφόσον χρειάζεται τις νέες τιμές δεδομένων που μετέφερε η συσκευή. (β) Ο επεξεργαστής ακυρώνει αντίγραφα δεδομένων που βρίσκονται στην κρυφή μνήμη όταν δει (παρακολουθώντας τις συναλλαγές που γίνονται στην αρτηρία και τις διευθύνσεις τις οποίες αφορούν αυτές οι συναλλαγές) ότι μία συσκευή DMA γράφει στις διευθύνσεις των αντιγράφων στην κύρια μνήμη. Και σε αυτή την περίπτωση ο επεξεργαστής δεν μπορεί να διαβάσει δεδομένα από τον ενταμιευτή κατά τη διάρκεια της DMA. Αν ο επεξεργαστής προηγηθεί της συσκευής στην προσπέλαση διευθύνσεων του ενταμιευτή, τότε θα διαβάσει παλαιές τιμές δεδομένων. Οποιαδήποτε εγγραφή στα δεδομένα αυτά από τη συσκευή ακολουθήσει θα προκαλέσει ακύρωση των αντιγράφων των δεδομένων στην κρυφή μνήμη του επεξεργαστή, ωστόσο αν ο επεξεργαστής χρησιμοποιήσει τα παλαιά δεδομένα που διάβασε σε πράξεις μεταξύ της ανάγνωσης (από αυτόν) και της ακύρωσης (από τη συσκευή) θα παραβιαστεί η συνέπεια.