ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ και ΥΠΟΛΟΓΙΣΤΩΝ

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ, 5^{ο} εξάμηνο

ΦΕΒΡΟΥΑΡΙΟΣ 2006

ΘΕΜΑΤΑ ΚΑΙ ΛΥΣΕΙΣ ΚΑΝΟΝΙΚΗΣ ΕΞΕΤΑΣΗΣ

Θέμα 1^{ο} (10%):

Στο πρότυπο IEEE κινητής υποδιαστολής που αναφέρεται στο βιβλίο (σελ. 22), για την απλή και διπλή ακρίβεια παράστασης, ποια η ελάχιστη και ποια η μέγιστη διαφορά κατά την παράσταση διαδοχικών αριθμών. Δώστε το ελάχιστο σφάλμα παράστασης ενός οποιουδήποτε αριθμού.

Λύση 1^{ου} Θέματος

Απλή ακρίβεια

Η μέγιστη ακρίβεια ⇔ ελάχιστη διαφορά μεταξύ διαδοχικών αριθμών εντοπίζεται στην περίπτωση που ο εκθέτης είναι ελάχιστος. Ο ελάχιστος εκθέτης είναι −126, τόσο στην περίπτωση των κανονικών αριθμών (1−127 = −126), όσο και στην περίπτωση των υπο-κανονικών αριθμών. Τότε, η απόσταση μεταξύ των διαδοχικών αριθμών που παριστάνονται είναι

$$2^{-126}\,2^{-23} = 2^{-149}$$

Η ελάχιστη ακρίβεια ⇔ μέγιστη διαφορά μεταξύ διαδοχικών αριθμών εντοπίζεται στην περίπτωση που ο εκθέτης είναι μέγιστος. Ο μέγιστος εκθέτης για τους πραγματικούς αριθμούς είναι 127, στην περίπτωση των κανονικών αριθμών (254−127 = 127). Τότε, η απόσταση μεταξύ των διαδοχικών αριθμών που παριστάνονται είναι

$$2^{127}\,2^{-23} = 2^{104}$$

Το σφάλμα κατά την αναπαράσταση ενός αριθμού μπορεί να είναι έως και ίσο με το μισό της διαφοράς δύο διαδοχικών αριθμών. Άρα, το ελάχιστο σφάλμα αναπαράστασης εντοπίζεται στην περίπτωση που έχουμε την ελάχιστη διαφορά μεταξύ διαδοχικών αριθμών και ισούται με

$$2^{-150}$$

Διπλή ακρίβεια

Ομοίως με την περίπτωση απλής ακρίβειας, η μέγιστη ακρίβεια ⇔ ελάχιστη διαφορά μεταξύ διαδοχικών αριθμών εντοπίζεται στην περίπτωση που ο εκθέτης είναι ελάχιστος. Ο ελάχιστος εκθέτης είναι −1022, τόσο στην περίπτωση των κανονικών αριθμών (1−1023 = −1022), όσο και στην περίπτωση των υπο-κανονικών αριθμών. Τότε, η απόσταση μεταξύ των διαδοχικών αριθμών που παριστάνονται είναι

$$2^{-1022}\,2^{-52} = 2^{-1074}$$

Η ελάχιστη ακρίβεια ⇔ μέγιστη διαφορά μεταξύ διαδοχικών αριθμών εντοπίζεται στην περίπτωση που ο εκθέτης είναι μέγιστος. Ο μέγιστος εκθέτης για τους πραγματικούς αριθμούς είναι 1026, στην περίπτωση των κανονικών αριθμών (2046−1023 = 1023). Τότε, η απόσταση μεταξύ των διαδοχικών αριθμών που παριστάνονται είναι

$$2^{1023}\,2^{-52} = 2^{971}$$

Το σφάλμα κατά την αναπαράσταση ενός αριθμού μπορεί να είναι έως και ίσο με το μισό της διαφοράς δύο διαδοχικών αριθμών. Άρα, το ελάχιστο σφάλμα αναπαράστασης εντοπίζεται στην περίπτωση που έχουμε την ελάχιστη διαφορά μεταξύ διαδοχικών αριθμών και ισούται με

$$2^{-1075}$$

Θέμα 2^{ο} (30%):

α) Δείξτε τις απαραίτητες τροποποιήσεις στην μονάδα ελέγχου του TRN που έχει υλοποιηθεί με καλωδιωμένη λογική, ώστε να υποστηρίζεται μια γενική εντολή άλματος **JPC address_offset** η οποία κάνει άλμα σχετικό με το PC:

JPC address_offset; PC← PC+ address_offset

Η JPC μπορεί να λειτουργεί ως σχετικό άλμα χωρίς ή υπό συνθήκη, με τη βοήθεια του καταχωρητή δείκτηI. Αν:

- I=10, τότε το άλμα γίνεται υπό συνθήκη A>0.

- I=01, τότε το άλμα γίνεται υπό συνθήκη A<0.
- I=11, τότε το άλμα γίνεται χωρίς συνθήκη.
- I=00, τότε το άλμα γίνεται υπό συνθήκη A=0.

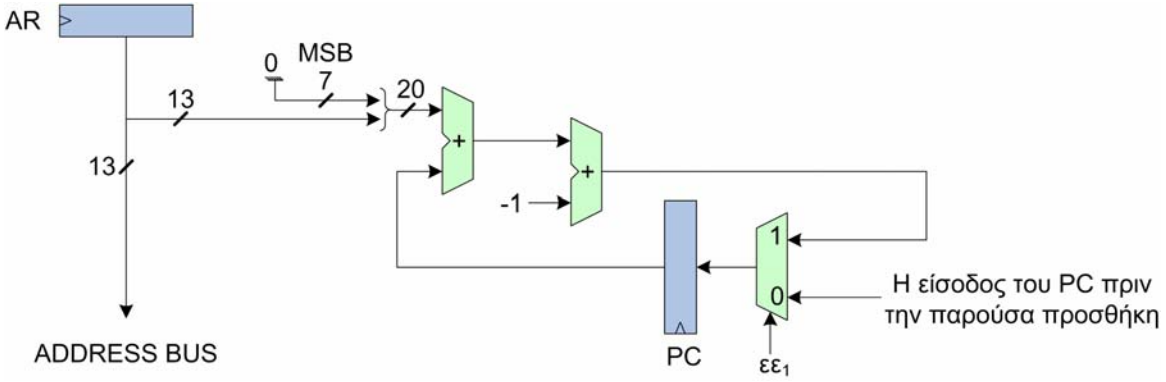
Δώστε την ακολουθία των σημάτων ελέγχου για την JPC. Βάλτε την JPC στη θέση της NOP. Μπορεί η JPC να αντικαταστήσει τη λειτουργία της NOP;

β) Δώστε τις αλλαγές που πρέπει να γίνουν για να υλοποιείται και έμμεση αναφορά, δηλαδή:

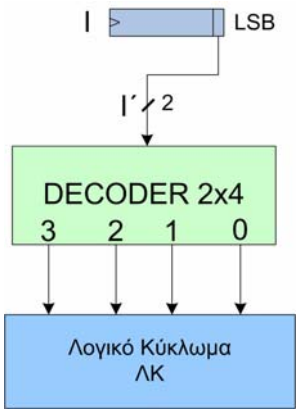
JPC (Address); PC← PC+ [0:12] M[address], όπου [0:12] τα bits 0-12

Λύση 2ου Θέματος

α) Προκειμένου να υποστηριχθεί η εντολή JPC από τον TRN, στη μονάδα επεξεργασίας του θα πρέπει να προστεθεί η δυνατότητα να προστίθενται τα περιεχόμενα των καταχωρητών AR και PC και η δυνατότητα αποθήκευσης του αθροίσματος αυτού στον PC. Αν δεν λάβουμε υπόψη τις περιπτώσεις δεικτοδοτημένης και έμμεσης αναφοράς, ο καταχωρητής AR περιέχει το address_offset. Επίσης, επειδή στη φάση ανάκλησης της εντολής πραγματοποιείται αύξηση του PC κατά 1 (PC ← PC + 1), θα πρέπει να μειώσουμε κατά ένα το άθροισμα του AR με τον PC, πριν αυτό δοθεί ως είσοδος στον PC, προκειμένου να επιτευχθεί η ζητούμενη λειτουργικότητα PC ← PC + address_offset. Οι προσθήκες στη μονάδα επεξεργασίας του TRN φαίνονται στο παρακάτω σχήμα.



Το σήμα $εε_1$ (τυχαία ονομασία) καθορίζει αν η επόμενη προς εκτέλεση εντολή ανακληθεί από τη διεύθυνση PC+1 (όταν $εε_1$ == 0) ή από τη διεύθυνση PC+address_offset (όταν $εε_1$ == 1). Η μονάδα ελέγχου του TRN είναι υπεύθυνη το καθορισμό της τιμής του σήματος $εε_1$, σύμφωνα με τις τιμές που μπορεί να πάρουν τα 2 λιγότερο σημαντικά bits του καταχωρητή I και ο καταχωρητής A. Μιας και τα κυκλώματα που ανιχνεύουν τις καταστάσεις A>0, A<0 και A==0 υπάρχουν ήδη στο Λογικό Κύκλωμα της μονάδας ελέγχου του TRN, στο σχήμα 6.4 του βιβλίου θα πρέπει να προστεθεί η δυνατότητα ανίχνευσης των καταστάσεων I==00, I==01, I==10 και I==11. Αυτό επιτυγχάνεται με την παρακάτω προσθήκη.



Χρησιμοποιώντας τις παραπάνω δυνατότητες, η ακολουθία των σημάτων ελέγχου είναι (φαίνεται μόνο η φάση εκτέλεσης, αφού όλες οι άλλες φάσεις παραμένουν οι ίδιες):

JPC $i_0 f_3 t_0$((I´==00)Z + (I´==01)(A<0) + (I´==10)(A>0) + (I´==11)):
PC ← PC + address_offset (δηλ. $εε_1$ = 1); F1←0; F2←0; CSC;

Όπως φαίνεται από το σήμα i_0 η εντολή έχει μπει στη θέση της NOP. Η εντολή JPC μπορεί να αντικαταστήσει την εντολή NOP αν γραφτεί ως JPC 1, αφού είτε γίνει είτε όχι το σχετικό άλμα, η επόμενη εντολή που θα εκτελεστεί είναι αυτή που βρίσκεται στο PC+1.

β) Για την υλοποίηση της έμμεσης αναφοράς δεν απαιτείται καμία τροποποίηση από το (α), αφού η φάση έμμεσης αναφοράς του TRN θα τοποθετήσει στον καταχωρητή AR τα περιεχόμενα της διεύθυνσης address_offset.

Θέμα 3ο (30%):

Θέλουμε να υλοποιήσουμε στον TRN, με τη βοήθεια μικροπρογραμματιζόμενης λογικής, μια νέα εντολή **LDA_IFAP Address_1, Address_2** (load A, if A positive), που λειτουργεί ως εξής:

LDA_IFAP: A ← (A>0 ? M[Address_1] : M[Address_2]),

δηλαδή φορτώνει στον A το περιεχόμενο της διεύθυνσης Address_1 αν το A>0, αλλιώς φορτώνει στον A το περιεχόμενο της διεύθυνσης Address_2.

Γράψτε το αντίστοιχο μικροπρόγραμμα που υλοποιεί την LDA_IFAP και ενσωματώστε το στην μικρομνήμη του TRN (παράγραφο 7.7 βιβλίου). Βγάλτε όσο το δυνατό λιγότερες εντολές από τη μικρομνήμη για να χωρέσει η υλοποίηση της LDA_IFAP. Επειδή η LDA_IFAP έχει 2 ορίσματα, η αποθήκευσή της γίνεται σε 2 διαδοχικές θέσεις μνήμης. Κάντε τις αλλαγές στην μικροπρογραμματιζόμενη μονάδα ελέγχου ώστε να εκτελείται σωστά η LDA_IFAP.

Λύση 3ου Θέματος

Θεωρούμε ότι η εντολή LDA_IFAP καταλαμβάνει 2 διαδοχικές θέσεις μνήμης με αυτή τη μορφή:

LDA_IFAP	0	0	Address_1
			Address_2

Θεωρούμε επίσης ότι δεν υπάρχει η περίπτωση δεικτοδοτημένης και έμμεσης αναφοράς, έχοντας στα αντίστοιχα πεδία την τιμή 0 (δηλ. E=0, D=0). Το περιεχόμενο των κενών τμημάτων είναι αδιάφορο.

Θα προσπαθήσουμε να αντικαταστήσουμε μόνο **μία** εντολή από τη μικρομνήμη και για τον επιπλέον χώρο που θα χρειαστούμε θα αξιοποιήσουμε τμήματα της μικρομνήμης που είναι αχρησιμοποίητα (π.χ. διεύθυνση 33-35).

Έστω ότι αντικαθιστούμε την εντολή INP (opcode = 29, ORG = 116). Ο αντίστοιχος κώδικας που υλοποιεί την LDA_IFAP είναι:

```
            ORG 116
LDA_IFAP:   JMP (Z, L2)             * Έλεγχος αν A=0
            JMP (S, L2)             * Έλεγχος αν A<0

L1:         JMP (U, LDA), INCPC     * Εδώ καταλήγουμε αν A>=0
                                    * Στην περίπτωση αυτή εκτελείται η ίδια
                                    * ακολουθία εντολών με την LDA, αφού πρώτα
                                    * αυξήσουμε τον PC κατά 1, ώστε να μην θεωρηθεί
                                    * ως ξεχωριστή εντολή η δεύτερη λέξη που
                                    * συνθέτει την εντολή LDA_IFAP.
            ORG 33
L2:         JMP (U, CONT), TPCAR    * Εδώ καταλήγουμε αν A<=0
            JMP (U, CONT), READ, INCPC * Αφού αυξήσουμε κατά 1 τον PC,
            JMP (U, LDA), TBRAR     * μεταφέρουμε τη διεύθυνση Address_2 στον AR,
                                    * ώστε κατά την εκτέλεση της LDA που ακολουθεί,
                                    * να μεταφερθεί στον A το περιεχόμενο της
                                    * Address_2
```

Θέμα 4 (30%):

Έστω ότι η κρυφή μνήμη (cache memory) ενός υπολογιστή είναι υλοποιημένη με την μέθοδο απευθείας οργάνωσης (direct mapped organization) και αποτελείται από 1024 γραμμές (cache lines ή memory blocks) των 4 λέξεων (words) η κάθε μία. Η κάθε λέξη έχει μήκος 32 bit. Το μήκος διεύθυνσης του υπολογιστή μας είναι 32 bit. Επιπλέον, σε κάθε γραμμή υπάρχει και ένα Dirty/Valid bit. Επιθυμούμε να έχουμε ξεχωριστή διευθυνσιοδότηση για κάθε ένα από τα 4 bytes που αποτελούν τη λέξη.

α) Δώστε το σχηματικό διάγραμμα της κρυφής μνήμης. Πόσο μέγεθος σε bytes έχει συνολικά; Πόσες λέξεις χωράει;
β) Διαθέτετε ψηφίδες SRAM μεγέθους 512 x 8 bit. Σχεδιάστε το σύστημα κρυφής μνήμης για τον υπολογιστή αυτό. Πόσες ψηφίδες θα χρειαστείτε? Δείξτε πως ενεργοποιείται η κάθε ψηφίδα, με βάση τη διεύθυνση στην κρυφή μνήμη.
γ) Αν θέλατε να αλλάξετε την οργάνωση της παραπάνω κρυφής μνήμης, έχοντας δεδομένο το συνολικό μέγεθος της σε bytes που βρήκατε στο Α ερώτημα, ώστε πλέον να είναι υλοποιημένη με οργάνωση συνόλου-συσχέτισης 2 δρόμων (2–way set associative), όπου επίσης η κάθε γραμμή αποτελείται από 4 λέξεις των 32 bit, πόσες γραμμές θα έχει πλέον η νέα σας κρυφή μνήμη; Πόσες λέξεις χωράει πλέον, περισσότερες ή λιγότερες από πριν?Δώστε το σχηματικό διάγραμμα.

Λύση 4ου Θέματος

α) Το σχηματικό διάγραμμα της κρυφής μνήμης φαίνεται παρακάτω:
Κάθε cache line περιέχει εκτός από τις 4 λέξεις δεδομένων, επιπλέον το τμήμα tag της διεύθυνσης (για την αναγνώριση των πολλών διαφορετικών γραμμών της κύριας μνήμης που μπορούν να αντιστοιχισθούν σε κάθε cache line της κρυφής μνήμης). Τέλος περιέχει και 1 bit για την αποθήκευση της ένδειξης Dirty/Valid.

	Dirty/ Valid	Tag	Λέξη 0	Λέξη 1	Λέξη 2	Λέξη 3
Cache line 0						
Cache line 1						
Cache line 2						
Cache line 3						
…						
…						
…						
…						
Cache line 1022						
Cache line 1023						
	1	18	32	32	32	32

Το μήκος διεύθυνσης του υπολογιστή μας είναι 32 bit.
Από αυτά 2 bits αφιερώνονται στη διευθυνσιοδότηση των 4 bytes ($=2^2$ bytes) που αποτελούν κάθε λέξη.
2 bits αφιερώνονται στη διευθυνσιοδότηση των 4 λέξεων ($=2^2$ λέξεις) που αποτελούν κάθε cache line.
10 bits αφιερώνονται στη διευθυνσιοδότηση των 1024 cache lines ($=2^{10}$ cache lines) της κρυφής μνήμης (index)

Τα υπόλοιπα (32 – 2 – 2 – 10) = 18 bits είναι το τμήμα tag της διεύθυνσης.

Επομένως για την αποθήκευση 1 cache line χρειαζόμαστε:
(32 × 4) = 128 bits για τα δεδομένα των 4 λέξεων της cache line.
18 bits για το tag της διεύθυνσης
1 bit Dirty/Valid.
Συνολικά για μία cache line: 128 + 18 + 1 = 147 bits.
Συνολικά για την κρυφή μνήμη: 147 × 1024 bits = 150.528bits = 18.816 bytes.

Οι λέξεις (δεδομένων) που χωράει η κρυφή μνήμη είναι:
4 λέξεις/cache line × 1024 cache lines = 4.096 λέξεις δεδομένων.

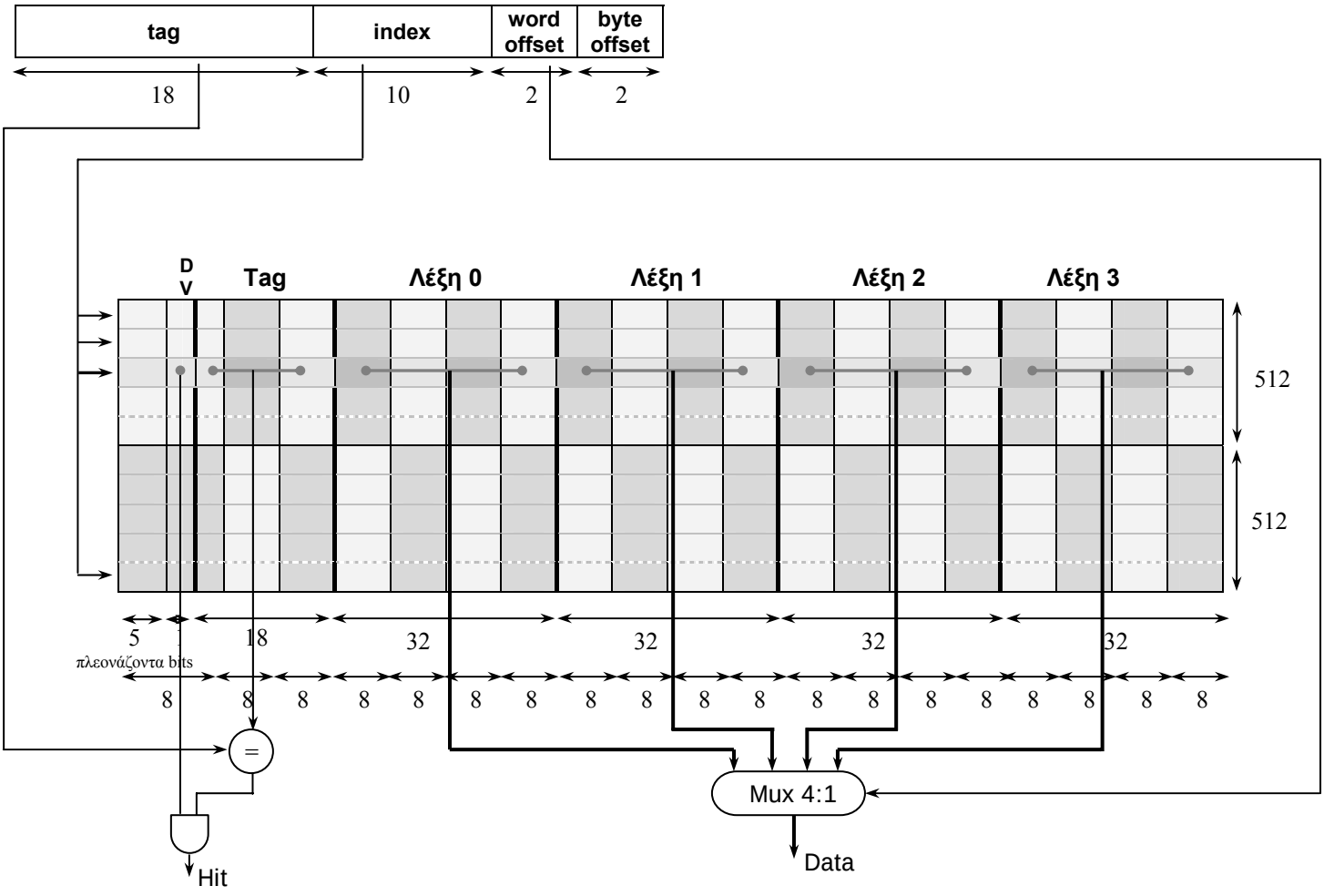
β) Για να σχηματισθεί η cache line κατά μήκος, χρειάζονται: $\left\lceil \frac{147}{8} \right\rceil = 19$ ψηφίδες SRAM.

Και στο ύψος: $\frac{1024}{512} = 2$ ψηφίδες SRAM.

Συνολικά: 2 × 19 = 38 ψηφίδες SRAM

Η διάταξη είναι η εξής:

Στο παρακάτω διάγραμμα, η κάθε ψηφίδα SRAM χρωματίζεται διαφορετικά από τις γειτονικές της, προκειμένου να φαίνονται τα όρια μεταξύ τους. Επίσης φαίνεται η ενεργοποίηση των ψηφίδων, ανάλογα με το περιεχόμενο της διεύθυνσης.



γ) Για σταθερό συνολικό μέγεθος κρυφής μνήμης, όταν αυξηθεί ο βαθμός συσχέτισης (associativity) μειώνεται ο αριθμός των sets που την αποτελούν. Κατά συνέπεια, μειώνεται το μέγεθος του index και αυξάνεται το μέγεθος tag. Άρα, για την αποθήκευση 1 cache line δεδομένων, χρειάζονται πλέον περισσότερα bits. Αυτό σημαίνει ότι θα πρέπει να μειωθεί ο συνολικός αριθμός cache lines της κρυφής μνήμης, και επομένως ο αριθμός των λέξεων που μπορούν να αποθηκευτούν σε αυτήν.

Στην περίπτωση που η οργάνωση της μνήμης είναι 2-way set associative, η κρυφή μνήμη περιέχει το πολύ:

$$\frac{1024}{2} = 512 \text{ sets}$$

Άρα το πολύ 9 bits αφιερώνονται στη διευθυνσιοδότηση των 512 (=2^9) sets από cache lines της κρυφής μνήμης. (Αν ο αριθμός των sets μειωθεί σε λιγότερα από 256, θα χρειάζονται 8 ή λιγότερα bits από τη διεύθυνση για το τμήμα index.)

Έστω ότι ο αριθμός των sets της κρυφής μνήμης ανήκει στο διάστημα (512,1024), όπου χρειάζονται 9 bits για το τμήμα index.
Τα υπόλοιπα (32 – 2 – 2 – 9) = 19 bits είναι το τμήμα tag της διεύθυνσης.

Ο ακριβής αριθμός των cache lines της νέας κρυφής μνήμης θα είναι (A):

$$[(32 \times 4) + 19 + 1] \times A \approx 150.528 \text{ bits}$$

$$A = \left\lfloor \frac{150.528}{148} \right\rfloor = 1017$$

Προκύπτει, δηλαδή, ότι πράγματι A > 512, και επομένως σωστά έγινε η υπόθεση ότι για το index χρειάζονται 9 bits.
Επειδή, ωστόσο μία πραγματική 2-way set associative κρυφή μνήμη περιέχει ακέραιο αριθμό από sets, και επομένως άρτιο αριθμό cache lines:
A = 1016.
Τελικά [(32 × 4) + 19 + 1] × 1016 = 150.368 bits < 150.528bits

Άρα η νέα κρυφή μνήμη περιέχει 1016 cache lines και 1016 × 4 = 4.064 λέξεις (< 4.096)