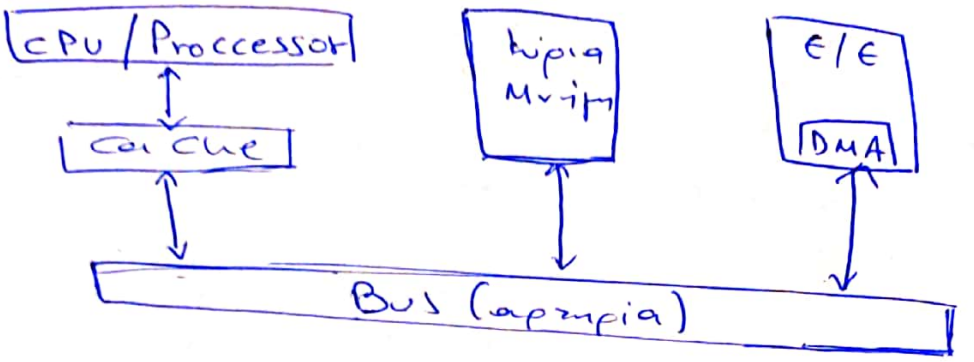
Εργαστήριο 14 :

κων/νος Δεστοράκης
csd3716

14.1 :



α) Αν υπάρχουν δεδομένα στην κρυφή μνήμη θα διαβάσει τα παλιά ενώ αν είναι στην κύρια θα διαβάσει τα νέα τα οποία δεδομένα.

β) Στην περίπτωση που έχουμε write-through δεν γίνεται να υπάρχουν λανθασμένα δεδομένα καθώς αυτά ήδη προϋπάρχουν από τα πάνω.

γ) Στην περίπτωση που έχουμε write-back όταν αυτεί που ζητάει ο επεξεργαστής δεν υπάρχουν στην κρυφή μνήμη τότε παίρνει στην κύρια μνήμη και γράφει τα καινούρια στην περιφερ. συσκευή μέσω του DMA Αν τα δεδομένα υπάρχουν ο επεξεργαστής τα γράφει στην cache αρα θα έχω και τα νεότερα επειδή είναι στην κύρια μνήμη. Αν υπάρχουν στην κρυφή επειδή η μνήμη είναι write-back, τότε το DMA θα μεταφέρει παλιές τιμές στην περιφερειακή συσκευή.

14.3

α) Στο i) δεν γίνεται λάθη στο ii) ο επεξεργαστής διαβάζει την καινόχρωμη μεταβλητή που βρίσκεται στην κρυφή και δεν την έχει και την δίνει στην κύρια μνήμη iii) Ο επεξεργαστής ζητάει από την καινόχρωμη μνήμη από την κρυφή μνήμη, δεν την έχει και γι' αυτό την δίνει η κύρια iv) Ο Α τώρα βρίσκει στην κρυφή μνήμη το S και έχει από Ο→1 και επειδή η κρυφή μνήμη είναι write-back δεν γίνεται λάθη στην κύρια αν ήταν write-through θα γινόταν. v) Ο Β θα ζητήσει την καινή μεταβλητή θα την βρει στην κρυφή του μνήμη και επειδή δεν υπάρχει πρότυπο συνοχής θα διαβάσει το Ο και όχι τα νέα δεδομένα που έχει ~~έχει~~ το Α.

β) Ο Α θα ζητήσει την καινόχρωμη μεταβλητή, δεν υπάρχει στην κρυφή μνήμη άρα ζητείται από την κύρια μνήμη, θα έρθει το block μέσω της αρτηρίας και θα μπει στην κρυφή μνήμη ως shared. Ο Β ζητάει την μεταβλητή αυτή δεν υπάρχει στην αρτηρία και την ζητάει από το Α και αυτό μπαίνει σαν shared σε αυτήν. Το Α αλλάζει την κρυφή του στην κύρια μνήμη άρα γίνεται modified καθώς αλλάζει και στέλνεται ένα invalidate μήνυμα στην κύρια μνήμη και απορρίπτει σβήνει στην δική του (το Β), το Β μετά προσπαθεί να διαβάσει τα στοιχεία που είχαν σβηστεί αλλά υπάρχει ασυχία και επειδή του την δίνει ο Α, ο Α την γράφει στην κύρια μνήμη και την έχουν και οι δύο σε κατ. shared άρα θα γίνουν οι νέοι υπολ. με σειρά Α επί πριν ήτα με σειρά Ο.

γ) Στο 2° βήμα αυτό το block όταν έχουμε MESI το προϊόν είναι exclusive στο βήμα 2 ενώ στο MSI είναι shared και στο 4° βήμα επειδή έχουμε από exclusive → modified στο MESI ενώ στο MSI shared → modified ενώ στο 5° δεν έχουμε.

14.4:

α)
```
lw    t4, 120(x0)
addi  t4, t4, 1
sw    t4, 128(x0)
lw    t4, 124(x0)
addi  t4, t4, 2
sw    t4, 132(x0)
```

β) Επειδή χρησιμοποιούνται την lw χάνει 2 κύκλους από τις εξαρτημένες load και add.

```
lw    t4, 120(x0)
lw    t5, 124(x0)
addi  t4, t4, 1
sw    t4, 128(x0)
addi  t5, t5, 2
sw    t5, 132(x0)
```

Με την αναδιάταξη προστάθ. την χάνουμε κύκλους και εξαοτανίζεται με αυτή την αλλαγή.

γ)
```
lw    t1, 0(x14)
addi  t1, t1, 1
sw    t1, 0(x16)
lw    t2, 0(x15)
addi  t2, t2, 2
sw    t2, 0(x17)
```

- Δεν μπορεί γιατί δεν ξέρει αν είναι ίδια ή διαφορετικά (το px με το pb)

- Στην μια περ. υποδομή το (*pa)+1 και το (*pb)+2 ενώ στην άλλη υποδομή το (*pa)+3 αν αυτά είναι ίδια.

- Όχι όπως είπαμε και προηγουμένως.

β) Στην περίπτωση του 1% που θα είναι ίδια η load πρέπει να περιμένει εώς ότου το 99% των περιπτώσεων θα πρέπει να έχει ανιχνεύσει εάν εξαρτάται η επόμενη load από την προηγ. store και με την προωθ. οι pointers pb και px να έχουν την ίδια τιμή. Το hardware ελέγχει τις τιμές των δεικτών και ξέρουν αν είχαν ίδια τιμή ή όχι, θα χάνονται 1% χρόνος και 99% δεν θα χάνεται χρόνος.

14.5

α) Αν ο επεξεργ. ήταν ίσως και δεν είχε multithread και είναι out of order θα χαίρει κύκλους ρολογιού δηλαδή 8.

β) Το νήμα πρόγραμμα A θα τρέξει σε περ. χρόνο και όχι σε διπλάσιο. και θα κάνει 80 (άρα συνολικά δεν χάνει κάτι).

γ) Συνολικά δεν θα χαθεί κανένας κύκλος και αφού στο Β θα χαθεί για τους 80 κύκλους.