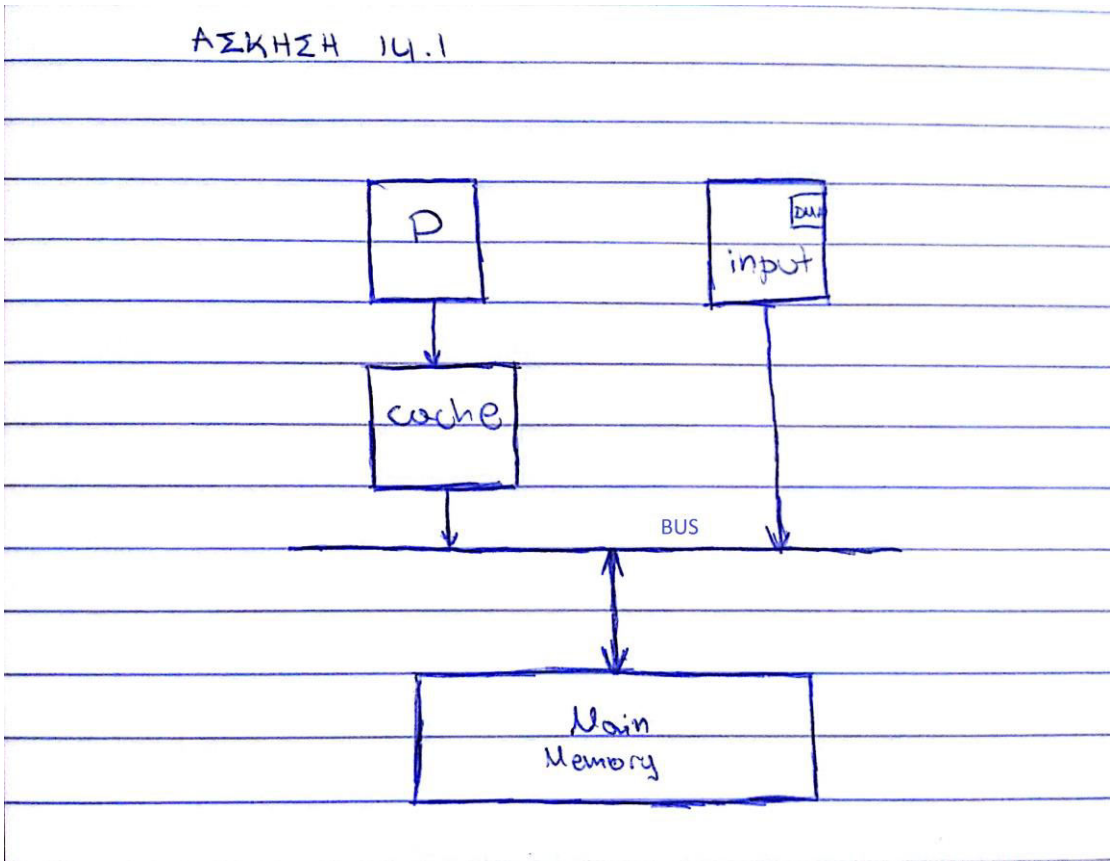
ΜΑΡΑΝΤΟΣ ΑΛΕΞΑΝΔΡΟΣ 3329

14Η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ ΗΥ-225

ΑΣΚΗΣΗ 14.1



Α)

i)

Αν διαβαζουμε δεδομενα από την εισοδο τοτε θα ψαξει να τα βρει στην κρυφη μνημη. Επειδη όμως εδώ το γραψιμο γινεται αμεσως, με DMA στην μνημη, μπορει τα δεδομενα να μην είναι up-to-date αφου δεν τα γραφει στην κρυφη μνημη

ii)

Αν τα δεδομενα από την εισοδο εχουν ορισθει ως non cacheable τοτε αφου τα δεδομενα δεν θα βρισκονται στην κρυφη μνημη θα ανατρεξει στην main memory

Β)

Εφοσον χρησιμοποιειται το πρωτοκολλο ονοματι «write through» οτιδηποτε γραφεται στην cache θα γραφεται αμεσως και στην main memory ώστε να μην χανονται δεδομενα

Γ)

Εφοσον το συστημα μας χρησιμοποιει wirite back, η αντιγραφη δεδομενων στην εξωτερικη συσκευη θα είναι ορθη αφου πρωτα εχουν αντιγραφει πληρως τα περιεχομενα της κρυφης μνημης στη main memory.

Αν προσπαθησουμε να αντιγραψουμε δεδομενα πριν την απαραιτητη επανεγγραφη των νεων δεδομενων στην κυρια μνημη, θα γεμισουμε σκουπιδια.

ΑΣΚΗΣΗ 14.3

Α)

i) sharedVar = 0 (κρυφη μνημη μονο)

ii) Ο P1 διαβαζει sharedVar . Το BUS καλει την κυρια μνημη να την φερει καθως δεν υπαρχει στην κρυφη μνημη . sharedVar1

iii) Ο P2 διαβαζει sharedVar. Το BUS καλει την κυρια μνημη να την φερει καθως δεν υπαρχει στην κρυφη μνημη . sharedVar2

iv) sharedVar1 = 11 .

v) Ο P2 διαβαζει την sharedVar2. Υπαρχει στην κρυφη του μνημη που εχει τιμη 0 (παλια)

Αν στο συστημα υπηρχε write through όπως οι μεταβλητες που αλλαζουν στην cache θα αλλαζαν και στην main χωρις να πειραζονται όμως οι άλλες caches

Β)

Αυτή τη φορα υπαρχει επικοινωνια μεταξυ P1 και P2

i) sharedVar = 0 (κρυφη μνημη μονο)

ii) Ο P1 διαβαζει sharedVar . Το BUS καλει την κυρια μνημη να την φερει καθως δεν υπαρχει στην κρυφη μνημη . sharedVar1

iii) Ο P2 διαβαζει sharedVar. Το BUS καλει την κυρια μνημη να την φερει καθως δεν υπαρχει στην κρυφη μνημη . sharedVar2

iv) sharedVar1 = 11 . Μεσω του bus γινεται ενημερωση ότι αλλαξε η τιμη και ενημερωνονται και οι υπολοιπες caches

v) Ο P2 διαβαζει την sharedVar2. Υπαρχει στην κρυφη του μνημη που εχει τιμη 11

Γ)

Όταν μεταβαλλεται η μεταβλητη x στην cache

a) Στο MSI η κρυφη μνημη ετσι κανει invalidate τις υπολοιπες κρυφες μνημες σχετικα με αυτή τη μεταβλητη ανεξαρτητα από το αν την εχουν η όχι.
b) Στην MESI το invalidate δεν χρειαζεται στα υπολοιπα αφου υπαρχει το tag exclusive που λεει πως μονο αυτή η κρυφη μνημη εχει την πληροφορια αυτή.

ΑΣΚΗΣΗ 14.4

Α)

x = a+2;

y = b-1;

το a ειναι στη θέση 120

το b ειναι στη θέση 124

το x ειναι στη θέση 128

το y ειναι στη θέση 132

Κωδικας:

```
la $t0 ,120($0) //t0 =a
addi $t0 , $t0, 2
addi $t1 ,$t1 ,-1
la $t1, 124($0) // t1=b
sw $t2, 128($t0) //t2=x
sw $t3, 132($t0) //t3=y
```

Β)

Χανουμε 2 κυκλους ρολογιου, ο ενας για κάθε load αφου για κάθε addi πρεπει να εχει ηδη υπολογιστειη διευθυνση της μεταβλητης, πραξη που χρειαζεται έναν κυκλο.Αν γραφουν και οι 2 load μαζι, στον ιδιο κυκλο θα εκτελεστουν και οι 2 εντολες μαζι

Κωδικας:

```
lw $t0 ,120($0) //t0 =a
lw $t1, 124($0) // t1=b
addi $t0 , $t0, 2
addi $t1 ,$t1 ,-1
sw $t2, 128($t0) //t2=x
sw $t3, 132($t0) //t3=y
```

Γ)

(*px) = (*pa)+2;

(*py) = (*pb)-1;

Κωδικας

la $t0, 0($12);//t0=*pa

addi $t0 , $t0 , 2 //*pa=*pa +2

sw $14, 0($t0) //*px=*pa+2

la $t0 , 0($13) //t0=*pb

addi $t0, $t0 -1 //*pb=*pb-1

sw $15,0($t0) //*py=*pb-1

Για να καθορισει αν θα κανουμε instruction scheduling πρεπει να ξερουμε κατά ποσο οι μεταβλητες px pb είναι ιδιες ή διαφορετικες. Αν δεν ξερουμε όλα αυτά δεν μπορουμε να κανουμε την παραπανω διαδικασια

Αν οι τιμες τους δεν είναι ιδιες τοτε όπως και στο β μπορουμε να κανουμε τις 2 load μαζι για οικονομια κυκλων ρολογιου

Ενώ αν είναι ιδιες τοτε χρειαζομαστε έναν κυκλο παραπανω μεν αλλα δεν χρειαζεται το επιπλεον load.

Δ)

Η $2^{η}$ load ειανι εξαρτημενη από άλλες προηγουμενες εντολες αν οι μεταβλητες px pb είναι ιδιες. Οποτε τωρα δεν γινεται να εκτελεστει η εντολη αυτή αν πρωτα δεν εχει υπολογιστει η $1^{η}$ μεταβλητη. Ενω στην περιπτωση που είναι διαφορετικες δεν θα είναι εξαρτημενη

ΑΣΚΗΣΗ 14.5

Α) Αν δεν ειχαμε Multithreading οι εντολες που θα χανονταν θα ηταν 80-8 = 72