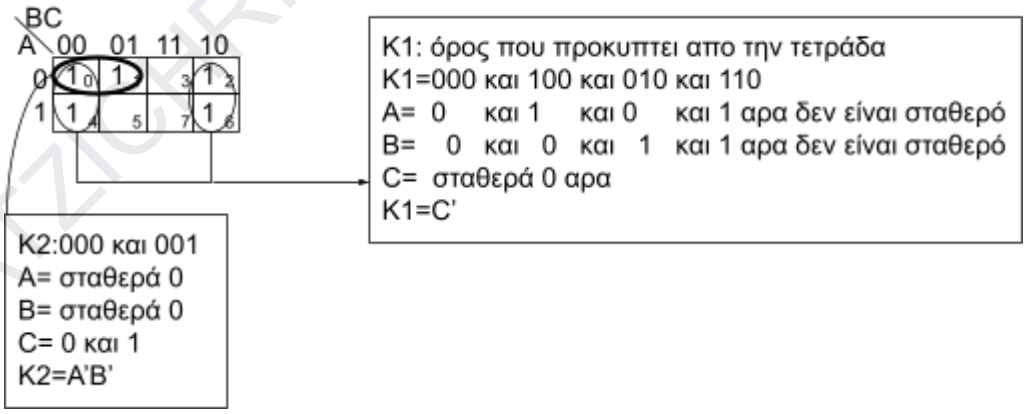
Ασκήσεις Αρχιτεκτονικής Εξηγήσεις

Στο παρακάτω έγγραφο θα δείτε τον τρόπο επίλυσής βήμα-βήμα διαφορών ασκήσεων που μπορούν να πέσουν στις εξεταστικές .

- ## Απλοποιήσεις με χάρτες Karnaugh (Εισαγ. Διάλεξη)

1) Βάσει του πλήθος μεταβλητών επιλέγουμε τον κατάλληλο χάρτη
2) Εκφράζουμε την F ως άθροισμα ελαχιστόρων . Δηλαδη Σ(1,2,3,..) και τοποθετούμε τις μονάδες στις θέσεις του χάρτη που αντιστοιχούν σε αυτούς τους ελαχιστόρους
3) Δημιουργούμε ομάδες από άσσους οι οποίες
   a) το πλήθος των ασσων σε κάθε ομάδα είναι δύναμη του 2 (1,2,4,8,16,32)
   b) κάθε ομάδα πρέπει να περιέχει ΜΕΓΙΣΤΟ πλήθος άσσων δηλαδή αν εχω την δυνατότητα να πάρω μια τετράδα δεν θα πάρω δυο δυάδες
   c) πρέπει όλοι οι ασσοι να βρεθούν σε τουλάχιστον μια ομάδα
   d) μπορούμε έναν άσσο να το συμπεριλάβουμε σε περισσότερες από 1 ομάδες
4) Κάθε ομάδα δημιουργεί εναν *ΑΠΛΟΠΟΙΗΜΈΝΟ* όρο γινομένου, Αν έχουμε π.χ. 2 ομάδες θα έχουμε 2 ορους γινομένου, οι οποιοι θα αθροιστούν. Δηλαδή Ν ομαδες → Ν όρους γινομένου οπου για π.χ:



5) Αθροίζουμε τα γινομενα απο το d

F1=K1+K2

F1=C'+A'B'

- **Pipeline 1/2 (Διάλεξη 14)**

Έστω η πραξή Ai x Bi + Ci, i=1….t

1)Φορτώνουμε τα Ai,Bi

δηλαδή : R1 ← Ai , R2 ← Bi

2)Πολλαπλασιαζουμε Ai * Bi και φέρνουμε απο την μνήμη το Ci

δηλαδή : R3 ← Ai*Bi (R3 ← R1*R2)

R4 ← Ci

3)Προσθέτουμε το γινόμενο του βήματα 2(R3) με το Ci (R4)

R5 ← R3 + R4

4) Στην συνέχεια εχουμε 3 Segments

2 καταχωρητές R1,R2	-2 καταχωρητές R3,R4 -1 κυκλωμα πολλαπλασιαστή	-1 καταχωρητη RS -1 αθροιστής

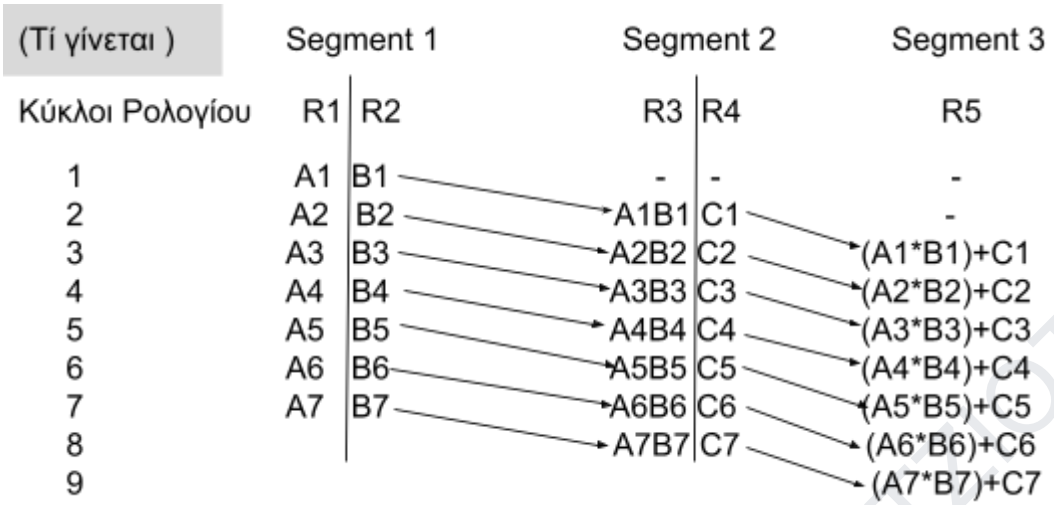
5) Εδω ουσιαστικα βλεπουμε τα γινόμενα να πηγαίνουν απο Segment σε Segment με την Μορφή που αναφέρεται στο **3)**

με το

- R1 ← Ai , R2 ← Bi , ουσιαστικά εδω τα βάζουμε απλα στο Pipeline
- R3 ← Ai*Bi (R3 ← R1*R2) ,R4 ← Ci , εδώ έχει γίνει η πρώτη πράξη και περνάνε στο Segment 2
- R5 ← R3 + R4 , τελος μπαινει στο Segment 3 με την τελευταια πράξη που δείξαμε στο **4)**

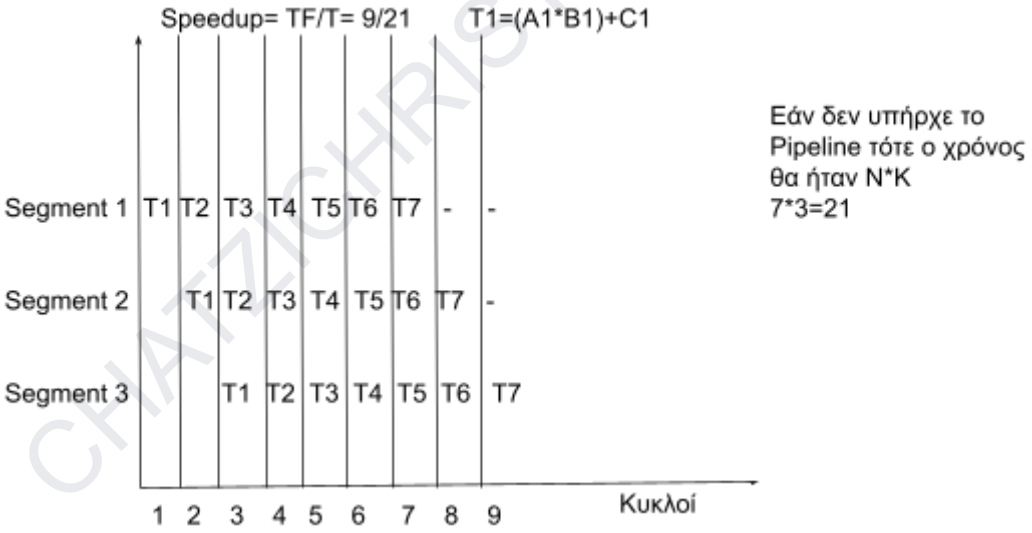
	Segment 1		Segment 2		Segment 3
Κύκλοι Ρολογίου	R1	R2	R3	R4	R5
1	A1	B1	-	-	-
2	A2	B2	A1B1	C1	-
3	A3	B3	A2B2	C2	(A1*B1)+C1
4	A4	B4	A3B3	C3	(A2*B2)+C2
5	A5	B5	A4B4	C4	(A3*B3)+C3
6	A6	B6	A5B5	C5	(A4*B4)+C4
7	A7	B7	A6B6	C6	(A5*B5)+C5
8			A7B7	C7	(A6*B6)+C6
9					(A7*B7)+C7

- **Pipeline 2/2**



6) Εαν ζητηθει και ο χρόνος :
N=εργασίες=7 επι K=στάδια=3 είναι ο χρονος που απαιτείται χωρίς την ύπαρξή Pipeline (εδώ T=7*3=21)

Χρόνος μετα την χρήση Pipeline : N+K-1=TF 7+3-1=9
Το Speedup θα είναι TF/T=9/21=0.42 η 42%



Οπού:
Ανάγνωση των A1,B1 από την T1 δηλαδη T1= (A1*B1) + C1
μνήμη R1,R2 στο R3 δηλαδή R3←R1R2

Αποκωδικοποιητές (Διάλεξη 1)

Πιθανά ερωτήματα :

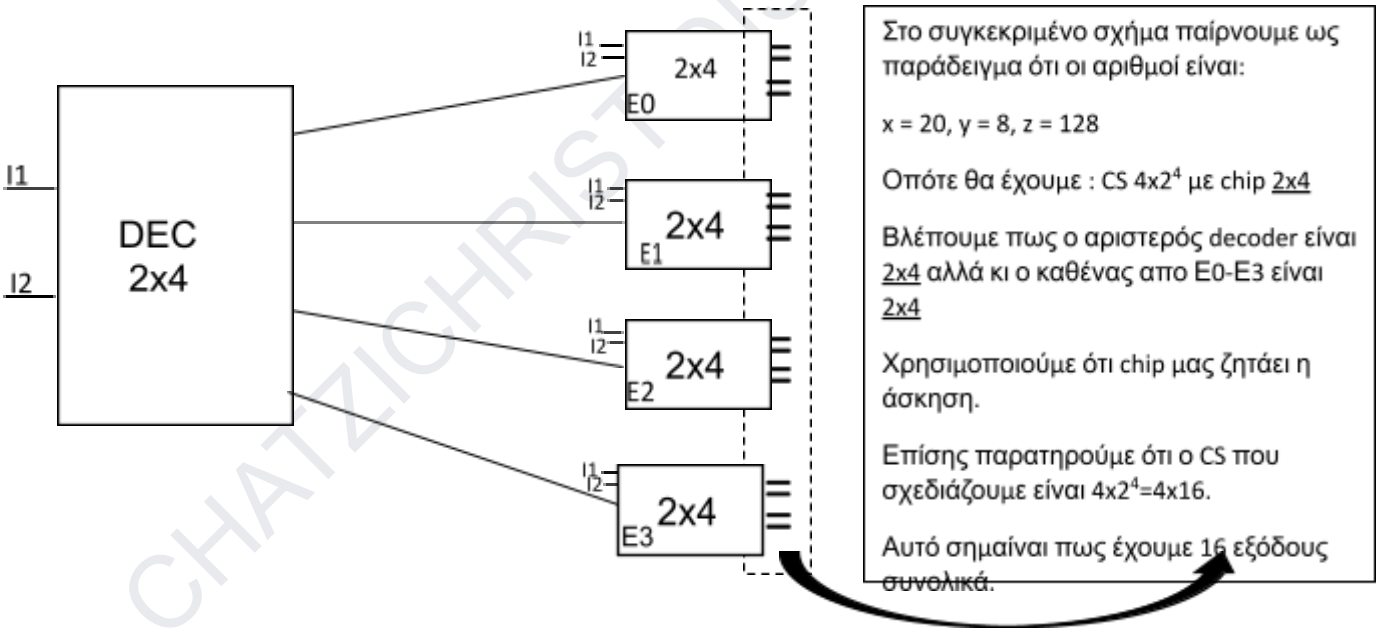
A. Πόσα chips με 2^x λέξεις (μέγεθος λέξης y bytes) χρειάζονται για να κατασκεύασουμε μνήμη z bytes ; **
B. Να σχεδιαστούν CS/WS με decoder NxM
C. Να δείξετε την αποκωδικοποίηση της λέξης με διεύθυνση .....

(Σημείωση: συνήθως ο ο αριθμός y, δηλαδή το μέγεθος λέξης θα είναι 8)

Λύση :

1. Μετατρέπουμε τον αριθμό z σε δύναμη του 2.
2. Μετατρέπουμε τον αριθμό y σε δύναμη του 2 (όπως είπαμε συνήθως 2^3).
3. Μία μνήμη με z bytes , διαθέτει z / y = a λέξεις των y bytes.
    a. (Στην διαίρεση z / y οι αριθμοί είναι σε μορφή δύναμης του 2)
4. a / 2^x = 2^b chips με 2^x λέξεις των y byte απαιτούνται για αυτή τη μνήμη.
5. CS → b x 2^b
6. WS→ x x 2^x

Σχεδίαση CS:



- Γενικά για την σχεδίαση του chip ξεκινάμε υπολογίζοντας πόσες εισόδους και πόσες εξόδους θα έχει βάσει της εκφώνησης. Συνεχίζουμε σχεδιάζοντας τον πρώτο αποκωδικοποιητή(αριστερά), έπειτα σχεδιάζουμε τις εισόδους(I1,I2,...,In) του και τις εξόδους του(D1,D2,...,Dm). Τέλος, κάνουμε το ίδιο για τα υπόλοιπα chip(δεξιά).
- Σημειώνεται πως στο παραπάνω παράδειγμα τα chip που χρησιμοποιούνται είνα 2x4 οπότε έχουν απο 2 εισόδους και 4 εξόδους το καθένα.

Σχεδίαση μονάδας ελέγχου 1/2 (Διάλεξη 9)

Έστω ότι το σήμα που θα ζητηθεί θα είναι το MARin

1) Ψάχνουμε σε ποίες εντολές βρίσκεται MAR. Αφού έχουμε MARin θα κοιτάξουμε μόνο τις εντολές στις οποίες είναι αριστερά από το βελάκι (←) (μπορεί στην εκφώνηση να ζητηθούν μόνο μερικές από αυτές τις εντολές)
2) Βλέπουμε ότι το MARin το έχουμε στις εντολές

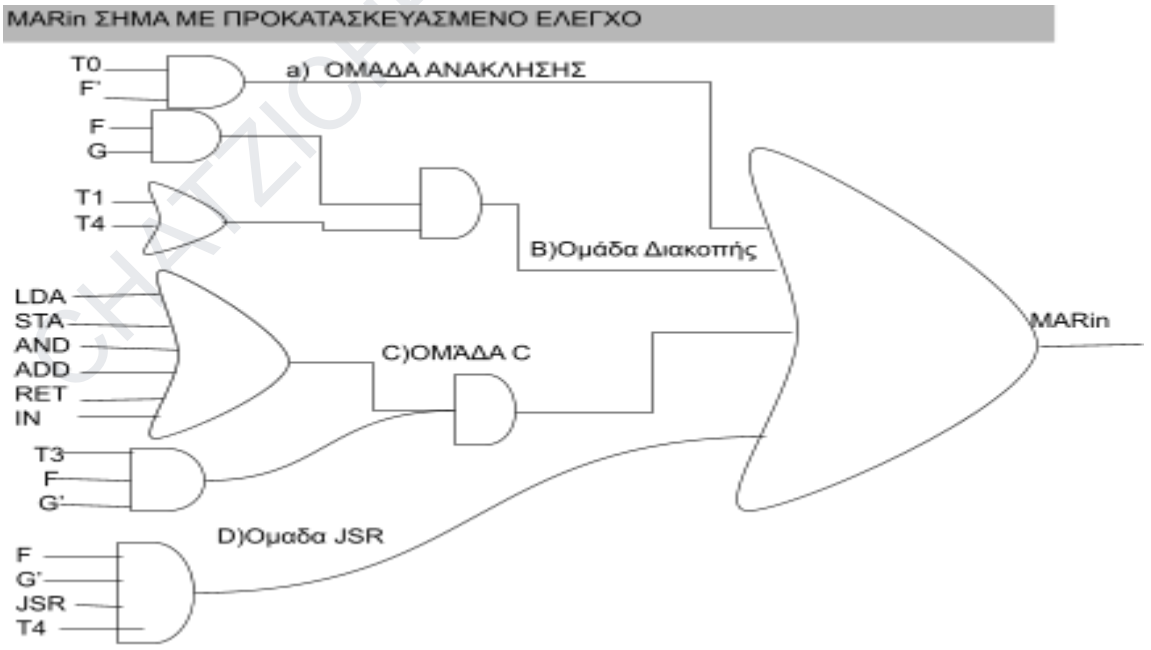
1)Ανάκληση
T0:MAR←PC
F=1
2)STA
T3:MAR←MDR[ADDRESS 1]
F=0 else G=1
3)AND
T3:MAR←MDR[ADDRESS 1]
F=0 else G=1
4)OUT
T3:MAR← MDR[ADDRESS 1]
F=0 else G=1
9) Κυκλο διακοπής
T1:MAR←Z
T4:MAR←Z
F=0 G=0
5)LDA
T3:MAR←MDR[ADDRESS 1]
F=0 else G=1
6)ADD
T3:MAR←MDR[ADDRESS 1]
F=0 else G=1
7)INC
T3:MAR← MDR[ADDRESS 1]
F=0 else G=1
8)RET
T3:MAR←SP
F=0 else G=1
10)JSR
T4:MAR←Z(Address)
F=0 else G=1

3) Τα διαχωρίζουμε σε ομάδες αναλογά με τον χρόνο και τις τιμές τον F, G(κοιτάμε για τιμές ανακλ. και διακοπής τον πινακα για τα F,G)
4) Οι ομάδες εδώ είναι 4
    a) Ομάδα Ανάκλησης με T0 kai F=0
    b) Ομάδα διακοπής με T1,T4
    c) Ομάδα των LDA,STA,AND,ADD,RET,INC με T3 F=1 καi G=0
    d) Ομάδα JSR με T4
5) Η εντολές εχουν F=1 και G=0 πέρα απο την Ανάκληση που εχει F=0 δηλαδή F^και την διακοπη που εχει F G . Δηλαδή F=1 G=1. Θα το καταλάβουμε καλύτερα απο το πινακάκι

Flip-flop F	Flip-flop G	Κατάσταση
0	0	Ανάκληση εντολής
0	1	Στάση (halt)
1	0	Εκτέλεση εντολής
1	1	Κύκλος διακοπής

Σχεδίαση μονάδας ελέγχου 2/2 (Διάλεξη 9)

6) Δημιουργούμε για κάθε ομάδα και 1 είσοδο
7) Οτιδήποτε είναι σταθερό στις ομάδες μπαίνει σε and αλλιώς θα μπει σε or
8) **Α)**Στην Ομάδα ανάκλησης θα χρειαστεί μονάχα μια AND με τις 2 **σταθερές** τιμές που ξέρουμε T0 και την F^ αφου στην ανάκληση εχουμε F=0(ΚΟΙΤΑΩ ΠΙΝΑΚΑΚΙ)
Β)Στην ομάδα διακοπής θα έχουμε μια Or με τις τιμές T1/T4 αφού **δεν είναι σταθερές** αλλά **αλλάζει** το T σε 1 και 4
και μια and με τις τιμές F=1 G=1 δηλαδή F και G αφού είναι **ετσι στο πινακάκι**
C) Στην ομάδα με τις πολλές εντολές βάζουμε όλες τις εντολές **(αφού αλλάζουν)** σε μια OR να ενώνονται με τις 3 σταθερές μας δηλαδή T3 και τα F=1 και G=0 ξανά απο πινακάκι αφου εχουμε εκτέλεση εντολής σε αυτην την ομάδα
D)Στην ομάδα JSR θα βάλουμε όλες τις σταθερές μας σε μια and και τίποτα άλλο αφού δεν έχουμε στοιχεία που αλλάζουν δηλαδή F=1 G=0 αφου είναι εκτέλεση εντολής **(ΚΟΙΤΆΩ ΠΙΝΑΚΆΚΙ)** και την T4 που είναι σταθερή
(Η JSR είναι ομάδα μόνη της επειδή είναι η μοναδική εντολή εδώ που έχει T4)



ΜΕ ΤΗΝ ΊΔΙΑ ΛΟΓΙΚΉ ΛΥΝΕΙΣ ΚΑΙ ΚΑΘΕ ΑΛΛΟ ΣΗΜΑ ΕΛΕΓΧΟΥ

- **Κρυφή μνήμη (Διαλεξη 11,12,13)**

  3 μορφές οργάνωσης:

  Άμεση συσχέτιση: Κάθε Block αντιστοιχεί σε 1 γραμμή

  Συσχέτιση Συνόλων:Κάθε Block αντιστοιχίζεται σε Α σύνολο.
  Άρα αντιστοιχίζεται σε Κ Γραμμές οπού Κ οι δρόμοι

  Πλήρη συσχέτιση: Αντιστοίχιση σε όλες

Άμεση συσχέτιση

Δεδομένα :

1. Μέγεθος λέξης.
2. Μέγεθος RAM και block RAM.
3. Μέγεθος cache και γραμμής cache.

Πιθανά ζητούμενα :

1. Να αναλύσετε τη διεύθυνση της RAM.
2. Να δώσετε τις διευθύνσεις των λέξεων που σχετίζονται με την γραμμή ..... της cache αν έχουμε τεχνική άμεσης συσχέτιση
3. Έστω η CPU ζητάει διαδοχικά τις διευθύνσεις .... , ... , ...
   a) Για κάθε διεύθυνση να βρείτε αν υπάρχει hit ή miss.
   b) Αν κάθε προσπέλαση της RAM απαιτεί Α χρονικές μονάδες και κάθε προσπέλαση της cache Β χρονικές μονάδες, να βρείτε τον συνολικό χρόνο προσπέλασης των διευθύνσεων.

Λύση:

1. **ΤΥΠΟΛΟΓΙΟ**

a) Μέγεθος RAM = 2^N bytes => Μέγεθος διεύθυνσης RAM = N bit
b) Πλήθος block RAM = Μέγεθος RAM / Μέγεθος block
c) Πλήθος γραμμών cache = Μέγεθος cache / Μέγεθος γραμμής
d) Για να βρούμε το πλήθος των block :
   Μέγεθος RAM / Μέγεθος block
e) Για να βρούμε το μέγεθος του tag :
   Πλήθος block / Πλήθος γραμμών cache
f) Για να βρούμε το πλήθος δυαδικών ψηφίων του tag:
   Μέγεθος tag = 2^x => x δυαδικά ψηφία.
g) Για να βρούμε το πλήθος δυαδικών ψηφίων της γραμμής:
   Μέγεθος line = 2^y => y δυαδικά ψηφία.
h) Για να βρούμε το πλήθος δυαδικών ψηφίων του byte:
   Μέγεθος διεύθυνσης$_{(a)}$ – Μέγεθος tag$_{(e)}$ – Μέγεθος γραμμής$_{(g)}$ = z δυαδικά ψηφία
i) Άρα η διεύθυνση θα έχει τη μορφή :

Tag	Γραμμή	Byte
x bits	y bits	z bits

2. Άμεση συσχέτιση :
   Για να σχετίζονται οι διευθύνσεις των λέξεων με την ..... γραμμή της cache, θα πρέπει το πεδίο γραμμής να δείχνει στο αντίστοιχο δυαδικό της γραμμής. Δηλαδή

Tag	Line	Byte
	……	

   Σημείωση: όπου βλέπουμε ..... είναι ο δοσμένος αριθμός από εκφώνηση.
   Οι διευθύνσεις των ζητούμενων λέξεων θα είναι όλες όσες έχουν τον συγκεκριμένο δυαδικό αριθμό στη γραμμή, από όλα τα block της RAM.

3.

α)

1. Φτιάχνουμε ένα Tag Directory με μέγεθος = πλήθος γραμμών cache x μήκος tag.
2. Επειδή η μνήμη είναι άμεσης συσχέτισης, αν υπάρχει μέσα στη cache θα βρίσκεται μαζί με ολόκληρο το block στο οποίο ανήκει η γραμμή της διεύθυνσης.
   Σημείωση: Κάθε χρονική στιγμή μπορεί να υπάρχει ένα block και αυτό καθορίζεται από την τιμή του tag της διεύθυνσης.
3. Έρχεται η πρώτη ζητούμενη διεύθυνση.Πρέπει να την αναζητήσουμε στη cache
   Διαβάζουμε την τιμή πεδίου Γραμμή
   Πηγαίνουμε στην αντίστοιχη γραμμή του tag directory και διαβάζουμε το αποθηκευμένο tag
   Αν είναι ίσο, τότε η ζητούμενη διεύθυνση μαζί με όλο το block υπάρχει στη cache => έχουμε hit. Αλλιώς έχουμε miss.
   Συνεχίζουμε για όλες τις διευθύνσεις που ζητούνται ομοίως.

b)

1. $t = hits \times t_{cache} + misses \times t_{RAM}$
   όπου t : ο συνολικός χρόνος προσπέλασης
   t_{cache} : χρόνος προσπέλασης cache
   t_{RAM} : χρόνος προσπέλασης RAM

Συσχέτιση συνόλων (K-way)

Ζητούμενα :
Ανάλυση διεύθυνσης.

Λύση :

Για να αναλύσουμε τη διεύθυνση εδώ πρέπει να βρούμε το πλήθος των συνόλων.
Κάθε σύνολο περιέχει K γραμμές.

Η μορφή της διεύθυνσης εδώ θα είναι :

Tag	Set	Byte

a) Μέγεθος RAM = 2^N bytes => N bit διευθυνσιοδότηση
b) Πλήθος block RAM = Μέγεθος RAM / Μέγεθος block
c) Πλήθος συνόλων = Πλήθος γραμμών cache / K (Πλήθος δρόμων)
d) Πλήθος block / Πλήθος συνόλων = Πλήθος block που αντιστοιχίζονται σε κάθε σύνολο.
e) Κάνουμε το πλήθος block που αντιστοιχίζονται σε κάθε σύνολο δύναμη του 2.
   Ο εκθέτης μας δίνει το μέγεθος του tag.
f) Κάνουμε το πλήθος των συνόλων δύναμη του 2
   Ο εκθέτης μας δίνει το μέγεθος του set.
g) Κάνουμε το μέγεθος λέξης δύναμη του 2
   Ο εκθέτης μας δίνει το μέγεθος του byte.

Πλήρης συσχέτιση 1/2

Το πλήθος bit του tag δείχνει πόσα block αντιστοιχίζονται σε κάθε γραμμή της cache

δηλ. $2^{\text{πλ bit cache}}$ = block ανα γραμμή

Έχω τόσους συγκριτές όσες οι γραμμές τις cache

Πλεονέκτημα : Κάθε block πάει παντού, δε χρειάζεται συνεχείς αντικαταστάσεις

Μειονέκτημα : Αύξηση του hardware

Ζητούμενα :

1. Ανάλυση διεύθυνσης.
2. Εξήγηση του tag (τι δείχνει)
3. Hit/Miss με πολιτική αντικατάστασης LRU/FIFO

Λύση :

1. Η μορφή της διεύθυνσης εδώ θα είναι :

Tag	Byte

Το πεδίο γραμμής δε χρειάζεται, αφού κάθε block μπορεί να πάει παντού.

2. Το πλήθος bit του tag εδώ μας δείχνει πόσα block αντιστοιχίζονται σε κάθε γραμμή της cache (π.χ. 32byte cache = 2^5 => 5bit tag => 5 block μνήμης σε κάθε γραμμή

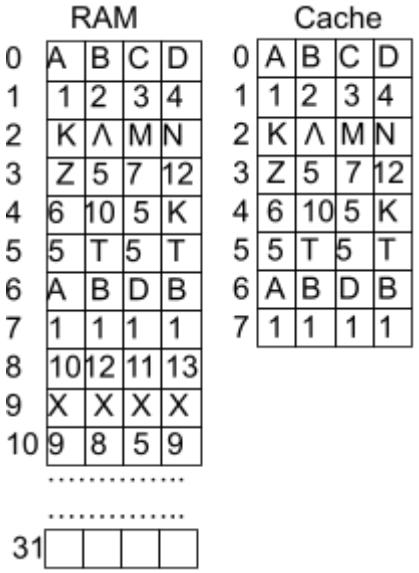
Byte βρίσκουμε απο τον τύπο : $2^{byte} = 2^{\text{μέγεθος λέξης}} / 2^{\text{γραμμές}}$

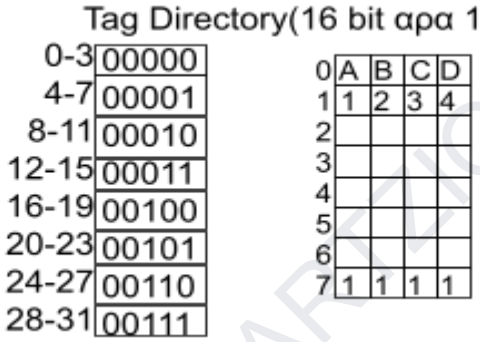
3.Ελέγχουμε αν το tag των διευθύνσεων που μας ζητούνται υπάρχουν στο Tag Directory. Αν υπάρχει τότε λέμε ότι έχουμε X συγκρίσεις, όπου X ο αριθμός γραμμών της cache, άρα και X συγκριτές και Hit. Αν δεν υπάρχει τότε λέμε ότι έχουμε Miss.

Πλήρης συσχέτιση 2/2

ΠΑΡΑΔΕΙΓΜΑ:

Σε μία χρονική στιγμή, τα Block της RAM 0-7 έχουν φορτωθεί αντίστοιχα στις γραμμές 0-7 της Cache. Για τις διευθύνσεις: 32,33,60 Νά εξετάσετε αν υπάρχει HIT η MISS .

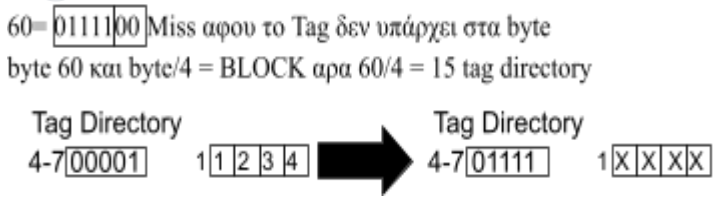




1) Πλήρη συσχέτιση :τόσοι συγκριτές οσες οι γραμμές της Cache, δηλαδή 8
2) Επειδή 32/4=8 , η ζητούμενη διεύθυνση βρίσκεται στο Block 8
3) το Tag 01000 δεν υπάρχει στο Tag Directory. Αρα Miss

Βγάζουμε τη γραμμή που μπήκε πρώτη (αφού έχουμε FIFO)και τοποθετούμε το νεο Block

Το 33 ομως αφου είναι 01000 01 και το TAG Directory τωρα έχει το 01000 μέσα απο το 32 που μπήκε θα είναι hit
33=01000 01 αρα είναι hit αφού το tag directory πλέον έχει το 01000 μέσα
Το 60 όμως για παράδειγμα που αλλάζει και δεν έχει το 01000 άλλα έχει 01111 στο TAG DIRECTORY μπροστά θα είναι και αυτό ξανά miss



Για την LRU (least recently used) θα μας δίνεται λίστα για το ποιες έχουν χρησιμοποιηθεί λιγότερο από τις διευθύνσεις και θα τα πάρουμε αυτά με τη σειρά. Ο τρόπος παραμένει ο ίδιος

Flip-Flop και Clock (Διάλεξη 5)

Λειτουργία :

1. CLK: 0→1 θετική μετάβαση/ακμοπυροδότηση

   Η τιμή της D έρχεται στις γραμμές εισόδου (απο MDR → δίαυλο δεδομένων)

2. CLK=1(θετικό) : δεν αποθηκεύεται καμία αλλαγή

3. CLK=0(αρνητικό) : δεν γίνεται καμία αλλαγή

Αλλαγή της τιμής που αποθηκεύει η Flip-Flop μπορεί να γίνει σε έναν από τους δύο χρόνους:

A. Όταν CLK από 0 γίνεται 1 (τη στιγμή που αλλάζει)

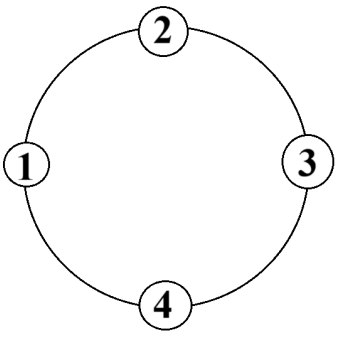
B. Όταν CLK από 1 γίνεται 0 (τη στιγμή που αλλάζει)

Η τιμή που αποθηκεύεται στη μνήμη είναι η τιμή της εξόδου Q

Τα Flip-Flop έχουν τη δυνατότητα παραγωγής και της συμπληρωματικής εξόδου Q΄

<u>Καταστάσεις CLK :</u>

1. Παραμένει 0
2. Μεταβαίνει από 0 σε 1
3. Παραμένει 1
4. Μεταβαίνει από 1 σε 0



Το ρολόι CLK εναλλάσσεται σε τιμές 0 και 1

1. Έστω ότι αρχικά CLK = 0 και Q = 0.

   CLK = 0 σημαίνει ότι οι πύλες 2 και 3 είναι ίσες με 1, δηλ. S = R = 1.

   Αν Q = 0 => Q΄ = 1 , άρα η έξοδος της πύλης 6 θα παραμείνει 1

   άρα η έξοδος της πύλης 5 θα παραμείνει 0 } κύκλωμα αμετάβλητο

2. Το ρολόι μεταβαίνει απο 0 σε 1.

Κατά τη στιγμή της μετάβασης ήταν S = R = 1.

Για να δούμε τι θα συμβεί εξετάζουμε την τιμή του D

A. Αν D = 0 η πύλη 4 θα έχει έξοδο 1

   Ø Η (4) δίνει έξοδο 1

   Ø Αυτό το 1 είναι είσοδος της (1) και μαζί με το S = 1 θα κάνουν την έξοδο της (1) ίση με το 0.

   Ø Η (2) θα έχει μία είσοδο ίση με 0, άρα το S θα παραμείνει 1

   Ø Όμως, το CLK = 1, S = 1, (4) = 1, άρα (3) = 0

   Ø Άρα Q΄= 1 και επειδή Q΄= S = 1 θα είναι Q = 0

B. Αν D = 1, S = R = 1

   Ø Η έξοδος της (4) θα είναι 0 , γιατί R = D = 1.

   Ø Επειδή S = 1 και (4) = 0, η (1) θα έχει έξοδο 1

   Ø Άρα, οι είσοδοι της (2) είναι 1 (από την (1)) και 1 από το CLK

   Ø S = 0 => Q = 1

   Ø Επειδή Q = 1 και R = 1 => Q΄= 0

Συμπέρασμα: Όταν έχουμε θετική μετάβαση (CLK: 0→1) , τότε Q = D .

3. Έστω Q = 1, Q΄= 0 το CLK παραμένει 1, R = 1 και S = 0

Έρχεται ένα bit με τιμή 0 και τροφοδοτεί το D. Άρα, το D αλλάζει.

D = 0 => (4) = 1 Άρα, το S θα παραμείνει 0 => το R θα παραμείνει 1

Τελικά S = 0 , R = 1 και Q = 1, Q΄= 0 => Το κύκλωμα μένει στην ίδια κατάσταση

Η τιμή του D δεν αποθηκεύεται όσο το CLK = 1

Το ρολόι κάποια στιγμή θα πάει στο 0 => σταθερό κύκλωμα

Το ρολόι κάποια στιγμή θα πάει στο 1 => η οποιαδήποτε τιμή του D θα αποθηκευτεί