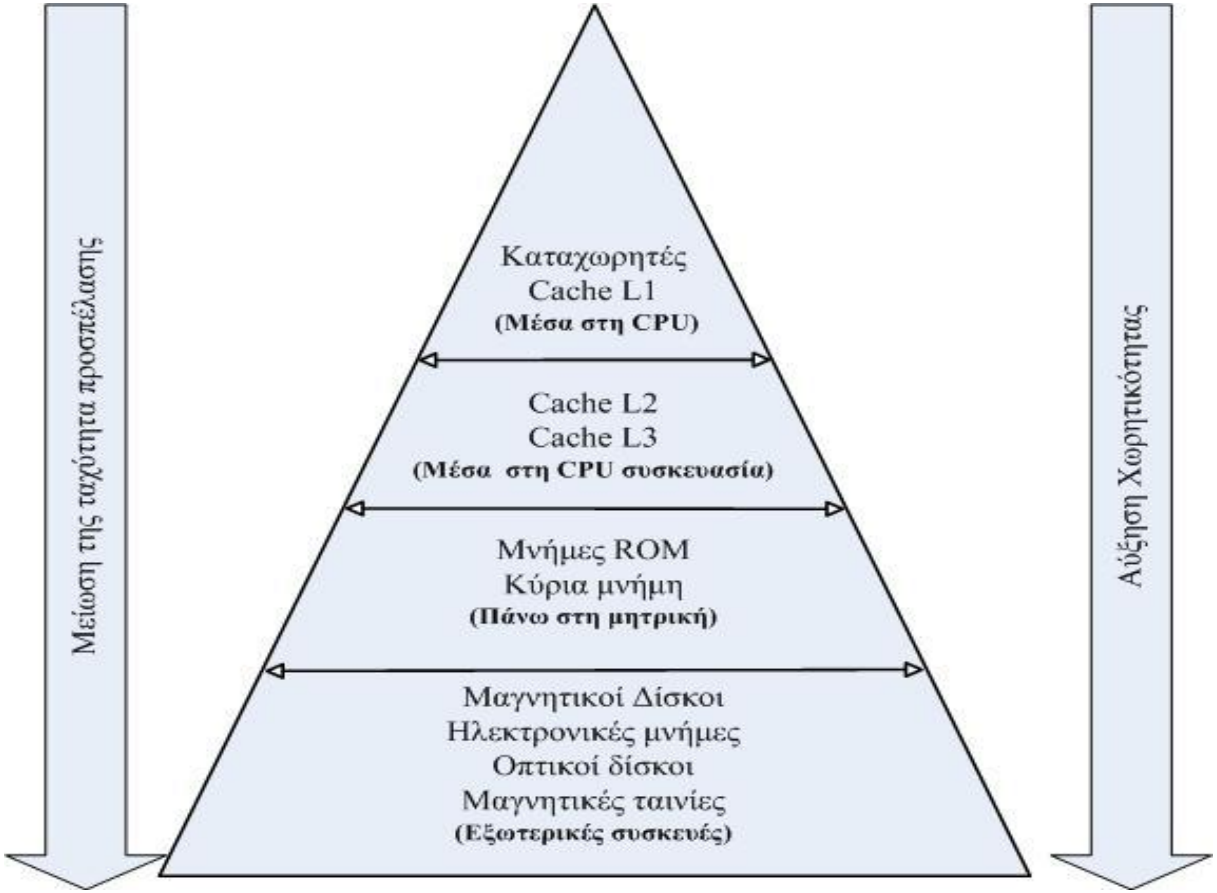
Προηγμένες αρχιτεκτονικές υπολογιστών και προγραμματισμός παραλλήλων συστημάτων

05. Συστήματα Μνήμης – Βασικές Αρχές Cache

Περιεχόμενα

- Συστήματα Μνήμης.
  - Η τεχνολογία της ιεραρχημένης μνήμης.
  - Οργάνωση της μνήμης cache.
    - Η βασική αρχή λειτουργίας.
    - Οι παράμετροι της cache.
    - Απόδοση της cache.
  - Οργάνωση της Κύριας Μνήμης.
    - Φυσική οργάνωση.
    - Λογική οργάνωση.
    - Απόδοση συστήματος cache - Κύριας Μνήμης.

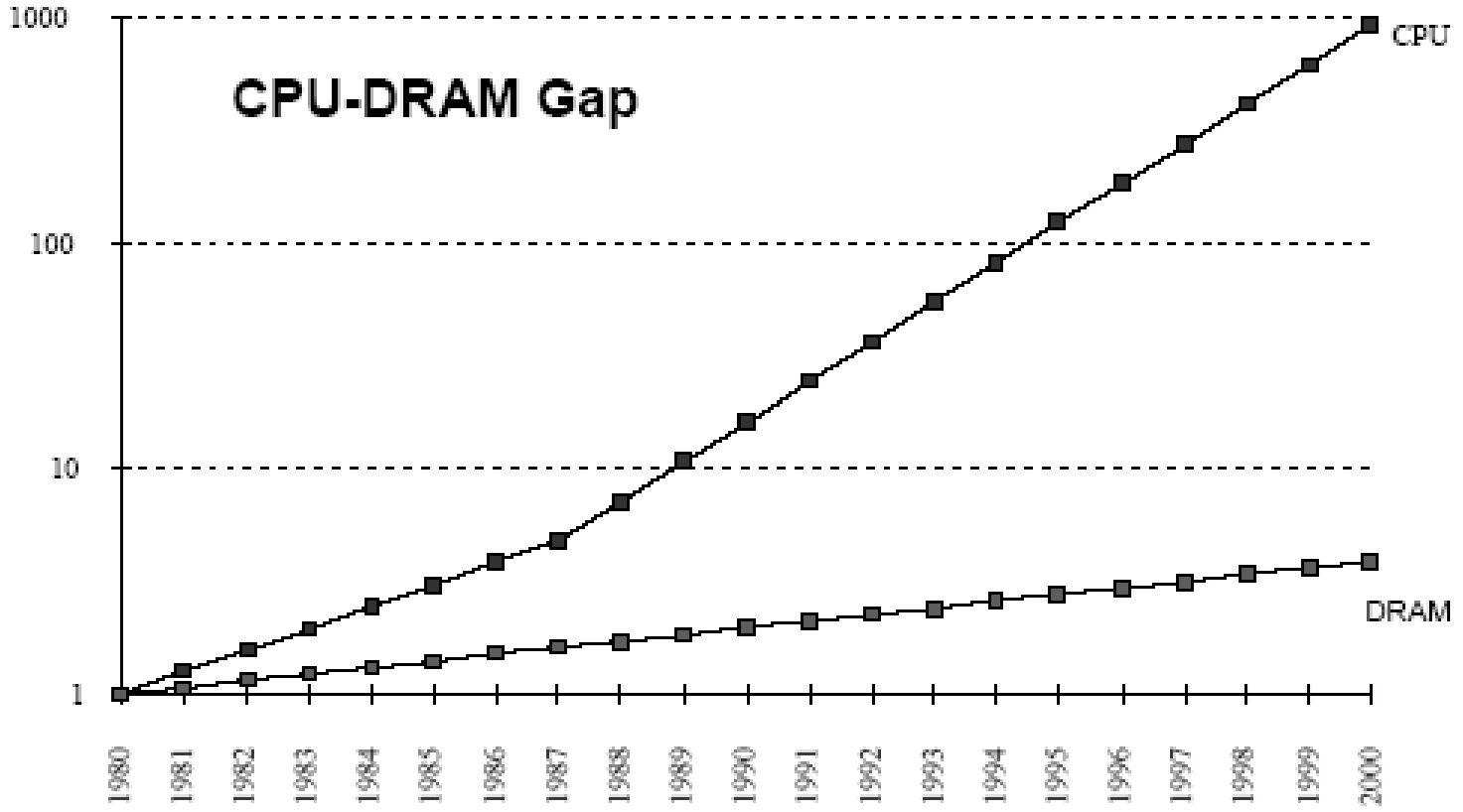
Η Ιεραρχία μνήμης



Μνήμες - χωρητικότητες- ταχύτητες

	Χωρητικότητα	**Χρόνος προσπέλασης**	**Τεχνολογία**
Καταχωρητές	1KB – 8KB	0.2 – 0.5 ns	CMOS
Cache L1	8KB – 64KB	1 – 2 ns	CMOS SRAM
Cache L2	64KB – 256KB	2 – 5 ns	CMOS SRAM
Cache L3	2MB – 16MB	10 – 20 ns	CMOS SRAM
Κύρια Μνήμη	2GB – 16GB	>20 ns	CMOS DRAM
Δίσκος	>100GB	5 – 8 ms	Μαγνητικό μέσο

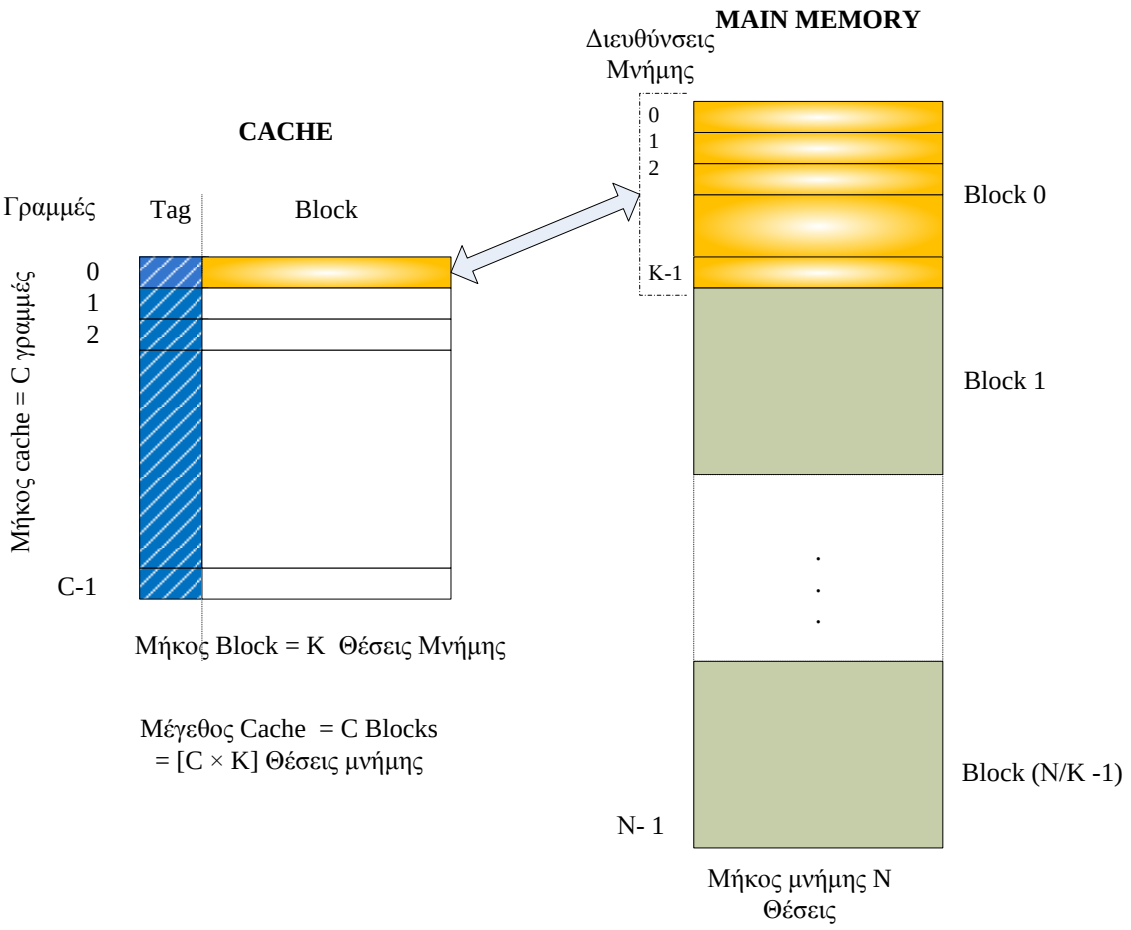
Απόκλιση ταχυτήτων CPU και DRAM



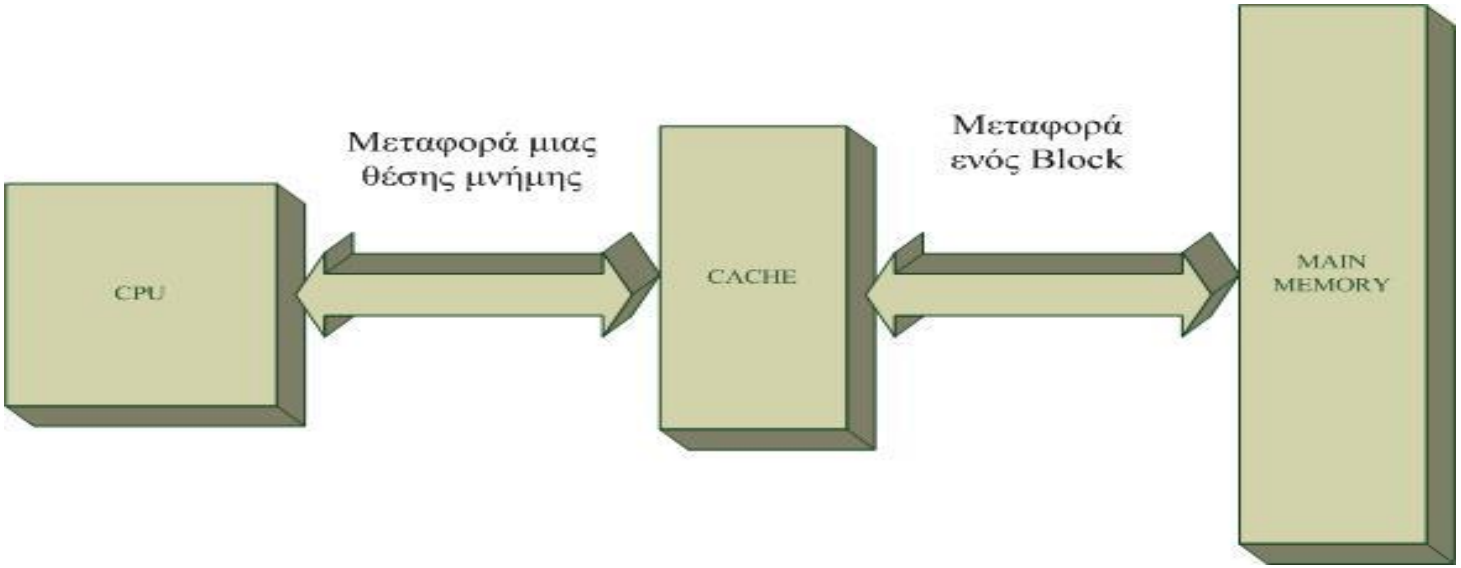
Τοπικότητα των κλήσεων

- **Αρχή της τοπικότητας (locality principle)**: Στα περισσότερα προγράμματα διαδοχικές (στο χρόνο) εντολές ή δεδομένα καλούνται από διαδοχικές ή κοντινές περιοχές της μνήμης.
- Η **χρονική τοπικότητα** (temporal locality/locality on time) αναφέρεται στο γεγονός ότι διευθύνσεις της μνήμης που χρησιμοποιήθηκαν κάποια χρονική στιγμή έχουν μεγάλη πιθανότητα να ξαναχρησιμοποιηθούν σύντομα.
- Η **χωρική τοπικότητα** (spatial locality/locality in space) αναφέρεται στο γεγονός ότι δύο γειτονικές διευθύνσεις θα χρησιμοποιηθούν διαδοχικά στο μέλλον.

Το συγκρότημα cache – κύρια μνήμη

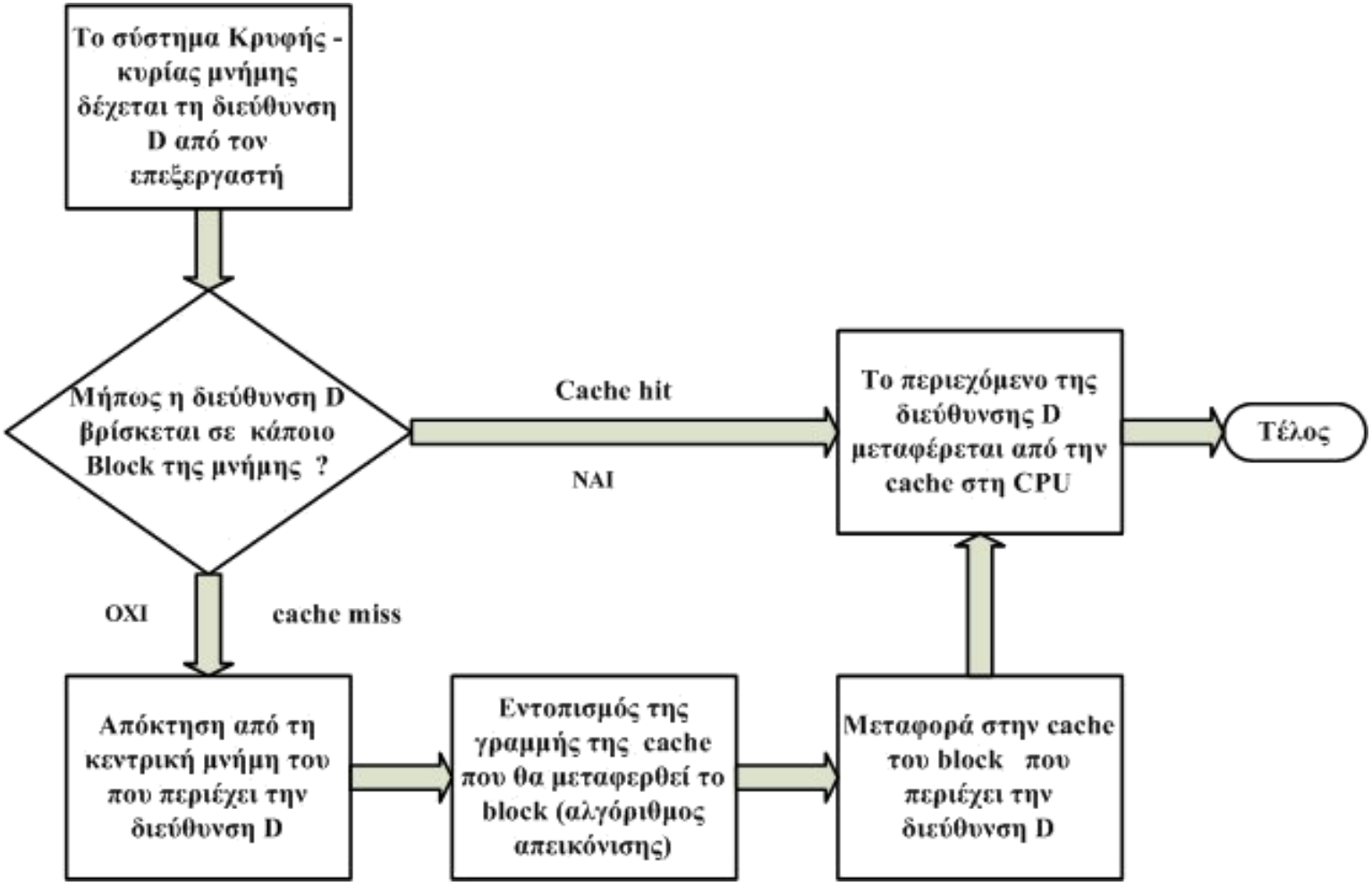


Η βασική ιδέα



- **Cache hit**: Η CPU βρίσκει τη διεύθυνση στη cache.
- **Cache miss**: Η CPU δεν βρίσκει τη διεύθυνση στην cache.
- **Hit rate**: Ποσοστό επιτυχημένων προσπελάσεων.
- **Miss rate**: Ποσοστό αποτυχημένων προσπελάσεων.

Ο αλγόριθμος λειτουργίας



Χρησιμότητα της cache

- Βελτιώνει τον χρόνο προσπέλασης των δεδομένων από την CPU.
- Οι σημερινές cache έχουν hit rate ≥ 0.95 (δηλαδή ≥ 95 στις 100 φορές η ζητούμενη από τη CPU διεύθυνση βρίσκεται στην cache).
- Παράδειγμα:
- Κεντρική μνήμη με access time 30ns και cache με access time 1ns και hit rate 0.95 ή 95%.
- Τότε: Μέσος χρόνος κλήσης = 0.95×1ns + 0.05×(1ns+30ns) = 2.5ns.
- Βελτίωση = 30/2.5 = 12 φορές.

Τύποι μέσου χρόνου T_{AV} και βελτίωσης n

- Γενικά αν T_{cache}, T_{mem} είναι οι χρόνοι απόκρισης της μνήμης cache και της κύριας μνήμης αντίστοιχα, τότε:
- Μέσος χρόνος απόκρισης για hit rate [hr %] είναι:

$$T_{AV} = [hr/100] \times (T_{cache}) + [1 - hr/100] \times (T_{cache} + T_{mem})$$

- Δηλαδή επιτυγχάνεται βελτίωση n φορές στο χρόνο απόκρισης όπου:

$$n = T_{mem}/T_{AV} = 1 / [1 - (hr/100) + (T_{cache}/T_{mem})]$$

Παράμετροι cache

- Μέγεθος (Cache size):
  - αρκετά μικρό ώστε η cache να μην κοστίζει πολύ, αλλά και
  - αρκετά μεγάλο ώστε ο χρόνος προσπέλασης του συστήματος cache - K. Μνήμη να είναι κοντά σε αυτόν της cache.
- Συναρτήσεις απεικόνισης
  - Direct mapping (άμεσης απεικόνισης).
  - Full associative mapping (πλήρως συσχετική).
  - Set associative mapping (τμηματικά συσχετική).
- Αλγόριθμοι αντικατάστασης (Replacement algorithms).
- Πολιτική εγγραφής (Write policy).
- Μήκος γραμμής cache (Line size or Block size).
- Επίπεδα cache (number of cache).

Τυπικά μεγέθη cache ανά CPU

Επεξεργαστής	Έτος κατασκευής	Cache L1	Cache L2	Cache L3
IBM 360.85	1968	16-32 KB		
PDP 11/70	1975	1 KB		
VAX 11/780	1978	16 KB		
IBM 3090	1985	128-256 KB		
Intel 80486	1989	8 KB		
Pentium	1993	8 KB/8KB	256-512 KB	
Power PC 601	1993	32 KB		
Power PC 620	1996	32 KB /32 KB	256 – 1000 KB	2 MB
Power PC G4	1999	32 KB /32 KB	256 – 1000 KB	
Pentium 4	2000	8 KB / 8KB	256 KB	
Itanium	2001	16 KB /16 KB	96 KB	4 MB

Η συνάρτηση απεικόνισης

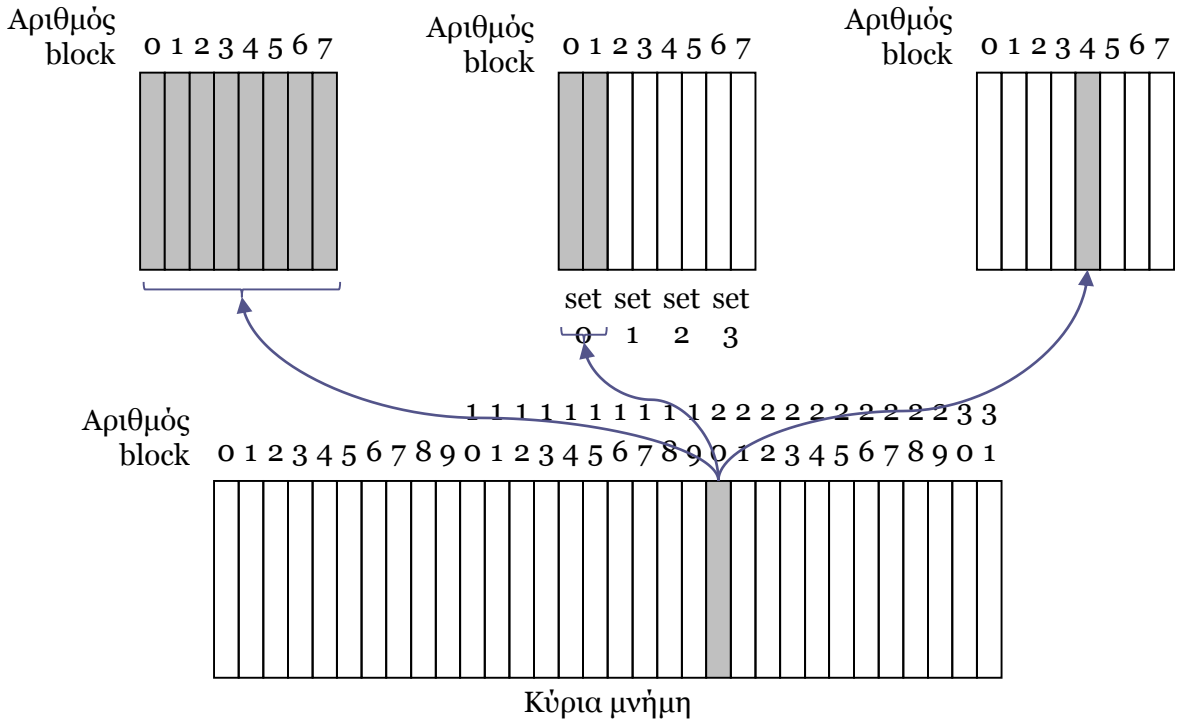
- Η άμεσης απεικόνισης (direct mapping).
- Η πλήρως συσχετική απεικόνιση (fully associative mapping).
- Η τμηματικά συσχετική απεικόνιση (set associative mapping).

Τεχνικές οργάνωσης της cache

Πλήρως συσχετιστική: το block 20 μπορεί να τοποθετηθεί οπουδήποτε

Τμηματικά συσχετιστική: το block 20 μπορεί να τοποθετηθεί οπουδήποτε μέσα στο set 0 (0 = 20 mod 4)

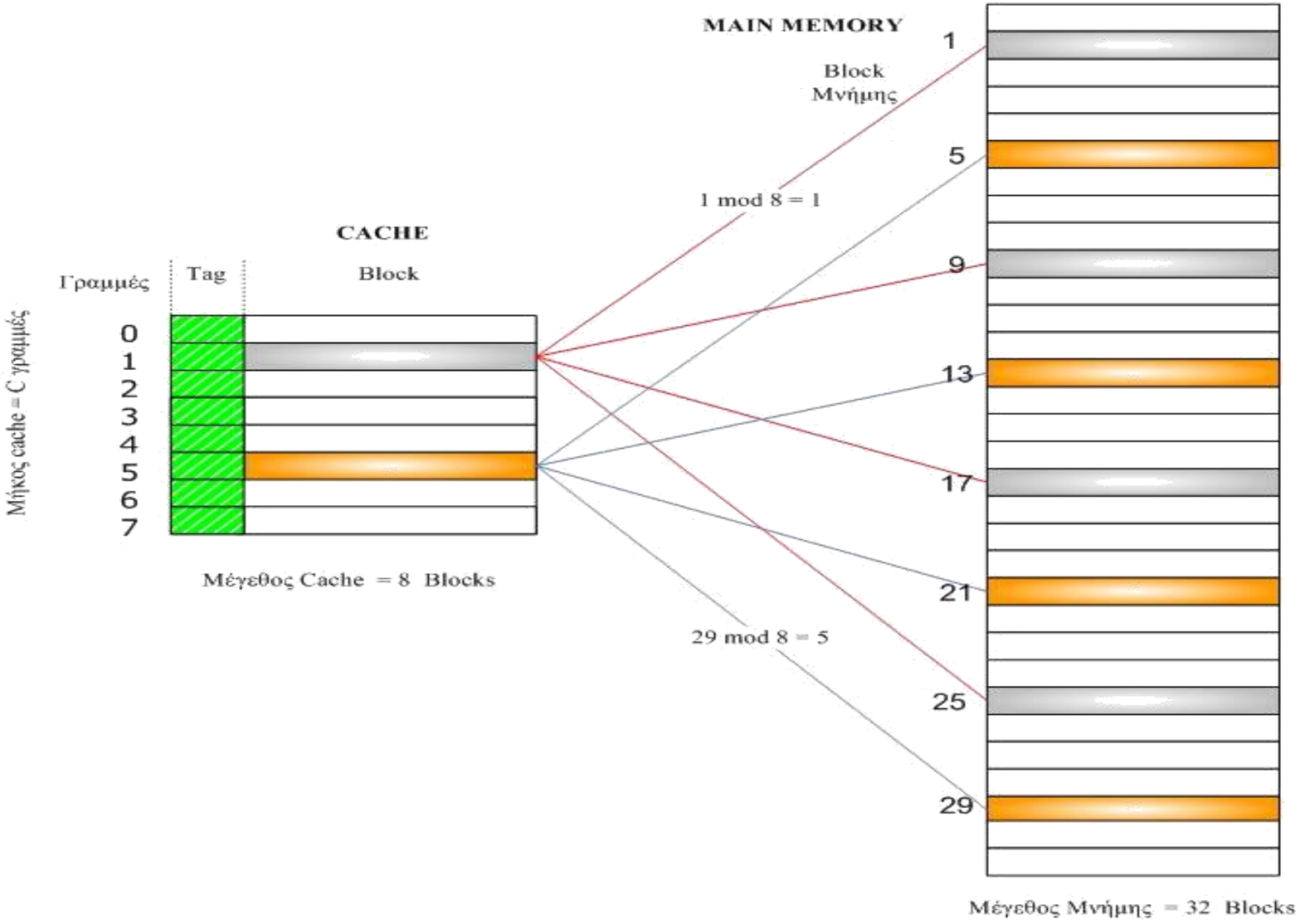
Άμεσης απεικόνισης: το block 20 μπορεί να τοποθετηθεί μόνο στη θέση 4 (4 = 20 mod 8)



Οργάνωση άμεσης απεικόνισης (direct mapping)

- Κάθε block στην κεντρική μνήμη αντιστοιχεί μόνο σε μια συγκεκριμένη cache line (ένα block πάντοτε μεταφέρεται στην ίδια διεύθυνση της cache η οποία υπολογίζεται από την συνάρτηση απεικόνισης).
- Συνάρτηση απεικόνισης (mapping function):
  - Αν j = η διεύθυνση ο ενός block στην κεντρική μνήμη και m = το μέγεθος της cache σε γραμμές.
  - Τότε $L = j$ modulo m είναι η διεύθυνση της γραμμής στην cache που αυτό θα τοποθετηθεί.

Άμεση απεικόνιση – Διευθύνσεις



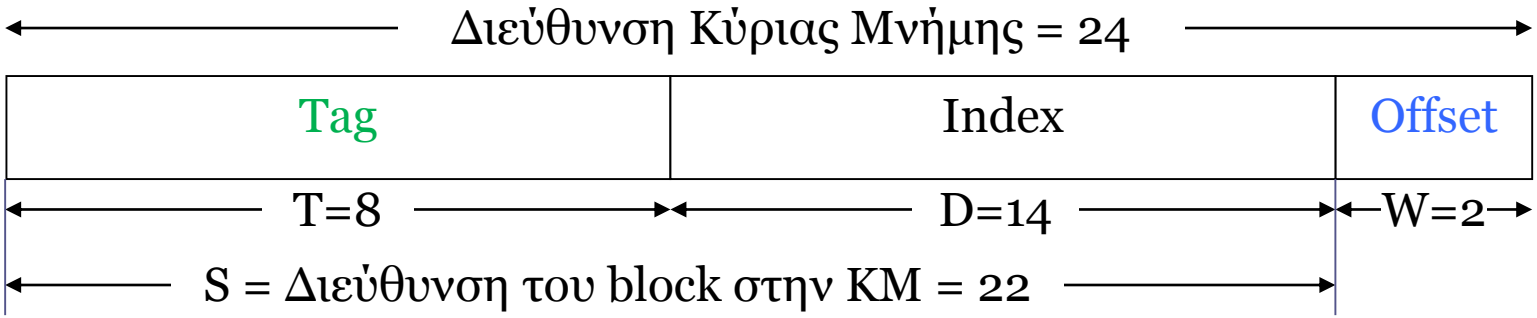
Υποθέσεις για τα παραδείγματα

- Στα επόμενα θα υποθέσουμε ότι έχουμε:
  - CACHE:
    - Μέγεθος Cache: 64 kByte = 216 (καθαρό χωρίς τα tags).
    - Μέγεθος block: 4 bytes = 22 bytes.
    - Πλήθος blocks: 16 k (214 = 216/22).
    - Μήκος διεύθυνσης block: 14 bit.
  - ΚΥΡΙΑ ΜΝΗΜΗ:
    - Μέγεθος: 16 Mbytes.
    - Byte addressable memory (Διεύθυνση ανά byte).
    - Μήκος διεύθυνσης κύριας μνήμης : 24 bit (224 = 16 M).

Άμεση απεικόνιση: το φαινόμενο thrasing

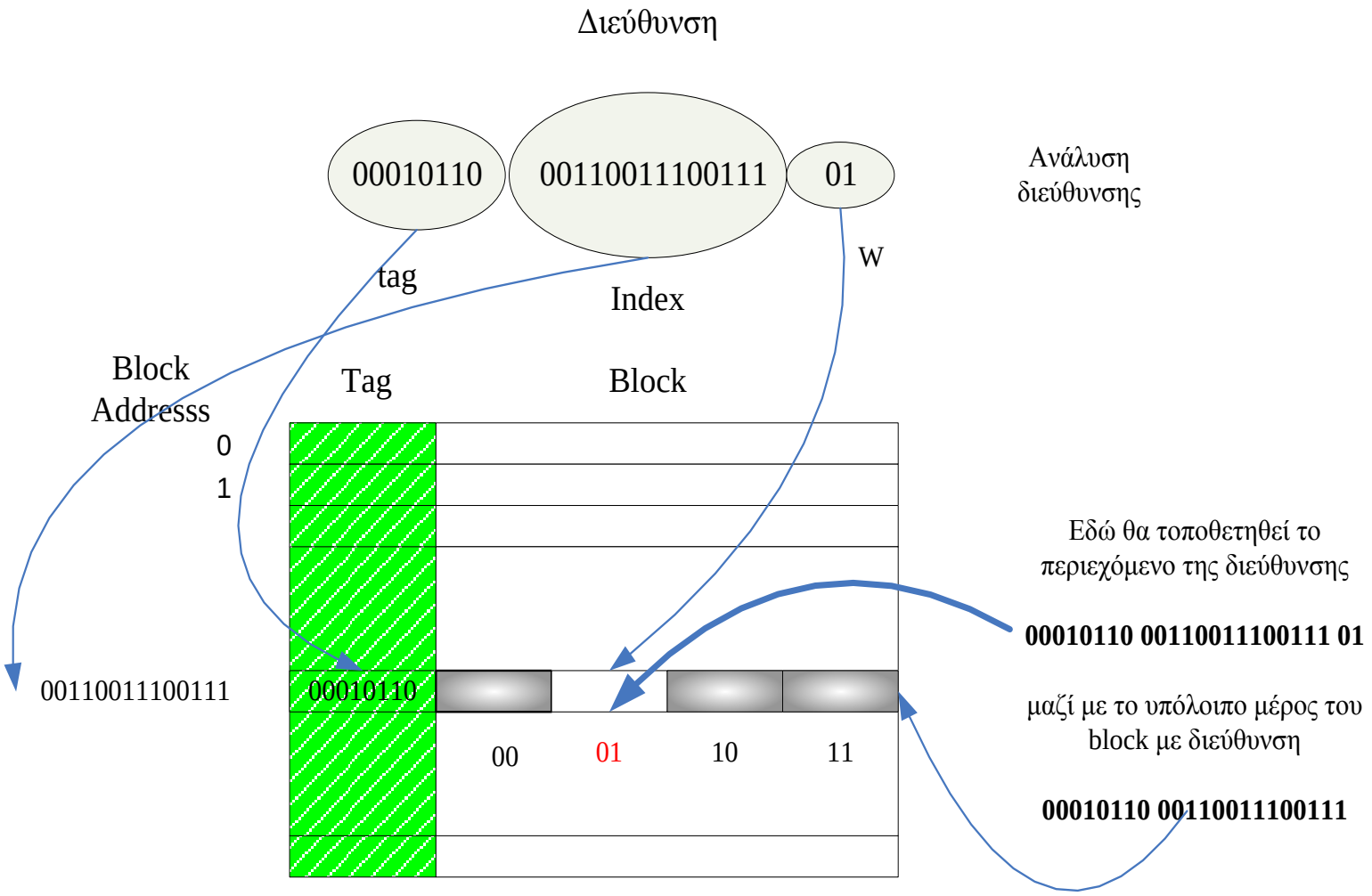
- Ας υποθέσουμε ότι ένα πρόγραμμα ζητάει συνεχώς δεδομένα εναλλάξ από δύο θέσεις της μνήμης που έχουν το ίδιο index.
- Επειδή και οι δύο δεν μπορούν να βρίσκονται στην cache ταυτόχρονα εναλλάσσονται συνεχώς μεταξύ της μνήμης και της cache.
- Το φαινόμενο αυτό ονομάζεται thrasing (συνεχές χτύπημα).

Άμεση απεικόνιση - Διεύθυνση μνήμης



- Offset : Η διεύθυνση ενός byte μέσα στο block.
- Index : Η διεύθυνση ενός block μέσα στην cache.
- Tag : Ετικέτα αναγνώρισης (μέρος της διεύθυνσης).
- Μέγεθος Block: 4 bytes = 2^w bytes → W = 2
- Μέγεθος Μνήμης: 16 Mbytes = 2^{24} = 2^{s+w}
- Μήκος διεύθυνσης Κ.Μ.: (S + W) bits = 24 bits → S = 22
- Πλήθος blocks στη μνήμη: $2^s = 2^{22}$ = 4 Mblocks
- Μέγεθος Cache: 64 Kbytes = 2^{16} bytes
- Πλήθος blocks στην cache: $2^D = 2^{16} / 2^2 = 2^{14}$ → D = 14
- Μέγεθος tag: T = S−D bits = 8

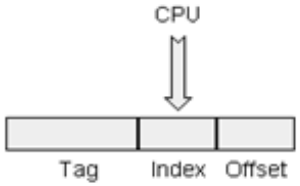
Τοποθέτηση ενός δεδομένου στη cache



Άμεση οργάνωση - Αλγόριθμος Αναζήτησης - hit

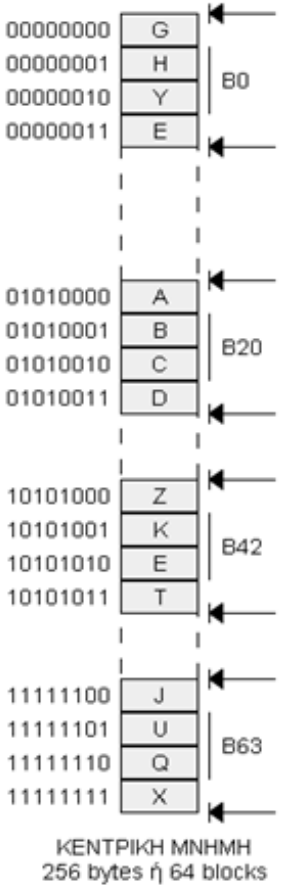
Βήμα 0: Ανάλυση της διεύθυνσης στα πεδία, tag, index, W **(01 0100 10).**
Βήμα 1: Η index διεύθυνση μας οδηγεί σε μια γραμμή της cache στην οποία, **ενδεχομένως,** να βρίσκεται το περιεχόμενο της διεύθυνσης που αναζητούμε.
Βήμα 2: Το tag της διεύθυνσης συγκρίνεται με ο tag της γραμμής της cache στην οποία μας οδήγησε η διεύθυνση index.
Βήμα 3: Αν τα δύο tag είναι ίσα τότε έχουμε **cache hit.** Η διεύθυνση βρίσκεται στη γραμμή (block) της cache με διεύθυνση index.
Βήμα 4: Το πεδίο W (offset) βοηθά στον εντοπισμό της θέσης μνήμης **(01 0100 10)** μέσα στο block.
Βήμα 5: Μεταφέρεται έξω από την cache το μέρος του block που έχει διεύθυνση **(01 0100 10).**

Άμεση οργάνωση – Cache Hit (0)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

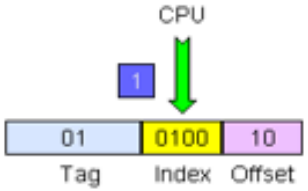
CACHE 16 Blocks



ΚΕΝΤΡΙΚΗ ΜΝΗΜΗ
256 bytes ή 64 blocks

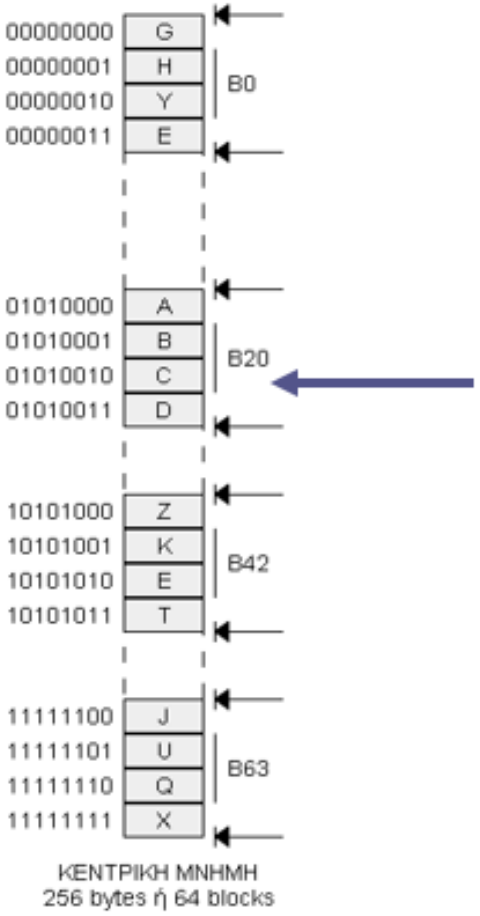
ΣΥΓΚΡΙΣΗ TAG

Άμεση οργάνωση – Cache Hit (1)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

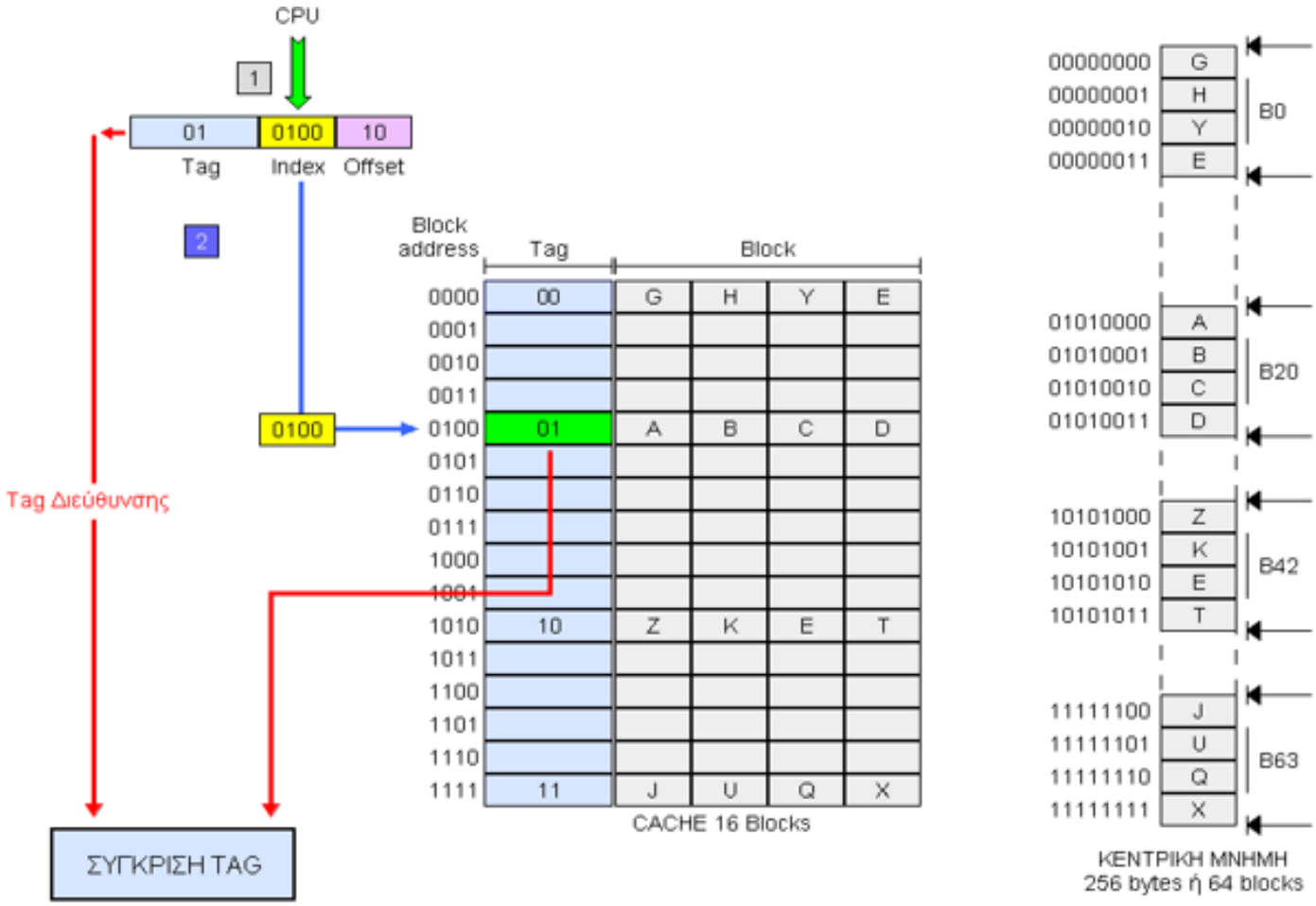
CACHE 16 Blocks



ΣΥΓΚΡΙΣΗ TAG

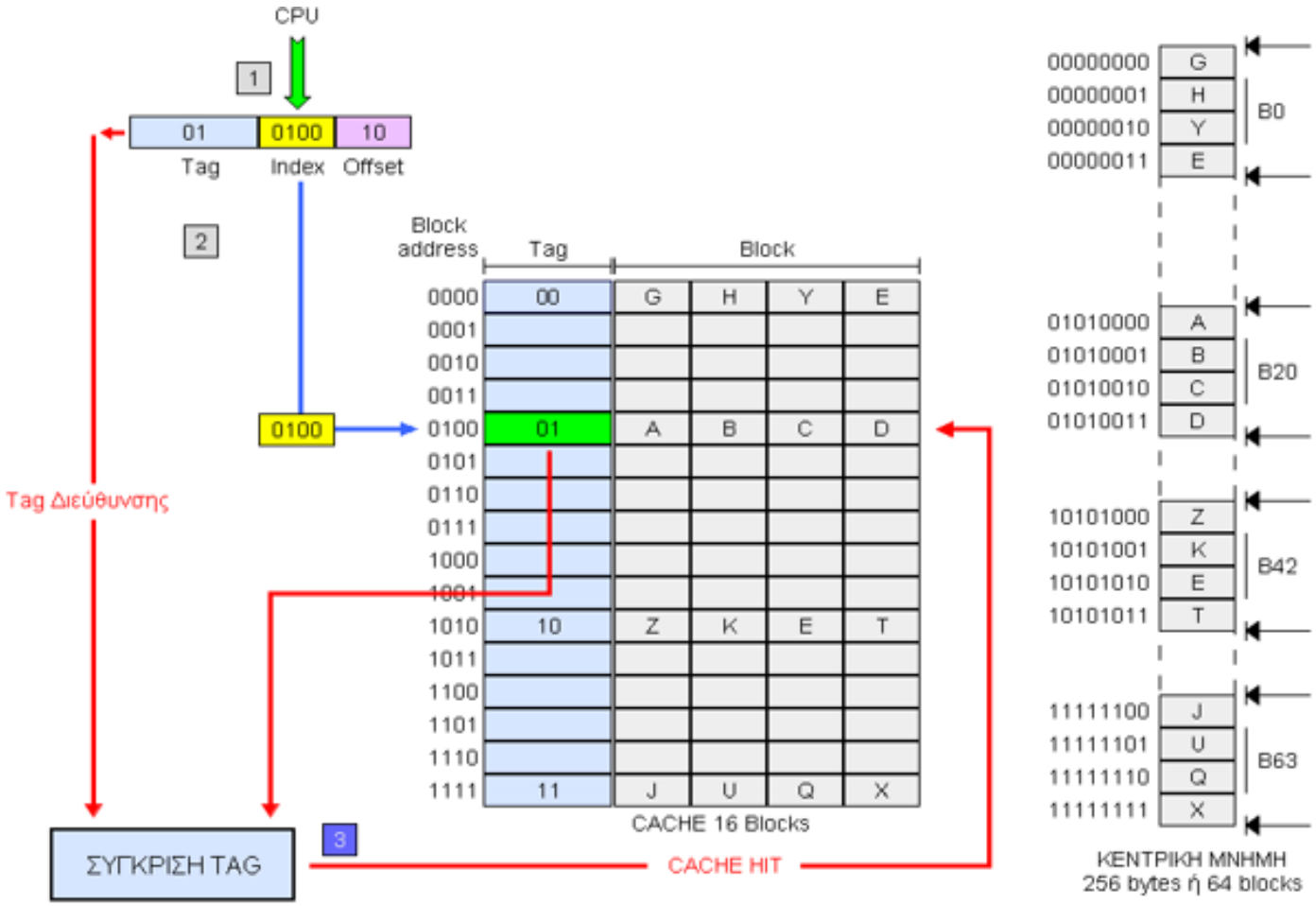
1. Η διεύθυνση εισέρχεται από τη CPU και χωρίζεται σε Tag, Index και Offset

Άμεση οργάνωση – Cache Hit (2)



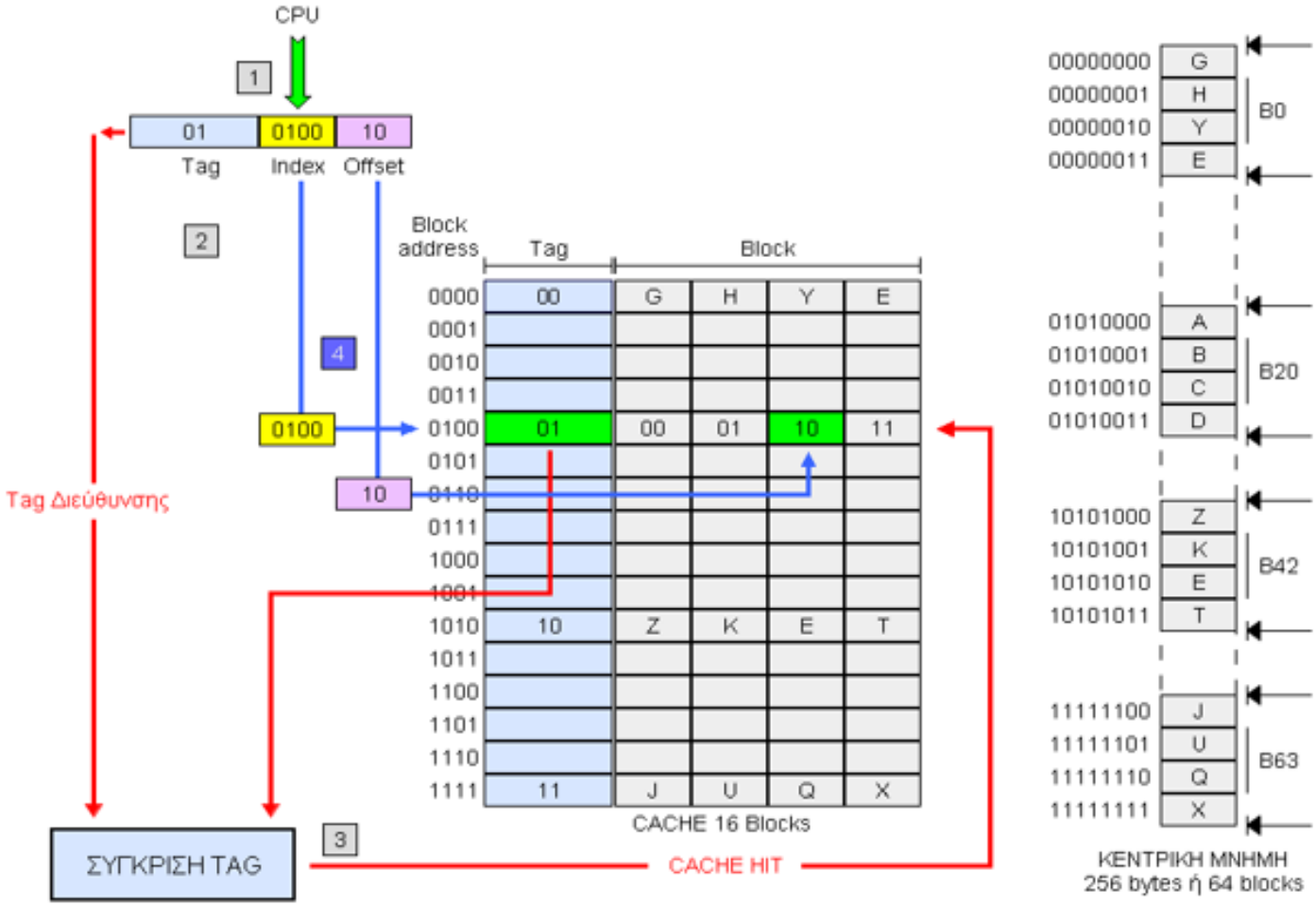
2. Το Tag της διεύθυνσης συγκρίνεται με το Tag του Block της CACHE που αντιστοιχεί στο Index της διεύθυνσης

Άμεση οργάνωση – Cache Hit (3)



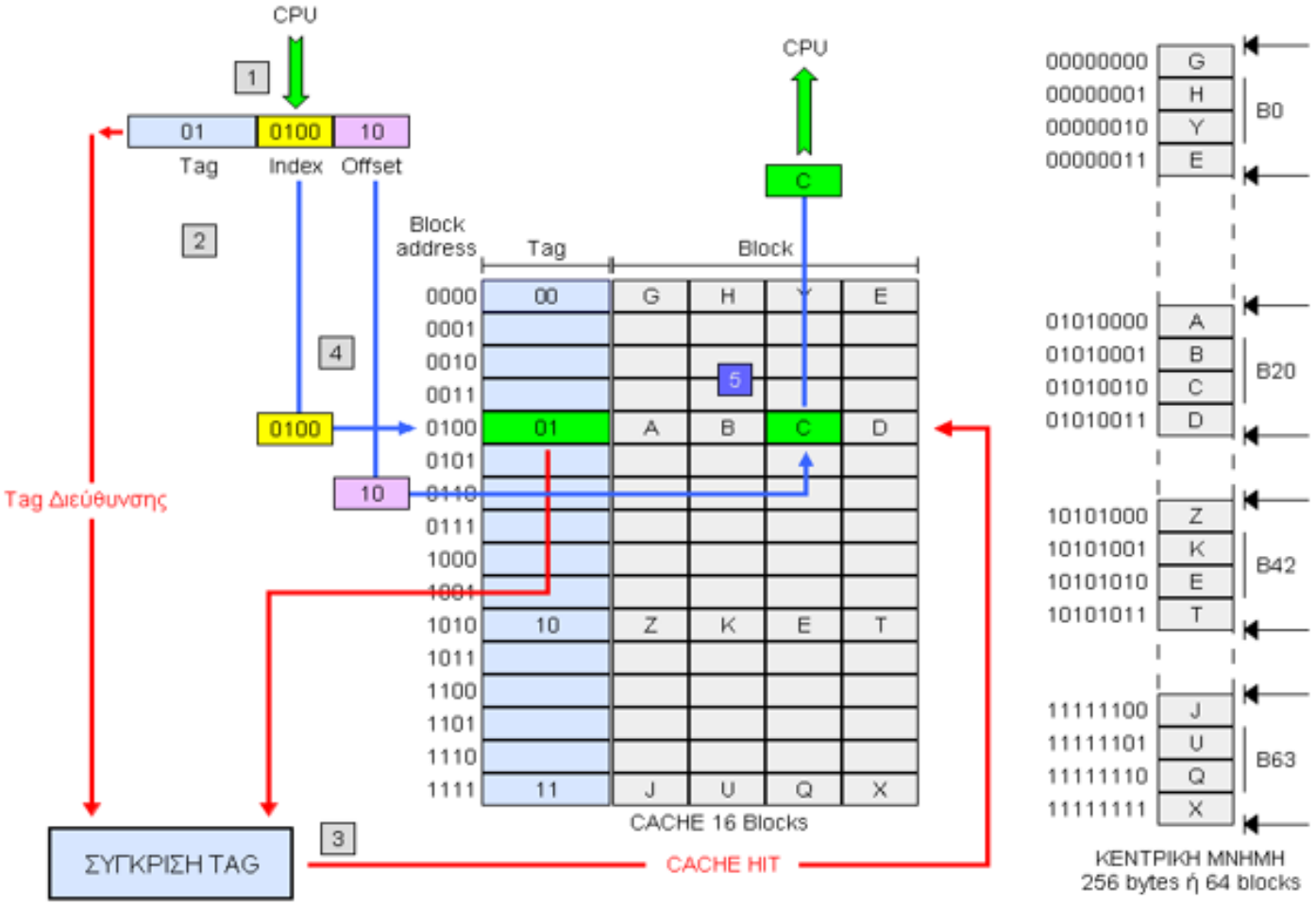
3. Τα Tag συμπίπτουν οπότε έχουμε CACHE HIT

Άμεση οργάνωση – Cache Hit (4)



4. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

Άμεση οργάνωση – Cache Hit (5)

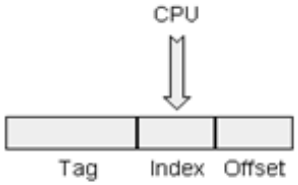


5. Η θέση μνήμης μεταφέρεται στη CPU

Άμεση οργάνωση - Αλγόριθμος Αναζήτησης - miss

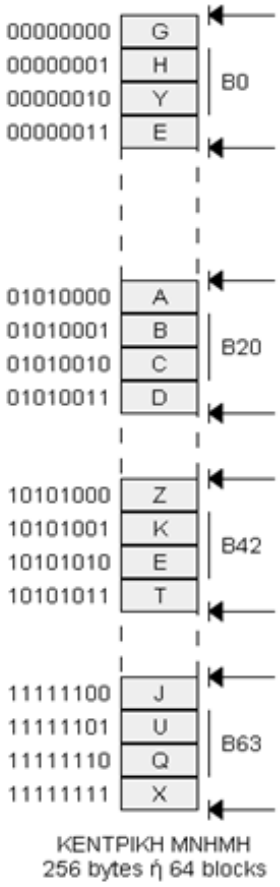
- **Βήμα 0:** Ανάλυση της διεύθυνσης στα πεδία, tag, index, W **(10 1010 10)**.
- **Βήμα 1:** Η index διεύθυνση μας οδηγεί σε μια γραμμή της cache στην οποία, **ενδεχομένως,** να βρίσκεται το περιεχόμενο της διεύθυνση που αναζητούμε.
- **Βήμα 2:** Το tag της διεύθυνσης συγκρίνεται με ο tag της γραμμής της cache στην οποία μας οδήγησε η διεύθυνση index.
- **Βήμα 3:** Αν τα δύο tag ΔΕΝ είναι ίσα τότε έχουμε **cache miss** και η διεύθυνση δεν βρίσκεται στη γραμμή (block) της cache με διεύθυνση index.
- **Βήμα 4:** Η κύρια μνήμη μεταφέρει στη cache και στη line **(1010)** όλο block με διεύθυνση **(10 1010 00)**.
- **Βήμα 5:** Το πεδίο w (offset) βοήθα τον εντοπισμό της θέσης μνήμης **(10 1010 10)** μέσα στο block.
- **Βήμα 6:** Μεταφέρεται έξω από την cache το μέρος του block που έχει διεύθυνση **(10 1010 10)**.

Άμεση οργάνωση – Cache Miss (0)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

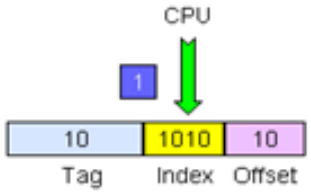
CACHE 16 Blocks



ΚΕΝΤΡΙΚΗ ΜΝΗΜΗ
256 bytes ή 64 blocks

ΣΥΓΚΡΙΣΗ TAG

Άμεση οργάνωση – Cache Miss (1)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	00	R	L	N	H
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

CACHE 16 Blocks

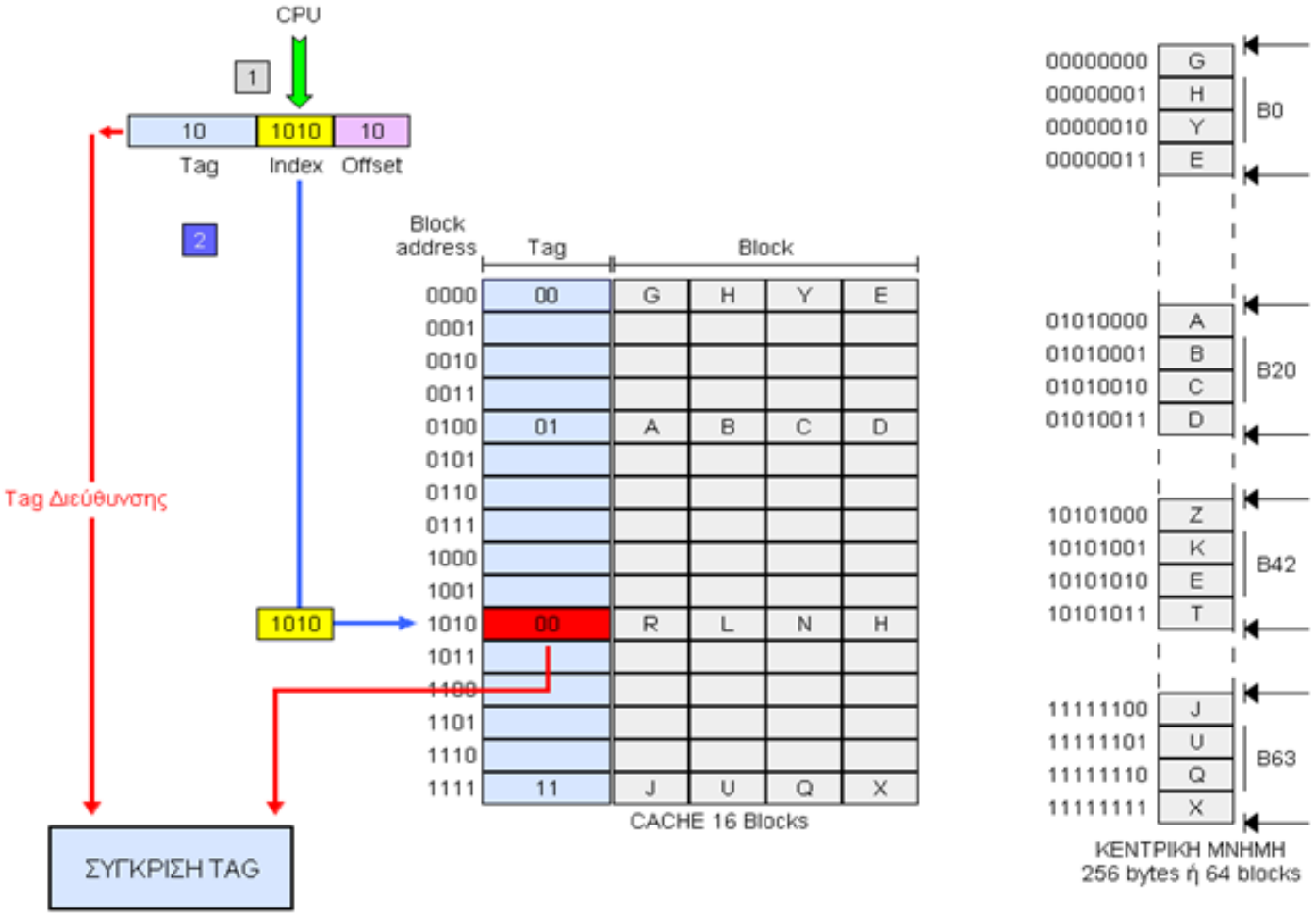
00000000	G	B0
00000001	H	
00000010	Y	
00000011	E	
01010000	A	B20
01010001	B	
01010010	C	
01010011	D	
10101000	Z	B42
10101001	K	
10101010	E	
10101011	T	
11111100	J	B63
11111101	U	
11111110	Q	
11111111	X	

ΚΕΝΤΡΙΚΗ ΜΝΗΜΗ
256 bytes ή 64 blocks

ΣΥΓΚΡΙΣΗ TAG

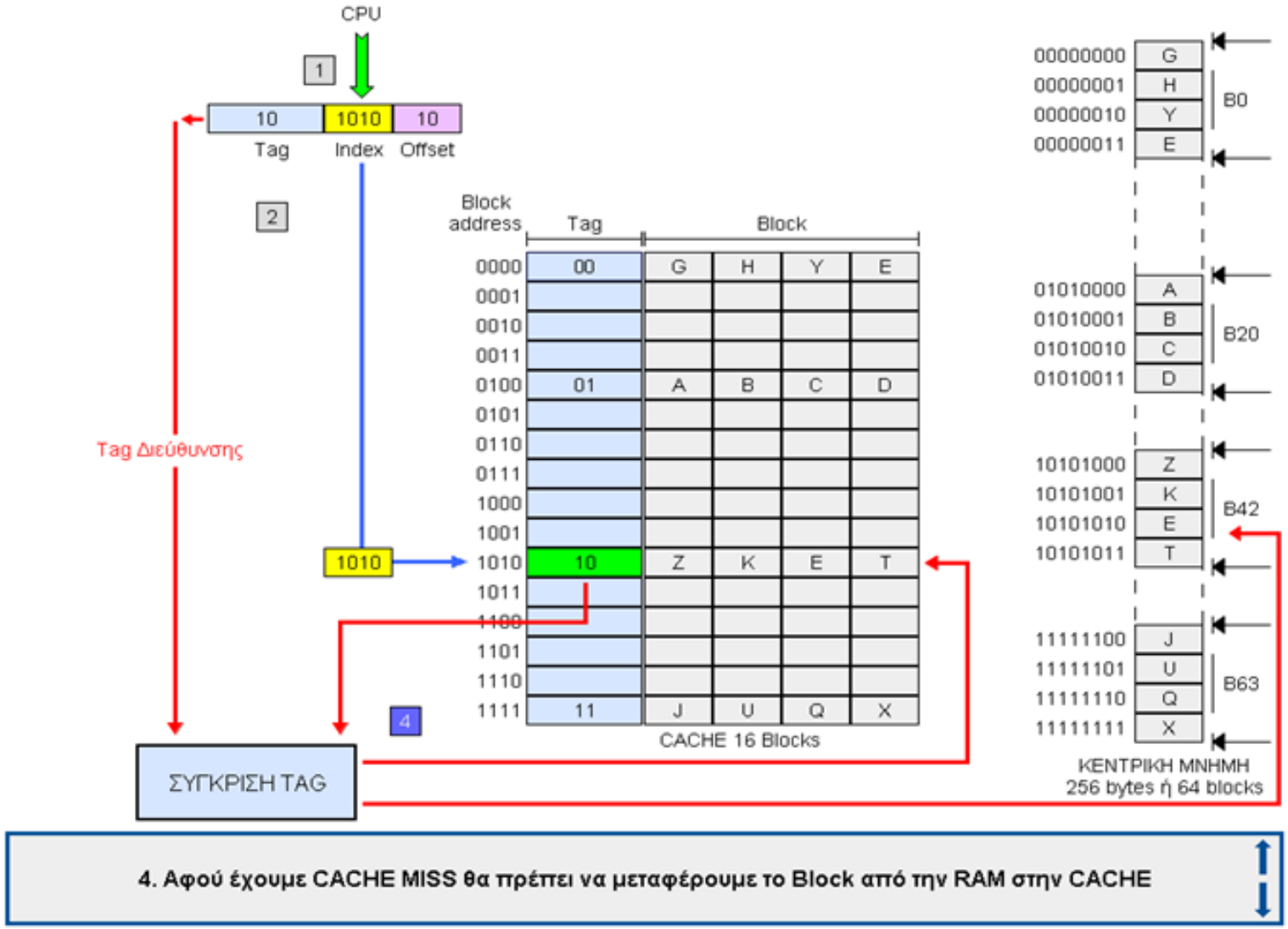
1. Η διεύθυνση εισέρχεται από τη CPU και χωρίζεται σε Tag, Index και Offset

Άμεση οργάνωση – Cache Miss (3)



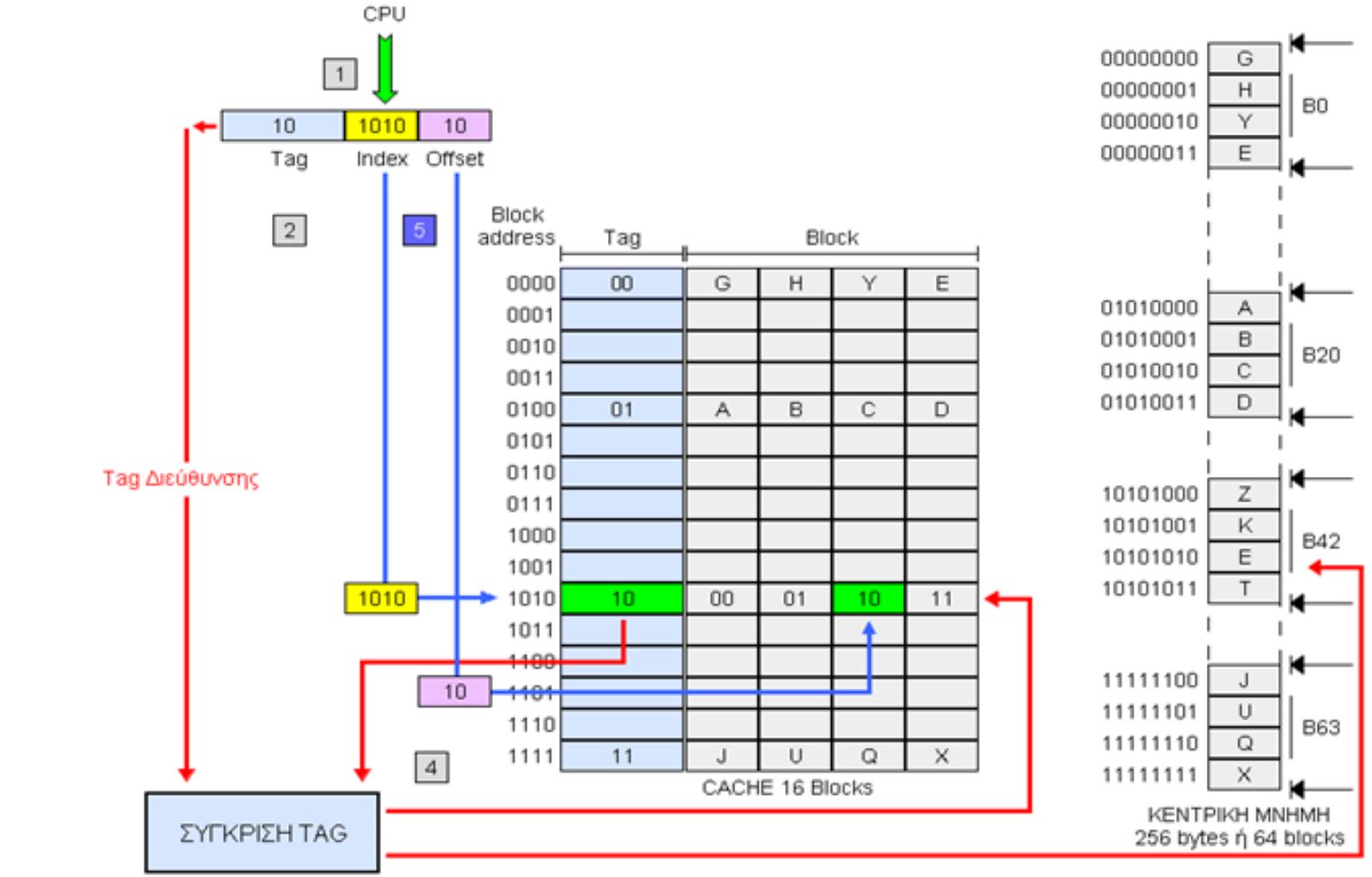
3. Τα Tag είναι διαφορετικά, οπότε έχουμε CACHE MISS

Άμεση οργάνωση – Cache Miss (4)



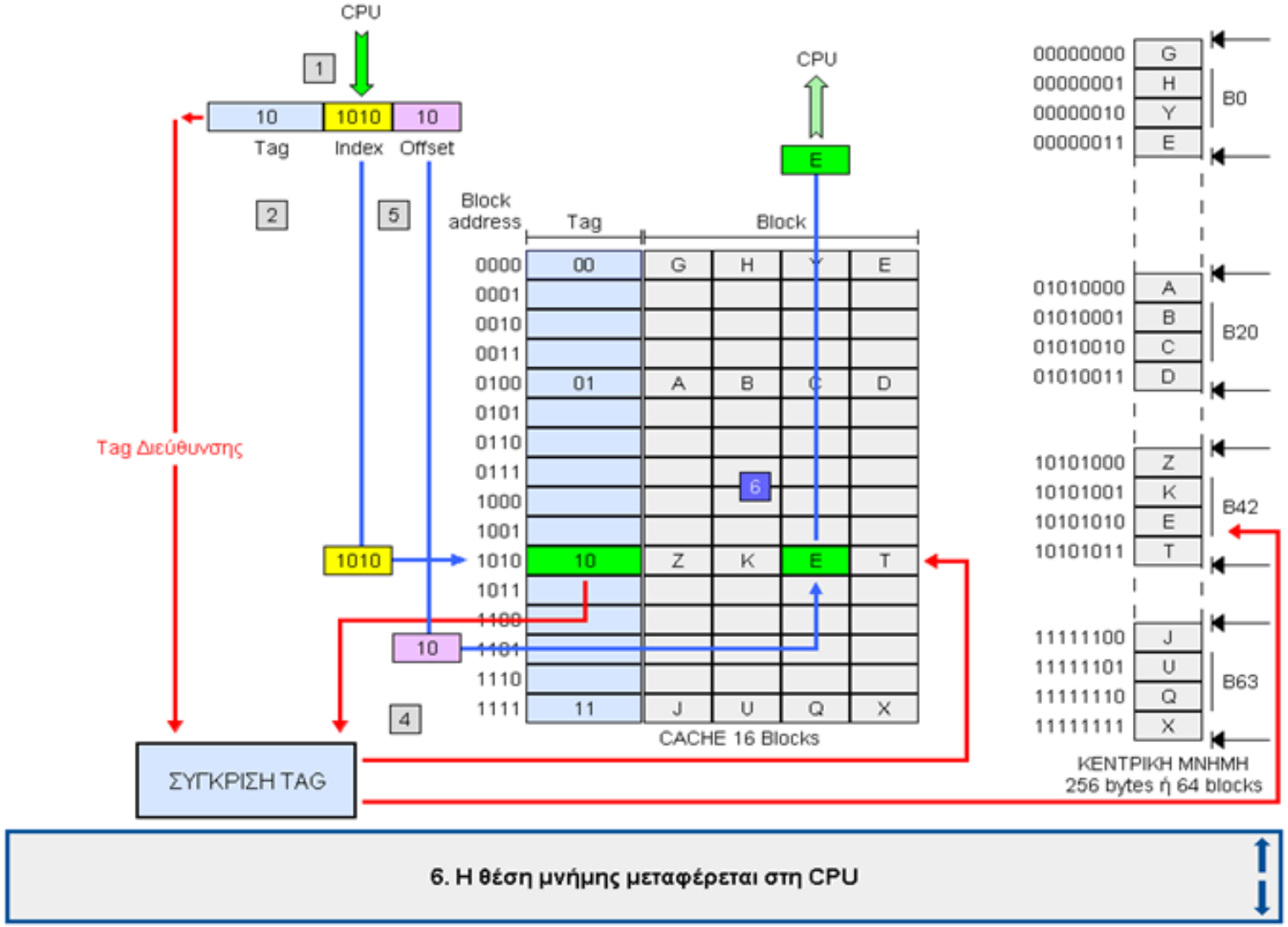
4. Αφού έχουμε CACHE MISS θα πρέπει να μεταφέρουμε το Block από την RAM στην CACHE

Άμεση οργάνωση – Cache Miss (5)



5. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

Άμεση οργάνωση – Cache Miss (6)



Άμεση απεικόνιση – Ανάλυση Διεύθυνσης

Διεύθυνση μνήμης:

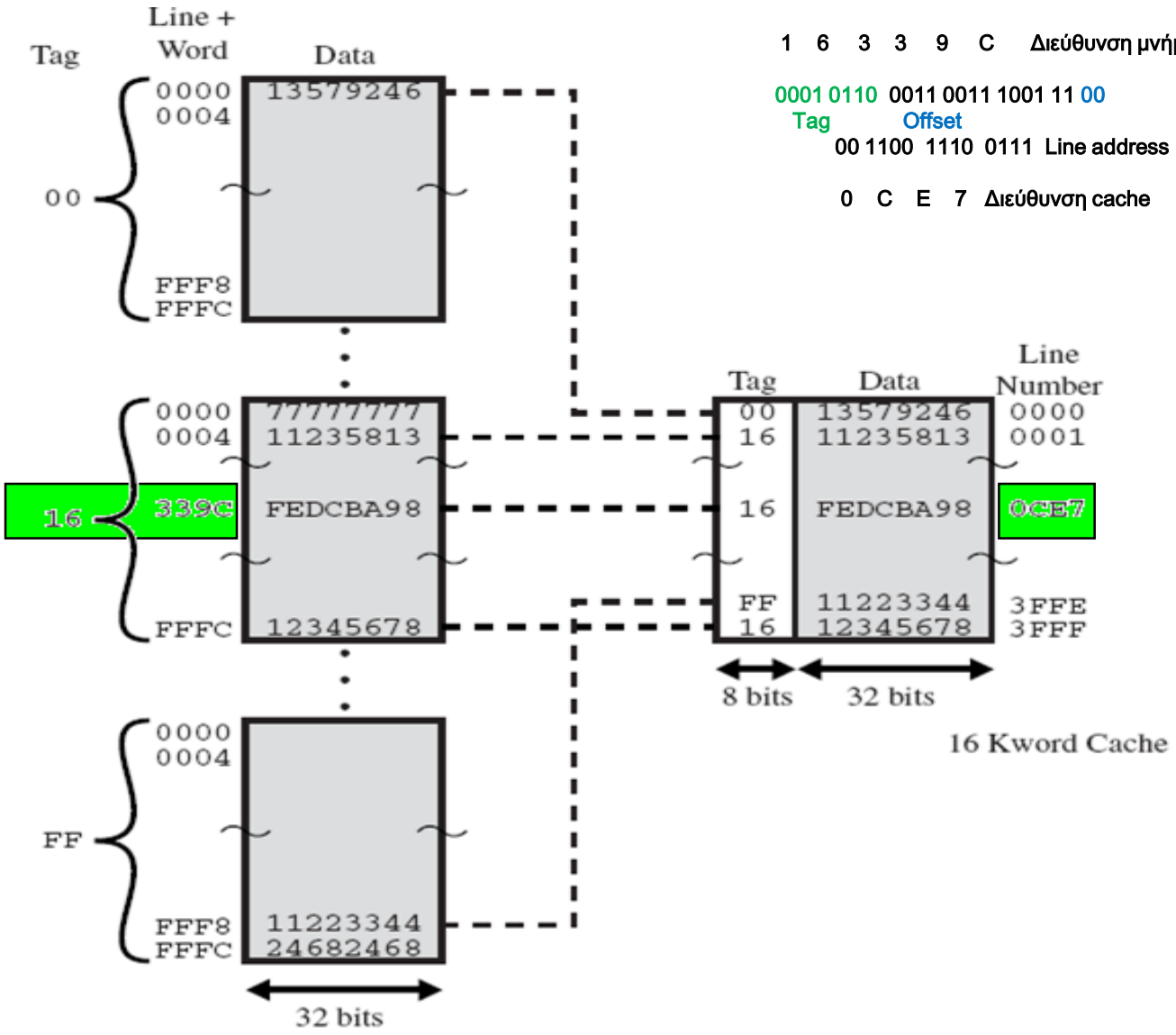
1 6 3 3 9 C

Δυαδική μορφή:

00 1100 1110 0111 Line address

0 C E 7 (Διεύθυνση cache)

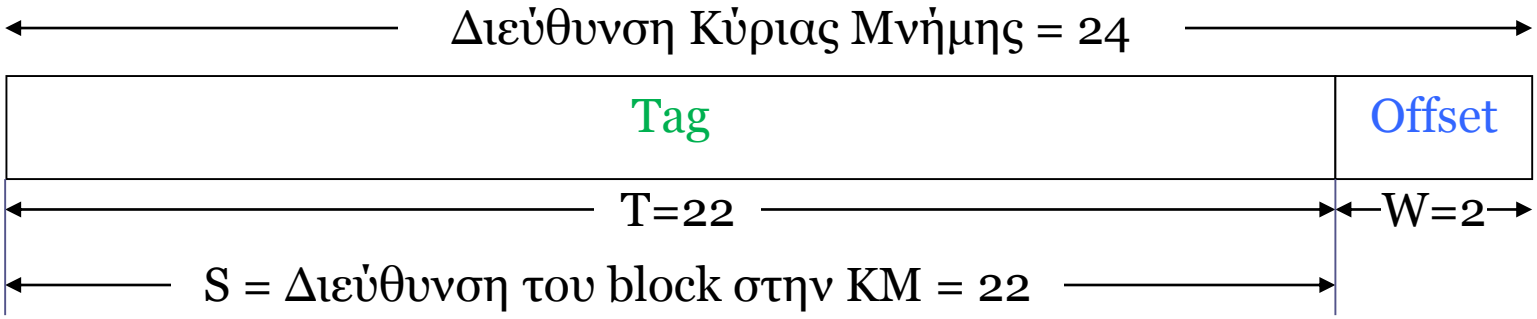
Άμεση απεικόνιση - Παράδειγμα



Πλήρως συσχετιστική cache (fully associative)

- **Βασική αρχή:** Κάθε block της κεντρικής μνήμης τοποθετείται στη πρώτη κενή θέση της cache.
- Αν η cache είναι γεμάτη τότε τοποθετείται στην κενή θέση που δημιουργεί ο αλγόριθμος αντικατάστασης της cache.
- Συνάρτηση απεικόνισης: δεν υπάρχει.

Πλήρως Συσχετιστική - Διεύθυνση μνήμης

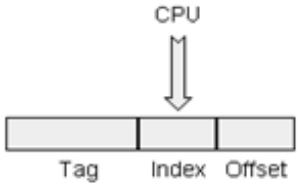


- Offset : Η διεύθυνση ενός byte μέσα στο block
- Index : Δεν υπάρχει
- Tag : Ετικέτα αναγνώρισης (μέρος της διεύθυνσης)
- Μέγεθος Block: 4 bytes = 2^w bytes → W = 2
- Μέγεθος Μνήμης: 16 Mbytes = 2^{24} = 2^{s+w}
- Μήκος διεύθυνσης Κ.Μ.: (S + W) bits = 24 bits → S = 22
- Πλήθος blocks στη μνήμη: $2^s = 2^{22}$ = 4 Mblocks
- Μέγεθος cache: 64 Kbytes = 2^{16} bytes
- Πλήθος blocks στην cache: $2^D = 2^{16} / 2^2 = 2^{14}$ → D = 14
- Μέγεθος tag: T = S−D bits = 8

Πλήρως συσχετιστική – Αλγόριθμος προσπέλασης-hit

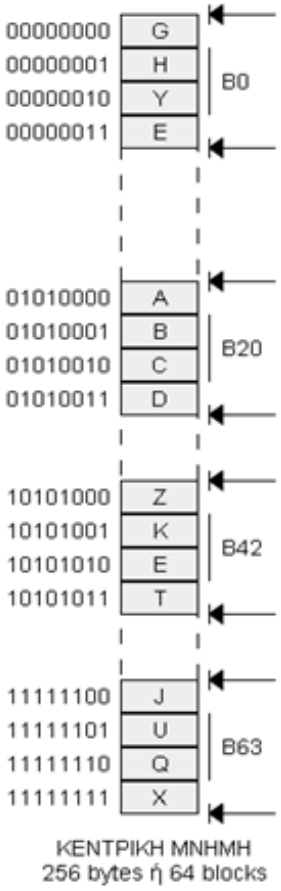
- **Βήμα 0:** Ανάλυση της διεύθυνσης στα πεδία, tag, και w **(010100 10)**.
- **Βήμα 1-2:** Το tag της διεύθυνσης συγκρίνεται με τα tag όλων των γραμμών της cache.
- **Βήμα 3:** Αν κάποιο tag της cache είναι ίσο με το tag της διεύθυνσης τότε έχουμε **cache hit** και η γραμμή της cache η οποία έχει το ίδιο tag με αυτό της διεύθυνσης περιέχει την διεύθυνση.
- **Βήμα 4:** Το πεδίο w (offset) βοήθα τον εντοπισμό της θέσης μνήμης **(010100 10)** μέσα στο block.
- **Βήμα 5:** Μεταφέρεται έξω από την cache το μέρος του block που έχει διεύθυνση **(010100 10)**.

Πλήρως Συσχετιστική – Cache Hit (0)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

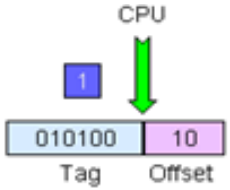
CACHE 16 Blocks



ΚΕΝΤΡΙΚΗ ΜΝΗΜΗ
256 bytes ή 64 blocks

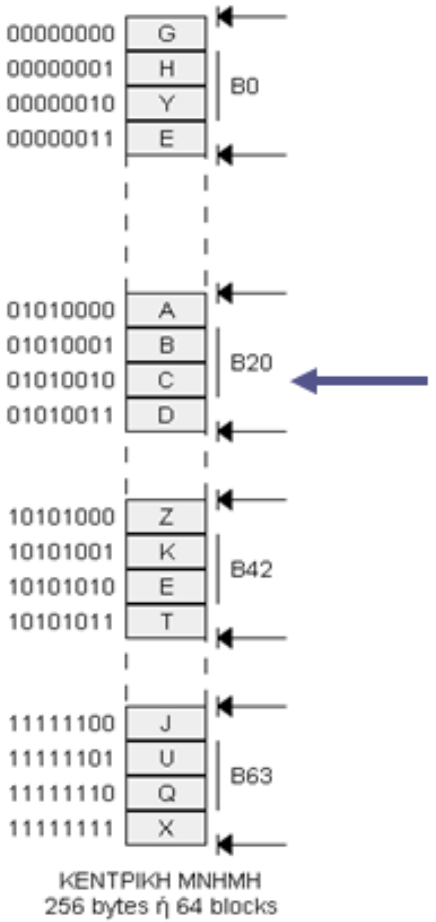
ΣΥΓΚΡΙΣΗ TAG

Πλήρως Συσχετιστική – Cache Hit (1)



Block address	Tag	Block			
0000	101010	Z	K	E	T
0001					
0010					
0011					
0100	111111	J	U	Q	X
0101					
0110					
0111	010100	A	B	C	D
1000					
1001					
1010					
1011					
1100	000000	G	H	Y	E
1101					
1110					
1111					

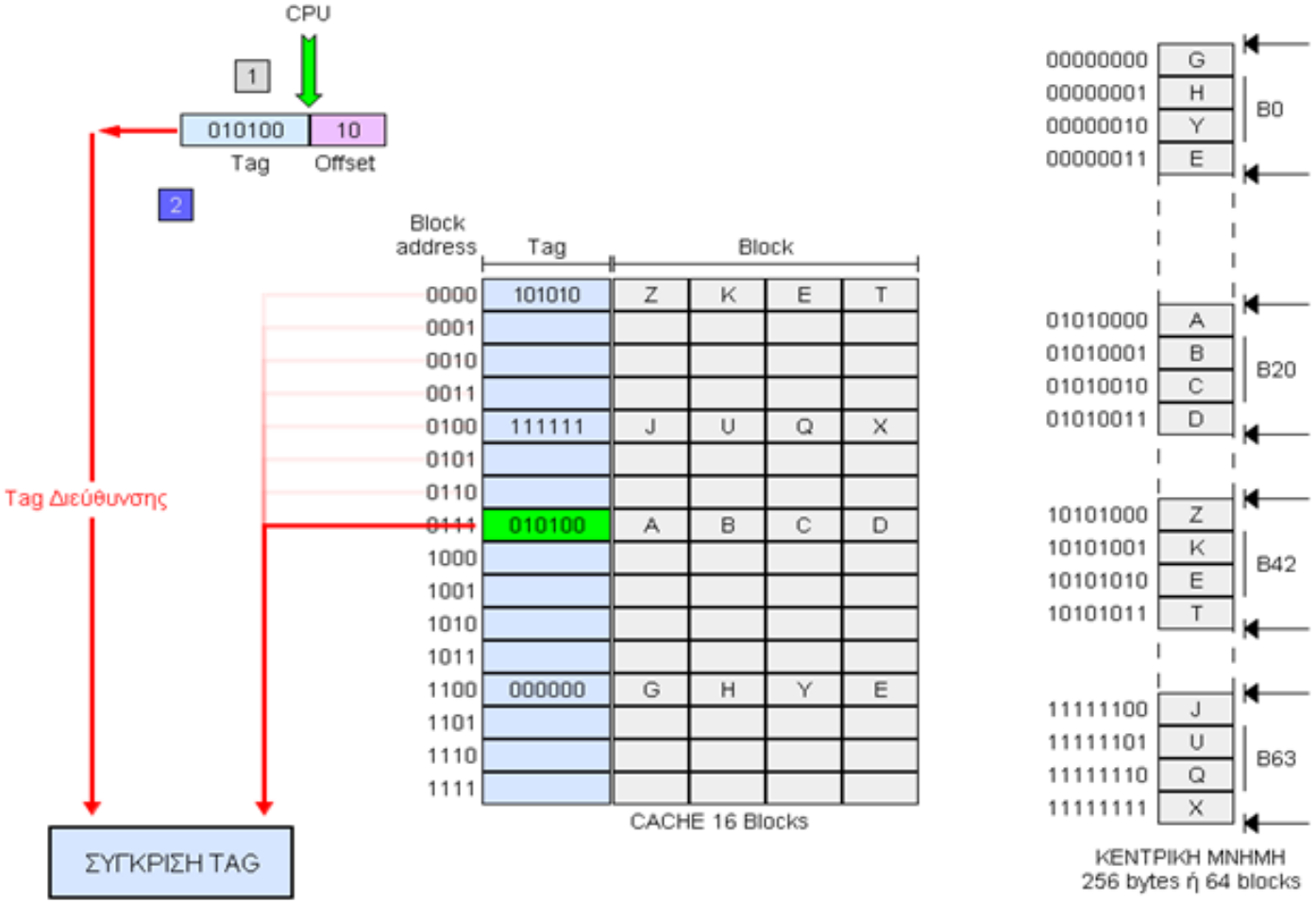
CACHE 16 Blocks



ΣΥΓΚΡΙΣΗ TAG

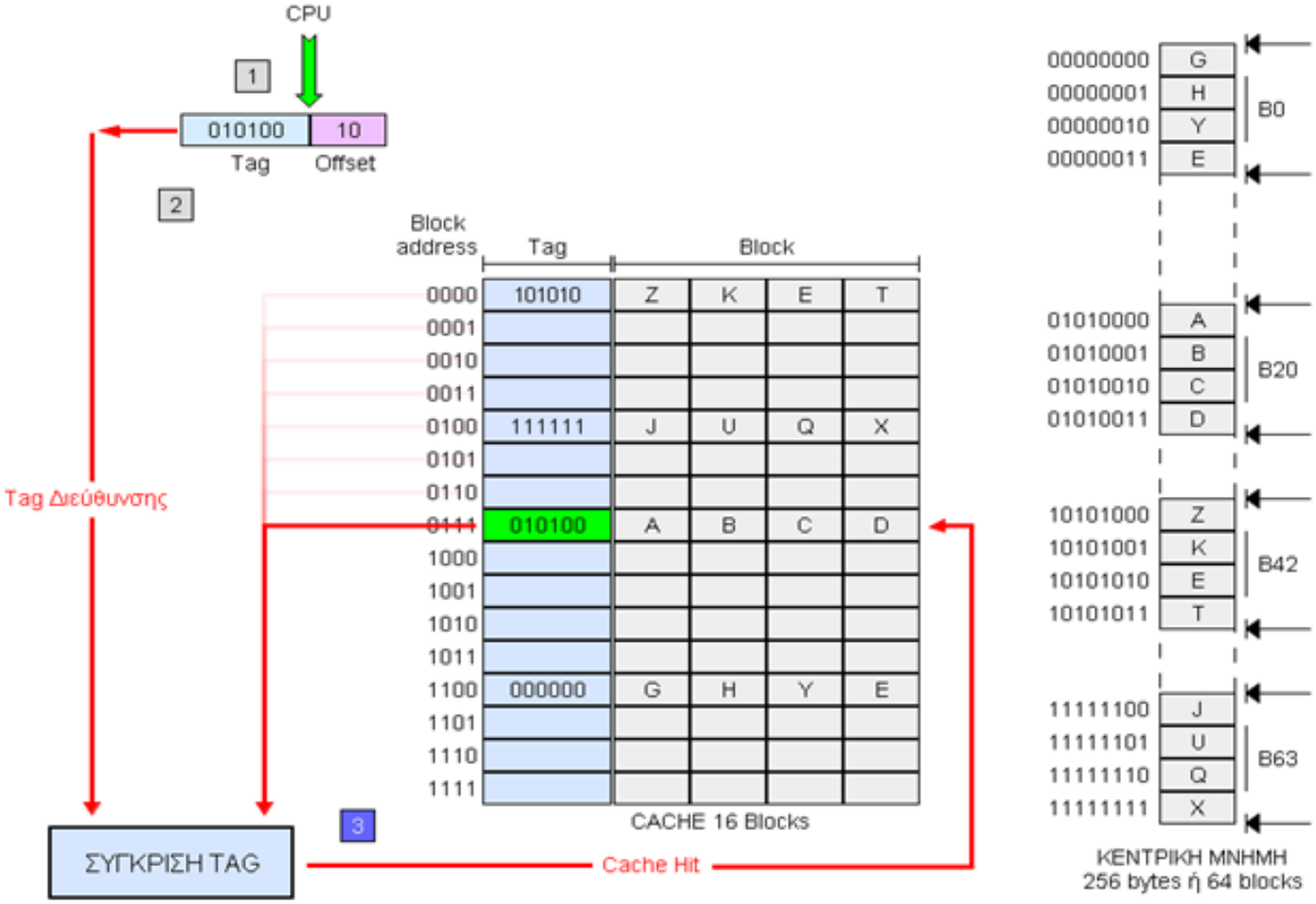
1. Η διεύθυνση εισέρχεται από τη CPU και χωρίζεται σε Tag και Offset

Πλήρως Συσχετιστική – Cache Hit (2)



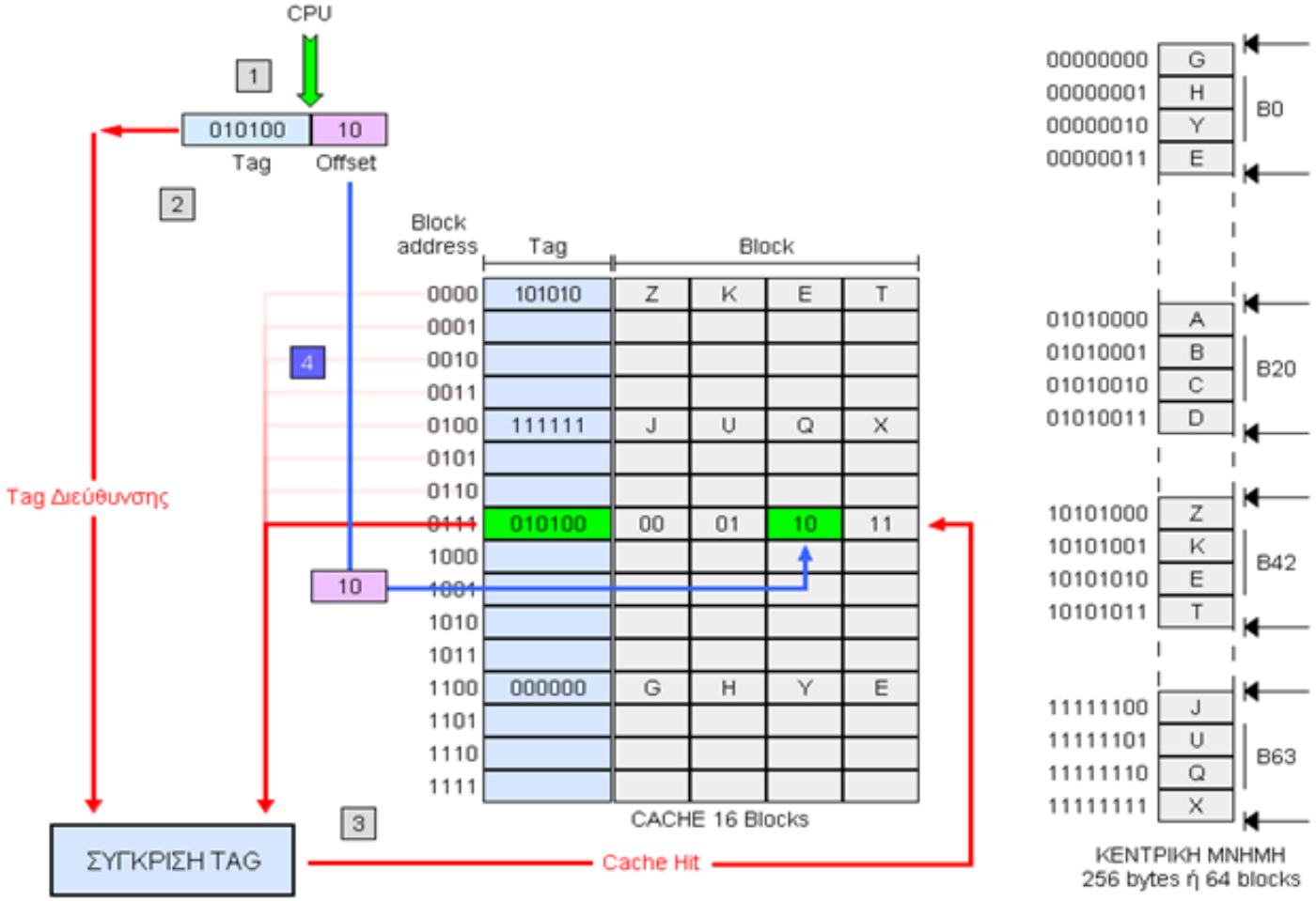
2. Το Tag της διεύθυνσης συγκρίνεται διαδοχικά με όλα τα Tag της Cache

Πλήρως Συσχετιστική – Cache Hit (3)



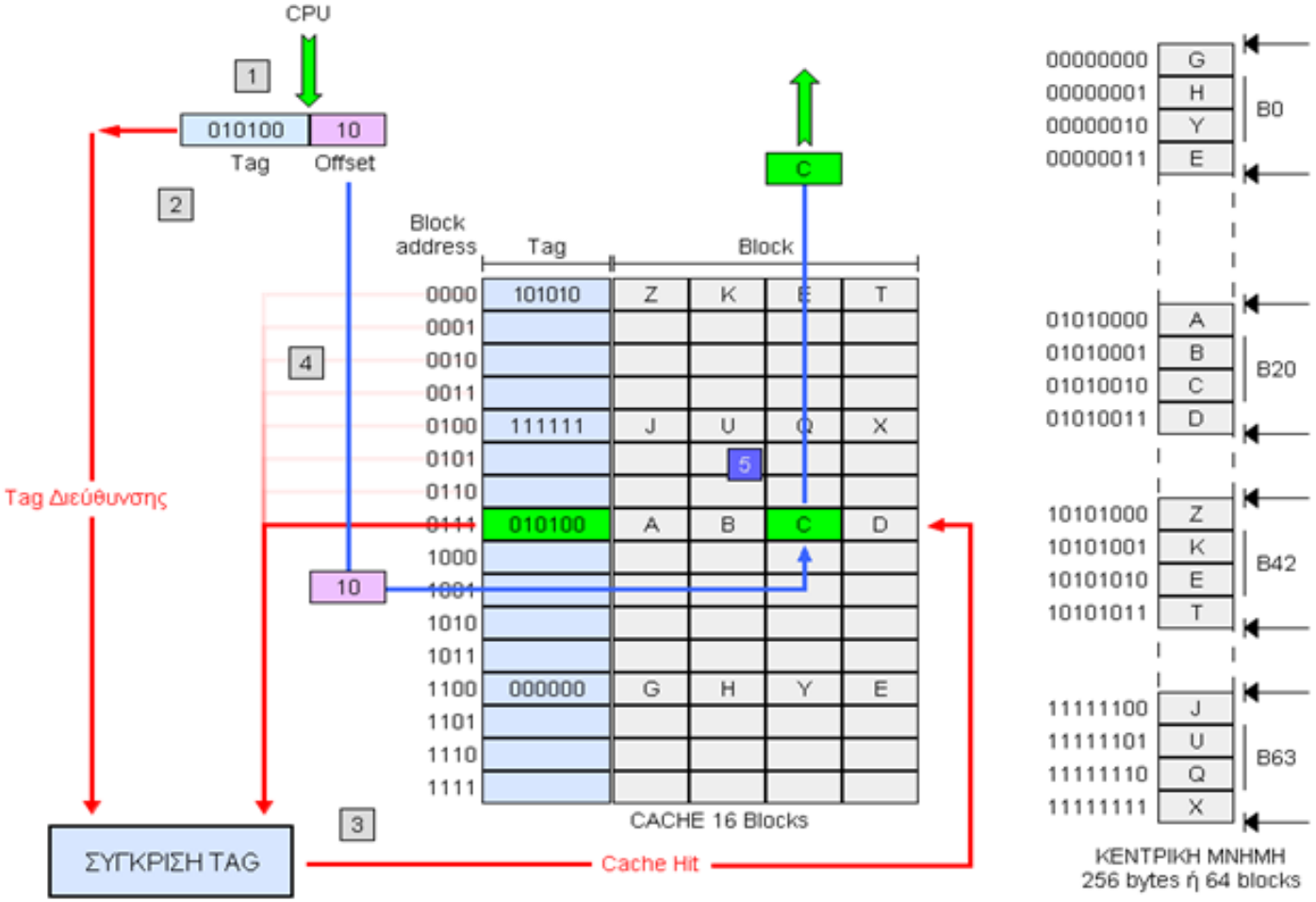
3. Ένα από τα Tag της Cache συμπίπτει με αυτό της διεύθυνσης οπότε έχουμε CACHE HIT

Πλήρως Συσχετιστική – Cache Hit (4)



4. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

Πλήρως Συσχετιστική – Cache Hit (5)

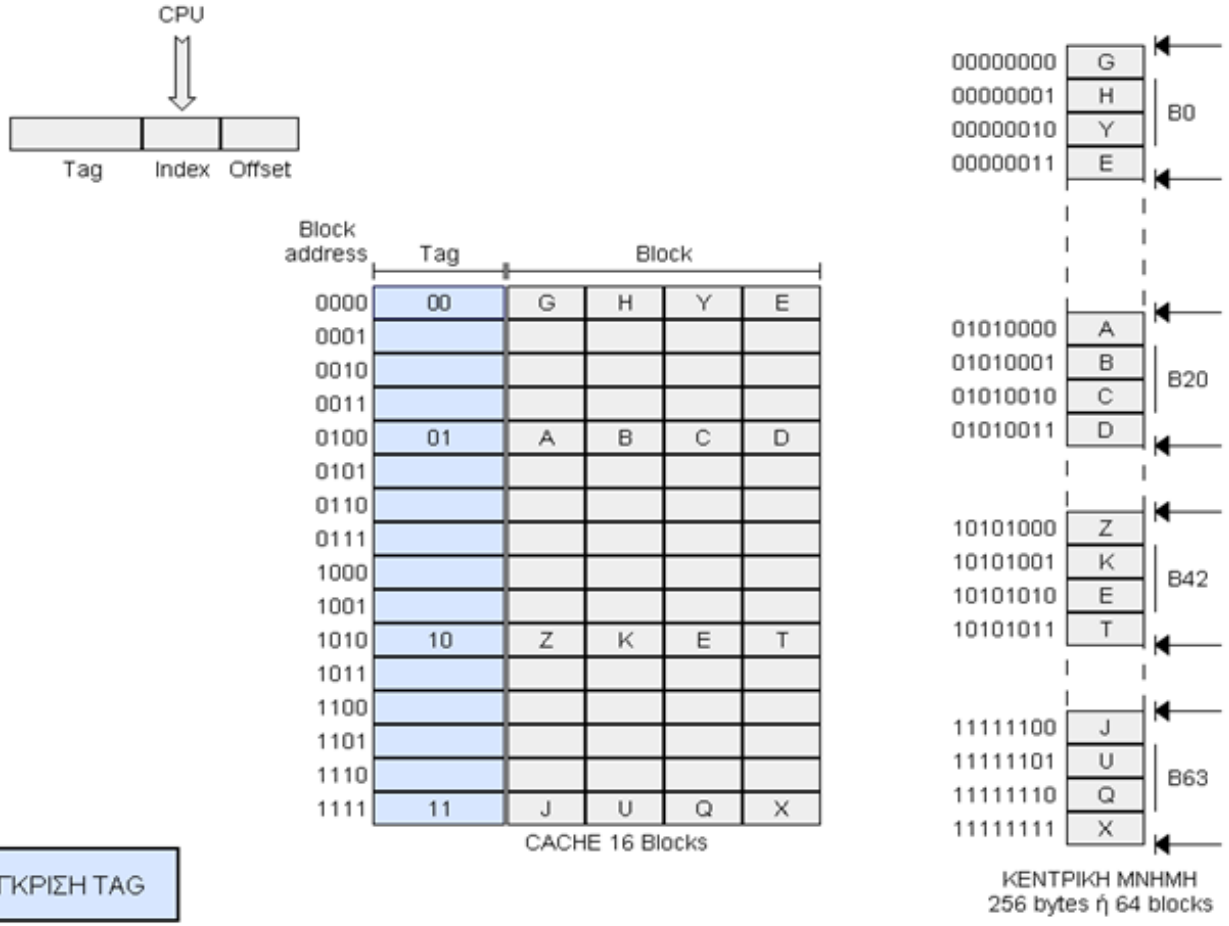


5. Η θέση μνήμης μεταφέρεται στη CPU

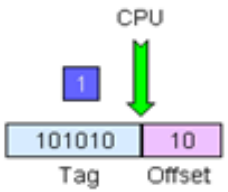
Πλήρως συσχετιστική – Αλγόριθμος προσπέλασης - miss

- **Βήμα 0:** Ανάλυση της διεύθυνσης στα πεδία, tag, w (101010 10).
- **Βήμα 1-2:** Το tag της διεύθυνσης συγκρίνεται με τα tag όλων των γραμμών της cache.
- **Βήμα 3:** Αν κανένα από tag της cache δεν είναι ίσο με το tag της διεύθυνσης, τότε έχουμε cache miss.
- **Βήμα 4:** Ο αλγόριθμος αντικατάστασης αποφασίζει ποιο από τα block της cache θα απομακρυνθεί (έστω ότι είναι το block της cache με διεύθυνση 1000).
- **Βήμα 5:** Η κ. μνήμη μεταφέρει στην cache και στη line (0100) όλο το block με διεύθυνση (101010 00).
- **Βήμα 6:** Το πεδίο w (offset) βοήθα τον εντοπισμό της θέσης μνήμης (101010 10) μέσα στο block.
- **Βήμα 7:** Μεταφέρεται έξω από την cache το μέρος του block που έχει διεύθυνση (101010 10).

Πλήρως Συσχετιστική – Cache Miss (0)

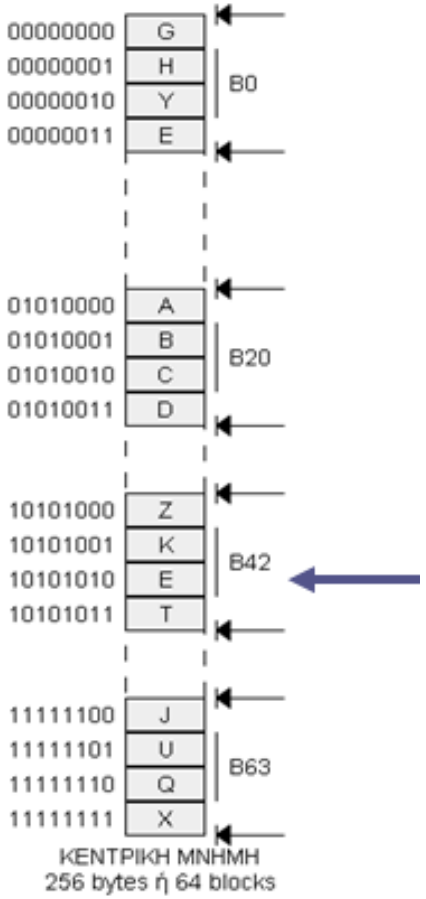


Πλήρως Συσχετιστική – Cache Miss (1)



Block address	Tag	Block			
0000	001110	T	M	O	F
0001					
0010					
0011					
0100	111111	J	U	Q	X
0101					
0110					
0111	010100	A	B	C	D
1000					
1001					
1010					
1011					
1100	000000	G	H	Y	E
1101					
1110					
1111					

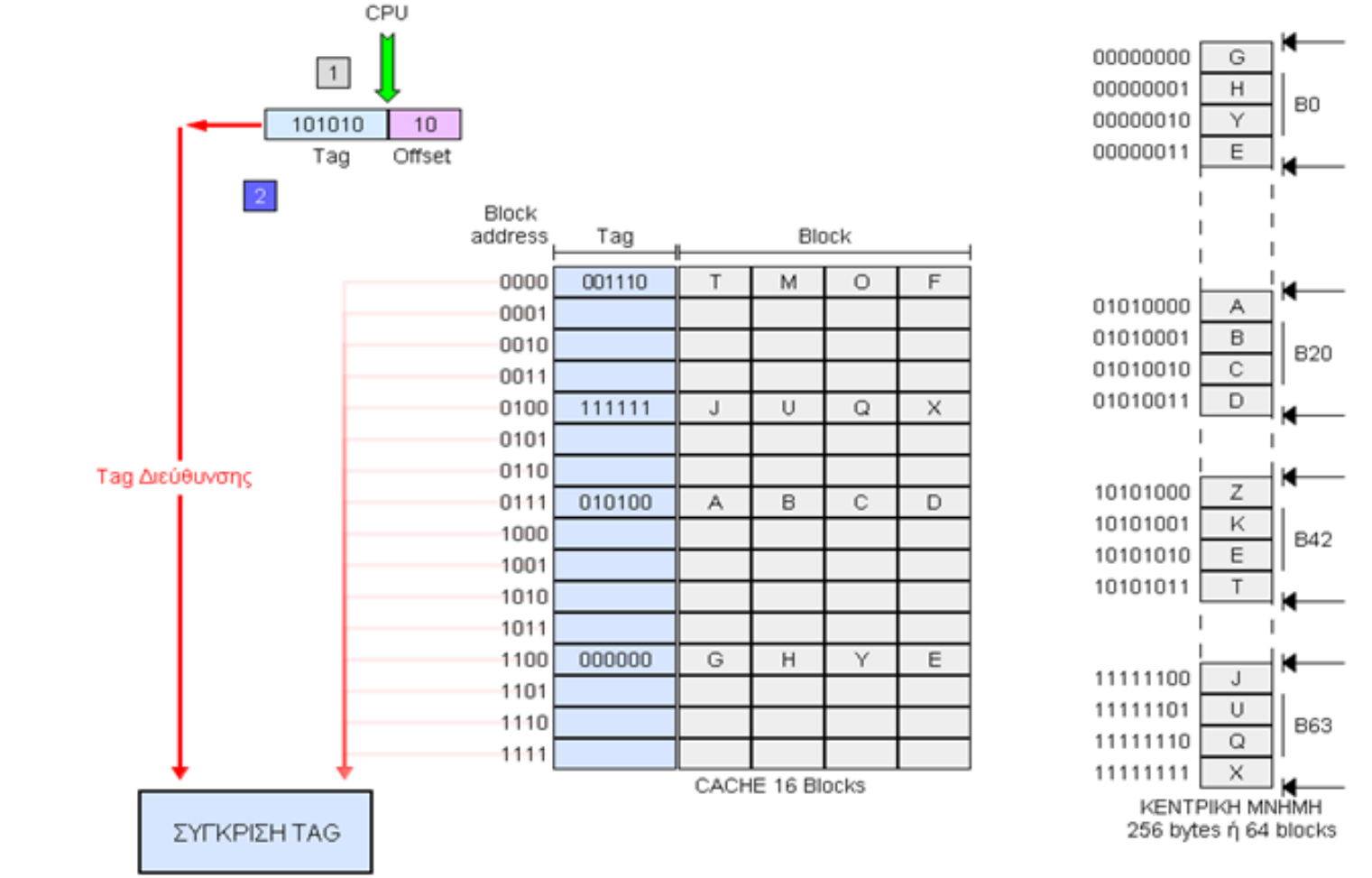
CACHE 16 Blocks



ΣΥΓΚΡΙΣΗ TAG

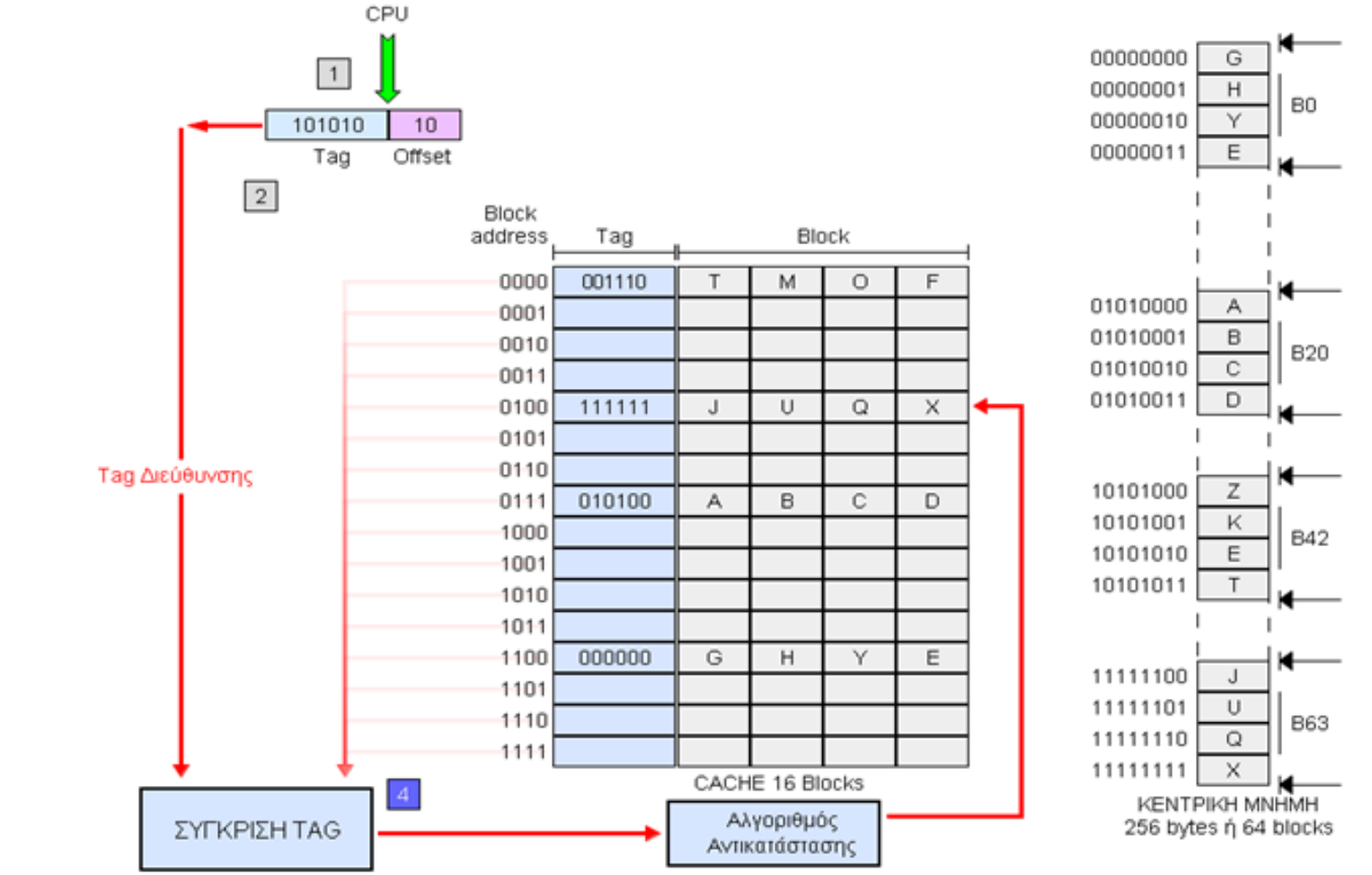
1. Η διεύθυνση εισέρχεται από τη CPU και χωρίζεται σε Tag και OffSet

Πλήρως Συσχετιστική – Cache Miss (3)



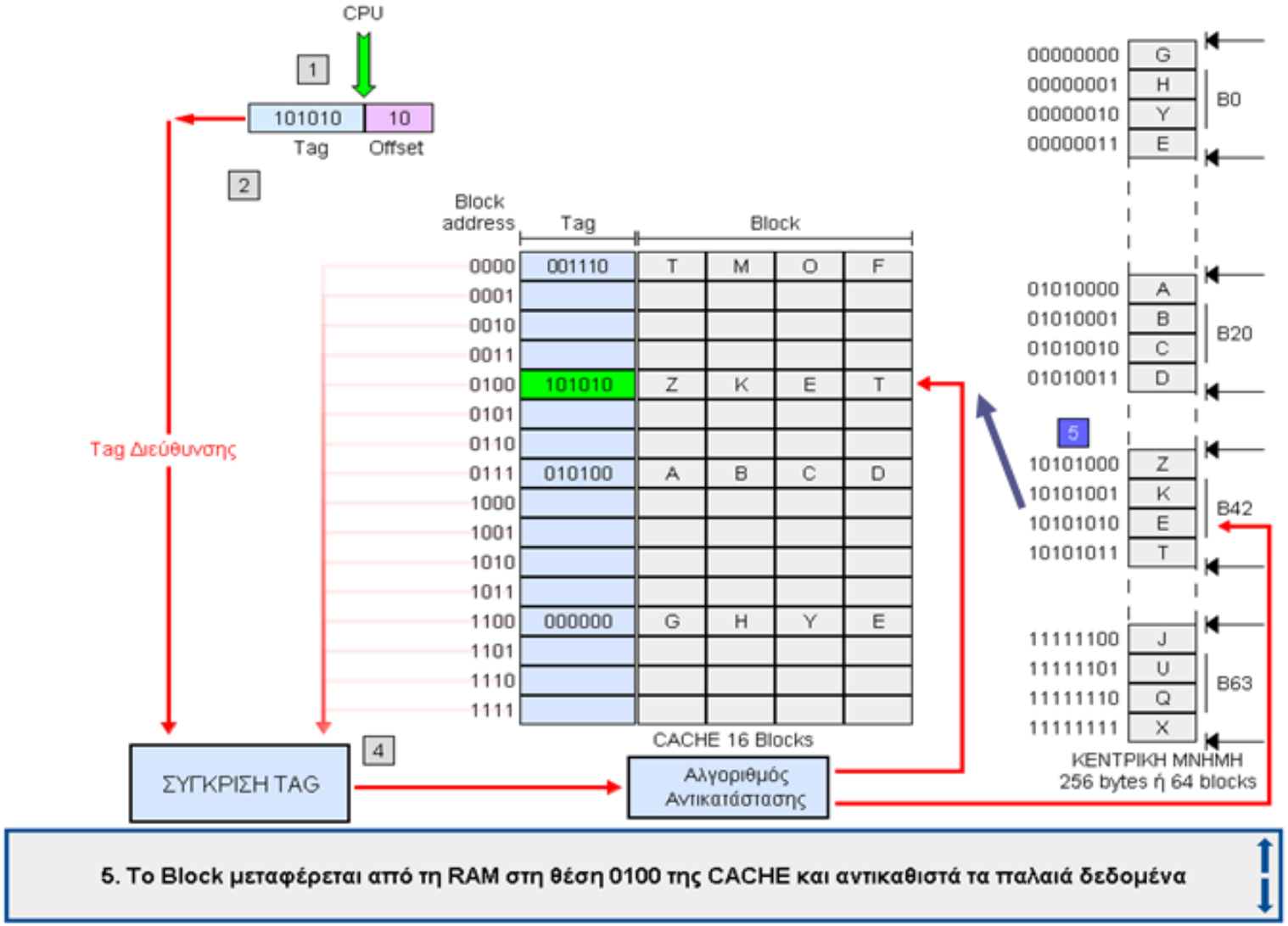
3. Το Tag δεν βρέθηκε στην CACHE. Οπότε έχουμε CACHE MISS

Πλήρως Συσχετιστική – Cache Miss (4)

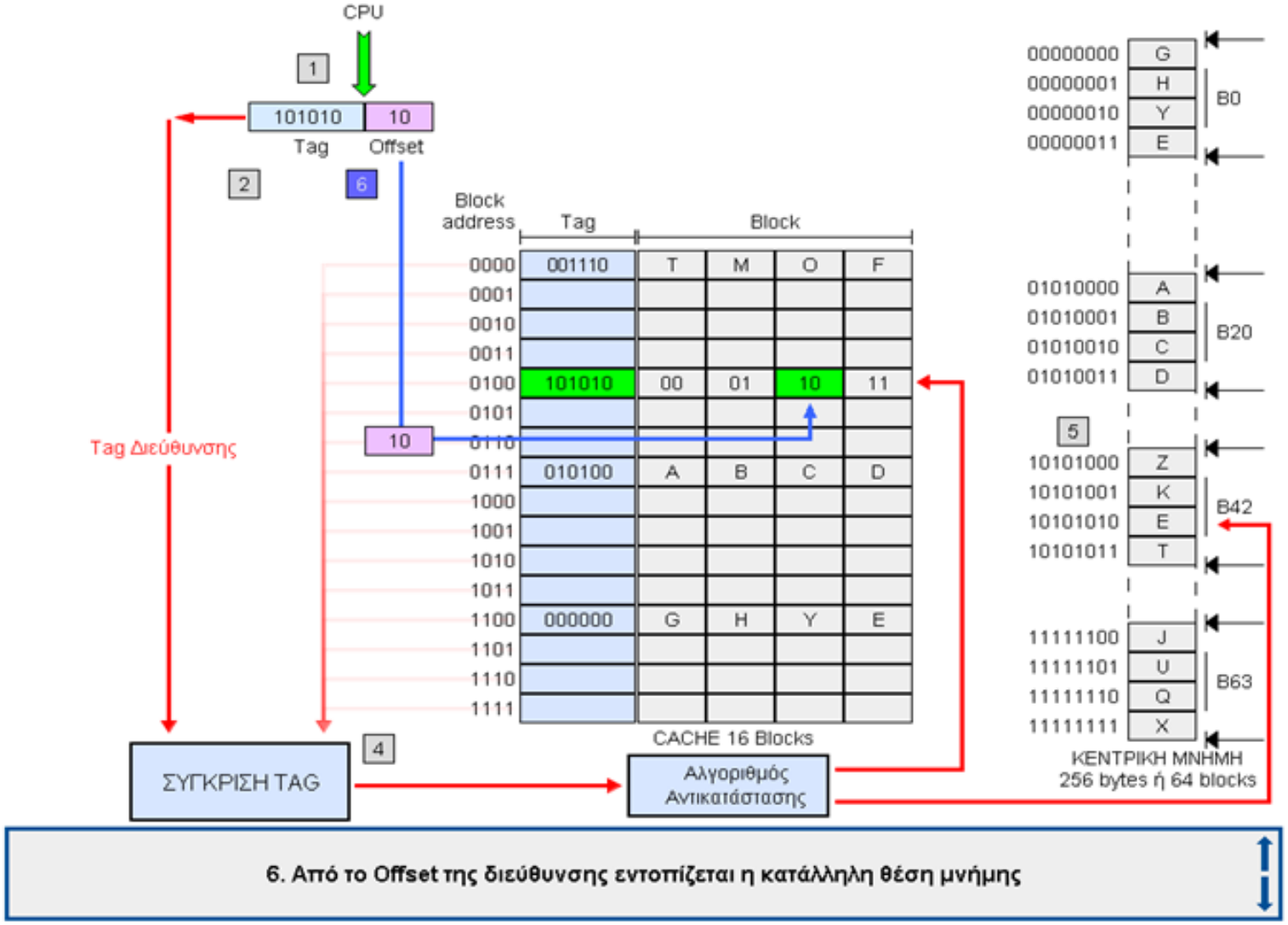


4. Ο Αλγόριθμος Αντικατάστασης επιλέγει το block της Cache που πρέπει να αντικατασταθεί (0100)

Πλήρως Συσχετιστική – Cache Miss (5)

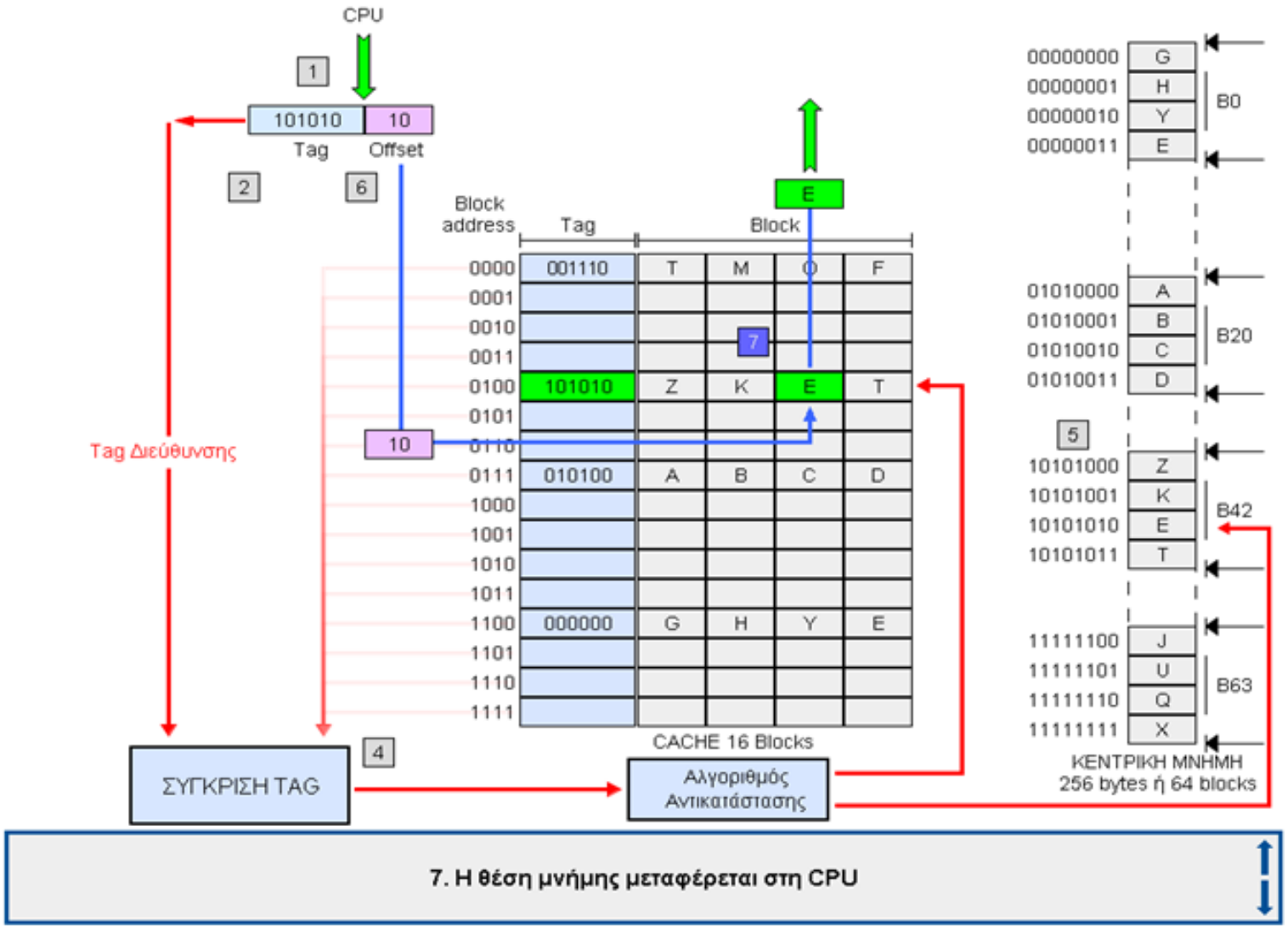


Πλήρως Συσχετιστική – Cache Miss (6)



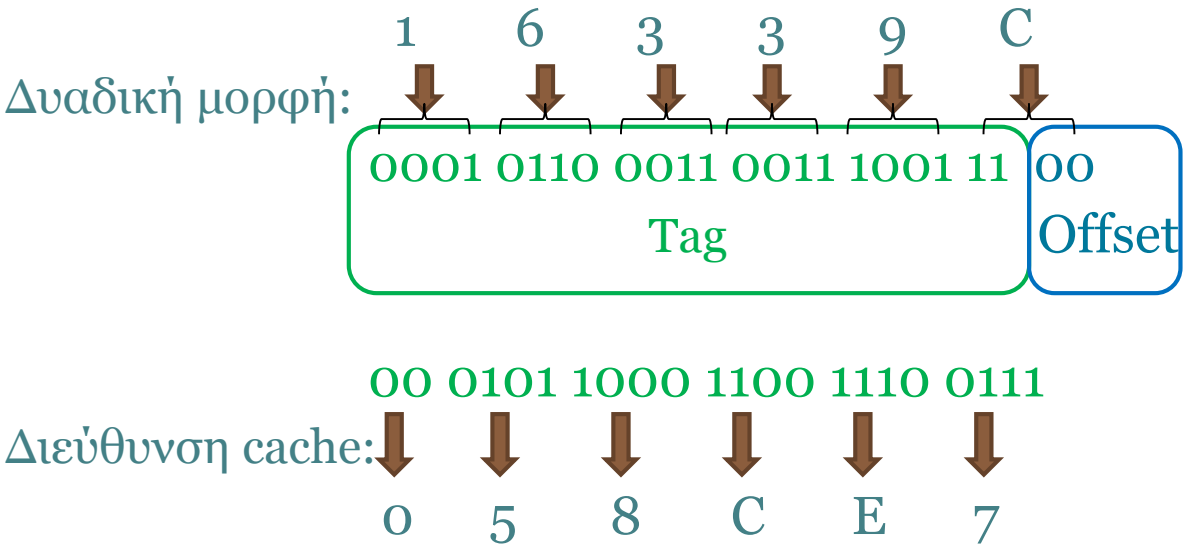
6. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

Πλήρως Συσχετιστική – Cache Miss (7)



Πλήρως Συσχετιστική – Ανάλυση Διεύθυνσης

Διεύθυνση μνήμης:

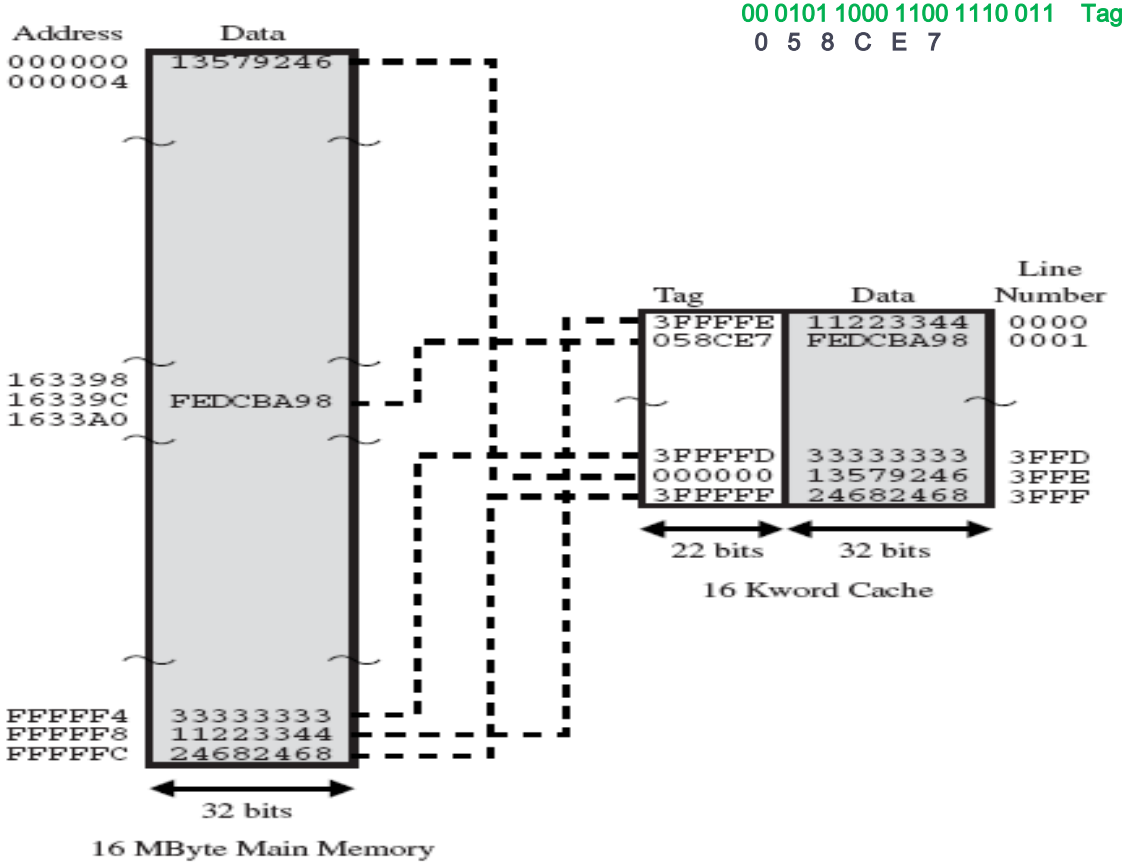


Πλήρως Συσχετιστική – παράδειγμα

1 6 3 3 9 C (Διεύθυνση μνήμης)

0001 0110 0011 0011 1001 11 00
Offset

00 0101 1000 1100 1110 011 Tag
0 5 8 C E 7



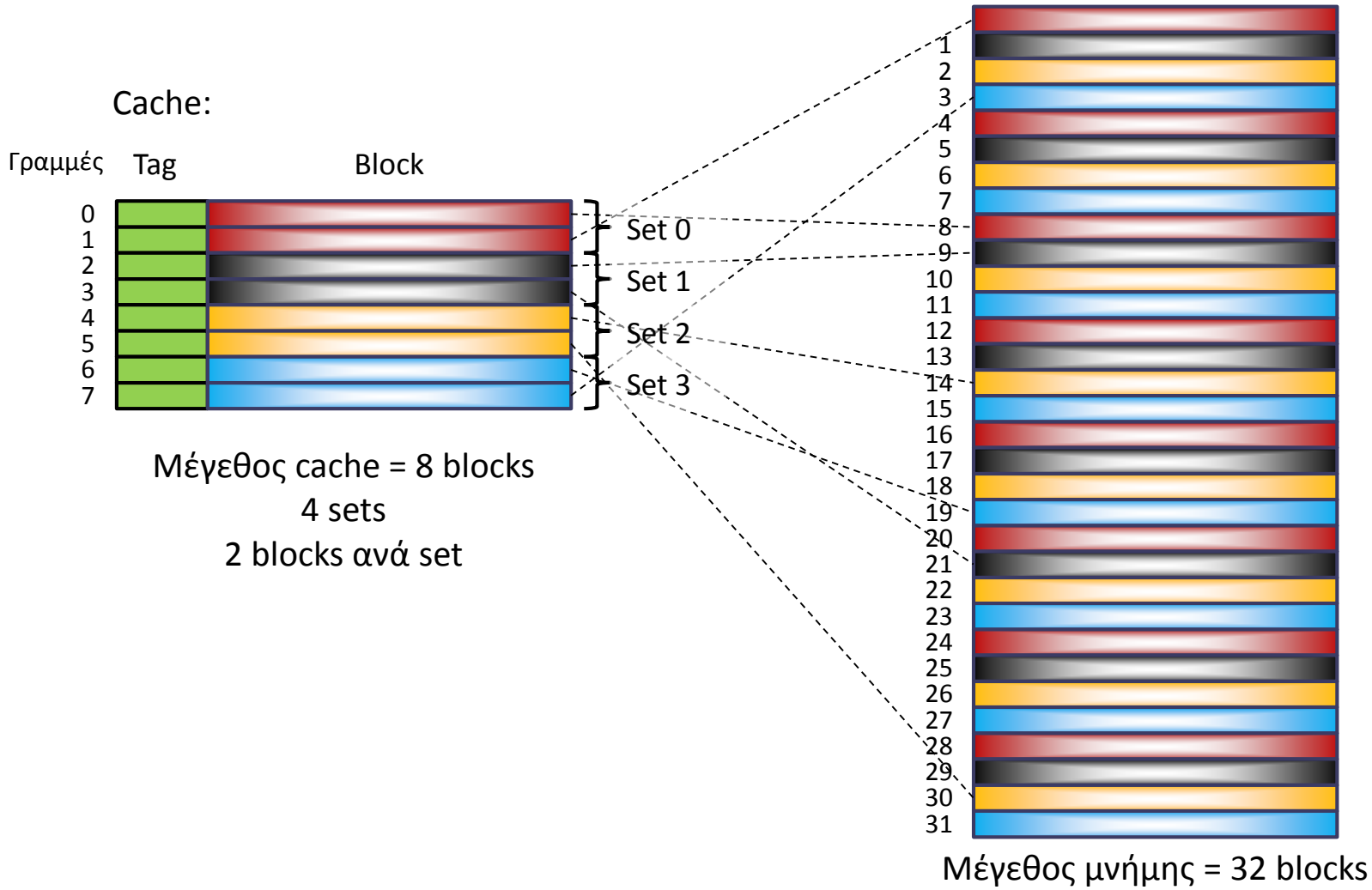
Τμηματικά συσχετική cache (Set Associative)

- **Υπόθεση:** Η cache διαιρείται σε n sets των B blocks.
- Πλήθος blocks της cache: m = n*B.
- Ένα block τοποθετείται οπουδήποτε μέσα σε ένα set.
- Σε ποιο set θα τοποθετηθεί το αποφασίζει η συνάρτηση απεικόνισης.

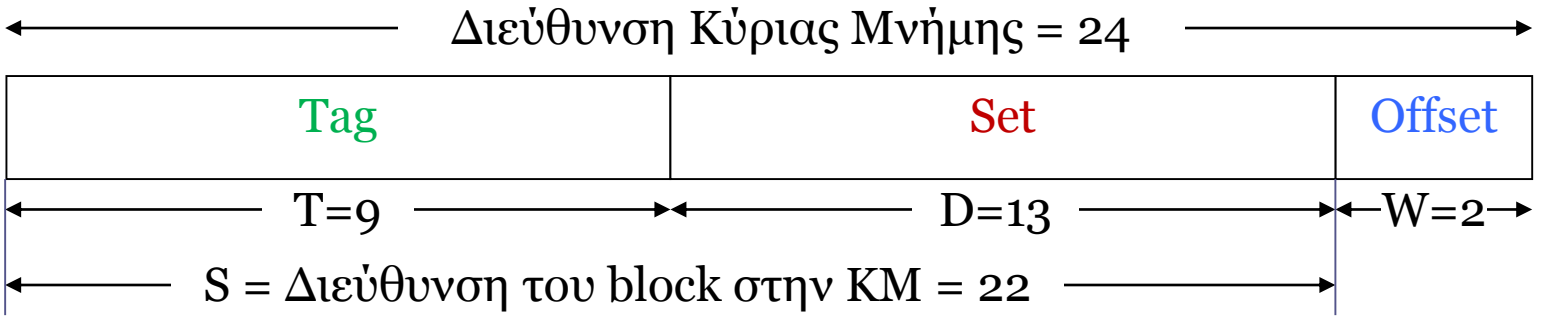
- **Συνάρτηση απεικόνισης:** (mapping function)
  αν j η διεύθυνση ο ενός block στη κεντρική μνήμη
  i η διεύθυνση ενός set
  i = j modulo n

  Δηλαδή το block της μνήμης με διεύθυνση j θα τοποθετηθεί οπουδήποτε μέσα στο στο i set
  Μια τέτοια cache ονομάζεται B-way set associative

Τμηματικά συσχετιστική cache (2-way set associative)



Τμηματικά συσχετική cache (Set Associative)

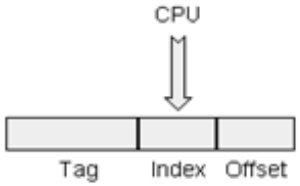


- Offset: Η διεύθυνση ενός byte μέσα στο block
- Set : Η διεύθυνση ενός set
- Tag : Ετικέτα αναγνώρισης (μέρος της διεύθυνσης)
- Μέγεθος Block : 4 bytes = 2^w bytes → W = 2
- Μέγεθος Μνήμης : 16 Mbytes = 2^{24} = 2^{s+w}
- Μήκος διεύθυνσης Κ.Μ. : (S + W) bits = 24 bits → S = 22
- Πλήθος blocks στη μνήμη : $2^s = 2^{22}$ = 4 Mblocks
- 2 blocks ανά set (2-way set associative)
- Μέγεθος Cache : 64 Kbytes = 2^{16} bytes
- Πλήθος blocks στην cache: $2^{16} / 2^2 = 2^{14}$
- Πλήθος sets στην cache: $2^D = 2^{14} / 2 = 2^{13}$ → D = 13
- Μέγεθος tag : T = S−D bits = 9

Τμηματικά Συσχετική – Αλγόριθμος Προσπέλασης

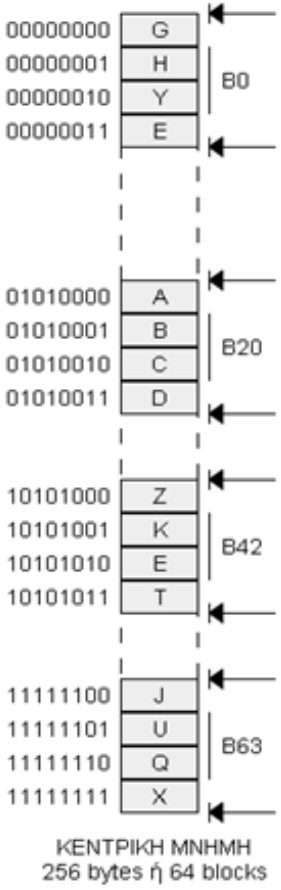
- **Βήμα 0:** Ανάλυση της διεύθυνσης Α στα πεδία, tag, set, w.
- **Βήμα 1:** Η set διεύθυνση μας οδηγεί σε ένα set, μέσα στα block του οποίου, **ενδεχομένως,** να βρίσκεται το περιεχόμενο της διεύθυνσης που αναζητούμε.
- **Βήμα 2:** Το tag της διεύθυνσης συγκρίνεται με όλα τα tag των γραμμών του set στην οποία μας οδήγησε η διεύθυνση set
- **Βήμα 3:** Αν κάποιο tag είναι ίσο με το tag της διεύθυνσης τότε έχουμε **cache hit** και η γραμμή της cache η οποία έχει το ίδιο tag με αυτό της διεύθυνσης Α περιέχει την διεύθυνση Α
- διαφορετικά έχουμε **cache miss:** η κ. μνήμη μεταφέρει στην cache όλο το block με διεύθυνση (**tag, set, 00**), στο οποίο βρίσκεται η διεύθυνση Α.
- **Βήμα 4:** Μεταφέρεται έξω από την cache το μέρος του block που έχει διεύθυνση **(tag, set, w)**

Τμηματικά Συσχετιστική – Cache Hit (0)



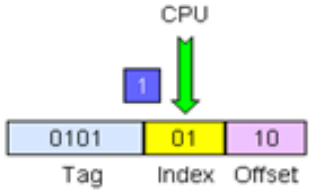
Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

CACHE 16 Blocks



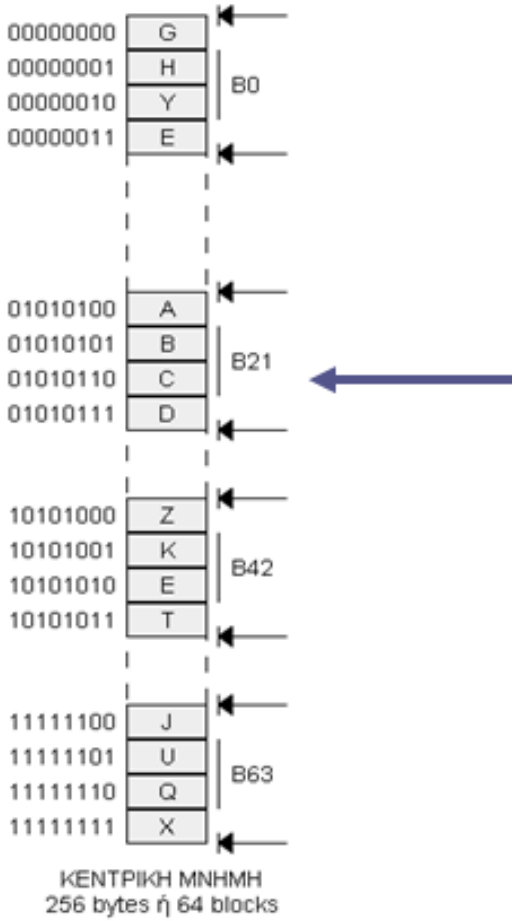
ΣΥΓΚΡΙΣΗ TAG

Τμηματικά Συσχετιστική – Cache Hit (1)



Block address	Tag	Block			
0000	0000	G	H	Y	E
0001					
0010					
0011					
0100	1111	R	E	Q	Z
0101					
0110					
0111	0101	A	B	C	D
1000					
1001					
1010					
1011					
1100					
1101					
1110	1111	J	U	Q	X
1111					

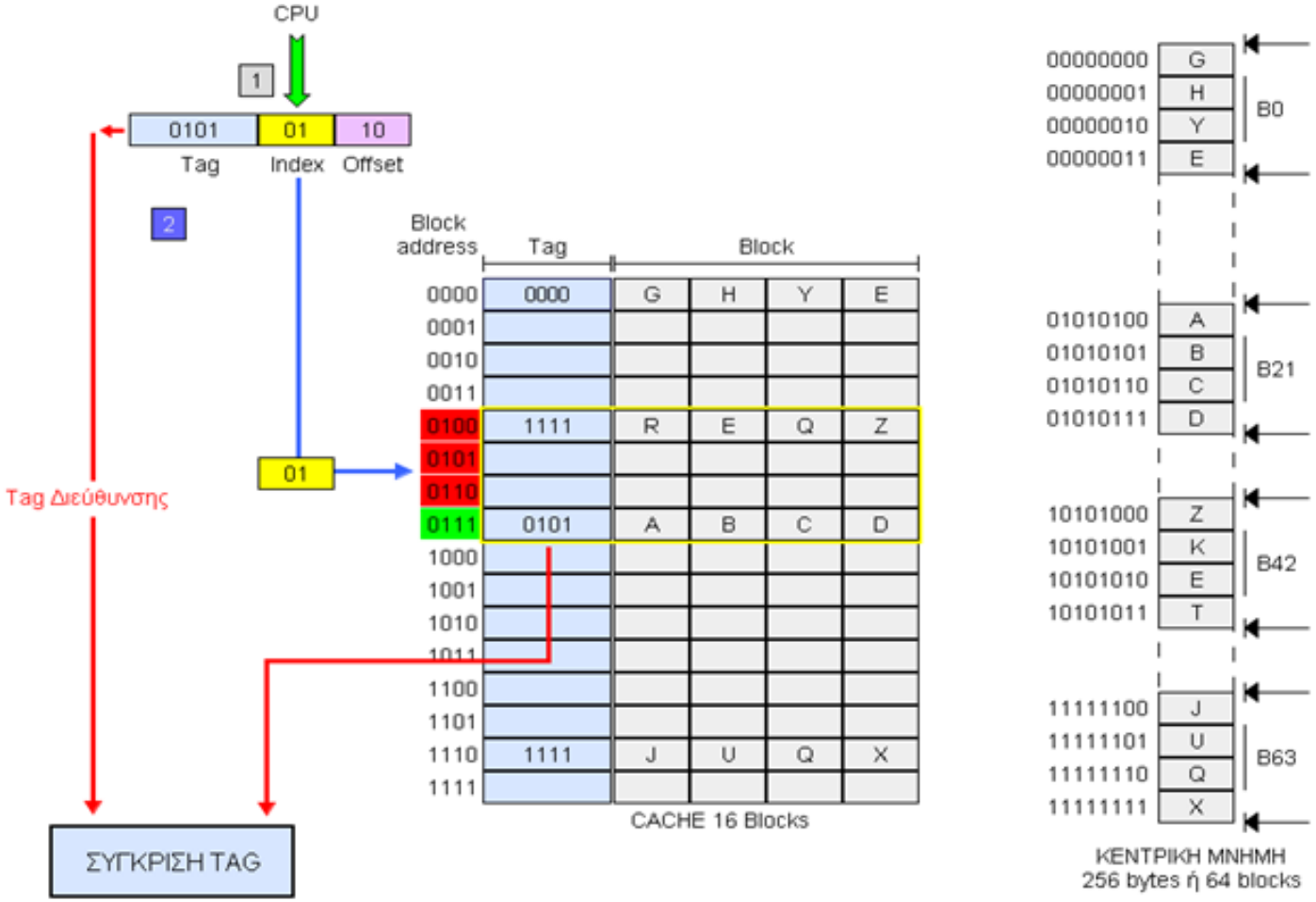
CACHE 16 Blocks



ΣΥΓΚΡΙΣΗ TAG

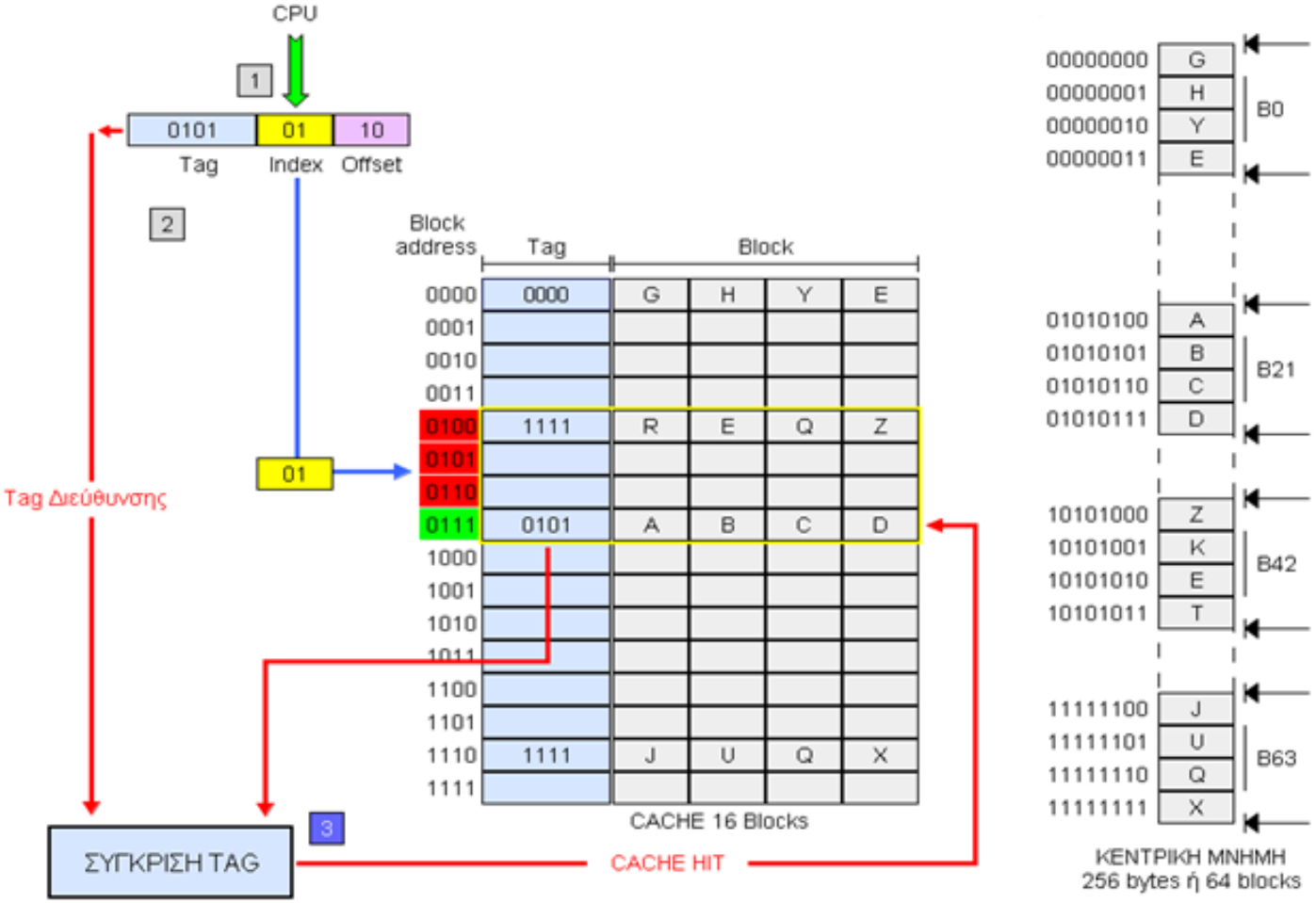
1. Η διεύθυνση εισέρχεται απο τη CPU και χωρίζεται σε Tag, Index και Offset

Τμηματικά Συσχετιστική – Cache Hit (2)



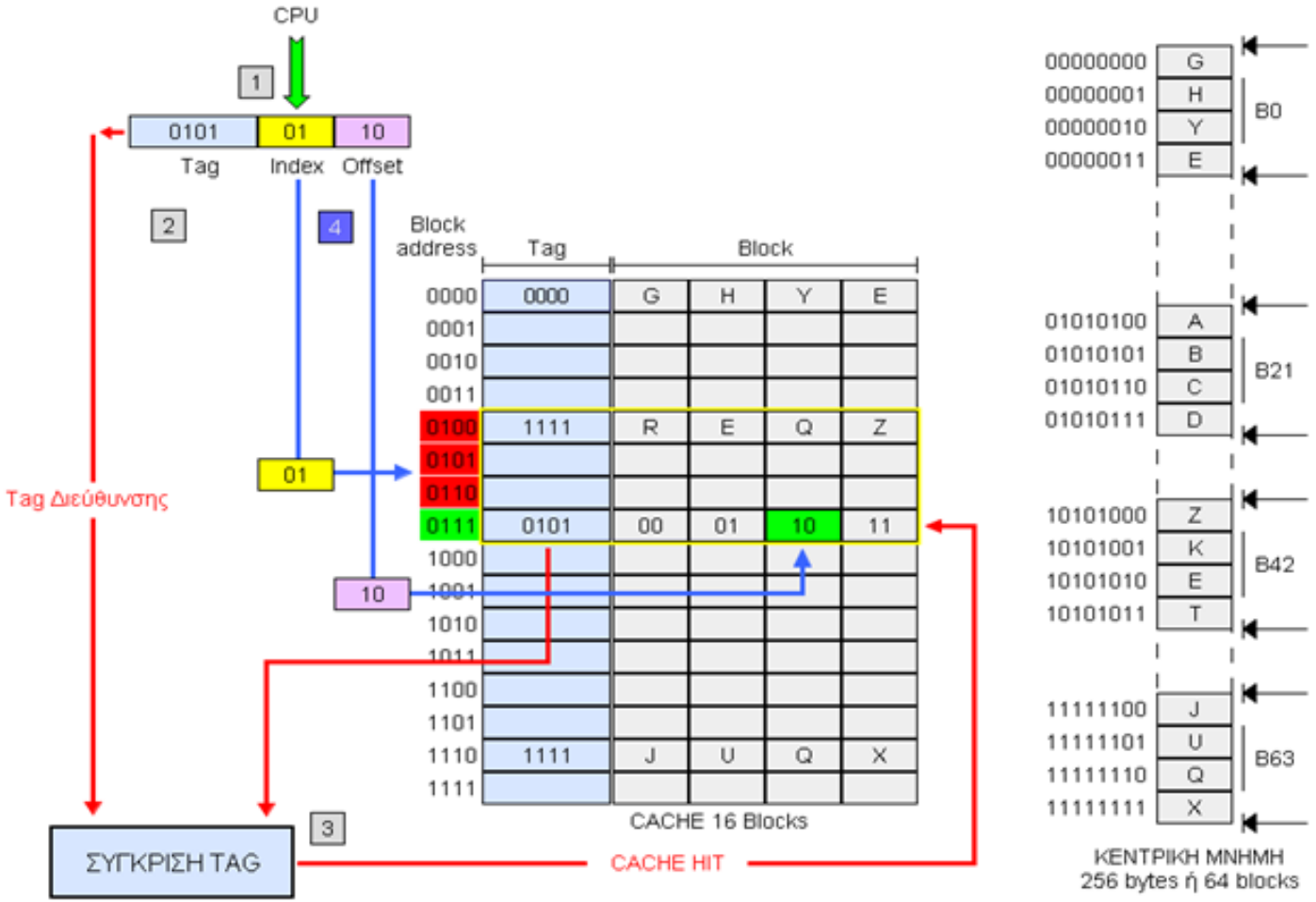
2. Το Tag της διεύθυνσης συγκρίνεται με τα Tag του Block της Cache, που αντιστοιχεί στο Index της διεύθυνσης

Τμηματικά Συσχετιστική – Cache Hit (3)



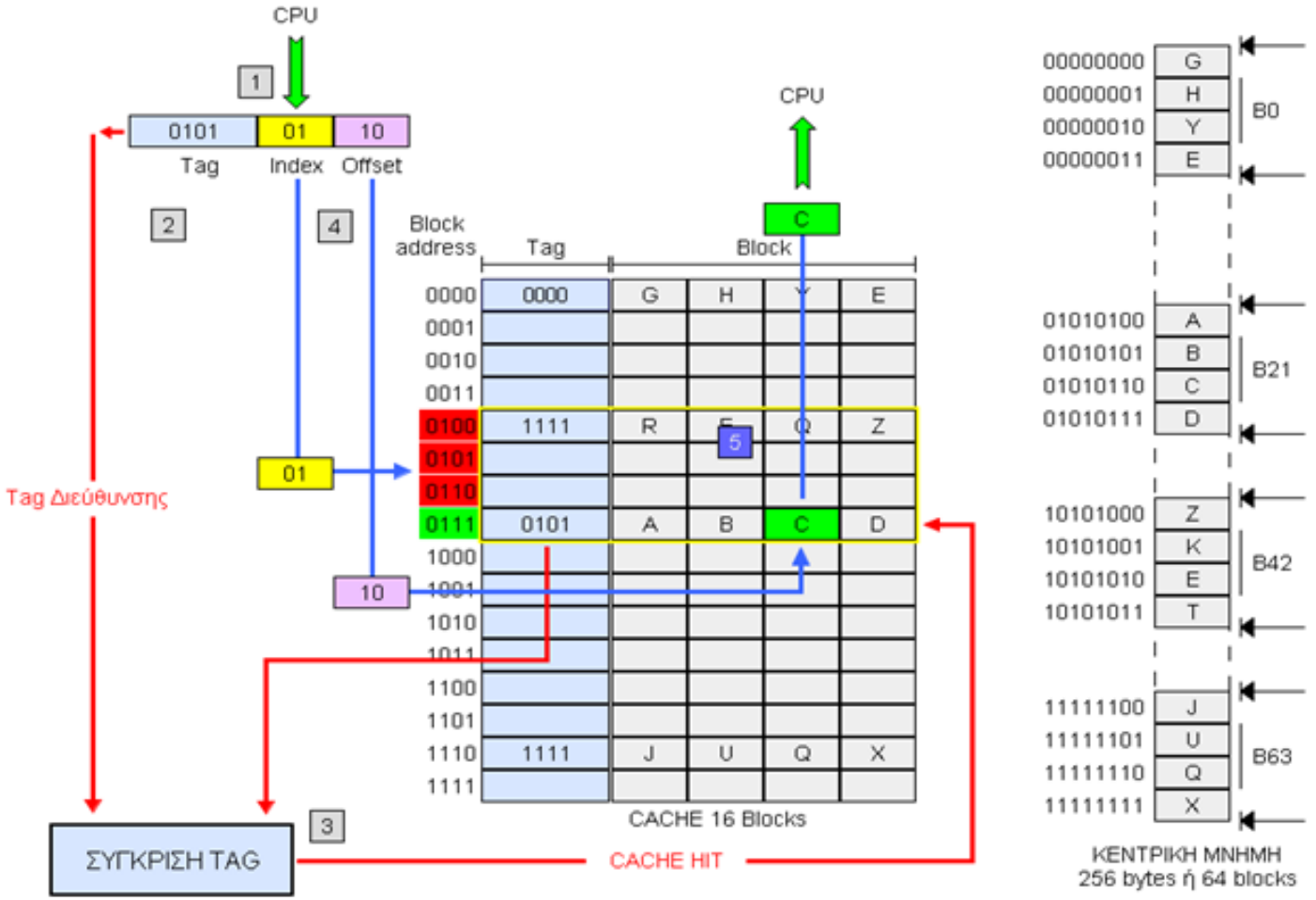
3. Ένα από τα Tag του συμπίπτει με το Tag της διεύθυνσης, οπότε έχουμε CACHE HIT

Τμηματικά Συσχετιστική – Cache Hit (4)



4. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

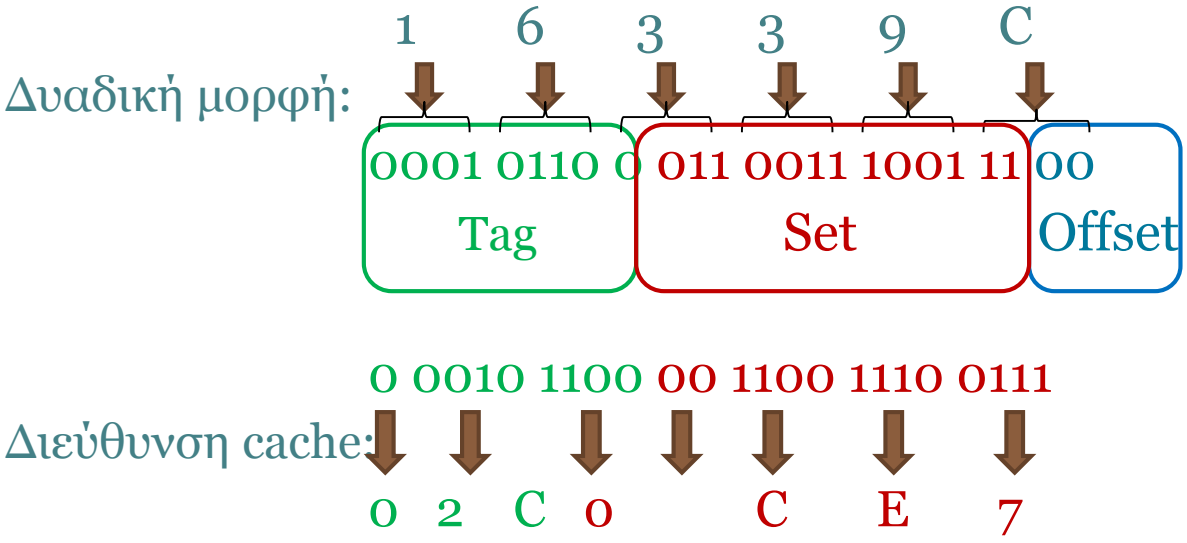
Τμηματικά Συσχετιστική – Cache Hit (5)



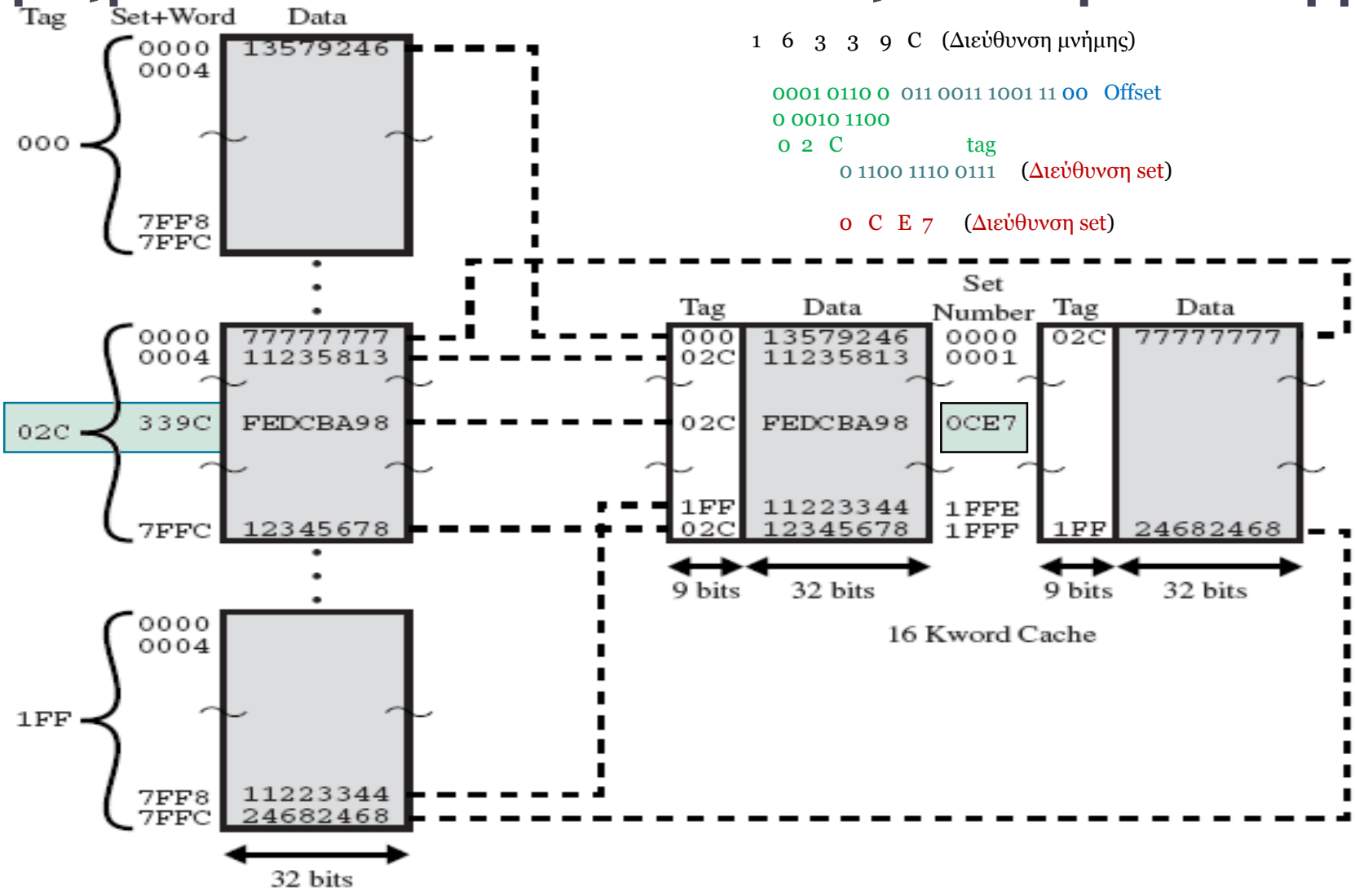
5. Η θέση μνήμης μεταφέρεται στη CPU

Τμηματικά Συσχετιστική – Ανάλυση Διεύθυνσης

Διεύθυνση μνήμης:



Τμηματικά Συσχετιστική - παράδειγμα



1 6 3 3 9 C (Διεύθυνση μνήμης)

0001 0110 0 011 0011 1001 11 00 Offset

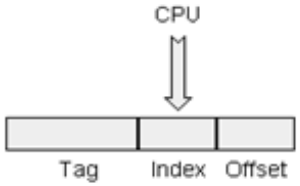
0 0010 1100

0 2 C tag

0 1100 1110 0111 (Διεύθυνση set)

0 C E 7 (Διεύθυνση set)

Τμηματικά Συσχετιστική – Cache Miss (0)



Block address	Tag	Block			
0000	00	G	H	Y	E
0001					
0010					
0011					
0100	01	A	B	C	D
0101					
0110					
0111					
1000					
1001					
1010	10	Z	K	E	T
1011					
1100					
1101					
1110					
1111	11	J	U	Q	X

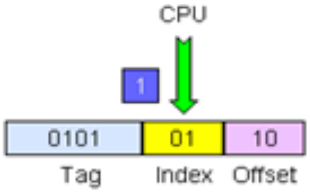
CACHE 16 Blocks

ΣΥΓΚΡΙΣΗ TAG

00000000	G	B0
00000001	H	
00000010	Y	
00000011	E	
01010000	A	B20
01010001	B	
01010010	C	
01010011	D	
10101000	Z	B42
10101001	K	
10101010	E	
10101011	T	
11111100	J	B63
11111101	U	
11111110	Q	
11111111	X	

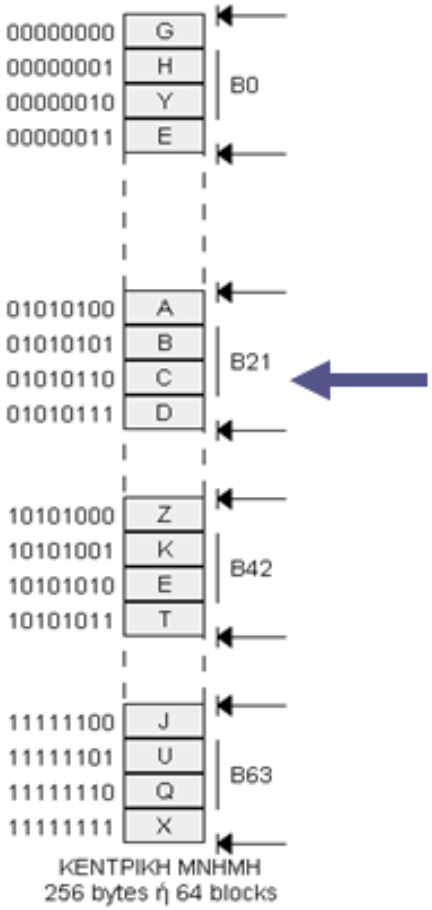
ΚΕΝΤΡΙΚΗ ΜΝΗΜΗ
256 bytes ή 64 blocks

Τμηματικά Συσχετιστική – Cache Miss (1)



Block address	Tag	Block			
0000	0000	G	H	Y	E
0001					
0010					
0011					
0100	1111	R	E	Q	Z
0101	0111	H	Y	T	S
0110	0110	O	F	Z	A
0111	1101	K	U	C	L
1000					
1001					
1010	1000	Q	R	V	F
1011					
1100	0000	T	M	O	F
1101					
1110					
1111	1111	J	U	Q	X

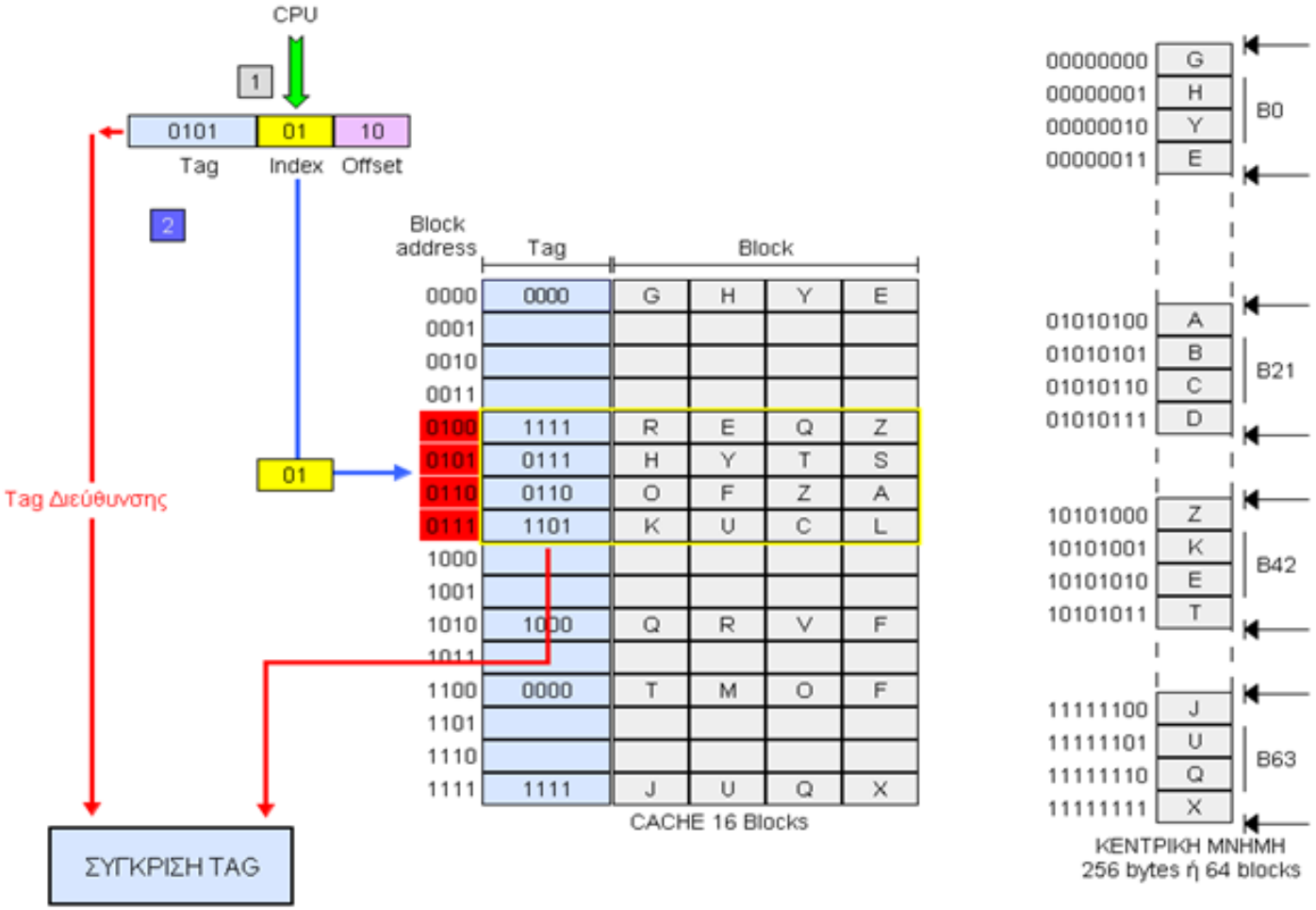
CACHE 16 Blocks



ΣΥΓΚΡΙΣΗ TAG

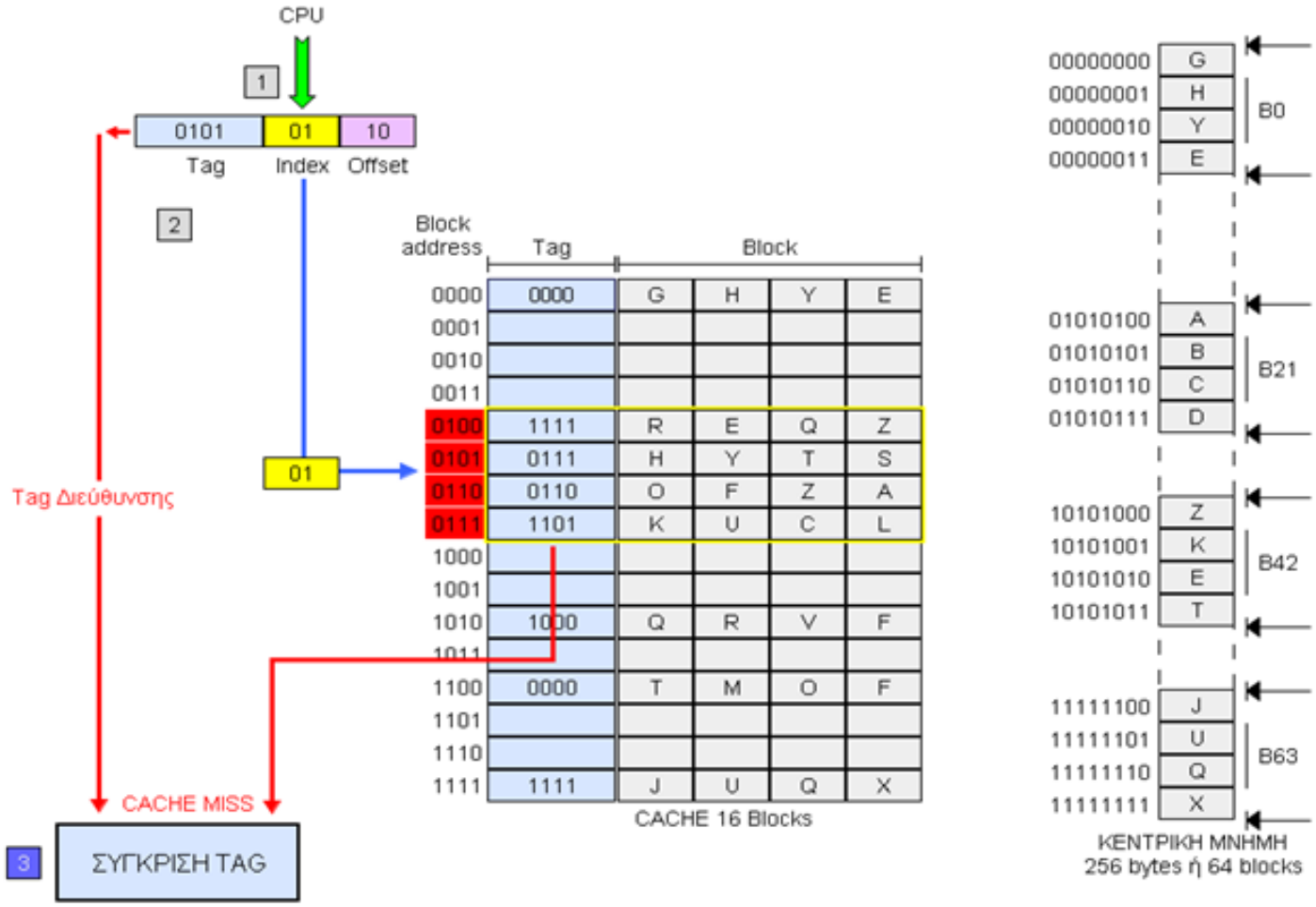
1. Η διεύθυνση εισέρχεται απο την CPU και χωρίζεται σε Tag, Index και Offset

Τμηματικά Συσχετιστική – Cache Miss (2)



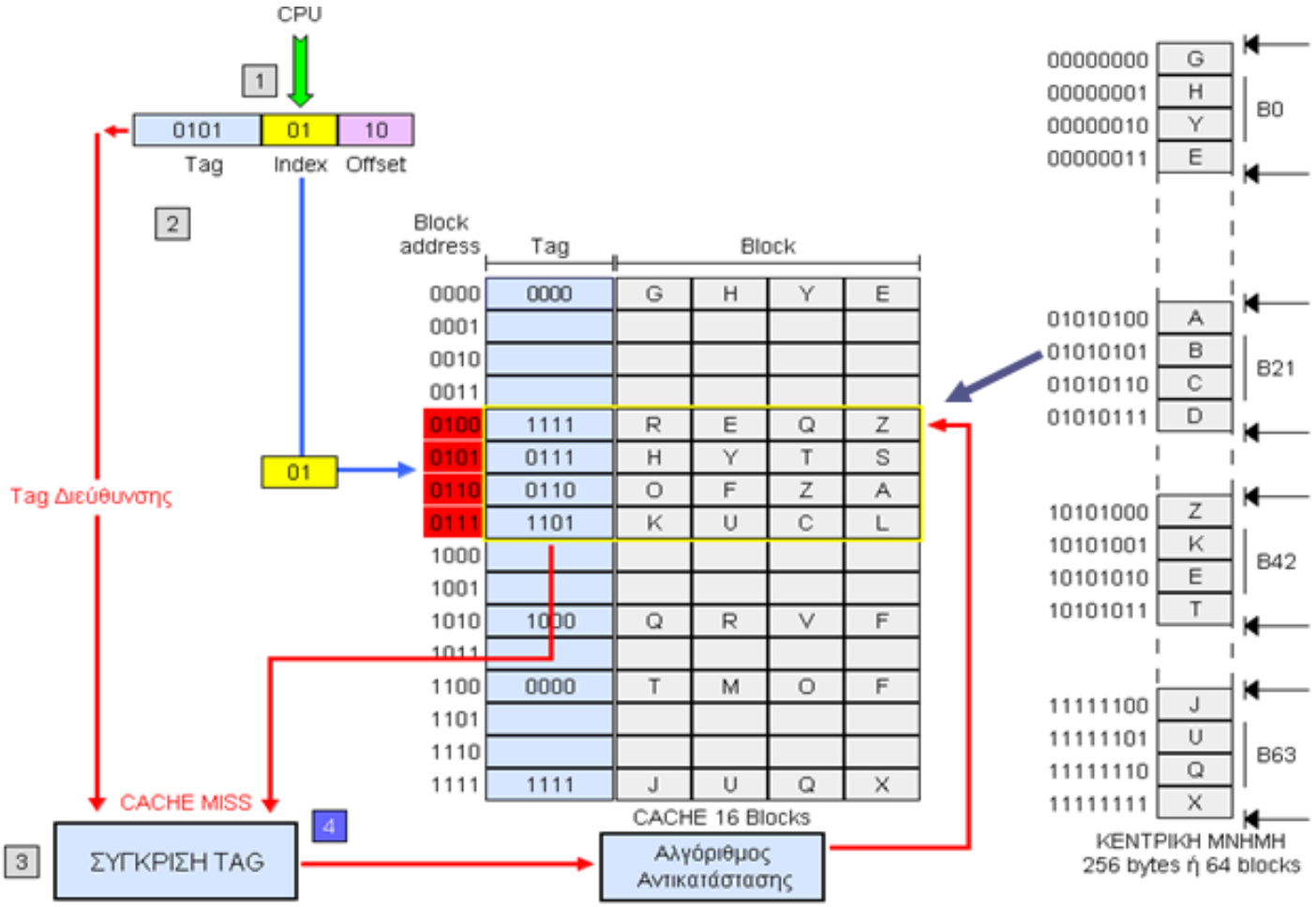
2. Το Tag της διεύθυνσης συγκρίνεται με τα Tag του Block της Cache, που αντιστοιχεί στο Index της διεύθυνσης

Τμηματικά Συσχετιστική – Cache Miss (3)



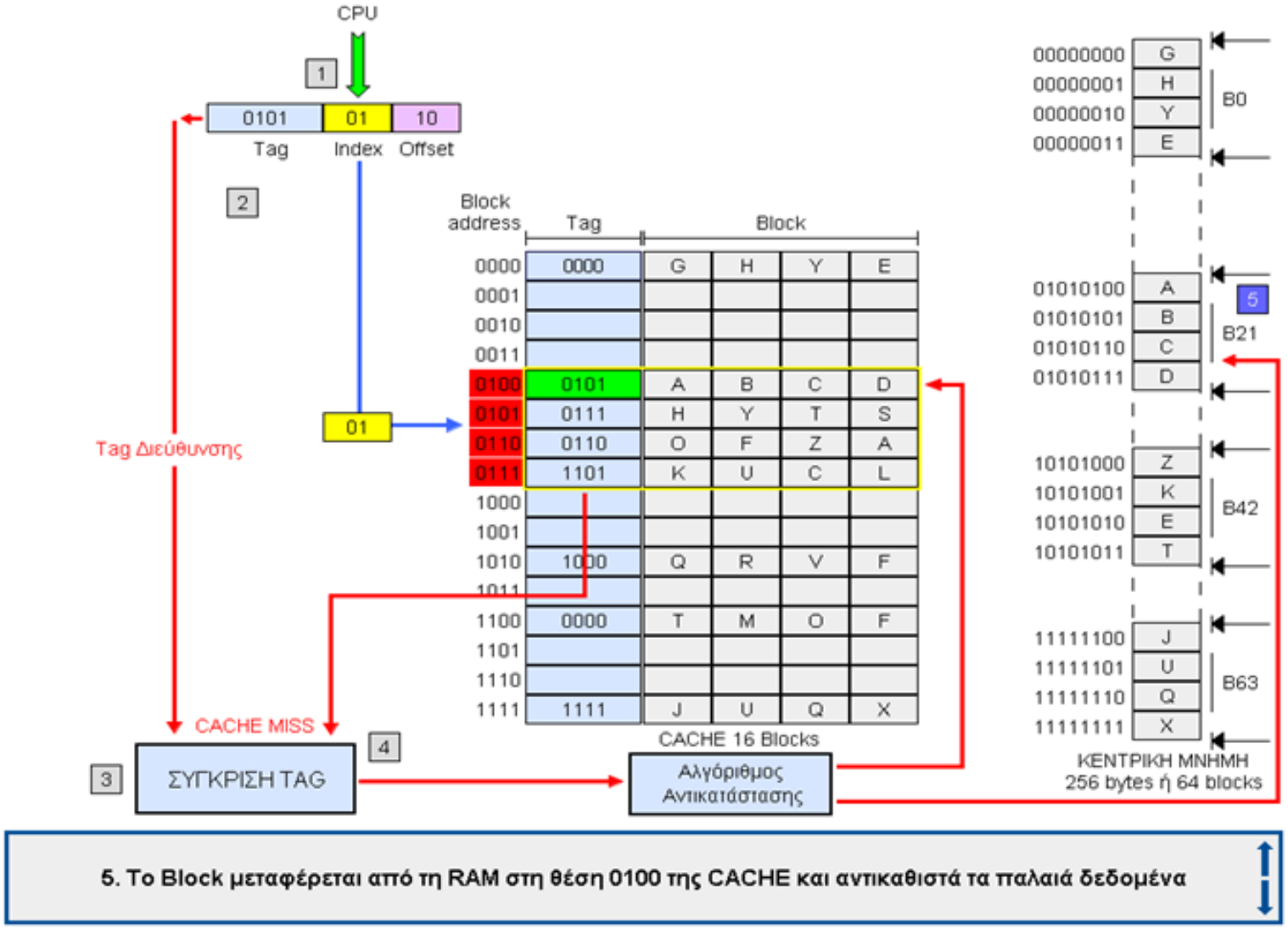
3. Κανένα από τα Tag δεν συμπίπτει, οπότε έχουμε CACHE MISS

Τμηματικά Συσχετιστική – Cache Miss (4)



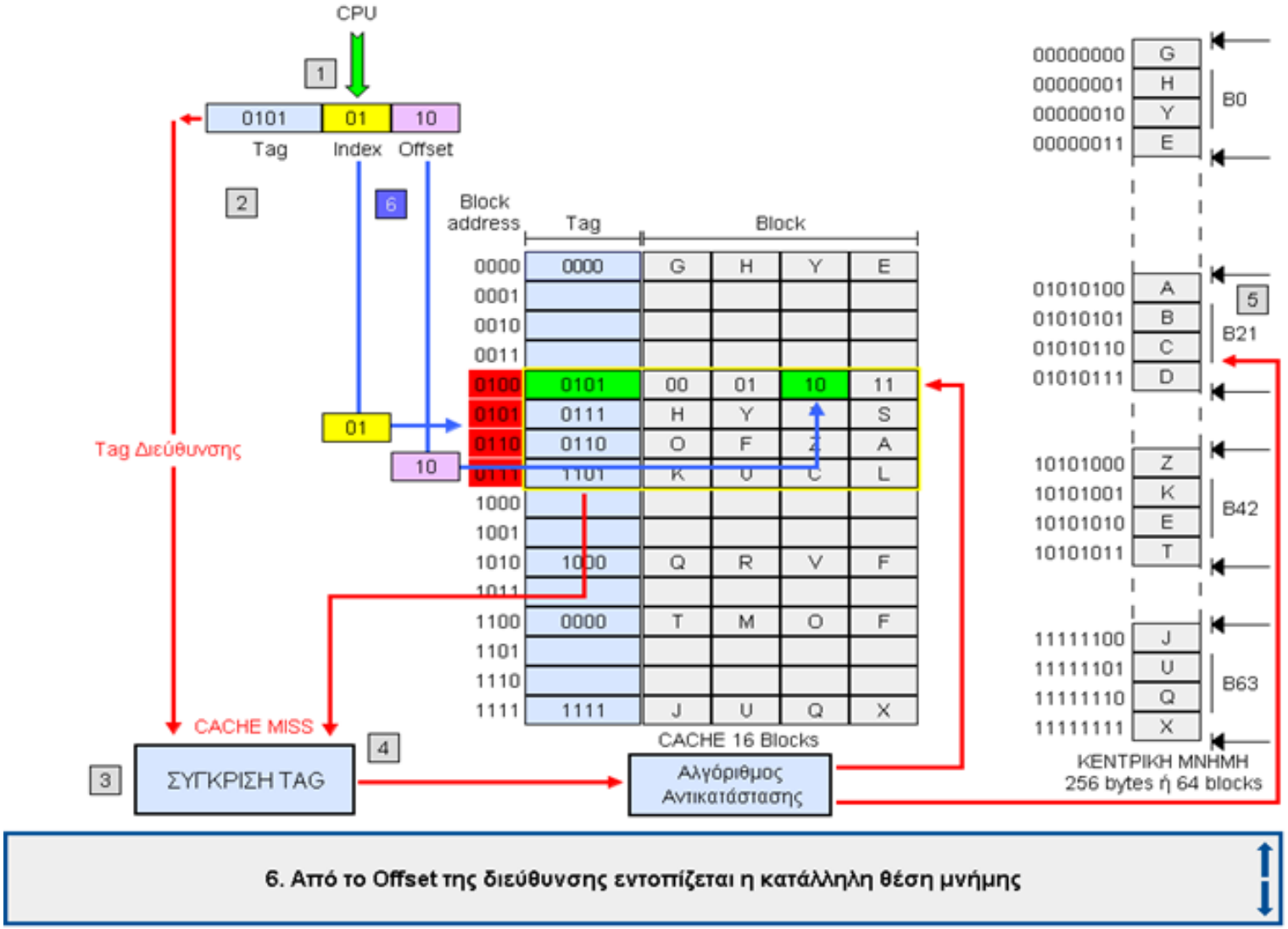
4. Ο Αλγόριθμος Αντικατάστασης επιλέγει το block της Cache που πρέπει να αντικατασταθεί (0100)

Τμηματικά Συσχετιστική – Cache Miss (5)



5. Το Block μεταφέρεται από τη RAM στη θέση 0100 της CACHE και αντικαθιστά τα παλαιά δεδομένα

Τμηματικά Συσχετιστική – Cache Miss (6)



6. Από το Offset της διεύθυνσης εντοπίζεται η κατάλληλη θέση μνήμης

Τμηματικά Συσχετιστική – Cache Miss (7)