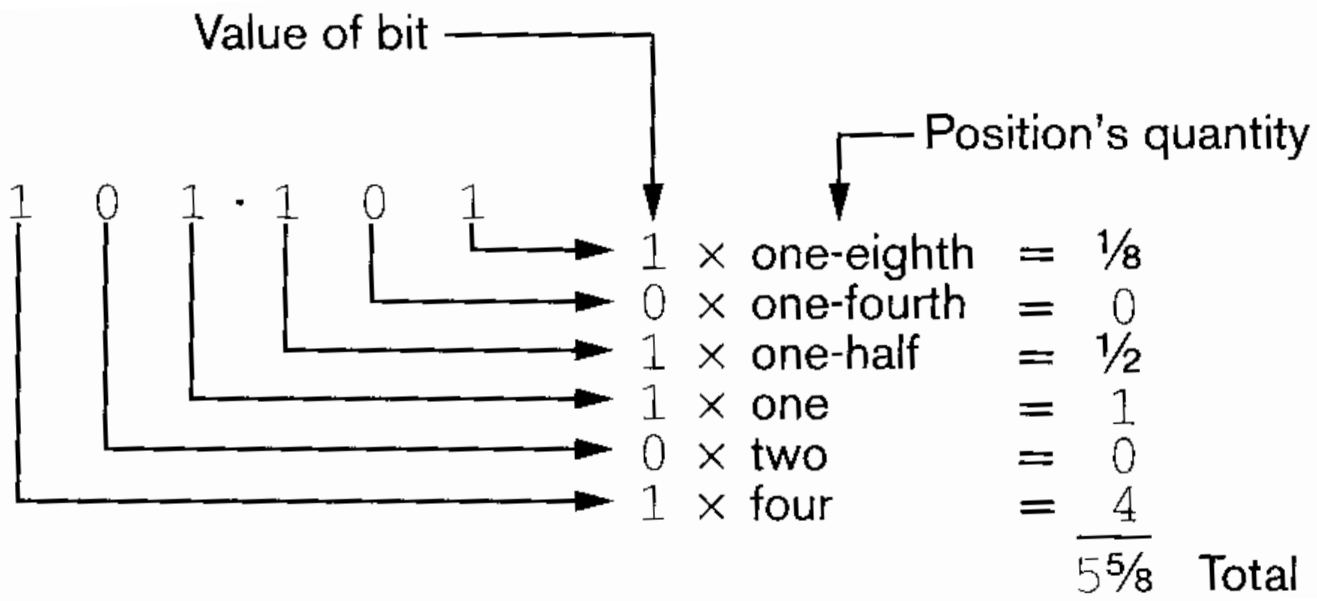
Εισαγωγή στην Επιστήμη &
Τεχνολογία της Πληροφορικής

Αρχιτεκτονική Υπολογιστών

1η Ενότητα

Χειμερινό εξάμηνο 2022

Αναπαράσταση κλασμάτων σε δυαδική μορφή



Παραδείγματα:

11.01 → 3 ¼
101.111 → 5 7/8

2 3/4 → 10.11
5/16 → 0.0101

1010.001
\+ 1.101

1011.110

Συμπλήρωμα ως προς 2

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	-1
110	-2
101	-3
100	-4

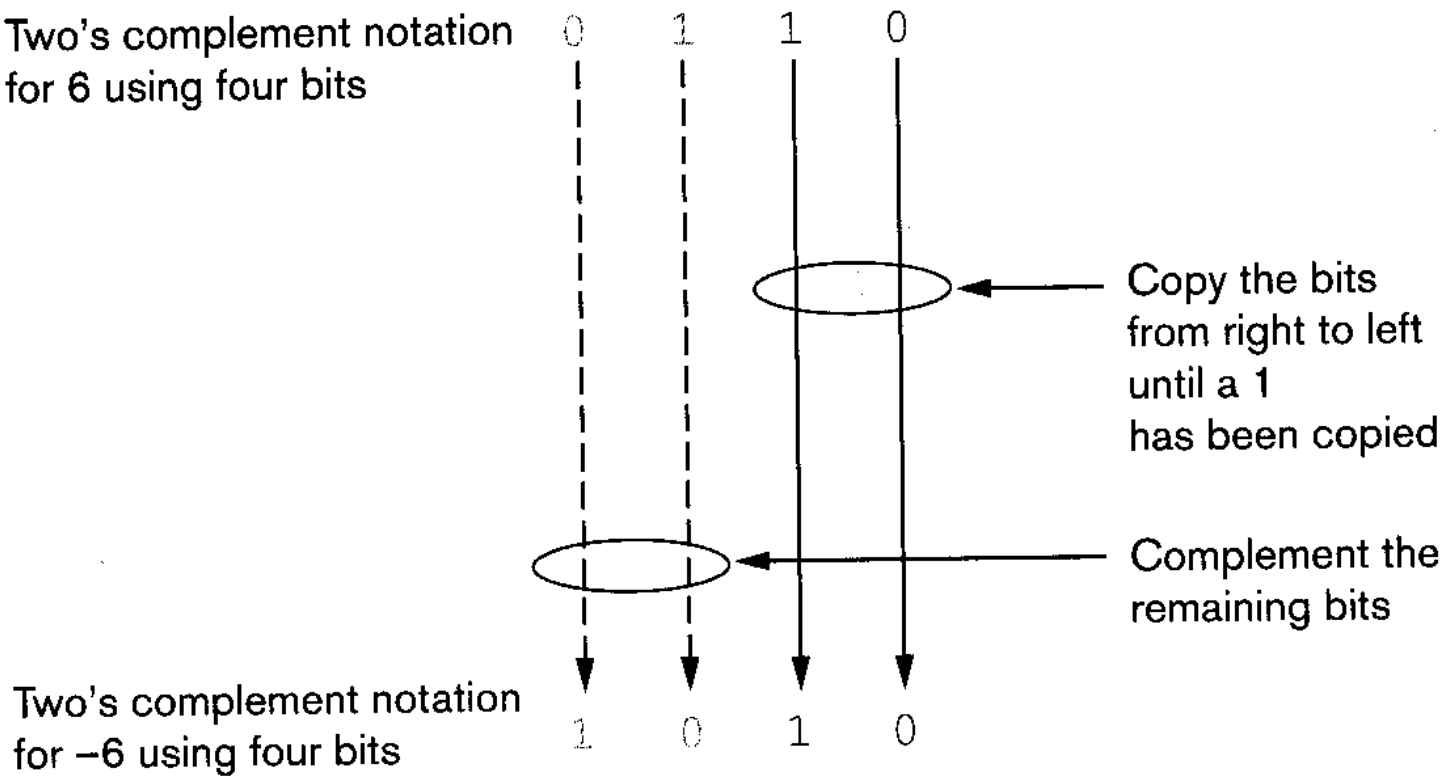
Sign bit

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

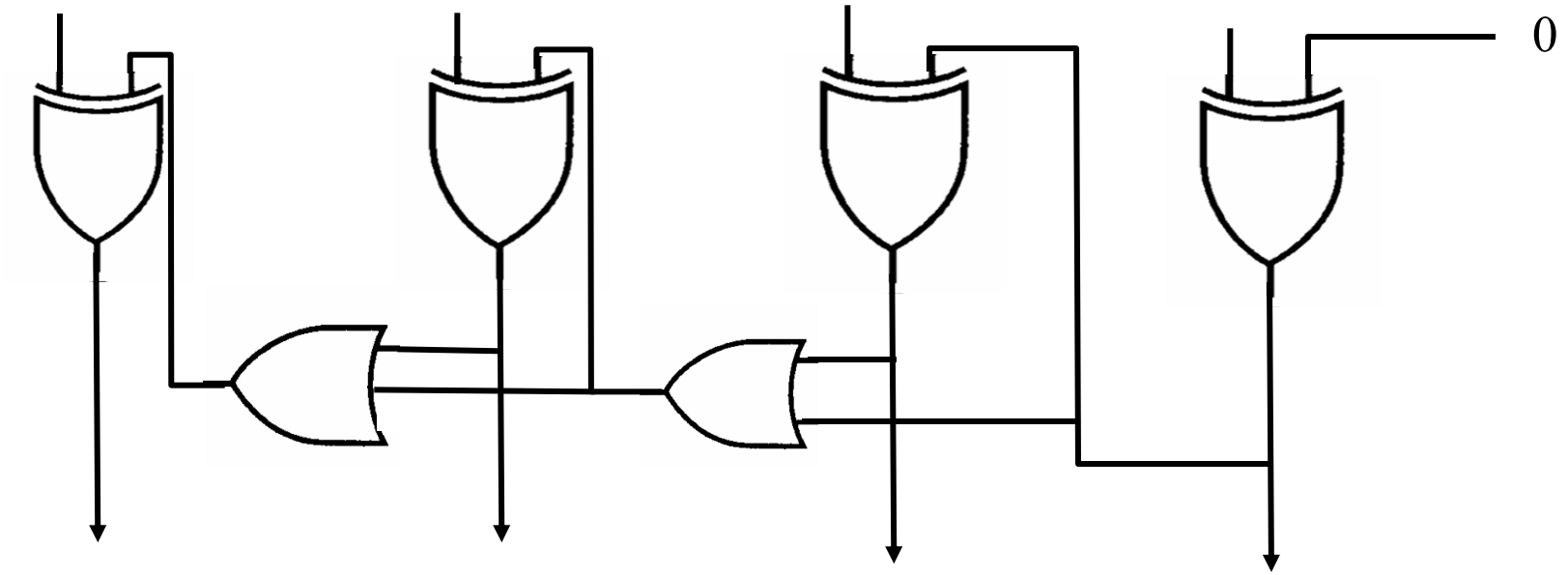
Αλλαγή μετά τον πρώτο άσσο

Συμπλήρωμα ενός αριθμού: π.χ., 0110 & 1001

Συμπλήρωμα ως προς 2

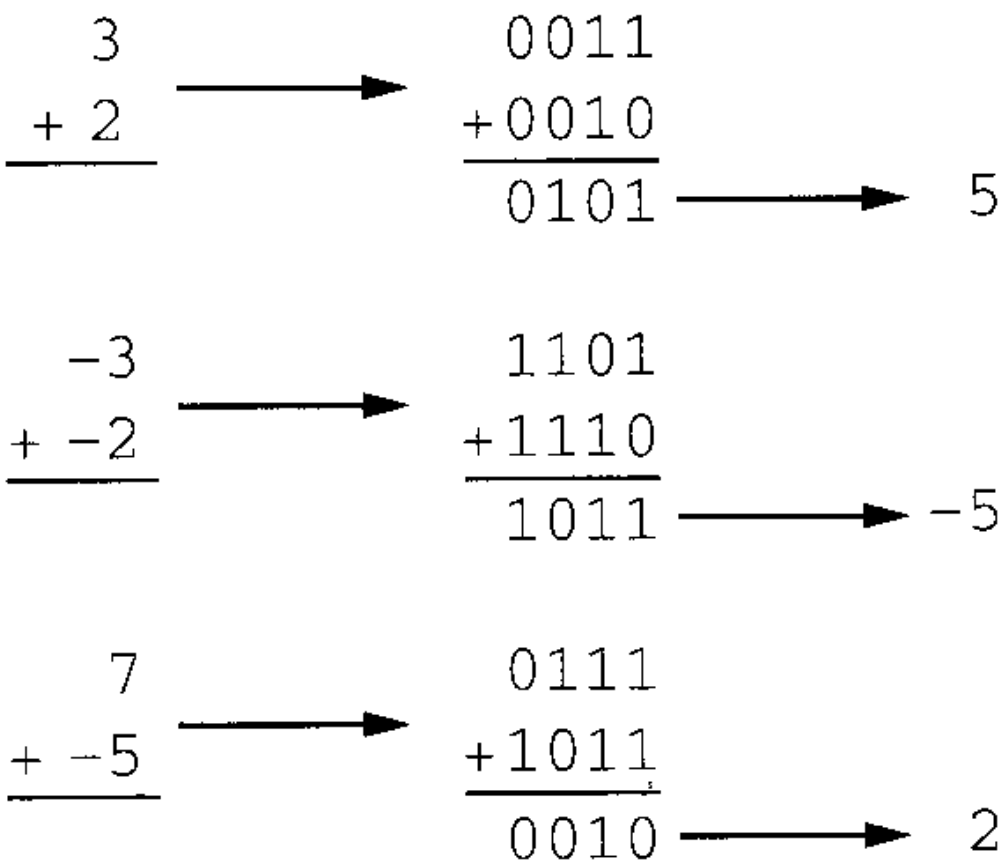


Συμπλήρωμα ως προς 2



Συμπλήρωμα ως προς 2

Πρόσθεση



- Αποκοπή του αριστερότερου ψηφίου αν το μήκος του αποτελέσματος ξεπερνάει αυτό των όρων που προστίθενται
- Πρόσθεση & αφαίρεση πραγματοποιούνται με τον ίδιο τρόπο

Το πρόβλημα της υπερχείλισης (overflow)

- 0101 + 0100 = 1001 (9 ή -7???)
- Υπερχείλιση: όταν θέλουμε να αναπαραστήσουμε μια τιμή η οποία πέφτει έξω από το πεδίο των τιμών που μπορούν να αναπαρασταθούν
- Σήμερα είναι κοινή η χρήση 32 bits

2.147.483.647 θετικές τιμές

- Εντοπισμός: πρόσθεση θετικών να δίνει αρνητικό αποτέλεσμα ή αντίστροφα
- Παλαιότερα χρήση 16 bits ή 32,768. Πρόγραμμα σε νοσοκομείο σταμάτησε να δουλεύει στις 19/9/1989 ή 32768 μέρες μετά την 1/1/1900...

Το πρόβλημα της υπερχείλισης (overflow)

0101	1010	0111
0110	1010	0001
-------	-------	-------
1011	0100	1000

Μπορούμε να έχουμε overflow όταν προσθέτουμε θετικό με αρνητικό αριθμό?

Αναπαράσταση με τον κώδικα excess

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

Η πρώτη εμφάνιση του 1 σε αυτή τη θέση καθορίζει το 0

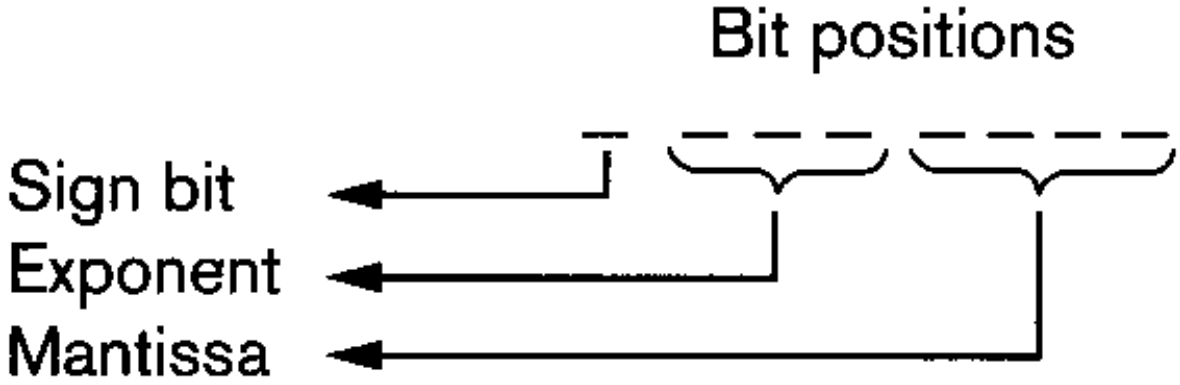
Excess 8 Notation: Η διαφορά που έχει ένας αριθμός από την τυπική αναπαράσταση είναι πάντα 8

Excess ?? Notation: αν κάνουμε χρήση 5 ψηφίων

Excess 4

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

Αναπαράσταση κινητής υποδιαστολής (floating point representation)



01101011:
1: Exponent 110 (excess - 4) notation = 2
Mantissa: .1011
2. Mantissa: 10.11 = $2^{3/4}$
3. +2 3/4

10111100:
1: Exponent 011 (excess - 4) notation = -1
Mantissa: .1100
2. Mantissa: .01100 = 3/8
3. -3/8

Αναπαράσταση κινητής υποδιαστολής (floating point representation)

Κωδικοποίηση του 1 1/8:

1: δυαδική αναπαράσταση 1.001

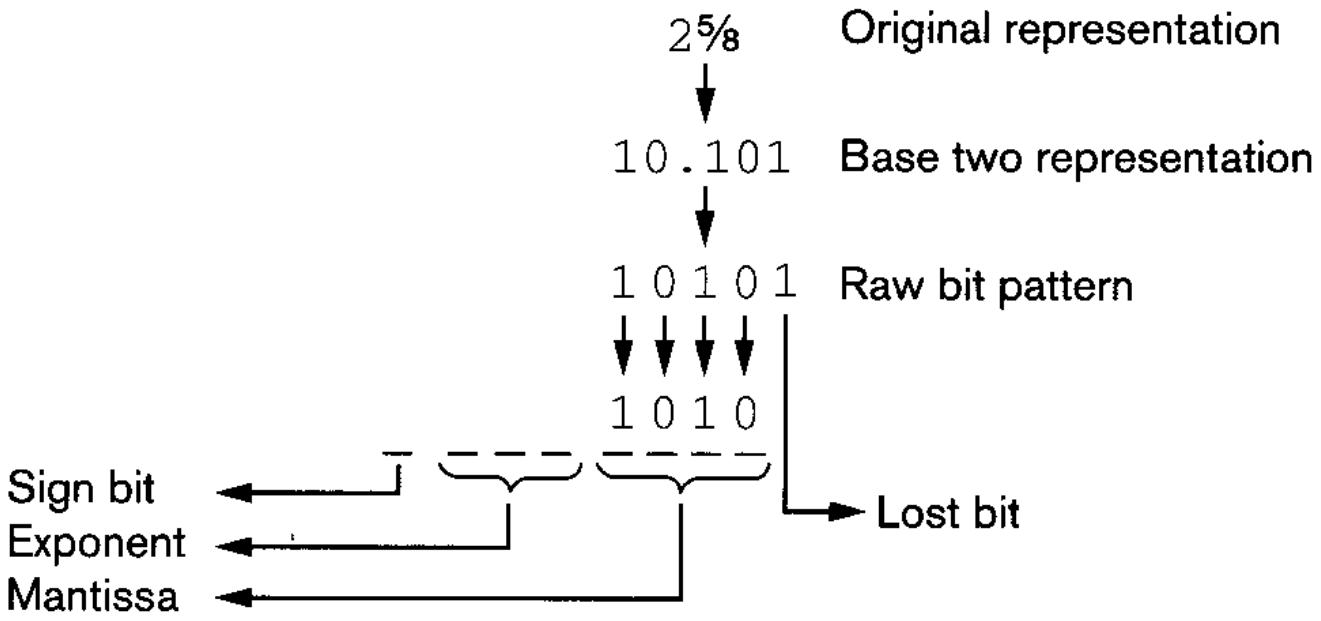
2. Mantissa .1001

3. Exponent: 101 (Excess -4 = +1)

4. Πρόσημο:0

5. Αποτέλεσμα: 01011001

Λάθη αποκοπής (truncation errors)



- Αν συνεχίζαμε το αποτέλεσμα θα ήταν 01101010 = 2 ½ (λάθος αποκοπής ή στρογγυλοποίησης)
- Άλλη πηγή τέτοιων λαθών οι άρρητοι αριθμοί
- Ακόμα και αυξήσουμε το μήκος των πεδίων πάντα θα υπάρχουν περιπτώσεις που θα απαιτούμε μεγαλύτερη ακρίβεια

Λάθη αποκοπής (truncation errors)

- 2 1/2 + 1/8 + 1/8 = 2 ¾
- 2 ½ + 1/8 = 2 5/8 ή 10.101 αποθήκευση ως 01101010 = 2 ½
- Επαναλαμβάνοντας την πράξη παίρνουμε ως τελικό αποτέλεσμα 2 ½
- Εναλλακτικά μπορούμε να προσθέσουμε

  1/8 +1/8 = 2/8=1/4 ή 00111000

  ¼ + 2 ½ = 01101011 ή 2 ¾
- Πρόβλημα στην πρόσθεση μεγάλων τιμών με μικρές τιμές (σημαντικές επιπτώσεις σε συγκεκριμένες περιπτώσεις π.χ., συστήματα πλοήγησης)

Συμπίεση δεδομένων

Run Length Encoding: αντικατάσταση επαναλαμβανόμενων τιμών από ένα κωδικό και τον αριθμό των επαναλήψεων (π.χ., 253 άσσοι 118 μηδενικά και 87 άσσοι χρειάζονται λιγότερα bits από 458)

Relative Encoding: κωδικοποίηση μεγάλων κομματιών πληροφορίας που διαφέρουν ελάχιστα μεταξύ τους (π.χ., συνεχόμενες εικόνες σε ταινία)

Frequency Dependent Encoding: το μήκος μιας σειράς από bits είναι αντιστρόφως ανάλογο της συχνότητας με την οποία εμφανίζεται η σειρά (π.χ., τα α, ε, ο μικρότερο μήκος από τα χ, ψ, ξ) Αλγόριθμοι Huffman

Lempel – Ziv Encoding (adaptive dictionary encoding): Τεχνικές κωδικοποίησης γενικού σκοπού (π.χ., zip).

Συμπίεση δεδομένων

Lempel-Ziv LZ77Q

(a) Count backward 5 symbols.

α β α α β ρ β

(b) Identify the 4-bit segment to be appended to the end of the string.

α β α α β ρ β α α β ρ

(c) Copy the 4-bit segment onto the end of the message.

α β α α β ρ β α α β ρ α

(d) Add the symbol identified in the triple to the end of the message.

Decompressing αβααβρβ (5, 4, α)

Συμπίεση δεδομένων

Lempel-Ziv LZ77Q

1. αβααβρβ (5,4,α) (0,0,δ) (8,6,β)
2. αβααβρβααβρα (0,0,δ) (8,6,β)
3. αβααβρβααβραδ (8,6,β)
4. αβααβρβααβραδρβααβρβ

Συμπίεση δεδομένων

Συμπίεση εικόνας

GIF (Graphic Interchange Format) – Compuserve:

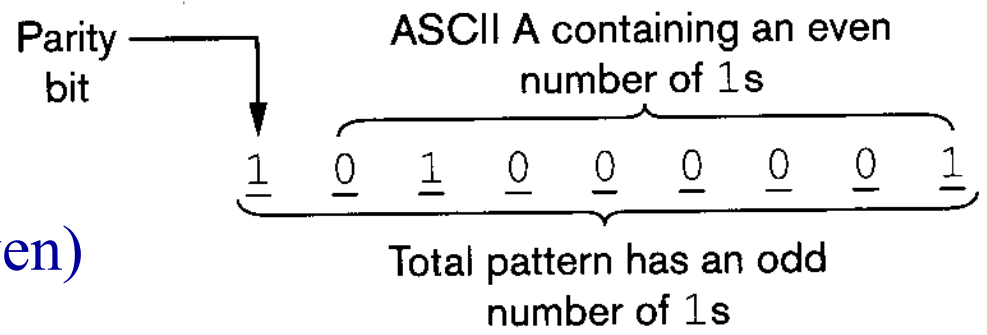
Αριθμός χρωμάτων σε κάθε pixel μόνο 256 = αντιστοιχία σε ένα συνδυασμό κόκκινου - μπλε-πράσινου. Μία από αυτές τις τιμές ορίζεται και ως διαφανής. Η συγκεκριμένη κωδικοποίηση είναι κατάλληλη για παιχνίδια όπου υπάρχει κίνηση.

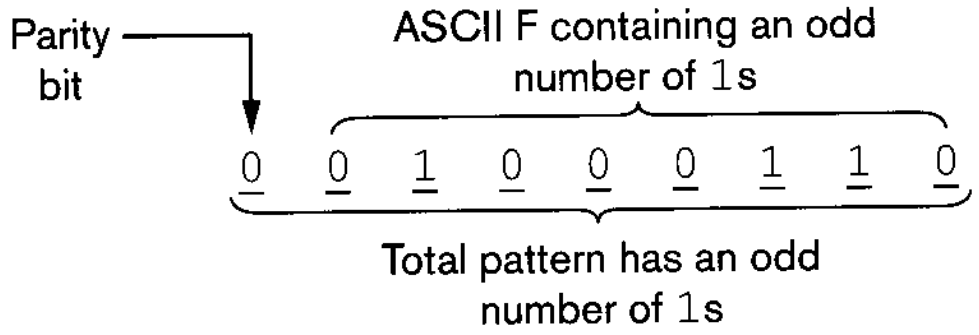
JPEG (Joint Photographic Experts Group) – ISO:

Η συγκεκριμένη κωδικοποίηση είναι κατάλληλη για έγχρωμες φωτογραφίες. Δυνατότητα λειτουργίας σε "lossless mode" όπου δεν υπάρχουν απώλειες γιατί η κωδικοποίηση κάθε pixel γίνεται σε σχέση με τις διαφορές από το επόμενο. Το αποτέλεσμα πάντως είναι η δημιουργία μεγάλων αρχείων. Χρήση κυρίως της βασικής JPEG κωδικοποίησης (τιμή για φωτεινότητα και δύο τιμές για το χρώμα).

Λάθη στη μετάδοση πληροφορίας

Parity Bit (odd or even)





The ASCII codes for the letters A and F adjusted for odd parity

Κώδικες διόρθωσης λαθών (Error Correcting Codes)

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

An error-correcting code

- Απόσταση Hamming: Ο αριθμός των bits που διαφέρουν δύο σειρές π.χ., (B-C = 3)
- Απόσταση Hamming στο παράδειγμα ≥ 3
- 010100???

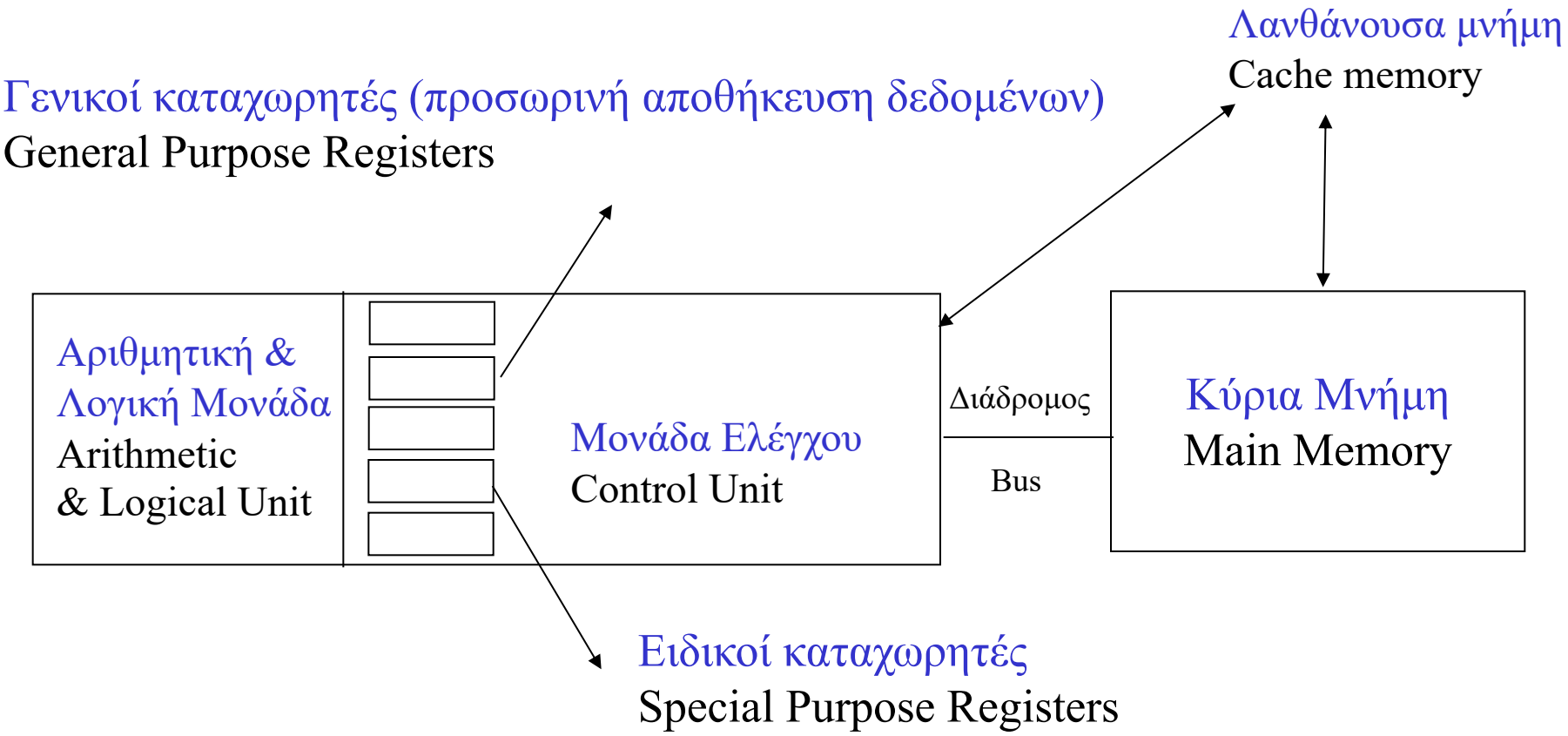
Κώδικες διόρθωσης λαθών (Error Correcting Codes)

Character	Distance between the received pattern and the character being considered	
A	2	
B	4	
C	3	
D	1	(Smallest distance)
E	3	
F	5	
G	2	
H	4	

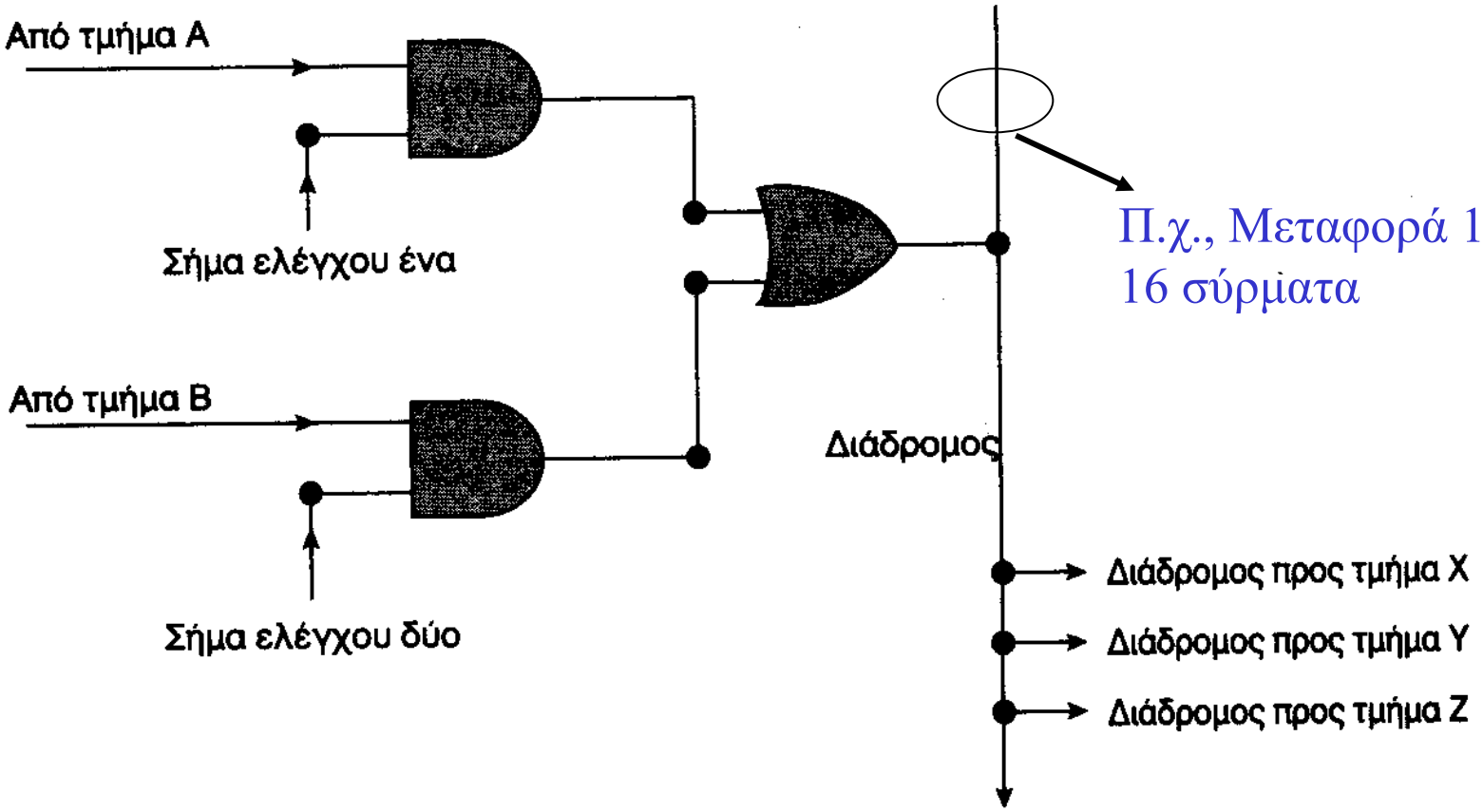
Περιεχόμενα ενότητας

- Κεντρική Μονάδα Επεξεργασίας
- Αριθμητικές και Λογικές εντολές
- Εντολές μηχανής
- Παράδειγμα εκτέλεσης προγράμματος

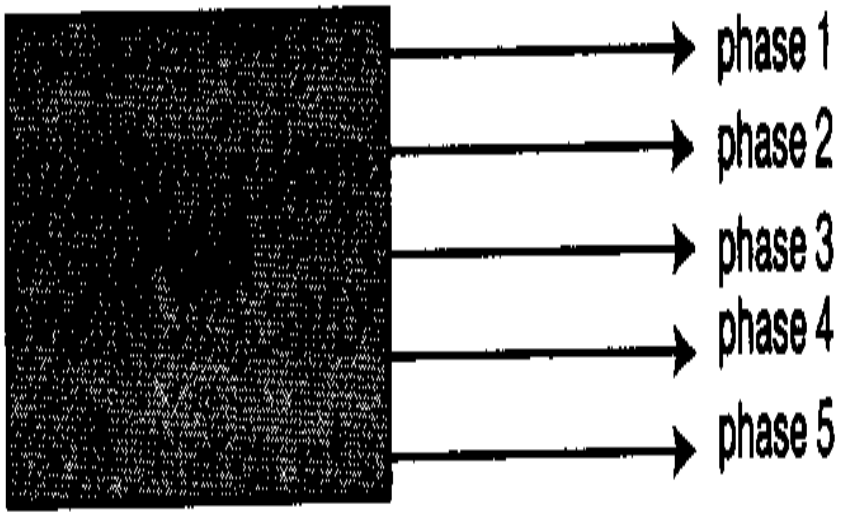
Η κεντρική μονάδα επεξεργασίας (Central Processing Unit – CPU)



Διάδρομοι δεδομένων



Ρολόι



- Ταχύτητα ρολογιού hertz (Hz). 1 Hz = 1 κύκλος/sec
- Τυπικές σημερινές ταχύτητες 2-3GHz
- Σημειώστε ότι διαφορετικές αρχιτεκτονικές και υλοποιήσεις καθιστούν αδύνατη την άμεση σύγκριση. Καλύτερη σύγκριση μέσω benchmarking

Αριθμητικές και λογικές εντολές

Λογικές πράξεις

AND

```
  10011010
  11001001
  ----------
  10001000
```

OR

```
  10011010
  11001001
  ----------
  11011011
```

XOR

```
  10011010
  11001001
  ----------
  01010011
```

AND

```
  00001111
  10101010
  ----------
  00001010
```

masking

OR

```
  11110000
  10101010
  ----------
  11111010
```

XOR

```
  11111111
  10101010
  ----------
  01010101
```

Συμπλήρωμα

Αριθμητικές και λογικές εντολές

Περιστροφή και αλλαγή θέσης:

- Rotation = μεταφορά των τελευταίων (πρώτων) bits στην αρχή (στο τέλος)

- Shift = μετακίνηση των bits και απώλεια τους
  - Left Shift = πολλαπλασιασμός χ2
  - Right Shift = διαίρεση δια 2

Αριθμητικές Πράξεις

Λειτουργία της CPU

Π.χ., πρόσθεση δύο τιμών αποθηκευμένες στη μνήμη, και αποθήκευση του αποτελέσματος στη μνήμη

Βήμα 1: Παίρνουμε την πρώτη τιμή και την τοποθετούμε σε ένα καταχωρητή

Βήμα 2: Παίρνουμε τη δεύτερη τιμή και την τοποθετούμε σε ένα δεύτερο καταχωρητή

Βήμα 3: Ενεργοποιούμε το κύκλωμα του αθροιστή με εισόδους τους δύο προαναφερθέντες καταχωρητές, και ορίζουμε ως έξοδο ένα τρίτο καταχωρητή για την αποθήκευση του αποτελέσματος

Βήμα 4: Μεταφορά του αποτελέσματος στη μνήμη

Βήμα 5: Τέλος

Εντολές Μηχανής

Περιορισμένος αριθμός εντολών

- **Εντολές μεταφοράς δεδομένων** (1,2,4): Αντιγραφή δεδομένων

  LOAD (Μνήμη → Καταχωρητής) & STORE (Καταχωρητής → Μνήμη)

- **Εντολές αριθμητικής & Λογικής επεξεργασίας** (3): Πράξεις αριθμητικής και AND, OR, XOR, SHIFT, ROTATE

- **Εντολές ελέγχου**: Έλεγχος της εκτέλεσης ενός προγράμματος. JUMP (υπό όρους ή χωρίς όρους)

Εντολές Μηχανής

Παράδειγμα:

Βήμα 1: LOAD την πρώτη τιμή από τη μνήμη σε ένα καταχωρητή

Βήμα 2: LOAD τη δεύτερη τιμή σε ένα δεύτερο καταχωρητή

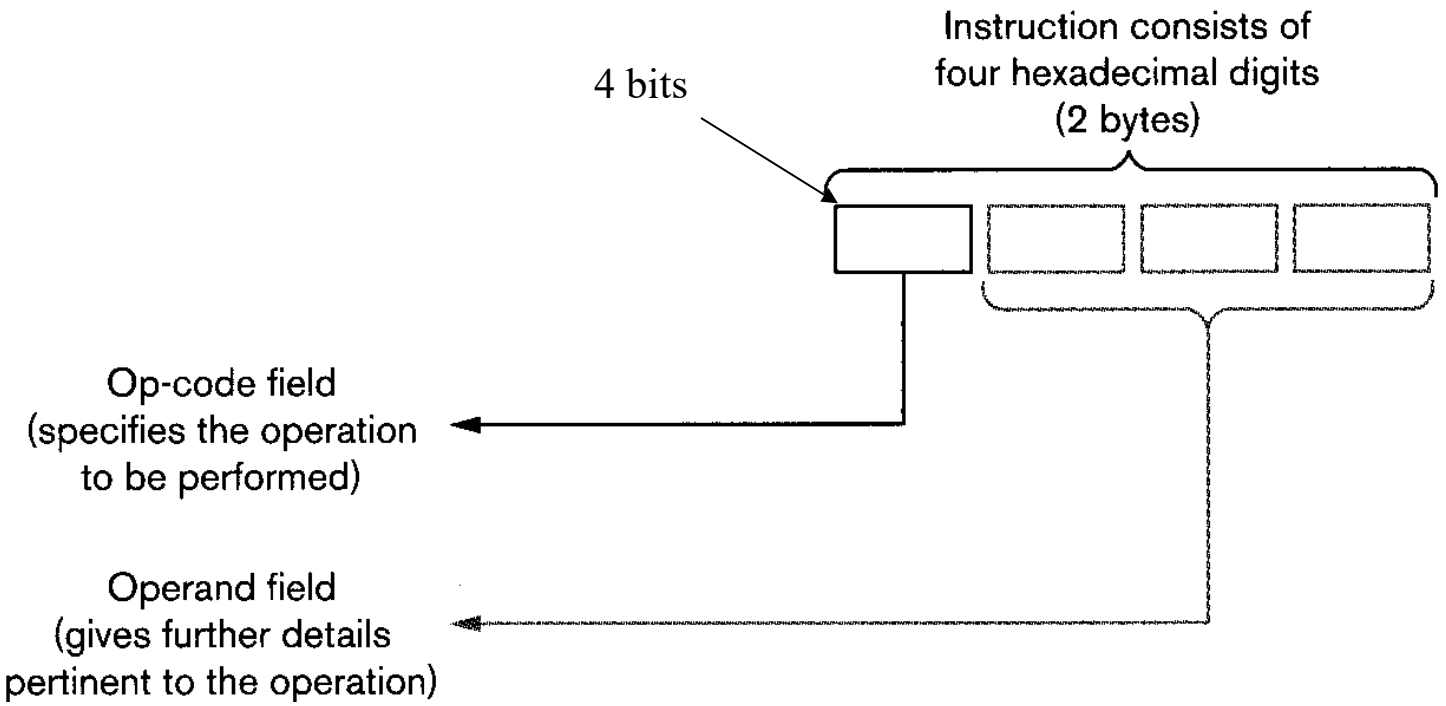
Βήμα 3: Αν η δεύτερη τιμή είναι 0 JUMP στο 6ο βήμα

Βήμα 4: Διαίρεσε το περιεχόμενο του πρώτου καταχωρητή με τα περιεχόμενα του δεύτερου καταχωρητή και τοποθέτησε το αποτέλεσμα σε ένα τρίτο καταχωρητή

Βήμα 5: STORE τα περιεχόμενα του τρίτου καταχωρητή στη μνήμη

Βημα 6: STOP

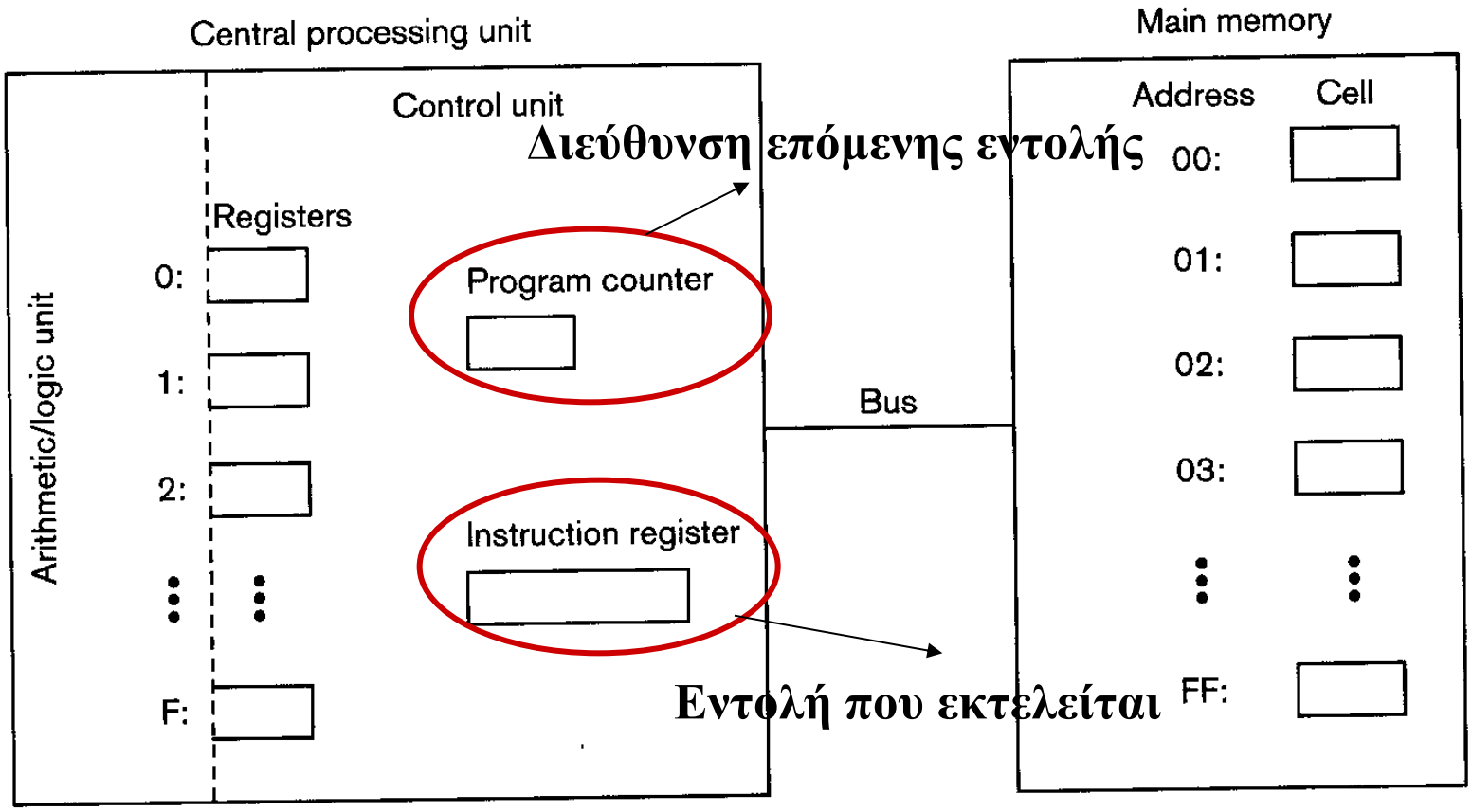
Κωδικοποίηση εντολών μηχανής



Κωδικοποίηση εντολών μηχανής

Op – code	Operand	Description
1	RXY	LOAD register R with contents of memory at address XY (Hex)
2	RXY	LOAD register R with bit pattern XY
3	RXY	STORE contents of R into mem at the address XY
4	0RS	MOVE pattern from register R to register S
5	RST	ADD register S & register T and SAVE result in register R (συμπ. 2)
6	RST	ADD register S & register T and SAVE result in register R (fl point)
7	RST	OR register S & register T and SAVE result in register R
8	RST	AND register S & register T and SAVE result in register R
9	RST	XOR register S & register T and SAVE result in register R
A	R0X	ROTATE register R to the right X times
B	RXY	JUMP to mem XY if R=Reg0
C	000	HALT execution

Η αρχιτεκτονική της μηχανής



Εντολή 1347 → LOAD Register 3 με τα περιεχόμενα της θέσης μνήμης 47

70C5→ Εκτέλεσε το OR στους καταχωρητές C & 5 και αποθήκευσε το αποτέλεσμα στη θέση 0

Παραδείγματα

- **Ποια η διαφορά ανάμεσα στους δύο τύπους LOAD? (15AB? 25AB?)**
- **Τι συμβαίνει στην περίπτωση B058, B258?**

Τι εκτελεί το παρακάτω πρόγραμμα?

156C *LOAD στον R5 το περιεχόμενο του κελιού μνήμης με διεύθυνση 6C*

166D *LOAD στον R6 το περιεχόμενο του κελιού μνήμης με διεύθυνση 6D*

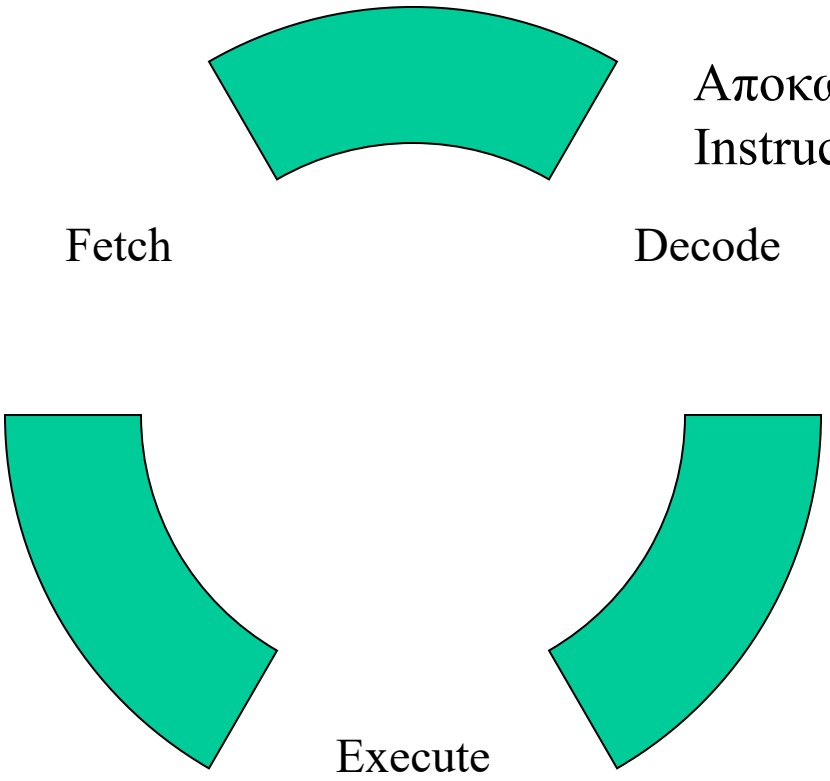
5056 *ADD R5 & R6 και αποθήκευσε στον R0 (συμπλ. ως προς 2)*

306E *Βάλε το αποτέλεσμα των πράξεων στο κελί μνήμης με διεύθυνση 6E*

C000 *HALT*

Ο κύκλος μηχανής

Φέρνουμε την επόμενη εντολή από τη μνήμη (από τα στοιχεία του PC) και αυξάνουμε το PC κατά 2



Εκτέλεση της εντολής στον Instruction Register

Τι ακριβώς γίνεται με την εκτέλεση της εντολής B258?

Παράδειγμα εκτέλεσης προγράμματος

Address	Contents
A0	15
A1	6C
A2	16
A3	6D
A4	50
A5	56
A6	30
A7	6E
A8	C0
A9	00

Πρόγραμμα

156C *LOAD στον R5 το περιεχόμενο της 6C*
166D *LOAD στον R6 το περιεχόμενο της 6D*
5056 *ADD R5 & R6 και αποθήκευσε στον R0 (συμπλ. ως προς 2)*
306E *Βάλε το αποτέλεσμα των πράξεων στη θέση 6E*
C000 *HALT*

Κύκλος Εκτέλεσης

Fetch: 156C → Instruction register & PC= PC+2 (A2)
Decode:
Execute: LOAD R5 with 6C
Fetch: 166D → Instruction register & PC= PC+2 (A4)
Decode:
Execute: LOAD R6 with 6D

Παράδειγμα εκτέλεσης προγράμματος

Address	Contents
A0	15
A1	6C
A2	16
A3	6D
A4	50
A5	56
A6	30
A7	6E
A8	C0
A9	00

Πρόγραμμα

156C *LOAD στον R5 το περιεχόμενο της 6C*
166D *LOAD στον R6 το περιεχόμενο της 6D*
5056 *ADD R5 & R6 και αποθήκευσε στον R0 (συμπλ. ως προς 2)*
306E *Βάλε το αποτέλεσμα των πράξεων στη θέση 6E*
C000 *HALT*

Κύκλος Εκτέλεσης

Fetch: 5056 → Instruction register & PC= PC+2(A6)
Decode:
Execute: ADD R5+R6 (complement 2) → R0
Fetch: 306E → Instruction register & PC= PC+2 (A8)
Decode:
Execute: STORE R0 in 6E
Fetch: C000 → Instruction register & PC= PC+2 (A10)
Decode:
Execute: HALT

Παραδείγματα

Διεύθυνση	Περιεχόμενα
00	14
01	02
02	34
03	17
04	C0
05	00

αποτέλεσμα → Θέση 17: 34 Hex

Διεύθυνση	Περιεχόμενα
B0	13
B1	B8
B2	A3
B3	02
B4	33
B5	B8
B6	C0
B7	00
B8	0F

αποτέλεσμα → 1ο βήμα R3 : 0F Hex
HALT: B8: C3

Παραδείγματα

Διεύθυνση	Περιεχόμενα
A4	20
A5	00
A6	21
A7	03
A8	22
A9	01
AA	B1
AB	B0
AC	50
AD	02
AE	B0
AF	AA
B0	C0
B1	00

- Η τιμή του R0 όταν εκτελείται για πρώτη φορά η AA? 00
- Η τιμή του R0 όταν εκτελείται για δεύτερη φορά η AA? 01
- Πόσες φορές γίνεται εκτέλεση της AA μέχρι να σταματήσει? 4

Παραδείγματα

Διεύθυνση	Περιεχόμενα
F0	20
F1	C0
F2	30
F3	F8
F4	20
F5	00
F6	30
F7	F9
F8	FF
F9	FF

- Τι γίνεται όταν φτάσουμε στην F8? ⟶ HALT