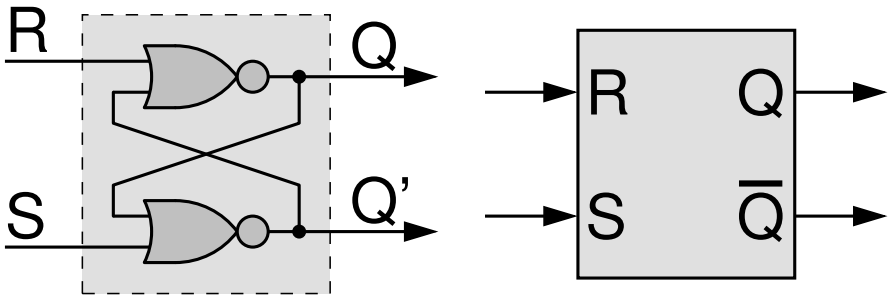
Επανάληψη 3: Ακολουθιακά Κυκλώματα, Μνήμες, FSM, Απλός Υπολογιστής

Φθινόπωρο 2020 – Μανόλης Κατεβαίνης

Μανταλωτής (Latch) τύπου RS (είδος "flip-flop")

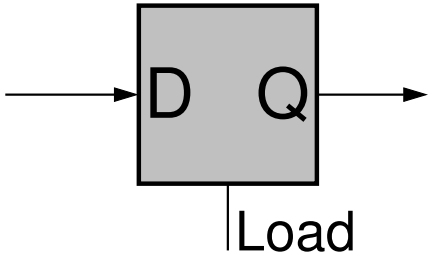


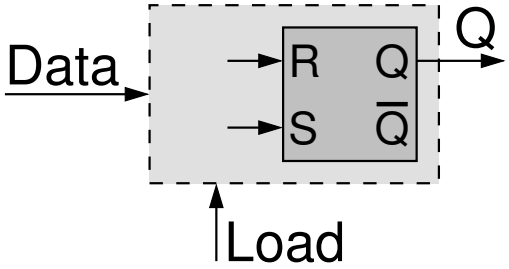
R	S	Q
0	0	«προηγούμενο» Q
0	1	1
1	0	0
1	1	«ανεπιθύμητες» είσοδοι/καταστ.

Ακολουθιακό Κύκλωμα

- *Συνδυαστικό*: οι έξοδοι εξαρτώνται *μόνον* από την *τρέχουσα* τιμή των εισόδων
- *Ακολουθιακό*: οι εξοδοι εξαρτώνται *και από το παρελθόν*
- Πίνακας Αληθείας: ανεπίσημος, μετρίως βολικός
- Εάν R=S=1: «κακό» (Q=Q'=0 (!), επόμενη κατάσταση=??)

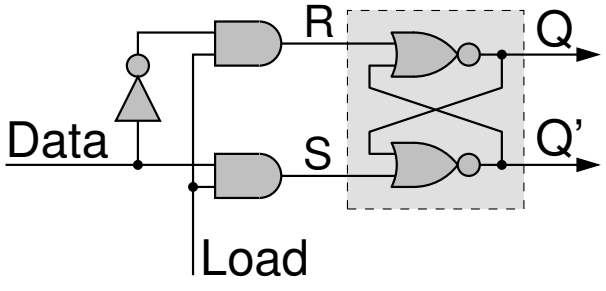
Μανταλωτής (Latch) *τύπου D* (Data - Load)

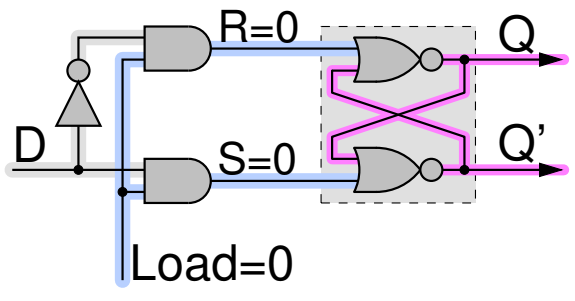


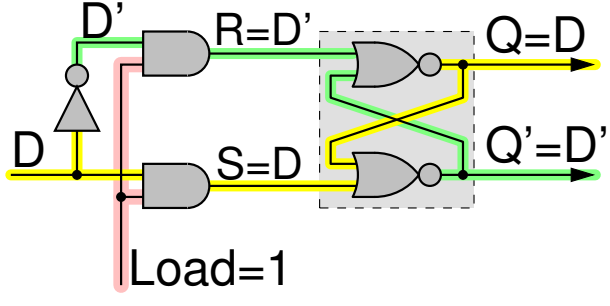


Load	Data	Q	R	S
0	x	«προηγούμενο» Q	0	0
1	0	0	1	0
1	1	1	0	1

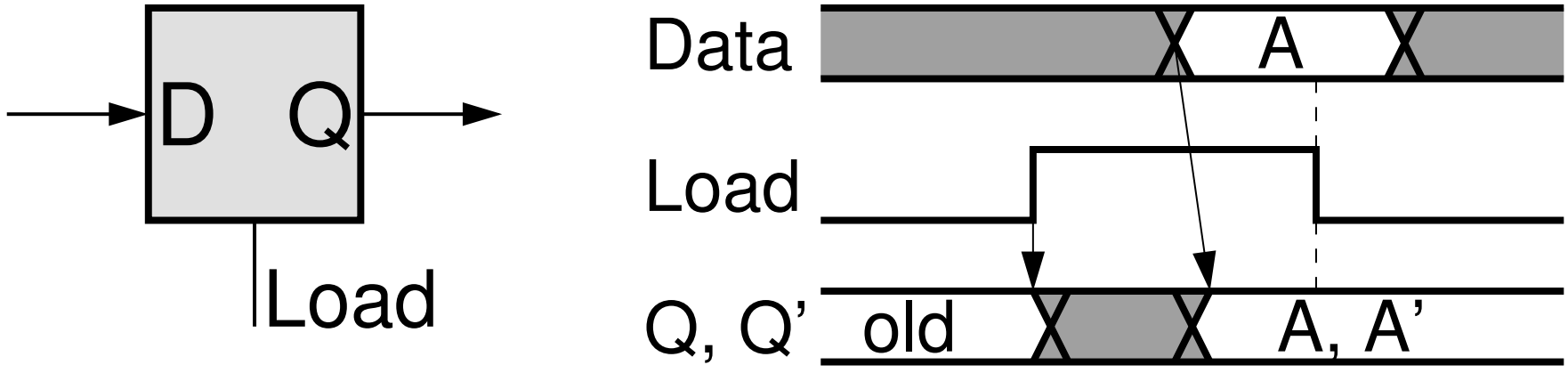
- Χωριστά ο έλεγχος (Load) και τα Data
- Κανείς «ανεπιθύμητος» συνδ. εισόδων







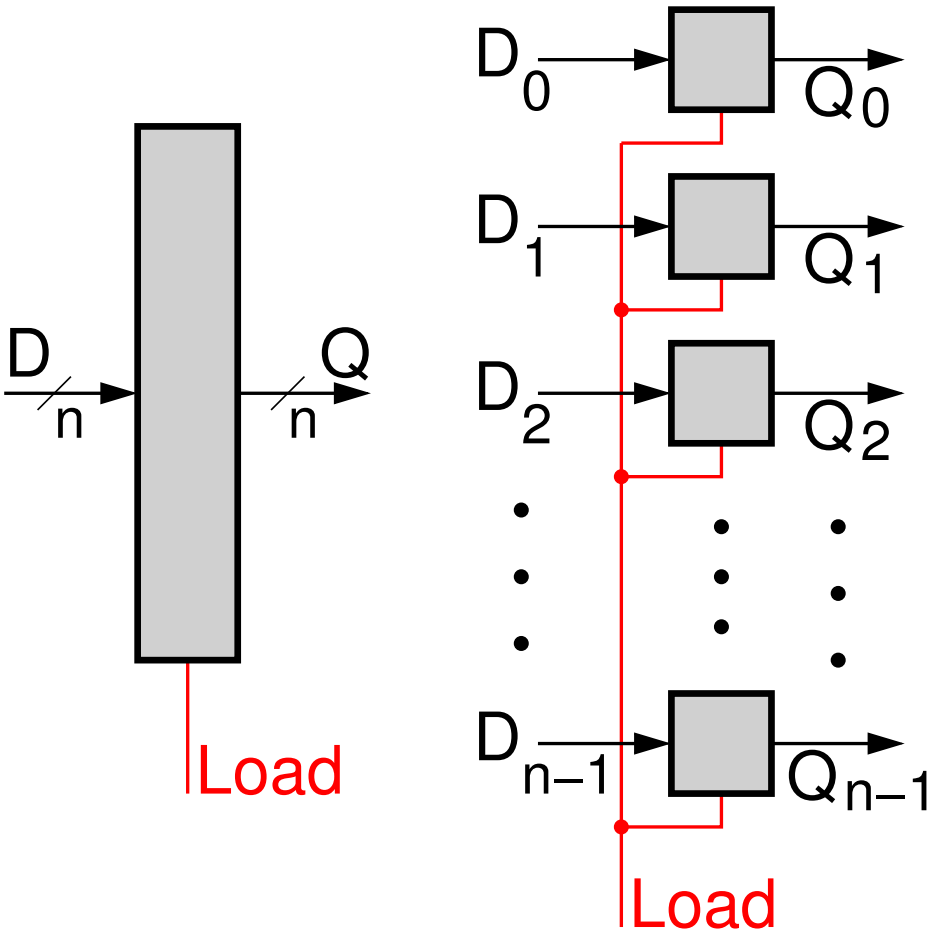
Διάγρ. Χρονισμού Μανταλωτή D – άλλοι συμβολισμοί



- Γκριζαρισμένο διάστημα: τιμή 0 ή 1 ή μεταβαλόμενη
- Γραμμή πάνω-κάτω & «καθαρό» ενδιάμεσα: σταθερά 0 ή σταθερά 1
- Κατακόρυφες ή λίγο πλάγιες γραμμές: αλλαγές μεταξύ 0 και 1

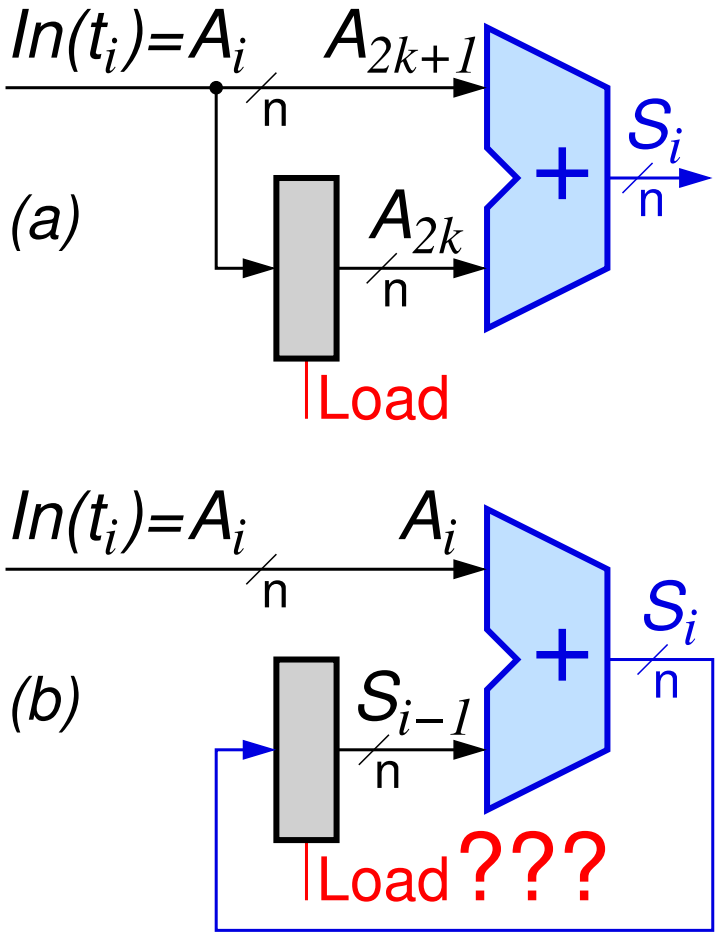
- Όση ώρα Load=1: Έξοδος Q = Είσοδος D
- Όποτε Load=0: Έξοδος Q σταθερή, παλαιά τιμή, ανεξαρ. D

Καταχωρητές (Registers): πολύμπιτοι μανταλωτές



- Εγγραφή (αποθήκευση) μιάς ολόκληρης λέξης (*n* bits) όλης μαζί
- Κοινό σήμα ελέγχου φόρτωσης (*Load*) και γιά τους *n* μανταλωτές
- «Καταχωρητής»: πολύμπιτη λέξη από flip-flops
  - υπάρχουν και άλλων μορφών («ακμοπυροδότητα») flip-flops & καταχ.

Μανταλωτές σε βρόχο: Πρόβλημα Χρονισμού



a) Ο καταχωρητής/μανταλωτής δεν είναι σε βρόχο: έξοδός του δεν επηρεάζει την είσοδό του

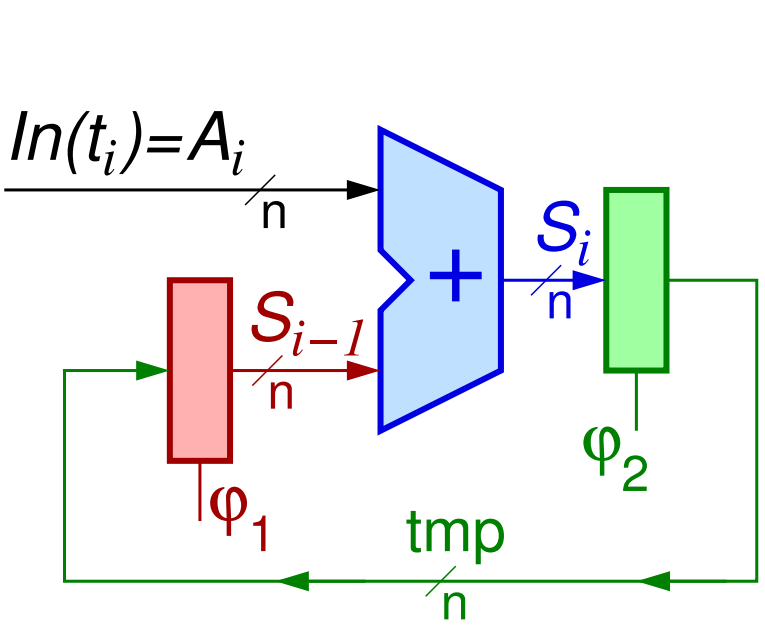
b) Άθροισμα όλων των A_k από $k=0$ έως και το τρέχον i @t_i

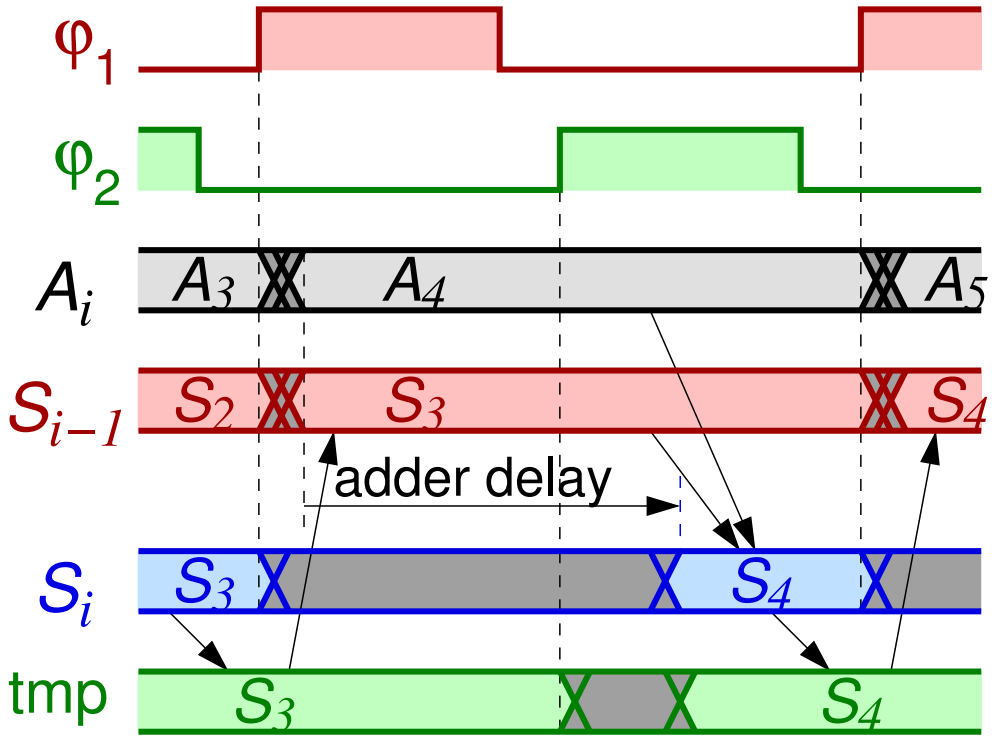
→ Μανταλωτής σε βρόχο!

→ Είσοδός του = *f(* έξοδός του *)*

→ Όταν ανάψει το Load, το περιεχόμενό του *αυτοκαταστρέφεται* από την ανάδραση!!

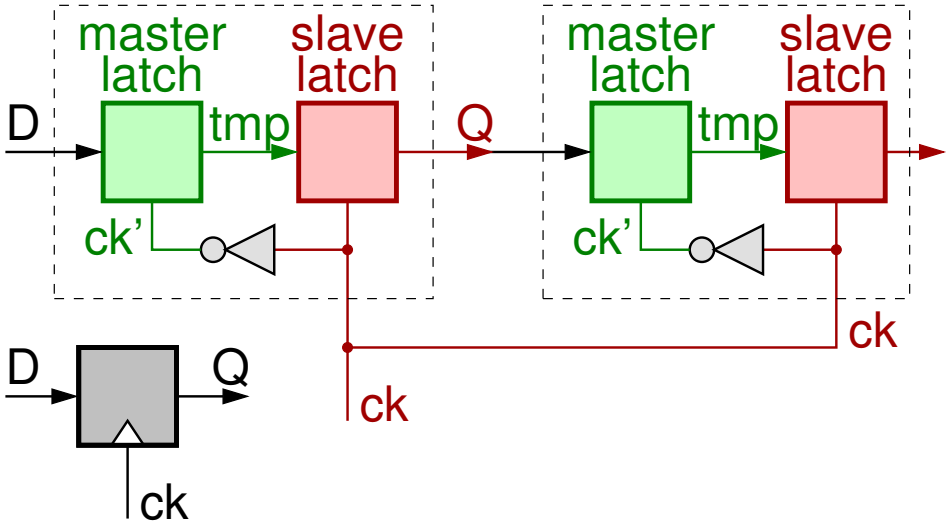
Το Σωρευτικό Άθροισμα με Διφασικό Ρολόϊ

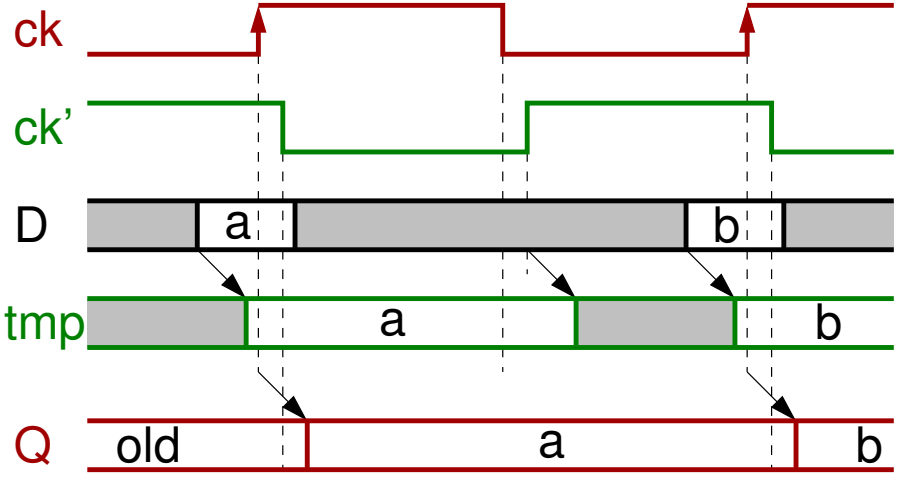




- φ_1: άφιξη νέου αριθμού και προηγ. αθροίσματος στον αθροιστή
- φ_2: αποθήκευση νέου αθροίσματος με ασφάλεια γιά το μέλλον

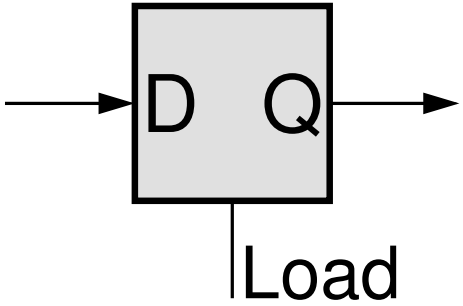
Flip-flop Αφέντη-Σκλάβου – Ακμοπυροδότηση

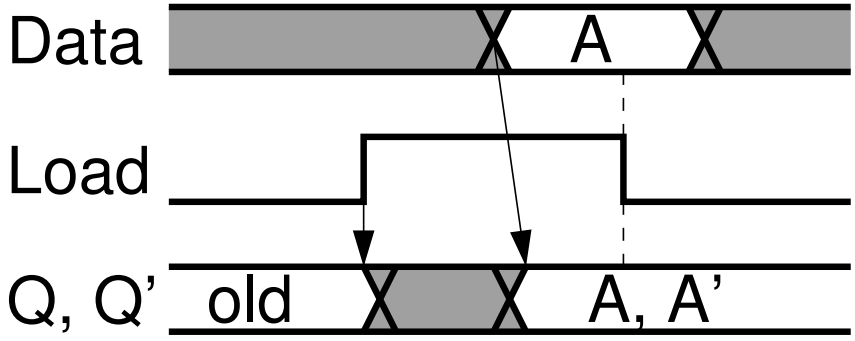


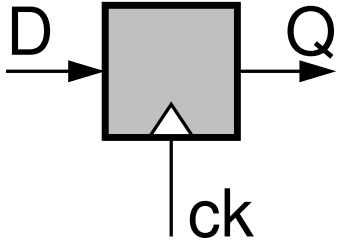


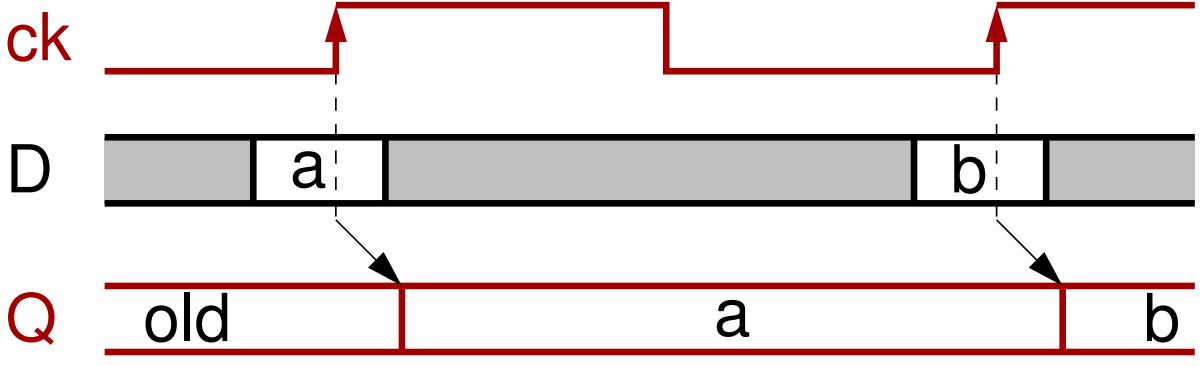
- Μαζί οι μανταλωτές των δύο φάσεων, σε ενιαίο “flip-flop”
- Πολύ προσεκτική δημιουργία της βοηθητικής φάσης, ck’, από την κύρια φάση, ck: ο αντιστροφέας πρέπει να είναι εγγυημένα γρηγορότερος από 2 πύλες του προηγ. σκλάβου

Διαφορές Μανταλωτή – Ακμοπυροδότησης



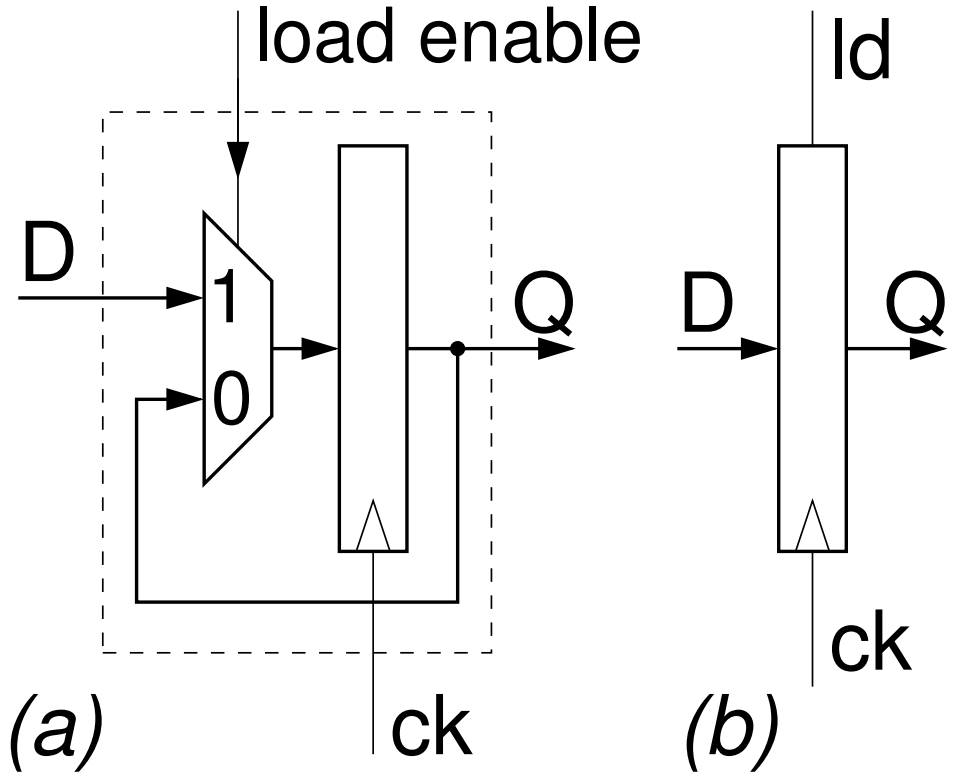






- Μανταλωτής: *καθ' όλη τη διάρκεια* που Load==1
- Ακμοπυροδότηση: *μόνον τη στιγμή* που clock 0 ⟶ 1

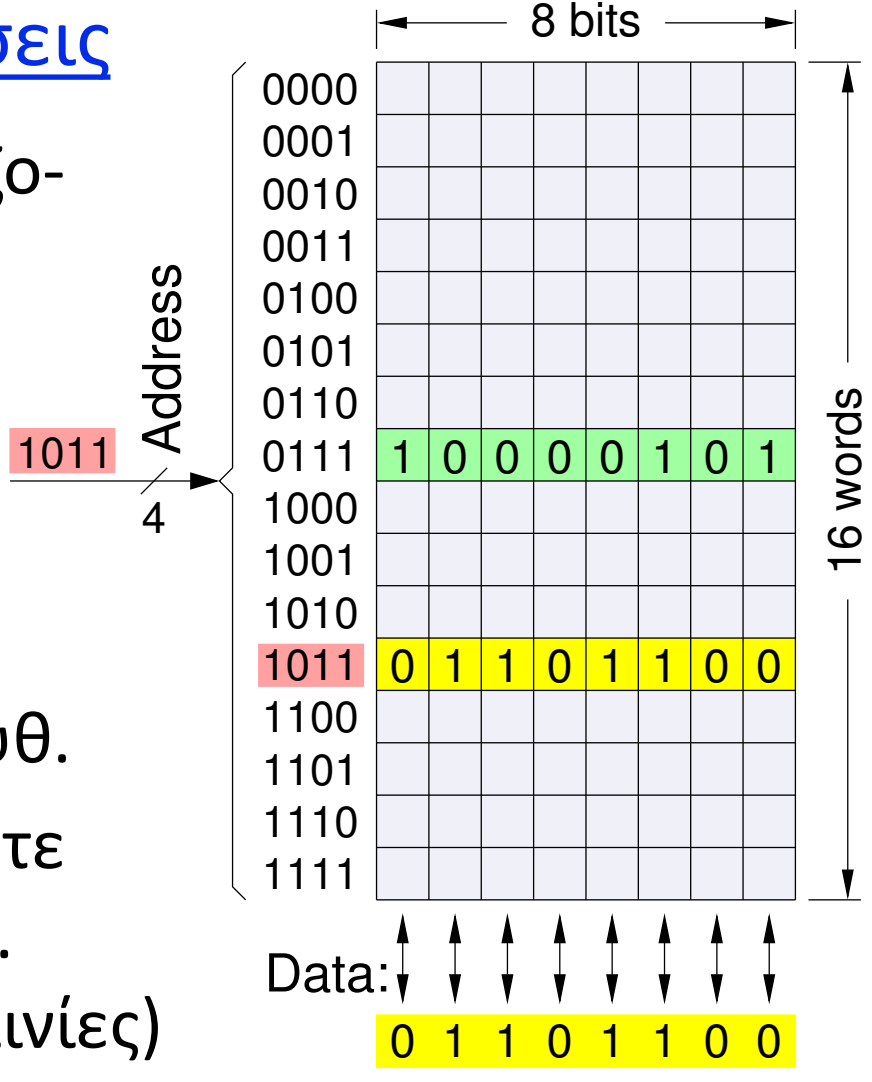
Καταχωρητές με Επίτρεψη Φόρτωσης



- Δεν θέλουμε πάντα όλοι οι ακμοπυροδότητοι καταχωρητές μας να φορτώνουνται στην κάθε ακμή ρολογιού:
- Σήμα ελέγχου σε ποιές ακμές να φορτώνονται και σε ποιές όχι
- Π.χ. ο ACC στις εντολές *Store, Jump* να διατηρεί το περιεχόμενό του

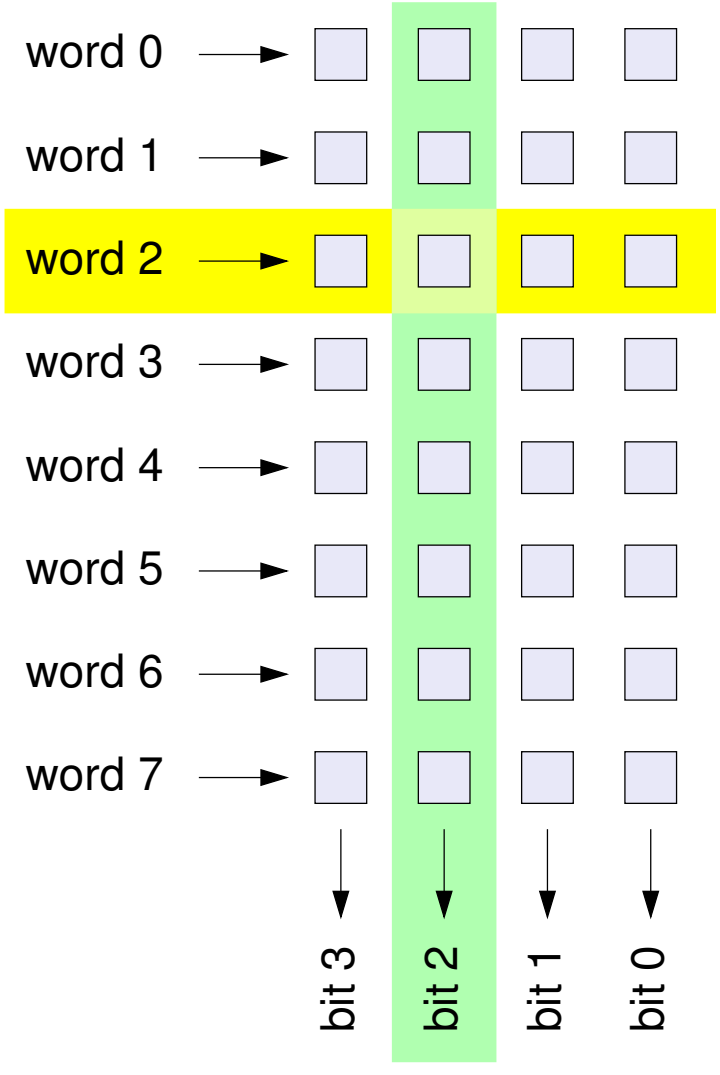
Μνήμες: Λέξεις, Διευθύνσεις

- *Λέξη (Word)*: bits που διαβάζονται ή γράφονται όλα μαζί
- *Διεύθυνση (Address)*: η θέση της λέξης μέσα στη μνήμη
- *Δεδομένα (Data)*: η περιεχόμενη πληροφορία (bits) στη θέση που υποδεικνύει η Διεύθ.
- *Random Access*: την οιαδήποτε διεύθυνση οποτεδήποτε, π.χ. όχι μόνο σειριακά (δίσκοι, ταινίες)

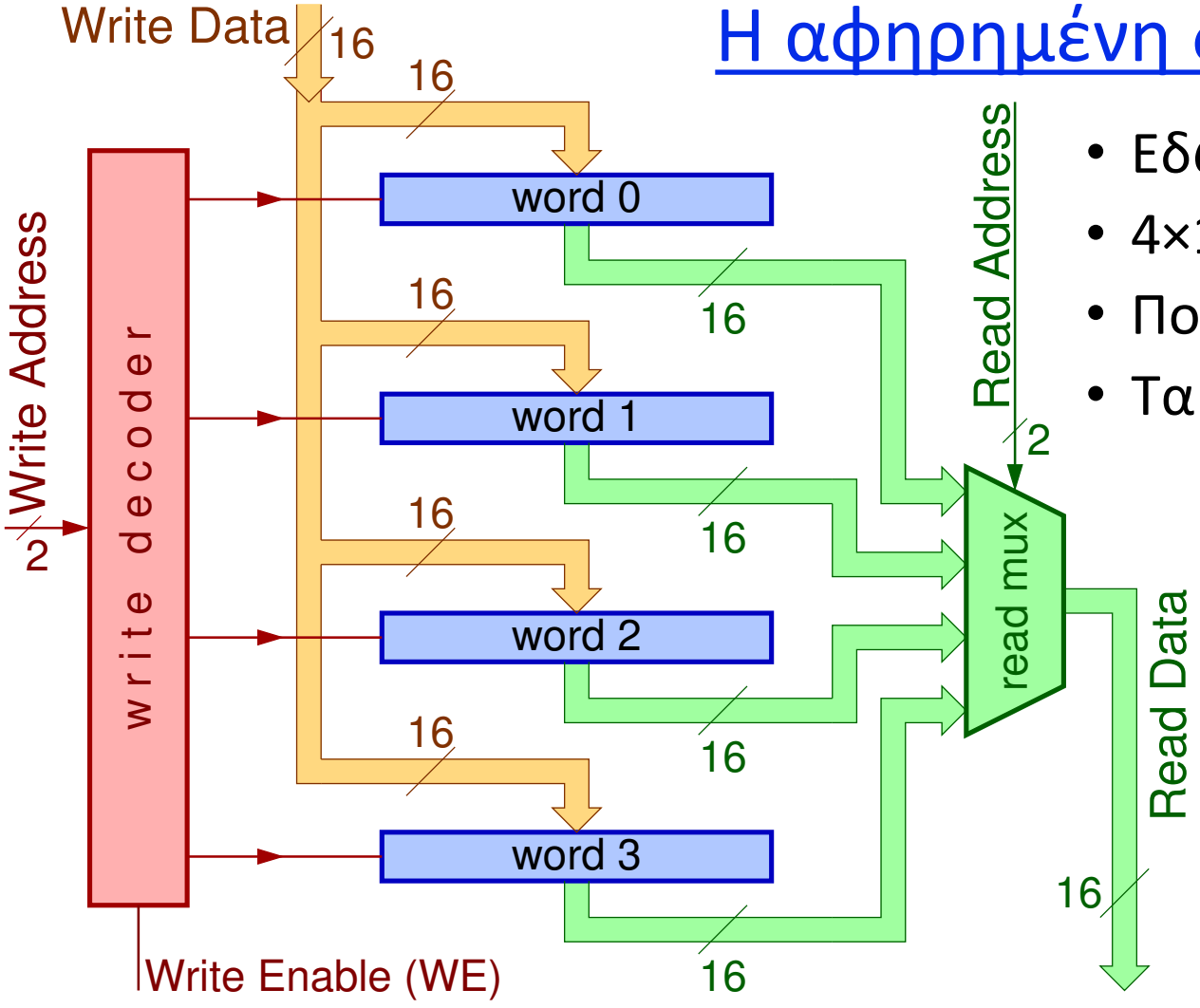


Ανάγνωση = Πολυπλέκτης

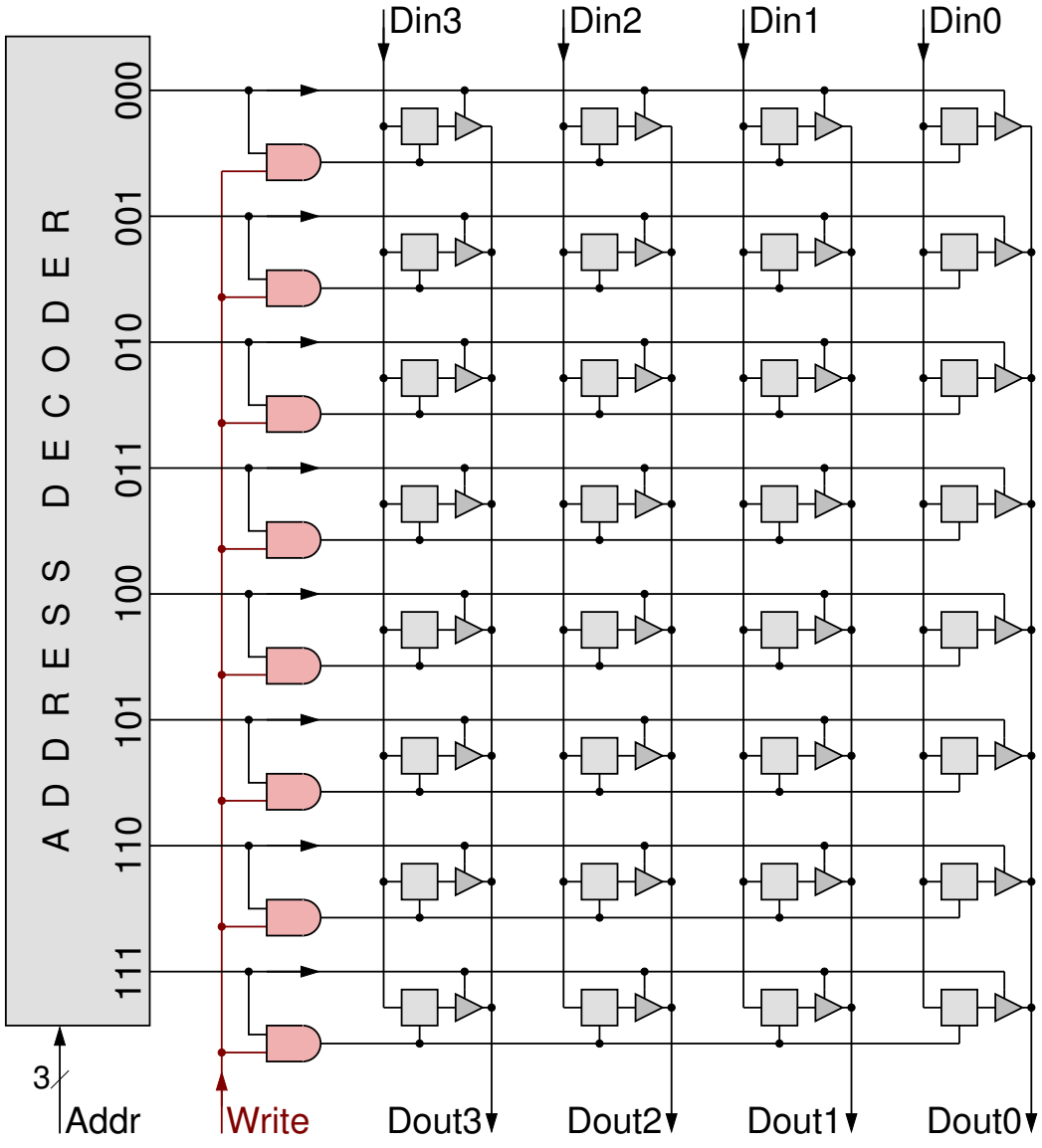
- Επιλογή μιάς από τις λέξεις:
  - τα bits της επιλεγμένης λέξης πρέπει να εμφανιστούν στα σύρματα εξόδου, κάτω
- Πολυπλέκτες Ανάγνωσης:
  - κάθε σύρμα εξόδου δεδομένων επιλέγει ένα από τα bits της στήλης του γιά να το εμφανίσει στον έξω κόσμο (κάτω)
  - κάθε σύρμα εξόδου είναι ένας *πολυπλέκτης*



Η αφηρημένη δομή της RAM



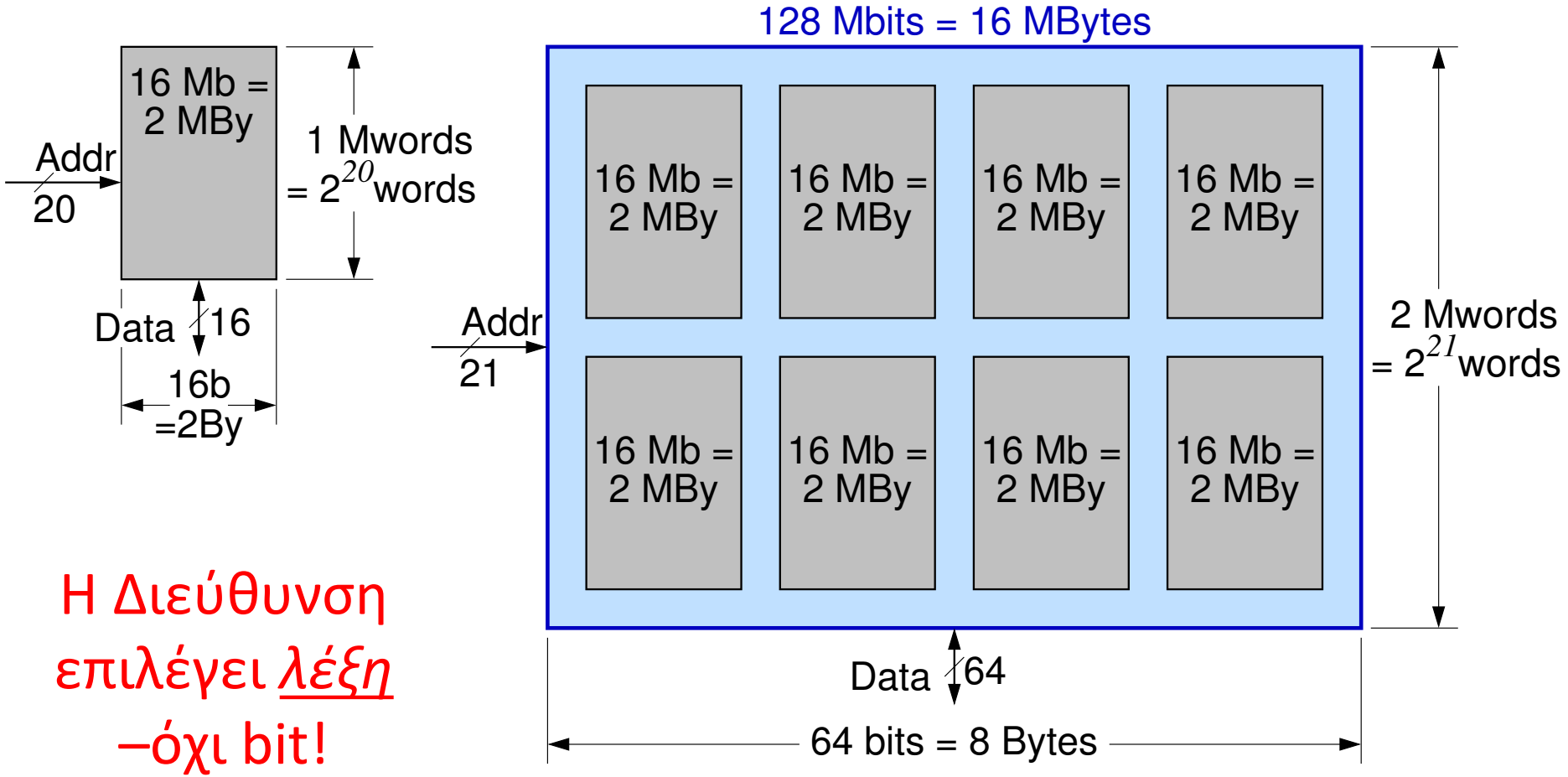
- Εδώ 4 λέξεις × 16 bits/w
- 4×16 = 64 μανταλωτές
- Πολυπλέκτης Ανάγνωσης
- Τα δεδομένα εγγραφής τα βλέπουν όλες οι λέξεις, αλλά:
  - Γράφονται το πολύ σε μία λέξη
  - Έλεγχος εγγραφής βάσει Διεύθυνσης *και* Επίτρεψης Εγγραφής, *WE*



Όλη η Μνήμη

- Ίδια: 8 λέξεις ×4 bits/w
- Κοινός αποκωδικοπ. διευθύνσεων γιά αναγνώσεις και εγγραφές
  - μόνο μία προσπέλαση κάθε φορά – είτε ανάγνωση, είτε εγγραφή
- Ανάγκη γιά σήμα Επίτρεψης Εγγραφής (Write Enable – *WE*)

Μεγέθη Μνημών, πλήθος bits Διεύθυνσης



Η Διεύθυνση επιλέγει *λέξη* –όχι bit!

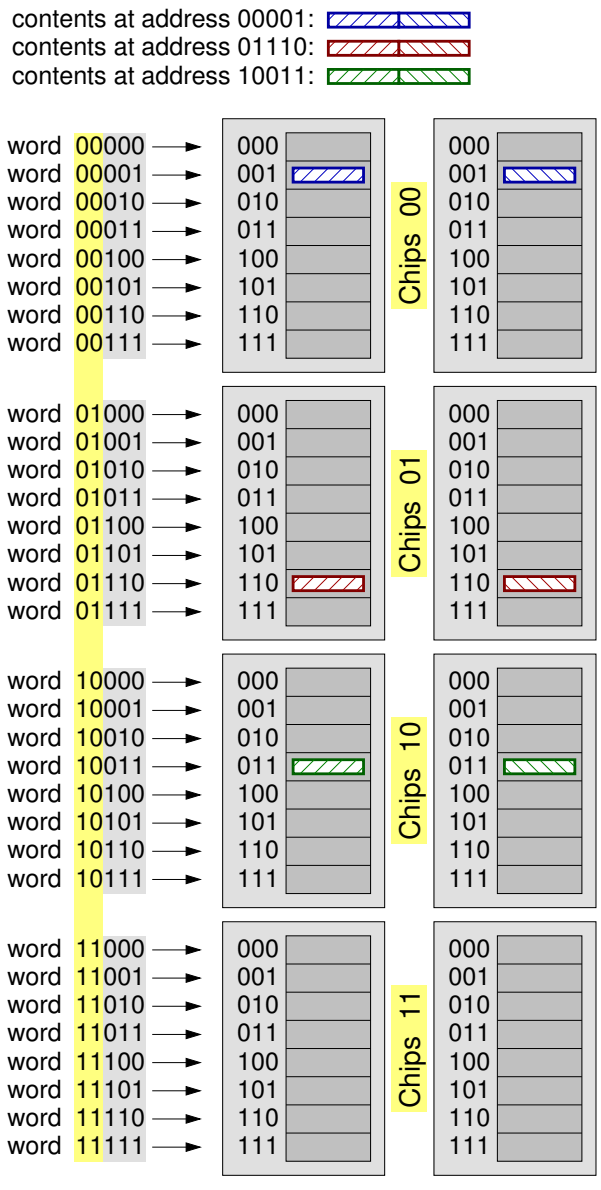
n bits $\Rightarrow$ 2^n Συνδυασμοί

και: 1 By (Byte) = 8 b (bits)

n	2^n
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512

n	2^n
10	1024 = 1 K
11	2 K
12	4 K
13	8 K
14	16 K
15	32 K
16	64 K
17	128 K
18	256 K
19	512 K

n	2^n
20	1,048,576 = 1 M (Mega)
21	2 M
28	256 M
30	1024 M = 1 G (Giga)
32	4 G
40	1024 G = 1 T (Tera)
50	1024 T = 1 P (Peta)
60	1024 P = 1 E (Exa)

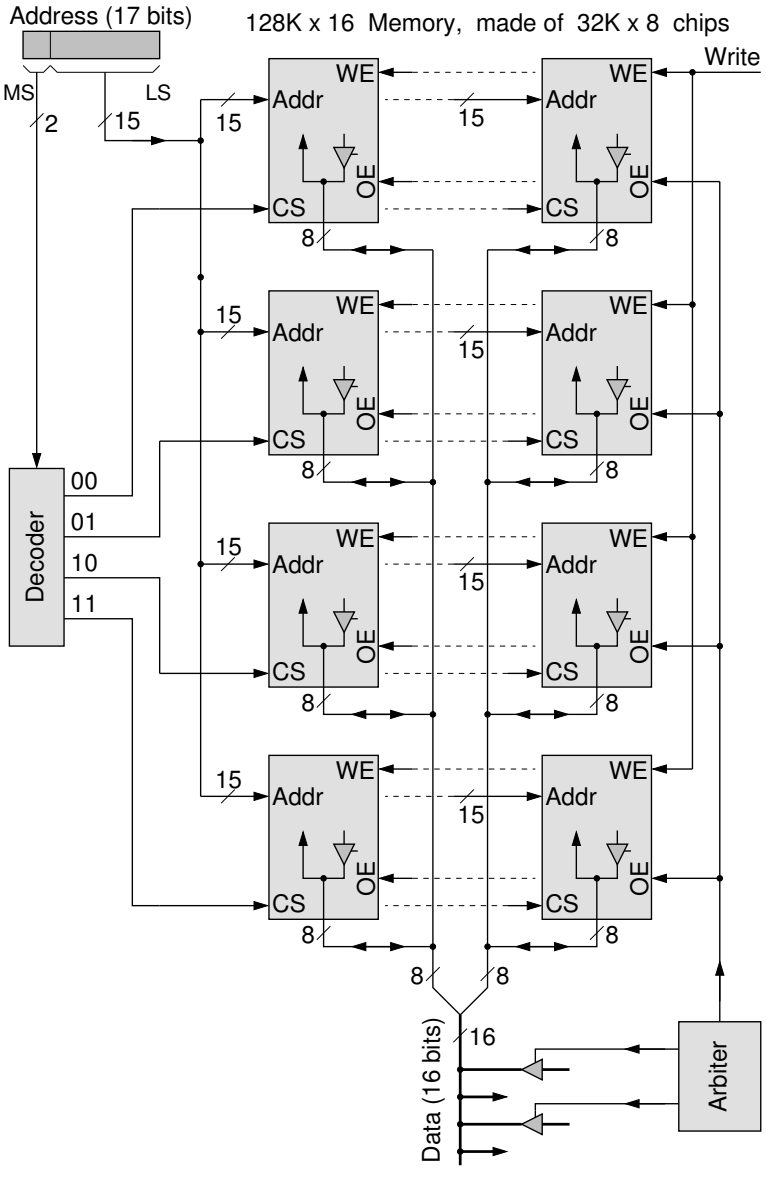


Μνήμη πολλαπλών Chips

- Κάθε chip: 8 λέξεις × *n* bits/word
- Ολόκληρη: 32 λέξεις × *2n* bits/word

⇒ 4 «όροφοι» από chips κατακόρυφα
2 chips «πλάτος» οριζόντια
  - ανάγν./εγγρ. σε *δύο* chips κάθε φορά

- Διεύθυνση ολόκληρης της μνήμης = 5 bits (γιά 32 = 2^5 λέξεις):
  - 2 MS bits: ποιόν «όροφο» ενεργοπ.
  - 3 LS bits: ποιά λέξη μέσα στο κάθε ενεργοποιημένο chip

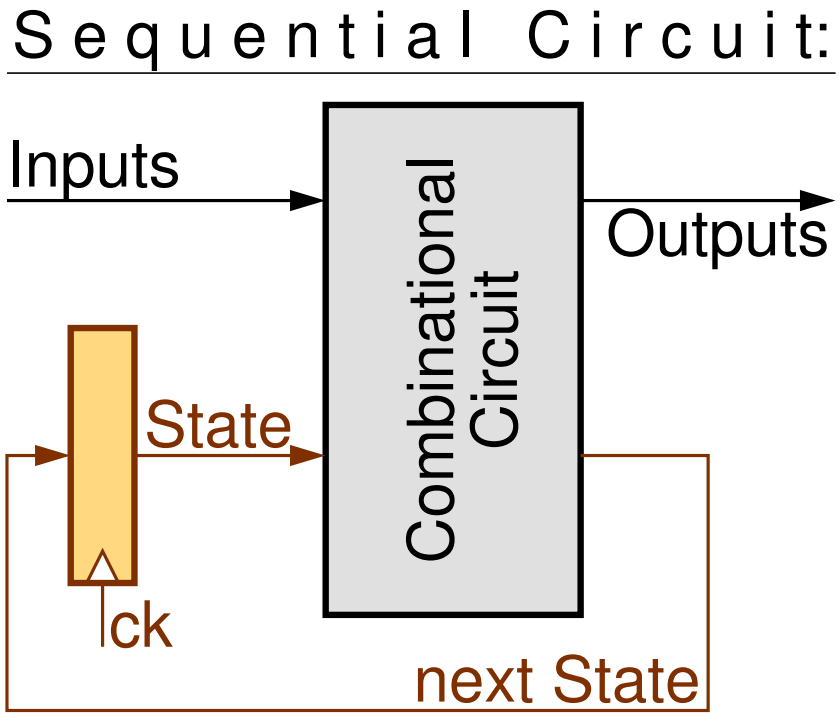


Μνήμη πολλαπλών Chips με τρικατάστατα Data pins

- Παράδειγμα με τα chips PD43256B του Εργαστηρίου: 32K x 8 το καθένα
- Ολική μνήμη 128K x 16 (128=4x32, 16=2x8)
- Data pins συνδέονται απευθείας
- LS bits διεύθυνσης (όσα χρειάζονται) κοινά γιά όλα τα chips
- Chip Select (CS) από την αποκωδικοποίηση των MS bits διεύθυνσης
- Write Enable (WE) κοινό γιά όλους, χάρις στο εσωτερικό **AND** με CS
- Ομοίως γιά Output Enable (OE)

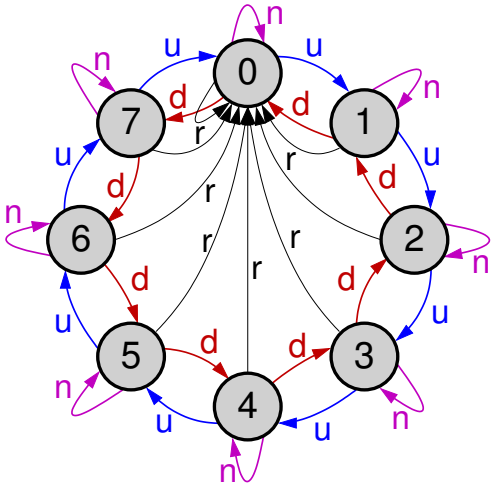
FSM: τρόπος να βλέπουμε τα Ακολουθιακά Κυκλώματα

- Μηχανή Πεπερ. Καταστ.
- Διαχωρισμός στοιχείων μνήμης από συνδυαστικά
- *Κατάσταση* (*State*) = ό,τι θυμάται από το παρελθόν
  - Ξεκινάμε ορίζοντας αυτές & *σημασία τους πολύ προσεκτ*.
- Η συμπεριφορά (έξοδοι και επόμενη κατάσταση) εξαρτάται από το τι βλέπουμε τώρα (είσοδοι) σε συνδυασμό με το τι θυμόμαστε από το παρελθόν



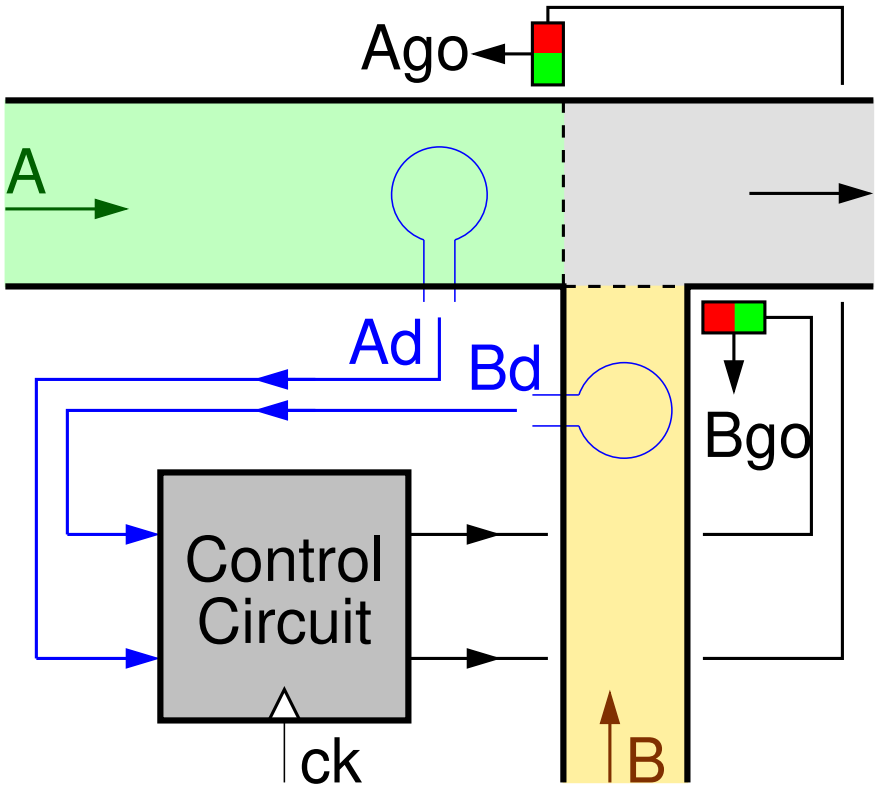
Διάγραμμα (Μετάβασης) Καταστάσεων μιάς FSM

- Ένας κύκλος γιά την κάθε κατάσταση
- Ένα βέλος γιά την κάθε κατάσταση ***και*** την κάθε πιθανή επόμενή της
- Ετικέτα πάνω στο βέλος με τις συνθήκες (εισόδους) υπό τις οποίες γίνεται αυτή η μετάβαση στην επόμενη ακμή ρολογιού

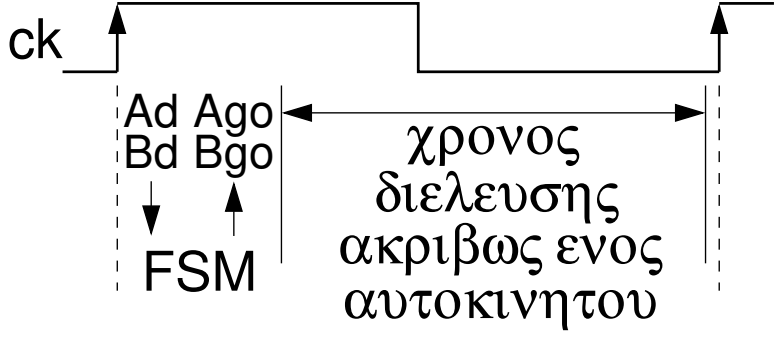


- Οι τιμές των εξόδων μπορούν να γραφτούν:
  - μέσα στις καταστάσεις, εάν εξαρτώνται μόνον από την κατάσταση
  - πάνω στα βέλη (π.χ. δεξιότερα), εάν εξαρτώνται και από εισόδους
- Συνολικά το διάγραμμα ισοδυναμεί με τον Πίνακα Αληθείας του Συνδυαστικού μέρους της FSM, αλλά πιό κατανοητό

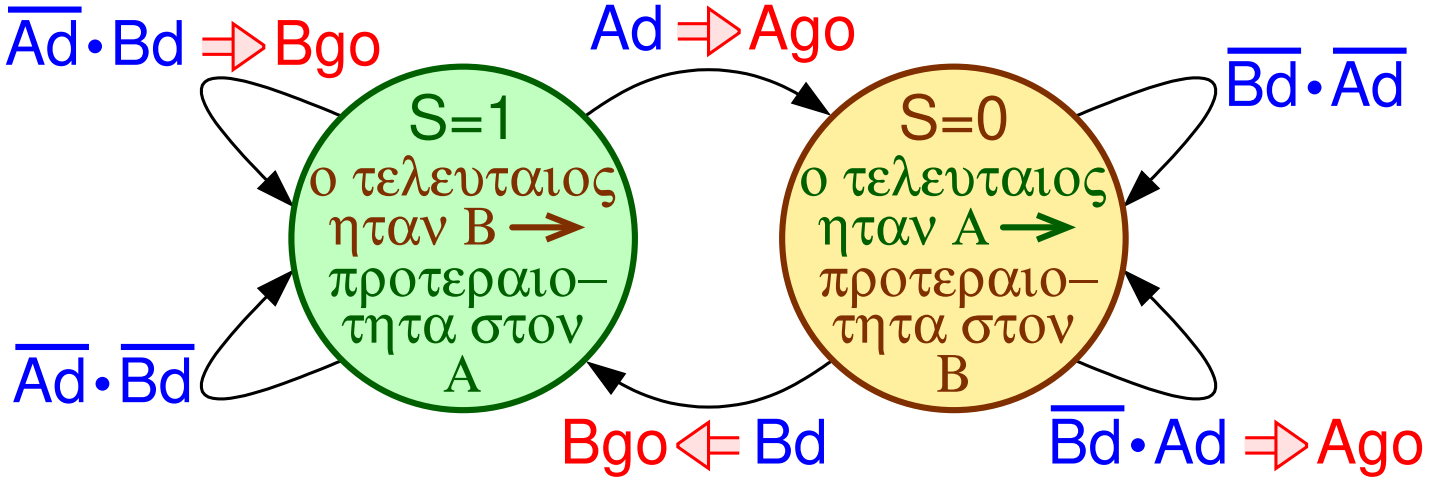
Παράδειγμα: Φωτεινοί Σηματοδότες σε Διασταύρωση



- Διασταύρωση δύο μονόδρομων
- Α: κεντρικός, Β: δευτερεύων
- Ανιχνευτές κίνησης Ad, Bd
- Φωτεινοί σηματοδότες Ago, Bgo
- Απλοποιητική παραδοχή χρονισμού:

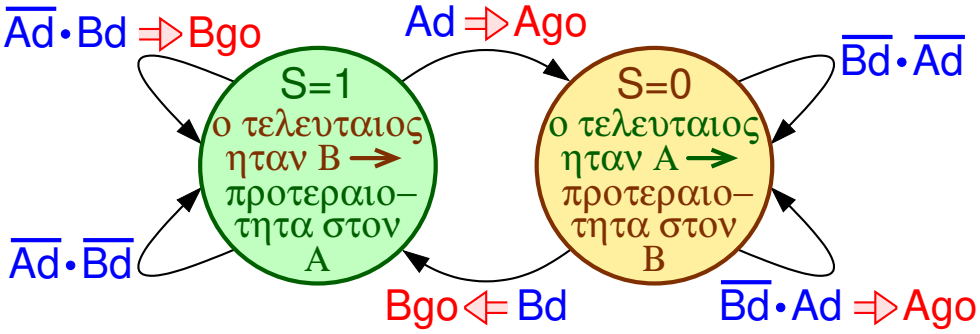


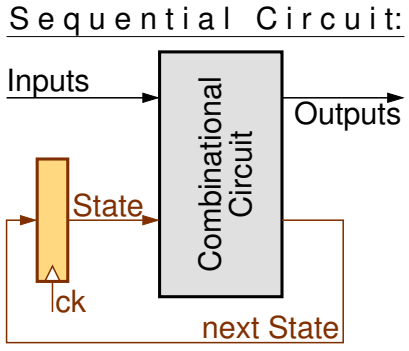
Εναλλαγή Προτεραιοτήτων (αναλ. 1:1) – *Round-Robin*



- Μετά από ένα αυτοκίνητο που περνά από έναν δρόμο, προτεραι-ότητα έχει ο άλλος δρόμος – *Round-Robin / Γύρω-Γύρω Όλοι*
- Εάν υπάρχει αυτοκ. στον δρόμο προτεραιότητας, τότε αυτό περνά
- Αλλιώς, εάν υπάρχει στον άλλον, περνά από εκεί (work conserving)
- Επόμενη κατάσταση = από εκεί που πέρασε αυτοκ., αλλιώς η ίδια

Κύκλωμα της FSM γιά Round-Robin με αναλογία 1:1

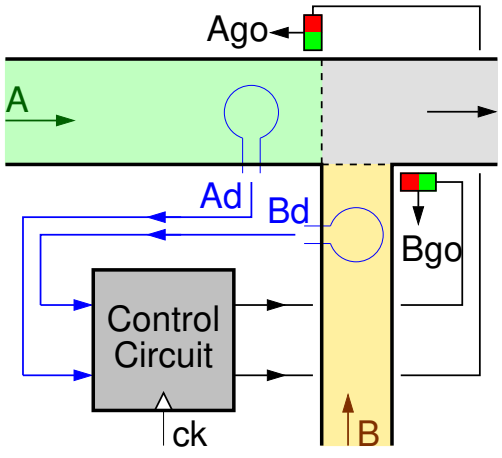




S	Ad	Bd	Ago	Bgo	nS
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	0	0

- Ago = Ad · (S + Bd')
- Bgo = Bd · (S' + Ad')
- nxtS = S·Ad' + S'·Bd
- Δεν χρειάζεται reset (δεν πειράζει να ξεκινήσει σε οιαδήποτε κατάσταση)

FSM Εναλλαγής Προτεραιοτήτων με αναλογία 2:1



$\overline{Ad} \cdot \overline{Bd}$

S=01
δυο τελ/ταιοι ηταν Α → προτεραιο–τητα στον Β

$\overline{Bd} \cdot Ad \Rightarrow Ago$

$Bd \Rightarrow Bgo$

$Ad \Rightarrow Ago$

$\overline{Ad} \cdot Bd \Rightarrow Bgo$

$\overline{Bd} \cdot \overline{Ad}$

S=11
προτ/ταιος Β τελ. Α → προτεραιο–τητα στον Α

S=10
ο τελευταιος ηταν Β → προτεραιο–τητα στον Α

$\overline{Ad} \cdot \overline{Bd}$

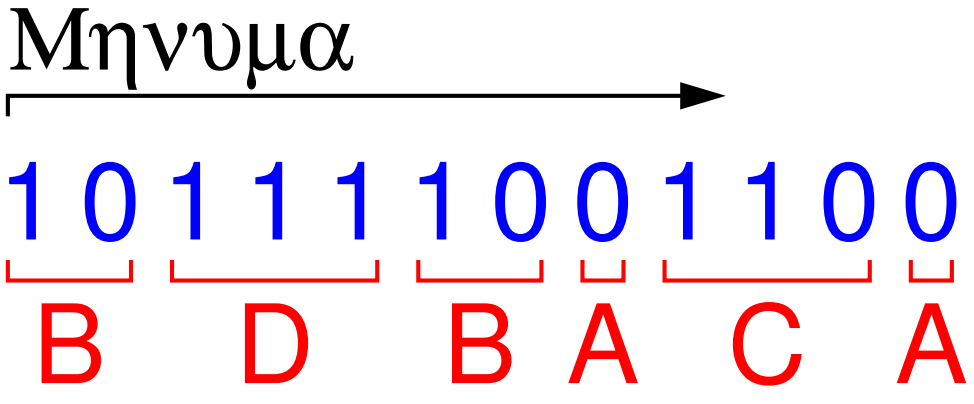
$Ago \Leftarrow Ad$

$\overline{Ad} \cdot Bd \Rightarrow Bgo$

Κώδικες Μεταβλητού Μήκους: Λήψη, Ερμηνεία

Σ’ εμάς η ερμηνεία ήταν απλή, διότι:

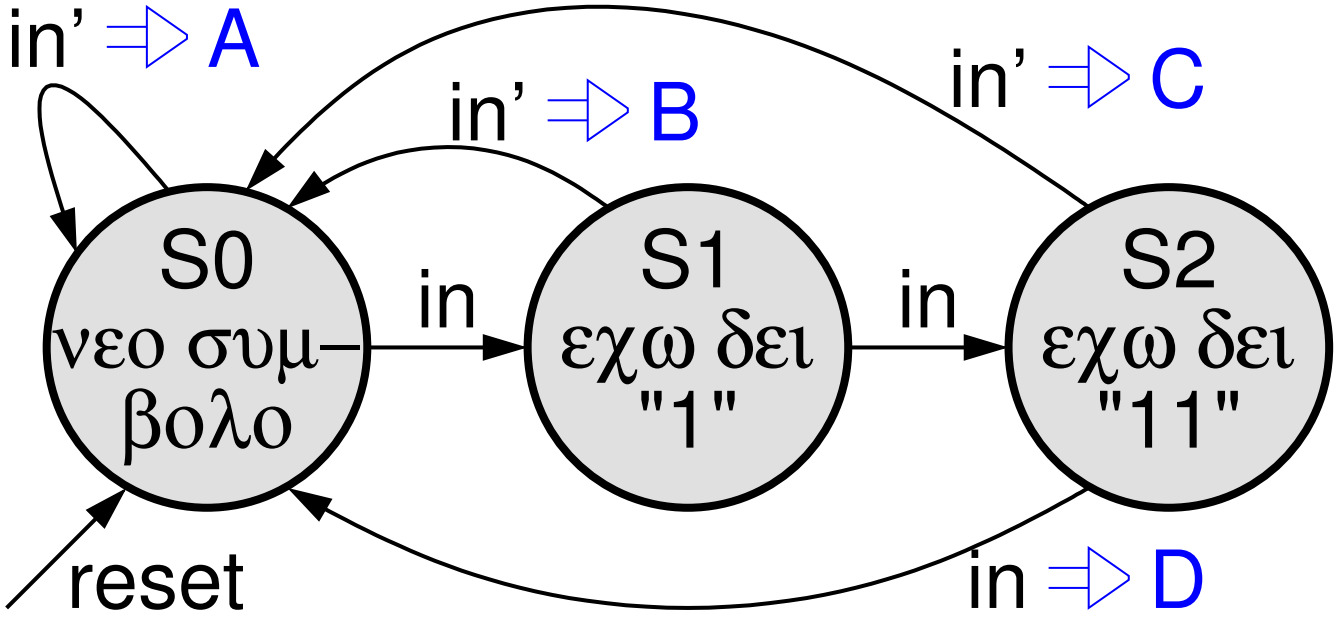
- Κανένα βραχύτερο σύμβολο δεν αποτελεί πρόθεμα (prefix) κανενός μακρύτερου συμβόλου



Απλό Παράδειγμα

Σύμ-βολο	Κώ-δικας
A	0
B	10
C	110
D	111

FSM Αποκωδικοποίησης απλού κώδικα Huffman

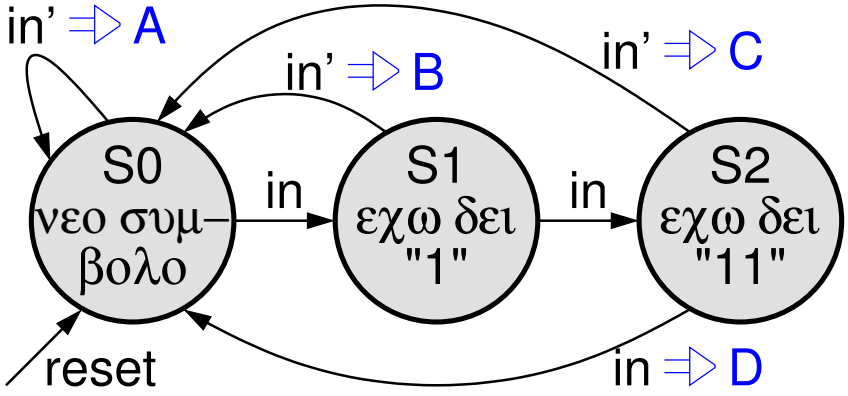


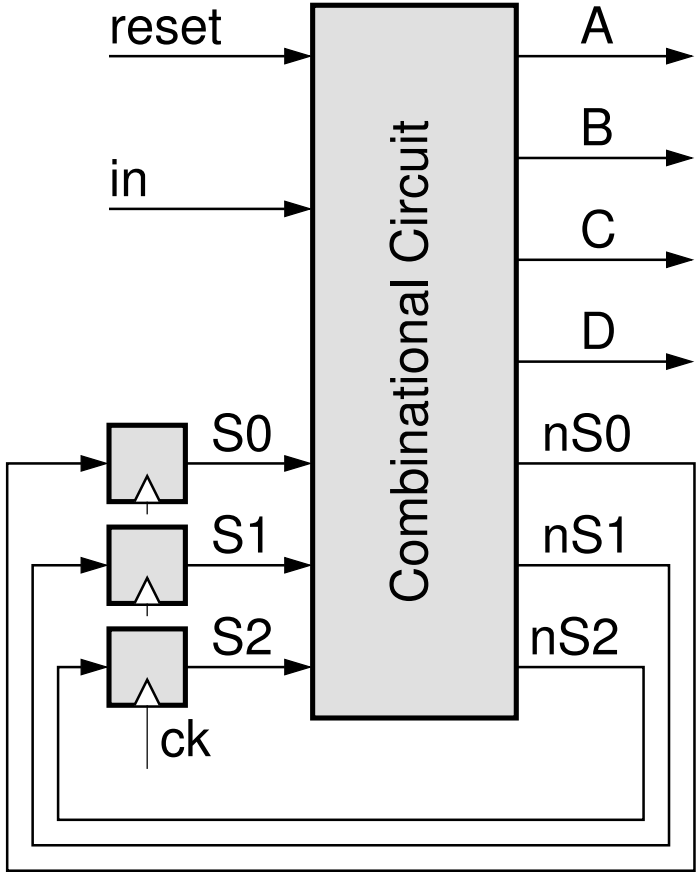
Απλό Παράδειγμα

Σύμ-βολο	Κώ-δικας
A	0
B	10
C	110
D	111

- Απαιτείται Reset
- Υπόθεση χρονισμού εξόδων: δηλώνουμε την αναγνώριση συμβόλου στον ίδιο κύκλο με το τελευταίο bit του – αλλιώς θα χρειάζονταν πολύ περισσοτ. καταστ.

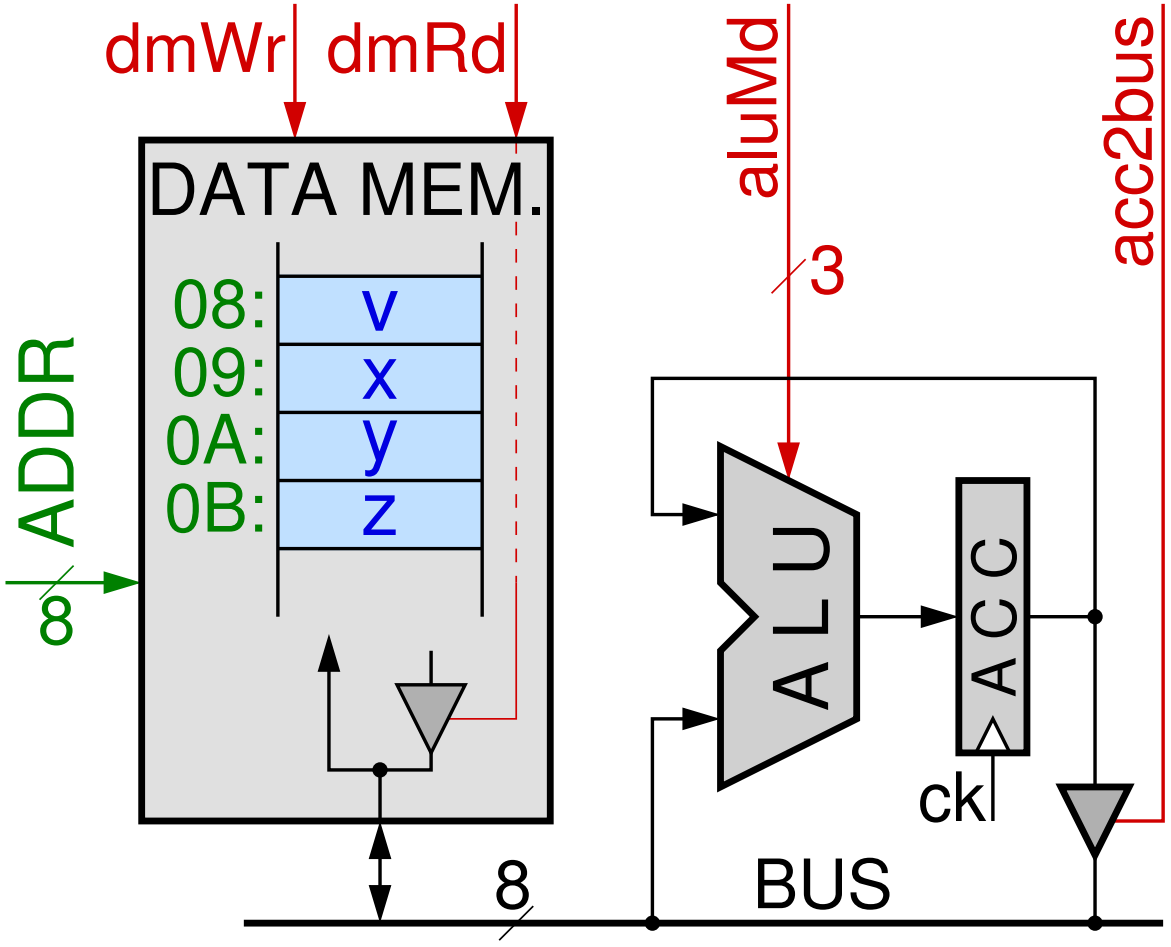
Κωδικοποίηση Καταστάσεων *One-Hot:* η ιδέα





- *One-Hot*: φροντίζουμε πάντα ένα και μόνον ένα από τα flip-flops κατάστασης να ανάβει
- Ισοδύναμο με κλασικά flip-flops κατάστασης ακολουθούμενα από έναν αποκωδικοποιητή

Απλός Υπολογιστής τ. Συσσωρευτή (Accumulator)



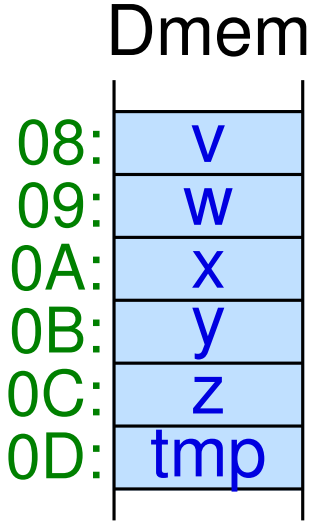
- 2 από τους 3 «τελεστέους» εξυπακούονται
1. Προηγούμενο αποτέλεσμα από τον Συσσωρευτή
2. Πράξη με κάτι νέο από μνήμη
3. Αποτέλεσμα στον Συσσωρευτή

Πολυπλοκότερες εκφράσεις: ενδιάμεσα αποτελέσματα

v = x + y∗z;

(έστω ότι έχουμε εντολή πολλαπλασιασμού: *mul*)

```
load  0B    # y
mul   0C    # z
add   0A    # x
store 08    # v
```

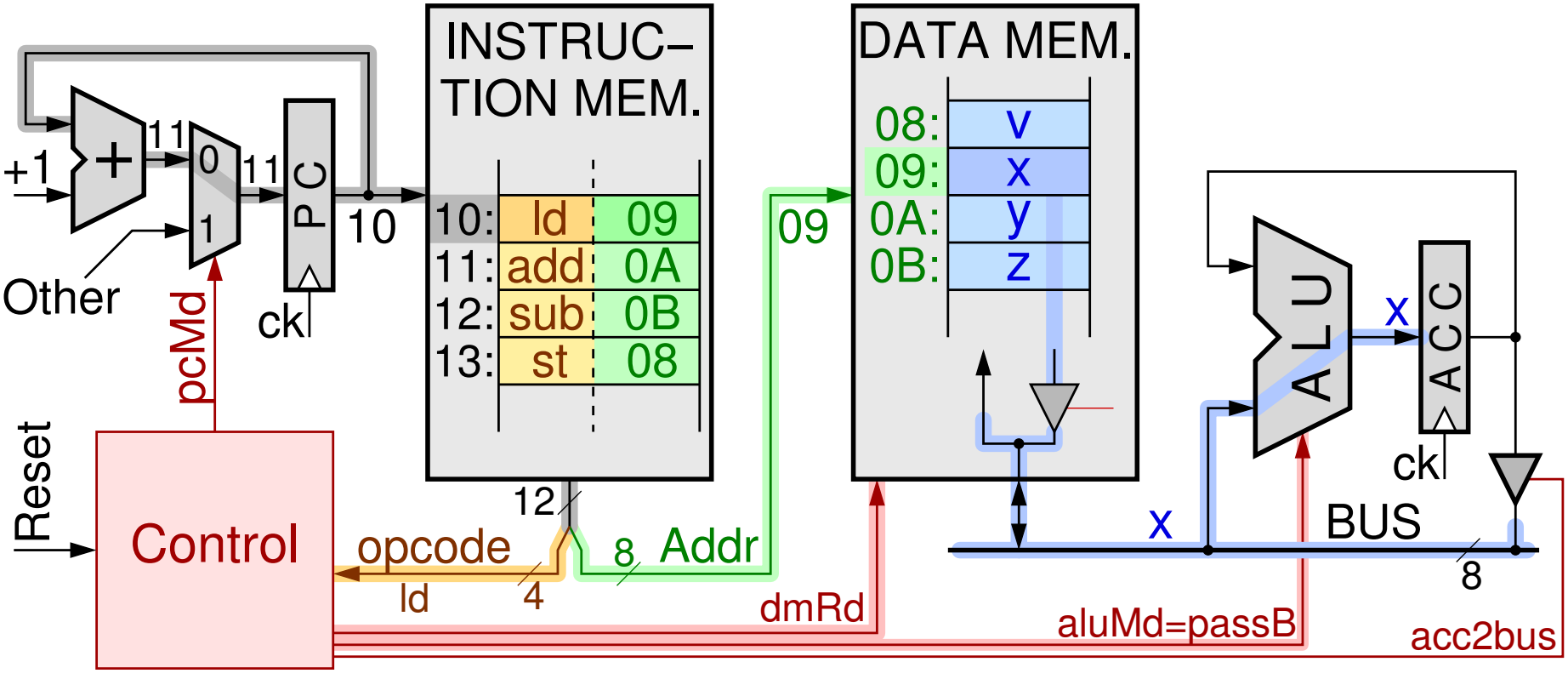


v = x∗w + y∗z;

```
load  0A    # x
mul   09    # w
store 0D    # tmp
load  0B    # y
mul   0C    # z
add   0D    # tmp
store 08    # v
```

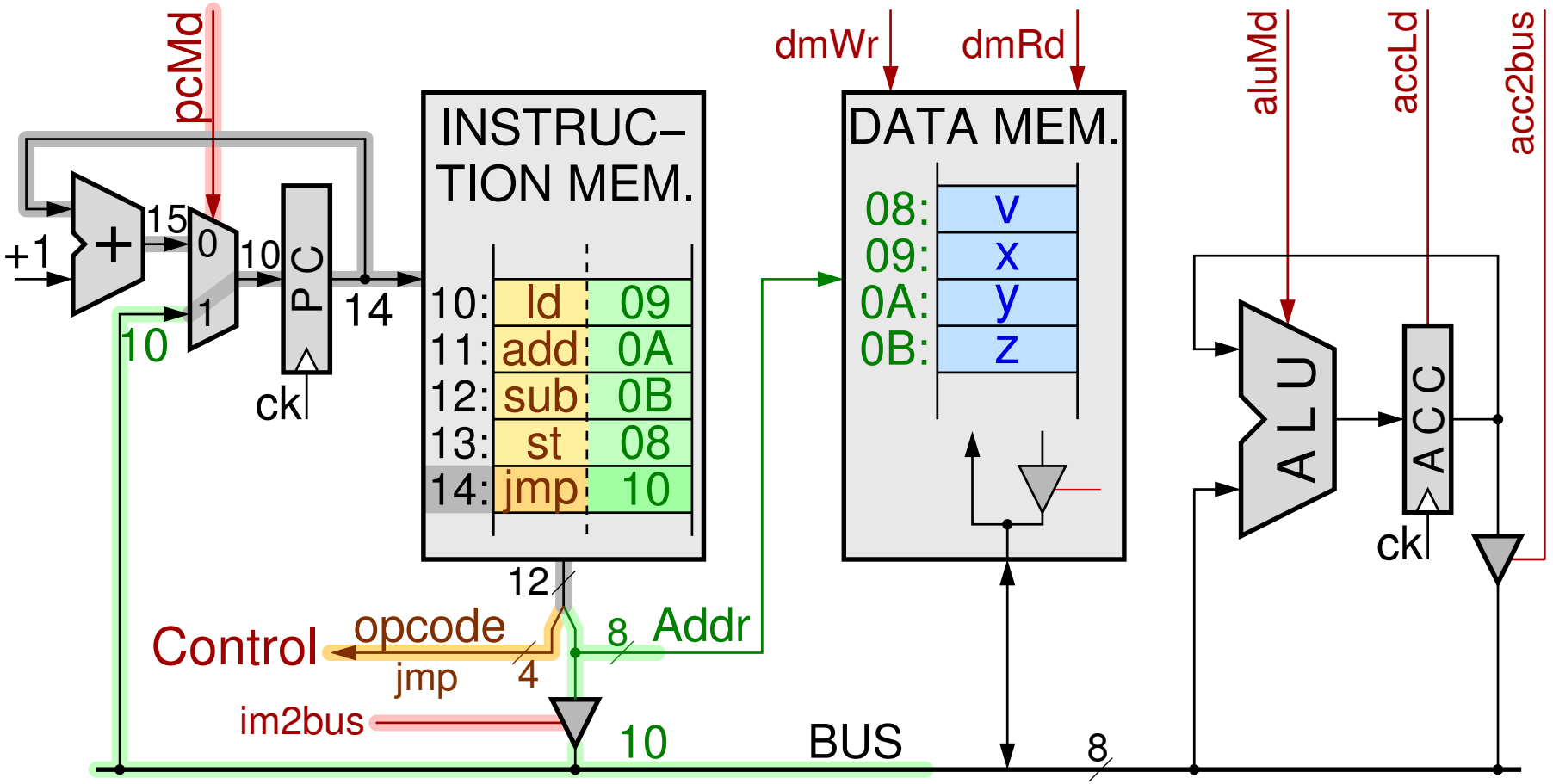
(θα μπορούσε και το *v* να παίξει το ρόλο του *tmp*, αφού δεν εμφανίζεται το *v* στη δεξιά πλευρά της εκχώρησης)

Εντολές: Τι να κάνουμε, σε ποιόν να το κάνουμε

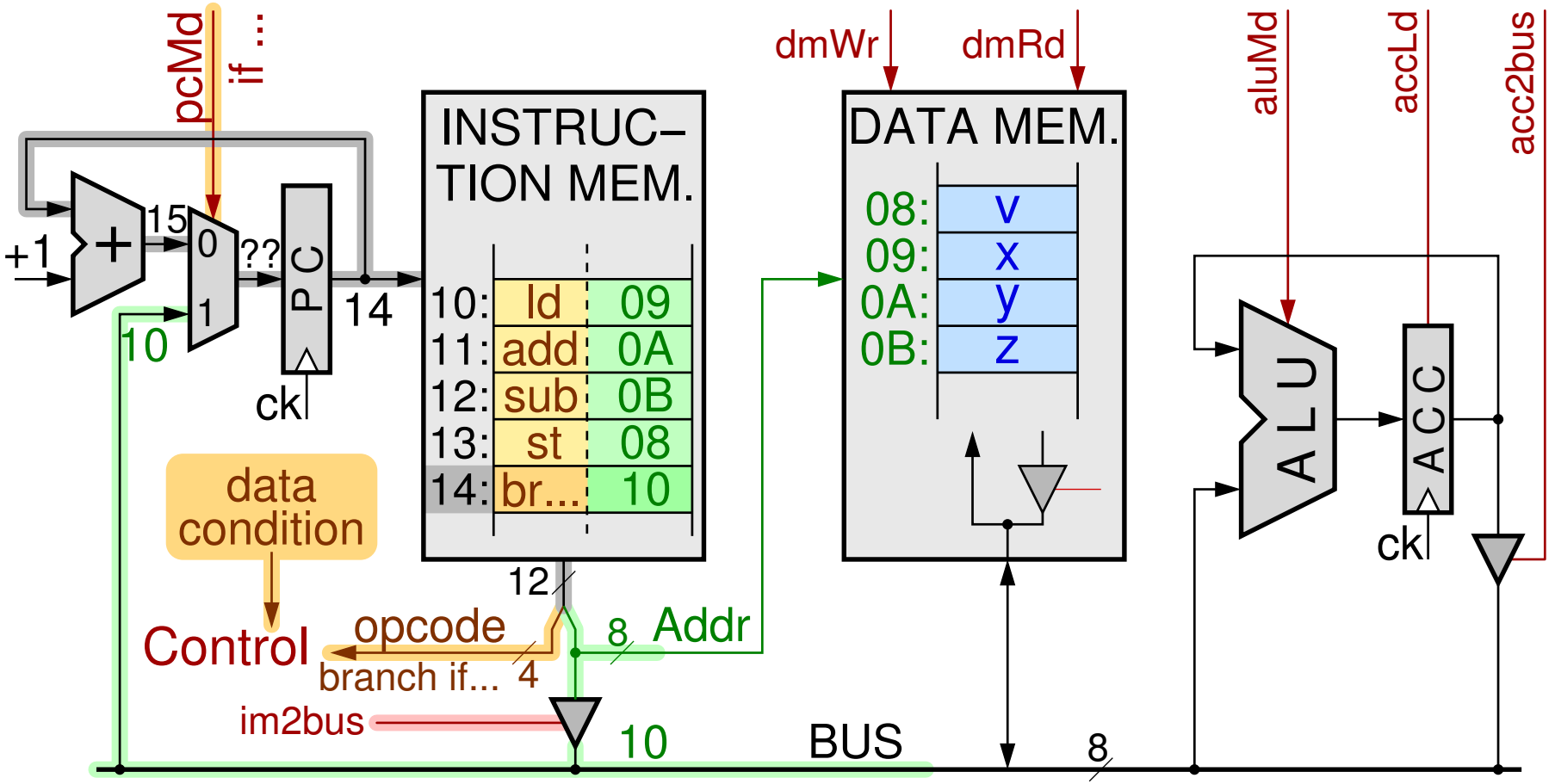


- Τι: *opcode* – Έλεγχος
- Σε ποιόν: Διεύθυνση τελεστέου

Jump: Επόμενη εντολή όχι «η από κάτω»



Διακλαδώσεις: Εάν... Επόμενη εντολή όχι «η από κάτω»

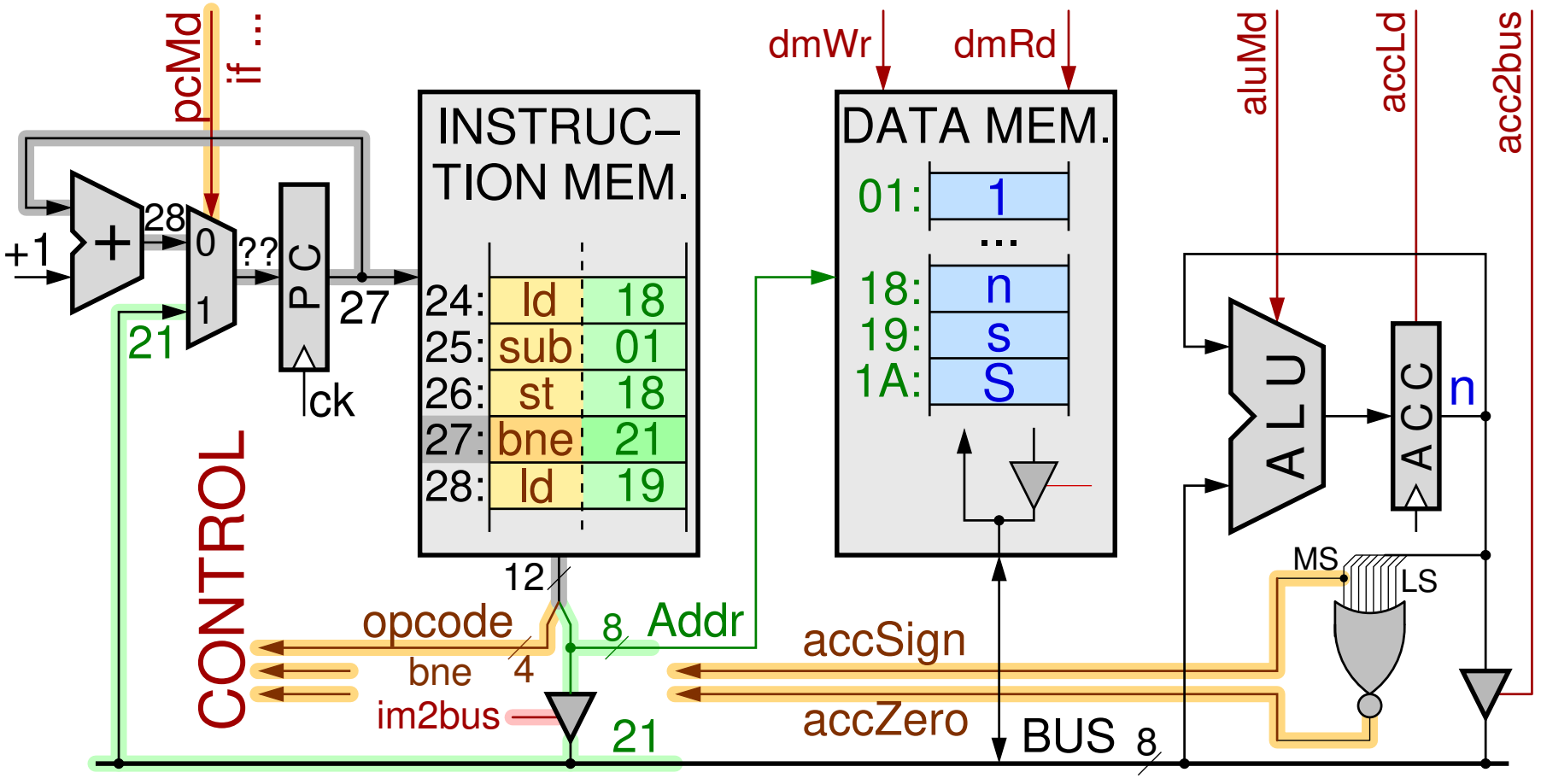


Οι 4 εντολές διακλάδωσης του απλού υπολογιστή

- beq – branch if equal
  - if ACC == 0
- bne – branch if not equal
  - if ACC != 0
- blt – branch if less than
  - if ACC < 0
- bge – branch if greater or equal
  - if ACC ≥ 0

- Ομοίως και στους πραγματικούς επεξεργαστές, αλλά με δύο συγκρινόμενους τελεστέους σε καταχωρητές
- Σύγκριση ≤ δεν χρειάζεται: αντιμεταθέτουμε τους δύο τελεστέους στη σύγκριση ≥
- Ομοίως η > μέσω της <

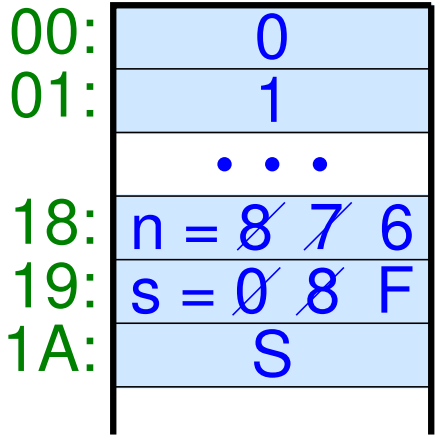
Τα σήματα Συνθήκης Δεδ. γιά τον έλεγχο Διακλάδωσης



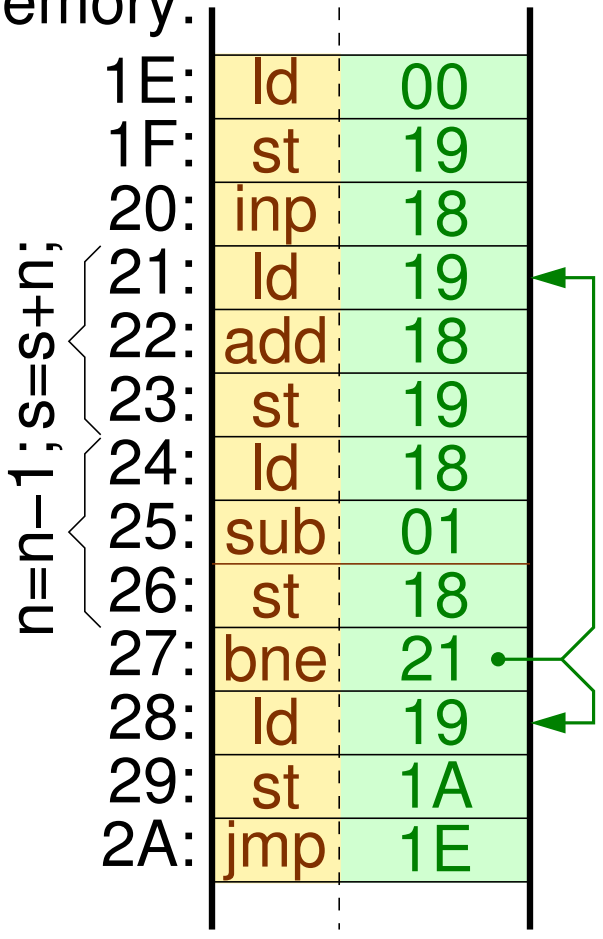
Απλός βρόχος: Άθροισμα (n)+(n-1)+(n-2)+…+(2)+(1)

```
s = 0;  n = input();
do { s = s+n;  n = n–1;
     } while (n != 0)
S = s;  /* print sum */
```

Data Memory:

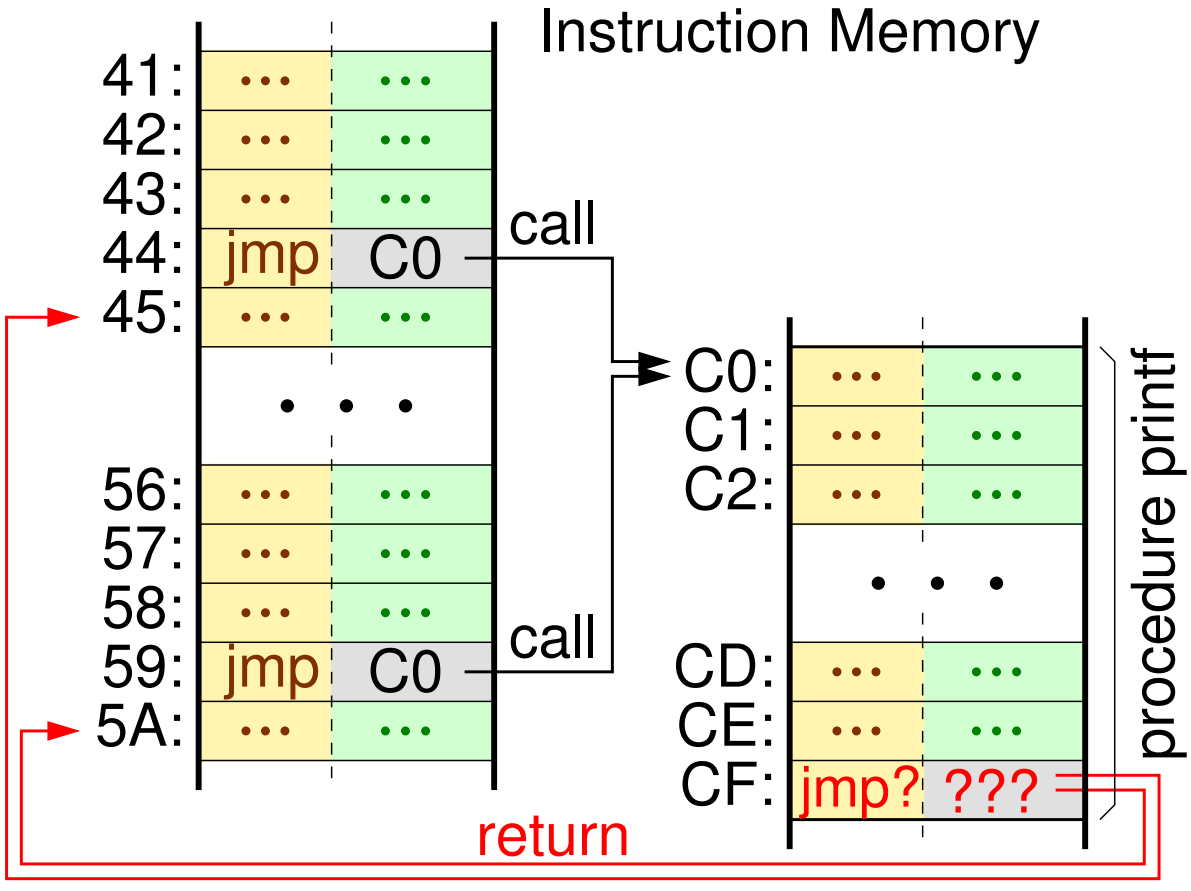


Instr. Memory:



- *bne*: branch if not equal
- Εξυπακούεται: ACC, to zero
- Εάν ACC != 0, τότε επόμενη εντολή από 21
- Αλλιώς, επόμενη εντολή η «από κάτω»

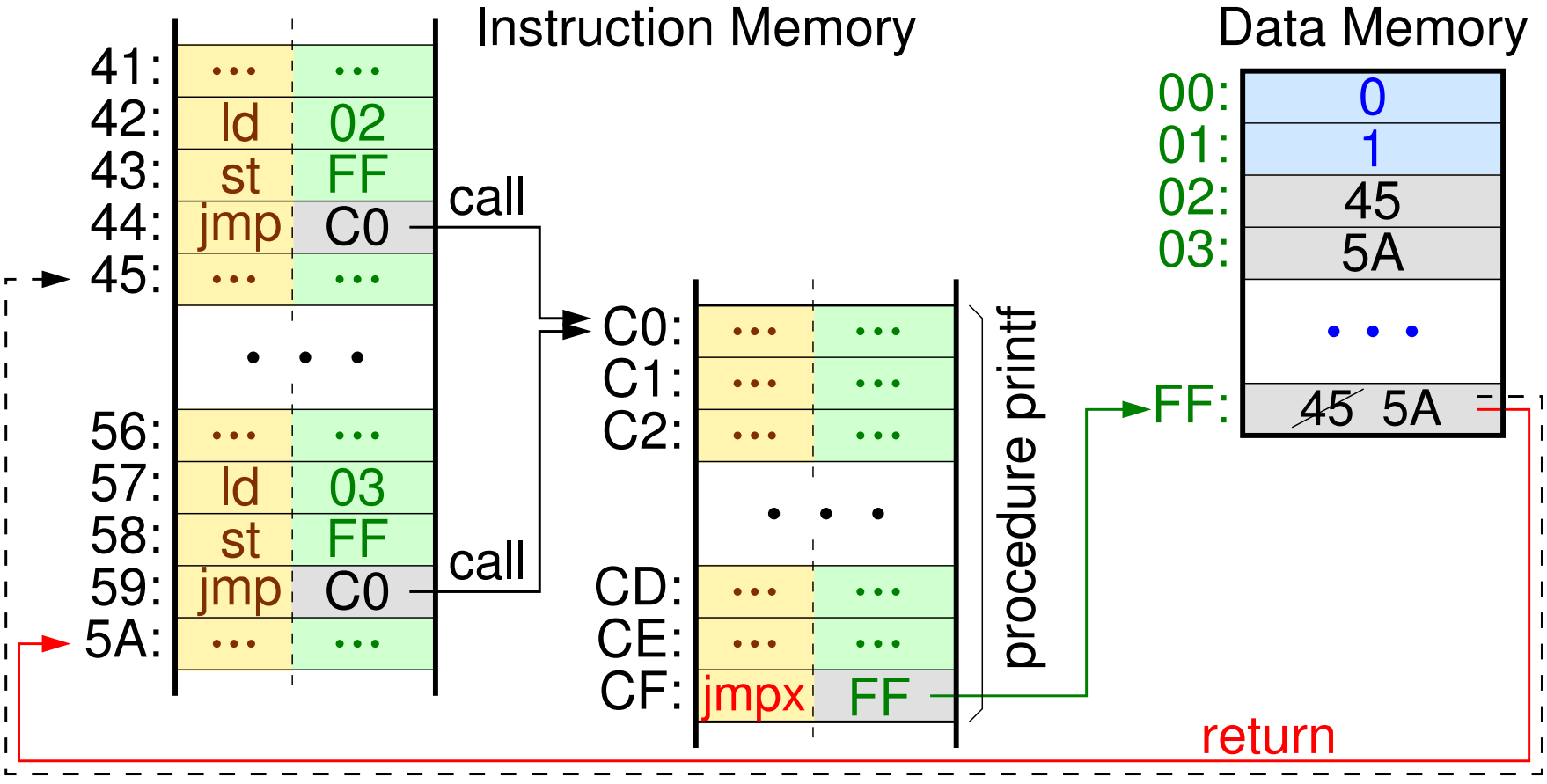
Κάλεσμα Διαδικασιών: Πώς επιστρέφουμε;



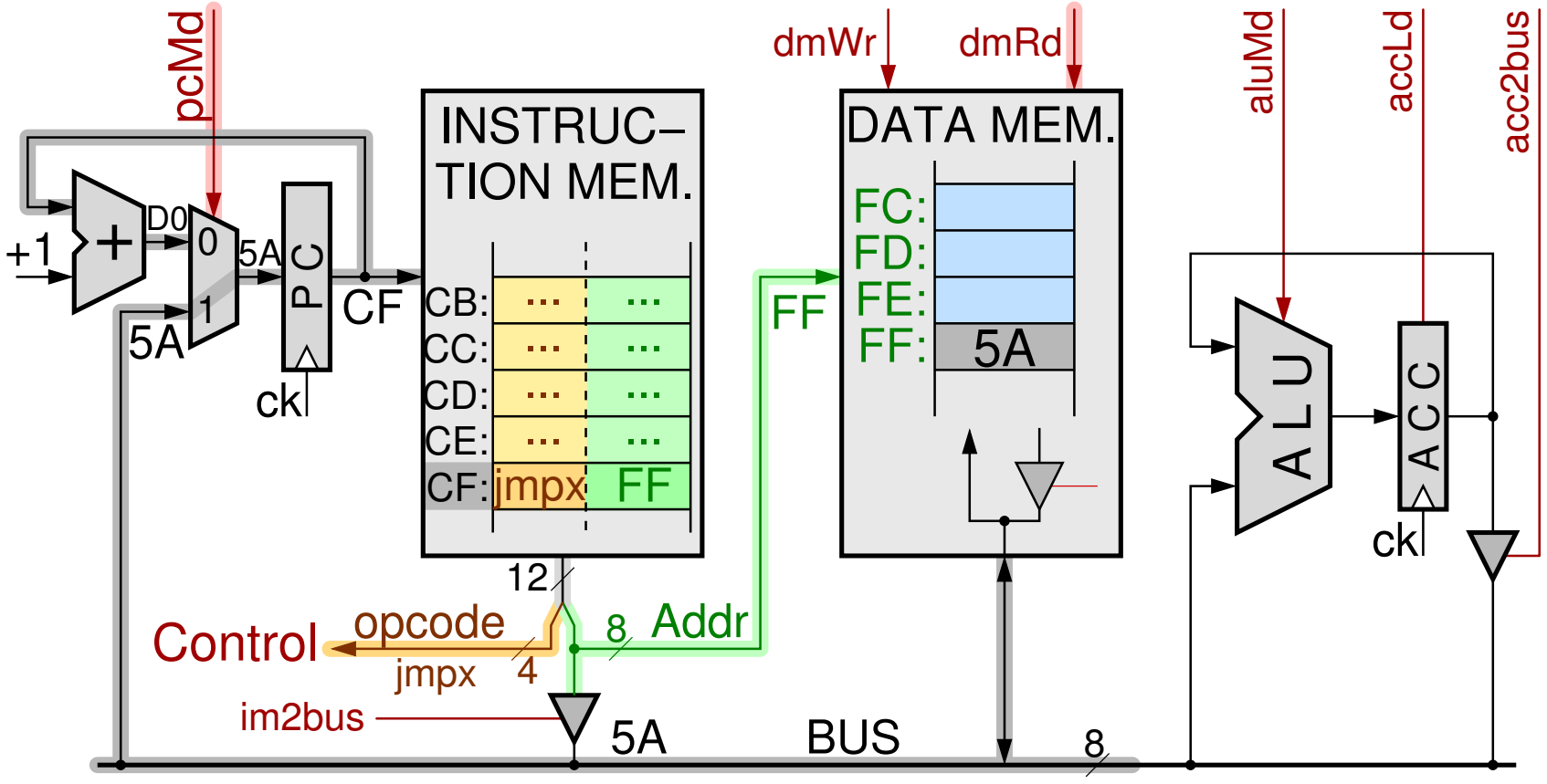
- Το πεδίο Addr μέσα στις εντολές είναι *σταθερό*
- Δεν μπορεί να αλλάζει κάθε φορά που η διαδικασία επιστρέφει

⇒ Πρέπει η διεύθυνση επιστροφής να έρχεται από Data Mem.

Jump Indexed: Εμμέσως μέσω Data Memory



Υλοποίηση της *Jumpx*

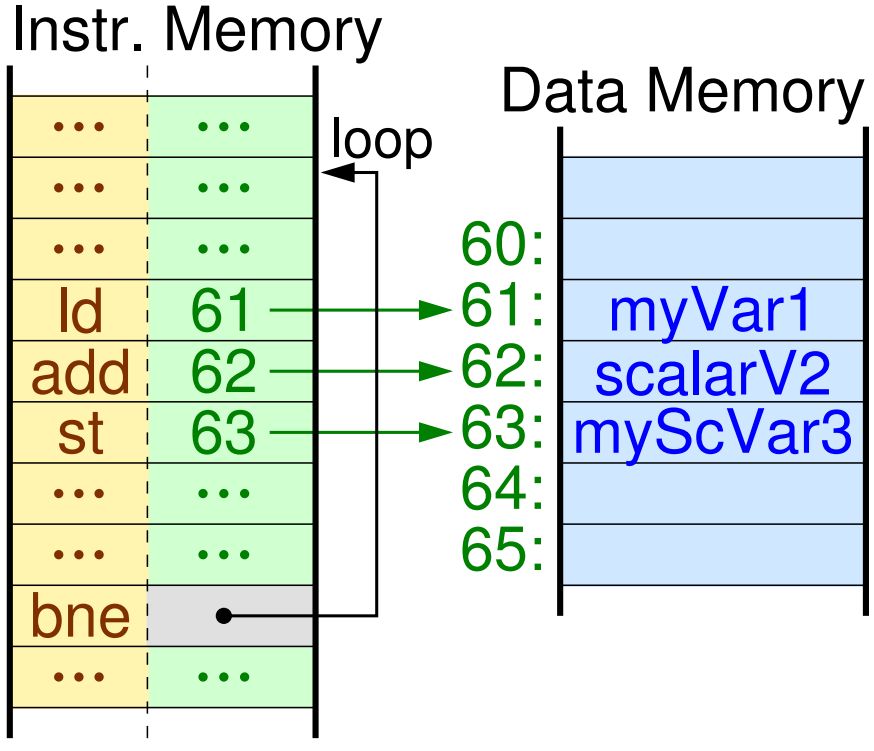


- Όπως η απλή *Jump*, αλλά ανάβει το *dmRd* αντί του *Im2bus*

Σταθερές Διευθύνσεις ⇒ Βαθμωτές Μεταβλητές μόνον

- Μη αυτομεταβαλόμενα προγράμματα
- Σταθερές οι Διευθύνσεις μέσα στις εντολές

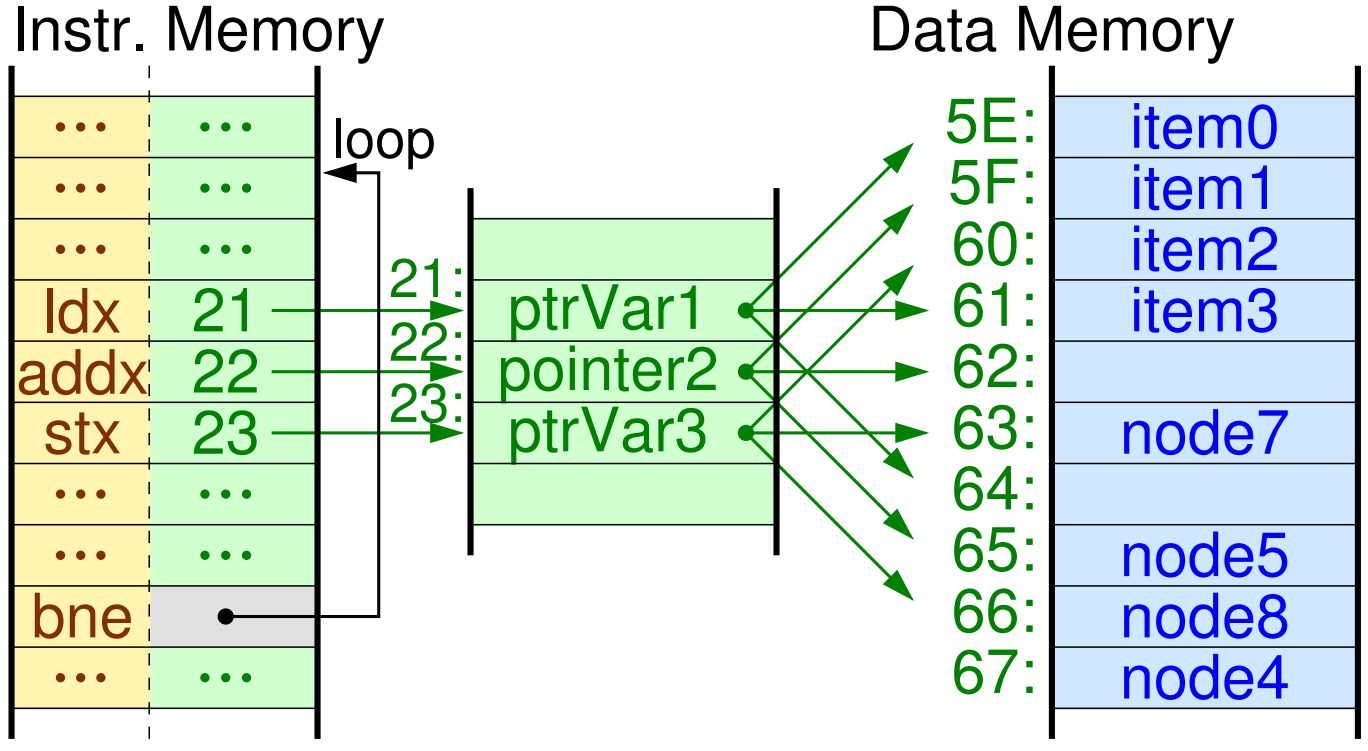
⇒ Όλες οι επαναλήψεις των βρόχων προσπελά-ζουν πάντα τις ίδιες (*βαθμωτές*) μεταβλητές με τις μέχρι τώρα εντολές



- Γιά την επεξεργασία στοιχείων Δομών Δεδομένων απαιτούνται *μεταβλητές* Διευθύνσεις

Έμμεση Πρόσβαση γιά μεταβλητές Διευθύνσεις

Νέες εντολές "indexed": προσπε-λάζουν τα δεδομένα εμμέσως, μέσω μετα-βλητών δεί-κτη (pointer)

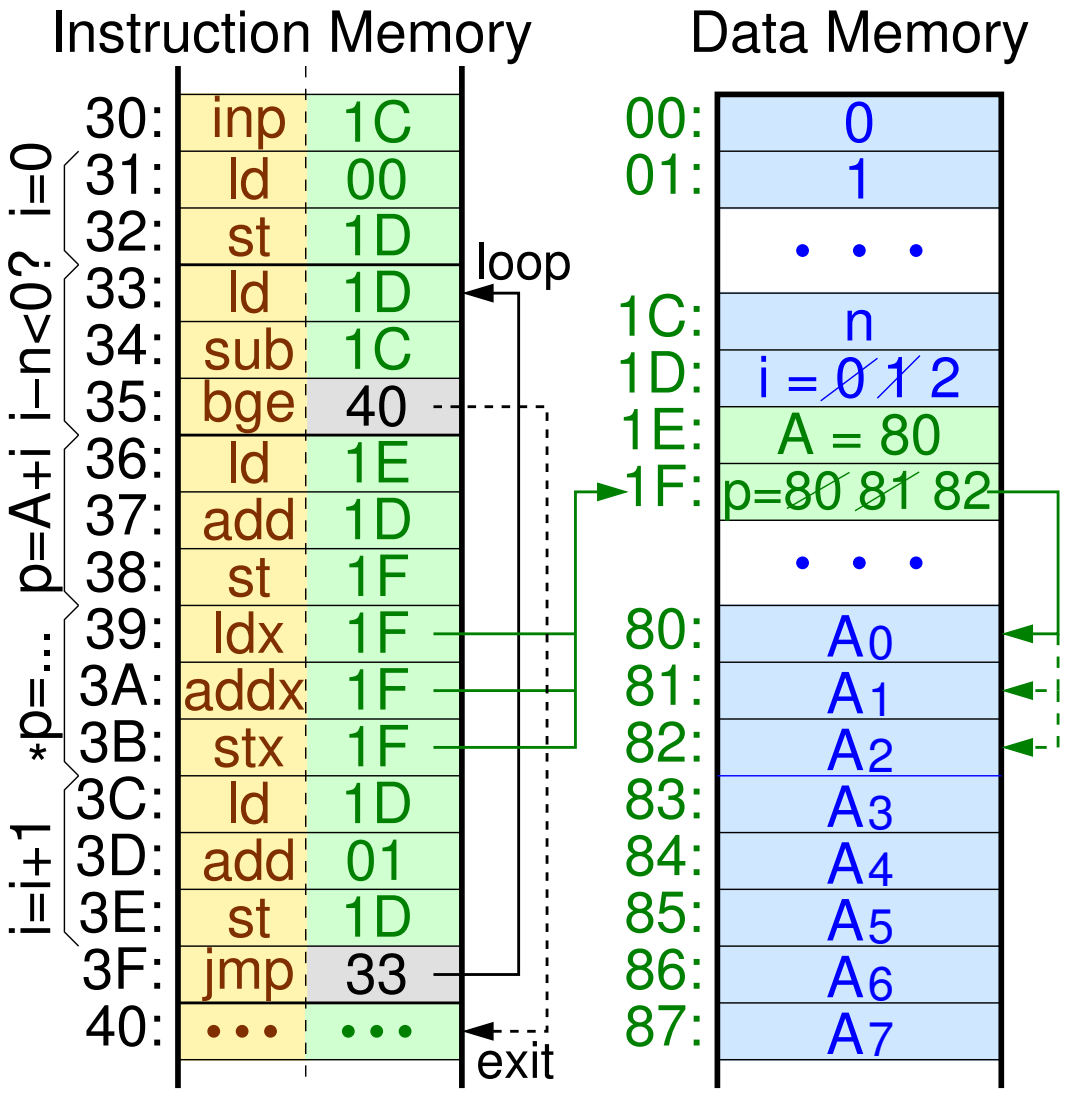


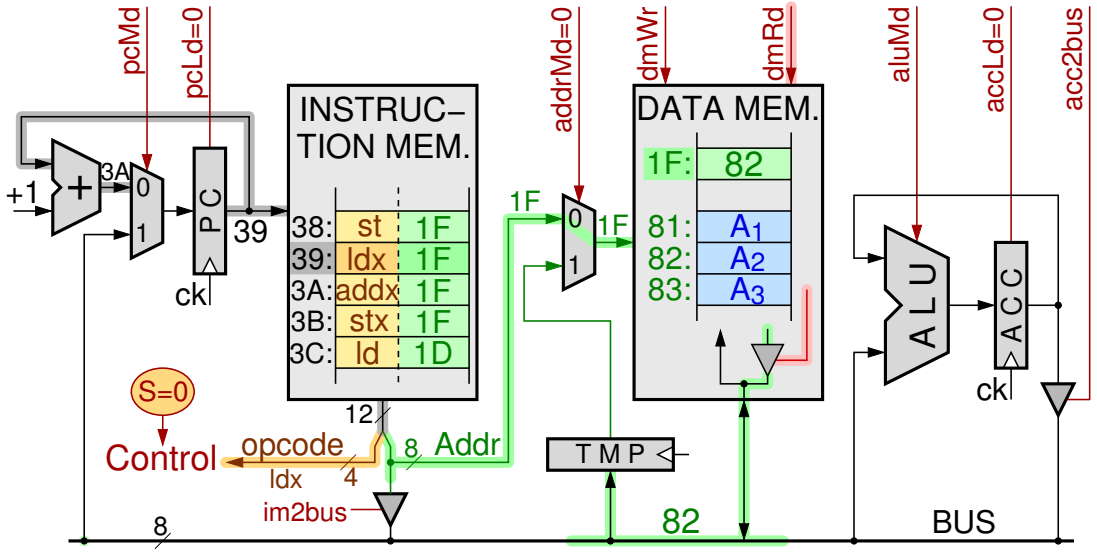
Ένας φημισμένος αφορισμός από τον David Wheeler ισχυρίζεται ότι: "All problems in computer science can be solved by another level of indirection". Μερικοί συμπληρώνουν σκωπτικά: "...except for the problem of too many layers of indirection".

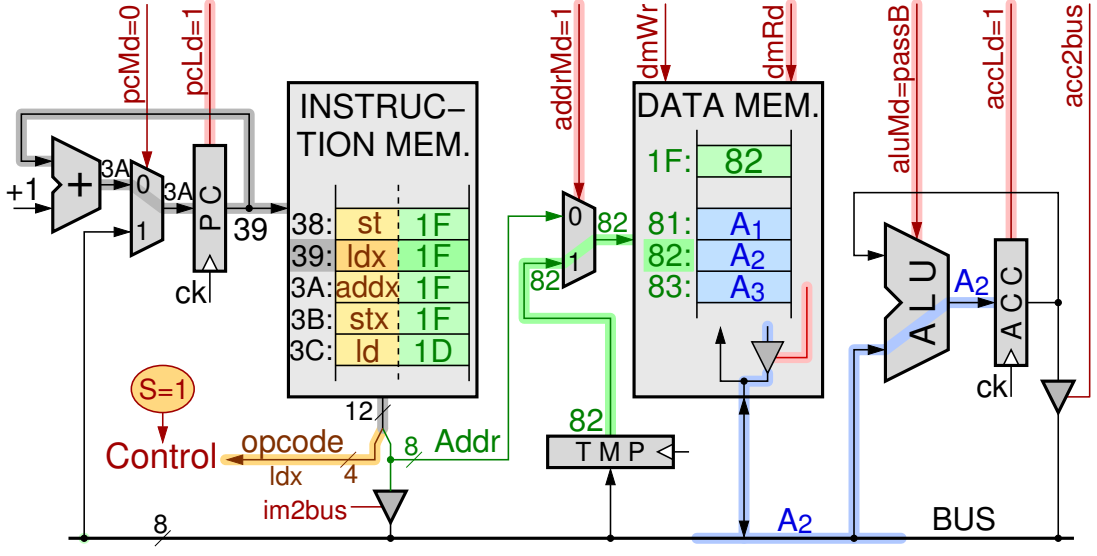
Βρόχος επεξεργασίας Πίνακα

```
n=input();  i=0;
while ( i<n ) {
    A[i] = 2*A[i];
    i = i+1;
}
```

- p = διεύθυνση του A[i]
  = διευθ. του A[0] + i
- Εντολές Indexed:
  «πήγαινε να ρωτήσεις να σου πούν πού είναι η μεταβλητή που θέλω»





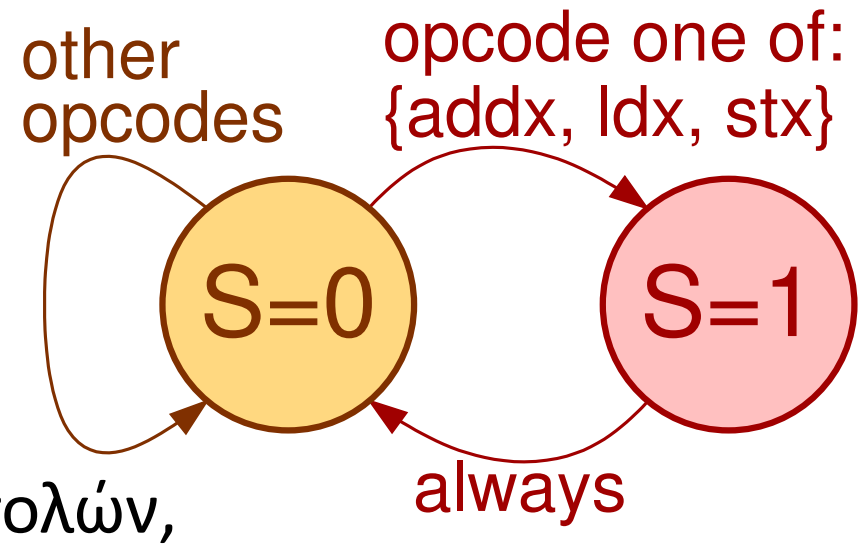


Οι δύο κύκλοι συγκριτικά

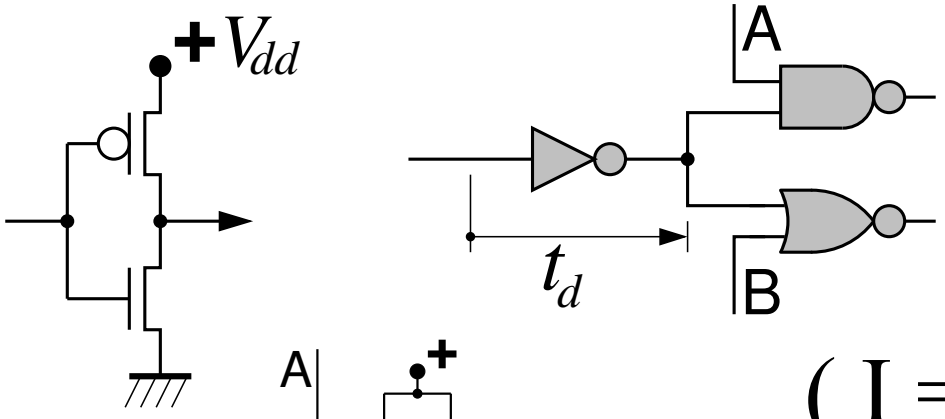
- Δύο προσπελάσεις στην Data Mem. ⇒ δύο κύκλοι
- Ακολουθ. Έλεγχος (FSM)
- Πρώτος κύκλος (S=0):
  - pcLd=0: εντολή δεν τελείωσε
  - AccLd=0: όχι ακόμα τα σωστά data, δεν χαλάμε τα παλαιά
  - addrMd=0: ανάγνωση του pointer από διεύθ. εντολής
- Πρώτος κύκλος (S=0):
  - pcLd=1: εντολή τελειώνει
  - AccLd=1: σωστά νέα data
  - TMP=pointer (BUS προηγ. κ.)
  - addrMd=1: ανάγν. data

1ος – 2ος κύκλος εντολών Indexed: η FSM Ελέγχου

- Ο Έλεγχος είναι πλέον ακολουθιακό κύκλωμα, όχι συνδυαστικό όπως πριν
- S=0 είναι η κατάσταση (ο κύκλος ρολογιού) εκτέλεσης όλων των προηγουμένων εντολών, καθώς και ο πρώτος κύκλος εκτέλεσης των νέων
- S=1 είναι ο δεύτερος (και τελευταίος) κύκλος εκτέλεσης των νέων εντολών Indexed load/store & αριθμητικών

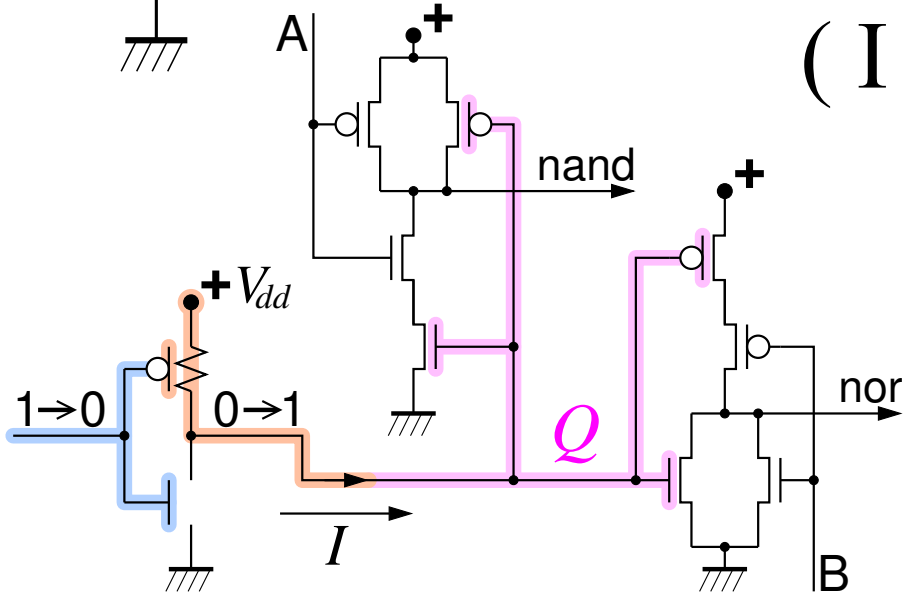


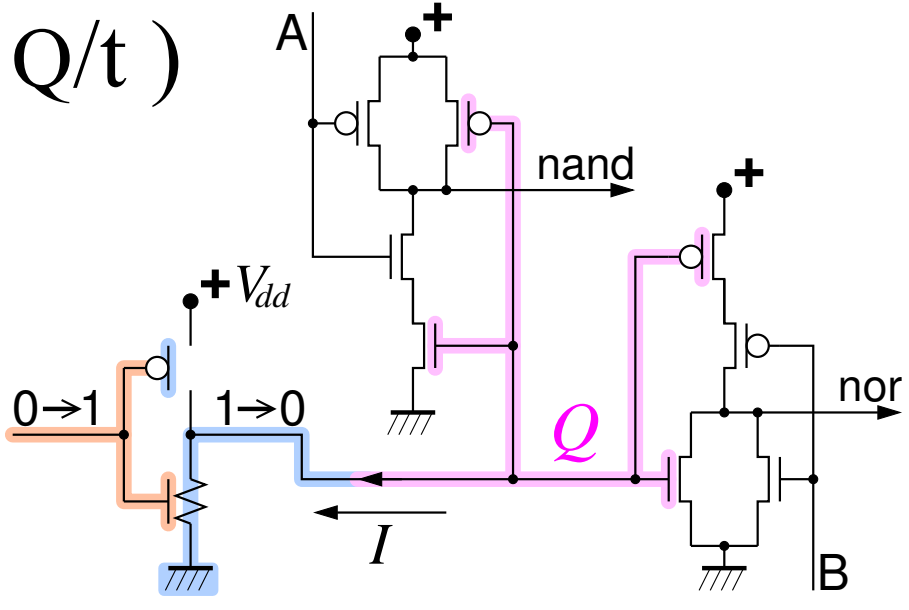
Καθυστέρηση: παροχή/απορρόφηση Φορτίου



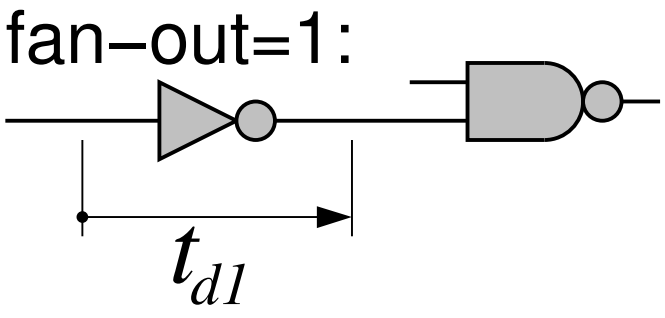
$$t_d \approx Q / I$$

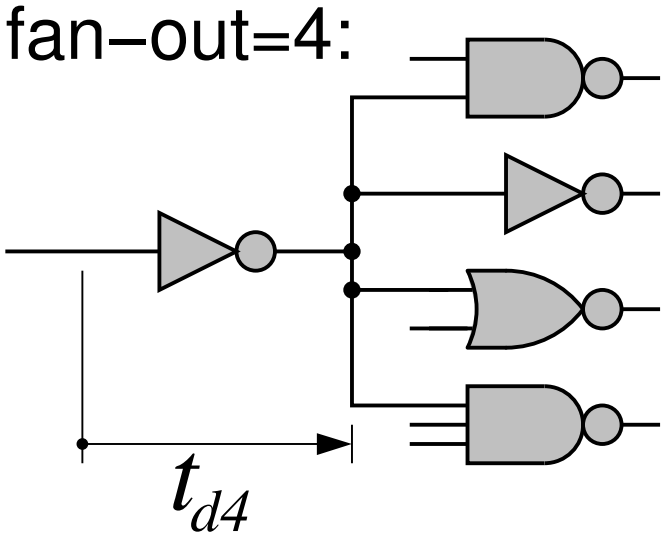
(I = Q/t)





Καθυστέρηση ~ Fan-Out (Πλήθος Ακροατών)





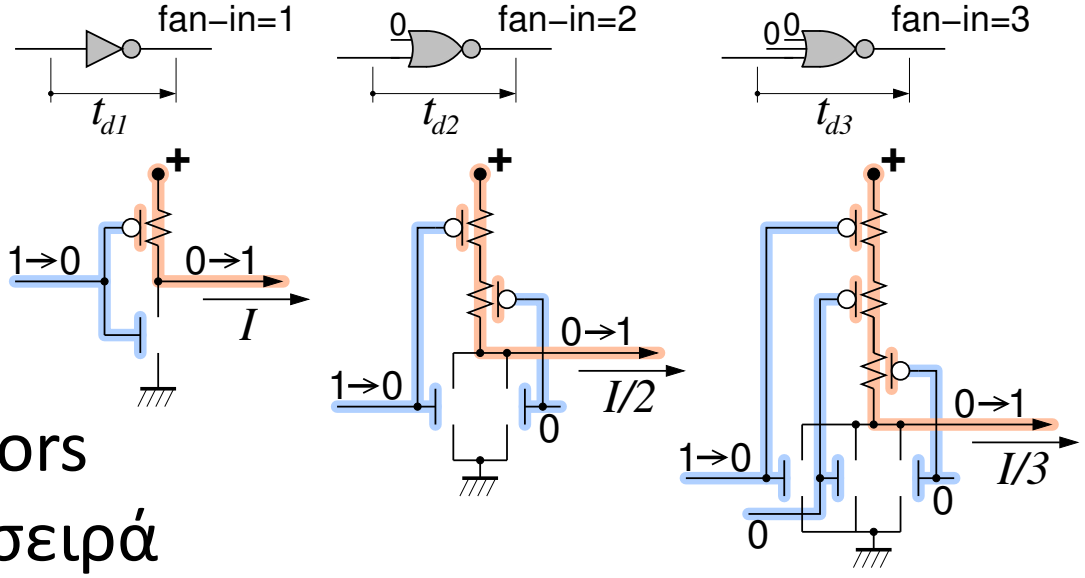
- Η κάθε είσοδος ακροατή:
  - συνήθως 2 transistors, ανεξαρτή-των πλήθους εισόδων ακροατή
- Τα δύο transistors κάθε ακροατή χρειάζονται το φορτίο τους
- Ολικό φορτίο ~ πλήθος ακροατών
- Καθυστέρηση ~ ολικό φορτίο

⇒ Καθυστέρηση ~ *Fan-out*
  - fan-out = πλήθος ακροατών

$$\Rightarrow t_{d4} \approx 4 \times t_{d1}$$

Καθυστέρηση ~ Fan-In (Πλήθος Εισόδων)

- Σε πρώτη προσέγγιση, το ρεύμα φόρτισης/ εκφόρτισης είναι αντίστροφα ανάλογο προς το πλήθος transistors οδήγησης που είναι εν σειρά



- $t_d \approx Q / I$

$\Rightarrow$ μικρότερο ρεύμα $\Rightarrow$ μεγαλύτερη καθυστέρηση

$\Rightarrow$ Καθυστέρηση (περίπου) ανάλογη (και) προς fan-in

Όσο απλούστερα & μικρότερα, τόσο πιό γρήγορα!

- Συμπέρασμα: *Απλά και μικρά κυκλώματα*, γιά ταχύτητα!
- Λιγότεροι ακροατές (fan-out) $\Rightarrow$ απλούστερο
- Λιγότερες είσοδοι (fan-in) $\Rightarrow$ απλούστερο
- Λιγότερες πύλες, ακροατές, είσοδοι $\Rightarrow$ μικρότερο
- Τη δεκαετία του ’80, αυτή η παρατήρηση υπήρξε η απαρχή του «κινήματος» *RISC*: Reduced Instruction Set Computer – Υπολογιστές ελαττωμένου (απλούστερου) Ρεπερτορίου Εντολών, που σήμερα είναι σημαντικά δημοφιλείς (και θα είναι το παράδειγμά μας στο HY-225)

Ολική Ενέργεια = $\Sigma_{κόμβοι}\ C \cdot V_{dd}^2 \cdot Πλήθος\ Ανεβοκατεβ.$

- Ολική Ενέργεια απαιτούμενη γιά έναν υπολογισμό:
- Πόσοι ηλεκτρικοί κόμβοι θα ανεβοκατέβουν;
  - μόνον όταν αλλάζει η τιμή (τάση) ενός κόμβου καταναλ. ενέργεια
  - πολλοί κόμβοι δεν αλλάζουν τιμή σε κάθε κύκλο ρολογιού
- Πόσο πολλές φορές θα ανεβοκατέβουν αυτοί;
  - πόση «εργασία» χρειάζεται ο υπολογισμός που θα γίνει
- Πόση ηλ. χωρητικότητα έχουν αυτοί οι κόμβοι;
  - συνάρτηση fan-out, πλάτους transistors, ελαχ. διαστ. τεχνολογίας
- Μειώστε την Τάση Τροφοδοσίας!!
  - Ταχύτητα ανάλογη V_{dd}, ενέργεια ανάλογη V_{dd}^2