

# Εργαστήριο 1: Λογική με Διακόπτες, Πολυπλέκτες, Μνήμη ROM

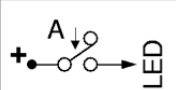
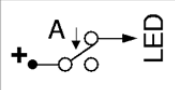
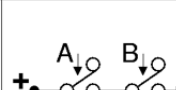
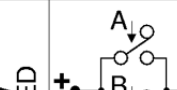
2η εβδομάδα του εξαμήνου

[Βιβλία: *προαιρετικά* μπορείτε να διαβάσετε: Dally: §4.1 (switch logic), §8.3 (την πρώτη σελίδα) (MUX), §8.8 (ROM)· ή Mano (5η έκδοση): § 1.7.μέρος (σελ. 26-27), § 4.11.μέρος (σελ. 158-159)· ή Wakerly (3η έκδοση): § 2.15.1 (σελ. 68-70), § 5.7.0 (σελ. 471-472).] Διαφάνειες διδασκαλίας: 01a, 01b.

## 1.1 Λογικές Πράξεις και Πίνακας Αληθείας

Το εργαστήριο αυτό αποτελεί συνέχεια και εμπάθυνση του εργ. 0, και ιδιαίτερα του πώς κυκλώματα διακοπών υλοποιούν τις λογικές πράξεις **OXI** (§0.9), **ΚΑΙ** (§0.10), **Η** (§0.11). Αν δεν προλάβετε να κάνετε τα αντίστοιχα πειράματα την προηγούμενη βδομάδα, ή να τα καταλάβετε σε βάθος, ξεκινήστε σήμερα με εκείνα, ή μελετήστε πρώτα εκείνες τις σημειώσεις για να τις καταλάβετε σε βάθος --αποτελούν τη βάση για όλο το μάθημα.

Ο πίνακας που ακολουθεί αποτελεί μία περίληψη των παραπάνω παραγράφων 0.6 - 0.8. Για πληρότητα, προσθέσαμε και το κύκλωμα της §0.8 με το όνομα "ταυτότητα" (identity), αν και συνήθως δεν θεωρούμε ενδιαφέρουσα αυτήν την τόσο απλή λογική πράξη. Στον πίνακα αυτόν χρησιμοποιούμε τον όρο "OFF" όταν μεν πρόκειται για διακόπτες για να σημαίνει "ελεύθερος" (όχι πατημένος), όταν δε πρόκειται για LED για να σημαίνει "σβηστή": αντίστροφα, ο όρος "ON" για μεν τους διακόπτες σημαίνει "πατημένος" για δε τις LED σημαίνει "αναμμένη".

Διακοπτής <b>A</b>	Διακοπτής <b>B</b>	 <b>IDENTITY</b> (ταυτότητα)	 <b>NOT</b> (οχι)	 <b>AND</b> (και)	 <b>OR</b> (ή)
OFF (ελεύθερος)	OFF	OFF (σβηστή)	ON (αναμμένη)	OFF	OFF
	ON	OFF (σβηστή)	OFF (σβηστή)	OFF	ON
ON (πατημένος)	OFF	ON (αναμμένη)	ON (αναμμένη)	OFF	ON
	ON	ON (αναμμένη)	OFF (σβηστή)	ON	ON

Ο πίνακας αυτός δίνει αναλυτικά τη συμπεριφορά των κυκλωμάτων και των αντίστοιχων "Λογικών Συναρτήσεων" (πράξεων) για τις διάφορες περιπτώσεις εισόδων τους, δηλαδή κατάστασης των διακοπών· τέτοιους πίνακες τους λέμε *Πίνακες Αληθείας (truth tables)*. Για το κύκλωμα της ταυτότητας, ο πίνακας αληθείας μας λέει ότι η έξοδος του (φωτοβολία της LED) βρίσκεται πάντα σε ευθεία αντιστοιχία με την είσοδό του (κατάσταση του διακόπτη A): OFF για OFF και ON για ON. Για τη λογική πράξη OXI (NOT), ο πίνακας μας δείχνει ότι η έξοδος της είναι πάντα το αντίστροφο (ανάποδο) της εισόδου της A, εξ' ου και το όνομα του κυκλώματος αυτού που συχνά λέγεται "αντιστροφέας" (inverter). Για την λογική πράξη ΚΑΙ (AND), η έξοδος της είναι συνάρτηση των δύο εισόδων της, A και B: είναι ON μόνον στον έναν από τους 4 συνδυασμούς τιμών των A και B, όταν A και B είναι ON· αυτό προκύπτει όταν οι διακόπτες είναι συνδεδεμένοι "εν σειρά". Τέλος, η λογική συνάρτηση Η (OR) είναι ON όταν A είναι ON ή B είναι ON (ή και οι δύο είναι ON)· αυτό προκύπτει όταν οι διακόπτες είναι συνδεδεμένοι "εν παραλλήλω".

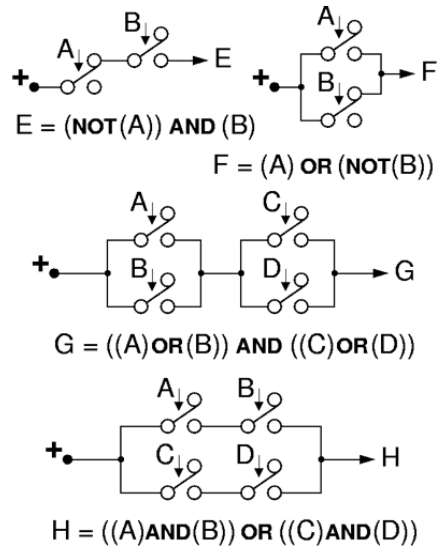
Οι λογικές πράξεις **AND** και **OR** είναι ανάλογες πλήθους εννοιών της καθημερινής ζωής, π.χ.:

- Για να τρέξει νερό από τη βρύση πρέπει να ανοίξουμε τη βρύση **και** να είναι ανοικτός και ο γενικός διακόπτης νερού του διαμερίσματος.

- Θα υπάρξει υπερκατανάλωση νερού αν έχουμε διαρροή στο καζανάκι ή αν στάζει η βρύση του μπάνιου ή της κουζίνας (ή και περισσότερα από ένα από αυτά ταυτόχρονα).
- Για να μπώ στο αυτοκίνητό μου που βρίσκεται στο γκαράζ πρέπει να ανοίξω την πόρτα του γκαράζ **και** την πόρτα του αυτοκινήτου.
- Μπορώ να μπώ στο εξοχικό μου σπίτι από την κύρια πόρτα ή από την πίσω πόρτα του κήπου.
- Από τους περισσότερους κοινούς λογαριασμούς τράπεζας μπορεί να κάνει ανάληψη ο δικαιούχος A ή ο δικαιούχος B. Υπάρχουν όμως και λογαριασμοί, π.χ. εταιρειών, όπου για να γίνει ανάληψη πρέπει να υπογράψουν π.χ. **και** ο διευθυντής **και** ο ταμίας.

### 1.2 Κυκλώματα για Σύνθετες Λογικές Προάξεις

Οι παραπάνω βασικές συνδεσμολογίες διακοπών μπορούν να συνδυαστούν μεταξύ τους σε οσοδήποτε σύνθετους συνδυασμούς: στο σχήμα δεξιά φαίνονται μερικά παραδείγματα (στα σημεία E, F, G, H, υποτίθεται ότι συνδέονται LED's μέσω αντιστάσεων). Στο πρώτο κύκλωμα, για να περάσει ρεύμα προς το E, πρέπει "όχι A πατημένος", επειδή η σύνδεση έγινε στον πάνω δεξιά ακροδέκτη (T0) του A, **ΚΑΙ** (σύνδεση εν σειρά) "B πατημένος", επειδή το E συνδέεται στον κάτω δεξιά ακροδέκτη (T1) του B. Στο δεύτερο κύκλωμα, θα περνάει ρεύμα προς το F όποτε "A πατημένος" (σύνδεση κάτω δεξιά (T1) του A), **Η** (σύνδεση εν παραλλήλω) "όχι B πατημένος" (σύνδεση πάνω δεξιά (T0) του B). Στο τρίτο κύκλωμα, για να περάσει ρεύμα προς το G πρέπει να υπάρχει δρόμος μέσω του αριστερού ζεύγους διακοπών **ΚΑΙ** μέσω του δεξιού ζεύγους: από το αριστερό ζεύγος μπορεί να περάσει ρεύμα όποτε "A πατημένος" **Η** "B πατημένος": ομοίως από το δεξί όποτε "C πατημένος" **Η** "D πατημένος". Τέλος, στο κύκλωμα που τροφοδοτεί το H, ρεύμα μπορεί να περάσει από το επάνω ζευγάρι διακοπών **Η** από το κάτω για να περάσει από επάνω πρέπει "A πατημένος" **ΚΑΙ** "B πατημένος", αντίστοιχα δε από κάτω.



### Πειράματα 1.3: Λογική με Διακόπτες

#### Προετοιμασία πριν φτάσετε στο Εργαστήριο:

Σε αυτό και σε όλα τα υπόλοιπα εργαστήρια από 'δώ και μπρος, **πριν** φτάσετε στο εργαστηριακό σας τμήμα, θα έχετε διαβάσει λεπτομερώς και προσεκτικά ολόκληρη την εκφώνηση της άσκησης, και θα έχετε ετοιμάσει **γραπτά**, καθαρά, και λεπτομερώς τα πλήρη σχεδιαγράμματα όλων των κυκλωμάτων που σας ζητούνται στην εκφώνηση, καθώς και όλες τις απαντήσεις στα ερωτήματα που σας θέτει η εκφώνηση. Την γραπτή αυτή προεργασία θα την εξετάζει ο βοηθός σας στην αρχή του εργαστηρίου, και θα του την παραδίσετε στο τέλος, από αυτήν δε θα προκύπτει ένα σεβαστό ποσοστό του βαθμού του εργαστηρίου σας. Ένα άλλο σεβαστό ποσοστό θα αντανακλά την από μέρους σας κατανόηση της θεωρίας του μαθήματος που σχετίζεται με το εργαστήριο, όπως αυτή φαίνεται από τις απαντήσεις σας στις σχετικές ερωτήσεις του βοηθού. Σε αυτό εδώ το πείραμα, **πριν** φτάσετε στο εργαστήριο:

- Σχεδιάστε τα παρακάτω κυκλώματα που σας ζητώνται. Οι διακόπτες A, B, C είναι όπως αυτοί που είδαμε στην § 0.8, και το κάθε κύκλωμα οδηγεί μία LED μέσω μιάς αντίστασης όπως στην § 0.7.
- Φτιάξτε τον πίνακα αληθείας του κάθε κυκλώματος, όπως είδαμε παραπάνω στην § 1.1. Αφού τα κυκλώματα αυτά έχουν 3 εισόδους (A, B, C), οι πίνακες αληθείας τους θα έχουν 8 γραμμές: μία τετράδα για A=OFF και μία τετράδα για A=ON· σε κάθε τετράδα, θα υπάρχει ένα ζευγάρι για B=OFF και ένα για B=ON· και σε κάθε ζευγάρι, θα υπάρχει μία γραμμή για C=OFF και μία για C=ON.
- Διατυπώστε γραπτός τη συνθήκη για την άρνηση της κάθε εξόδου, δηλαδή για το πότε η φωτοδίοδος θα είναι σβηστή ή το πότε (1) η μπιανιέρα δεν γεμίζει, ή (2) δεν μπορώ να μπώ στο σπίτι, ή (3) δεν μπορώ να φύγω με το αυτοκίνητο, ή (4) δεν θα πάω το αυτοκίνητο στο συνεργείο. Διατυπώστε τη συνθήκη με λόγια, χρησιμοποιώντας την καθημερινή μας λογική, και δείτε ότι εφαρμόζεται πάλι η αρχή του δυϊσμού των παραγράφων 0.10 και 0.11· διασταυρώστε την ορθότητα της συνθήκης με τον πίνακα αληθείας.

Στο εργαστήριο, κατασκευάστε τα κυκλώματα, ελέγξτε τα, και δείξτε τα στο βοηθό. Για να ελεγχθεί πλήρως το κάθε κύκλωμα πρέπει να τού εφαρμόσετε καθέναν από τους 8 συνδυασμούς εισόδων που έχει ο πίνακας αληθείας, και να διαπιστώσετε ότι η LED κάνει το σωστό για καθέναν· καθώς τα ελέγχετε, διασταυρώστε την ορθότητα της συνθήκης για την άρνηση της κάθε εξόδου.

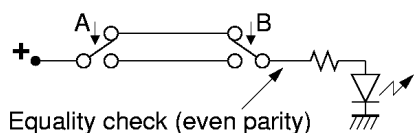
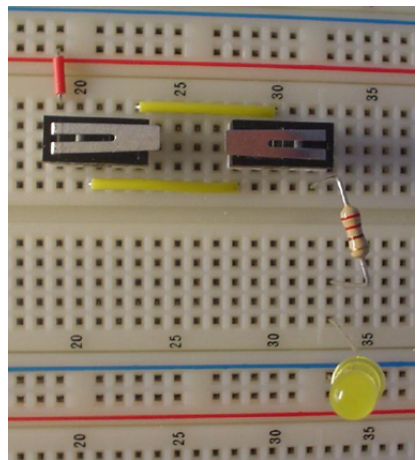
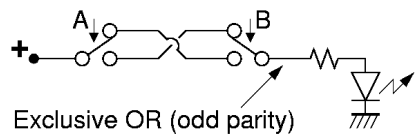
- Η LED να ανάβει όταν **θα γεμίσει η μπανιέρα**, δηλαδή όταν:
  - Έχω κλείσει την τάπα (διακόπτης A πατημένος), **ΚΑΙ**
  - ανοίγω τη βρύση του κρύου (διακόπτης B πατημένος) **Ή** ανοίγω αυτήν του ζεστού (διακόπτης C πατημένος).
- Η LED να ανάβει όταν **μπορώ να μπό στο σπίτι**, δηλαδή όταν:
  - έχω το κλειδί της κεντρικής πόρτας (A πατημένος), **Η**
  - έχω το κλειδί της μπαλκονόπορτας (B πατημένος) **ΚΑΙ** το πατζούρι της μπαλκονόπορτας είναι ανοικτό (C πατημένος).
- Η LED να ανάβει όταν **μπορώ να φύγω με το αυτοκίνητο**, δηλαδή όταν:
  - έχω το κλειδί του γκαράζ (A πατημένος), **ΚΑΙ**
  - έχω το κλειδί του αυτοκινήτου (B πατημένος), **ΚΑΙ**
  - ΔΕΝ** υπάρχει βλάβη στη μηχανή.
 Όποτε υπάρχει βλάβη στη μηχανή, θα πατιέται ο διακόπτης C.
- Η LED να ανάβει όταν **πρέπει να πάω το αυτοκίνητο στο συνεργείο**, δηλαδή όταν:
  - υπάρχει βλάβη στη μηχανή (A πατημένος), **Η**
  - πέρασαν 12 μήνες από το προηγούμενο service (B πατημένος) **ΚΑΙ ΔΕΝ** έχω μείνει "πανί-με-πανί".
 Όποτε έχω μείνει "πανί-με-πανί", θα πατιέται ο διακόπτης C.

#### Πείραμα 1.4: Διακόπτες "Aller-Retour" - Αποκλειστικό Ή, Έλεγχος Ισότητας

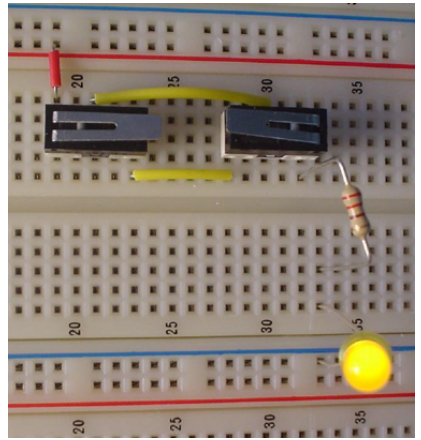
Παρατηρήστε ότι σε όλα τα παραπάνω κυκλώματα υπάρχουν ορισμένες καταστάσεις μερικών από τις εισόδους οι οποίες κάνουν την έξοδο "αναίσθητη" στις (ανεξάρτητη από τις) άλλες εισόδους. Για παράδειγμα, στο κύκλωμα ΚΑΙ, όταν ο ένας διακόπτης είναι ελεύθερος, η LED παραμένει σβηστή ό,τι και να κάνει ο άλλος διακόπτης· στο κύκλωμα Ή, όταν ένας διακόπτης είναι πατημένος, η LED ανάβει ό,τι και να κάνει ο άλλος διακόπτης. Σκεφτείτε τώρα τις κρεββατοκάμαρες των σπιτιών, όπου οι διακόπτες για τα φώτα είναι συνήθως τύπου "aller-retour", δηλαδή σε όποια κατάσταση και να έχει μείνει ο ένας διακόπτης (π.χ. της πόρτας), ο άλλος διακόπτης (π.χ. του κρεβατιού) μπορεί πάντα να αλλάξει την κατάσταση του φωτός (να το ανάψει ή να το σβήσει). Αυτό γίνεται με ένα από τα δύο κυκλώματα που φαίνονται δεξιά.

Το πρώτο κύκλωμα υλοποιεί τη λογική πράξη "αποκλειστικό Ή" (exclusive OR - XOR), διότι η LED ανάβει όταν είναι πατημένος αποκλειστικά ο ένας από τους δύο διακόπτες και όχι και οι δύο μαζί. Παρατηρήστε το κύκλωμα (όπου τα δύο χιαστί σύρματα διασταυρώνονται χωρίς να κάνουν επαφή μεταξύ τους): ρεύμα μπορεί να περάσει προς την LED είτε όταν ο A είναι πατημένος και ο B δεν είναι (σύρμα από κάτω αριστερά προς πάνω δεξιά), είτε όταν ο A δεν είναι πατημένος και ο B είναι (σύρμα από πάνω αριστερά προς κάτω δεξιά). Επομένως, η LED ανάβει όταν  $((A)\text{ΚΑΙ}(\text{ΟΧΙ}(B)))$  Ή  $((\text{ΟΧΙ}(A))\text{ΚΑΙ}(B))$ , δηλαδή είναι πατημένος (μόνο ο A) ή (μόνο ο B).

Το δεύτερο κύκλωμα υλοποιεί τη λογική πράξη **ελέγχου ισότητας** (equality check), διότι η LED ανάβει μόνον όταν οι δύο διακόπτες είναι στην ίδια (ίση) κατάσταση --και οι δύο πατημένοι ή και οι δύο ελεύθεροι. Ρεύμα μπορεί να περάσει προς την LED είτε από το επάνω σύρμα (και οι δύο διακόπτες ελεύθεροι) είτε από το κάτω σύρμα (και οι δύο διακόπτες πατημένοι), επομένως η LED ανάβει όταν  $((\text{ΟΧΙ}(A))\text{ΚΑΙ}(\text{ΟΧΙ}(B)))$  Ή  $((A)\text{ΚΑΙ}(B))$ , δηλαδή όταν A και B είναι (και οι δύο OFF) ή (και οι δύο ON).



**Πριν φτάσετε στο εργαστήριο**, γράψτε τον πίνακα αληθείας των εξόδων των δύο κυκλωμάτων. Παρατηρήστε ότι, ανεξαρτήτως της κατάστασης του ενός διακόπτη, ο άλλος μπορεί πάντα, ανοιγοκλείνοντας, να αναβοσβήσει το φως. **Στο εργαστήριο**, φτιάξτε τα δύο κυκλώματα, δείξτε τα στο βοηθό σας, και επαληθεύστε πειραματικά τις παραπάνω ιδιότητες.



### 1.4b Περιττή και Άρτια Ισοτιμία (Odd/Even Parity)

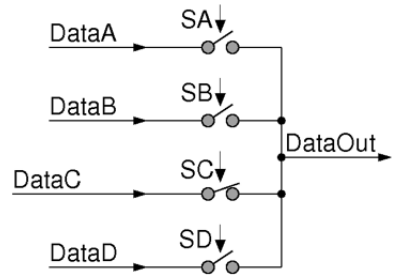
Η γενίκευση της συνάρτησης αποκλειστικού-Ή σε περισσότερες εισόδους είναι η *περιττή ισοτιμία* (odd parity), η δε γενίκευση της συνάρτησης ελέγχου ισότητας είναι η *άρτια ισοτιμία* (even parity). Όταν έχουμε πολλές εισόδους --πολλούς διακόπτες-- μετράμε το πλήθος τους που είναι πατημένοι (ON) σε δεδομένη στιγμή. Εάν το πλήθος αυτό είναι αριθμός περιττός (μονός - μη ακέραιο πολλαπλάσιο του 2), τότε λέμε ότι έχουμε περιττή ισοτιμία, και η αντίστοιχη συνάρτηση είναι ON (αναμενόμενη), αλλιώς η συνάρτηση αυτή είναι OFF (σβηστή). Αντίστροφα, όταν το πλήθος των εισόδων που είναι ON είναι αριθμός άρτιος (ζυγός - ακέραιο πολλαπλάσιο του 2), η συνάρτηση άρτιας ισοτιμίας είναι ON, αλλιώς είναι OFF. Επομένως, οι συναρτήσεις περιττής και άρτιας ισοτιμίας είναι πάντα η μία το αντίστροφο (το λογικό ΟΧΙ) της άλλης (άρα αρκεί να ξέρουμε τη μία τους για να βρούμε άμεσα και την άλλη).

Εάν ένας από τους διακόπτες αλλάξει κατάσταση, το πλήθος των εισόδων που είναι ON αλλάζει κατά ένα (+1 ή -1): αν ο διακόπτης που άλλαξε ήταν σβηστός (OFF) και άναψε (ON), το πλήθος αυξήθηκε κατά 1, ενώ αν ήταν αναμένος και έσβησε τότε το πλήθος μειώθηκε κατά 1. Οποιαδήποτε από τα δύο και να συμβαίνει, η ισοτιμία αλλάζει από άρτια σε περιττή ή από περιττή σε άρτια! Βλέπουμε λοιπόν ότι διατηρείται η βασική ιδιότητα με την οποία ξεκινήσαμε: σε οποιαδήποτε κατάσταση και να βρίσκονται οι διακόπτες, αρκεί οιοσδήποτε ένας από αυτούς να αλλάξει κατάσταση για να αλλάξει τιμή η έξοδος (από σβηστή να ανάψει ή από αναμενόμενη να σβήσει). Η ιδιότητα αυτή αποτελεί τη βάση για την κύρια εφαρμογή των συναρτήσεων ισοτιμίας:

Οι συναρτήσεις ισοτιμίας χρησιμοποιούνται σαν η απλούστερη μορφή **κώδικα ανίχνευσης σφαλμάτων** (error detection codes): αφού μεταδώσουμε μέσα από ένα τηλεπικοινωνιακό δίκτυο κάμποσες πληροφορίες (κάμποσα σύρματα, καθένα "αναμένο" ή "σβηστό"), μεταδίδουμε στο τέλος ακόμα μία επιπλέον πληροφορία που είναι π.χ. η άρτια ισοτιμία όλων των προηγούμενων. Συνήθως, οι πληροφορίες που μεταδίδουμε φτάνουν στην άλλη ακρή όλες σωστές· σε σπάνιες περιπτώσεις, λόγω θορύβου, μία από τις πληροφορίες μπορεί να φτάσει λάθος (ON αντί OFF, ή OFF αντί ON)· σε πολύ σπανιότερες περιπτώσεις μπορεί δύο ή περισσότερες πληροφορίες να φτάσουν λάθος (π.χ. αν η πιθανότητα ενός λάθους είναι μία στις χίλιες, και αν τα λάθη είναι ανεξάρτητα μεταξύ τους, τότε η πιθανότητα δύο λαθών είναι περίπου μία στο εκατομμύριο (περίπου το τετράγωνο της πιθανότητας ενός λάθους)). Εάν συμβεί ένα λάθος στη μετάδοση, τότε θα αλλάξει η ισοτιμία των πληροφοριών που στείλαμε, ακριβώς λόγω της παραπάνω ιδιότητας: οιοσδήποτε και αν είναι οι πληροφορίες, η αλλαγή μίας οιασδήποτε από αυτές αλλάζει την ισοτιμία. Αν όμως αλλάξει η ισοτιμία, θα το καταλάβουμε, διότι η επιπλέον πληροφορία ισοτιμίας που στείλαμε δεν θα συμπίπτει πλέον με την ισοτιμία που βλέπει ο παραλήπτης! Έτσι επιτυγχάνεται ο στόχος της ανίχνευσης σφαλμάτων στις περιπτώσεις που συμβαίνει μόνο ένα σφάλμα, που είναι και οι πιο συχνές. Άπαξ και διαπιστωθεί η ύπαρξη σφάλματος, η διόρθωση του μπορεί να γίνει π.χ. με μία αίτηση αναμετάδοσης ("ξαναπές το --δεν άκουσα καλά"). Ο κύριος περιορισμός των συναρτήσεων ισοτιμίας στην ανίχνευση σφαλμάτων είναι ότι αυτές ανιχνεύουν μόνο 1, 3, 5, κλπ. σφάλματα, ενώ τους διαφεύγουν 2, 4, 6, κλπ. σφάλματα. Εάν τα σφάλματα είναι ανεξάρτητα μεταξύ τους, και εάν η πιθανότητα ενός σφάλματος είναι πολύ μικρή, τότε 2 ή περισσότερα "μαζεμένα" σφάλματα είναι πολύ σπάνια. Εάν όμως τα σφάλματα είναι συχνά ή δεν είναι ανεξάρτητα μεταξύ τους, τότε χρειαζόμαστε άλλους, πιο πολύπλοκους κώδικες για την ανίχνευσή τους· τέτοια περίπτωση "ομοβροντίας (εκρηκτικών) σφαλμάτων" (burst errors) έχουμε π.χ. όταν ένα κινητό τηλέφωνο, κινούμενο, περνάει για λίγο πίσω από μία "σκιά" της κεραίας του σταθμού βάσης· αρκετά όμως είπαμε για τώρα --για περισσότερες πληροφορίες θα πρέπει να πάρετε κάποιο μάθημα τηλεπικοινωνιών και κωδικοποίησης....

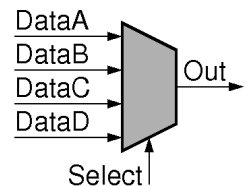
## 1.5 Πολύπλεξη: Επιλογή Πληροφορίας μεταξύ Πολλαπλών Εισόδων

Ένας από τους βασικότερους δομικούς λίθους όλων των ψηφιακών συστημάτων είναι ο *πολυπλέκτης* (*multiplexor*, ή εν συντομία *mux*). Είναι ένα κύκλωμα που επιλέγει μία από τις πολλές εισόδους του, και μεταφέρει (αντιγράφει) την τιμή (κατάσταση) της στην έξοδό του. Δεξιά φαίνεται ένα παράδειγμα πολυπλέκτη 4-σε-1 (4-to-1 mux). Έχει τέσσερις "Εισόδους Δεδομένων", DataA, DataB, DataC, και DataD, και μία έξοδο (δεδομένων) DataOut. Κάθε φορά, μία και μόνο μία από τις εισόδους μεταφέρεται (αντιγράφεται) και εμφανίζεται στην έξοδο, DataOut. Είναι η ίδια λειτουργία όπως το ραδιόφωνο ή η τηλεόραση, που στη μεν κεραία τους φτάνουν τα σήματα από όλους τους σταθμούς που εκπέμπουν στην περιοχή, αλλά που στη συνέχεια το ραδιόφωνο ή η τηλεόραση, χρησιμοποιώντας τη μέθοδο του συντονισμού, επιλέγει (συντονίζεται σε) έναν από αυτούς τους σταθμούς για να περάσει το σήμα του στην "έξοδο" --το μεγάφωνο ή την οθόνη.

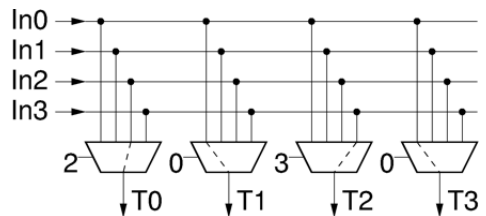


Το ποιά είσοδος επιλέγεται μπορεί να αλλάζει "δυναμικά", δηλαδή την ώρα της λειτουργίας του κυκλώματος, και για να γίνεται η επιλογή αυτή ο πολυπλέκτης έχει άλλες, επιπλέον εισόδους, τις *εισόδους ελέγχου* (control inputs) ή *εισόδους επιλογής* (selection inputs), που εδώ είναι οι SA, SB, SC, και SD. Στο απλό κύκλωμα πολυπλέκτη που δείχνουμε εδώ, η επιλογή εισόδου γίνεται με τον εξής τρόπο: ανά πάσα στιγμή, ένας και μόνον ένας από τους διακόπτες SA, SB, SC, SD πρέπει να είναι πατημένος. Αυτός που είναι πατημένος ενώνει την είσοδο δεδομένων του με την έξοδο, "μεταφέροντας" έτσι ό,τι "πληροφορία" υπάρχει σε αυτή την είσοδο προς την έξοδο. Στο σχήμα εδώ, αυτή τη στιγμή, ο διακόπτης SC είναι πατημένος, και γι' αυτό η έξοδος DataOut ισούται αυτή τη στιγμή με την είσοδο DataC. Η κατάσταση των υπολοίπων εισόδων δεν μπορεί να επηρεάσει την έξοδο, διότι εκείνες είναι ασύνδετες με την έξοδο.

Το σύμβολο του πολυπλέκτη είναι ένα τραπέζιο, όπως φαίνεται στο δεύτερο σχήμα δεξιά για το παραπάνω παράδειγμα πολυπλέκτη 4-σε-1. Για συντομία, όλες οι εισόδους επιλογής (ελέγχου) σημειώνονται με μία μόνο γραμμούλα, "Select". Σε επόμενο μάθημα θα πούμε περισσότερα πάνω σε αυτό το θέμα. Μία σημαντικότερη εφαρμογή των πολυπλεκτών είναι στις μνήμες, όπως θα δούμε αμέσως μετά, σε αυτό το εργαστήριο.



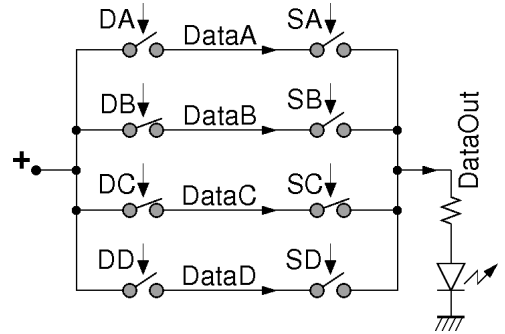
Μία δεύτερη, εξ ίσου σημαντική εφαρμογή των πολυπλεκτών είναι στα δίκτυα επικοινωνίας, εκεί όπου επιλέγεται ποιός θα μιλήσει με ποιόν κάθε στιγμή. Αυτό μπορεί να είναι σ' έναν υπολογιστή ή άλλο ψηφιακό σύστημα, ή σ' έναν δρομολογητή (router) ή μεταγωγέα (switch) ενός δικτύου υπολογιστών ή τηλεπικοινωνιών. Στο σχήμα φαίνεται ένα απλό παράδειγμα μεταγωγέα 4x4 τύπου *σταυραγωγού* (*crossbar*) εμπνευσμένου από την κλασική τηλεφωνία. Είναι κατασκευασμένος από 4 πολυπλέκτες, καθένας μεγέθους 4-σε-1. Μπορούμε να φανταστούμε ότι οι 4 εισόδους, *In0-In3*, είναι τα μικρόφωνα των τηλεφώνων τεσσάρων σπιτιών, οι δε 4 έξοδοι, *T0-T3*, τροφοδοτούν τα ακουστικά τεσσάρων άλλων τηλεφώνων. Ελέγχοντας κατάλληλα τα σήματα επιλογής των πολυπλεκτών πραγματοποιούμε τις τηλεφωνικές συνδέσεις που μας ζητώνται. Στο παράδειγμα, η είσοδος *In2* έχει ζητήσει να μιλήσει στην έξοδο *T0*, και η *In3* στην *T2*. Η είσοδος *In0* μεταδίδεται σε περισσότερους από έναν ακροατές --στους *T1* και *T3*-- κάτι γνωστό σαν *multicasting* στα δίκτυα. Η είσοδος *In1* είναι αδρανής προς στιγμήν --ή τουλάχιστο, κανείς δεν την ακούει.



Ακόμα μία εφαρμογή των πολυπλεκτών είναι στη *σειριακή* (serial) μετάδοση δεδομένων, στα δίκτυα επικοινωνίας. Στο πρώτο σχήμα παραπάνω, έστω ότι έχουμε τέσσερις πληροφορίες στις τέσσερις εισόδους DataA έως DataD, και θέλουμε να τις στείλουμε σε κάποιον μακρυνά, μέσω ενός και μόνο σύρματος, του DataOut (τα 4 σύρματα θα κόστιζαν πολύ, λόγω απόστασης). Για να το πετύχουμε, αρκεί να αλλάξουμε διαδοχικά τα σήματα επιλογής του πολυπλέκτη από το SA στο SB, στο SC, και στο SD, οπότε οι 4 πληροφορίες από τις 4 εισόδους θα τοποθετηθούν διαδοχικά πάνω στο σύρμα DataOut και θα μεταδοθούν "σειριακά" (με τη σειρά) στον παραλήπτη. Βεβαίως, για να μπορέσει ο παραλήπτης να τα καταλάβει σωστά, πρέπει να υπάρχει μία πρόσθετη συμφωνία (σύμβαση) *χρονισμού*, δηλαδή να ξέρει σε ποιά χρονική στιγμή μεταδίδεται η κάθε μία από τις τέσσερις πληροφορίες.

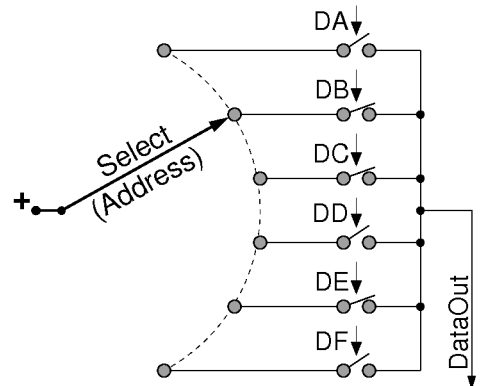
## 1.6 Η Πολύπλεξη σαν το Λογικό Ή πολλαπλών ΚΑΙ

Το κύκλωμα πολυπλέκτη με διακόπτες που είδαμε στο πρώτο σχήμα στην προηγούμενη παράγραφο, §1.5, ήταν σχεδιασμένο με πολύ αφηρημένο τρόπο από ηλεκτρική άποψη --ίσα-ίσα για να δώσει τη γενική ιδέα της πολύπλεξης, αλλά χωρίς να δείχνει τις ηλεκτρικές λεπτομέρειες του τι είναι τα "δεδομένα εισόδου" και τι η "έξοδος". Το σχήμα εδώ δεξιά δείχνει τον ίδιο πολυπλέκτη 4-σε-1, αλλά προσδιορίζοντας αυτή τη φορά το από πού έρχονται οι εισόδους δεδομένων και πού πηγαίνει η έξοδος· έτσι, εδώ έχουμε ένα κανονικό ηλεκτρικό κύκλωμα, που ξεκινά από το θετικό πόλο της μπαταρίας, αριστερά, και καταλήγει πίσω στη μπαταρία (αρνητικό πόλο) δεξιά-κάτω. *DA* είναι ο διακόπτης που μας τροφοδοτεί τα δεδομένα εισόδου *DataA* που έδειχνε το προηγούμενο σχήμα: όταν ο *DA* είναι ελεύθερος, όπως στο εδώ σχήμα, τότε η "πληροφορία" *DataA* λέει "ας μείνει η λάμπα σβηστή". Από την άλλη, ο διακόπτης *DB* που είναι πατημένος μας τροφοδοτεί το "δεδομένο εισόδου" *DataB* που λέει "θέλω η λάμπα να ανάψει". Βέβαια, στην προκειμένη περίπτωση, ούτε ο διακόπτης *DA* καταφέρνει να κρατήσει τη λάμπα σβηστή, ούτε ο διακόπτης *DB* είναι αυτός που ανάβει τη λάμπα --απλούστατα διότι τα "σήματα επιλογής" *SA* και *SB* δεν επιλέγουν τις εισόδους *DataA* ή *DataB* για να περάσουν στην έξοδο (λάμπα) *DataOut* --ούτε το *SD* επιλέγει την *DataD*-- παρά μόνο το *SC* επιλέγει την *DataC* για να περάσει.



Όπως βλέπουμε, τελικά ο πολυπλέκτης είναι ένα μεγάλο λογικό Ή (δηλαδή εν παραλλήλω σύνδεση) από όρους που ο καθένας τους είναι το λογικό ΚΑΙ δύο πραγμάτων (δηλαδή δύο διακόπτες εν σειρά): ενός δεδομένου εισόδου, και του αντίστοιχου σήματος επιλογής. Μία μόνο από τις εισόδους επιλέγεται κάθε φορά, δηλαδή ένα μόνο από τα σήματα επιλογής είναι αναμένο (διακόπτης πατημένος). Συνεπώς, ένας μόνο από τους παράλληλα-συνδεδεμένους κλάδους μπορεί να άγει ρεύμα --εκείνος που το σήμα επιλογής του είναι αναμένο. Αυτός ο επιλεγμένος κλάδος, τελικά, θα άγει ή δεν θα άγει ρεύμα ανάλογα με το τι κάνει το σήμα δεδομένων του: εάν αυτός ο επιλεγμένος διακόπτης δεδομένων πεί να ανάψει η λάμπα, αυτή θα ανάψει, επειδή είναι αναμένο και το σήμα επιλογής και το ρεύμα μπορεί να περάσει προς τη λάμπα (έξοδος)· εάν ο επιλεγμένος διακόπτης δεδομένων είναι σβηστός, τότε και η λάμπα θα μείνει σβηστή, αφού ο μεν επιλεγμένος κλάδος δεν άγει ρεύμα λόγω των δεδομένων του, οι δε υπόλοιποι κλάδοι δεν άγουν λόγω των σβηστών διακοπών επιλογής τους. Τελικά λοιπόν, η λάμπα θα κάνει ό,τι κάνει και ο διακόπτης δεδομένων (data) του μοναδικού επιλεγμένου κλάδου --του μοναδικού κλάδου που έχει αναμένο διακόπτη "Select".

Τα προηγούμενα σχήματα σχεδιάστηκαν θεωρώντας ότι τα δεδομένα (data) "έρχονται από μακριά" (αριστερά), και η επιλογή γίνεται "τοπικά", δηλαδή δεξιά, κοντά στην έξοδο, όπως π.χ. στα τηλεπικοινωνιακά συστήματα. Η άλλη περίπτωση είναι αυτό που συμβαίνει στη **Μνήμη**: τα δεδομένα είναι εδώ, κοντά μας, αποθηκευμένα μέσα στη μνήμη, και κάποιος έρχεται απ' έξω και ζητάει να διαβάσει ένα από αυτά τα δεδομένα (ή μία μικρή ομάδα από αυτά, όπως θα πούμε παρακάτω). Στην περίπτωση αυτή, ταιριάζει να αντιστρέψουμε τη θέση των διακοπών δεδομένων και επιλογής, όπως φαίνεται στο δεύτερο σχήμα. Φυσικά, η λογική λειτουργία ΚΑΙ είναι η ίδια, ανεξαρτήτως θέσης των δύο εν σειρά διακοπών, αφού για να περάσει ρεύμα πρέπει να είναι και οι δύο αναμένοι, με όποια σειρά και αν τους βάλουμε. Η άλλη αλλαγή που κάναμε σε αυτό το δεύτερο σχήμα είναι να αντικαταστήσουμε, συμβολικά, τους πολλαπλούς διακόπτες επιλογής με κάτι που θυμίζει έναν μόνο μεγάλο διακόπτη με πολλαπλές επαφές, αριστερά: Αφού ένας και μόνον ένας από τους διακόπτες επιλογής πρέπει να είναι αναμένος, σημαίνει ότι η τροφοδοσία ρεύματος από τον θετικό πόλο της πηγής κατευθύνεται σε έναν και μόνον έναν από τους κλάδους. Αυτό δείχνει και το σχήμα συμβολικά, με το μεγάλο βέλος που κινείται κυκλικά, τροφοδοτώντας τον εκάστοτε επιλεγμένο κλάδο του πολυπλέκτη. Στο σχήμα έχουμε 6 δεδομένα, *DA*, *DB*, *DC*, *DD*, *DE*, και *DF*, με τους αντίστοιχους 6 κλάδους και 6 επιλογές για

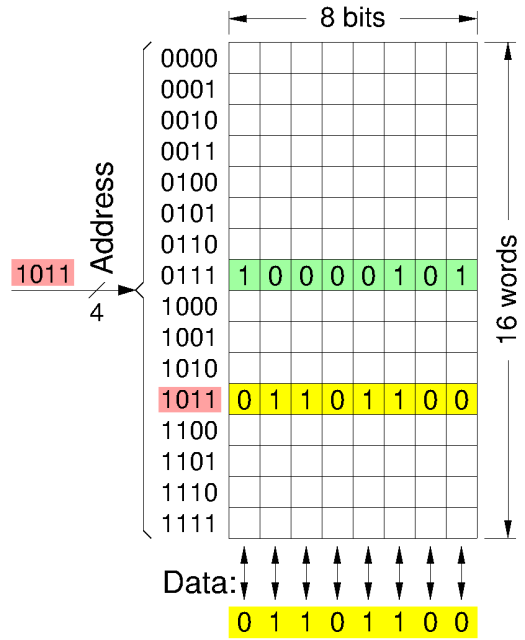


τον μεγάλο κυκλικό διακόπτη αριστερά. Στις μνήμες λέμε ότι η *Διεύθυνση (Address)* είναι αυτή που επιλέγει (select) το ποιο από τα δεδομένα θέλουμε εκάστοτε να οδηγηθεί στην έξοδο --δηλαδή να το διαβάσουμε. Στο παράδειγμα του παραπάνω σχήματος, όποτε "διαβάζουμε" (επιλέγουμε) τις πληροφορίες (data) DA, DD, ή DF θα τις βρίσκουμε σβηστές (σβηστοί δαικόπτες), δηλαδή η λάμπα που θα συνδέεται στην έξοδο DataOut θα μένει σβηστή· αντίθετα, όποτε διαβάζουμε τις θέσεις DB, DC, ή DE θα παίρνουμε παίρνουμε την απάντηση "αναμένη" (αναμένοι διακόπτες), δηλαδή η λάμπα στο DataOut θα ανάβει.

### 1.7 Η γενική Ιδέα της Μνήμης

Το σχήμα δεξιά δείχνει τη γενική ιδέα των μνημών στους υπολογιστές και τα ψηφιακά συστήματα. Φανταζόμαστε τις μνήμες σαν μία σειρά από λέξεις (words), αποθηκευμένες η μία κάτω από την άλλη, όπου οι λέξεις αυτές αποτελούν τα δεδομένα (data) --τις πληροφορίες-- που είναι αποθηκευμένα στη μνήμη. Στο σχήμα εδώ, η μνήμη περιέχει 16 λέξεις, που είναι οι 16 γραμμές από τετράγωνα κουτάκια. Η κάθε λέξη θέλουμε να περιέχει μία ποσότητα πληροφορίας αρκετά μεγάλη για να είναι χρήσιμη για διάφορους σκοπούς, αλλά και όχι τόσο μεγάλη που να δυσκολευόμαστε στη χρήση της. Η βασική πληροφορία για μας μέχρι τώρα ήταν αυτό που έλεγε σε μία λαμπίτσα να ανάψει ή να σβήσει. Λίγο παρακάτω θα πούμε ότι αυτή τη βασική μονάδα πληροφορίας την λέμε ένα bit. Συνήθως, ένα bit από μόνο του είναι "μικρή" ποσότητα πληροφορίας, και δεν μας επαρκεί για τις περισσότερες δουλειές που θέλουμε να κάνουμε.

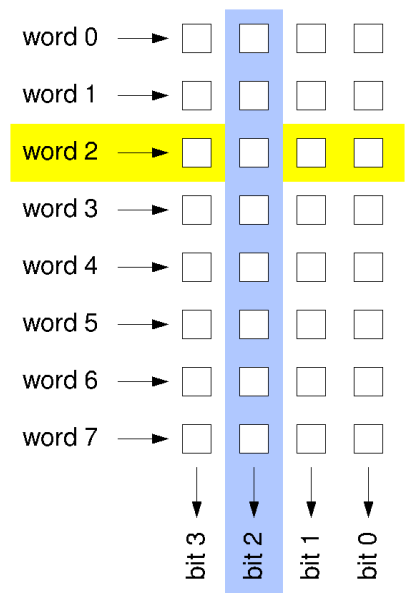
Έτσι, συνήθως, φτιάχνουμε τις μνήμες έτσι που η κάθε λέξη τους να περιέχει κάμποσα bits. Στο σχήμα δεξιά, οι λέξεις περιέχουν οκτώ (8) bits η καθεμία: είναι τα 8 μικρά κουτάκια που φαίνονται σε κάθε λέξη, το ένα δίπλα στο άλλο. Καθένα τους μπορεί, π.χ., να πει σε μία λαμπίτσα να ανάψει ή να σβήσει, άρα όλα μαζί τα 8 bits της κάθε μίας λέξης εδώ, μπορούν να αναβοσβήνουν 8 λαμπίτσες --όπως θα κάνουμε και στο επόμενο πείραμα.



Στο παραπάνω σχήμα, από τις 16 λέξεις που υπάρχουν, έχουμε ξεχωρίσει δύο τυχαίες --την πράσινη και την κίτρινη. Η πράσινη λέξη περιέχει (αυτή τη στιγμή) την πληροφορία "10000101", δηλαδή λέει στην αριστερή λάμπα να ανάψει, στις επόμενες 4 λάμπες να σβήσουν, κ.ο.κ. Η κίτρινη λέξη περιέχει (αυτή τη στιγμή) την πληροφορία "01101100", δηλαδή λέει στην αριστερή λάμπα να σβήσει, στις επόμενες 2 λάμπες να ανάψουν, κ.ο.κ. Οι συνηθισμένες μνήμες έχουν αποθηκευμένα μέσα τους χιλιάδες, εκατομμύρια, δισεκατομμύρια, ή και τρισεκατομμύρια bits --δεν χρειαζόμαστε, και δεν μπορούμε να τα βλέπουμε και να τα επεξεργαζόμαστε όλα αυτά μαζί! Κάθε φορά λοιπόν, θέλουμε να βλέπουμε και να επεξεργαζόμαστε (όλα) τα bits μίας (και μόνο μίας) λέξης. Εδώ, η μνήμη μας περιέχει συνολικά 16x8 = 128 bits (κουτάκια), από τα οποία, κάθε φορά, θέλουμε να βλέπουμε και να επεξεργαζόμαστε 8 μόνον από αυτά --τα 8 bits μίας (και μόνο μίας) από τις 16 λέξεις. Για να μπορούμε λοιπόν να βλέπουμε ("διαβάζουμε") τα 8 bits μίας (διαφορετικής κάθε φορά) λέξης, η μνήμη μας πρέπει να έχει 8 σύρματα "δεδομένων" (data), που στο σχήμα φαίνονται να βγαίνουν στο κάτω μέρος της μνήμης.

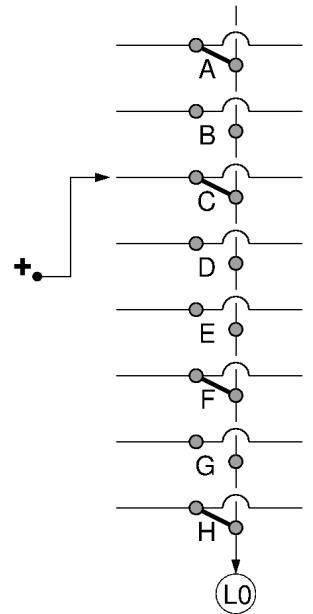
Στο παράδειγμα του σχήματος, αυτή τη στιγμή, θέλουμε να δούμε (διαβάσουμε) την κίτρινη λέξη --την πράσινη θα την αναζητήσουμε αργότερα, πάντως όχι τώρα. Αυτό σημαίνει ότι --τώρα-- τα 8 σύρματα των data, κάτω, πρέπει να πάρουν τις τιμές "01101100", δηλαδή το αριστερό σβηστό, τα 2 επόμενα αναμένα, κ.ο.κ. Λίγο αργότερα, όταν θελήσουμε να διαβάσουμε την πράσινη λέξη, τότε τα σύρματα αυτά θα πρέπει να πάρουν την τιμή "10000101", δηλαδή το αριστερό αναμένο, τα 4 επόμενα σβηστά, κ.ο.κ. Για να γίνουν όλα αυτά, κάποιος πρέπει να πει στη μνήμη ότι τώρα "θέλω" την κίτρινη λέξη, και λίγο αργότερα ότι τότε θα θελήσω την πράσινη. Αυτός ο "κάποιος" είναι το σήμα *Επιλογής (Select)*, ή *Διεύθυνσης (Address)*, στο οποίο αναφερθήκαμε στη ακριβώς προηγούμενη ενότητα και θα ξανα-αναφερθούμε και στις επόμενες. Στο σχήμα, η επιλογή αυτή φαίνεται κόκκινη. Το πώς αυτή η επιλογή μπορεί να "κωδικοποιηθεί" --εδώ με 4 bits-- θα το πούμε στο επόμενο εργαστήριο.

Το επόμενο σχήμα, εδώ δεξιά, μας υποδεικνύει τι χρειαζόμαστε για να φτιάξουμε μιά μνήμη. Εδώ, για απλότητα, μικρώναμε λίγο τη μνήμη: αυτή αποτελείται από 8 λέξεις (τις word0, word1, ..., word7), και η κάθε λέξη αποτελείται από 4 bits. Εδώ εμείς θα κοιτάξουμε πώς θα φτιάξουμε το δεύτερο bit από αριστερά –το bit2, το γαλάζιο– και κατ' αναλογία θα φτιαχτούν και τα υπόλοιπα. Αν κοιτάξουμε προσεκτικά το σχήμα, θα διαπιστώσουμε ότι το "γαλάζιο" σύρμα, "bit2", κάτω, πρέπει να είναι η έξοδος ενός *πολυπλέκτη* ο οποίος θα επιλέγει ένα από τα 8 γαλάζια κουτάκια (bits). Όταν η επιλογή λέει, όπως εδώ, ότι "θέλω την κίτρινη λέξη", word2, τότε ο πολυπλέκτης μας θα πρέπει να επιλέγει το τρίτο κουτάκι από επάνω, και ό,τι πληροφορία περιέχει αυτό να την βγάζει στο γαλάζιο σύρμα, κάτω. Όταν η Επιλογή (Διεύθυνση) λέει ότι θέλουμε μιάν άλλη λέξη, π.χ. την word7, τότε ο γαλάζιος πολυπλέκτης μας θα πρέπει να επιλέγει το κάτω-κάτω γαλάζιο κουτάκι (πληροφορία, bit) και να το βγάζει στο κάτω γαλάζιο σύρμα. Αυτήν ακριβώς τη λειτουργία πολυπλέκτη, για να δουλέψει μιά μνήμη, θα την δούμε στην επόμενη ενότητα, και θα την φτιάξετε στο παρακάτω πείραμα.



### 1.8 Μνήμη Χειροκίνητης Εγγραφής και Ηλεκτρονικής Ανάγνωσης

Θα δούμε τώρα πώς φτιάχνονται οι μνήμες, βασικά με μερικούς μεγάλους πολυπλέκτες. Οι πολυπλέκτες είναι για την *ανάγνωση* από τις μνήμες. Η εγγραφή στη μνήμη απαιτεί άλλα κυκλώματα, πιο πολύπλοκα, που θα τα δούμε λίγο αργότερα. Η μνήμη που θα δούμε εδώ, επειδή ηλεκτρονικά μόνο να την διαβάσει κανείς μπορεί, λέγεται *ROM* (Read-Only Memory). Παρ' ότι δεν μπορούμε να γράψουμε ηλεκτρονικά στη μνήμη που θα δούμε, εν τούτοις θα μπορούμε να γράψουμε *χειροκίνητα* σε αυτήν --δηλαδή να αλλάξουμε με το χέρι τις πληροφορίες που είναι αποθηκευμένες σε αυτήν.



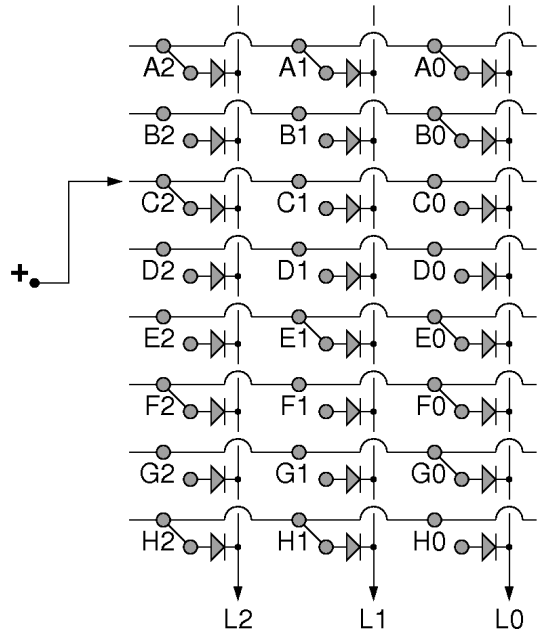
Το σχήμα δεξιά είναι παρόμοιο με το προηγούμενο, δηλαδή είναι ένας πολυπλέκτης, μεγέθους, εδώ, 8-σε-1. Η κύρια διαφορά από το προηγούμενο --πέρα από το σχήμα των συρμάτων, που όμως δεν επηρεάζει τη λειτουργία τους-- είναι ότι στη θέση των διακοπών δεδομένων βάλουμε *βραχυκυκλωτήρες* (jumpers): Σε κάθε διασταύρωση οριζώντιου σύρματος με το κατακόρυφο, υπάρχουν δύο επαφές --από μία σε κάθε σύρμα. Σε μερικά από αυτά τα ζευγάρια (και συγκεκριμένα στα A, C, F, και H) τοποθετήσαμε (με το χέρι!) "βραχυκυκλωτήρες", δηλαδή σύρματα (σχεδιασμένα διαγώνια) που ενώνουν ηλεκτρικά μεταξύ τους τις δύο επαφές, άρα το οριζόντιο εκείνο σύρμα με το κατακόρυφο. Στις άλλες διασταυρώσεις (στο παράδειγμά μας, στις B, D, E, και G) δεν τοποθετήσαμε βραχυκυκλωτήρες, άρα τα αντίστοιχα οριζόντια σύρματα *δεν* συνδέονται με το κατακόρυφο. Η διαφορά με το προηγούμενο κύκλωμα είναι ότι ο βραχυκυκλωτήρας *παραμένει* στη θέση του από μόνος του, χωρίς εξωτερική επέμβαση ή πίεση, σε αντίθεση με το διακόπτη που για να μείνει αναμένος πρέπει κάποιος συνεχώς να τον πατάει. Με άλλα λόγια, ο βραχυκυκλωτήρας *έχει μνήμη*, αφού "θυμάται" πού τον βάλουμε, και παραμένει εκεί, σε αντίθεση με τον απλό διακόπτη (bouton) που δεν έχει μνήμη, αφού μόλις τον αφήσεις μόνο του αυτός πάντα σβήνει, ό,τι και να του'χεις κάνει πριν.

Έτσι, το σχήμα εδώ είναι μιά *Μνήμη με 8 αποθηκευμένες πληροφορίες*: στις θέσεις A, C, F, και H η αποθηκευμένη πληροφορία λέει "άναψε λάμπα L0", ενώ στις θέσεις B, D, E, και G η αποθηκευμένη πληροφορία λέει "μείνε σβηστή λάμπα L0". Εάν θέλουμε να αλλάξουμε την πληροφορία που είναι αποθηκευμένη σε κάποια θέση, σε αυτήν εδώ τη μνήμη μόνο χειροκίνητα μπορούμε να το κάνουμε: να βγάλουμε από εκεί τον βραχυκυκλωτήρα που υπήρχε, ή να βάλουμε εκεί έναν αν δεν υπήρχε. Στο σχήμα, ο χρήστης επέλεξε, με το σύρμα αριστερά από την θετική τροφοδοσία, να "διαβάσει" τη θέση C της μνήμης, οδηγώντας την αποθηκευμένη πληροφορία της στην έξοδο (λάμπα) L0: επειδή



στη θέση C υπάρχει βραχυκυκλωτήρας, η "πληροφορία" αυτή κάνει την λάμπα L0 να ανάψει. Εάν, αντίθετα, ο χρήστης μετακινήσει το σύρμα επιλογής (αριστερά) π.χ. στη θέση G, επειδή στη θέση εκείνη δεν υπάρχει βραχυκυκλωτήρας και τα σύρματα παραμένουν ασύνδετα, η πληροφορία αυτή θα αντικατοπτριστεί στην L0 με σβηστή λάμπα.

Μία μνήμη με ένα μόνο σύρμα εξόδου δεν είναι πολύ χρήσιμη --συνήθως χρειαζόμαστε μεγαλύτερες (φαρδύτερες) μνήμες, που όταν διαβάζουμε από μία θέση τους να διαβάζουμε περισσότερες από μία πληροφορίες. Στο σχήμα εδώ δεξιά "τριπλασιάσαμε" (οριζόντια) την προηγούμενη μνήμη: έχουμε εδώ μία **Μνήμη 8x3**, δηλαδή με 8 ομάδες (γραμμές) πληροφοριών --τις λέμε **8 λέξεις**-- και όπου η κάθε μία από αυτές τις 8 λέξεις περιέχει 3 "πληροφορίες" μέσα της, οι οποίες ελέγχουν 3 λαμπάκια, L2, L1, και L0 --τα λέμε **3 bits ανά λέξη**. Ο περίφημος όρος **bit** προέρχεται από τις λέξεις **binary digit**, και Ελληνικά αποδίδεται πολλές φορές σαν **δυφίο**, από τις λέξεις **δυαδικό ψηφίο**. Ένα bit είναι η ποσότητα της πληροφορίας που μπορεί μόλις να επιλέξει μία από **δύο** επιλογές --ας πούμε, εδώ, σβηστή ή αναμένη λαμπίτσα. Για τις μνήμες που είναι φαρδύτερες από ένα bit, όπως αυτή εδώ, απαιτείται η χρήση διόδων στα σημεία διασταύρωσης, όπως δείχνει το σχήμα, προκειμένου αυτές να επιβάλουν το ρεύμα να ρέει πάντα από την είσοδο επιλογής (αριστερά) προς τις εξόδους δεδομένων (κάτω), και ποτέ ανάποδα από ένα κατακόρυφο σύρμα πολυπλέκτη πίσω σε οριζόντια σύρματα άλλα από το μοναδικό επιλεγμένο οριζόντιο σύρμα.



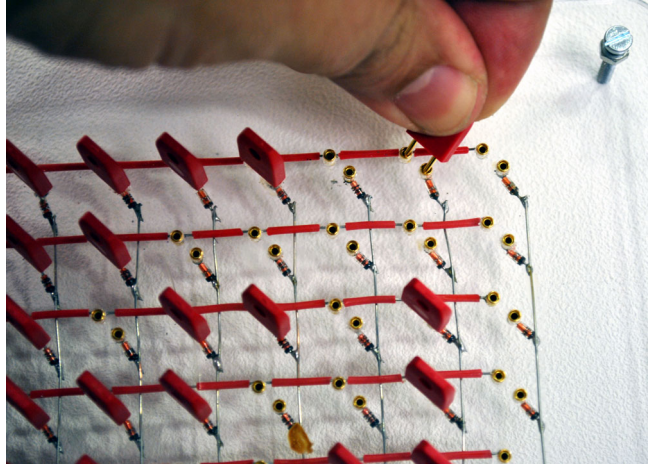
Οι μνήμες είναι βασικά στοιχεία όλων των ψηφιακών συστημάτων και των υπολογιστών. Η βασική οργάνωσή τους, που φαίνεται εδώ, είναι πρωταρχικής σημασίας: *λέξεις*, που αποτελούν οριζόντιες ομάδες από bits, γραμμένες ή μία κάτω από την άλλη, όπου κάθε λέξη αποτελείται από *bits*, που γράφονται μέσα στη λέξη, το ένα δίπλα στο άλλο. Όποτε διαβάζουμε κάτι από τη μνήμη, διαβάζουμε πάντα όλα τα bits μίας **ολόκληρης λέξης**. Επομένως, η μνήμη έχει τόσα σύρματα εξόδου δεδομένων (κάτω στο σχήμα) όσα bits έχει η κάθε λέξη της-- στο εδώ παράδειγμα έχουμε 3 εξόδους δεδομένων, L2, L1, και L0, και κάθε λέξη έχει 3 bits: η λέξη A έχει τα bits A2, A1, A0, κ.ο.κ. Στο σχήμα, η είσοδος επιλογής (το σύρμα αριστερά από τον θετικό πόλο της μπαταρίας) αυτή τη στιγμή επιλέγει να διαβάσουμε τη λέξη C, η οποία αποτελείται από 3 bits: C2, C1, C0. Στο C2 έχουμε βάλει βραχυκυκλωτήρα, ενώ στα C1 και C0 όχι. Επομένως, η πληροφορία που είναι αποθηκευμένη σε αυτή τη λέξη είναι: "αναμένη, σβηστή, σβηστή", και όντως, το σύρμα L2 θα πάρει ρεύμα μέσω του βραχυκυκλωτήρα C2, ενώ τα σύρματα L1 και L0 θα παραμείνουν χωρίς ρεύμα (σβηστά) επειδή είναι ασύνδετα με το οριζόντιο σύρμα C --που είναι το μοναδικό επιλεγμένο (τροφοδοτημένο από την μπαταρία) οριζόντιο σύρμα. Εάν το σύρμα επιλογής, αριστερά, μετακινηθεί π.χ. στη θέση F, τότε οι εξόδοι δεδομένων θα γίνουν "αναμένη, σβηστή, αναμένη", επειδή οι διασταυρώσεις F2 και F0 έχουν βραχυκυκλωτήρες, ενώ η F1 δεν έχει. Έτσι, τότε, θα έχουμε διαβάσει τα 3 bits της λέξης F, αστα οποία είχαμε αποθηκεύσει (στο παρελθόν, χειροκίνητα, μέσω των βραχυκυκλωτήρων) τις πληροφορίες (3 bits) "αναμένο, σβηστό, αναμένο".

### **Πείραμα 1.9: Μνήμη 8x8 με Jumpers και οδήγηση 7-Segment-Display**

Στο εργαστήριο θα βρείτε πλακέτες (διαφανείς) με το παραπάνω κύκλωμα μνήμης ROM, μεγέθους 8x8, δηλαδή 8 λέξεις (8 γραμμές), η κάθε μία των 8 bits (8 στήλες). Οι βραχυκυκλωτήρες (jumpers) είναι μικρά κόκκινα "πριζάκια", με δύο ποδαράκια καθέναν· εσωτερικά, τα δύο ποδαράκια είναι ενωμένα (βραχυκυκλωμένα) μεταξύ τους. Στη φωτογραφία δεξιά βλέπετε ένα κοντινό πλάνο της πάνω δεξιάς γωνίας της μνήμης, και φαίνεται πώς μπαίνουν και βγαίνουν (χειροκίνητα!) τα πριζάκια/jumpers. Παρακαλούμε προσέχετε να **μην χάνετε τους βραχυκυκλωτήρες** αυτούς!

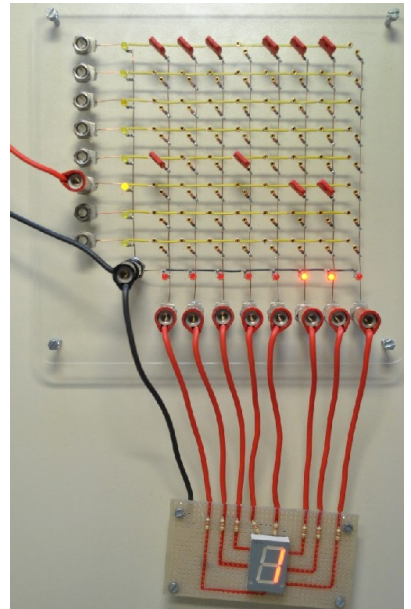
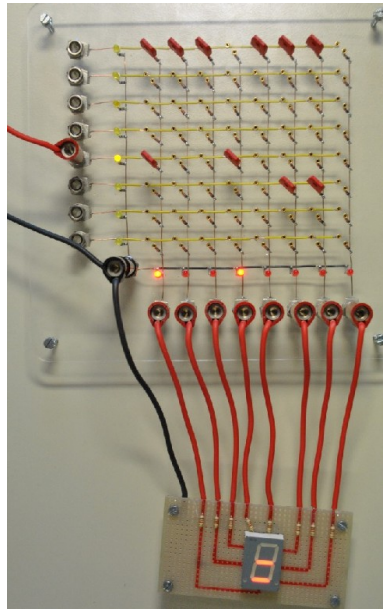
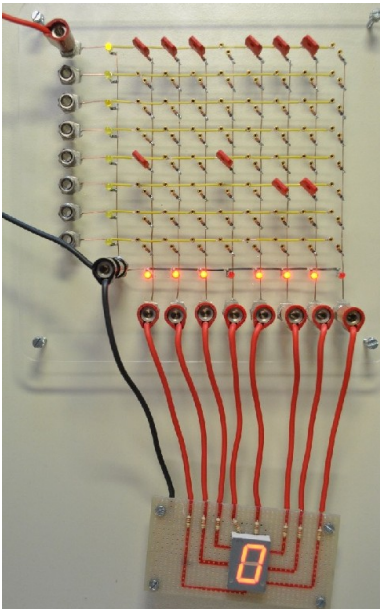
Θα βρείτε την πλακέτα συνδεδεμένη σε έναν ενδείκτη 7 τμημάτων (7-segment display), όπως δείχνουν οι παρακάτω φωτογραφίες. Ο ενδείκτης 7 τμημάτων έχει 7 γραμμικές LED's που σχηματίζουν το γνώριμό μας σχήμα ενός δεκαδικού ψηφίου (και έχει και μία δεκαδική

υποδιαστολή --συνολικά 8 LED's πουτροοδοτούνται από 8 σύρματα). Η πλακέτα μνήμης παίρνει τροφοδοσία 12 Volt όπως φαίνεται στις φωτογραφίες: γείωση (αρνητικό - μαύρο) στην μαύρη υποδοχή, και θετικό (κόκκινο) σε **μία και μόνο μία** γραμμή-λέξη, την "επιλεγμένη" λέξη την οποία θέλετε να διαβάσετε. Επίσης, η πλακέτα μνήμης πρέπει να δίνει γείωση στην πλακέτα 7-segment, μέσω του μαύρου καλωδίου. Η θετική τροφοδοσία φτάνει στις LED του 7-segment display μέσα από τις εξόδους δεδομένων της μνήμης, με 8 κόκκινα καλώδια.



**ΠΡΟΣΟΧΗ:** μην βραχυκνκλώσετε (ενώσετε κατευθείαν) ποτέ οιαδήποτε έξοδο δεδομένων της μνήμης (οιαδήποτε από τις κάτω δεξιά οκτώ υποδοχές) με την "γείωση" --την μαύρη υποδοχή: εάν το κάνετε, θα κάψετε την δίοδο που υπάρχει στο bit εκείνο της μνήμης στη γραμμή που έχετε εκείνη τη στιγμή ενεργοποιημένη!

Στις παρακάτω φωτογραφίες έχουμε αποθηκεύσει στη μνήμη (μέσω της θέσης των jumpers) τις πληροφορίες (bits) που χρειάζονται ώστε να εμφανίζονται στον ενδείκτη: το ψηφίο "0" όταν επιλέγεται (διαβάζεται) η πρώτη (επάνω) λέξη (γραμμή) της μνήμης: το σύμβολο "=" όταν διαβάζουμε την πέμπτη γραμμή της μνήμης: και το ψηφίο "1" όταν διαβάζουμε την έκτη γραμμή της μνήμης. Η επιλογή της λέξης που θέλουμε να διαβάσουμε γίνεται μετακινώντας το κόκκινο καλώδιο τροφοδοσίας.



Αποθηκεύστε (γράψτε χειροκίνητα, βγάζοντας/βάζοντας πριζάκια) στη μνήμη σας τις κατάλληλες πληροφορίες για να εμφανίζονται στον ενδείκτη σας διάφορα σύμβολα, όπως π.χ.: (α) 0, 1, 2, 3, 4, 5, 6, 7· ή (β) A, b, c, d, E, F, H, J· ή (γ) 0, 2, 4, 6, 8, =, Ξ, Π· κ.ο.κ. Αφού τα γράψετε, μετά διαβάστε τα περιεχόμενα της μνήμης σας, πολλαπλές φορές, με τη σειρά ή ανακατωμένα, και διαπιστώστε ότι η μνήμη σας όντως θυμάται ότι της γράψατε τελευταίο!

[Up to the Home Page of CS-120](#)

© copyright University of Crete, Greece.  
last updated: 28 Sep. 2023, by [M. Katevenis](#).