

Εισαγωγή στους Η/Υ:
Οργάνωση και Λειτουργία

Σημειώσεις του μαθήματος

Πάτρα, 2022

Περιεχόμενα

1^ο ΚΕΦΑΛΑΙΟ – Η/Υ: ΜΙΑ ΣΥΝΤΟΜΗ ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ.....	4
1.1 ΠΡΟΒΛΗΜΑΤΑ, ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΤΑ ΟΡΙΑ ΤΗΣ ΑΛΓΟΡΙΘΜΙΚΗΣ ΥΠΟΛΟΓΙΣΙΜΟΤΗΤΑΣ: ΘΕΩΡΗΤΙΚΕΣ ΘΕΜΕΛΙΩΣΕΙΣ ΤΗΣ ΕΠΙΣΤΗΜΗΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ.....	4
1.2 ΤΕΧΝΟΛΟΓΙΚΕΣ ΕΞΕΛΙΞΕΙΣ.....	7
2^ο ΚΕΦΑΛΑΙΟ – ΟΡΓΑΝΩΣΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΕΝΟΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΥΠΟΛΟΓΙΣΤΗ.....	9
2.1 Η ΚΕΝΤΡΙΚΗ ΜΟΝΑΔΑ ΕΠΕΞΕΡΓΑΣΙΑΣ (CPU)	13
2.2 ΜΝΗΜΗ CACHE.....	16
2.3 Ο ΚΥΚΛΟΣ ΜΗΧΑΝΗΣ ΚΑΙ Ο ΚΥΚΛΟΣ ΕΝΤΟΛΗΣ ΜΙΑΣ CPU	20
2.4 Η ΚΥΡΙΑ ΜΝΗΜΗ	26
2.5 ΣΥΣΤΗΜΑΤΑ ΑΡΙΘΜΗΣΗΣ.....	27
2.5.1 Μετατροπή από οποιοδήποτε Σύστημα Αρίθμησης στο Δεκαδικό Σύστημα.....	29
2.5.2 Μετατροπή από το Δεκαδικό σε οποιοδήποτε άλλο Σύστημα Αρίθμησης	29
2.6 ΕΙΣΑΓΩΓΗ ΣΤΑ ΛΟΓΙΚΑ ΚΥΚΛΩΜΑΤΑ	31
2.6.1 Άλγεβρα Boole.....	31
2.6.2 Συναρτήσεις Boole.....	32
2.6.3 Λογικές Πύλες.....	32
2.6.4 Πύλες με περισσότερες από δύο εισόδους.....	34
2.7 ΔΙΑΓΡΑΜΜΑΤΑ ΡΟΗΣ (FLOW CHARTS)	35
2.8 ΣΥΝΟΠΤΙΚΟΣ ΟΔΗΓΟΣ ΨΕΥΔΟΚΩΔΙΚΑ	38
Μεταβλητή.....	38
Τύπος.....	38
Σταθερές.....	39
Τιμή	39
Τελεστές	39
Εκφράσεις	39
Δηλώσεις μεταβλητών.....	40
Δηλώσεις Σταθερών	40
Τύπος πίνακα	41
Βασικές εντολές ψευδοκώδικα	42
Εντολή καταχώρησης (:=)	43
Εντολή για είσοδο δεδομένων (ΔΙΑΒΑΣΕ)	43
Εντολή για έξοδο αποτελεσμάτων (ΤΥΠΩΣΕ)	44
Εντολή/Δομή επιλογής	45
Σύνταξη.....	45
Λειτουργία.....	46
Εντολές/Δομές επανάληψης	47
ΓΙΑ-ΕΠΑΝΑΛΑΒΕ.....	47
ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ	49
ΕΠΑΝΑΛΑΒΕ-ΜΕΧΡΙ	50
3^ο ΚΕΦΑΛΑΙΟ – ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΕΣ ΑΣΚΗΣΕΙΣ.....	52
3.1 ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΕΣ ΑΣΚΗΣΕΙΣ ΜΕΤΑΤΡΟΠΩΝ ΜΕΤΑΞΥ ΔΥΑΔΙΚΟΥ ΚΑΙ ΔΕΚΑΔΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ..	52
3.2 ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΗ ΑΣΚΗΣΗ ΓΙΑ ΕΚΤΕΛΕΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΑΠΟ ΜΙΑ CPU	53
3.3 ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΗ ΑΣΚΗΣΗ ΓΙΑ ΛΟΓΙΚΑ ΚΥΚΛΩΜΑΤΑ.....	54
3.4 ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΗ ΑΣΚΗΣΗ ΓΙΑ ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΨΕΥΔΟΚΩΔΙΚΑ.....	54
3.5 ΑΝΤΙΠΡΟΣΩΠΕΥΤΙΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA	56
4 ΚΕΦΑΛΑΙΟ – ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ	58
4.1 ΚΡΥΠΤΟΓΡΑΦΙΑ	62
4.1.1 Συμμετρικά και Μη Συμμετρικά Κρυπτογραφικά Συστήματα.....	65
4.1.2 Ψηφιακές Υπογραφές.....	71
4.2 ΠΡΟΣΩΠΙΚΑ ΔΕΔΟΜΕΝΑ ΚΑΙ ΙΔΙΩΤΙΚΟΤΗΤΑ	72
4.2.1 Ανωνυμία.....	74
4.2.2 Μη-συνδεσιμότητα.....	77

4.2.3 Ψευδωνυμία.....	78
4.2.4 Διαχείριση Ταυτότητας.....	78
4.3 ΑΠΑΙΤΗΣΕΙΣ ΤΑΥΤΟΠΟΙΗΣΗΣ	79
4.3.1 Αυθεντικοποίηση.....	80
4.3.2 Εξουσιοδότηση	80

1^ο Κεφάλαιο – Η/Υ: Μία σύντομη ιστορική αναδρομή

1.1 Προβλήματα, αλγόριθμοι και τα όρια της αλγοριθμικής υπολογισιμότητας: Θεωρητικές Θεμελιώσεις της Επιστήμης της Πληροφορικής

Στις αρχές του αιώνα μας, ένας από τους μεγαλύτερους μαθηματικούς, ο David Hilbert, στα πλαίσια της περίφημης διάλεξής του στο Διεθνές Συνέδριο Μαθηματικών στο Παρίσι το 1900 αναφέρθηκε σε 23 ανοικτά προβλήματα τα οποία θεωρούσε ως τα πιο σημαντικά για εκείνη την εποχή. Το δέκατο από αυτά ρωτούσε εάν μπορούσε να βρεθεί μια διαδικασία που να απαντά στο εάν μία δοσμένη *Διοφαντική Εξίσωση* έχει λύση ή όχι.

Αρκετά χρόνια μετά, το 1928, στο Διεθνές Συνέδριο Μαθηματικών στη Μπολόνια και ως συνέχεια της προηγούμενης διαλέξής του, έθετε το πρόβλημα του να βρεθεί διαδικασία που θα μπορούσε να ελέγχει εάν ένα μαθηματικό (λογικό) σύστημα, όπως είναι για παράδειγμα η Ευκλείδειος Γεωμετρία, είναι συνεπές, δηλαδή δεν περιέχει αντιφάσεις. Είναι προφανές το πόσο χρήσιμη θα ήταν μια τέτοια διαδικασία, καθώς ένας μαθηματικός που προτείνει ένα πολύπλοκο μαθηματικό σύστημα, το οποίο λόγω του όγκου των αξιωμάτων ή της πληθώρας των αποδεικτικών μεθόδων που περιέχει ξεφεύγει από την δυνατότητα ελέγχου από τον άνθρωπο, απλά θα τροφοδοτούσε (με μια κατάλληλη αναπαράσταση) το μαθηματικό του σύστημα στη μηχανική διαδικασία και η διαδικασία θα απαντούσε στο ερώτημα του αν το σύστημα αυτό περικλείει αντιφάσεις. Ένα άλλο πρόβλημα που έθεσε ο Hilbert στη διαλέξή του το 1928 και που ήταν κατά κάποιον τρόπο γενίκευση του δέκατου προβλήματος της διάλεξης του 1900, ήταν δοθέντος ενός μαθηματικού συστήματος να διαπιστωθεί εάν μια δοσμένη μαθηματική πρόταση αποτελεί θεώρημα του συστήματος ή όχι, το περίφημο Entscheidungsproblem. Ο Hilbert ήλπιζε ότι για τα παραπάνω προβλήματα θα έπρεπε να υπήρχε κάποια διαδικασία που να τα απαντά μέσα σε πεπερασμένο αριθμό βημάτων (περατοκρατική διαδικασία ή αλγόριθμος, όπως θα δούμε πιο κάτω).

Το 1931, ο μαθητής του Hilbert και εξίσου μεγάλος μαθηματικός Kurt Gödel, απέδειξε σε μία από τις πιο σημαντικές εργασίες του ότι, δυστυχώς δύο από τις πεποιθήσεις του Hilbert ήταν λανθασμένες. Πιο συγκεκριμένα, ο Gödel παρουσίασε μια μαθηματική πρόταση στα πλαίσια του αξιωματικού συστήματος των φυσικών αριθμών (η πρώτη προσπάθεια αξιωματοποίησης του συστήματος των φυσικών

αριθμών έγινε το 1910 από τους Whitehead και Russell, στο τρίτομο έργο τους Principia Mathematica για την οποία δεν ήταν δυνατό να αποδειχθεί με χρήση των αποδεικτικών μεθόδων του συστήματος αυτού ούτε ότι είναι θεώρημα ούτε ότι δεν είναι. Αυτή η μεγάλη ανακάλυψη, είναι το περίφημο θεώρημα μη πληρότητας του Gödel. Επιπλέον, ο Gödel απέδειξε ότι η συνέπεια ενός μαθηματικού συστήματος δεν είναι δυνατόν να αποδειχθεί με κανόνες και μεθόδους του ίδιου του συστήματος.

Το 1936 ο Alan Turing γινόταν ο θεμελιωτής της επιστήμης των υπολογιστών, με την ιστορική εργασία του “On computable numbers, with an application to the Entscheidungsproblem”. Στην εργασία αυτή ο Turing όρισε ένα υπολογιστικό μοντέλο, που ονομάστηκε μετά προς τιμή του Μηχανή Turing (Turing Machine) το οποίο δεν είναι τίποτε άλλο παρά μια αφαιρετική περιγραφή, ένα μοντέλο, της έννοιας της αλγοριθμικής υπολογισιμότητας και του ηλεκτρονικού υπολογιστή όπως τον ξέρουμε σήμερα.

Ο Alan Turing ακολούθησε μία «ταπεινή», αλλά μεγαλοφυή (στην απλότητά της), προσέγγιση. Σκέφτηκε ως εξής: «Δεν ξέρω πώς «υλοποιούνται» οι εκπληκτικές ικανότητες του ανθρώπου, όπως η αντίληψη νοημάτων, η αφαιρετικότητα εννοιών, ακόμη και η συνείδηση όμως ξέρω κάτι «ταπεινό» που κάνει ο άνθρωπος που σίγουρα μπορώ να το «μηχανοποιήσω»: την πρόσθεση δύο ακεραίων.» Αυτό ήταν! Ο Turing είχε θέσει τις βάσεις για την μαθηματική διερεύνηση της βαθιάς έννοιας της υπολογισιμότητας! Το απόσταγμα της οπτικής γωνίας με την οποία εξέτασε ο Turing την έννοια της «μηχανικής υπολογισιμότητας ήταν το περίφημο *Αίτημα των Church-Turing* (Church-Turing Thesis) που συνοψίζεται ως εξής:

«Οι αποτελεσματικά υπολογίσιμες συναρτήσεις και τα αλγοριθμικά επιλύσιμα προβλήματα είναι αυτά ακριβώς που μπορούν να υπολογίσουν οι μηχανές Turing (οι H/Y δηλαδή).»

Αυτή η διατύπωση καλείται «αίτημα» (Thesis) και όχι «θεώρημα» (Theorem) καθώς κανείς δεν γνωρίζει (και δεν είναι δυνατόν να γνωρίζει!) αν ο μόνος τρόπος υπολογισμού στο σύμπαν είναι ο τρόπος των μηχανών Turing ή υπάρχουν και διαφορετικά μοντέλα υπολογισμού!

Επιπλέον, απαντώντας αρνητικά και στο τρίτο πρόβλημα του Hilbert, Entscheidungsproblem, ο Alan Turing απέδειξε ότι το πρόβλημα του εάν μια Μηχανή Turing τερματίζει για μια δοσμένη είσοδο, είναι μη αλγοριθμικά αποφασίσιμο. Με

άλλα λόγια, δεν υπάρχει αλγόριθμος (Μηχανή Turing δηλαδή) που να δέχεται ως είσοδο την περιγραφή της μηχανής και της εισόδου της και να αποφασίζει αν κάποτε θα σταματήσει η μηχανή.

Η συνεισφορά του Turing ήταν διπλή. Κατ' αρχήν συμπλήρωσε το αποτέλεσμα του Gödel. Παρεκκλίνοντας λίγο, στην πραγματικότητα, η απόδειξη του Turing ήταν μια απλούστερη έκδοση της απόδειξης μη πληρότητας του Gödel. Όμως, ο Turing είχε το πλεονέκτημα ότι εργαζόταν σε ένα μαθηματικό σύστημα που διευκόλυνε το έργο της επίδειξης μιας πρότασης για την οποία δεν υπάρχει αλγόριθμος που να αποφασίζει εάν είναι θεώρημα ή όχι. Η πρόταση, όπως είπαμε, είναι η εξής: Η μηχανή Turing Μ τερματίζει κάποτε με είσοδο τη συμβολοσειρά x ; Ο Gödel, από την άλλη μεριά, είχε το μειονέκτημα ότι εργαζόταν σε ένα κάπως δύσκαμπτο φορμαλισμό, αυτόν των αναδρομικών συναρτήσεων, και έπρεπε να εφεύρει, κατά κάποιον τρόπο, μία μέθοδο γραφής μαθηματικών προτάσεων με χρήση απλά και μόνο αναδρομικών συναρτήσεων σε φυσικούς αριθμούς. Εάν προσέξει κανείς την απόδειξή του, θα διαπιστώσει ότι, πριν επιδείξει την περίφημη αυτοαναφερόμενη πρόταση που δεν μπορεί να αποδειχθεί εάν είναι θεώρημα ή όχι, χτίζει βήμα-βήμα μια μηχανή, που προσομοιάζει το μηχανισμό υπολογισμού της γνωστής συναρτησιακής γλώσσας προγραμματισμού Lisp. Αυτή είναι και η ομορφιά της απόδειξης του. Επανερχόμενοι όμως στην επόμενη συνεισφορά του Turing, η οποία είχε μεγάλη επίδραση στη μετέπειτα πορεία της επιστήμης των υπολογιστών, μέσα από τη χρησιμοποίηση της μηχανής που πρότεινε για την περιγραφή αλγόριθμων για διάφορα προβλήματα, υπήρξε η συνειδητοποίηση ότι πέρα από τη διαπίστωση ότι ένα πρόβλημα είναι επιλύσιμο (με το να επιδείξουμε έναν αλγόριθμο, ή μηχανή Turing, που να το επιλύει) μας ενδιαφέρει η επίλυση να γίνεται όσο το δυνατόν γρηγορότερα.

Ο πρώτος που διερεύνησε ζητήματα σχετικά με την πολυπλοκότητα της αλγοριθμικής επίλυσης προβλημάτων, ήταν ο ίδιος ο Gödel το 1956, πάλι σε σχέση με απόδειξη θεωρημάτων στα πλαίσια αξιωματικών συστημάτων. Με ένα γράμμα του προς τον John von Neumann, ο Gödel αναρωτιέται σχετικά με την πολυπλοκότητα εύρεσης αποδείξεων σε θεωρήματα που είναι διατυπωμένα με βάση τους κανόνες ενός τυπικού συστήματος. Πιο συγκεκριμένα, αναφέρεται στη μηχανή Turing ως υπολογιστικό μοντέλο, και μετά ρωτά ποια είναι η συνάρτηση που φράσσει τον αριθμό των βημάτων που χρειάζονται για να βρεθούν αποδείξεις μήκους n . Στην πραγματικότητα ο Gödel ρωτούσε τον von Neumann, σύμφωνα με τη σημερινή τεχνική ορολογία, για την ντετερμινιστική (deterministic) υπολογιστική πολυπλοκότητα του

προβλήματος της απόδειξης θεωρημάτων (theorem proving). Στο ίδιο γράμμα, ο Gödel επίσης ρωτάει σχετικά με την υπολογιστική πολυπλοκότητα του ελέγχου εάν ένας φυσικός αριθμός είναι πρώτος (primality testing), και μας εκπλήσσει κάπως το γεγονός ότι εκφράζει την πεποίθηση ότι η απόδειξη θεωρημάτων δεν πρέπει να είναι και τόσο δύσκολο πρόβλημα υπολογιστικά. Δυστυχώς, ο von Neumann ήδη έπασχε από καρκίνο και πέθανε ένα χρόνο αργότερα. Ποτέ δεν υπήρξε απάντηση στο γράμμα, και φαίνεται ότι ο Gödel δεν προσπάθησε να διερευνήσει περισσότερο το πολύ σημαντικό ερώτημα που έθεσε.

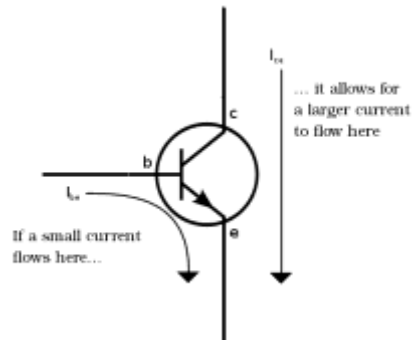
Τέλος, πριν μιλήσουμε για πιο πρακτικά ζητήματα που άπτονται της μηχανικής υπολογισιμότητας όπως αυτή «ενσαρκώνεται» από τους Ηλεκτρονικούς Υπολογιστές με τις διάφορες μορφές που λάμβαναν από το 1936 μέχρι και σήμερα, ανάλογα με τη διαθέσιμη τεχνολογία, πρέπει να τονίσουμε το εξής: ο H/Y είναι απλά ένα εργαλείο της ανθρώπινης διάνοησης, όπως όλα τα άλλα που μας περιτριγυρίζουν και διευκολύνουν τη ζωή και την καθημερινότητά μας. Ο H/Y, όμως διαφέρει από όλες τις υπόλοιπες επινοήσεις του ανθρώπου στα εξής σημεία:

- Είναι ένα εργαλείο «πολυμορφικό», δηλαδή εκτελεί πολλές (ουσιαστικά χωρίς όρια) διαφορετικές υπολογιστικές εργασίες που περιγράφει ο άνθρωπος χωρίς να μεταβάλλεται, ως υλικό και κατασκευή.
- Δέχεται, από τον άνθρωπο, περιγραφές εικονικών κόσμων και μαθηματικών μοντέλων του φυσικού κόσμου και τους «δίνει ζωή» διευκολύνοντας την διερεύνησή τους από την ασφάλεια και άνεση ενός εργαστηρίου H/Y.
- Πολλοί H/Y μπορούν να διασυνδεθούν δημιουργώντας έναν ισχυρότερο «εικονικό» H/Y (με τον μεγαλύτερο από αυτούς ιδεατούς H/Y να είναι το ίδιο το Διαδίκτυο και ο Παγκόσμιος Ιστός που εδράζεται στην υποδομή του Διαδικτύου).
- Είναι, από όλα τα δημιουργήματα του ανθρώπου, ότι πιο κοντινό στην ανθρώπινη νόηση με αποτέλεσμα να μπορεί να τον βοηθήσει ακόμη στην κατανόηση της ίδιας τη διαδικασίας της ανθρώπινης νόησης!

1.2 Τεχνολογικές εξελίξεις

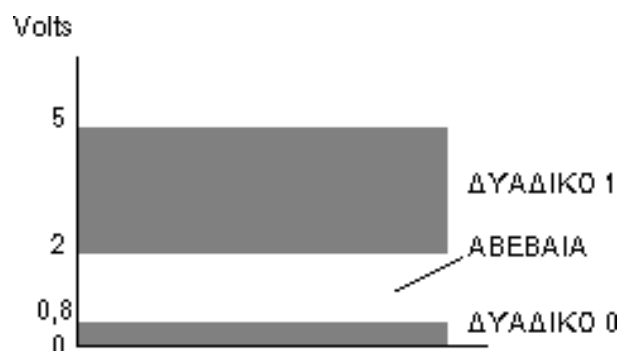
Πολύ σημαντική ιστορική στιγμή ήταν η ανακάλυψη του τρανζίστορ το 1947, καθώς κατάργησε τις λυχνίες κενού που χρησιμοποιούνταν μέχρι τότε για την υλοποίηση λογικών πυλών και κυκλωμάτων και οδήγησε έτσι στη δραματική μείωση του μεγέθους των κυκλωμάτων και κατά συνέπεια των υπολογιστών.

Το ηλεκτρονικό διάγραμμα ενός τυπικού τρανζίστορ φαίνεται στο επόμενο σχήμα:



Στους Η/Υ, το τρανζίστορ λειτουργεί ως διακόπτης on/off (όπως, ακριβώς, ο διακόπτης στα φώτα του σπιτιού μας) και αναπαριστά την στοιχειώδη μονάδα πληροφορίας, το bit, που μπορεί να λαμβάνει τιμές 0 ή 1. Κατά συνέπεια, ένας ηλεκτρονικός υπολογιστής, που σήμερα περιέχει δισεκατομμύρια τρανζίστορ για τους υπολογισμούς του, λειτουργεί χρησιμοποιώντας αναπαράσταση των δεδομένων κάθε είδους με το *δυναδικό σύστημα αρίθμησης* δηλαδή χρησιμοποιώντας για μονάδα πληροφορίας το ένα δυαδικό ψηφίο ή binary digit (bit). Ομαδοποιώντας πολλά bits μαζί μπορεί να αναπαραστήσει οποιαδήποτε πληροφορία, αριθμό, κείμενο, εικόνα ή βίντεο. Συνήθως, οι ομαδοποιήσεις είναι σε δυνάμεις του 2 και, ειδικότερα, 8 bits (= 1 byte), 16 bits, 32 bits, και 64 bits.

Η αιτία όμως που χρησιμοποιούνται ψηφιακά ηλεκτρονικά στους Η/Υ και που μάλιστα ακολουθούν το δυαδικό σύστημα, είναι ότι στην πράξη δεν είναι σχεδόν ποτέ δυνατόν να επιτύχουμε ακριβή επίπεδα τάσεως. Για να φθάσουμε επομένως σε αποδεκτό επίπεδο αξιοπιστίας, είναι αναγκαίο να υιοθετήσουμε περιοχές τάσεων για κάθε τιμή bit. Ένα τέτοιο παράδειγμα φαίνεται στο πιο κάτω σχήμα, όπου το bit 1 αντιστοιχεί στην περιοχή τιμών τάσεως από 2 - 5 Volts και το bit 0 στην περιοχή τιμών τάσεως από 0 - 0,8 Volts. Η περιοχή ανάμεσα στα 0,8 και 2 V θεωρείται περιοχή αβεβαιότητας και δεν αντιστοιχεί σε καμία δυαδική τιμή.



2^ο Κεφάλαιο – Οργάνωση και λειτουργία ενός Ηλεκτρονικού Υπολογιστή

Ένας ηλεκτρονικός υπολογιστής ή υπολογιστής, για συντομία, είναι μια συσκευή επεξεργασίας δεδομένων η οποία εκτελεί τέσσερις βασικές λειτουργίες:

1. Είσοδο: Συγκεντρώνει δεδομένα ή επιτρέπει στους χρήστες να εισάγουν δεδομένα.
2. Επεξεργασία: Χειρίζεται, υπολογίζει ή οργανώνει αυτά τα δεδομένα σε πληροφορίες.
3. Έξοδο: Εμφανίζει δεδομένα και πληροφορίες σε μια μορφή που μπορεί να γίνει κατανοητή από το χρήστη.
4. Αποθήκευση: Αποθηκεύει δεδομένα και πληροφορίες για μεταγενέστερη χρήση.

Ποια, όμως, είναι η διαφορά ανάμεσα στα δεδομένα και τις πληροφορίες; Συχνά χρησιμοποιούμε τους όρους δεδομένα και πληροφορίες για να αναφερθούμε στο ίδιο πράγμα. Ωστόσο, αν και μπορεί να έχουν την ίδια σημασία σε μια απλή συζήτηση, η

διάκριση ανάμεσα στα δεδομένα και τις πληροφορίες είναι σημαντική στην Πληροφορική.

Σε όρους Πληροφορικής, τα *δεδομένα* είναι αναπαράσταση ενός γεγονότος, μιας εικόνας ή μιας ιδέας. Τα δεδομένα μπορεί να είναι ένας αριθμός, μια λέξη, μια εικόνα ή ακόμα και μια ηχογράφηση. Για παράδειγμα, ο αριθμός 7135553297 και τα ονόματα Zoe και Richardson είναι δεδομένα. Από μόνα τους, πιθανόν δε σημαίνουν κάτι για κανέναν. Οι *πληροφορίες* είναι δεδομένα τα οποία έχουν οργανωθεί, μετά από επεξεργασία, και παρουσιάζονται με μια μορφή που μπορεί να γίνει κατανοητή. Όταν ο υπολογιστής σας παρέχει μια εγγραφή επαφής σας που δείχνει ότι μπορείτε να επικοινωνήσετε με τη Zoe Richardson στον αριθμό (713)555-3297, τα *δεδομένα* γίνονται *χρήσιμα*, δηλαδή γίνονται *πληροφορίες*.

Οι υπολογιστές είναι άριστοι στην *επεξεργασία* (χειρισμός, υπολογισμός ή οργάνωση) των δεδομένων μας. Όταν πήγατε για πρώτη φορά στο πανεπιστήμιο, πιθανόν σας κατηύθυναν σε ένα μέρος όπου θα μπορούσατε να βγάλετε φοιτητική ταυτότητα. Φυσιολογικά θα δώσατε σε κάποιον υπάλληλο τα στοιχεία σας, τα οποία εισήχθησαν σε έναν υπολογιστή. Ο υπάλληλος τράβηξε μια φωτογραφία σας με μια ψηφιακή φωτογραφική μηχανή (συλλέγοντας έτσι περισσότερα δεδομένα) ή παραδώσατε μία που είχατε μαζί σας. Όλα τα δεδομένα υποβλήθηκαν σε κατάλληλη επεξεργασία, ώστε να μπορούν να εκτυπωθούν στην ταυτότητά σας. Αυτή η οργανωμένη έξοδος δεδομένων στην ταυτότητά σας είναι μια «χρήσιμη πληροφορία».

Αντίθετα, όμως, από τους ανθρώπους, οι υπολογιστές εργάζονται αποκλειστικά με αριθμούς (όχι λέξεις). Για να επεξεργαστούν δεδομένα, πρέπει να λειτουργούν σε μια γλώσσα που καταλαβαίνουν. Αυτή η γλώσσα, η *δυαδική γλώσσα*, αποτελείται από δύο μόνο ψηφία: 0 (καλείται και «False») και 1 (καλείται και «True»). Οτιδήποτε κάνει ένας υπολογιστής, όπως η επεξεργασία δεδομένων, η εκτύπωση μιας αναφοράς ή η επεξεργασία μιας φωτογραφίας, αναλύεται σε μια σειρά από 0 και 1. Κάθε 0 και 1 είναι ένα *δυαδικό ψηφίο* ή *bit* (Binary digIT), όπως ονομάζεται.

Οκτώ δυαδικά ψηφία (ή bit) συνδυάζονται για να δημιουργήσουν ένα *byte*. Στους υπολογιστές, κάθε γράμμα της αλφαβήτου, κάθε αριθμός και κάθε ειδικός χαρακτήρας (όπως το @) αποτελείται από έναν μοναδικό συνδυασμό οκτώ bit, ή μια ακολουθία από οκτώ 0 και 1. Έτσι, για παράδειγμα, στη δυαδική γλώσσα, το γράμμα *K* αναπαρίσταται ως 01001011. Αυτά είναι οκτώ bit ή ένα byte.

Πώς γίνονται οι υπολογισμοί όμως; Οι υπολογιστές καταλαβαίνουν μόνο δύο καταστάσεις: ανοιχτό και κλειστό (on και off). Μέσα σε έναν υπολογιστή, αυτές οι δύο

καταστάσεις ορίζονται με τη βοήθεια των αριθμών 1 και 0. Οι ηλεκτρικοί διακόπτες είναι τα στοιχεία μέσα στον υπολογιστή που εναλλάσσονται μεταξύ των δύο καταστάσεων 1 και 0, σηματοδοτώντας το ανοιχτό και το κλειστό. Ένα σύστημα υπολογιστή μπορεί μάλιστα να θεωρηθεί ως μια τεράστια συλλογή τέτοιων διακοπών on/off. Αυτοί οι διακόπτες on/off συνδυάζονται με διάφορους τρόπους για να εκτελούν πρόσθεση και αφαίρεση και να μετακινούν δεδομένα σε όλο το σύστημα του υπολογιστή.

Στην καθημερινότητά σας χρησιμοποιείτε διάφορες μορφές διακοπών. Ένας διακόπτης φωτός είναι είτε κλειστό, ώστε το ρεύμα να ρέει από τη λυχνία, ή ανοιχτός. Ένας άλλος διακόπτης είναι η μπαταρία του νερού. Όπως δείχνει η Εικόνα 2.2, το κλείσιμό της αναπαριστά την τιμή 0 και το άνοιγμά της την τιμή 1.

Τι είδους διακόπτες, όμως, είναι αυτοί που χρησιμοποιούνται στους υπολογιστές; Οι πρώτοι υπολογιστές χρησιμοποιούσαν τρανζίστορ. Τα τρανζίστορ είναι ηλεκτρονικοί διακόπτες που κατασκευάζονται από ένα ειδικό υλικό με ιδιότητες ημιαγωγού. Ημιαγωγός είναι οποιοδήποτε υλικό για το οποίο μπορούμε να έχουμε έλεγχο ως προς το αν θα άγει ηλεκτρισμό ή θα λειτουργεί ως μονωτικό υλικό (ώστε να εμποδίζει τη διέλευση ηλεκτρισμού). Το πυρίτιο (σιλικόνη), το οποίο απαντάται στην άμμο, είναι ο ημιαγωγός που χρησιμοποιείται για την κατασκευή των τρανζίστορ.

Το πυρίτιο μόνο του δεν μπορεί να άγει ηλεκτρισμό ιδιαίτερα καλά, αλλά αν αναμειχθεί με ελεγχόμενο τρόπο με κάποια συγκεκριμένα υλικά, συμπεριφέρεται, τελικά, ως ένας διακόπτης ανοιχτού/κλειστού. Το πυρίτιο επιτρέπει τη ροή του ηλεκτρικού ρεύματος όταν εφαρμόζεται συγκεκριμένη τάση, αλλιώς εμποδίζει αυτήν τη ροή.

Οι εξελίξεις στην τεχνολογία άρχισαν να απαιτούν περισσότερα τρανζίστορ απ' όσα μπορούν να χωρέσουν σε μια μητρική κάρτα. Έπρεπε να βρεθεί κάτι που θα χωρούσε περισσότερα τρανζίστορ σε μικρότερο χώρο. Ως εκ τούτου, αναπτύχθηκαν τα ολοκληρωμένα κυκλώματα ή «τσιπ». Αυτά είναι μικροσκοπικά κομμάτια ημιαγωγών που περιέχουν τεράστιες ποσότητες τρανζίστορ. Τα περισσότερα ολοκληρωμένα κυκλώματα δεν είναι μεγαλύτερα από το ένα τέταρτο της ίντσας, αλλά καταφέρνουν να περιέχουν δισεκατομμύρια τρανζίστορ. Αυτή η εξέλιξη δίνει στους σχεδιαστές υπολογιστών τη δυνατότητα να δημιουργούν μικρούς αλλά πανίσχυρους μικροεπεξεργαστές, οι οποίοι είναι τα τσιπ που περιέχουν την κεντρική μονάδα επεξεργασίας (CPU ή επεξεργαστής). Η CPU μπορεί να θεωρηθεί ως ο «εγκέφαλος» του υπολογιστή επειδή εκεί γίνεται η επεξεργασία των δεδομένων και η μετατροπή

τους σε πληροφορίες. Σήμερα, σε ένα τσιπ μεγέθους ίσου με το νύχι του μικρού σας δακτύλου μπορούν να χωρέσουν περισσότερα από 2 δισεκατομμύρια τρανζίστορ!

Τα bit και τα byte δε χρησιμοποιούνται μόνο ως η γλώσσα του υπολογιστή, αλλά και για να αναπαραστήσουν την ποσότητα των δεδομένων και των πληροφοριών που ο υπολογιστής δέχεται και εξάγει. Τα αρχεία επεξεργασίας κειμένου, οι ψηφιακές εικόνες, ακόμα και το λογισμικό αναπαρίστανται μέσα σε έναν υπολογιστή ως μια σειρά από bit και byte. Αυτά τα αρχεία και οι εφαρμογές μπορεί να είναι πολύ μεγάλα και να περιέχουν χιλιάδες ή εκατομμύρια byte.

Για να υπολογιστεί πιο εύκολα το μέγεθος τέτοιων αρχείων, απαιτούνται μονάδες μέτρησης μεγαλύτερες από 1 byte. Τα kilobyte, τα megabyte και τα gigabyte είναι απλώς μεγαλύτερες ποσότητες byte: 1 **kilobyte (KB)** είναι περίπου 1.000 byte, 1 **megabyte (MB)** είναι περίπου 1 εκατομμύριο byte και 1 **gigabyte (GB)** είναι περίπου 1 δισεκατομμύριο byte. Σήμερα, οι προσωπικοί υπολογιστές έχουν τη δυνατότητα να αποθηκεύουν **terabyte (TB)** δεδομένων (περίπου 1 τρισεκατομμύριο byte), ενώ πολλοί επαγγελματικοί υπολογιστές μπορούν να αποθηκεύσουν έως 1 **petabyte (PB)** (1.000 terabyte) δεδομένων. Η μηχανή αναζήτησης της Google επεξεργάζεται περισσότερα από ένα PB δεδομένων που παράγονται από τους χρήστες κάθε ώρα!

μονάδα	πλήθος μπάιτ	προσέγγιση
1 kilobyte (KB)	$2^{10}=1.024$	10^3
1 megabyte (MB)	$2^{20}=1.048.576$	10^6
1 gigabyte (GB)	$2^{30}=1.073.741.824$	10^9
1 terabyte (TB)	2^{40}	10^{12}
1 petabyte (PB)	2^{50}	10^{15}
1 exabyte (EB)	2^{60}	10^{18}

1 zettabyte (ZB), κατά προσέγγιση 10^{21} bytes! Το 2011 υπήρχαν αποθηκευμένα στον κόσμο περίπου 1,8 ZB ενώ σήμερα 13,7 ZB! Το 2021 ο αριθμός των αποθηκευμένων δεδομένων πιθανότατα θα ξεπεράσει τα 40ZB!

Πώς επεξεργάζεται, όμως, ο υπολογιστής τόσες πληροφορίες χρησιμοποιώντας την αναπαράστασή τους σε bits; Ο υπολογιστής χρησιμοποιεί *υλικό* και *λογισμικό* για να επεξεργαστεί και να μετατρέψει δεδομένα σε πληροφορίες, έτσι ώστε να έχετε τη

δυνατότητα να εκτελείτε εργασίες όπως η σύνταξη μιας επιστολής, ή απλώς να παίζετε. Το *υλικό* (hardware) είναι οποιοδήποτε μέρος του υπολογιστή μπορείτε να αγγίζετε. Ωστόσο, ένας υπολογιστής δεν μπορεί να λειτουργήσει μόνο με το υλικό, χρειάζεται και το λογισμικό. Το *λογισμικό* (software) είναι το σύνολο των προγραμμάτων που δίνουν στο υλικό τη δυνατότητα να εκτελεί διαφορετικές εργασίες.

Το *λογισμικό εφαρμογών* είναι το σύνολο των προγραμμάτων που χρησιμοποιείτε σε έναν υπολογιστή, ώστε να είναι δυνατή η εκτέλεση εργασιών όπως η σύνταξη μιας έρευνας. Το *λογισμικό συστημάτων* είναι το σύνολο των προγραμμάτων που επιτρέπουν τη συνεργασία ανάμεσα στις συσκευές υλικού και στο λογισμικό εφαρμογών του υπολογιστή σας. Ο πιο κοινός τύπος λογισμικού συστημάτων είναι το *λειτουργικό σύστημα* (Operating System, OS), το πρόγραμμα που ελέγχει πώς λειτουργεί το σύστημα του υπολογιστή σας. Το πιθανότερο είναι ότι ο υπολογιστής που έχετε ή χρησιμοποιείτε εκτελεί μια έκδοση των Windows, ή του macOS της Apple.

2.1 Η Κεντρική Μονάδα Επεξεργασίας (CPU)

Ποια είναι τα βασικά εξαρτήματα της CPU; Η CPU αποτελείται από δύο μονάδες: τη μονάδα ελέγχου (CU, Control Unit) και την αριθμητική λογική μονάδα (ALU, Αριθμητική λογική μονάδα). Η αριθμητική λογική μονάδα είναι υπεύθυνη για την εκτέλεση όλων των αριθμητικών πράξεων (πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση). Λαμβάνει επίσης αποφάσεις που βασίζονται στη λογική Boole και τη σύγκριση δεδομένων. Για παράδειγμα, μπορεί να συγκρίνει δύο αριθμητικά στοιχεία για να βρει ποιο είναι το μεγαλύτερο, ποιο είναι το μικρότερο ή αν είναι ίσα μεταξύ τους.

Η μονάδα ελέγχου της CPU διαχειρίζεται τους τα ηλεκτρονικά στοιχεία/διακόπτες (που είναι τα μικροσκοπικά τρανζίστορ) στο εσωτερικό της CPU. Προγραμματίζεται από τους σχεδιαστές της CPU ώστε να θυμάται την ακολουθία των σταδίων επεξεργασίας για τη συγκεκριμένη CPU και πώς πρέπει να ρυθμιστεί κάθε διακόπτης στο εσωτερικό της κεντρικής μονάδας επεξεργασίας (για παράδειγμα on ή off) ανάλογα με το στάδιο επεξεργασίας. Σε κάθε χτύπο του ρολογιού του συστήματος, η μονάδα ελέγχου θέτει τη λειτουργία on/off του διακόπτη στη σωστή ρύθμιση και κατόπιν εκτελεί την εργασία αυτού του σταδίου.

Για τη μετάβαση από το ένα στάδιο του κύκλου επεξεργασίας στο επόμενο, η μητρική κάρτα χρησιμοποιεί ένα ενσωματωμένο ρολόι συστήματος. Αυτό το

εσωτερικό ρολόι είναι στην πραγματικότητα ένας ειδικός κρύσταλλος που συμπεριφέρεται σαν ένας μετρονόμος, διατηρώντας ένα σταθερό ρυθμό και ελέγχοντας όταν τότε η CPU προχωρά στο επόμενο στάδιο επεξεργασίας.

Αυτοί οι σταθεροί χτύποι του ρολογιού του συστήματος, οι οποίοι ονομάζονται κύκλος ρολογιού, ορίζουν το ρυθμό με τον οποίο ένας υπολογιστής προχωρά από ένα στάδιο επεξεργασίας στο επόμενο. Ο ρυθμός, η ή ταχύτητα ρολογιού, μετριέται σε hertz (Hz) και περιγράφει πόσες φορές ανά δευτερόλεπτο συμβαίνει κάτι. Η μέτρηση των σύγχρονων ρολογιών συστήματος γίνεται σε Gigahertz (GHz), ή αλλιώς, ένα δισεκατομμύριο χτύποι ανά δευτερόλεπτο. Για παράδειγμα, σε ένα σύστημα με ρολόι 3 GHz, υπάρχουν τρία δισεκατομμύρια χτύποι ανά δευτερόλεπτο.

Σε τι διαφέρει, γενικά, η μια CPU από την άλλη; Μπορεί να καταβάλετε μεγαλύτερο ποσό για έναν υπολογιστή με επεξεργαστή Intel i7 από έναν με i5 εξαιτίας της αυξημένης επεξεργαστικής ισχύος του. Η ισχύς επεξεργασίας μιας CPU καθορίζεται, κυρίως, από τους εξής παράγοντες:

- Την ταχύτητα ρολογιού της.
- Αν έχει πολλαπλούς πυρήνες και πόσους.
- Την ποσότητα μνήμης cache (δείτε πιο κάτω) που διαθέτει.

Πώς μπορεί να με βοηθήσει μια CPU με υψηλότερη ταχύτητα ρολογιού; Όσο πιο υψηλή είναι η ταχύτητα ρολογιού τόσο πιο γρήγορα θα γίνει η επεξεργασία της επόμενης οδηγίας. Οι CPU έχουν ταχύτητες ρολογιών τη δεδομένη στιγμή έως 4 GHz. Κάποιοι χρήστες αναγκάζουν το υλικό τους να λειτουργεί πιο γρήγορα, εφαρμόζοντας υπερχρονισμό στον επεξεργαστή τους.

Ο υπερχρονισμός (overclocking) σημαίνει ότι λειτουργείτε τη CPU σε υψηλότερη ταχύτητα από τις συστάσεις του κατασκευαστή της. Μπορεί, συνεπώς, να παράγεται περισσότερη θερμότητα, η οποία συνεπάγεται μικρότερη διάρκεια ζωής για τη CPU, ενώ συνήθως επιφέρει την κατάργηση οποιασδήποτε εγγύησης. Πάντως, ειδικά σε συστήματα για παιχνίδια θα το δείτε να γίνεται αρκετά συχνά.

Πώς μπορεί να με βοηθήσει μια CPU πολλαπλών πυρήνων; Ένας πυρήνας σε μια CPU περιέχει τα μέρη της CPU που απαιτούνται για την επεξεργασία μιας οδηγίας. Με την τεχνολογία πολλαπλών πυρήνων, δύο ή περισσότεροι πλήρεις επεξεργαστές μπορούν να υπάρχουν στο ίδιο τσιπ, επιτρέποντας την ανεξάρτητη εκτέλεση δύο ή περισσότερων συνόλων οδηγιών ταυτόχρονα.

Εάν είχατε έναν κλώνο του εαυτού σας δίπλα σας στη δουλειά σας, θα μπορούσατε να παράγετε διπλάσιο έργο από όσο κάνετε τώρα, και αυτή ακριβώς είναι η ιδέα πίσω από την επεξεργασία πολλαπλών πυρήνων. Με την επεξεργασία πολλαπλών πυρήνων, οι εφαρμογές που εκτελούνται πάντα στο παρασκήνιο, όπως το λογισμικό προστασίας από ιούς και το λειτουργικό σύστημα (ΛΣ), μπορούν να έχουν τον δικό τους αποκλειστικό πυρήνα, απελευθερώνοντας τους υπόλοιπους για πιο αποτελεσματική εκτέλεση άλλων εφαρμογών. Το αποτέλεσμα είναι ταχύτερη επεξεργασία και ομαλότερη πολυεπεξεργασία. Τα τσιπ με δυνατότητες επεξεργασίας σε τέσσερις πυρήνες χρησιμοποιούν τέσσερις διαφορετικές παράλληλες διαδρομές επεξεργασίας, ώστε να είναι περίπου τόσο γρήγορα όσο θα ήταν τέσσερις διαφορετικές CPU εργαζόμενες παράλληλα. Για παράδειγμα, στην πιο κάτω εικόνα, βλέπουμε μια CPU με 4 πυρήνες (quad core) η οποία, χονδρικά, λειτουργεί ως να ήταν 4 διαφορετικοί επεξεργαστές που εργάζονται παράλληλα.



Παρατηρούμε ότι η μία αυτή CPU εκτελεί, παράλληλα και ταυτόχρονα, 4 διαφορετικά προγράμματα και, άρα, η συνολική ταχύτητα επεξεργασίας είναι τέσσερις φορές μεγαλύτερη από την ταχύτητα επεξεργασίας αν η CPU περιείχε μόνο έναν πυρήνα.

Βέβαια, συνολικά, η ταχύτητα δεν είναι ακριβώς τέσσερις φορές πιο μεγαλύτερη επειδή το σύστημα επιβαρύνεται με κάποιες επιπλέον διεργασίες σχετικά με την απόφαση για το ποιος επεξεργαστής θα εκτελεί κάθε κομμάτι του προβλήματος καθώς και για την ανασυγκρότηση των αποτελεσμάτων που παράγει κάθε CPU.

Υπάρχουν επίσης επεξεργαστές δέκα πυρήνων (όπως ο Intel i7 Extreme Broadwell-E), οι οποίοι εκτελούν δέκα ξεχωριστές διαδρομές επεξεργασίας. Τα συστήματα πολλαπλών επεξεργαστών χρησιμοποιούνται συχνά όταν πρέπει να επιλυθούν σημαντικά προβλήματα απαιτητικών πράξεων σε κλάδους όπως οι

προσομοιώσεις σε υπολογιστή, η παραγωγή βίντεο και η επεξεργασία γραφικών. Τέτοια συστήματα χρησιμοποιούν επίσης όσοι παίζουν ιδιαίτερα απαιτητικά παιχνίδια σε οικιακούς υπολογιστές.

Οι CPU πολλαπλών πυρήνων είναι ο μοναδικός τρόπος επίτευξης ταυτόχρονης επεξεργασίας; Ορισμένοι τύποι προβλημάτων ταιριάζουν ιδανικά στο περιβάλλον παράλληλης επεξεργασίας, αν και αυτή η προσέγγιση δεν εφαρμόζεται σε προσωπικούς υπολογιστές. Στην παράλληλη επεξεργασία, υπάρχει ένα μεγάλο δίκτυο υπολογιστών, με κάθε υπολογιστή να απασχολείται σ ένα τμήμα του ίδιου προβλήματος ταυτόχρονα. Για να ταιριάζει κάποιο πρόβλημα στην έννοια της παράλληλης επεξεργασίας, πρέπει να είναι δυνατός ο χωρισμός του προβλήματος σε μια σειρά από εργασίες οι οποίες να μπορούν να εκτελεστούν ταυτόχρονα. Για παράδειγμα, ένα πρόβλημα όπου εκατομμύρια πρόσωπα συγκρίνονται με μία εικόνα για λόγους αναγνώρισης, προσαρμόζεται εύκολα στην παράλληλη επεξεργασία. Το πρόσωπο της εικόνας μπορεί να συγκριθεί με πολλές εκατοντάδες πρόσωπα ταυτόχρονα. Αλλά αν το επόμενο βήμα ενός αλγόριθμου μπορεί να ξεκινήσει μόνο όταν θα έχει τα αποτελέσματα του προηγούμενου, η παράλληλη επεξεργασία δεν βοηθά σε κάτι.

2.2 Μνήμη cache

Ποιοι άλλοι παράγοντες, όμως, μπορούν να επηρεάζουν την ταχύτητα επεξεργασίας; Η μνήμη επιτάχυνσης λειτουργιών της CPU, η οποία είναι γνωστή ως μνήμη cache, κρυφή ή λανθάνουσα μνήμη. Η μνήμη cache είναι ένας τύπος RAM που τροφοδοτεί με δεδομένα τη CPU για επεξεργασία με πολύ υψηλότερη ταχύτητα σε σχέση με την ταχύτητα της κύριας RAM του υπολογιστή.

Η μνήμη cache έχει σχετικά (με την κύρια μνήμη) μικρό μέγεθος, αλλά είναι πολύ πιο γρήγορη (σε τυπικούς H/Y έως τρεις φορές γρηγορότερη). Ο τύπος αυτής της μνήμης στηρίζεται στην εμπειρική Αρχή της Τοπικότητας (Principle of Locality), σύμφωνα με την οποία η συντριπτική πλειοψηφία των προγραμμάτων δεν προσπελαύνει ομοιόμορφα όλες τις περιοχές της μνήμης, αλλά τείνει να χρησιμοποιεί τις ίδιες περιοχές για μεγάλα χρονικά διαστήματα. Ένα απλό παράδειγμα αποτελεί η ύπαρξη επαναλήψεων (loops), όπου κώδικας και δεδομένα επαναχρησιμοποιούνται πολλές φορές:


```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

Data references

- Reference array elements in succession (stride-1 reference pattern).
- Reference variable `sum` each iteration.

Spatial locality

Temporal locality

Instruction references

- Reference instructions in sequence.
- Cycle through loop repeatedly.

Spatial locality

Temporal locality

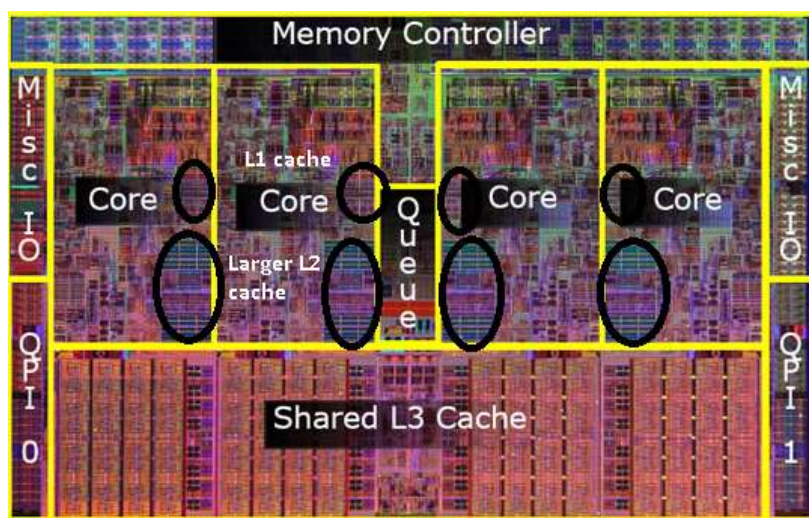
Έχοντας αντιγράψει, με βάση τη χρήση τους από το πρόγραμμα που τρέχει στην CPU, τα περιεχόμενα αυτών των περιοχών μνήμης, μέσα στην γρήγορη λανθάνουσα μνήμη, τις περισσότερες φορές η CPU δεν χρειάζεται να προσπελάσει την κύρια μνήμη, αλλά την λανθάνουσα. Η cache δεν είναι τμήμα της κύριας μνήμης. Ονομάζεται λανθάνουσα (ή και κρυφή) μνήμη, επειδή η CPU την προσπελαύνει έχοντας την εντύπωση ότι διαβάσει από την κύρια μνήμη και όχι από αυτήν.

Υπάρχουν τρία επίπεδα μνήμης cache, τα οποία ορίζονται με βάση το πόσο «κοντά» βρίσκονται στην CPU:

- Η cache επιπέδου 1 ή *L1 cache* είναι ένα κομμάτι γρήγορης μνήμης που ενσωματώνεται στο ίδιο το τσιπ της CPU για αποθήκευση δεδομένων ή εντολών που χρησιμοποιήθηκαν πρόσφατα. Χάρη στη θέση της δίπλα στις μονάδες επεξεργασίας της CPU, τα δεδομένα που βρίσκονται στην L1 cache μεταφέρονται στους καταχωρητές της CPU προς επεξεργασία απίστευτα γρήγορα.
- Η cache επιπέδου 2 ή *L2 cache* βρίσκεται στο τσιπ της CPU, αλλά ελαφρώς μακρύτερα από την cache επιπέδου 1, από την περιοχή στο τσιπ όπου βρίσκονται τα κυκλώματα επεξεργασίας δεδομένων, με συνέπεια η πρόσβαση σε αυτήν να χρειάζεται λίγο χρόνο παραπάνω σε σχέση με την L1 cache. Η L2 cache περιέχει περισσότερο αποθηκευτικό χώρο την L1 cache.

- Τέλος, η cache επιπέδου 3 ή *L3 cache* βρίσκεται και αυτή στο τσιπ της CPU, όμως προσπελαύνεται λίγο πιο αργά από τη CPU σε σχέση με την L2 cache αλλά είναι μεγαλύτερη σε μέγεθος από αυτήν.

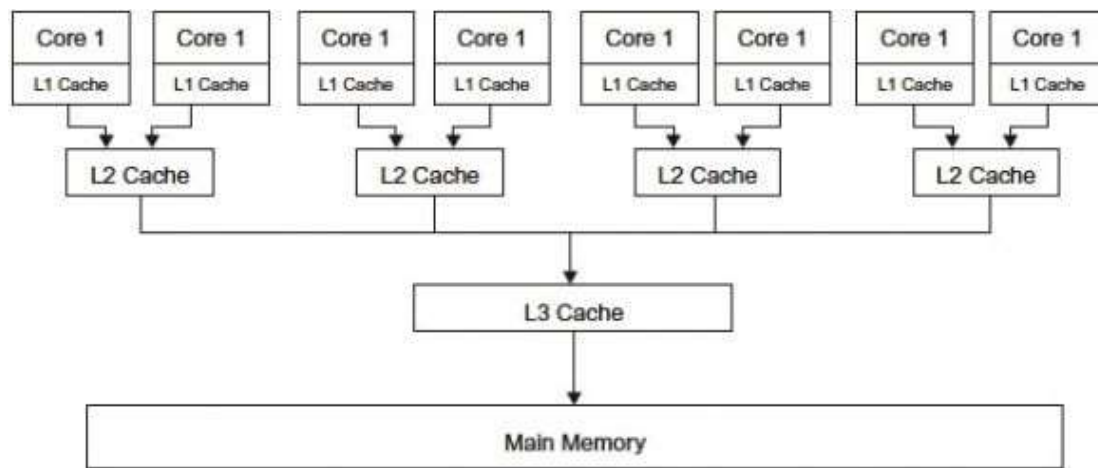
Στο πιο κάτω σχήμα φαίνεται η δομή ενός ολοκληρωμένου κυκλώματος CPU με 4 πυρήνες (cores) και τρία επίπεδα cache: L1, L2, και L3. Στις θέσεις των σχημάτων οβάλ είναι τοποθετημένες, από επάνω προς τα κάτω, οι μνήμες cache επιπέδου 1 και 2 (L1 και L2 αντίστοιχα) κάθε πυρήνα ενώ στο κάτω μέρος φαίνεται η μνήμη cache επιπέδου 3 (L3).



Μπορείτε να δείτε τα σχετικά τους μεγέθη και να συγκρίνετε τη χωρητικότητα και την ταχύτητά τους:

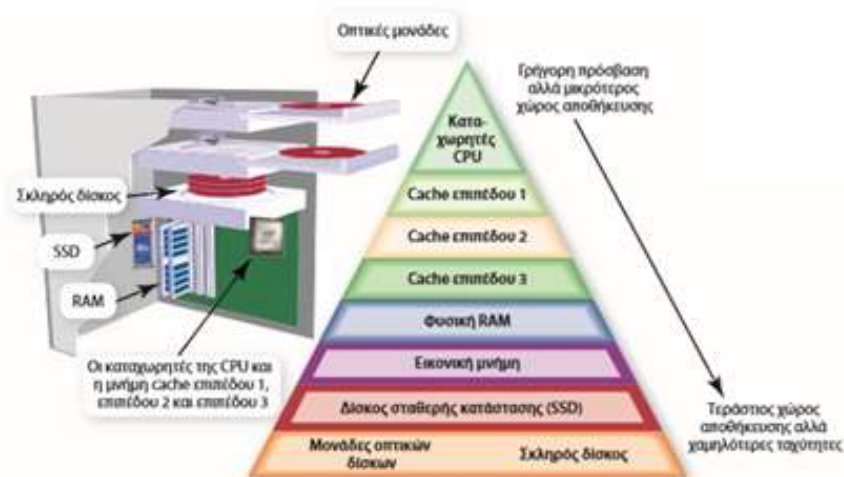
- L1 cache: μικρότερο μέγεθος, μεγαλύτερη ταχύτητα
- L2 cache: μεσαίο μέγεθος, μεσαία ταχύτητα
- L3 cache: μεγαλύτερο μέγεθος, μικρότερη ταχύτητα
- RAM H/Y: πολύ μεγαλύτερο μέγεθος, πολύ μικρότερη ταχύτητα, και πολύ μικρότερο κόστος ανά bit (από όλα τα τρία επίπεδα cache L1, L2, και L3).

Στο πιο κάτω σχήμα απεικονίζεται η ιεραρχία των τριών επιπέδων cache (Core: πυρήνας της CPU) καθώς και της κύριας μνήμης (RAM) του H/Y:



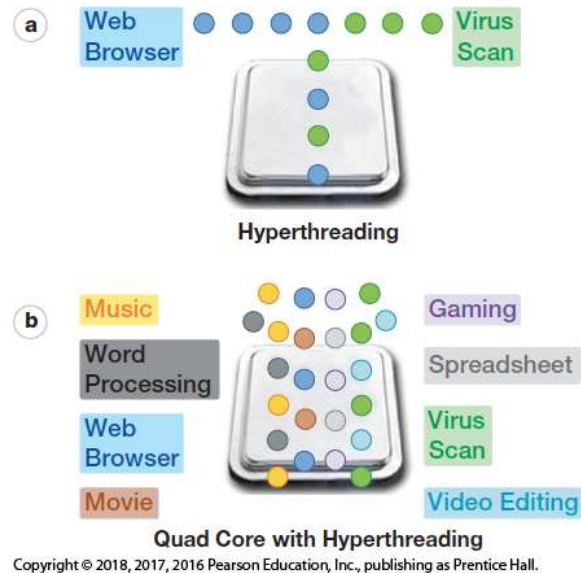
Γενικά, όσο πιο ακριβή είναι η CPU τόσο περισσότερη μνήμη cache αναμένεται να διαθέτει. Ως τελικοί χρήστες προγραμμάτων υπολογιστών, δεν κάνετε κάτι ιδιαίτερο για να χρησιμοποιήσετε την κρυφή μνήμη, δηλαδή την cache.

Δυστυχώς, επειδή η μνήμη cache ενσωματώνεται στο ολοκληρωμένο κύκλωμα της CPU, δεν μπορείτε να την αναβαθμίσετε καθώς αποτελεί μέρος του αρχικού σχεδιασμού της CPU. Επομένως, όπως με τη RAM, όταν αγοράζετε υπολογιστή, είναι σημαντικό να λάβετε υπόψη σας ποια CPU προσφέρει τη μεγαλύτερη μνήμη cache, όταν όλα τα υπόλοιπα στοιχεία (π.χ. ταχύτητα ρολογιού) είναι ισοδύναμα.



Οι CPU ξεκίνησαν να εκτελούν περισσότερες από μία εντολές όταν παρουσιάστηκε η υπερνημάτωση (hyper-threading) το 2002. Η υπερνημάτωση παρέχει ταχύτερη επεξεργασία πληροφοριών, δίνοντας τη δυνατότητα να αρχίσει η εκτέλεση ενός νέου συνόλου οδηγιών πριν ολοκληρωθεί η εκτέλεση του προηγούμενου συνόλου. Όπως

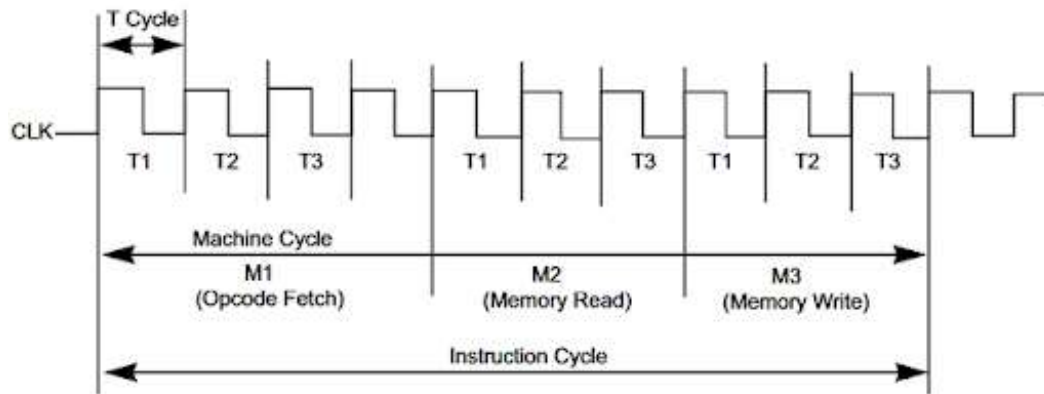
βλέπετε στην εικόνα που ακολουθεί, η υπερνημάτωση επιτρέπει την ταυτόχρονη επεξεργασία δύο διαφορετικών προγραμμάτων, τα οποία όμως μοιράζονται τους πόρους του ίδιου κυκλώματος, της CPU.



Όλοι οι επεξεργαστές Core της Intel, για παράδειγμα, διαθέτουν πολλαπλούς πυρήνες και υπερνημάτωση. Ο Intel i7-6950X έχει δέκα πυρήνες και όλοι χρησιμοποιούν υπερνημάτωση, που σημαίνει ότι ένας επεξεργαστής προσομοιώνει έναν αριθμό 20 επεξεργαστών.

2.3 Ο κύκλος μηχανής και ο κύκλος εντολής μίας CPU

Ο *κύκλος μηχανής* είναι τα τέσσερα στάδια που εκτελούνται από τη CPU για την επεξεργασία και αποθήκευση δεδομένων. Η γενική εικόνα κάθε κύκλου εντολής είναι η εξής:

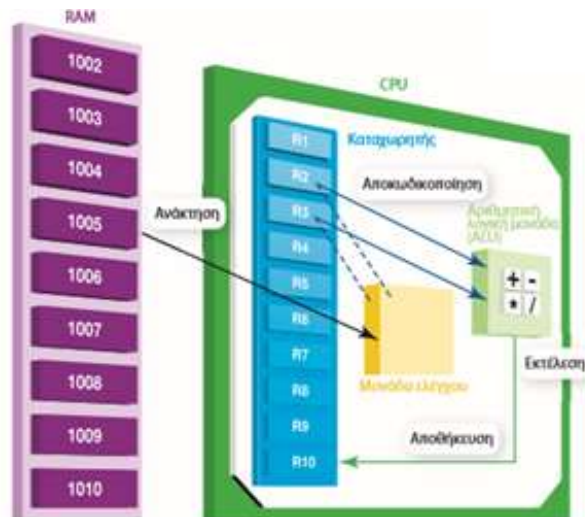


Basic CPU Timing Example

Σε κάθε μία περίοδο του ρολογιού, δηλαδή ένα T-cycle (κύκλος μηχανής) γίνονται μεταβολές στην κατάσταση της CPU. Οι μεταβολές αυτές (υπό τον έλεγχο της μονάδας ελέγχου της CPU) οδηγούν στην ανάκτηση της εντολής από τη μνήμη και την αποκωδικοποίησή της, δηλαδή την εύρεση του τύπου της εντολής και του τρόπου εκτέλεσής της από την CPU (φάση M1 στο πιο πάνω σχήμα), την ανάγνωση τυχόν δεδομένων που χρειάζεται η εντολή από τη μνήμη (φάση M2), και την αποθήκευση τυχόν αποτελεσμάτων από την εκτέλεση της εντολής στην μνήμη (φάση M3).

Σ' αυτήν την ενότητα θα αναλύσουμε τα στάδια που συμπεριλαμβάνει ένας κύκλος εντολής. Πιο αναλυτικά, θα δούμε τα εξής βήματα:

- Ανάκτηση των απαιτούμενων δεδομένων ή οδηγιών από τη RAM.
- Αποκωδικοποίηση των εντολών σε ακολουθίες που ο υπολογιστής μπορεί να καταλάβει.
- Εκτέλεση της εντολής.
- Αποθήκευση του αποτελέσματος στη RAM πριν την ανάκτηση της επόμενης εντολής.



Στάδιο 1: Το στάδιο ανάκτησης

Οι εντολές και τα δεδομένα των προγραμμάτων που «τρέχει» η CPU αποθηκεύονται σε διαφορετικές περιοχές του συστήματος του υπολογιστή. Μεταφέρονται μεταξύ των περιοχών αυτών ανάλογα με την ανάγκη της CPU για επεξεργασία. Τα προγράμματα αποθηκεύονται μόνιμα στον σκληρό δίσκο, καθώς ο δίσκος προσφέρει σταθερότητα στην αποθήκευση. Παρ' όλα αυτά, όταν ξεκινάτε ένα πρόγραμμα, το ίδιο το πρόγραμμα, ή κάποιες φορές μόνο τα απαραίτητα κομμάτια του, μεταφέρονται από το σκληρό δίσκο στη RAM.

Το πρόγραμμα μεταφέρεται στη RAM επειδή η CPU μπορεί να προσπελάει τα δεδομένα και τις οδηγίες εντολές των προγραμμάτων που αποθηκεύονται στη RAM ένα εκατομμύριο φορές πιο γρήγορα από το να βρίσκονταν στο σκληρό δίσκο. Αυτό συμβαίνει εν μέρει επειδή η RAM βρίσκεται πιο κοντά στη CPU από το σκληρό δίσκο. Ένας άλλος λόγος για την καθυστέρηση της διαβίβασης των δεδομένων και των οδηγιών εντολών του προγράμματος από το σκληρό δίσκο στη CPU είναι η σχετικά χαμηλή ταχύτητα των μηχανικών σκληρών δίσκων. Οι κεφαλές ανάγνωσης/εγγραφής πρέπει να σαρώνουν τους περιστρεφόμενους δίσκους, κάτι που απαιτεί χρόνο. Ακόμα και οι μη μηχανικοί σκληροί δίσκοι SSD έχουν χαμηλότερες ταχύτητες πρόσβασης προσπέλασης από τη RAM. Η RAM είναι ένας τύπος μνήμης που δίνει πολύ γρήγορα άμεση πρόσβαση στα δεδομένα.

Δεδομένου ότι απαιτούνται συγκεκριμένες εντολές από το πρόγραμμα, αυτές μεταφέρονται από τη RAM σε καταχωρητές (ειδικές θέσεις μνήμης που βρίσκονται μέσα στην ίδια τη CPU), όπου περιμένουν για να εκτελεστούν.

Ο χώρος αποθήκευσης της CPU δεν είναι αρκετά μεγάλος για να κρατά οτιδήποτε πρέπει να υποβληθεί σε επεξεργασία ταυτόχρονα με κάτι άλλο. Εάν υπήρχε αρκετή μνήμη στο ίδιο το τσιπ της CPU, ένα ολόκληρο πρόγραμμα θα μπορούσε να αντιγραφεί στη CPU από τη RAM πριν εκτελεστεί. Αυτό θα αύξανε την ταχύτητα και την απόδοση του υπολογιστή, επειδή δεν θα υπήρχε καθυστέρηση όσο η CPU θα σταματούσε την επεξεργασία προκειμένου να ανακτήσει τις οδηγίες εντολές από τη μνήμη RAM. Ωστόσο, η συγκέντρωση τόσο μεγάλης ποσότητας μνήμης σε ένα τσιπ CPU θα μεγέθυνε εκτίνασε σημαντικά την τιμή του. Επιπλέον, ο σχεδιασμός της CPU είναι τόσο πολύπλοκος, ώστε μόνο μια περιορισμένη ποσότητα χώρου αποθήκευσης διατίθεται αναλογεί ως χώρος αποθήκευσης για την ίδια τη CPU.

Η CPU, όμως, στην πραγματικότητα δεν χρειάζεται να ανακτά κάθε εντολή από τη μνήμη RAM κάθε φορά που περνά επαναλαμβάνειμέσα από έναν ακόμη κύκλο εντολής. Ένα άλλο επίπεδο μέσο αποθήκευσης, η κρυφή μνήμη (cache), έχει προσφέρει ακόμη ταχύτερη πρόσβαση στη CPU από ότι η RAM. Η κρυφή μνήμη αποτελείται από μικρά μπλοκ μνήμης που βρίσκονται ακριβώς πάνω και δίπλα στο τσιπ της CPU. Αυτά τα μπλοκ μνήμης κατέχουν περιέχουν θέσεις για πρόσφατες εντολές ή για εντολές και δεδομένα που χρησιμοποιούνται συχνά, τα οποία η CPU χρειάζεται περισσότερο. Όταν αυτές οι οδηγίες εντολές ή τα δεδομένα αποθηκεύονται στην κρυφή μνήμη, η CPU μπορεί να τα ανακτά πιο γρήγορα από ό,τι αν θα έπρεπε να τα προσπελάσει στη RAM.

Στάδιο 2: Το στάδιο της αποκωδικοποίησης

Το στάδιο της αποκωδικοποίησης διακρίνεται συνίσταται στις προσπάθειες της μονάδας ελέγχου της CPU να μεταφράσει (αποκωδικοποιήσει) τις οδηγίες/εντολές του προγράμματος σε εντολές που μπορεί να καταλάβει η CPU. Η συλλογή των εντολών που μπορεί να εκτελέσει μια συγκεκριμένη CPU ονομάζεται σύνολο οδηγιών εντολών για το συγκεκριμένο σύστημα. Κάθε CPU έχει το δικό της μοναδικό σεντ εντολών.η δική της μοναδική σειρά οδηγιών. Η μονάδα ελέγχου ερμηνεύει τον κώδικα κάθε εντολής σύμφωνα με τις οδηγίες που οι σχεδιαστές της CPU όρισαν για τη συγκεκριμένη CPU. Η μονάδα ελέγχου γνωρίζει πώς να ρυθμίσει όλα τα ηλεκτρονικά στοιχεία της CPU, ώστε το πρόγραμμα να εκτελεστεί σωστά.

Επειδή αυτοί που γράφουν τις αρχικές οδηγίες εντολές είναι άνθρωποι, όλες οι εντολές γράφονται σε με χρήση ενός συνόλου εντολών μιας γλώσσας που ονομάζεται

συμβολική γλώσσα συμβολομετάφρασης(assembly), με την οποία άνθρωποι μπορούν να εργάζονται πιο εύκολα από' ότι με το δυαδικό σύστημα, το οποίο «καταλαβαίνει», όμως, απ' ευθείας η CPU. Πολλοί επεξεργαστές έχουν παρόμοιες συμβολικές εντολές συμβολομετάφρασης στα σύνολα οδηγιών εντολών τους. Οι CPU διαφέρουν στην επιλογή των συμπληρωματικών εντολών που επέλεξε περιλαμβάνει η συμβολική γλώσσα συμβολομετάφρασης για το σύνολο οδηγιών εντολών τους. Κάθε ομάδα σχεδιασμού CPU επιχειρεί να αναπτύξει ένα σύνολο οδηγιών εντολών που να είναι ισχυρό, αλλά και γρήγορο.

Οι CPU διαφέρουν στην επιλογή των συμπληρωματικών εντολών που επέλεξε η γλώσσα συμβολομετάφρασης για το σύνολο οδηγιών. Κάθε ομάδα σχεδιασμού CPU επιχειρεί να αναπτύξει ένα σύνολο οδηγιών που είναι ισχυρό, αλλά και γρήγορο.

Ωστόσο, επειδή η CPU γνωρίζει και αναγνωρίζει «καταλαβαίνει» μόνο πρότυπα ακολουθίες από 0 και 1, δεν μπορεί να καταλάβει τη γλώσσα συμβολομετάφρασης άμεσα και γι' αυτό αυτές οι συμβολικές εντολές οδηγίες που μπορούμε να διαβάζουμε κατανοούμε εμείς μεταφράζονται σε δυαδικό κώδικα. Η μονάδα ελέγχου χρησιμοποιεί αυτές τις μεγάλες ακολουθίες από 0 και 1 δυαδικού κώδικα, τη γλώσσα μηχανής δηλαδή, ώστε να ρυθμίσει συντονίσει το υλικό στη της CPU για τις υπόλοιπες εργασίες που πρέπει να εκτελέσει. Η γλώσσα μηχανής είναι ένας δυαδικός κώδικας αναπαράστασης εντολών που μπορεί να εκτελέσει μία CPU όπως και ο κώδικας ASCII είναι ένας δυαδικός κώδικας αναπαράστασης γραμμάτων και χαρακτήρων. Όπως κάθε γράμμα ή χαρακτήρας έχει το δικό του μοναδικό συνδυασμό από 0 και 1, έτσι και η CPU έχει ένα σύνολο κωδικών που αποτελούνται από συνδυασμούς 0 και 1 για καθεμία από τις εντολές που αναγνωρίζει και εκτελεί από την κατασκευή της. Αν η CPU διαπιστώσει ότι έχει μπροστά της μία συγκεκριμένη ακολουθία από bits, γνωρίζει τι πρέπει να κάνει. Ο πιο κάτω πίνακας παρουσιάζει μερικές εντολές στη συμβολική γλώσσα Assembly και στη γλώσσα μηχανής.

Αναπαραστάσεις μερικών εντολών μιας CPU

Η κατανοητή για τους ανθρώπους ονομασία για μια εντολή	Η εντολή προς τη CPU στη γλώσσα Assembly (η συμβολική γλώσσα που χρησιμοποιούν οι προγραμματιστές)	Η εντολή προς τη CPU στη γλώσσα μηχανής (η γλώσσα που αναπαριστά το σύνολο εντολών της CPU)	
Πρόσθεσε	ADD	1110	1010
Αφαίρεσε	SUB	0001	0101
Πολλαπλασίασε	MUL	1111	0000

Διαίρεσε	DIV	0000	1111
----------	-----	------	------

Ο αριθμός εντολών, με τις διάφορες παραλλαγές τους, που μπορεί να εκτελέσει μία σύγχρονη CPU μπορεί να ξεπερνά τις 400 και δεν περιορίζεται στην εκτέλεση αριθμητικών πράξεων αλλά περιέχει, για παράδειγμα, εντολές που διαβάζουν δεδομένα από τη κύρια μνήμη (RAM) του υπολογιστή αλλά και εντολές που αποθηκεύουν δεδομένα στην μνήμη αυτή.

Στάδιο 3: Το στάδιο της εκτέλεσης

Η αριθμητική λογική μονάδα (ALU) είναι το τμήμα της μονάδας επεξεργασίας που σχεδιάστηκε ώστε να εκτελεί αριθμητικές πράξεις, όπως πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση και να συγκρίνει τις τιμές με τελεστές όπως «μεγαλύτερο από», «μικρότερο από» και «ίσο με». Για παράδειγμα, κατά τον υπολογισμό ενός μέσου όρου δοσμένων αριθμών, στην ALU γίνονται, επαναληπτικά, *προσθέσεις* για να υπολογιστεί το άθροισμα των αριθμών αυτών και στο τέλος μία *διαίρεση* με το πλήθος τους.

Η ALU εκτελεί επίσης και *λογικές* (Boolean) πράξεις OR, AND και NOT. Για παράδειγμα, για να καθοριστεί αν ένας φοιτητής μπορεί να αποφοιτήσει, η ALU θα πρέπει να εξακριβώσει αν ο φοιτητής δήλωσε όλα τα υποχρεωτικά μαθήματα *και* (AND) τα πέρασε όλα. Η ALU σχεδιάζεται ειδικά για την εκτέλεση, σε επίπεδο υλικού, τέτοιων πράξεων χωρίς σφάλματα και με απίστευτη ταχύτητα.

Η ALU παίρνει δεδομένα που έχουν μεταφερθεί από την κύρια μνήμη (RAM) στους καταχωρητές της CPU. Η ποσότητα των δεδομένων που η CPU μπορεί να επεξεργαστεί κάθε φορά βασίζεται εν μέρει στην ποσότητα των δεδομένων που μπορεί να αποθηκεύσει κάθε καταχωρητής. Ο αριθμός των bit που μπορεί να επεξεργαστεί ένας υπολογιστής κάθε φορά αναφέρεται ως *μέγεθος λέξης*. Ως εκ τούτου, ένας επεξεργαστής των 64 bit μπορεί να επεξεργαστεί περισσότερες πληροφορίες και πιο γρήγορα από έναν επεξεργαστή των 32 bit.

Στάδιο 4: Το στάδιο της αποθήκευσης

Στο τελικό στάδιο, το αποτέλεσμα που προκύπτει από την ALU αποθηκεύεται πάλι στα καταχωρητές. Η ίδια η εντολή καθορίζει, συνήθως, ποιοι καταχωρητές θα πρέπει να χρησιμοποιηθούν για την αποθήκευση του αποτελέσματός της (π.χ. του αθροίσματος, αν η εντολή είναι εντολή πρόσθεσης).

Μόλις ολοκληρωθεί η εκτέλεση της τρέχουσας εντολής, ξεκινά η ανάκτηση, από την RAM, η επόμενη εντολή και, έτσι, ξεκινά, πάλι, η ακολουθία ανάκτησης-αποκωδικοποίησης-εκτέλεσης-αποθήκευσης του κύκλου εντολής της CPU.

2.4 Η κύρια μνήμη

Θα μελετήσουμε τώρα το υποσύστημα μνήμης του υπολογιστή σας. Αυτό το υποσύστημα μπορεί να έχει τεράστια επίδραση στην ταχύτητα επεξεργασίας του συστήματός σας αν ταιριάζει καλά με την ισχύ της CPU σας. Πιο κάτω, θα συζητήσουμε το υποσύστημα μνήμης του υπολογιστή. Θα ξεκινήσουμε με την *κύρια μνήμη* (main memory) του υπολογιστή, την RAM. Τι είναι RAM; Η μνήμη τυχαίας προσπέλασης (RAM, Random Access Memory) είναι ο προσωρινός χώρος αποθήκευσης του υπολογιστή σας. Αν και αναφερόμαστε στη RAM ως μια μορφή χώρου αποθήκευσης, στην πραγματικότητα είναι η βραχυπρόθεσμη μνήμη του υπολογιστή. Θυμάται οτιδήποτε χρειάζεται ο υπολογιστής προκειμένου να επεξεργάζεται δεδομένα σε πληροφορίες, όπως δεδομένα τα οποία έχουν εισαχθεί, καθώς και οι εντολές επεξεργασίας τους (πρόγραμμα), αλλά μόνο όσο είναι ενεργοποιημένος. Η RAM είναι ένα παράδειγμα ασταθούς αποθήκευσης. Όταν ο υπολογιστής σβήσει, τα δεδομένα που έχουν αποθηκευτεί σε αυτή διαγράφονται. Γι' αυτό, εκτός από τη RAM, τα συστήματα πάντα περιλαμβάνουν συσκευές σταθερής αποθήκευσης για μόνιμη αποθήκευση οδηγιών και δεδομένων. Η μνήμη μόνο ανάγνωσης (ROM, Read-Only Memory), για παράδειγμα, διατηρεί τις κρίσιμες οδηγίες εκκίνησης. Οι σκληροί δίσκοι παρέχουν τη μεγαλύτερη χωρητικότητα σταθερής αποθήκευσης στο σύστημα του υπολογιστή. Σε φορητές συσκευές που δεν έχουν σκληρούς δίσκους, τα τσιπ μνήμης αποτελούν τη σταθερή αποθήκευσή τους.

Γιατί να μη χρησιμοποιώ έναν σκληρό δίσκο για την αποθήκευση δεδομένων και οδηγιών; Η ταχύτητα ανάκτησης δεδομένων από τη RAM είναι περίπου ένα εκατομμύριο φορές πιο γρήγορη για τη CPU από την ταχύτητα ανάκτησης από έναν μηχανικό σκληρό δίσκο. Ο χρόνος που χρειάζεται η CPU για να πάρει δεδομένα από τη RAM μετριέται σε δισεκατομμυριοστά του δευτερολέπτου, ενώ για την εξαγωγή δεδομένων από έναν γρήγορο μηχανικό σκληρό δίσκο χρειάζονται κατά μέσο όρο 10 χιλιοστά του δευτερολέπτου (ms). Ακόμα κι αν χρησιμοποιείτε δίσκο σταθερής κατάστασης SSD, ο οποίος είναι περίπου οκτώ φορές ταχύτερος από ένα συμβατικό σκληρό δίσκο, η ταχύτητα ανάκτησης δεδομένων από τη RAM είναι ασύγκριτη.

2.5 Συστήματα Αρίθμησης

Ένα Αριθμητικό Σύστημα αποτελείται από ένα σύνολο ψηφίων, ένα σύνολο συμβόλων (πράξεων) και τους κανόνες εκτέλεσης των πράξεων ανάμεσα στους αριθμούς με βάση τα ψηφία αυτά. Βάση (base) ενός Αριθμητικού Συστήματος είναι ένας ακέραιος αριθμός b , ο οποίος χαρακτηρίζει το σύστημα και ο οποίος είναι ίσος με το πλήθος των διαφορετικών ψηφίων του συστήματος. Τα πιο συχνά χρησιμοποιούμενα συστήματα είναι το δεκαδικό (με βάση το 10), το οποίο χρησιμοποιούμε στην καθημερινή ζωή, το δυαδικό (με βάση το 2), το οκταδικό (με βάση το 8) και το δεκαεξαδικό (με βάση το 16).

Η γνώση του δυαδικού συστήματος είναι ιδιαίτερα χρήσιμη στην κατανόηση των αρχών λειτουργίας των υπολογιστικών συστημάτων, διότι η απεικόνιση της πληροφορίας και οι πράξεις στους υπολογιστές μπορούν να αναπαρασταθούν με το δυαδικό σύστημα αρίθμησης, καθώς οι υπολογιστές χρησιμοποιούν την ψηφιακή λογική. Το δεκαεξαδικό σύστημα, από την άλλη μεριά, έχει το πλεονέκτημα ότι υπάρχει ένας εύκολος τρόπος μετατροπής των αριθμών από το δυαδικό στο δεκαεξαδικό σύστημα και αντίστροφα, ενώ το πλήθος των ψηφίων ενός αριθμού στο δεκαεξαδικό σύστημα είναι πολύ μικρότερο από το πλήθος των ψηφίων του ίδιου αριθμού στο δυαδικό σύστημα. Έτσι, συχνά στους υπολογιστές αντί να αναφέρουμε τη δυαδική αναπαράσταση ενός αριθμού χρησιμοποιούμε για πρακτικούς λόγους τη δεκαεξαδική αναπαράσταση. Στον επόμενο πίνακα φαίνονται τα ψηφία τα οποία χρησιμοποιούνται σε κάθε ένα από τα συστήματα αυτά.

Δυαδικό σύστημα	Οκταδικό σύστημα	Δεκαδικό σύστημα	Δεκαεξαδικό σύστημα
--------------------	---------------------	---------------------	------------------------

0	0	0	0
1	1	1	1
	2	2	2
	3	3	3
	4	4	4
	5	5	5
	6	6	6
	7	7	7
		8	8
		9	9
			A
			B
			C
			D
			E
			F

Τα ψηφία A, B, C, D, E, F χρησιμοποιούνται στο δεκαεξαδικό σύστημα για να εκφράσουν τους αριθμούς 10, 11, 12, 13, 14, 15, για τους οποίους δεν υπάρχουν αντίστοιχα ψηφία στο δεκαδικό σύστημα.

Η γενική μορφή παράστασης ενός αριθμού σε ένα Αριθμητικό σύστημα είναι:

$$a_{m-1}a_{m-2}\dots a_1a_0, a_{-1}a_{-2}\dots a_{-n}$$

όπου οι αριθμοί m και n αναφέρονται σε θέσεις ψηφίων. Τα ψηφία $a_{m-1}, a_{m-2}, \dots, a_1$ και a_0 είναι το ακέραιο μέρος του αριθμού, ενώ τα $a_{-1}, a_{-2}, \dots, a_{-n}$ είναι το κλασματικό του μέρος.

Όταν η βάση του Αριθμητικού Συστήματος είναι b, τότε η αξία του αριθμού (δηλαδή, η αντίστοιχη τιμή στο Δεκαδικό Σύστημα Αρίθμησης) είναι:

$$N = a_{m-1} b^{m-1} + a_{m-2} b^{m-2} + \dots a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots a_{-n} b^{-n}$$

Ένας αριθμός X μπορεί να εκφραστεί σε οποιοδήποτε Αριθμητικό Σύστημα με βάση β. Για αυτή την έκφραση, χρησιμοποιούμε το συμβολισμό $(X)_\beta$. Είναι δυνατό να επιβεβαιώσει κανείς ότι ο ίδιος αριθμός εκφράζεται με διαφορετικές ακολουθίες ψηφίων σε άλλα Συστήματα Αρίθμησης, όπως φαίνεται στα πιο κάτω παραδείγματα που αφορούν στον αριθμό 28 του Δεκαδικού Συστήματος.

$$(28)_{10} = 2 \times 10^1 + 8 \times 10^0$$

$$(34)_8 = 3 \times 8^1 + 4 \times 8^0 = (28)_{10}$$

$$(1C)_{16} = 1 \times 16^1 + 12 \times 16^0 = (28)_{10}$$

$$(11100)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = (28)_{10}$$

Αντίστροφα, η ίδια ακολουθία ψηφίων μπορεί να συμβολίζει διαφορετικούς αριθμούς σε διαφορετικά συστήματα, για παράδειγμα,

$$(11)_{16} = 1 \times 16^1 + 1 \times 16^0 = 16 + 1 = (17)_{10}$$

$$(11)_8 = 1 \times 8^1 + 1 \times 8^0 = (9)_{10}$$

$$(11)_2 = 1 \times 2^1 + 1 \times 2^0 = 2 + 1 = (3)_{10}$$

Στις ενότητες που ακολουθούν θα αναφερθούμε στις διαδικασίες μετατροπής ενός αριθμού από ένα Σύστημα Αρίθμησης σε κάποιο άλλο.

2.5.1 Μετατροπή από οποιοδήποτε Σύστημα Αρίθμησης στο Δεκαδικό Σύστημα

Για να μετατρέψουμε έναν αριθμό από οποιοδήποτε σύστημα στο δεκαδικό, υπολογίζουμε την τιμή της παράστασης (η υποδιαστολή βρίσκεται μεταξύ των ψηφίων a_0 και a_{-1}).

$$a_{m-1} b^{m-1} + a_{m-2} b^{m-2} + \dots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots + a_{-n} b^{-n}$$

όπου με b συμβολίζουμε τη βάση του συστήματος.

Για παράδειγμα, για να μετατρέψουμε τον αριθμό $(11001)_2$ στο δεκαδικό σύστημα, υπολογίζουμε την τιμή της παράστασης:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 16 + 8 + 1 = (25)_{10}$$

Για να μετατρέψουμε τον αριθμό $(11,01)_2$ στο δεκαδικό σύστημα, τότε θα έχουμε:

$$1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 3 + 0,25 = (3,25)_{10}$$

Για να μετατρέψουμε τον αριθμό $(45,34)_8$ από το οκταδικό σύστημα προς το δεκαδικό, υπολογίζουμε την τιμή της παράστασης:

$$4 \times 8^1 + 5 \times 8^0 + 3 \times 8^{-1} + 4 \times 8^{-2} = 32 + 5 + 0,375 + 0,0625 = (37,4375)_{10}$$

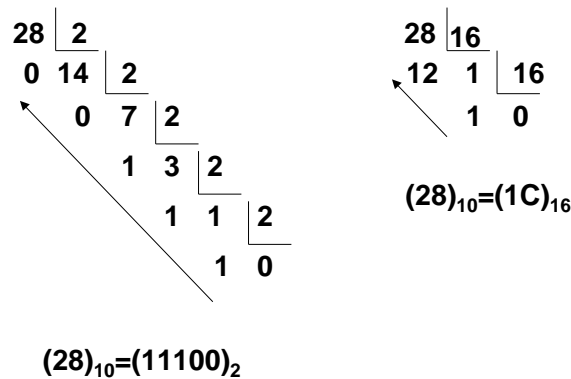
2.5.2 Μετατροπή από το Δεκαδικό σε οποιοδήποτε άλλο Σύστημα Αρίθμησης

Η μετατροπή αυτή γίνεται σε δυο φάσεις. Στην πρώτη φάση μετατρέπεται το ακέραιο μέρος του αριθμού, ενώ στη δεύτερη μετατρέπεται το κλασματικό μέρος.

Μετατροπή ακεραίου μέρους

Για να μετατρέψουμε το ακέραιο μέρος του αριθμού, το διαιρούμε με τη βάση του συστήματος, b , και παίρνουμε ένα υπόλοιπο (Y) και ένα πηλίκο (Π). Το πηλίκο διαιρείται και πάλι με το b και παίρνουμε ένα νέο πηλίκο Π και υπόλοιπο Y . Η διαδικασία αυτή επαναλαμβάνεται μέχρι το πηλίκο Π να γίνει 0. Η ζητούμενη αναπαράσταση είναι τα υπόλοιπα (Y), με την αντίστροφη σειρά από εκείνη που τα βρήκαμε.

Για παράδειγμα, στο Σχήμα 1 φαίνεται η διαδικασία μετατροπής του αριθμού 28 στο δυαδικό και το δεκαεξαδικό σύστημα αντίστοιχα. Οι αναπαραστάσεις του δεκαδικού αριθμού 28 στα δύο συστήματα είναι $(11100)_2$ και $(1C)_{16}$. Στη μετατροπή στο δεκαεξαδικό σύστημα, το 12, μετατρέπεται στο ψηφίο C στο οποίο και αντιστοιχεί.

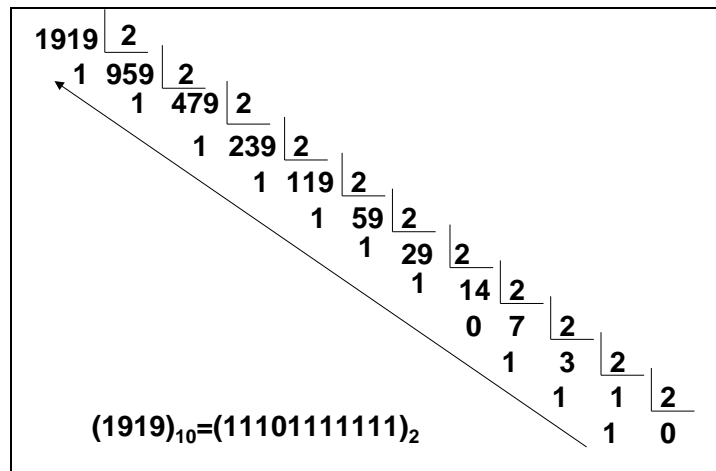


Σχήμα 1: Μετατροπή του αριθμού $(28)_{10}$ στο Δυαδικό και το Δεκαεξαδικό Σύστημα Αρίθμησης

Για να μετατρέψουμε το δεκαεξαδικό αριθμό $(77F)_{16}$ στο δυαδικό σύστημα μπορούμε να τον μετατρέψουμε πρώτα στο δεκαδικό αριθμό:

$$7 \times 16^2 + 7 \times 16^1 + 15 \times 16^0 = (1919)_{10}$$

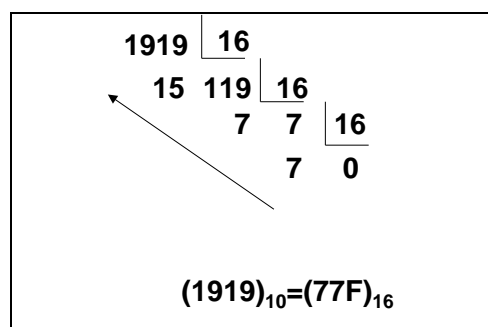
Στη συνέχεια μετατρέπουμε το δεκαδικό αριθμό στον αντίστοιχο δυαδικό αριθμό:



Σχήμα 2: Μετατροπή του αριθμού $(1919)_{10}$ στο δυαδικό σύστημα αρίθμησης
 Επομένως, η δυαδική παράσταση του αριθμού $(77F)_{16}$ είναι η $(1110111111)_2$.
 Αντίστροφα, για τη δεκαεξαδική παράσταση του αριθμού $(1110111111)_2$ βρίσκουμε
 πρώτα τη δεκαδική αναπαράσταση που είναι:

$$2^{10} + 2^9 + 2^8 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 1919$$

Στη συνέχεια η μετατροπή στο δεκαεξαδικό σύστημα θα δώσει $(77F)_{16}$.



Σχήμα 3: Μετατροπή του αριθμού $(1919)_{10}$ στο δεκαεξαδικό σύστημα αρίθμησης

2.6 Εισαγωγή στα λογικά κυκλώματα

2.6.1 Άλγεβρα Boole

Η δίτιμη άλγεβρα Boole είναι ένα αλγεβρικό σύστημα το οποίο περιλαμβάνει δύο στοιχεία (τα συμβολίζουμε συνήθως με '0' και '1') και τρεις τελεστές, τους οποίους συμβολίζουμε με:

+ (OR, Ή)

• (AND, ΚΑΙ)

~ (NOT, ΟΧΙ)

Συχνά, αντί για το σύμβολο ~ χρησιμοποιούμε το σύμβολο ', ή μια παύλα πάνω από τη μεταβλητή ($\sim x = x' = \bar{x}$), ενώ το σύμβολο • συχνά το παραλείπουμε (συμβολίζοντας

την πράξη ΚΑΙ ως $x \cdot y$ ή απλά xy). Οι κανόνες για τους τελεστές αυτούς ορίζονται σύμφωνα με τον ακόλουθο πίνακα:

x	y	$x + y$	x	y	$x \cdot y$	x	$\sim x$
0	0	0	0	0	0	0	1
0	1	1	0	1	0	1	0
1	0	1	1	0	0		
1	1	1	1	1	1		

Πίνακας: Τελεστές της δίτιμης άλγεβρας Boole

Οι μεταβλητές της άλγεβρας Boole μπορούν να πάρουν δύο τιμές, '0' ή '1'.

2.6.2 Συναρτήσεις Boole

Μια συνάρτηση Boole είναι μια έκφραση που σχηματίζεται από δυαδικές μεταβλητές, τους τελεστές Ή, ΚΑΙ, ΟΧΙ και παρενθέσεις. Για μια δεδομένη τιμή των μεταβλητών, η τιμή της συνάρτησης μπορεί να είναι είτε '0' είτε '1'. Μια συνάρτηση Boole μπορεί να οριστεί είτε με τον τύπο της, είτε με τον πίνακα αληθείας της. Για παράδειγμα, η συνάρτηση

$$F = xyz + xyz'$$

Παίρνει την τιμή '1' είτε όταν $x=y=z=1$ είτε όταν $x=y=1$, και $z=0$.

Ο πίνακας αληθείας της συνάρτησης F δίνεται στη συνέχεια.

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Πίνακας: Πίνακας αληθείας της συνάρτησης $F = xyz + xyz'$

2.6.3 Λογικές Πύλες


Μια λογική πύλη υλοποιεί μια λογική πράξη Boole και μπορεί να έχει μία, δύο ή περισσότερες εισόδους και μία έξοδο. Κάθε λογική πύλη έχει ένα σύμβολο και έναν πίνακα αληθείας. Ο πίνακας αληθείας δείχνει την τιμή της εξόδου για τις διαφορετικές τιμές των εισόδων. Οι λογικές πύλες χρησιμοποιούνται ακόμη στην υλοποίηση των άλλων μονάδων (αποκωδικοποιητών, πολυπλεκτών, αθροιστών κλπ.) που θα περιγραφούν στη συνέχεια.

Οι πιο γνωστές λογικές πράξεις είναι η σύζευξη (AND), η διάζευξη (OR) και η άρνηση (NOT). Οι πράξεις αυτές υλοποιούνται με τις λογικές πύλες AND, OR και NOT αντίστοιχα. Ακόμη, πολύ συχνά χρησιμοποιούνται οι πύλες NAND και NOR, οι οποίες εκτελούν τις αντίστροφες λογικές πράξεις από τις AND και OR αντίστοιχα, καθώς και οι πύλες της αποκλειστικής διάζευξης (XOR) και της άρνησής της (XNOR).

Πύλη NOT

Η λογική πράξη της άρνησης έχει μια μεταβλητή εισόδου και μία έξοδο, της οποίας η τιμή είναι '1' αν η είσοδος είναι '0', διαφορετικά η έξοδος είναι '0'. Στο Σχήμα 10 παρουσιάζουμε το λογικό σύμβολο της πύλης NOT, καθώς και τον πίνακα αληθείας της.

α	x
0	1
1	0



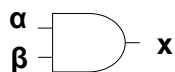
Σχήμα 10: Λογικό σύμβολο της πύλης NOT και πίνακας αληθείας

Συχνά, στην υλοποίηση λογικών συναρτήσεων, αντί για το σύμβολο της πύλης NOT χρησιμοποιούμε για απλότητα ένα κυκλάκι.

Πύλη AND

Η λογική πράξη της σύζευξης έχει δύο (ή περισσότερους) τελεστέους και μία έξοδο, η τιμή της οποίας είναι '1' αν όλα τα δεδομένα είναι '1'. Σε κάθε άλλη περίπτωση, το αποτέλεσμα είναι '0'. Η πράξη αυτή υλοποιείται με την πύλη AND. Στο Σχήμα 11 παρουσιάζουμε το λογικό σύμβολο της πύλης AND δύο εισόδων, καθώς και τον πίνακα αληθείας της.

α	β	x
0	0	0
0	1	0
1	0	0
1	1	1



Σχήμα 11: Λογικό σύμβολο της πύλης AND και πίνακας αληθείας

Πύλη OR

Η λογική πράξη της διάζευξης έχει δύο (ή περισσότερους) τελεστέους και μία έξοδο, η τιμή της οποίας είναι '0' αν όλα τα δεδομένα είναι '0'. Σε κάθε άλλη περίπτωση, η τιμή της εξόδου είναι '1'. Η πράξη αυτή υλοποιείται με την πύλη OR. Στο Σχήμα 12 παρουσιάζουμε το λογικό σύμβολο της πύλης OR δύο εισόδων, καθώς και τον πίνακα αληθείας της.

α	β	x
0	0	0
0	1	1
1	0	1
1	1	1



Σχήμα 12: Λογικό σύμβολο της πύλης OR και πίνακας αληθείας

2.6.4 Πύλες με περισσότερες από δύο εισόδους

Οι πύλες AND, OR, NAND, NOR, XOR και XNOR μπορούν να έχουν περισσότερες από δύο εισόδους. Στην περίπτωση αυτή, η έξοδος προσδιορίζεται με παρόμοιο τρόπο με εκείνον που γίνεται στην περίπτωση των πυλών με δύο εισόδους. Για παράδειγμα, ο ακόλουθος πίνακας δίνει την τιμή της εξόδου για πύλες AND και OR στην περίπτωση που έχουμε τρεις εισόδους (τις οποίες ονομάζουμε A, B, C).

A	B	C	AND	OR
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2.7 Διαγράμματα ροής (flow charts)

Από τα πρώτα βήματα που απαιτούνται να ακολουθήσει ο Μηχανικός Πληροφορικής με σκοπό τη μετατροπή ενός προβλήματος σε πρόγραμμα, είναι τα ακόλουθα: Αφού προσδιοριστεί πλήρως το πρόβλημα που καλείται να επιλύσει θα πρέπει να προχωρήσει στη διατύπωση του αλγορίθμου, δηλαδή ενός τυποποιημένου τρόπου λύσης. Προκειμένου να περιγράψει αυτόν τον αλγόριθμο, μπορεί δημιουργήσει το διάγραμμα ροής του προγράμματος.

Αυτό θα του δώσει τη δυνατότητα να ελέγξει τη λογική της επιλεγμένης / προτεινόμενης μεθόδου λύσης ώστε στη συνέχεια να προχωρήσει στη κωδικοποίηση σε κάποια γλώσσα προγραμματισμού, δηλαδή τη συγγραφή του αλγορίθμου σε γλώσσα κατανοητή από τον Η/Υ. Το αποτέλεσμα της κωδικοποίησης, ο κώδικας θα πρέπει να μεταγλωττισθεί, διορθωθεί και ελεγχθεί και θα αποτελέσει το πρόγραμμα που θα επιλύει το πρόβλημα που του τέθηκε.

Όπως ήδη αναφέρθηκε, για την επίδειξη της ακολουθίας των βημάτων ενός αλγορίθμου οι προγραμματιστές χρησιμοποιούν συχνά ένα διάγραμμα ροής (flow chart) που αποτελεί ένα καλό τρόπο διαγραμματικής αναπαράστασης των βημάτων αυτών. Για τους σκοπούς αυτούς χρησιμοποιούνται διάφορα γεωμετρικά σχήματα με καθορισμένη σημασία, σαν αυτά του Σχήματος 1.



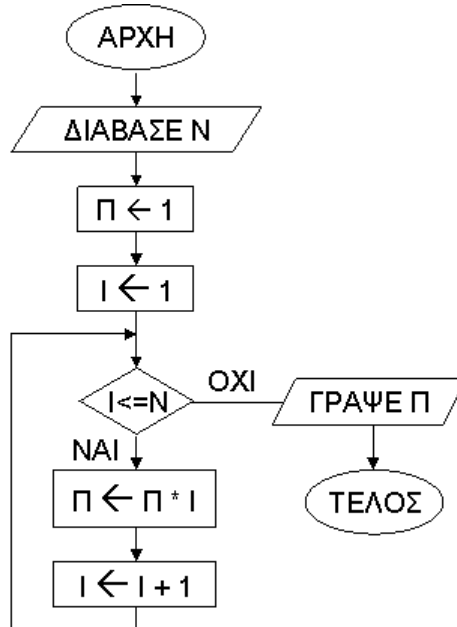
Στη συνέχεια δείτε 4 επιλεγμένα παραδείγματα εφαρμογής τέτοιων Διαγραμμάτων Ροής.

Παράδειγμα 1: Το παραγοντικό (factorial) ενός φυσικού αριθμού ορίζεται από τον ακόλουθο τύπο:

$N! = N * (N-1) * (N-2) * \dots * 1$ για $N \geq 1$, και $N! = 1$ για $N = 0$.

Να δημιουργηθεί Διάγραμμα Ροής που να περιγράφει τον παραπάνω τρόπο υπολογισμού του παραγοντικού.

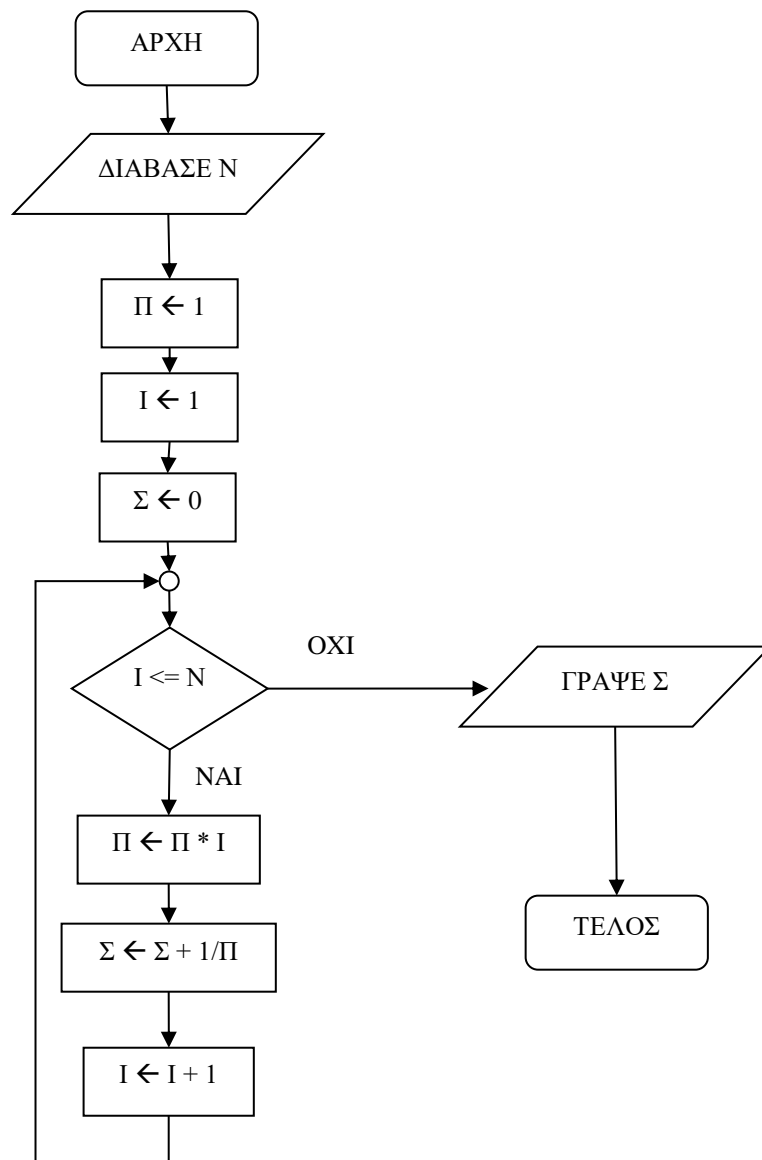
Απάντηση παραδείγματος 1:



Παράδειγμα 2: Δώστε το Διάγραμμα Ροής Προγράμματος για τον υπολογισμό του κάτωθι αθροίσματος:

$$S = 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \dots + \frac{1}{N!}, \text{ δεδομένου ότι } N > 0.$$

Απάντηση παραδείγματος 2:



2.8 Συνοπτικός Οδηγός Ψευδοκώδικα

Στην παρούσα εισαγωγή παρουσιάζονται οι βασικές έννοιες για την περιγραφή αλγορίθμων με χρήση ψευδοκώδικα στην ελληνική γλώσσα.

Μεταβλητή

Είναι μια υπολογιστική οντότητα που παριστάνει ένα μέγεθος (έννοια) που σχετίζεται με το πρόβλημα που λύνει ο αλγόριθμος, οι τιμές του οποίου (μεγέθους) χρειάζεται να αποθηκευτούν στη μνήμη του υπολογιστή. Κάθε μέγεθος (έννοια) του προβλήματος που εμπλέκεται στη λύση του πρέπει να αντιπροσωπεύεται από μια μεταβλητή στον αλγόριθμο. Κάθε μεταβλητή μπορεί να μην έχει σταθερή τιμή, αλλά να παίρνει διάφορες τιμές, που αντιπροσωπεύουν το αντίστοιχο μέγεθος (έννοια) που παριστάνει, κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου (προγράμματος). Το όνομα μιας μεταβλητής περιέχει γράμματα, αριθμούς και την παύλα ('-'), αλλά αρχίζει πάντα από γράμμα. Π.χ. X, X1, X-1, MAX, MAX2, MAX-2, είναι σωστά ονόματα μεταβλητών. Κάθε μεταβλητή επίσης αντιπροσωπεύει (ονοματίζει) και τη θέση μνήμης στην οποία βρίσκεται αποθηκευμένη η εκάστοτε τιμή του μεγέθους που παριστάνει. Σημειώστε ότι για τον μεταγλωττιστή του ΕΑΠ θα πρέπει το όνομα της μεταβλητής να γραφεί με λατινικούς χαρακτήρες.

Τύπος

Τα μεγέθη που παριστάνουν οι μεταβλητές δεν είναι όλα όμοια, διαφέρουν ως προς τις ιδιότητες και το είδος των τιμών που παίρνουν. Οι μεταβλητές, ανάλογα με το τι μέγεθος παριστάνουν, δηλ. τι είδους τιμές παίρνουν, είναι και διαφορετικού τύπου. Υπάρχουν τέσσερις βασικοί (ή πρωτογενείς) τύποι μεταβλητών:

- **INTEGER** (όταν η μεταβλητή παίρνει σαν τιμές ακέραιους αριθμούς, π.χ. 5, 18, -17)
- **REAL** (όταν η μεταβλητή παίρνει σαν τιμές πραγματικούς αριθμούς, π.χ. 3.1, -24.6)
- **CHAR** (όταν η μεταβλητή παίρνει σαν τιμή ένα μόνο χαρακτήρα, π.χ. "Α", "7",)
- **BOOLEAN** (όταν η μεταβλητή παίρνει μια από τις δύο λογικές τιμές: TRUE ή FALSE).

- **STRING** (όταν η μεταβλητή παίρνει σαν τιμές ένα σύνολο (μια ακολουθία) χαρακτήρων, π.χ. “DEKA”, “NIKOS!”, “ANNA1”)

Σταθερές

Είναι υπολογιστικές οντότητες που έχουν μια σταθερή τιμή, η οποία δεν αλλάζει κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου (προγράμματος). Τα ονόματά τους σχηματίζονται με τους ίδιους κανόνες όπως και των μεταβλητών. Δεν χρειάζεται να δηλώσουμε τον τύπο μιας σταθεράς, διότι αυτός προκύπτει από την τιμή της.

Τιμή

Οτιδήποτε δεν χρειάζεται αποθήκευση. Οι τιμές έχουν και αυτές τύπους, που όμως δεν χρειάζεται να δηλωθούν, διότι αναγνωρίζονται από τη ίδια τους την εμφάνιση. Π.χ. τα 12, 13.6, -23, “N”, TRUE είναι τιμές τύπου INTEGER (τα 12, -23), REAL (το 13.6), CHAR (το “N”), BOOLEAN (το TRUE).

Τελεστές

Είναι σύμβολα που παριστάνουν κάποια στοιχειώδη λειτουργία ή επεξεργασία. Υπάρχουν διάφορες κατηγορίες:

- **Αριθμητικοί τελεστές:** +, -, *, /, DIV, MOD (Το DIV είναι ο τελεστής ακέραιας διαίρεσης, δηλ. έχει ως αποτέλεσμα το πηλίκο μιας διαίρεσης ακεραίων: $5 \text{ DIV } 2 = 2$, ενώ $5/2=2.5$, και ο MOD είναι ο τελεστής υπολοίπου ακέραιας διαίρεσης, δηλ. έχει ως αποτέλεσμα το υπόλοιπο μιας διαίρεσης ακεραίων: $5 \text{ MOD } 2 = 1$).
- **Τελεστές σύγκρισης ή Συσχετιστικοί τελεστές:** =, >, >=, <, <=, <>
- **Λογικοί τελεστές:** NOT, OR, AND

Εκφράσεις

Οι εκφράσεις είναι στοιχεία του ψευδοκώδικα που παράγονται από το συνδυασμό τελεστών και μεταβλητών ή τιμών. Διακρίνουμε τις παρακάτω κατηγορίες:

- **Αριθμητικές:** Παράγονται από το συνδυασμό αριθμητικών τελεστών, παρενθέσεων και αριθμητικών μεταβλητών ή σταθερών (δηλ. τύπου INTEGER ή REAL). Π.χ. $(5 * X + Y)$, $((K + 3)/4) * L$, $K \text{ DIV } 3$ είναι αριθμητικές εκφράσεις. Οι αριθμητικές εκφράσεις επιστρέφουν αριθμητικές τιμές (δηλ. τύπου INTEGER ή REAL), ανάλογα με τον τύπο των μεταβλητών και των σταθερών που συμμετέχουν σ’ αυτές.

- **Σύγκρισης:** Παράγονται από το συνδυασμό τελεστών σύγκρισης, παρενθέσεων και αριθμητικών μεταβλητών ή σταθερών (δηλ. τύπου INTEGER ή REAL) αλλά και τύπου χαρακτήρα (σπανιότερα). Π.χ. $X \geq Y$, $A = B$, $K < L$ και $X \neq Y$ είναι εκφράσεις σύγκρισης. Η τιμή που επιστρέφει μια έκφραση σύγκρισης είναι τύπου BOOLEAN.
- **Λογικές:** Παράγονται από το συνδυασμό τελεστών σύγκρισης, λογικών τελεστών, παρενθέσεων και μεταβλητών ή σταθερών οποιουδήποτε τύπου. Π.χ. $(X > Y) \text{ AND } (X < 10)$, $(A \neq B) \text{ OR } (A \geq 5)$, $\text{NOT } (X < Y)$ είναι λογικές εκφράσεις. Η τιμή που επιστρέφει μια έκφραση σύγκρισης είναι επίσης τύπου BOOLEAN.

Δηλώσεις μεταβλητών

Η δήλωση των μεταβλητών γίνεται ως εξής:

<λίστα μεταβλητών> : <τύπος> ;

Για παράδειγμα:

I, J : INTEGER;

X, Y, Z : REAL;

C : CHAR;

TELOS : BOOLEAN;

Δηλώσεις Σταθερών

Η δήλωση των σταθερών γίνεται ως εξής:

ΣΤΑΘΕΡΕΣ

<όνομα σταθεράς 1> =<τιμή σταθεράς 1> ,

<όνομα σταθεράς 2> =<τιμή σταθεράς 2> ,

.....

<όνομα σταθεράς ν> =<τιμή σταθεράς ν> ;

Για παράδειγμα:

ΣΤΑΘΕΡΕΣ

MAX_DAYS = 31,

MAX = 45.0,

MIN = 0.0;

Τύπος πίνακα

Ο τύπος πίνακα είναι ένας σύνθετος τύπος. Δηλ. είναι ένας τύπος που οι τιμές που αντιπροσωπεύει δεν είναι απλές, αλλά σύνολα τιμών. Ο τύπος πίνακα χρησιμοποιείται όταν πρέπει να χρησιμοποιήσουμε μεταβλητές για να παραστήσουμε σχετικά μεγάλο αριθμό από ομοειδή μεγέθη.

Ένας μονοδιάστατος πίνακας δηλώνεται ως εξής:

ΌνομαΠίνακα : [1..πλήθοςστοιχείων] OF ΤύποςΠίνακα
--

Π.χ. αν θέλουμε να παραστήσουμε και να αποθηκεύσουμε τους βαθμούς 20 φοιτητών σ' ένα μάθημα, θα πρέπει να χρησιμοποιήσουμε 20 διαφορετικές μεταβλητές (μία για κάθε μαθητή) που η κάθε μια θα αντιπροσωπεύει το βαθμό ενός μαθητή. Επειδή αυτό δεν είναι βολικό (σκεφθείτε να είχαμε 200 φοιτητές ή και περισσότερους!!!), στην περίπτωση αυτή η λύση είναι να χρησιμοποιήσουμε μια μεταβλητή τύπου πίνακα. Τη μεταβλητή αυτή (ας την ονομάσουμε BATHMOS) τη δηλώνουμε ως εξής:

BATHMOS : ARRAY [1..20] OF INTEGER;

Δηλ. δηλώνουμε ότι η μεταβλητή BATHMOS είναι ένας πίνακας (ARRAY) που αποτελείται από 20 θέσεις (με δείκτες-ετικέτες 1, 2, ..., 20), όπου μπορούμε να αποθηκεύσουμε ακέραιες τιμές (INTEGER). Τα στοιχεία που είναι έντονα (bold) στην παραπάνω δήλωση δεν μπορούν να αλλάζουν, τα υπόλοιπα ναι. Αν θέλαμε π.χ. να μπορεί να αποθηκεύει πραγματικούς αριθμούς και να έχει 10 θέσεις η μεταβλητή μας, θα τη δηλώνουμε ως εξής:

BATHMOS : ARRAY [1..10] OF REAL;

Στο παρακάτω σχήμα φαίνεται ο πίνακας αυτός με διάφορες τιμές αποθηκευμένες σ' αυτόν.

BATHMOS

1	2	3	4	5	6	7	8	9	10
10	15.5	12	14	16.5	17.5	13	18	14.5	11

Η πρόσβαση στις τιμές (ή στοιχεία) ενός **μονοδιάστατου** πίνακα (ή με άλλα λόγια, μιας μεταβλητής τύπου πίνακα) γίνεται μέσω του ονόματος του πίνακα και **ενός** δείκτη που δείχνει τη θέση της τιμής στον πίνακα. Π.χ. η BATHMOS[3] αντιπροσωπεύει την τιμή του τρίτου στοιχείου του παραπάνω πίνακα. Το ίδιο ισχύει για την BATHMOS[I], η οποία αντιπροσωπεύει την τιμή του I-οστού στοιχείου (π.χ. αν το I έχει τιμή 4, του 4^{ου} στοιχείου). Οι BATHMOS[3], BATHMOS[I] ονομάζονται *μεταβλητές με δείκτη θέσης*. Τα στοιχεία ενός πίνακα αποθηκεύονται σε διαδοχικές θέσεις στη μνήμη. Οι μεταβλητές με δείκτη θέσης μπορούν να χρησιμοποιηθούν όπως ακριβώς και οι απλές μεταβλητές, π.χ. για το σχηματισμό μιας αριθμητικής έκφρασης.

Μπορεί να έχουμε πίνακες πολλών διαστάσεων. Ένας πίνακας δύο διαστάσεων δηλώνεται ως εξής:

ΌνομαΠίνακα : [1..αριθμόςγραμμών, 1..αριθμόςστηλών] OF ΤύποςΠίνακα

Π.χ. Π: **ARRAY**[1..10,1..10] **OF** INTEGER;

Η πρόσβαση στις τιμές (ή στοιχεία) ενός πίνακα **δύο διαστάσεων** γίνεται μέσω του ονόματος του πίνακα και **δύο** δεικτών θέσης που δείχνουν τη θέση (δηλαδή Γραμμή, Στήλη) της τιμής στον πίνακα.

Οι BATHMOS[3,2], BATHMOS[I,J] ονομάζονται μεταβλητές με δείκτη θέσης.

Οι τιμές των δεικτών για την πρώτη γραμμή και πρώτη στήλη είναι οι 1,1

Βασικές εντολές ψευδοκώδικα

Τρεις είναι οι βασικές εντολές του ψευδοκώδικα:

- Εντολή καταχώρισης
- Εντολή για είσοδο δεδομένων

- Εντολή για έξοδο αποτελεσμάτων

Εντολή καταχώρησης ($:=$)

Με την εντολή αυτή καταχωρούμε τιμές σε μεταβλητές.

Σύνταξη

$\langle \text{μεταβλητή} \rangle := \langle \text{παράσταση} \rangle$, όπου η $\langle \text{παράσταση} \rangle$ μπορεί να είναι μια μεταβλητή ή μια σταθερά ή μια τιμή ή μια έκφραση.

Π.χ. $X := 12$, $X := Y$, $X := (Y + 10)/4$ είναι εντολές καταχώρησης.

Λειτουργία

Υπολογίζεται πρώτα η $\langle \text{παράσταση} \rangle$ αν χρειάζεται, και στη συνέχεια το αποτέλεσμα (τιμή) καταχωρείται ως τιμή στη $\langle \text{μεταβλητή} \rangle$.

Για να καταχωρήσω μια τιμή σε μια θέση ενός πίνακα, καταχωρώ την τιμή στη αντίστοιχη μεταβλητή με δείκτη. Π.χ. αν θέλω να καταχωρήσω στον παραπάνω πίνακα στη θέση 5 τη τιμή 14.5, τότε χρησιμοποιώ την εντολή $\text{BATHMOS}[5] := 14.5$. Αποτέλεσμα αυτής της εντολής θα είναι να αλλάξει η τιμή του πίνακα στη θέση 5 από 16.5 σε 14.5.

Εντολή για είσοδο δεδομένων (ΔΙΑΒΑΣΕ)

Για να εισάγονται δεδομένα στον αλγόριθμο (πρόγραμμα) από το πληκτρολόγιο χρησιμοποιείται μια εντολή εισόδου.

Σύνταξη

ΔΙΑΒΑΣΕ ($\langle \text{λίστα μεταβλητών} \rangle$), όπου η $\langle \text{λίστα μεταβλητών} \rangle$ είναι μια σειρά από ονόματα μεταβλητών χωριζόμενα με κόμμα μεταξύ τους.

Π.χ. ΔΙΑΒΑΣΕ (X);

ΔΙΑΒΑΣΕ (X, Y);

Λειτουργία

Όταν εκτελείται η εντολή αυτή, παύει προσωρινά η εκτέλεση του αλγορίθμου (προγράμματος) και ο υπολογιστής αναμένει την είσοδο (πληκτρολόγηση) τόνων στοιχείων (αριθμών ή χαρακτήρων ή λέξεων) όσων και οι μεταβλητές που υπάρχουν στην εντολή ΔΙΑΒΑΣΕ. Στην πράξη, το τέλος της εισόδου δεδομένων σηματοδοτείται

(από το χρήστη) με την πίεση του πλήκτρου ENTER.. Όταν δοθούν τα δεδομένα καταχωρούνται στις αντίστοιχες μεταβλητές.

Εντολή για έξοδο αποτελεσμάτων (ΤΥΠΩΣΕ)

Για να εκτυπώνονται αποτελέσματα στην οθόνη χρησιμοποιείται μια εντολή εξόδου.

Σύνταξη

ΤΥΠΩΣΕ (<λίστα παραμέτρων>), όπου η <λίστα παραμέτρων> είναι μια σειρά από μεταβλητές ή σταθερές ή εκφράσεις ή μηνύματα χωριζόμενα με κόμμα μεταξύ τους.

Ένα μήνυμα είναι μια φράση ανάμεσα σε διπλά εισαγωγικά, όπως για παράδειγμα: “ΤΟ ΓΙΝΟΜΕΝΟ ΕΙΝΑΙ: ”, “X = ”.

Παραδείγματα

ΤΥΠΩΣΕ (“ΤΟ ΕΜΒΑΔΟΝ ΕΙΝΑΙ: ”, E);

ΤΥΠΩΣΕ (“X = ”, Y*(Z+2));

ΤΥΠΩΣΕ (“ΤΟ ΕΜΒΑΔΟΝ ΕΙΝΑΙ: ”, E, “ ΚΑΙ Ο ΟΓΚΟΣ: ”, V);

Λειτουργία

Όταν εκτελείται η εντολή αυτή, εκτυπώνεται στην οθόνη η τιμή κάθε στοιχείου της λίστας παραμέτρων με τη σειρά. Αν το στοιχείο είναι μεταβλητή τότε εκτυπώνεται η τιμή της μεταβλητής. Αν είναι μήνυμα, τότε εκτυπώνεται ό,τι βρίσκεται ανάμεσα στα εισαγωγικά όπως είναι. Αν είναι μια έκφραση, τότε υπολογίζεται η τιμή της και εκτυπώνεται το αποτέλεσμα. Π.χ. αν εκτελεστεί η τελευταία από τις παραπάνω εντολές-παραδείγματα, και οι τιμές των E και V είναι αντίστοιχα 15.35 και 28.9, θα εκτυπωθεί στην οθόνη το ακόλουθο:

ΤΟ ΕΜΒΑΔΟΝ ΕΙΝΑΙ: 15.35 ΚΑΙ Ο ΟΓΚΟΣ: 28.9

Να σημειωθεί ότι όταν ολοκληρώνεται μία ΤΥΠΩΣΕ στον μεταγλωττιστή του ΕΑΠ, δεν αλλάζει αυτόματα η γραμμή στην έξοδο. Για να το επιτύχουμε αυτό, μπορούμε να χρησιμοποιήσουμε την σταθερά αλλαγής γραμμής EOLN.

Παράδειγμα

Οι εντολές ΤΥΠΩΣΕ (“ΠΑΗ”);

ΤΥΠΩΣΕ (“10”, EOLN, “ΕΑΠ”, EOLN);

Θα εμφανίσουν:

ΠΛΗ10

ΕΑΠ

Εντολή/Δομή επιλογής

Σύνταξη

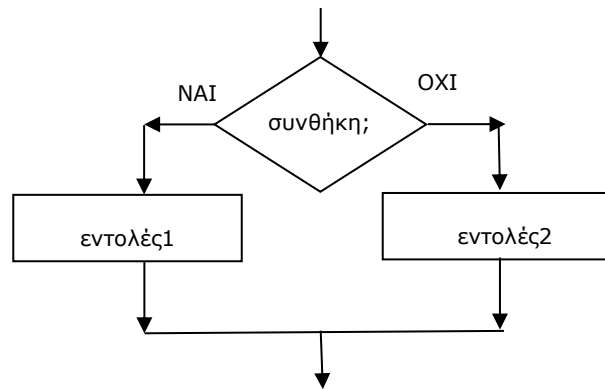
ΕΑΝ <συνθήκη> **ΤΟΤΕ**

<εντολές1>

ΑΛΛΙΩΣ

<εντολές2>

ΕΑΝ-ΤΕΛΟΣ



Το τμήμα “ΑΛΛΙΩΣ <εντολές>” είναι προαιρετικό. Η <συνθήκη> είναι μια έκφραση σύγκρισης ή μια λογική έκφραση. Τα τμήματα <εντολές1> και <εντολές2> αποτελούνται από μια ή περισσότερες εντολές. Π.χ.

Παράδειγμα 1

ΕΑΝ (X > 10) **ΤΟΤΕ**

Y := Y + X

ΑΛΛΙΩΣ

Y := Y - X

ΕΑΝ-ΤΕΛΟΣ

Παράδειγμα 2

ΕΑΝ (X > 0) **ΑΝΔ** (X < 5) **ΤΟΤΕ**

Y := Y * X;

X := X + 1

ΑΛΛΙΩΣ

Y := Y + X

ΕΑΝ-ΤΕΛΟΣ

Παράδειγμα 3

ΕΑΝ (X > 10) **ΤΟΤΕ**

X := X + 1;

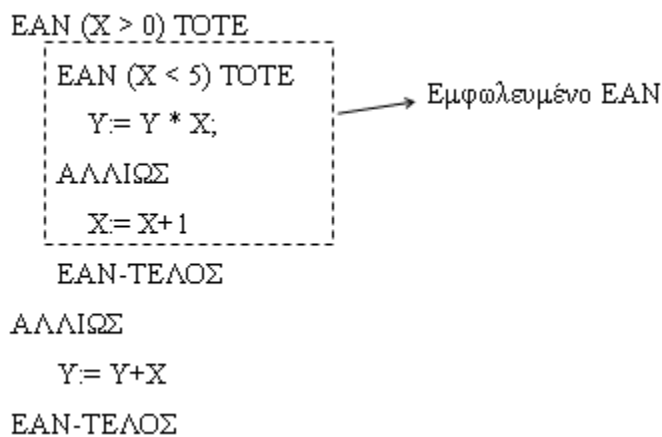
ΤΥΠΩΣΕ (X)

ΕΑΝ-ΤΕΛΟΣ

Λειτουργία

Όταν εκτελείται μια εντολή ΕΑΝ τότε, πρώτα εξετάζεται η <συνθήκη>. Αν αυτή είναι αληθής, τότε εκτελούνται οι εντολές που αποτελούν το τμήμα <εντολές1>, αλλιώς (αν δηλ. είναι ψευδής) εκτελούνται οι εντολές που αποτελούν το τμήμα <εντολές2>. Αν δεν υπάρχει το τμήμα “ΑΛΛΙΩΣ <εντολές2>”, τότε η εκτέλεση του αλγορίθμου συνεχίζει στην επόμενη εντολή του αλγορίθμου (αυτή δηλ. που βρίσκεται μετά το ΕΑΝ-ΤΕΛΟΣ).

Τα τμήματα <εντολές1> και <εντολές2> μπορεί να είναι άλλες εντολές ΕΑΝ, οπότε δημιουργούνται εμφωλευμένες ή σύνθετες εντολές ΕΑΝ. Π.χ.



Στο παραπάνω παράδειγμα, **αν** η τιμή που έχει η μεταβλητή X κάνει τη συνθήκη

“(X > 0) **AND** (X < 5)” αληθή, **τότε** εκτελείται η εντολή “Y := Y * X;”.

Αν η τιμή που έχει η μεταβλητή X κάνει τη συνθήκη

(X > 0) **AND** (X >= 5)) αληθή [δηλαδή: Αν (X > 0) **AND** (NOT(X < 5)) : αληθές] τότε εκτελείται η εντολή X := X + 1

Αν η τιμή που έχει η μεταβλητή X κάνει τη συνθήκη

(X <= 0) αληθή [δηλαδή: Αν (NOT(X > 0)): αληθές] τότε εκτελείται η εντολή Y := Y + X

Εντολές/Δομές επανάληψης

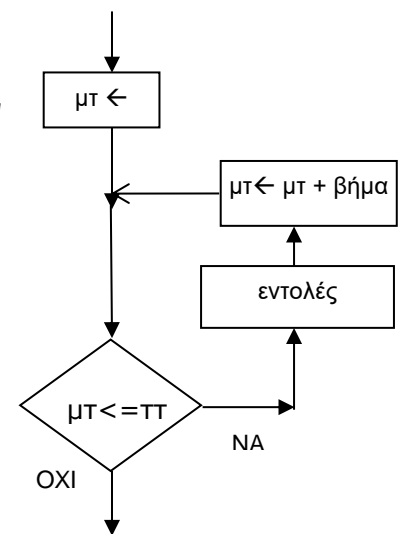
Υπάρχουν τρεις τύποι εντολών / δομών επανάληψης: ΓΙΑ-ΕΠΑΝΑΛΑΒΕ, ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ και ΕΠΑΝΑΛΑΒΕ-MEXPI.

ΓΙΑ-ΕΠΑΝΑΛΑΒΕ

Σύνταξη

ΓΙΑ <μτ>:=<ατ> **ΕΩΣ** <ττ> **ΜΕ ΒΗΜΑ** <βήμα> **ΕΠΑΝΑΛΑΒΕ**
<εντολές>

ΓΙΑ-ΤΕΛΟΣ



Το <μτ> είναι μια μεταβλητή, τα <ατ> και <ττ> είναι ακέραιοι αριθμοί ή μεταβλητές με ακέραιες τιμές ή εκφράσεις που το αποτέλεσμα τους είναι ακέραιος αριθμός. Ο <ατ> παριστάνει την αρχική τιμή της <μτ> και ο <ττ> την τελική τιμή της <μτ>.

Το τμήμα <εντολές> αντιπροσωπεύει μια ή περισσότερες εντολές, χωριζόμενες με το “;” και αποτελεί το ‘σώμα’ της επανάληψης, δηλ. το τμήμα του οποίου η εκτέλεση επαναλαμβάνεται ένα αριθμό από φορές. Η μεταβλητή <μτ> σε κάθε επανάληψη αυξάνει (ή μειώνει) την τιμή της σταθερά, τόσο όσο δηλώνεται από το <βήμα>. Εάν το <βήμα> δεν δηλωθεί, τότε θεωρείται ότι είναι ίσο με 1. Εάν το <βήμα> έχει αρνητική τιμή, τότε σε κάθε επανάληψη η μεταβλητή <μτ> μειώνεται κατά όσο ορίζει το <βήμα>. Για παράδειγμα:

Παράδειγμα 1

X:= 10; //αρχικοποίηση της X

ΓΙΑ I:= 1 **ΕΩΣ** 10 **ΕΠΑΝΑΛΑΒΕ**

Παράδειγμα 2

S:= 0; //αρχικοποίηση της S

ΓΙΑ K:= J **ΕΩΣ** (N+1) **ΕΠΑΝΑΛΑΒΕ**

$X := X + I;$

$S := S + K$

ΤΥΠΩΣΕ (X)

ΓΙΑ-ΤΕΛΟΣ

ΓΙΑ-ΤΕΛΟΣ

Παράδειγμα 3

ΓΙΑ $a := 2$ **ΕΩΣ** 10 **ΜΕ ΒΗΜΑ** 2

ΕΠΑΝΑΛΑΒΕ

ΤΥΠΩΣΕ (a)

ΓΙΑ-ΤΕΛΟΣ

Λειτουργία

Όταν εκτελείται μια εντολή **ΓΙΑ-ΕΠΑΝΑΛΑΒΕ** τότε, κατ' αρχήν η μεταβλητή <μτ> παίρνει ως τιμή την <ατ> και εκτελούνται οι <εντολές>.

Στη συνέχεια η τιμή της <μτ> αυξάνει ή μειώνει κατά όσο δηλώνεται από το <βήμα> και ξαναεκτελούνται οι <εντολές>.

Αυτό επαναλαμβάνεται μέχρι η <μτ> να πάρει τιμή μεγαλύτερη από <ττ>, οπότε σταματά η επανάληψη, δηλ. η εκτέλεση των εντολών και η εκτέλεση προχωρά στη επόμενη εντολή του αλγορίθμου.

Π.χ. στο Παράδειγμα 1 οι εντολές “ $X := X + I;$ **ΤΥΠΩΣΕ** (X)” θα εκτελεστούν 10 φορές (οπότε ως αποτέλεσμα, δεδομένου ότι το X έχει αρχική τιμή 10, θα τυπωθούν οι αριθμοί 11, 13, 16, 20, 25, 31, 38, 46, 55, 65). Ενώ στο Παράδειγμα 3 θα τυπωθούν μόνο οι άρτιοι:

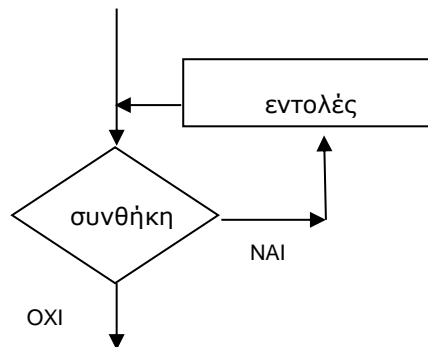
ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ

Σύνταξη

ΕΝΟΣΩ <συνθήκη> ΕΠΑΝΑΛΑΒΕ

<εντολές>

ΕΝΟΣΩ-ΤΕΛΟΣ



Η <συνθήκη> είναι μια έκφραση σύγκρισης ή μια λογική έκφραση. Το τμήμα <εντολές> αντιπροσωπεύει μια ή περισσότερες εντολές, χωριζόμενες με το “;”. Αποτελεί δε το «σώμα» της επανάληψης, δηλ. το τμήμα του οποίου η εκτέλεση επαναλαμβάνεται κάποιες φορές. Π.χ τα παραπάνω παραδείγματα με τη χρήση της ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ γίνονται:

Παράδειγμα 1

I:= 1; //αρχικοποίηση της I

X:= 10;

ΕΝΟΣΩ (I <= 10) ΕΠΑΝΑΛΑΒΕ

X:= X+I;

ΤΥΠΩΣΕ (X);

I:= I + 1

ΕΝΟΣΩ-ΤΕΛΟΣ

Παράδειγμα 2

K:= J; //αρχικοποίηση της K

ΕΝΟΣΩ (K <= (N+1)) ΕΠΑΝΑΛΑΒΕ

S:= S+K;

K:= K+1

ΕΝΟΣΩ-ΤΕΛΟΣ

Λειτουργία

Όταν εκτελείται μια εντολή ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ τότε, πρώτα εξετάζεται η <συνθήκη>. Αν είναι αληθής, εκτελούνται οι <εντολές> (αλλιώς η εκτέλεση του αλγορίθμου συνεχίζει στην επόμενη εντολή του αλγορίθμου, αυτή δηλ. που βρίσκεται

μετά το ΕΝΟΣΩ-ΤΕΛΟΣ). Αφού εκτελεστούν οι <εντολές>, επανεξετάζεται η <συνθήκη>. Αν βρεθεί πάλι αληθής, ξαναεκτελούνται οι <εντολές> κ.ο.κ. έως ότου η <συνθήκη> γίνει ψευδής, οπότε σταματά η εκτέλεση της επανάληψης και η εκτέλεση του αλγορίθμου συνεχίζεται με την επόμενη εντολή,, αυτή δηλ. που βρίσκεται μετά το ΕΝΟΣΩ-ΤΕΛΟΣ.

Παρατήρηση

Η <συνθήκη> πρέπει να περιέχει μεταβλητή που να μεταβάλλεται μέσα στο σώμα της επανάληψης (δηλ. στις <εντολές>), ώστε να μπορεί σε κάποια επανάληψη να γίνει η συνθήκη από αληθής ψευδής. Τέτοιες μεταβλητές στα παραπάνω παραδείγματα είναι οι I και K.

Στο παραπάνω Παράδειγμα 1, όταν συναντάται η εντολή ΕΝΟΣΩ-ΕΠΑΝΑΛΑΒΕ, επειδή η συνθήκη “ $I \leq 10$ ” είναι αληθής (επειδή η αρχική τιμή του I είναι 1), εκτελούνται οι εντολές “ $X := X + I$; ΤΥΠΩΣΕ (X); $I := I + 1$ ”, δηλ. το X θα γίνει $10 + 1 = 11$ και θα τυπωθεί, και το I θα γίνει $1 + 1 = 2$. Στη συνέχεια, εξετάζεται η συνθήκη, που είναι πάλι αληθής (αφού $2 < 10$), και ξαναεκτελούνται οι εντολές κ.ο.κ., μέχρις ότου το I γίνει 11, οπότε η συνθήκη είναι ψευδής (διότι $11 > 10$), σταματά η επανάληψη και η εκτέλεση του αλγορίθμου συνεχίζεται με την επόμενη εντολή, αυτή δηλ. που βρίσκεται μετά το ΕΝΟΣΩ-ΤΕΛΟΣ.

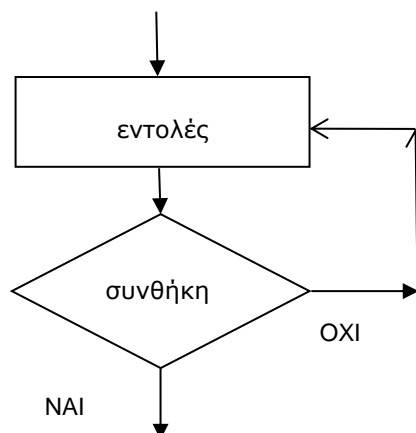
ΕΠΑΝΑΛΑΒΕ-MEXPI

Σύνταξη

ΕΠΑΝΑΛΑΒΕ

<εντολές>

MEXPI <συνθήκη>



Η <συνθήκη> είναι μια έκφραση σύγκρισης ή μια λογική έκφραση. Το τμήμα <εντολές> αντιπροσωπεύει μια ή περισσότερες εντολές, χωριζόμενες με το “;” αποτελεί το ‘σώμα’ της επανάληψης, δηλ. το τμήμα του οποίου η εκτέλεση επαναλαμβάνεται κάποιες φορές. Π.χ. τα παραπάνω παραδείγματα με τη χρήση της ΕΠΑΝΑΛΑΒΕ-MEXPI γίνονται:

Παράδειγμα 1	Παράδειγμα 2
I:= 1; //αρχικοποίηση της I	K:= J1; //αρχικοποίηση της K
X:= 10;	ΕΠΑΝΑΛΑΒΕ
ΕΠΑΝΑΛΑΒΕ	S:= S+K;
X:= X+I;	K:= K+1
ΤΥΠΩΣΕ (X);	MEXPI (K > (N+1))
I:= I + 1	
MEXPI (I > 10)	

Λειτουργία

Όταν εκτελείται μια εντολή ΕΠΑΝΑΛΑΒΕ-MEXPI τότε, πρώτα εκτελούνται οι <εντολές>. Αφού εκτελεστούν οι <εντολές>, εξετάζεται η <συνθήκη>. Αν βρεθεί ψευδής, ξαναεκτελούνται οι <εντολές> κ.ο.κ. έως ότου η <συνθήκη> γίνει αληθής, οπότε σταματά η εκτέλεση της επανάληψης και η εκτέλεση του αλγορίθμου συνεχίζεται με την επόμενη εντολή, αυτή δηλ. που βρίσκεται μετά το MEXPI <συνθήκη>.

Παρατήρηση

Η <συνθήκη> πρέπει να περιέχει μεταβλητή που να μεταβάλλεται μέσα στο σώμα της επανάληψης (δηλ. στις <εντολές>), ώστε να μπορεί σε κάποια επανάληψη να γίνει η συνθήκη από ψευδής αληθής. Τέτοιες μεταβλητές στα παραπάνω παραδείγματα είναι οι I και K.

Στο παραπάνω Παράδειγμα 1, όταν συναντάται η εντολή ΕΠΑΝΑΛΑΒΕ-MEXPI, εκτελούνται οι εντολές “X:= X+I; ΤΥΠΩΣΕ (X); I:= I + 1”, δηλ. το X θα γίνει 10+1=11

και θα τυπωθεί και το I θα γίνει $1+1=2$. Στη συνέχεια, εξετάζεται η συνθήκη, που είναι ψευδής (αφού $2 < 10$), και ξαναεκτελούνται οι εντολές κ.ο.κ., μέχρις ότου το I γίνει 11, οπότε η συνθήκη γίνεται αληθής (διότι $11 > 10$), σταματά η επανάληψη και η εκτέλεση του αλγορίθμου συνεχίζεται με την επόμενη εντολή, αυτή δηλ. που βρίσκεται μετά το MEXPI ($I > 10$).

3^ο Κεφάλαιο – Αντιπροσωπευτικές ασκήσεις

3.1 Αντιπροσωπευτικές ασκήσεις μετατροπών μεταξύ δυαδικού και δεκαδικού συστήματος αναπαράστασης

1. Να μετατραπεί ο αριθμός $(83)_{10}$ του δεκαδικού συστήματος αρίθμησης στο δυαδικό σύστημα αρίθμησης.

Ακολουθεί ο αλγόριθμος μετατροπής των διαδοχικών διαιρέσεων με το 2 (η διαίρεση είναι ακέραια) και την καταγραφή των υπολοίπων:

	πηλίκιο	υπόλοιπο
• $83 : 2 =$	41	1
• $41 : 2 =$	20	1
• $20 : 2 =$	10	0
• $10 : 2 =$	5	0
• $5 : 2 =$	2	1
• $2 : 2 =$	1	0
• $1 : 2 =$	0	1

Συνεπώς, η δυαδική αναπαράσταση του 83 είναι: 1010011

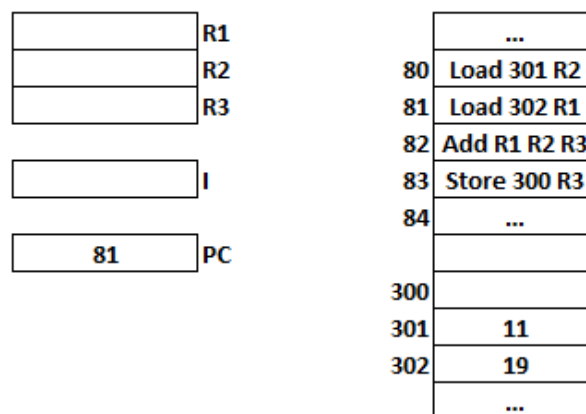
2. Να μετατραπεί ο δυαδικός αριθμός $(1010011)_2$ στο δεκαδικό σύστημα αρίθμησης.

Ακολουθεί η μετατροπή με χρήση του αλγορίθμου αθροίσματος των γινομένων ψηφίων επί των αντίστοιχων δυνάμεων του 2.

$$(1010011)_2 = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 64 + 16 + 2 + 1 = 83.$$

3.2 Αντιπροσωπευτική άσκηση για εκτέλεση προγράμματος από μία CPU

Με βάση το πιο κάτω πρόγραμμα



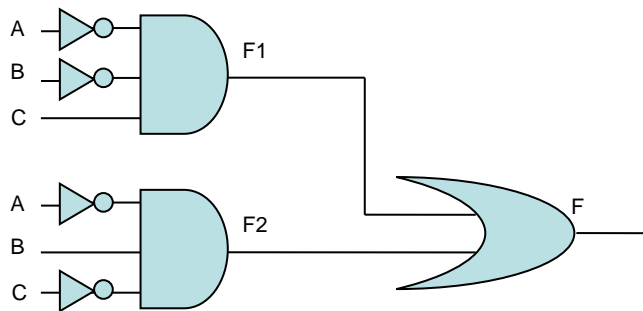
απαντήστε στα εξής:

Να γράψετε τα περιεχόμενα των καταχωρητών R1, R2, και R3 καθώς και της θέσης μνήμης 300 τη στιγμή που ο μετρητής προγράμματος, PC, θα έχει πάρει την τιμή 84 (δηλαδή θα έχει τελειώσει η εκτέλεση του προγράμματος που δίνεται) ξεκινώντας από την τιμή 80 (δηλαδή την αρχή του προγράμματος).

Η πρώτη εντολή στη θέση 80 (που τώρα εκτελείται, ενώ ο PC δείχνει την επόμενη εντολή που θα εκτελεστεί) φορτώνει τον R2 με την τιμή στη θέση μνήμης 301. Άρα, στο κουτάκι του R2 γράφουμε το 11. Ομοίως η επόμενη εντολή στη θέση 81 φορτώνει τον R1 με την τιμή 19. Στη συνέχεια, προστίθενται τα περιεχόμενα των R1, R2 (εντολή ADD στη θέση 82) και το αποτέλεσμα καταχωρείται στον R3 (δηλαδή βάζουμε την τιμή 30 στο κουτάκι του R3). Τέλος, η τιμή του R3 αποθηκεύεται στη θέση μνήμης 300, οπότε στο κουτάκι της θέσης μνήμης 300 βάζουμε το 30.

3.3 Αντιπροσωπευτική άσκηση για λογικά κυκλώματα

Δίνεται το εξής λογικό κύκλωμα:



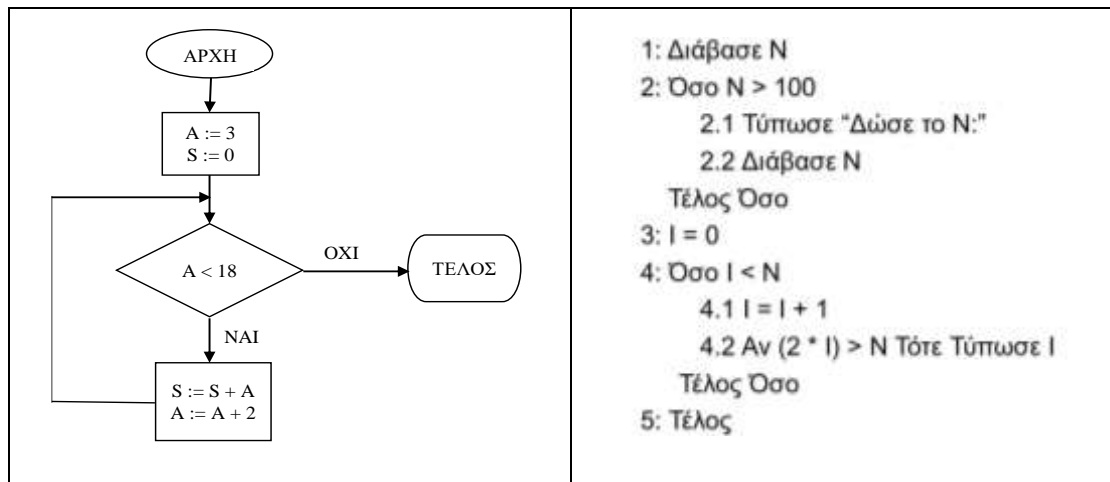
Να συμπληρώσετε τον πιο κάτω πίνακα αληθείας, για όλους τους συνδυασμούς των εισόδων (των A, B, και C δηλαδή) που δίνονται στον πίνακα αυτό:

A	B	C	A'	B'	C'	F1	F2	F
0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	1	0	1
0	1	0	1	0	1	0	1	1
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	0	0

Απλά εφαρμόσαμε τους ορισμούς των πυλών NOT, AND, OR (δείτε τις διαφάνειες του μαθήματος).

3.4 Αντιπροσωπευτική άσκηση για Διάγραμμα Ροής Προγράμματος και Ψευδοκώδικα

Με βάση το Διάγραμμα Ροής Προγράμματος στα αριστερά και τον Ψευδοκώδικα στα δεξιά, απαντήστε στα ερωτήματα που ακολουθούν:



Ερώτημα για το Διάγραμμα Ροής Προγράμματος:

Ποια θα είναι η τιμή των A και S στο σημείο «ΤΕΛΟΣ» του διαγράμματος (όταν τελειώσει, δηλαδή, ο υπολογισμός τον οποίο αναπαριστά το διάγραμμα);

Από τον ορισμό του παραλληλογράμμου, τα A, S αρχικοποιούνται στις τιμές 3 και 0 αντίστοιχα. Στη συνέχεια, ο ρόμβος σχηματίζει ένα βρόχο που εκτελείται ενόσω το A είναι μικρότερο από το 18. Σε κάθε επανάληψη, το τρέχον A προστίθεται στο S και μετά, το A, αυξάνεται κατά 2. Συνεπώς, εφ' όσον το A ξεκίνησε από το 3, στο S θα προστεθούν οι τιμές 3 (αρχική τιμή του A), 5, 7, 9, 11, 13, 15, 17. Στην τελευταία επανάληψη, το A από 17 θα γίνει 19 οπότε, από τον έλεγχο στον ρόμβο, θα σταματήσουν οι επαναλήψεις. Συνεπώς, η τελική τιμή του A είναι 19 και του S είναι $3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 = 80$.

Ερώτημα για τον Ψευδοκώδικα:

Αν διαβαστεί η τιμή 8 για το N, ποια θα είναι η τιμή του I στο τέλος του ψευδοκώδικα (δηλαδή όταν φτάσει ο ψευδοκώδικας στη γραμμή 5); Πόσες τιμές θα εκτυπωθούν, συνολικά, στη γραμμή 4.2 του ψευδοκώδικα;

Ο βρόχος στις γραμμές 2 έως 3 εξασφαλίζει ότι η τιμή 8 που διαβάστηκε είναι αποδεκτή. Ο δεύτερος βρόχος επαναλαμβάνεται ενόσω το I παραμένει μικρότερο από το N, έχοντας ξεκινήσει από την τιμή $I = 0$. Κάθε φορά το I αυξάνεται κατά 1 οπότε, όταν ο βρόχος τερματιστεί, το I θα έχει την τιμή 8 (τότε, για $I = 8$ και $N = 8$ η συνθήκη $I < N$, δηλαδή $8 < 8$, είναι ψευδής). Άρα, οι τιμές του I εντός του βρόχου θα είναι οι εξής:

1, 2, 3, 4, 5, 6, 7, 8

Οι τιμές του $2*I$ είναι οι εξής:

2, 4, 6, 8, 10, 12, 14, 16.

Από αυτές τις τιμές, τη συνθήκη $2*I > N$, δηλαδή $2*I > 8$, ικανοποιούν μόνο οι 10, 12, 14 και 16 που αντιστοιχούν σε τιμές του I ίσες με 5, 6, 7, και 8. Οπότε, τυπώνονται 4 τιμές (οι 5, 6, 7, και 8) από το βρόχο αυτό.

3.5 Αντιπροσωπευτικές ασκήσεις για τη γλώσσα προγραμματισμού Java

Ακολουθούν μερικά ενδεικτικά ερωτήματα για τη γλώσσα προγραμματισμού Java. Η σωστή απάντηση φαίνεται με το γκρι χρώμα. Θα τις συζητήσουμε και στο μάθημα όμως προσπαθήστε να αιτιολογήσετε την σωστή απάντηση που σας δίνεται.

1) Ποια θα είναι η τιμή του x μετά την εκτέλεση του πιο κάτω κώδικα;

```
x = 0;

if (x >= 0)
    x = x+1;
else
    if (x >= 1)
        x = x+2;
```

A	b	c	d	e
0	1	2	3	4

2) Ποια θα είναι η τιμή του x μετά την εκτέλεση του πιο κάτω κώδικα;

```
x = 0;

if (x >= 0)
    x = x+1;

if (x >= 1)
    x = x+2;
```

a	b	c	d	e
0	1	2	3	4

3) Εάν η μεταβλητή x έχει δηλωθεί σαν `int`, τότε εκτυπώνει το μήνυμα "Test B" ο παρακάτω κώδικας.


```

if ( x > 4 ) System.out.println("Test A");
else
    if ( x > 9 ) System.out.println("Test B");
    else System.out.println("Test C");

```

a	b	c	d	e
Για τιμές $x > 9$.	Για τιμές $x < 4$.	Για τιμές $9 > x > 4$.	Πάντα	Ποτέ

4) Ποια τιμή θα τυπωθεί όταν εκτελεστεί ο πιο κάτω κώδικας;

```

int n = 4;
int g = 1;

for (i=1; i < n; i = i + 1)
    g = 2 * g ;

System.out.println(g) ;

```

a	b	c	d	e
1	2	4	6	8

5) Τι εκτυπώνει το παρακάτω πρόγραμμα.

```

int i = 5, sum = 0 ;

do
{
    sum = sum + i ;
    i = i - 1 ;
} while (i > 0) ;

System.out.println(sum);

```

A	b	c	D	e
10	15	20	25	30

6) Ποιο είναι το αποτέλεσμα του παρακάτω κώδικα.

```

int i = 5, sum = 0, rem ;

while (i < 10)
{
    rem = i % 2 ; /* Εδώ λαμβάνεται το υπόλοιπο της διαίρεσης με το 2. */

    if (rem == 0) /* Με το “==” ελέγχεται η ισότητα μεταξύ των δύο μερών. */
        sum = sum + i ;

    i = i + 1 ;
}

```

```
System.out.println(sum);
```

a	b	c	d	e
4	6	10	14	18

4 Κεφάλαιο – Ασφάλεια Πληροφοριακών Συστημάτων

Στα πληροφοριακά συστήματα, στα ολοκληρωμένα πληροφοριακά συστήματα, στα συστήματα ηλεκτρονικής διακυβέρνησης (κρατικά πληροφοριακά συστήματα) καθώς και σε οποιαδήποτε άλλα συστήματα που εμπλέκονται άνθρωποι, διαδικασίες, ροή δεδομένων και πληροφοριών η συνεισφορά της ασφάλειας παίζει καθοριστικό ρόλο. Ήδη στις ημέρες μας αποτελεί γνωστικό πεδίο της επιστήμης της πληροφορικής λαμβάνοντας υπόψη στοιχεία και θεμελιώσεις από άλλες παραπλήσιες ή και μη επιστήμες. Παραπλήσιο γνωστικό πεδίο με μεγάλη συνεισφορά είναι η εφαρμοσμένη Κρυπτογραφία η οποία και αναλύεται σε επόμενες ενότητες. Εισαγωγικά θα μπορούσαμε να ορίσουμε την ασφάλεια ως την προστασία των υπολογιστικών συστημάτων και των δικτύων που διασυνδέονται με παράλληλη αποτροπή οποιασδήποτε μη εξουσιοδοτημένης πρόσβασης στη χρήση τους.

Ιστορικά η ασφάλεια των πληροφοριακών συστημάτων μελετήθηκε στις αρχές της δεκαετίας τους 1970 από το στρατό. Πιο συγκεκριμένα, εξετάστηκε το πρόβλημα χρήσης και επικοινωνίας των υπολογιστών εξ αποστάσεως. Προηγουμένως η πρόσβαση σε υπηρεσίες αφορούσε τη φυσική παρουσία του χρήστη στον κεντρικό υπολογιστή. Από τις σχετικές μελέτες προέκυψαν διάφορες καινοτόμες ιδέες (ειδικά για εκείνη την εποχή) οι οποίες αποτελούν μέχρι και σήμερα θεμελιώδη ζητήματα στην εφαρμογή των πολιτικών ασφαλείας. Μία τέτοια ιδέα ήταν η εύρεση κατάλληλης ισορροπίας μεταξύ της προστασίας των πληροφοριών και της ευκολίας στην εργασία

του χρήστη. Εκείνη τη δεκαετία εμφανίστηκε και ο πρώτο ιός με ονομασία Creeper στο στρατιωτικό δίκτυο ARPANET. Στη συγκεκριμένη δημοσίευση γίνεται επίσης λόγος για ύπαρξη κωδικών ασφαλείας για όλους τους χρήστες των πληροφοριακών συστημάτων και υποσυστημάτων μια ιδέα που υιοθετείται πλήρως αργότερα από τους δημιουργούς του Unix¹. Βέβαια ένα από τα πιο σημαντικά ζητήματα που αντιμετωπίζει η επιστημονική κοινότητα και δη αυτής της κρυπτογραφίας είναι η χρήση αδύναμων κωδικών πρόσβασης. Η ευκολία χρήσης υπαγορεύει αδύναμους κωδικούς ενώ η προστασία της πληροφορίας υπαγορεύει μεγάλους και δύσκολους κωδικούς πρόσβασης. Με λίγα λόγια οδηγούμαστε στον κλονισμό της ισορροπίας που αναφέρθηκε παραπάνω.

Κατά το σχεδιασμό ασφαλών πολιτικών στα πληροφοριακά συστήματα ενσωματώνονται διαδικασίες όχι μόνο τεχνικής φύσης αλλά κυρίως διοικητικές οι οποίες και θα πρέπει να συμβαδίζουν κάθε φορά με τις τρέχουσες ηθικές και κοινωνικές αντιλήψεις. Οι πολιτικές αυτές έχουν μοναδικό στόχο την προφύλαξη του πληροφοριακού συστήματος αλλά και ολόκληρου του οργανισμού από κάθε σκόπιμη ή τυχαία απειλή. Βασικό σημείο κατά τον παραπάνω σχεδιασμό είναι ο εντοπισμός και η κατηγοριοποίηση των πληροφοριών σε εμπιστευτικές ή μη εμπιστευτικές και οι οποίες πρόκειται να χρησιμοποιηθούν στο ευρύ ή σε ένα πιο συγκεκριμένο κοινό. Όλα τα παραπάνω προϋποθέτουν την ύπαρξη μίας ευρύτερης αντίληψης και βασικών αρχών που θα πρέπει να διέπουν τους σχεδιαστικούς στόχους των πληροφοριακών συστημάτων. Ειδικότερα, κάθε αντικείμενο του συστήματος θα πρέπει να αναγνωρίζεται και να αποτυπώνεται σε αυτό μία ένδειξη του βαθμού εμπιστευτικότητας.

Η πολιτική ασφάλειας που διέπει τον οργανισμό θα πρέπει να μεριμνά αποτελεσματικά για όλους τους εμπλεκόμενους στη χρήση και τη λειτουργία του Πληροφοριακού Συστήματος. Οι εμπλεκόμενοι συνήθως είναι οι χρήστες και διαχειριστές, τα διευθυντικά στελέχη και οι πελάτες του οργανισμού. Επιπρόσθετα οι νομικές και κανονιστικές διατάξεις του οργανισμού θα πρέπει να είναι ρητώς διατυπωμένες ούτως ώστε όλοι οι παραπάνω εμπλεκόμενοι να κινούνται σε συγκεκριμένο πλαίσιο αρμοδιοτήτων.

Όταν εφαρμόζεται μία πολιτική ασφάλειας είθισται να συμπεριλαμβάνουμε τα εξής:

¹ Το Unix είναι λειτουργικό σύστημα Ηλεκτρονικών Υπολογιστών, το οποίο αναπτύχθηκε κατά τις δεκαετίες του 1960 και του 1970 από τους Ken Thompson, Dennis Ritchie και Douglas McIlroy.

- Εφαρμογή πληρότητας όσον αφορά τις οδηγίες και τα μέτρα προστασίας των υπηρεσιών που προσφέρουμε και των λειτουργιών που εκτελούνται.
- Εφαρμογή επικαιρότητας όσον αφορά τις τρέχουσες τεχνολογικές εξελίξεις
- Εφαρμογή γενικευσιμότητας όσον αφορά τις επιπρόσθετες παρεμβάσεις, τροποποιήσεις ή προσθήκες που μπορεί να γίνουν στο πληροφοριακό σύστημα.

Η επιτυχία μιας πολιτικής ασφάλειας εξαρτάται, επίσης, και από τη διάθεση τήρησής της και τη συμμετοχή των εμπλεκόμενων φορέων. Με λίγα λόγια θα πρέπει να υπάρχει συνεχής και αδιάλειπτη εκπαίδευση των χρηστών, να αξιολογείται η πρόσβαση, η αμεσότητα και η ευκολία αυτής και να διενεργούνται τακτικά τεστ αντοχής (stress tests) ώστε να διαπιστώνεται το επίπεδο ετοιμότητας για την αντιμετώπιση και εξουδετέρωση οποιασδήποτε απειλής.

Οι βασικοί άξονες στους οποίους στηρίζεται η ασφάλεια των πληροφοριακών συστημάτων είναι οι παρακάτω:

- **Ακεραιότητα (Integrity):** Είναι πολύ σημαντικό τα δεδομένα του πληροφοριακού συστήματος να διατηρούνται σε μία γνώριμη κατάσταση, αναλλοίωτα, χωρίς να υποστούν οποιουδήποτε είδους τροποποίηση, αφαίρεση ή προσθήκη από άτομα εντός και εκτός του οργανισμού. Συνεπώς θα πρέπει να αποτρέπεται αυστηρώς η πρόσβαση στη χρήση μη εξουσιοδοτημένων ατόμων. *Παράδειγμα ακεραιότητας θα μπορούσε να αποτελέσει ένας εκδοτικός οργανισμός μέσω μαζικής ενημέρωσης όπου θα πρέπει τα άρθρα που δημοσιεύονται στο διαδίκτυο να είναι ασφαλή από οποιαδήποτε τροποποίηση και εισαγωγή λανθασμένων πληροφοριών.*
- **Διαθεσιμότητα (Availability):** Τα δεδομένα, τα πληροφοριακά υποσυστήματα, τα δίκτυα και οι υπολογιστικοί πόροι θα πρέπει να είναι στη διάθεση των χρηστών όποτε και για όσο απαιτείται η χρήση τους. Τα τελευταία χρόνια και εξαιτίας της αύξησης της χρήσης των διαδικτυακών υπηρεσιών συχνά παρατηρούνται φαινόμενα μη διαθεσιμότητας δεδομένων. *Οι πιο συχνές περιπτώσεις είναι επιθέσεις τύπου DDOS attack. Τέτοιες επιθέσεις έχουν στόχο την υπερφόρτωση των υπολογιστικών πόρων (από τους επιτιθέμενους) με σκοπό να τεθούν εκτός λειτουργίας διάφορες*

υπηρεσίες ενός οργανισμού. Βέβαια υπο φυσιολογικές συνθήκες παρατηρείται και το φαινόμενο *Slashdot* όπου εξαιτίας της υπερφόρτωσης από υψηλή επισκεψιμότητα και χαμηλού *transfer bandwidth* ενός διαδικτυακού ιστότοπου οι υπηρεσίες βρίσκονται προσωρινά εκτός λειτουργίας. Σε κάθε περίπτωση όμως (κακόβουλη και μη) η διαθεσιμότητα παραβιάζεται και θα πρέπει να ληφθούν άμεσα μέτρα και νέες πολιτικές ασφάλειας.

- **Εμπιστευτικότητα (Privacy ή Confidentiality):** Κάθε πληροφορία που ανταλλάσσεται στο σύστημα και ειδικά αν είναι ευαίσθητη, δεν θα πρέπει να αποκαλύπτεται σε μη εξουσιοδοτημένα άτομα. Απώλεια πληροφοριών θα μπορούσε να γίνει από τις πιο παραδοσιακές μεθόδους μέχρι τις πιο προηγμένες (πχ ψηφιακές υποκλοπές). *Μία παραδοσιακή μέθοδος θα ήταν η φυσική κλοπή hardware από το κατάλληλο τμήμα μίας εταιρείας.*
- **Πιστοποίηση και Αυθεντικότητα (authentication):** Στην περίπτωση αυτή υπάρχουν δύο βασικοί πυλώνες στους οποίους στηρίζεται η πιστοποίηση: την πιστοποίηση οντοτήτων (entity authentication ή identification) και την πιστοποίηση δεδομένων (data authentication). Με τον πρώτο πυλώνα εξασφαλίζουμε ότι κάθε οντότητα στο πληροφοριακό-επικοινωνιακό σύστημα είναι αυτή που ισχυρίζεται και δεν υπάρχει κάποια πλαστοπροσωπία. Με το δεύτερο πυλώνα εξασφαλίζουμε την τοποθεσία, τη χρονική στιγμή και το είδος των μηνυμάτων που ανταλλάσσονται σε μία επικοινωνία. *Παράδειγμα αυθεντικότητας αποτελεί η κάθε είδους εισβολή και υποκλοπή προσωπικών στοιχείων της πιστωτικής μας κάρτας κατά τη διαδικασία μίας διαδικτυακής αγοράς. Τόσο ο οργανισμός θα πρέπει να κατανοεί ότι η αγορά γίνεται από πιστοποιημένη οντότητα όσο και ο αγοραστής να είναι σε θέση να επιβεβαιώσει ότι η αγορά έγινε από τον ίδιο, σε συγκεκριμένη χρονική στιγμή και σε συγκεκριμένη τοποθεσία.*
- **Μη αποποίηση (non-repudiation):** Κανένας χρήστης, είτε αυτός είναι αποστολέας είτε παραλήπτης, δεν μπορεί να αποποιηθεί τις πράξεις που έκανε στο παρελθόν. Ένας από τους πιο διαδεδομένους αλλά και ταυτόχρονα αποτελεσματικούς τρόπους εφαρμογής της μη αποποίησης είναι η χρήση των ψηφιακών υπογραφών και από τις δύο πλευρές.

- **Εγκυρότητα (validity):** Αφορά κυρίως εκείνους που χρησιμοποιούν συγκεκριμένες πληροφορίες του συστήματος. Οι εμπλεκόμενες πλευρές ενδιαφέρονται για την πληρότητα και την ακρίβειά αυτών των πληροφοριών, λόγω των αποφάσεων που πρέπει να λάβουν. Θα μπορούσαμε να εκλάβουμε την Εγκυρότητα ως το άθροισμα της Ακεραιότητας και τη Αυθεντικότητας
- **Μοναδικότητα (Uniqueness):** Είναι η αδυναμία αντιγραφής και αναπαραγωγής της πληροφορίας χωρίς εξουσιοδότηση.

Πριν περάσουμε στο βασικό σκέλος της εργασίας μας και στην προστασία της ιδιωτικότητας αξίζει να αναφερθούμε σε ένα πολύ γνωστό επιστημονικό πεδίο, αυτό της Κρυπτογραφίας το οποίο διαδραματίζει αρκετά σημαντικό ρόλο στην επίτευξη της ασφάλειας των πληροφοριακών συστημάτων αλλά και των εμπλεκόμενων χρηστών.

4.1 Κρυπτογραφία

Ένας από τους πιο σημαντικούς κλάδους της επιστήμης των υπολογιστών είναι η κρυπτογραφία². Με τη λέξη κρυπτογραφία εννοούμε την επιστήμη εκείνη που ασχολείται με τη σχεδίαση κρυπτογραφικών συστημάτων. Πρόκειται για συστήματα τα οποία μετατρέπουν χρήσιμη και εμπιστευτική πληροφορία σε τέτοια μορφή, η οποία να είναι κατανοητή μόνο από τον αποστολέα στον παραλήπτη, ενώ ταυτόχρονα να είναι άχρηστη και ασήμαντη σε οποιοδήποτε μη εξουσιοδοτημένο πρόσωπο. Η κρυπτογραφία, λοιπόν δημιουργήθηκε από την ανάγκη των ανθρώπων να μεταφέρουν μηνύματα ο ένας στον άλλο, χωρίς κάποιος τρίτος να μπορεί να τα αντιληφθεί. Έτσι, οι κύριοι χαρακτήρες που χρησιμοποιούνται διεθνώς στη βιβλιογραφία για την περιγραφή τέτοιων συστημάτων είναι η Alice και ο Bob που επιθυμούν να επικοινωνούν μεταξύ τους, χωρίς η Eve (το τρίτο πρόσωπο) να μπορεί να τους αντιληφθεί.

Η κρυπτογραφία εμφανίστηκε αρχικά, τουλάχιστον από όσα γνωρίζουμε μέχρι σήμερα, στην εποχή των αρχαίων Αιγυπτίων (2.000 π.Χ.), παίζοντας ένα πολύ σημαντικό ρόλο καθ' όλη τη διάρκεια της ιστορίας. Ειδικότερα στον 2^ο παγκόσμιο

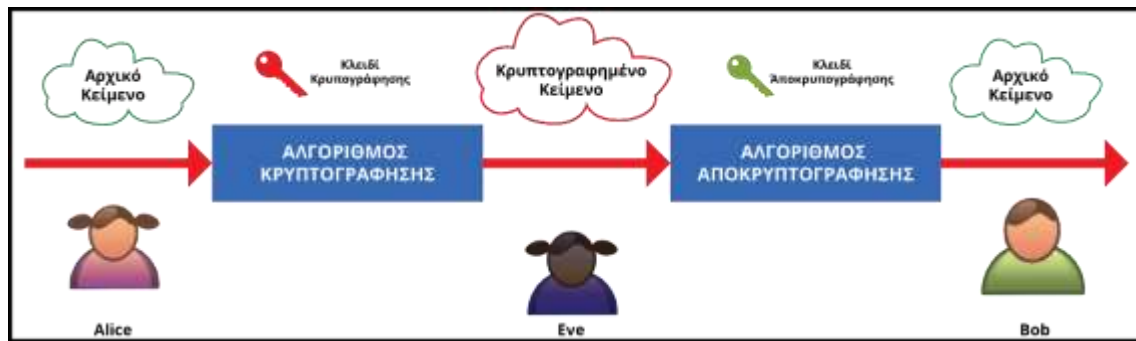
² Η ενότητα της κρυπτογραφίας βασίστηκε στις πανεπιστημιακές σημειώσεις του μαθήματος Κρυπτογραφίας που διεξάγεται στο Πανεπιστήμιου Αιγαίου καθώς και στις παραδόσεις του ίδιου μαθήματος που διεξήχθη το έτος 2012 στο Πανεπιστήμιου Πατρών και στο Δ.Π.Μ.Σ Μαθηματικά των Υπολογιστών και των Αποφάσεων με διδάσκων τον Δρ. Μελετίου Γεράσιμο.

πόλεμο, τόσο οι Γερμανοί όσο και οι Ιάπωνες κατάφεραν να αναπτύξουν κρυπτογραφικές μηχανές όπως η Enigma και η Purple Machine αντίστοιχα. Αξίζει να σημειωθεί ότι αργότερα πολύπλοκες κρυπτογραφικές μέθοδοι επικοινωνίας αναπτύχθηκαν από λαθρεμπόρους, αλλά και στη συνέχεια ακόμη πιο πολύπλοκες αναπτύχθηκαν από μυστικές υπηρεσίες κυβερνήσεων. Τέλος, στις μέρες μας, η ανάγκη για κρυπτογραφημένα μηνύματα είναι πολύ μεγάλη, αρκεί μόνο να συλλογιστούμε πόσο απαραίτητο μας είναι πολλές φορές το γεγονός να ανταλλάσσουμε πληροφορίες μέσω διαδικτύου. Για παράδειγμα, η μεταφορά χρηματικών ποσών από ένα λογαριασμό σε έναν άλλο, που πλέον γίνεται εύκολα μέσω διαδικτύου, επιθυμούμε να γίνεται με τέτοιο τρόπο ώστε να είναι αδύνατο οι δύο αριθμοί λογαριασμών να γίνουν γνωστοί σε κάποιον τρίτο.

Κάθε κρυπτογραφικό σύστημα θα πρέπει να έχει τέτοιες ιδιότητες, ώστε να παρέχεται η επιθυμητή ασφάλεια στους χρήστες, όπως αυτές αναφέρονται στην ενότητα 1.1.2. Τη σημαντικότητα των συγκεκριμένων ιδιοτήτων θα προσπαθήσουμε να παρουσιάσουμε στο παρακάτω παράδειγμα.

Έστω ότι η Alice επιθυμεί να μεταφέρει χρήματα από έναν δικό της τραπεζικό λογαριασμό σε έναν τραπεζικό λογαριασμό του Bob μέσω διαδικτύου. Η Alice, φυσικά και επιθυμεί, η συγκεκριμένη επικοινωνία να είναι εμπιστευτική, μιας και κανένας τρίτος δεν πρέπει να λάβει γνώση τόσων των δύο αριθμών λογαριασμού όσο και του μεταφερόμενου χρηματικού ποσού. Από την άλλη πλευρά, ο Bob θα πρέπει να λάβει το μήνυμα πιστοποιημένο, ότι δηλαδή αυτό προέρχεται πράγματι από την Alice. Και οι δύο πλευρές (Alice και Bob) θα πρέπει να είναι βέβαιες ότι η ακεραιότητα του μηνύματος διατηρείται, όπως για παράδειγμα ότι το μεταφερόμενο χρηματικό ποσό δε δύναται να αλλαχθεί από κάποιον τρίτο κατά τη διαδικασία. Τέλος, ο Bob επιθυμεί την αποποίηση, δηλαδή η Alice αργότερα να μην μπορεί να ισχυριστεί ότι δεν έδωσε την εντολή μεταφοράς του χρηματικού ποσού.

Έτσι, κρυπτογράφηση ονομάζεται η διαδικασία εκείνη κατά την οποία ένα απλό κείμενο (plain text) μετατρέπεται σε ένα κρυπτογραφημένο κείμενο (cipher text). Ενώ αποκρυπτογράφηση είναι η ακριβώς αντίστροφη διαδικασία. Και οι δύο αυτές διαδικασίες πραγματοποιούνται με τη χρήση κλειδιών κρυπτογράφησης και αποκρυπτογράφησης αντίστοιχα. Η γενική μορφή ενός κρυπτογραφικού συστήματος μετάδοσης μηνύματος παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 1. 1 Γενική μορφή ενός κρυπτογραφικού συστήματος μετάδοσης μηνύματος.

Όπως φαίνεται και στο πιο πάνω σχήμα, η Alice κρυπτογραφεί ένα μήνυμα χρησιμοποιώντας έναν αλγόριθμο κρυπτογράφησης και το αντίστοιχο κλειδί κρυπτογράφησης και στη συνέχεια στέλνει το κρυπτογραφημένο πια μήνυμα στον Bob. Εκείνος με τη σειρά του το αποκρυπτογραφεί χρησιμοποιώντας έναν αλγόριθμο αποκρυπτογράφησης και το κλειδί αποκρυπτογράφησης. Η Eve είναι μια οποιαδήποτε οντότητα (τρίτο πρόσωπο) η οποία προσπαθεί να παρέμβει με κακόβουλο τρόπο στην επικοινωνία μεταξύ της Alice και του Bob. Αυτή η οντότητα (τρίτο πρόσωπο) στην κρυπτογραφία αποκαλείται υποκλοπέας (interceptor), ωτακουστής (eavesdropper) ή απλά επιτιθέμενος (attacker). Η διαδικασία που εκτελεί ο υποκλοπέας ονομάζεται κρυπτανάλυση. Συνεπώς, κρυπτανάλυση είναι η διαδικασία εκείνη κατά την οποία ο υποκλοπέας μελετά το κρυπτογραφημένο μήνυμα και προσπαθεί με διάφορες τεχνικές και μεθόδους και χωρίς φυσικά να διαθέτει το κλειδί αποκρυπτογράφησης, να εξάγει την αρχική πληροφορία (ή μέρος αυτής) του κρυπτογραφημένου μηνύματος. Η έννοια της κρυπτολογίας περιλαμβάνει μαζί την έννοια της κρυπτογραφίας και της κρυπτανάλυσης.

Πολύ συχνά στα κρυπτογραφημένα μηνύματα ορίζεται ένα Έμπιστο Τρίτο Μέλος (Trusted Third Party - TTP). Ο ρόλος του μέλους αυτού είναι η επίλυση διαφωνιών μεταξύ των υπολοίπων χρηστών του συστήματος ή ακόμα και η διευκόλυνση της μεταξύ τους επικοινωνίας. Το Έμπιστο Τρίτο Μέλος διακρίνεται σε λειτουργικά έμπιστο (functionally trusted) και σε άνευ όρων έμπιστο (unconditionally trusted). Λειτουργικά έμπιστο μέλος θεωρείται ένα TTP το οποίο είναι έμπιστο για να λύσει διαφωνίες μεταξύ των χρηστών, αλλά δεν μπορεί να έχει πρόσβαση στα μυστικά κλειδιά τους. Ένα TTP μέλος θεωρείται άνευ όρων έμπιστο μέλος μόνον εφόσον όλοι οι χρήστες του κρυπτογραφημένου συστήματος μπορούν να το εμπιστευθούν από κάθε άποψη, όπως για παράδειγμα να είναι γνώστης όλων των μυστικών κλειδιών των χρηστών.

Τέλος, αξίζει να σημειωθεί ότι η ασφάλεια του συστήματος εξαρτάται μόνο από την μυστικότητα των κλειδιών αποκρυπτογράφησης, μιας και οι αλγόριθμοι κρυπτογράφησης και αποκρυπτογράφησης που χρησιμοποιούνται είναι γνωστοί σε όλους. Από αυτό και μόνο το γεγονός, είναι φανερό ότι η διανομή και διαχείριση των κλειδιών αποκρυπτογράφησης αποτελεί ένα από τα πιο βασικά και δύσκολα προβλήματα της επιστήμης της κρυπτογραφίας.

4.1.1 Συμμετρικά και Μη Συμμετρικά Κρυπτογραφικά Συστήματα

Τα κρυπτογραφικά συστήματα διαχωρίζονται σε δύο είδη με βάση τα κλειδιά αποκρυπτογράφησης. Έτσι έχουμε:

- τα συμμετρικά (symmetric) ή συμβατικά (conventional) κρυπτογραφικά συστήματα και
- τα μη συμμετρικά (asymmetric) ή δημοσίου κλειδιού (public key) κρυπτογραφικά συστήματα.

Ένα κρυπτογραφικό σύστημα ονομάζεται συμβατικό όταν το κλειδί αποκρυπτογράφησης του μπορεί εύκολα να εξακριβωθεί από το αντίστοιχο κλειδί κρυπτογράφησης. Στην πληθώρα των περιπτώσεων των συμβατικών κρυπτογραφικών συστημάτων, τα δύο κλειδιά είναι ίδια. Αντίστοιχα, ένα κρυπτογραφικό σύστημα ονομάζεται δημοσίου κλειδιού όταν το κλειδί αποκρυπτογράφησης του εξακριβώνεται δύσκολα από το αντίστοιχο κλειδί κρυπτογράφησης.

Είναι πλέον σαφές ότι η ασφάλεια όλων των κρυπτογραφικών συστημάτων βασίζεται στην μυστικότητα του κλειδιού αποκρυπτογράφησης, το οποίο και δεν πρέπει να αποκαλυφθεί σε κανένα άλλον, πέραν των πιστοποιημένων χρηστών του κρυπτογραφικού συστήματος. Έτσι, για να θεωρείται ένα συμβατικό (συμμετρικό) κρυπτογραφικό σύστημα ασφαλές, να είναι δηλαδή ο διάυλος επικοινωνίας των χρηστών του ασφαλής, θα πρέπει τόσο το κλειδί αποκρυπτογράφησης όσο και το κλειδί κρυπτογράφησης να είναι μυστικά. Αυτό το γεγονός από μόνο του αποτελεί ένα σημαντικό μειονέκτημα των συμβατικών κρυπτογραφικών συστημάτων. Αντίθετα, κάτι τέτοιο δεν απαιτείται για την ασφάλεια των κρυπτογραφικών συστημάτων δημοσίου κλειδιού. Μόνο και μόνο για το λόγο αυτό, η επινόησή τους αποτέλεσε μία επανάσταση στο χώρο της κρυπτογραφίας.

Στα κρυπτογραφικά συστήματα δημοσίου κλειδιού, οι αλγόριθμοι κρυπτογράφησης και αποκρυπτογράφησης είναι με τέτοιο τρόπο σχεδιασμένοι ώστε τα αντίστοιχα κλειδιά, κρυπτογράφησης και αποκρυπτογράφησης, να είναι διαφορετικά μεταξύ τους. Η επιτυχία, και συνεπώς η ασφάλεια, αυτού του είδους των κρυπτογραφικών συστημάτων βασίζεται στο ότι παρά το γεγονός ότι το δημόσιο κλειδί κρυπτογράφησης είναι γνωστό σε όλους, ο υπολογισμός του κλειδιού αποκρυπτογράφησης είναι δύσκολος έως αδύνατος. Στα συστήματα αυτά, δημόσιο κλειδί καλείται το κλειδί κρυπτογράφησης, που είναι σε όλους γνωστό και ιδιωτικό κλειδί καλείται το άλλο κλειδί, το κλειδί αποκρυπτογράφησης.

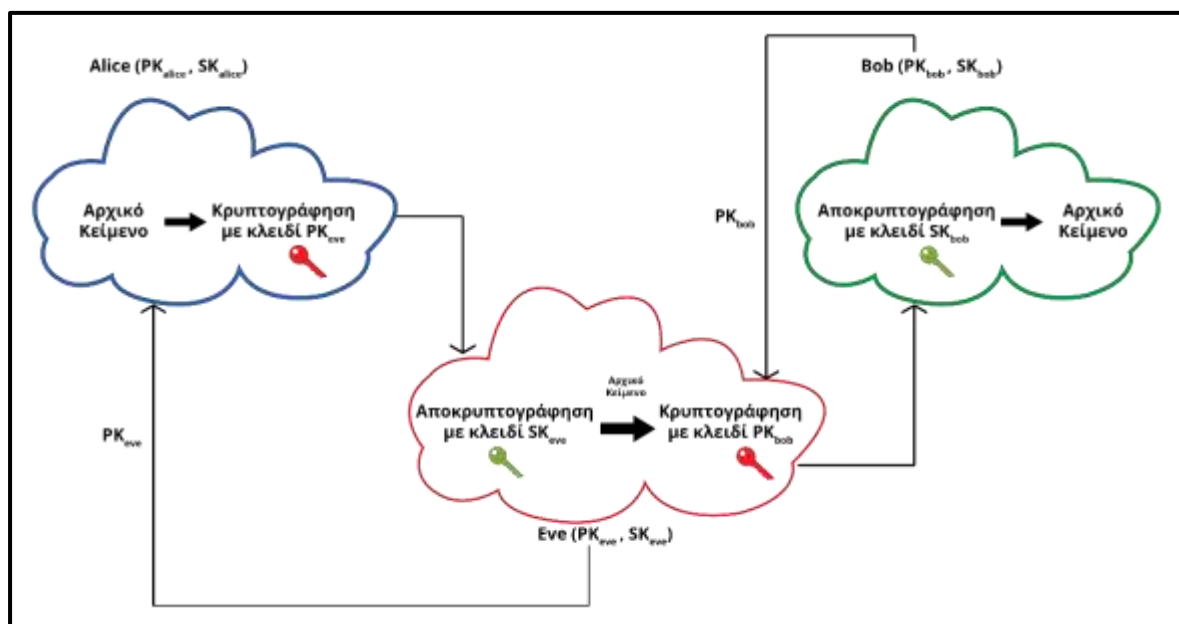
Σύμφωνα με όλα τα παραπάνω, αυτό που προσδίδει ασφάλεια στα κρυπτογραφικά συστήματα δημοσίου κλειδιού είναι στην ουσία η δυσκολία που παρουσιάζεται στην επίλυση ενός μαθηματικού προβλήματος. Πιο συγκεκριμένα, η εξακρίβωση του ιδιωτικού κλειδιού με βάση το δημόσιο κλειδί, αλλά και κάποιες ακόμα παραμέτρους του συστήματος είναι ένα τρομερά δύσκολο υπολογιστικά πρόβλημα. Όσο πιο δυσεπίλυτο το πρόβλημα αυτό, τόσο πιο αποδοτικό και ασφαλές είναι το κρυπτογραφικό σύστημα.

Ας προσπαθήσουμε όμως μέσω ενός απλού παραδείγματος να κατανοήσουμε τη λειτουργία των κρυπτογραφικών συστημάτων δημοσίου κλειδιού. Έστω ότι η Alice θέλει να μεταφέρει ένα μήνυμα στον Bob. Η Alice και ο Bob έχουν από ένα δημόσιο και ένα ιδιωτικό κλειδί ο καθένας. Χρησιμοποιώντας η Alice το δημόσιο κλειδί του Bob μεταφέρει το μήνυμα στον Bob, ο οποίος με τη σειρά του, χρησιμοποιώντας το ιδιωτικό του κλειδί αποκρυπτογραφεί το μήνυμα που του έστειλε η Alice. Στη συνέχεια απαντάει με ένα ακόμα κρυπτογραφημένο μήνυμα στην Alice, χρησιμοποιώντας το δημόσιο κλειδί της Alice. Στο σημείο αυτό, πρέπει να αναφερθεί ότι όποιος γνωρίζει το δημόσιο κλειδί της Alice, μπορεί να της στείλει ένα κρυπτογραφημένο μήνυμα. Και αντίστοιχα ισχύουν και για τον Bob. Η Alice με τη σειρά της, μπορεί να αποκρυπτογραφήσει τα μηνύματα που λαμβάνει χρησιμοποιώντας το ιδιωτικό της κλειδί.

Διαβάζοντας τα παραπάνω, θα έλεγε κανείς ότι τα κρυπτογραφικά συστήματα δημοσίου κλειδιού είναι τα πλέον ιδανικά. Δυστυχώς όμως κάτι τέτοιο δεν ισχύει. Στο πιο κάτω σχήμα, Σχήμα 1.3, παρουσιάζεται ο τρόπος με τον οποίο ένας κακόβουλος χρήστης μπορεί να επιτεθεί σε ένα κρυπτογραφικό σύστημα δημοσίου κλειδιού.

Επανερχόμενοι στο πιο πάνω παράδειγμα κρυπτογραφικού συστήματος δημοσίου κλειδιού, ας κάνουμε τώρα την υπόθεση ότι ο κακόβουλος χρήστης Eve θέλει

να ανακτήσει το μήνυμα της Alice προς τον Bob. Έτσι, εισβάλλει στην μεταξύ τους επικοινωνία παραπλανώντας τους, ενώ προσποιείται στην Alice ότι είναι ο Bob και στον Bob ότι είναι η Alice. Έτσι, η Eve κοινοποιεί στην Alice το δημόσιο κλειδί της, η οποία (Alice) θεωρεί ότι το δημόσιο κλειδί που της κοινοποιείται είναι του Bob και κρυπτογραφεί ένα μήνυμα. Το μήνυμα αυτό λαμβάνει η Eve, η οποία χρησιμοποιώντας το ιδιωτικό της κλειδί αποκρυπτογραφεί το μήνυμα της Alice κι έτσι το ανακτά. Στη συνέχεια, η Eve κρυπτογραφεί το μήνυμα της Alice, το οποίο έχει ανακτήσει, με το δημόσιο κλειδί του Bob. Αυτός με τη σειρά του λαμβάνει το μήνυμα της Alice, χωρίς όμως να γνωρίζει ότι δεν εστάλει από εκείνη, αλλά από την Eve. Στην προκειμένη δηλαδή περίπτωση, αποστολέας (Alice) και παραλήπτης (Bob) δεν μπορούν να γνωρίζουν τον υποκλοπέα (Eve) που παράκαμψε την επικοινωνία τους και έχει υποκλέψει το αρχικό μήνυμα. Σε άλλη περίπτωση, ο υποκλοπέας θα μπορούσε και να τροποποιήσει το αρχικό μήνυμα.



Σχήμα 1.2 Παρέμβαση τρίτου και υποκλοπή σε ένα κρυπτογραφικό σύστημα δημοσίου κλειδιού.

Το παράδειγμα αυτό κάνει σαφές ότι η πιστοποίηση των οντοτήτων ενός κρυπτογραφικού συστήματος δημοσίου κλειδιού είναι ιδιαίτερα σημαντική.

Οι επιθέσεις που μπορεί να παρατηρηθούν σε ένα κρυπτογραφικό σύστημα, είτε συμβατικό (συμμετρικό) είτε δημοσίου κλειδιού (μη συμμετρικό) διακρίνονται σε:

- Ενεργές επιθέσεις (Active Attacks). Μία ενεργή επίθεση παρουσιάζεται στο Σχήμα 1.3. Πρόκειται για τις επιθέσεις εκείνες όπου ο υποκλοπέας έχει τη

δυνατότητα να διαγράψει, προσθέσει ή να παρέμβει με οποιονδήποτε τρόπο στην επικοινωνία των δύο χρηστών.

- Παθητικές επιθέσεις (Passive Attacks). Τέτοιες επιθέσεις είναι οι ακόλουθες:

1. Επίθεση με γνωστό απλό κείμενο (Known – plain text attack). Κατά την επίθεση αυτή ο υποκλοπέας γνωρίζει κάποια από τα αρχικά κείμενα και τα αντίστοιχα κρυπτογραφημένα κείμενα και χρησιμοποιώντας τα προσπαθεί να ανακαλύψει το κλειδί αποκρυπτογράφησης.
2. Επίθεση με επιλεγμένο απλό κείμενο (Chosen – plain text attack). Σε αυτού του είδους την επίθεση ο υποκλοπέας επιλέγει ορισμένα αρχικά κείμενα και λαμβάνει τα κρυπτογραφημένα τους.
3. Προσαρμοζόμενη επίθεση με επιλεγμένο απλό κείμενο (Adaptive chosen – plain text attack). Είναι επίθεση παρόμοια με την προηγούμενη, με τη διαφορά ότι σε αυτή την περίπτωση ο υποκλοπέας μπορεί κατά τη διάρκεια της επίθεσης να αλλάξει τις επιλογές του για τα αρχικά κείμενα.
4. Επίθεση με επιλεγμένο κρυπτογραφημένο κείμενο (Chosen – cipher text attack). Στην προκειμένη περίπτωση συμβαίνει το αντίθετο από ότι προηγουμένως: ο υποκλοπέας επιλέγει ορισμένα κρυπτογραφημένα κείμενα και του παρέχονται με κάποιο τρόπο και τα αντίστοιχα αρχικά κείμενα.
5. Προσαρμοζόμενη επίθεση με επιλεγμένο κρυπτογραφημένο κείμενο (Adaptive chosen – cipher text attack). Πρόκειται για επίθεση παρόμοια με την προηγούμενη, με τη διαφορά όμως ότι ο υποκλοπέας μπορεί να αλλάξει τις επιλογές των κρυπτογραφημένων κειμένων κατά τη διάρκεια της επίθεσης.
6. Επίθεση στο κρυπτογραφημένο κείμενο (Cipher text – only attack). Σε αυτού του είδους την επίθεση ο υποκλοπέας προσπαθεί να ανακαλύψει το κλειδί αποκρυπτογράφησης ή το αρχικό κείμενο γνωρίζοντας μόνο το κρυπτογραφημένο κείμενο. Είναι προφανές ότι ένα τέτοιο

κρυπτογραφικό σύστημα, ευάλωτο σε τέτοιες επιθέσεις, είναι τελείως ανασφαλές.

Ολοκληρώνοντας, αξίζει να αναφερθεί ότι τόσο τα συμβατικά (συμμετρικά) συστήματα όσο και τα συστήματα δημοσίου κλειδιού (μη συμμετρικά) παρουσιάζουν πλεονεκτήματα και μειονεκτήματα. Συνήθως, τα πλεονεκτήματα των συμμετρικών κρυπτογραφικών συστημάτων είναι μειονεκτήματα των μη συμμετρικών κρυπτογραφικών συστημάτων και αντίστοιχα ισχύουν για τα μειονεκτήματα των πρώτων συστημάτων σε σχέση με αυτά των δεύτερων.

Πιο αναλυτικά:

Πλεονεκτήματα συμβατικών (συμμετρικών) κρυπτογραφικών συστημάτων

- Τα συμμετρικά κρυπτογραφικά συστήματα είναι παλαιότερα των μη συμμετρικών και συνεπώς περισσότερο χρόνο δοκιμασμένα στην πράξη.
- Τα κλειδιά που χρησιμοποιούνται είναι σχετικά μικρά και συνεπώς βρίσκουν πολύ εύκολα εφαρμογή και σε ποιο οικονομικές περιπτώσεις, όπως για παράδειγμα τα κινητά τηλέφωνα.
- Πολύ γρήγορη κρυπτογράφηση (π.χ. εκατοντάδες Mbytes είναι δυνατό να κρυπτογραφηθούν σε δευτερόλεπτα).
- Είναι δυνατή η χρήση τους στις γεννήτριες τυχαίων αριθμών ή συναρτήσεων κατακερματισμού.
- Τέλος, υπάρχει η δυνατότητα να χρησιμοποιηθούν συνδυαστικά μεταξύ τους και να δώσουν ακόμη πιο ισχυρούς συμμετρικούς αλγόριθμους.

Μειονεκτήματα συμβατικών (συμμετρικών) κρυπτογραφικών συστημάτων

- Για την κατά το δυνατό μεγαλύτερη ασφάλεια των συστημάτων αυτών, απαιτείται ένας καλός και ασφαλής δίαυλος επικοινωνίας στην ανταλλαγή των κλειδιών.
- Κάθε κλειδί που χρησιμοποιείται μεταξύ δύο χρηστών θα πρέπει να αντικαθίσταται αρκετά συχνά.

- Ανάμεσα σε δύο χρήστες απαιτείται ένα κλειδί. Συνεπώς σε περίπτωση δικτύου n χρηστών απαιτούνται $\frac{n(n-1)}{2}$ κλειδιά.
- Για να γίνει αποδοτικά η διαχείριση των μυστικών κλειδιών απαιτείται και η ύπαρξη ενός άνευ όρων Έμπιστου Τρίτου Μέλους - TTP.
- Τέλος, σημειώνεται ότι οι ψηφιακές υπογραφές, με τις οποίες θα ασχοληθούμε στη συνέχεια και οι οποίες βασίζονται στα συμμετρικά κρυπτογραφικά συστήματα, απαιτούν είτε την ύπαρξη ενός Έμπιστου Τρίτου Μέλους – TTP είτε την ύπαρξη μεγάλων κλειδιών.

Αντίστοιχα, τα πλεονεκτήματα και τα μειονεκτήματα των κρυπτογραφικών συστημάτων δημόσιου κλειδιού έχουν ως εξής:

Πλεονεκτήματα κρυπτογραφικών συστημάτων δημοσίου κλειδιού (μη συμμετρικά συστήματα)

- Δεν υπάρχει απαίτηση ασφαλούς δίαυλου επικοινωνίας.
- Δε θεωρείται απαραίτητη η αλλαγή των ζευγών των κλειδιών (δημόσιου και ιδιωτικού) ακόμα και για μεγάλο χρονικό διάστημα.
- Σε δίκτυα όπου ο αριθμός των χρηστών είναι n , τα κλειδιά που χρησιμοποιούνται είναι $2n$. Έτσι σε σχέση με τα συμμετρικά κρυπτογραφικά συστήματα πολλών χρηστών, τα μη συμμετρικά απαιτούν πολύ μικρότερο αριθμό κλειδιών.
- Για να γίνει αποδοτική η διαχείριση των κλειδιών στα μη συμμετρικά κρυπτογραφικά συστήματα δεν απαιτείται η ύπαρξη ενός άνευ όρων Έμπιστου Τρίτου Μέλους, αλλά αρκεί η ύπαρξη ενός λειτουργικά Έμπιστου Τρίτου Μέλους.
- Τέλος, σημειώνεται ότι οι αλγόριθμοι δημιουργίας ψηφιακών υπογραφών που βασίζονται σε τεχνικές δημοσίου κλειδιού είναι πολύ πιο αποδοτικοί από τους αντίστοιχους των συμμετρικών αλγορίθμων.

Μειονεκτήματα κρυπτογραφικών συστημάτων δημοσίου κλειδιού (μη συμμετρικά συστήματα)

- Τα συστήματα αυτά είναι κατά πολύ νεότερα των συμμετρικών και δεν έχουν δοκιμαστεί τόσο πολύ.
- Τα κλειδιά που χρησιμοποιούνται (δημόσια και ιδιωτικά) είναι αρκετά μεγάλα.
- Η διαδικασία κρυπτογράφησης – αποκρυπτογράφησης είναι περισσότερο χρονοβόρα από ότι αυτή των συμμετρικών κρυπτογραφικών συστημάτων.
- Η ασφάλειά τους βασίζεται στη θεωρία αριθμών, των οποίων η επίλυση θεωρείται πολύ δύσκολη, πράγμα όμως που ακόμα δεν έχει αποδειχθεί τουλάχιστον θεωρητικά.

Στην πράξη τα συμμετρικά κρυπτογραφικά συστήματα βρίσκουν περισσότερες εφαρμογές, ενώ εκείνα του δημοσίου κλειδιού χρησιμοποιούνται κυρίως για την εγκατάσταση συμμετρικών κλειδιών και για τη δημιουργία ψηφιακών υπογραφών. Έτσι, συνδυάζονται τα πλεονεκτήματα και των δύο περιπτώσεων.

4.1.2 Ψηφιακές Υπογραφές

Οι ψηφιακές υπογραφές (digital signatures), όπως και οι κοινές υπογραφές που χρησιμοποιούνται καθημερινά, αποτελούν ένα αξιόπιστο μέσο που συνδέει μία συγκεκριμένη οντότητα με μία συγκεκριμένη πληροφορία. Οι υπογραφές αυτές είναι στενά συνδεδεμένες με τις έννοιες της πιστοποίησης και της μη αποποίησης.

Κάθε πρωτόκολλο ψηφιακής υπογραφής αποτελείται στην ουσία από δύο αλγόριθμους, αυτόν της δημιουργίας της ψηφιακής υπογραφής και αυτόν της επαλήθευσής της. Ας το δούμε όμως αυτό μέσω ενός παραδείγματος. Έστω ότι η Alice θέλει να υπογράψει ένα μήνυμα m και να το στείλει στον Bob. Η υπογραφή της Alice είναι στην ουσία το αποτέλεσμα ενός αλγορίθμου, μίας συνάρτησης πιο απλά, έστω SA . Η είσοδος στη συνάρτηση είναι το μήνυμα της Alice, m , ενώ η έξοδος αυτής είναι η υπογραφή της Alice και η οποία εξαρτάται με κάποιο συγκεκριμένο τρόπο από μία πληροφορία που έχει στην κατοχή της μόνο η Alice. Η πληροφορία αυτή μπορεί να είναι το ιδιωτικό της κλειδί. Η έξοδος της συνάρτησης είναι: $s = SA(m)$ και είναι η υπογραφή της Alice πάνω στο μήνυμα m προς τον Bob. Με τη σειρά του ο Bob λαμβάνει το ζευγάρι (s, m) και χρησιμοποιώντας μία δική του συνάρτηση –αλγόριθμο– (VA) λαμβάνει ένα αποτέλεσμα έστω $u = VA(s, m)$. Το αποτέλεσμα αυτό μπορεί να πάρει τιμές 0 και 1, κάτι που εξαρτάται από μία δημόσια πληροφορία της Alice, όπως

για παράδειγμα το δημόσιο κλειδί της. Εάν η τιμή της συνάρτησης του Bob ισοδυναμεί με τον αριθμό 1, τότε σημαίνει ότι αυτή είναι η ψηφιακή υπογραφή της Alice και γίνεται δεκτή από τον Bob. Σε αντίθετη περίπτωση αυτή απορρίπτεται.

Είναι πλέον φανερό ότι για να είναι κρυπτογραφικά ασφαλείς οι αλγόριθμοι δημιουργίας της ψηφιακής υπογραφής και επαλήθευσης αυτής, θα πρέπει να ικανοποιούνται οι πιο κάτω προϋποθέσεις:

- Μία ψηφιακή υπογραφή είναι έγκυρη μόνο εφόσον ο αλγόριθμος επαλήθευσης επιστρέφει την τιμή 1.
- Θα πρέπει να είναι υπολογιστικά αδύνατο για οποιονδήποτε χρήστη εκτός του υπογράφοντος, να δημιουργήσει μια ψηφιακή υπογραφή για την οποία το αποτέλεσμα της συνάρτησης VA του παραλήπτη να είναι εσφαλμένα ίσο με την τιμή 1.

Στα πρωτόκολλα ψηφιακών υπογραφών υπάρχει συνήθως ένα λειτουργικά έμπιστο τρίτο μέλος (TTP). Το μέλος αυτό, στο παραπάνω παράδειγμα, θα επέλυε διαφωνίες του τύπου: η Alice αρνείται ότι υπέγραψε κάποιο μήνυμα ή ακόμα και τον ψευδή ισχυρισμό του Bob ότι έλαβε κάποιο μήνυμα από την Alice. Με τις ψηφιακές υπογραφές θα ασχοληθούμε στη συνέχεια.

4.2 Προσωπικά Δεδομένα και Ιδιωτικότητα

Σύμφωνα με την Αρχή Προστασίας Δεδομένων Προσωπικού Χαρακτήρα και το Νόμο 2472/1997 τα **Προσωπικά Δεδομένα ή Δεδομένα προσωπικού χαρακτήρα είναι**³:

“κάθε πληροφορία που αναφέρεται και περιγράφει ένα άτομο εν ζωή, όπως: στοιχεία αναγνώρισης (ονοματεπώνυμο, ηλικία, κατοικία, επάγγελμα, οικογενειακή κατάσταση κλπ.), φυσικά χαρακτηριστικά, εκπαίδευση, εργασία (προϋπηρεσία, εργασιακή συμπεριφορά κλπ), οικονομική κατάσταση (έσοδα, περιουσιακά στοιχεία, οικονομική συμπεριφορά), ενδιαφέροντα, δραστηριότητες, συνήθειες.”

³ Ο ορισμός αυτός αποτελεί αυτούσιο και αναπόσπαστο μέρος από τον ιστότοπο της Αρχής Προστασίας Δεδομένων Προσωπικού Χαρακτήρα <http://www.dpa.gr/> με ημερομηνία ανάκτησης 6/5/2015

Επιπροσθέτως, εκτός από τα Δεδομένα προσωπικού χαρακτήρα υπάρχουν και τα **Ευαίσθητα προσωπικά δεδομένα** τα οποία προστατεύονται από τον Νόμο με αυστηρότερες ρυθμίσεις από ότι τα απλά προσωπικά δεδομένα και ορίζονται ως⁴:

“τα προσωπικά δεδομένα ενός ατόμου που αναφέρονται στη φυλετική ή εθνική του προέλευση, στα πολιτικά του φρονήματα, στις θρησκευτικές ή φιλοσοφικές του πεποιθήσεις, στη συμμετοχή του σε συνδικαλιστική οργάνωση, στην υγεία του, στην κοινωνική του πρόνοια, στην ερωτική του ζωή, τις ποινικές διώξεις και καταδίκες του, καθώς και στη συμμετοχή του σε συναφείς με τα ανωτέρω ενώσεις προσώπων.”

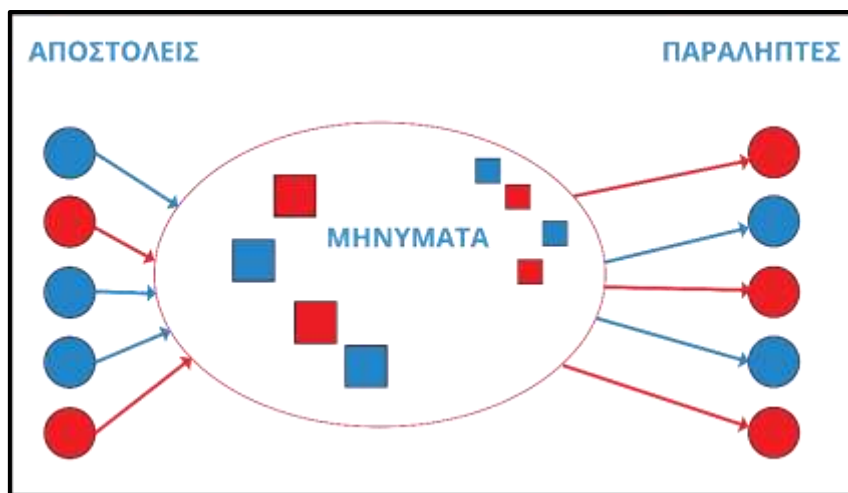
Από τη σκοπιά της πληροφορικής και της επικοινωνίας εντός δικτύου υπολογιστών η οντοτήτων αναφέρεται πολύ συχνά ο όρος **ιδιωτικότητα** και πιο συγκεκριμένα **προστασία της ιδιωτικότητας**. Στη συγκεκριμένη περίπτωση δεν αρκεί η Ελληνική ή αντίστοιχα η Ευρωπαϊκή νομοθεσία⁵ για να προστατέψει τα απλά και τα ευαίσθητα προσωπικά δεδομένα αλλά γίνεται λόγος κυρίως για τις τεχνολογίες που δημιουργήθηκαν για τον παραπάνω σκοπό. Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθούμε σε κάποιους περαιτέρω όρους που συνθέτουν την έννοια της ιδιωτικότητας. Οι όροι είναι βασισμένοι σε ένα **περιβάλλον**, όπου κάποιος χρήστης-αποστολέας στέλνει μηνύματα σε κάποιον ή κάποιους παραλήπτες μέσω ενός δικτύου επικοινωνίας. Εκτός από τις έννοιες του αποστολέα και παραλήπτη, **μια ενεργή οντότητα** μπορεί να είναι φυσικό πρόσωπο, νομικό πρόσωπο, υπολογιστής, ή διεργασία ενώ **ένα σύνολο ενεργών οντοτήτων** μπορεί να είναι ένας οργανισμός που δρα ως νομικό πρόσωπο δομημένο όμως από ενεργές οντότητες. Η έννοια του περιβάλλοντος μπορεί να γίνει ακόμη πιο σαφής με χρήση του συστήματος, το οποίο έχει τις εξής ιδιότητες:

- Το σύστημα έχει ένα εξωτερικό περιβάλλον, δηλ. τμήμα του κόσμου που βρίσκεται εκτός του συστήματος
- Η κατάσταση του συστήματος μπορεί να αλλάξει ανάλογα με τις ενέργειες εντός του συστήματος

⁴ Ομοίως με παραπάνω

⁵ Οδηγίες 95/46/EK, 2002/58/EK, 2006/24/EK, 2009/136/EK

Από την άλλη πλευρά ο επιτιθέμενος χρήστης ενδιαφέρεται για την παρακολούθηση όλων των επικοινωνιών εντός του συστήματος, την καταγραφή των προτύπων επικοινωνίας που χρησιμοποιούνται με τελικό σκοπό τη χειραγώγηση της επικοινωνίας. Ακόμη υποθέτουμε ότι ο επιτιθέμενος χρήστης μπορεί να βρίσκεται εκτός του συστήματος και να παρακολουθεί τις γραμμές επικοινωνίας, ή να βρίσκεται εντός του συστήματος και να ελέγχει σταθμούς εργασίας. Η χειραγώγηση της επικοινωνίας αφορά την αξιοποίηση όλων των παραπάνω πληροφοριών ώστε να καταλήξει στα στοιχεία που ενδιαφέρουν γνωστά και ως *Items of Interest - IOIs*, π.χ. ποιος έστειλε, ποιος παρέλαβε, ποια μηνύματα⁶.



Σχήμα 1. 3 Δίκτυο Επικοινωνίας ανταλλαγής μηνυμάτων με επιτιθέμενους

Όπως αναφέραμε η έννοια της ιδιωτικότητας συντίθεται από αρκετούς όρους και **απαιτήσεις** που εκτός των παραπάνω μερικές επιπρόσθετες αλλά εξίσου σημαντικές είναι οι εξής:

- Ανωνυμία
- Μη-Συνδεσιμότητα
- Ψευδωνυμία
- Διαχείριση Ταυτότητας

4.2.1 Ανωνυμία

⁶ Οι έννοιες και οι ερμηνείες σχετικά με την ιδιωτικότητα αντλήθηκαν (5/6/2015) από τους παρακάτω ηλεκτρονικούς συνδέσμους:

http://www.ct.aegean.gr/people/kalloniatis/Privacy_Intellectual_Property/PRIVACY_Terminology.pdf
http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

Ανωνυμία μιας οντότητας, σημαίνει ότι η οντότητα δεν είναι αναγνωρίσιμη δηλαδή δε διαφαίνονται τα μοναδικά της χαρακτηριστικά μέσα σε ένα σύνολο ανωνύμων οντοτήτων. Η ανωνυμία εξαρτάται από τη γνώση του εκάστοτε επιτιθέμενου και σε γενικές γραμμές εξασφαλίζει τη μη αποκάλυψη της ταυτότητας του χρήστη κατά την πρόσβαση σε δεδομένα ή υπηρεσίες. Βέβαια η ανωνυμία δεν αφορά την προστασία της ταυτότητας μιας οποιασδήποτε οντότητας-χρήστη αλλά προορίζεται για να επιβάλει στις οντότητες-χρήστες να μην μπορούν να ανακτήσουν την ταυτότητα του χρήστη-οντότητας μέσω της σύνδεσής του στο δίκτυο επικοινωνίας. Ο αποστολέας αλλά και ο παραλήπτης μπορεί να παραμένει ανώνυμος μόνον εντός ενός συνόλου αποστολέων και του συνόλου παραληπτών. Ακόμη, θεωρείται ως δεδομένο ότι ο επιτιθέμενος δεν ξεχνά καμία πληροφορία και η γνώση του συνεχώς αυξάνεται.

Ο επιτιθέμενος διατηρεί το δικό του σύνολο ανώνυμων οντοτήτων και στην περίπτωση όπου νέες οντότητες εισέρχονται στο σύστημα μετά από τον επιτιθέμενο, αυτές δε θα ανήκουν στο σύνολο ανωνύμων οντοτήτων του. Αν όμως δεν είναι γνωστός ο χρόνος εισαγωγής των οντοτήτων στο σύστημα, τότε το σύνολο των ανωνύμων οντοτήτων για τον επιτιθέμενο παραμένει σταθερό. Συμπερασματικά, ανωνυμία για μια οντότητα, σημαίνει ότι ο επιτιθέμενος δεν μπορεί να προσδιορίσει με ακρίβεια την οντότητα μέσα σε ένα σύνολο οντοτήτων. Τέλος μία συνολική ανωνυμία είναι ισχυρή όσο πιο ευρύ είναι το αντίστοιχο σύνολο ανωνύμων οντοτήτων και όσο πιο ισορροπημένη είναι η αποστολή και παραλαβή μηνυμάτων μεταξύ των οντοτήτων του συνόλου και συνεπώς ολόκληρου του συστήματος.

Προσοχή όμως! Η ανωνυμία δεδομένων δεν διασφαλίζει, κατ' ανάγκη, και την ιδιωτικότητα! Ας δούμε το εξής παράδειγμα μίας ιατρικής βάσης δεδομένων:

ID	QID			SA
Name	Zipcode	Age	Sex	Disease
Alice	47677	29	F	Ovarian Cancer
Betty	47602	22	F	Ovarian Cancer
Charles	47678	27	M	Prostate Cancer
David	47905	43	M	Flu
Emily	47909	52	F	Heart Disease
Fred	47906	47	M	Heart Disease

Η στήλη ID (Identity) περιέχει τα στοιχεία που *ταυτοποιούν* τον ασθενή πλήρως, π.χ. ονοματεπώνυμο και πατρώνυμο, ΑΜΚΑ, ΑΦΜ, αριθμός ταυτότητας κλπ. Οι στήλες QID (Quasi Identifier) περιέχουν στοιχεία που αποτελούν κάποια *μη ευαίσθητα* στοιχεία του ασθενούς που τον αφορούν αλλά δεν τον ταυτοποιούν. Τέλος τα στοιχεία στη στήλη SA (Sensitive Attributes). Προφανώς, δεν πρέπει, με κανέναν τρόπο, να διαρρεύσουν τα στοιχεία στη στήλη SA καθώς προδίδουν τις παθήσεις των ασθενών. Δείτε, τώρα, τον εξής ανωνυμοποιημένο πίνακα:

QID			SA
Zipcode	Age	Sex	Disease
47677	29	F	Ovarian Cancer
47602	22	F	Ovarian Cancer
47678	27	M	Prostate Cancer
47905	43	M	Flu
47909	52	F	Heart Disease
47906	47	M	Heart Disease

Κάποιος που βλέπει τον πίνακα αυτό συμπεραίνει ότι δεν είναι εφικτή η ταυτοποίηση ασθενών και η αποκάλυψη της πάθησής τους. Υποθέστε, όμως, ότι έχει αναρτηθεί κάπου στο διαδίκτυο ο εξής κατάλογος συμμετεχόντων π.χ. σε μία διαδικασία ψηφοφορίας ή συμμετοχής σε μία διαβούλευση, ο οποίος δεν περιέχει ευαίσθητα προσωπικά δεδομένα:

Name	Zipcode	Age	Sex
Alice	47677	29	F
Bob	47983	65	M
Carol	47677	22	F
Dan	47532	23	M
Ellen	46789	43	F

Φαινομενικά, από τους δύο προηγούμενους πίνακες δεν διαρρέει κάποια ευαίσθητη πληροφορία, δηλαδή ασθένεια κάποιου ασθενούς: ο πρώτος περιέχει *ανωνυμοποιημένη* πληροφορία και ο δεύτερος περιέχει *μη ευαίσθητη* πληροφορία. Μία πιο προσεκτική εξέταση, όμως, των δύο πινάκων *συνδυασμένα*, αποκαλύπτει την ασθένεια της Αλίκης! Προσέξτε τις δύο πρώτες γραμμές των δύο πινάκων. Ο δεύτερος πίνακας βοηθά, με τα δεδομένα στις στήλες QID, να συνδεθεί (*identification through linking* ονομάζεται το

αποτέλεσμα της σύνδεσης αυτής) το όνομα της Αλίκης με την εγγραφή του πρώτου πίνακα, αναιρώντας την ανωνυμία του πρώτου πίνακα όσον αφορά την Αλίκη. Το δίδαγμα είναι ότι τα ανώνυμα δεδομένα μπορεί να μην είναι και τόσο ανώνυμα τελικά!

4.2.2 Μη-συνδεσιμότητα

Η **μη-συνδεσιμότητα** δύο ή περισσότερων οντοτήτων, μηνυμάτων ή συμβάντων σημαίνει ότι, ο επιτιθέμενος στο σύστημά μας, δε βρίσκεται σε θέση να κατανοήσει αν τα παραπάνω στοιχεία συνδέονται μεταξύ τους. Με λίγα λόγια η μη-συνδεσιμότητα εξασφαλίζει ότι οι οντότητες ενός συστήματος δεν είναι σε θέση να εξακριβώσουν αν ένας χρήστης έχει λάβει μέρος σε συγκεκριμένες διαδικασίες εντός του συστήματος.

Η μέτρηση της μη-συνδεσιμότητας δύο ή περισσότερων IOIs προσδιορίζεται από τη διαφορά ανάμεσα στις παρατηρήσεις του επιτιθέμενου, δηλαδή *Μη συνδεσιμότητα* = *Νέα Γνώση* – *Προγενέστερη Γνώση*. Η διατήρηση της μη-συνδεσιμότητας μιας οντότητας παραμένει σταθερή όταν η παραπάνω σχέση είναι μηδέν. Για να ισχύει η Μη συνδεσιμότητα, δε θα πρέπει σε καμία περίπτωση το αποτέλεσμα της παραπάνω σχέσης να είναι θετικό. Μία τέτοια περίπτωση θα σήμαινε ότι η ικανότητα του επιτιθέμενου να συσχετίσει τα στοιχεία εντός του συστήματος έχει αυξηθεί.

Η Ανωνυμία και η Μη-συνδεσιμότητα είναι δύο έννοιες άρρηκτα συνδεδεμένες. Για την περιγραφή αυτής της σύνδεσης συνήθως μας ενδιαφέρει να προσδιορίσουμε: «Ποιος έχει αποστείλει ή παραλάβει, ποια μηνύματα;». Πιο αναλυτικά, μας ενδιαφέρει ο αποστολέας έστω s να είναι ανώνυμος σε σχέση με την αποστολή ενός συγκεκριμένου μηνύματος m , καθώς και να είναι ανώνυμος εντός του συνόλου των ανώνυμων αποστολέων. Το ίδιο ισχύει προφανώς και για ένα σύνολο από μηνύματα M καθώς και για το σύνολο των παραληπτών R . Ανωνυμία Αποστολέα ορίζεται ότι κάθε μήνυμα σχετικό με τον εν δυνάμει αποστολέα είναι μη συνδέσιμο ενώ αντιστοίχως Ανωνυμία Παραλήπτη ορίζεται ότι κάθε μήνυμα σχετικό με τον εν δυνάμει παραλήπτη είναι μη συνδέσιμο. Ακόμη, Ανωνυμία Σχέσης ορίζεται η μη ανιχνεύσιμη επικοινωνία ανάμεσα σε αποστολέα και παραλήπτη, δηλαδή ο αποστολέας και ο παραλήπτης είναι μη-συνδέσιμοι. Τέλος ως Σύνολο Ανωνύμων Σχέσεων ορίζεται το καρτεσιανό γινόμενο δύο διαφορετικών συνόλων. Για παράδειγμα όταν μία οντότητα στέλνει ή λαμβάνει ένα μήνυμα, το σύνολο ανωνύμων σχέσεων είναι το σύνολο όλων των πιθανών παραληπτών ή αποστολέων του συγκεκριμένου μηνύματος.

4.2.3 Ψευδωνυμία

Η **Ψευδωνυμία** αφορά τη δημιουργία ενός προσδιοριστικού, δηλαδή ενός ψευδωνύμου, που απευθύνεται σε ένα αποστολέα ή παραλήπτη αντί για το πραγματικό του όνομα. Το συγκεκριμένο προσδιοριστικό μπορεί να είναι κάποιο τυχαίο όνομα ή μία ακολουθία από bits και είναι ανεξάρτητο από το πραγματικό όνομα της οντότητας ή των ιδιοτήτων της. Κάτοχος του ψευδωνύμου ονομάζεται η οντότητα στην οποία αναφέρεται και η οποία χρησιμοποιεί το ψευδώνυμο. Υπάρχουν δύο είδη ψευδωνύμων: το *Αποκλειστικό* και το *Μεταβιβάσιμο* ψευδώνυμο. Το αποκλειστικό αναφέρεται σε ένα μοναδικό κάτοχο, δε μεταβιβάζεται σε άλλες οντότητες και είναι αμετάβλητο ενώ το μεταβιβάσιμο μπορεί να μεταβιβαστεί από έναν κάτοχο σε άλλη οντότητα, η οποία θα είναι πλέον ο νέος κάτοχος.

Με την ψευδωνυμία εξασφαλίζεται ότι η οντότητα-χρήστης μπορεί να προσπελάσει δεδομένα ή υπηρεσίες χωρίς να πρέπει να αποκαλύψει την ταυτότητα του και χωρίς να μπορεί να προσδιοριστεί η ταυτότητά του από ένα σύνολο οντοτήτων. Παρόλα αυτά η οντότητα-χρήστης είναι υπόλογος για την ενέργειά του και μπορεί να του αποδοθεί ευθύνη. Όσον αφορά την ανωνυμία της οντότητας, σε αυτό το σημείο θα πρέπει να αποσαφηνιστεί πρώτα η διαφορά των όρων ανώνυμος και ψευδώνυμος. Ο όρος ανώνυμος αφορά την ιδιότητα μιας οντότητας συγκριτικά με την αναγνωρισιμότητα, ενώ ο όρος ψευδώνυμος αφορά τη χρήση κάποιου μηχανισμού, όπως χρήση προσδιοριστικών. Γενικά όσο πιο μικρή είναι η γνώση για τη σχέση ψευδωνύμου-κατόχου, τόσο πιο ισχυρή είναι η ανωνυμία. Από την άλλη πλευρά η ισχύς της ανωνυμίας θα μειώνεται όσο αυξάνει η γνώση για τη σχέση ψευδωνύμου-κατόχου.

Συμπερασματικά η ανωνυμία θα είναι πιο ισχυρή εάν έχουμε στην διάθεσή μας όσο το δυνατόν λιγότερες προσωπικές πληροφορίες του κατόχου του ψευδωνύμου που σχετίζονται με το ψευδώνυμο. Επιπλέον όσο λιγότερο χρησιμοποιούνται τα ψευδώνυμα και σε όσο το δυνατόν λιγότερες διαδικασίες τόσο πιο λίγες πληροφορίες για τον κάτοχο μπορούν να συσχετιστούν με αυτά. Το ίδιο προφανώς ισχύει ένα επιλέγονται αρκετά συχνά ψευδώνυμα με τυχαίο και ανεξάρτητο τρόπο για νέες χρήσεις.

4.2.4 Διαχείριση Ταυτότητας

Η **Ταυτότητα** αφορά ένα υποσύνολο των ιδιοτήτων κάποιας οντότητας, είτε αυτή είναι φυσικό ή νομικό πρόσωπο, είτε υπολογιστής, είτε διεργασία και η οποία προσδιορίζει επακριβώς το πρόσωπο αυτό, σε ένα σύνολο προσώπων. Επειδή οι τιμές των ιδιοτήτων ή ακόμα και οι ιδιότητες μπορεί να αλλάζουν μέσα στο χρόνο δεν υφίσταται μία μοναδική ταυτότητα, αλλά πολλαπλές. Η Ψηφιακή ταυτότητα αφορά την απόδοση ιδιοτήτων σε μία συγκεκριμένη οντότητα και η οποία είναι άμεσα αξιοποιήσιμη (π.χ. το email σε μία λίστα).

Σε αντιστοιχία με το σύνολο των ανωνύμων οντοτήτων, μπορούμε να χρησιμοποιήσουμε και το σύνολο των αναγνωρίσιμων οντοτήτων προκειμένου να περιγράψουμε πιο αναλυτικά την ταυτότητα και την αναγνωρισιμότητα των οντοτήτων. Η αναγνωρισιμότητα μίας οντότητας αφορά στην ικανότητα του επιτιθέμενου να αναγνωρίσει την οντότητα ανάμεσα σε ένα σύνολο οντοτήτων.

Η **Διαχείριση Ταυτότητας** αφορά την προστασία της ιδιωτικότητας εφόσον διαφυλάττεται πλήρως η μη-συνδεσιμότητα μεταξύ των ταυτοτήτων. Η διαχείριση της ταυτότητας επιτυγχάνεται μέσω των συστημάτων διαχείρισης ταυτότητας (IMS), δηλαδή διάφορα εργαλεία, εφαρμογές και υποδομές που μπορούν να εγκατασταθούν και να λειτουργήσουν τόσο στην πλευρά του user όσο και του server και να λειτουργούν συνεργατικά. Τα συγκεκριμένα συστήματα ενισχύουν και προστατεύουν την ιδιωτικότητα όταν με βάση τους περιορισμούς από ένα σύνολο εφαρμογών λογισμικού, προστατεύουν τη μη-συνδεσιμότητα ανάμεσα σε ταυτότητες και τα αντίστοιχα ψευδώνυμα ενός φυσικού προσώπου.

Σε όλες τις παραπάνω απαιτήσεις θα μπορούσε να ενταχθεί και η **μη-παρατηρησιμότητα** η οποία και συναντάται στη βιβλιογραφία ως τέταρτη απαίτηση έναντι της διαχείρισης ταυτότητας. Η μη-παρατηρησιμότητα στην ουσία χρησιμοποιεί ένα φάσμα ισχυρών μηχανισμών για να προστατεύσει την ιδιωτικότητα των χρηστών από επιτιθέμενους. Απαγορεύει δηλαδή στους επιτιθέμενους να παρατηρήσουν ή ανιχνεύσουν τους χρήστες τη στιγμή χρησιμοποιούν μια υπηρεσία πχ όταν περιηγούνται στο διαδίκτυο. Τέλος θα μπορούσαμε να θεωρήσουμε την μη-παρατηρησιμότητα ως υποκατηγορία της ανωνυμίας και της μη συνδεσιμότητας.

4.3 Απαιτήσεις Ταυτοποίησης

Πριν προχωρήσουμε στις τεχνολογίες προστασίας της ιδιωτικότητας (PETS) αρκεί να αναφερθούμε συνοπτικά σε δύο κλασικές απαιτήσεις ταυτοποίησης χρηστών με

προστασία της ιδιωτικότητας. Οι συγκεκριμένες απαιτήσεις ανήκουν και αυτές στο ευρύτερο φάσμα των απαιτήσεων της ιδιωτικότητας και θα μπορούσαν να αναφερθούν παραπάνω, όμως για διευκόλυνση εισαγωγής νέας ορολογίας και για λόγους παρουσίας τις αναφέρουμε εδώ. **Ταυτοποίηση** μιας οντότητας καλείται η διαδικασία εκείνη, κατά την οποία η οντότητα παρέχει σε ένα Πληροφοριακό Σύστημα τις πληροφορίες που απαιτούνται προκειμένου να συσχετιστεί με μία οντότητα η οποία και δικαιούται προσπέλασης στους πόρους του.

4.3.1 Αυθεντικοποίηση

Αυθεντικοποίηση μίας οντότητας, καλείται η διαδικασία εκείνη, κατά την οποία η ίδια η οντότητας παρέχει σε ένα Πληροφοριακό Σύστημα τις πληροφορίες που απαιτούνται προκειμένου να ελεγχθεί η βασιμότητα της συσχέτισης που επιτεύχθηκε κατά τη διαδικασία της ταυτοποίησης. Με λίγα λόγια είναι η διαδικασία επιβεβαίωσης της ταυτότητας μιας οντότητας. Σε ιδιωτικά και δημόσια δίκτυα, η αυθεντικοποίηση υλοποιείται συνήθως με τη χρήση κωδικών πρόσβασης ενώ αποτελεί κυρίως απαίτηση ασφάλειας, παρά ιδιωτικότητας ενός συστήματος.

4.3.2 Εξουσιοδότηση

Εξουσιοδότηση σε μία οντότητα ορίζεται η διαδικασία κατά την οποία η οντότητα αποκτά δικαιώματα σε μια μεμονωμένη υπηρεσία ή σε ένα σύνολο υπηρεσιών ενός Πληροφοριακού Συστήματος. Σε ένα Πληροφοριακό Σύστημα με πολλούς χρήστες, ο διαχειριστής έχει το δικαίωμα να εξουσιοδοτεί τον κάθε χρήστη με αντίστοιχα δικαιώματα, ανάλογα με το ρόλο τους και τις υποχρεώσεις τους μέσα στο σύστημα.

Εν κατακλείδι θα μπορούσαμε να ισχυριστούμε ότι μέσω της αυθεντικοποίησης ελέγχεται η ταυτότητα μίας οντότητας με τις κατάλληλες εξουσιοδοτήσεις (δικαιώματα) και ταυτοποιείται ως ενεργός χρήστης του συστήματος.