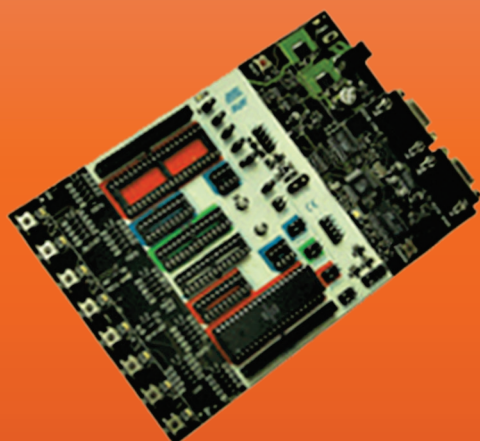


Τοπάλης Ευάγγελος
Δρ. Ηλεκτρολόγος Μηχανικός & Τεχνολογίας Υπολογιστών|

Μικροεπεξεργαστής AVR ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

2024



ISBN 978-618-00-5262-6

Copyright 2024
ISBN 978-618-00-5262-6

ΠΕΡΙΕΧΟΜΕΝΑ.....	2
ΚΕΦΑΛΑΙΟ 1 - ΜΙΚΡΟΪΠΟΛΟΓΙΣΤΕΣ.....	4
1.1 Εισαγωγή.....	4
1.2 Ο Μικροεπεξεργαστής	4
1.3 Βασική Δομή ενός Μικροϋπολογιστή	5
1.4 Ο Μικροελεγκτής	6
1.5 Εισαγωγή της έννοιας του Λογισμικού	6
1.6 Εφαρμογές των μικροϋπολογιστών	11
ΚΕΦΑΛΑΙΟ 2 - ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΟΥ AVR AT90S8515 ..	12
2.1. Διαθέσιμοι Καταχωρητές.....	13
2.2. Διαθέσιμα Είδη Μνημών.....	14
2.3. Υποστήριξη Στοιβάς.....	16
2.4. Αριθμητική και Λογική Μονάδα	17
2.5. Θύρες Επικοινωνίας με Περιφερειακές Συσκευές.....	17
2.6. ΕΠΪΛΟΓΟΣ.....	18
ΚΕΦΑΛΑΙΟ 3 - ΠΕΡΙΓΡΑΦΗ ΤΟΥ AVR STK500 ΜΙΚΡΟΚΙΤ.....	20
3.1 Γενικά	20
3.2 Διακόπτες και Leds	21
3.3 Σύνδεση του ΜικροΚΙΤ με τον υπολογιστή.....	23
3.4 Downloading του προγράμματος στο ΜικροΚΙΤ.....	23
ΚΕΦΑΛΑΙΟ 4 ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ AVR STUDIO 4.....	25
4.1 Εγκατάσταση του Λογισμικού AVR Studio 4	25
4.2 Περιβάλλον του Λογισμικού AVR Studio 4	26
ΚΕΦΑΛΑΙΟ 5 - ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ.....	45
ΑΣΚΗΣΗ 1 – Γενικά για τους Μικροϋπολογιστές, Εξοικείωση με το ΚΙΤ και το Περιβάλλον Εργασίας	45
ΑΣΚΗΣΗ 2 – Εντολές Μεταφοράς Δεδομένων σε Καταχωρητές και Μνήμη	46

ΑΣΚΗΣΗ 3 – Αριθμητικές Πράξεις δεκαεξαδικών 8-bit/16-bit αριθμών.....	53
ΑΣΚΗΣΗ 4 – Λογικές Πράξεις, Εντολές Αύξησης Μείωσης, Εντολή NOP.....	59
ΑΣΚΗΣΗ 5 – Εντολές διακλάδωσης ή μεταφοράς ελέγχου	66
ΑΣΚΗΣΗ 6 – Εύρεση Μεγίστου / Ελαχίστου	74
ΑΣΚΗΣΗ 7 – Συγκρίσεις, Διακλαδώσεις & Βρόγχοι - Διαγράμματα Ροής.....	76
ΑΣΚΗΣΗ 8 – I/O (Ports, Push Buttons, Leds), Υπορουτίνες Καθυστέρησης	77
ΑΣΚΗΣΗ 9 – I/O (Ports, Push Buttons, Leds).....	83
ΑΣΚΗΣΗ 10 – Αλγόριθμοι Ταξινόμησης (Επαναληπτική Άσκηση)	90
ΑΣΚΗΣΗ 11 – ΕΛΕΓΧΟΣ ΣΤΗΝ ΤΑΧΥΤΗΤΑ ΚΑΙ ΣΤΗ ΦΟΡΑ ΕΝΟΣ ΚΙΝΗΤΗΡΑ ΜΕ ΓΡΑΜΜΙΚΟ ΠΟΤΕΝΣΙΟΜΕΤΡΟ	93
(KIT και APPLICATION BOARD)	93
ΑΣΚΗΣΗ 12 – ΔΙΑΤΗΡΗΣΗ ΣΤΑΘΕΡΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΕΝΤΟΣ ΟΡΙΩΝ ΣΤΗ ΘΕΡΜΟΑΝΤΙΣΤΑΣΗ ΜΕ ΤΗΝ ΒΟΗΘΕΙΑ ΤΟΥ ΚΙΝΗΤΗΡΑ.....	108
(KIT και APPLICATION BOARD)	108
ΑΣΚΗΣΗ 13 – ΕΛΕΓΧΟΣ ΤΩΝ LEDS ΜΕ ΦΩΤΟΔΙΟΔΟ	112
(KIT και APPLICATION BOARD)	112
ΒΙΒΛΙΟΓΡΑΦΙΑ.....	117
ΠΑΡΑΡΤΗΜΑ 1 - ΡΕΠΕΡΤΟΡΙΟ ΕΝΤΟΛΩΝ ΤΟΥ AVR	118
Π1.1 Συμβολισμοί περιγραφής εντολών.....	118
Π1.2 Πλήρης περίληψη του συνόλου εντολών.....	118
Π1.2.1 ΕΝΤΟΛΕΣ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ	118
Π1.2.2 ΕΝΤΟΛΕΣ ΑΡΙΘΜΗΤΙΚΩΝ ΚΑΙ ΛΟΓΙΚΩΝ ΠΡΑΞΕΩΝ.....	119
Π1.2.3 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΔΙΑΚΛΑΔΩΣΗΣ	119
Π1.3 Περιγραφή Εντολών.....	120
Π1.3.1 ΕΝΤΟΛΕΣ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ	120
Π1.3.2 ΕΝΤΟΛΕΣ ΑΡΙΘΜΗΤΙΚΩΝ ΚΑΙ ΛΟΓΙΚΩΝ ΠΡΑΞΕΩΝ.....	125
Π1.3.3 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΔΙΑΚΛΑΔΩΣΗΣ	130
Π1.3.4 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΚΑΙ ΕΝΤΟΛΕΣ ΣΕ ΕΠΙΠΕΔΟ BIT	136

1.1 Εισαγωγή

Ο Μικροϋπολογιστής είναι ένας «μικρός» υπολογιστής και αποτελείται από τα εξής μέρη

Τον Μικροεπεξεργαστή ή CPU

Την Μνήμη (RAM, ROM, Flash)

Τα Κυκλώματα Εισόδου / Εξόδου ή όπως λέγονται I/O (Input / Output). Τα I/O χρησιμοποιούνται για την διασύνδεση περιφερειακών «εξωτερικών» συσκευών όπως η οθόνη, το πληκτρολόγιο, ο εκτυπωτής, ο δίσκος αποθήκευσης κλπ.

Όλα αυτά τα στοιχεία είναι τοποθετημένα πάνω σε μια πλακέτα (board). Ο μικροϋπολογιστής είναι το υπολογιστικό σύστημα πάνω στο οποίο αναπτύχθηκαν, αφενός οι προσωπικοί υπολογιστές (PC's) και αφετέρου τα σύγχρονα υπολογιστικά συστήματα βασισμένα στην αρχιτεκτονική της κατανεμημένης ή παράλληλης επεξεργασίας (distributed / parallel processing). Πέρα από αυτά όμως ο μικροϋπολογιστής σήμερα υπάρχει μέσα σε οποιαδήποτε ηλεκτρονική ή ηλεκτρική συσκευή χρειάζεται υπολογιστική νοημοσύνη. Έτσι ο μικροϋπολογιστής υπάρχει μέσα στο αυτοκίνητο (ABS, Injection, Cruise control, αερόσακοι, διαγνωστικά, κλπ.), μέσα σε συσκευές της κατοικίας (πλυντήριο, τηλεόραση, ψυγείο, κλπ.), στο γραφείο (τηλεφωνικό κέντρο, τηλεφωνική συσκευή, φαξ, κλπ.), στα συστήματα αυτοματισμού κτιρίων και κατοικιών (κλιματισμός, θέρμανση / ψύξη, ασφάλεια, φωτισμός, κλπ.), στις προσωπικές μας συσκευές (κινητό τηλέφωνο, ρολόι, ηλεκτρονική ατζέντα, έξυπνες κάρτες, κλπ.) και αλλού.

Δεν χρειάζεται να αναφερθεί η βιομηχανία, η βιοτεχνία και η εν γένει παραγωγή. Όλοι οι βιομηχανικοί αυτοματισμοί (PLC, industrial controllers, AC/DC Drives, Inverters, Converters, κλπ.) βασίζονται στην «νοημοσύνη» που παρέχει ο μικροϋπολογιστής.

1.2 Ο Μικροεπεξεργαστής

Η κεντρική επεξεργαστική μονάδα ή Central Processing Unit ή CPU ή απλά ο μικροεπεξεργαστής αποτελεί το βασικότερο μέρος ενός μικροϋπολογιστή. Είναι ένα πολύπλοκο ψηφιακό κύκλωμα το οποίο αποτελείται από τα εξής μέρη.

Καταχωρητές (Registers) για προσωρινή αποθήκευση δεδομένων (data), εντολών (commands), όπως και άλλων πληροφοριών.

Αποκωδικοποιητή Εντολών (Command Decoder) για την μετατροπή μιας εντολής προγράμματος σε μια σειρά λειτουργιών, μέσω των οποίων υλοποιείται η εν λόγω εντολή.

Αριθμητική και Λογική Μονάδα (Α / Λ Μ) (Arithmetic / Logic Unit – ALU) η οποία εκτελεί αριθμητικές (αθροιστής κλπ.) και λογικές πράξεις.

Κυκλώματα Χρονισμού και Ελέγχου (Timing / Control) τα οποία με ψηφιακό ρολόι και ψηφιακούς απαριθμητές παράγουν δευτερεύοντα περιοδικά και μη σήματα, για να ελέγχουν και διατηρούν τη σωστή ακολουθία των λειτουργιών στα πλαίσια της εκτέλεσης εντολών προγράμματος.

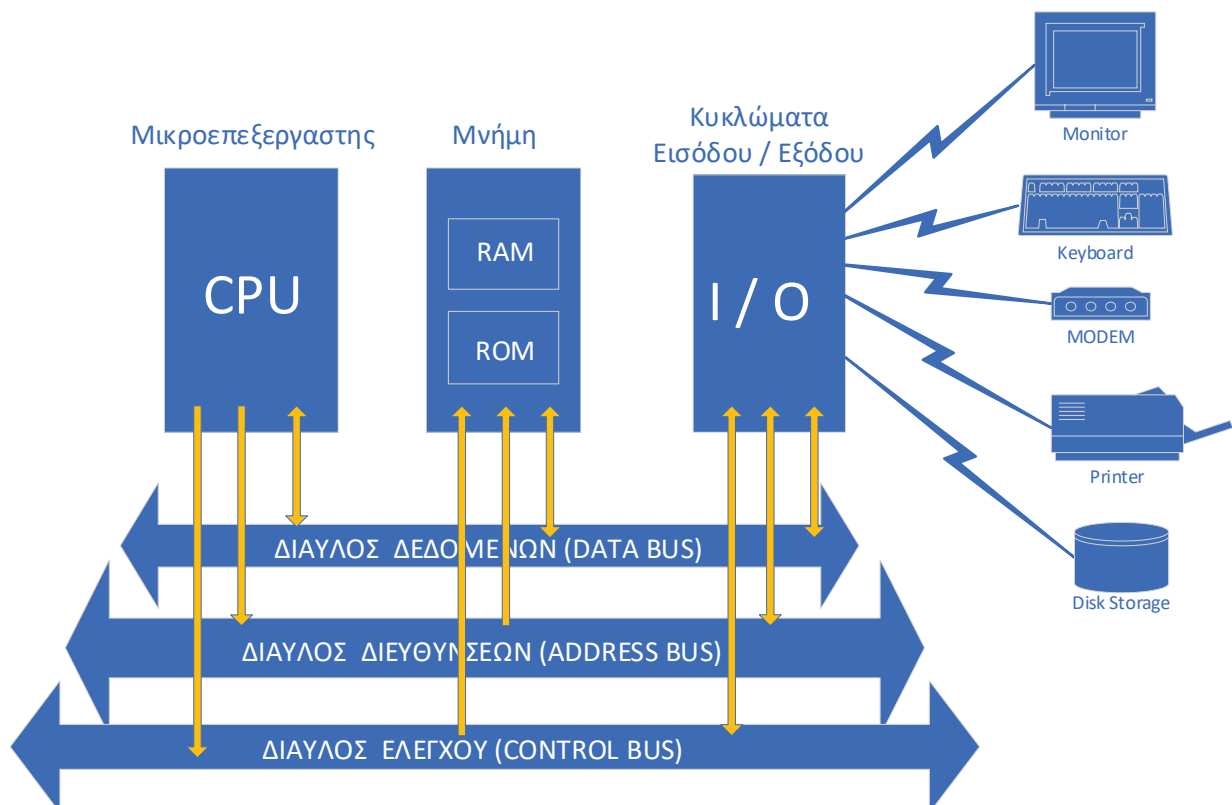
Εσωτερικούς διαύλους επικοινωνίας (Internal Bus) για την επικοινωνία όλων αυτών των μονάδων μεταξύ των, μέσα στην CPU.

Διασύνδεση των εσωτερικών διαύλων με τους εξωτερικούς διαύλους (Data / Address / Control Buses Interfaces) για την επικοινωνία της CPU με τα άλλα τμήματα του μικροϋπολογιστή όπως μνήμη και I/O μέσω των τριών διαύλων Δεδομένων / Διευθύνσεων / Ελέγχου.

Η CPU υλοποιείται από την συνεργασία και συνλειτουργία όλων αυτών των προαναφερομένων μερών ή μονάδων, και είναι «συγκεντρωμένη» μέσα σε ένα chip.

1.3 Βασική Δομή ενός Μικροϋπολογιστή

Η δομή αυτή φαίνεται στο παρακάτω σχήμα (Εικόνα 1.1). Τα τρία μέρη του μικροϋπολογιστή, δηλαδή η CPU, η Μνήμη και τα I/O επικοινωνούν με τρεις διαύλους επικοινωνίας, τον Δίαυλο Δεδομένων (Data Bus), τον Δίαυλο Διευθύνσεων (Address Bus) και τον Δίαυλο Ελέγχου (Control Bus).



Εικόνα 1.1

Η υπολογιστική ισχύς γενικά καθορίζεται από τα εξής στοιχεία:

Την ταχύτητα (το ρολόι) της CPU (πχ 700 MHz ή 2GHz). Όσο υψηλότερη τόσο ισχυρότερος.

Το εύρος του διαύλου δεδομένων (πχ 8-bit, 16-bit, 32-bit, 64-bit). Όσο περισσότερα bits τόσο ισχυρότερος.

Το εύρος του διαύλου διευθύνσεων (πχ 16-bit → 64 kBytes, 32-bit → 4 Gbytes). Όσο μεγαλύτερος τόσο περισσότερη μνήμη μπορεί να διευθυνσιοδοτηθεί και τόσο ισχυρότερο είναι το σύστημα.

Το πλήθος των εσωτερικών καταχωρητών της CPU. Όσο περισσότεροι τόσο ισχυρότερος.

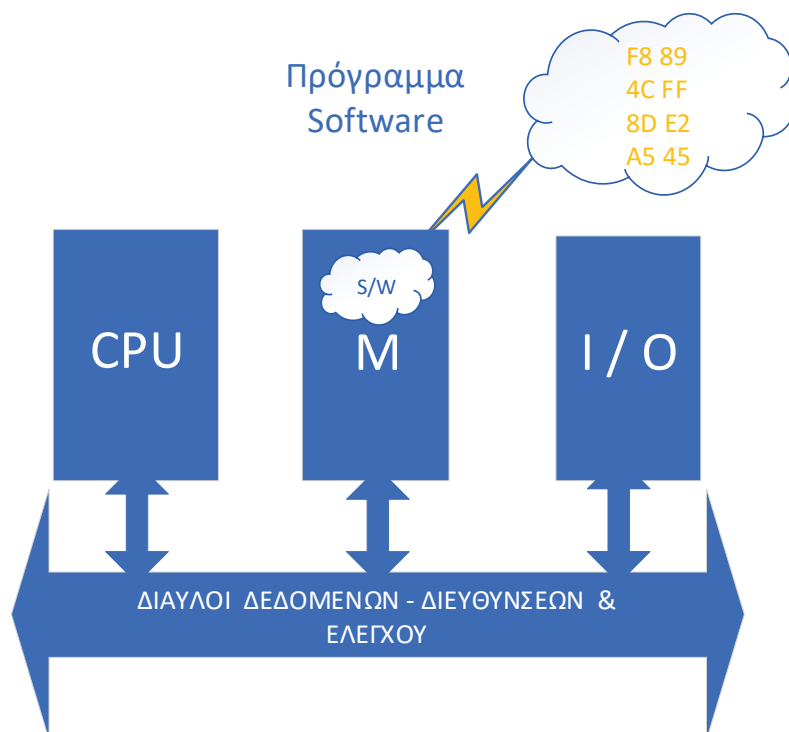
Την παρουσία ειδικής μνήμης “cache” δίπλα στην CPU.

1.4 Ο Μικροελεγκτής

Είναι ένας Μικροϋπολογιστής «συγκεντρωμένος» σε ένα chip. Δηλαδή CPU, Μνήμη και I/O είναι όλα συγκεντρωμένα στο ίδιο chip. Οι μικροελεγκτές χρησιμοποιούνται στις εφαρμογές αυτοματισμού και στην «τοποθέτηση» υπολογιστικής νοημοσύνης σε μικρές συσκευές (πχ κινητό τηλέφωνο).

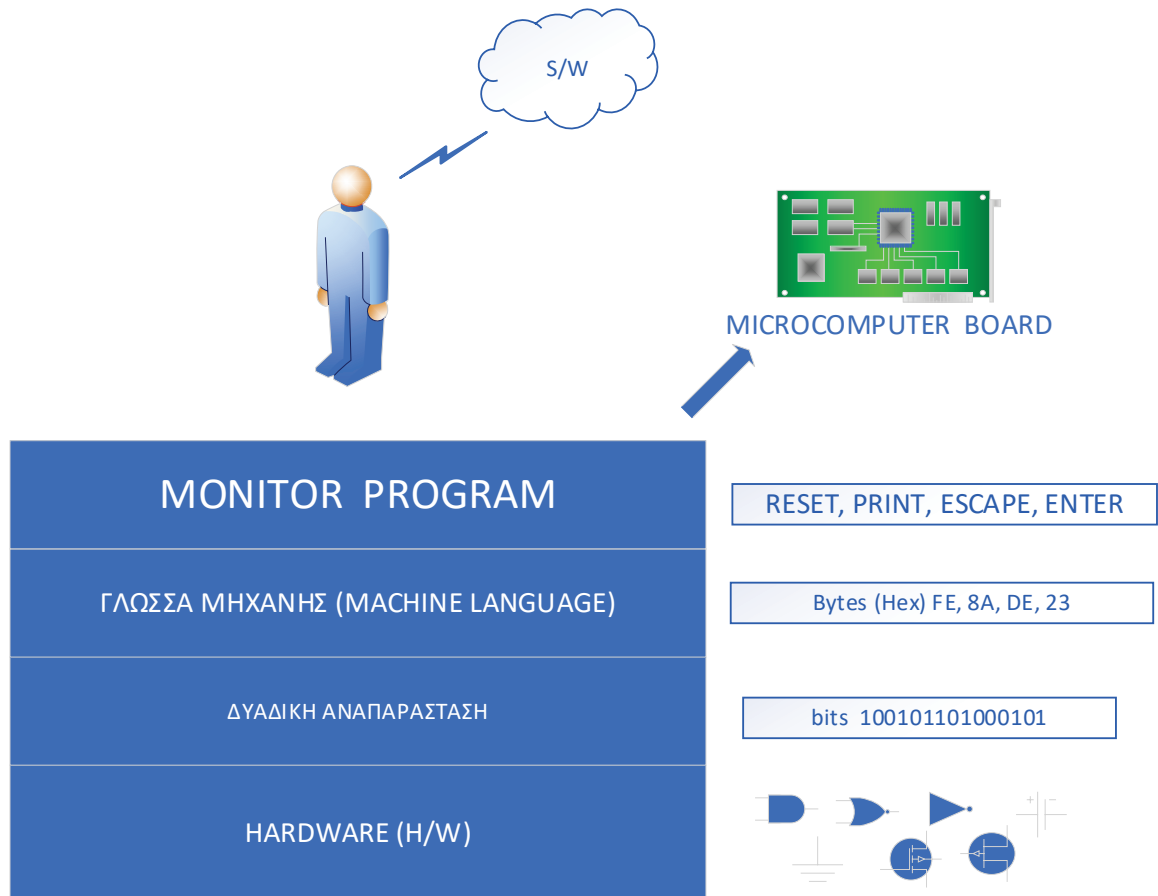
1.5 Εισαγωγή της έννοιας του Λογισμικού

Το Λογισμικό είναι πρόγραμμα φορτωμένο στην μνήμη του μικροϋπολογιστή όπως φαίνεται στο παρακάτω σχήμα (Εικόνα 1.2).



Εικόνα 1.2

Το Λογισμικό επιτρέπει τον διαρκή και αέναο «επαναπρογραμματισμό» του μικροϋπολογιστή ώστε να μπορεί να αλλάζει λειτουργία και να αντεπεξέρχεται σε διαφορετικά προβλήματα διατηρώντας την ίδια δομή και αρχιτεκτονική. Το λογισμικό δίνει την δυνατότητα της προηγμένης επικοινωνίας του ανθρώπου με το υλικό – Hardware του μικροϋπολογιστή. Η «θέση» του λογισμικού σε σχέση με το υλικό φαίνεται στο παρακάτω σχήμα (Εικόνα 1.3).

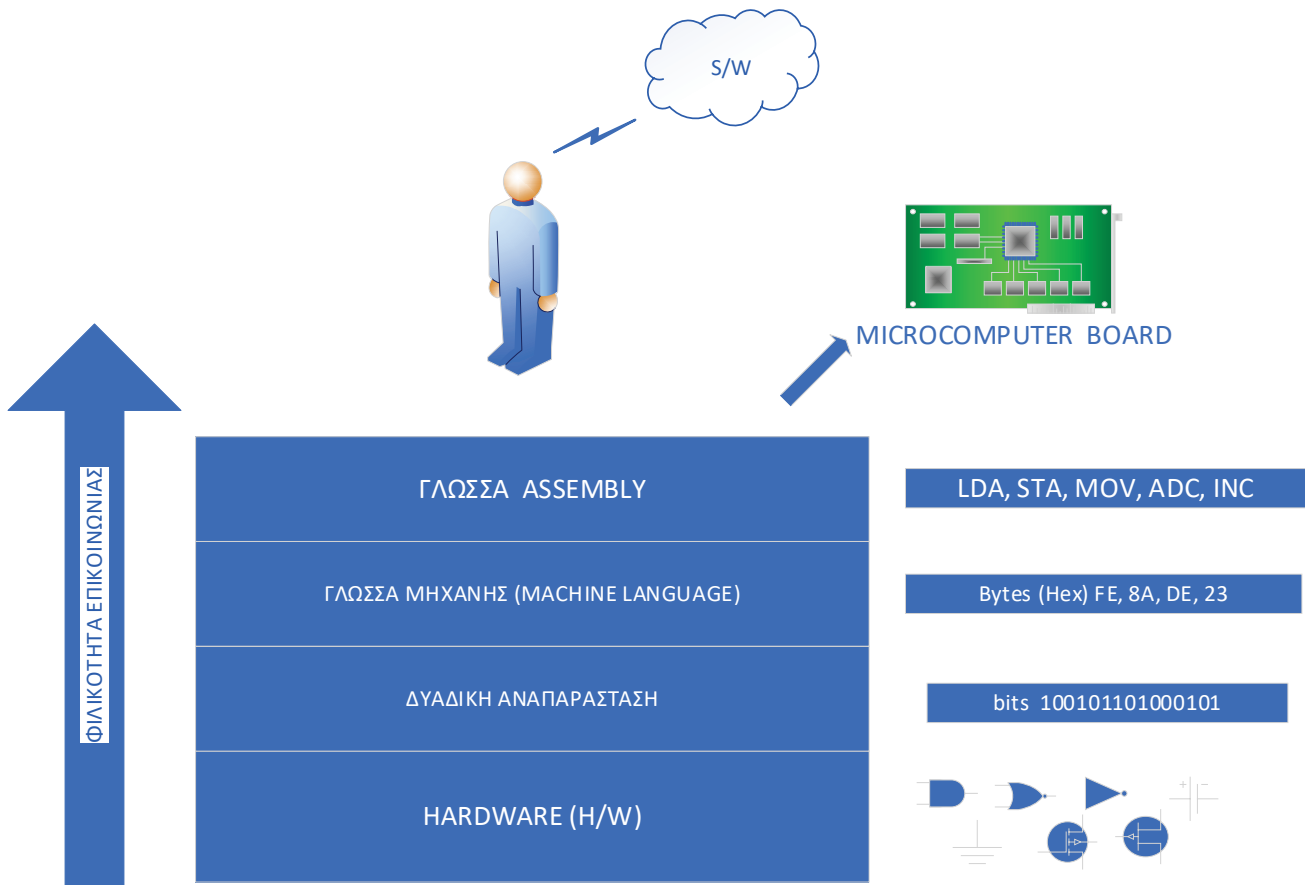


Εικόνα 1.3

Η **Γλώσσα Μηχανής** και το **Monitor Program** είναι τα δύο επίπεδα λογισμικού από την Μηχανή προς τον Άνθρωπο. Στο χαμηλότερο επίπεδο βρίσκεται το υλικό ή Hardware (H/W). Πάνω από το H/W υπάρχει «εικονικά» η δυαδική αναπαράσταση των ηλεκτρικών μεγεθών του H/W που είναι υψηλές (high=1) και χαμηλές (low=0) στάθμες τάσης. Πάνω από αυτό το επίπεδο υπάρχει η «ομαδοποίηση» των bits σε Bytes και η «εμφάνιση» της **Γλώσσας Μηχανής** στο δεκαεξαδικό σύστημα.

Από εδώ και πάνω εισάγεται η έννοια του λογισμικού ή προγράμματος (software – S/W). Το προφορτωμένο λογισμικό ή **Monitor Program** είναι το ανώτατο επίπεδο επικοινωνίας του χειριστή με τον μικροϋπολογιστή μέσω εντολών λειτουργικού προγράμματος. Από εδώ ο χρήστης χειρίζεται το σύστημα του μικροϋπολογιστή με εντολές.

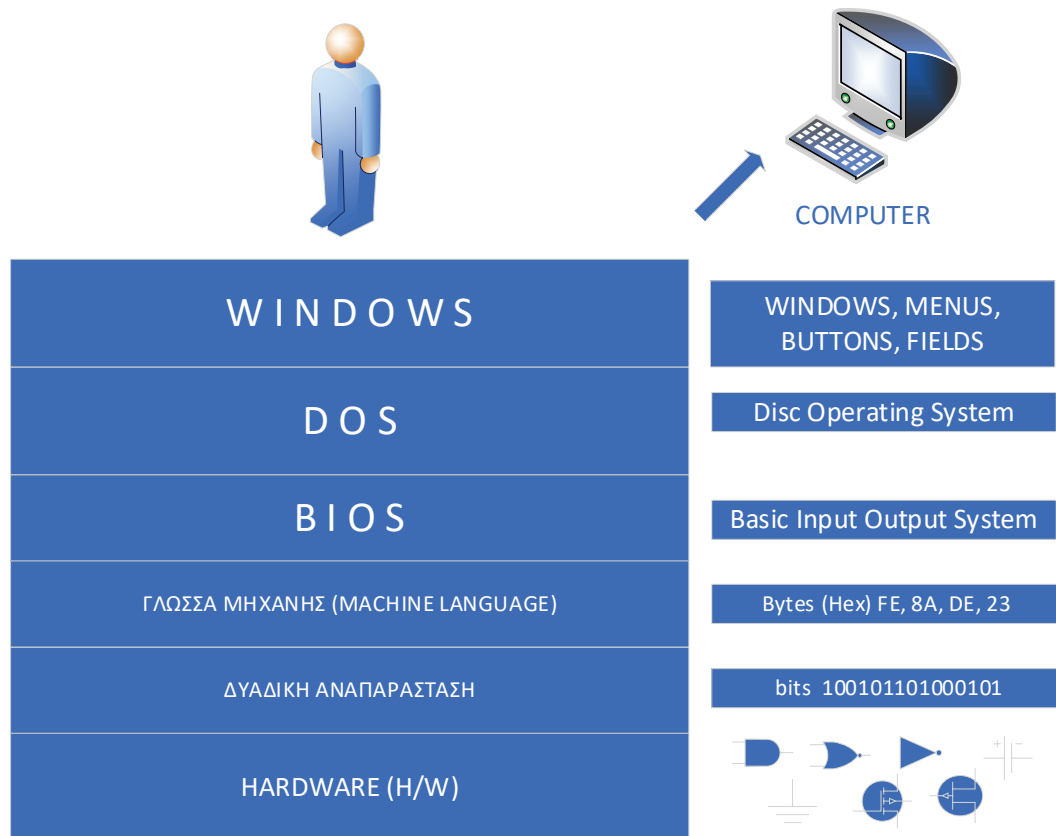
Ο χρήστης μπορεί να προγραμματίζει τον μικροϋπολογιστή να κάνει διάφορες εργασίες μέσω της γλώσσας προγραμματισμού Assembly. Η Assembly στην ουσία είναι μια «μετάφραση» της γλώσσας μηχανής σε «ανθρώπινα αντιληπτή μορφή» δηλαδή μοιάζει με ανθρώπινη γλώσσα επικοινωνίας. Το αντίστοιχο μοντέλο σε επίπεδα έχει την παρακάτω μορφή (Εικόνα 1.4).



Εικόνα 1.4

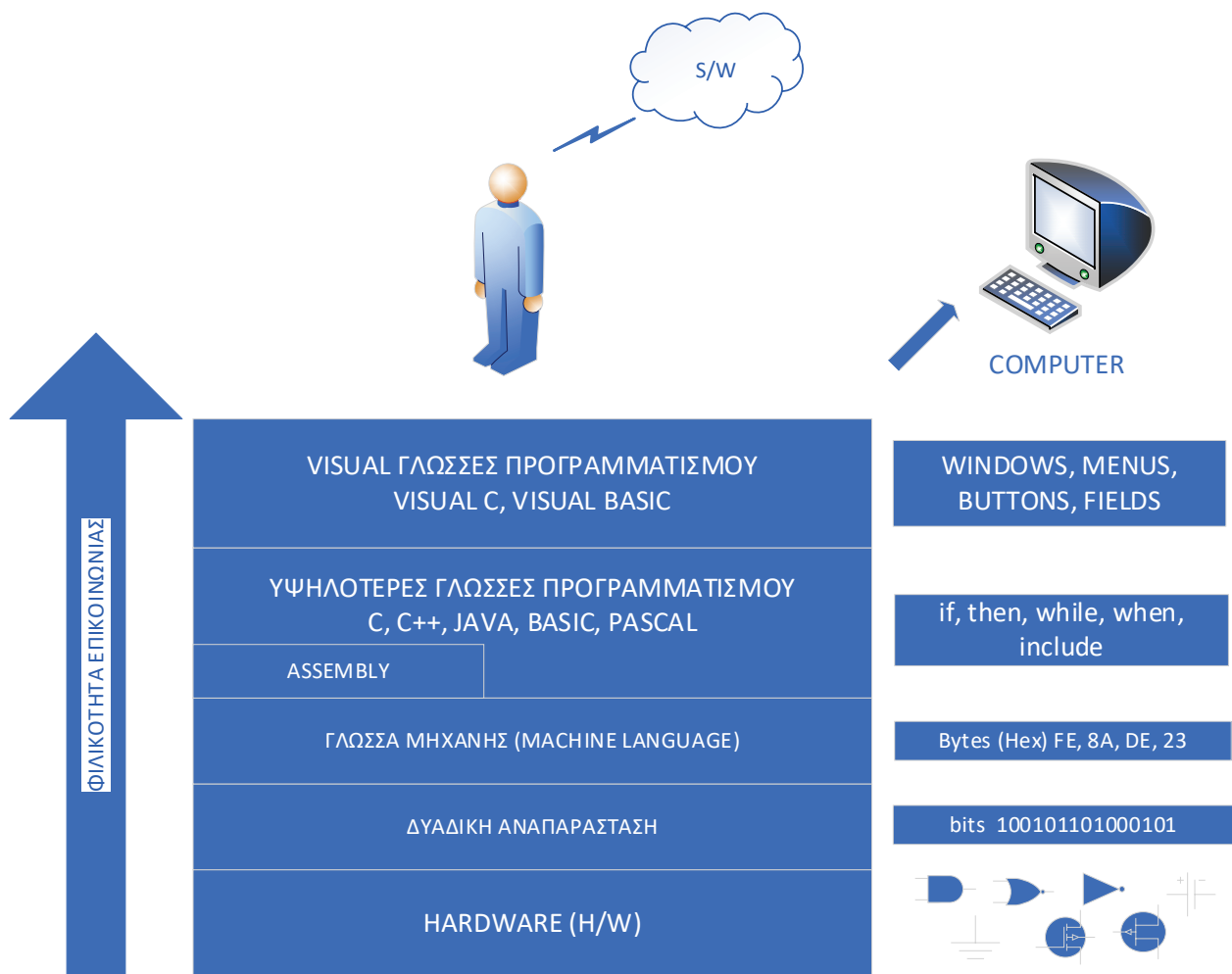
Από τα δύο προηγούμενα σχήματα φαίνεται ότι η Γλώσσα Assembly και το Monitor Program βρίσκονται στο ίδιο επίπεδο πάνω από την Γλώσσα Μηχανής. Πράγματι αυτά τα δύο λογισμικά βρίσκονται σε παράλληλη λειτουργία. Το Monitor είναι λογισμικό τοποθετημένο από τον κατασκευαστή και επιτρέπει τον γενικό έλεγχο του μικροϋπολογιστή ενώ η Assembly είναι λογισμικό που επιτρέπει τον προγραμματισμό σε γλώσσα Assembly.

Αν κάναμε ένα μοντέλο επιπέδων όπως τα δυο προηγούμενα, για τον Η/Υ ή τον PC μας τότε αυτά θα είχαν την εξής μορφή.



Εικόνα 1.5

Στον Η/Υ μας το ισχυρό λειτουργικό σύστημα που αντιστοιχεί στο Monitor Program του μικροϋπολογιστή, είναι τα επίπεδα από το BIOS έως και τα WINDOWS.



Εικόνα 1.6

Αντίστοιχα στο προγραμματιστικό περιβάλλον του Η/Υ, αντί της Assembly του μικροϋπολογιστή, έχουμε Υψηλότερες Γλώσσες Προγραμματισμού. Η Assembly μπορεί επίσης να συνεχίζει να υπάρχει πάνω από την γλώσσα μηχανής όπως φαίνεται στην παραπάνω εικόνα. Με αναφορά τον προσωπικό μας Η/Υ μας, μπορούμε να πούμε ότι αποτελείται από τα εξής μέρη, διότι βασίζεται στον μικροϋπολογιστή.

ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΗ (ταχύτητα 700 MHz – εύρος ψηφιακής λέξης 32 bit))

ΜΝΗΜΗ (ποσότητα πχ. 64 MBytes – ταχύτητα πχ. 133 MHz)

ΚΥΚΛΩΜΑΤΑ ΕΙΣΟΔΟΥ / ΕΞΟΔΟΥ (ποσότητα – είδη πχ. ISA, PCI, AGP, IDE, EIDE, SCSI, Serial, Parallel))

ΕΞΩΤΕΡΙΚΕΣ ΣΥΣΚΕΥΕΣ (οθόνη, πληκτρολόγιο, εκτυπωτής, δίσκος κλπ.)

ΔΙΑΥΛΟΙ ΕΠΙΚΟΙΝΩΝΙΑΣ

ΔΕΔΟΜΕΝΑ

ΔΙΕΥΘΥΝΣΕΙΣ

ΕΛΕΓΧΟΣ

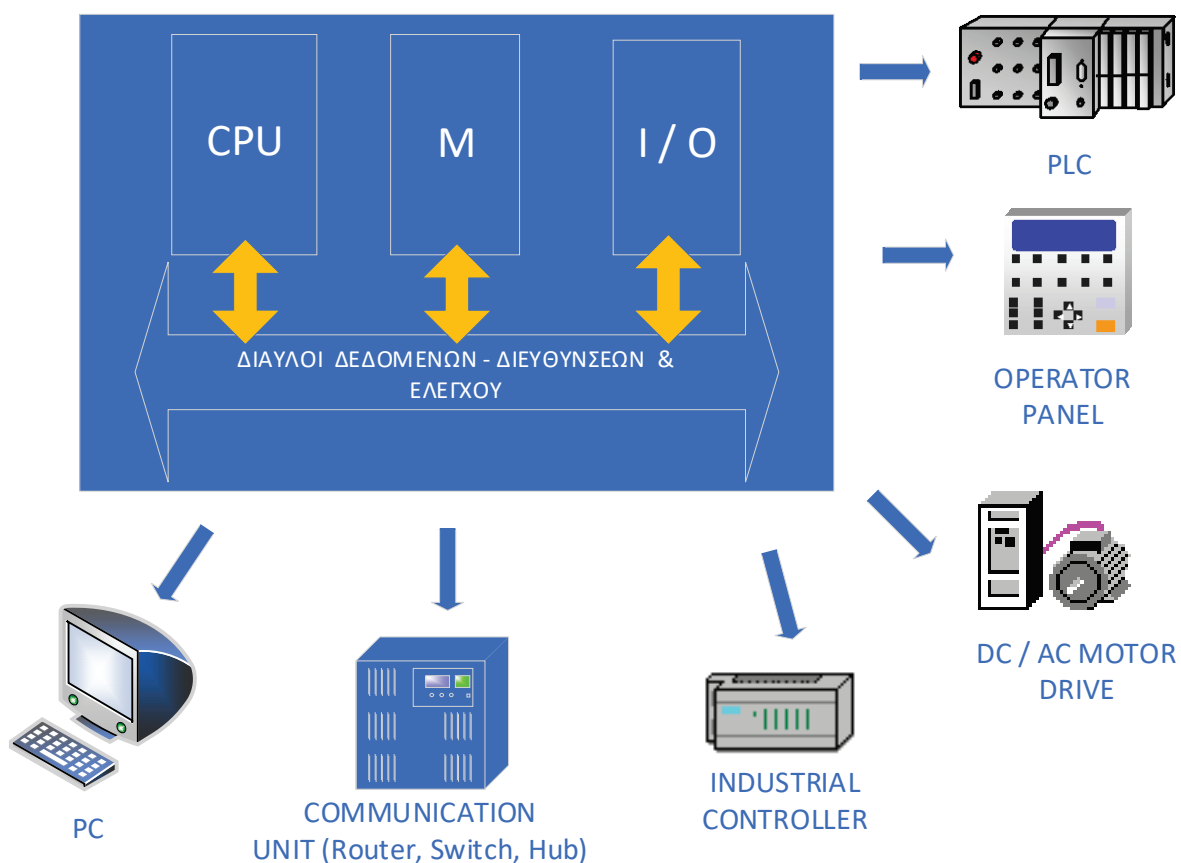
1.6 Εφαρμογές των μικροϋπολογιστών

Ο μικροϋπολογιστής μέσω του προγραμματισμού του από την γλώσσα Assembly μπορεί να κάνει διάφορες εργασίες είτε

«Εσωτερικές» όπως πχ να προσθέτει / αφαιρεί / πολλαπλασιάζει / διαιρεί αριθμούς, να υπολογίζει τύπους, να βρίσκει μέγιστο / ελάχιστο, να μεταφέρει δεδομένα από μια περιοχή μνήμης σε μια άλλη, να λύνει εξισώσεις κλπ. Ή

«Εξωτερικές» όπως να δέχεται από τις εισόδους δεδομένα, να τα επεξεργάζεται και να παράγει νέες εξόδους μέσω των κυκλωμάτων εισόδου / εξόδου (I/O – 6522, 6532)

Στο παρακάτω σχήμα φαίνεται που μπορεί να χρησιμοποιηθεί ένας μικροϋπολογιστής ή μέσα σε ποια συστήματα ενσωματώνεται. Ο μικροϋπολογιστής είναι το βασικό ψηφιακό ηλεκτρονικό κύκλωμα ή σύστημα που έχει υπολογιστική νοημοσύνη και μπορεί να κάνει εργασίες, να παίρνει αποφάσεις, να ελέγχει κλπ. Αποτελεί λοιπόν το βασικότερο στοιχείο σε συσκευές όπως είναι οι Η/Υ, τα PLC, οι βιομηχανικοί ελεγκτές, οι δικτυακές συσκευές, τα χειριστήρια ελέγχου και οποιοδήποτε σύστημα ή συσκευή πρέπει να είναι «Έξυπνο» δηλαδή να έχει δυναμική συμπεριφορά, να αντιδρά σε ερεθίσματα και χοντρικά να «σκέφτεται», βέβαια σε «χαμηλό» επίπεδο.



Εικόνα 1.7

Κεφάλαιο 2 - ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΤΟΥ AVR AT90S8515

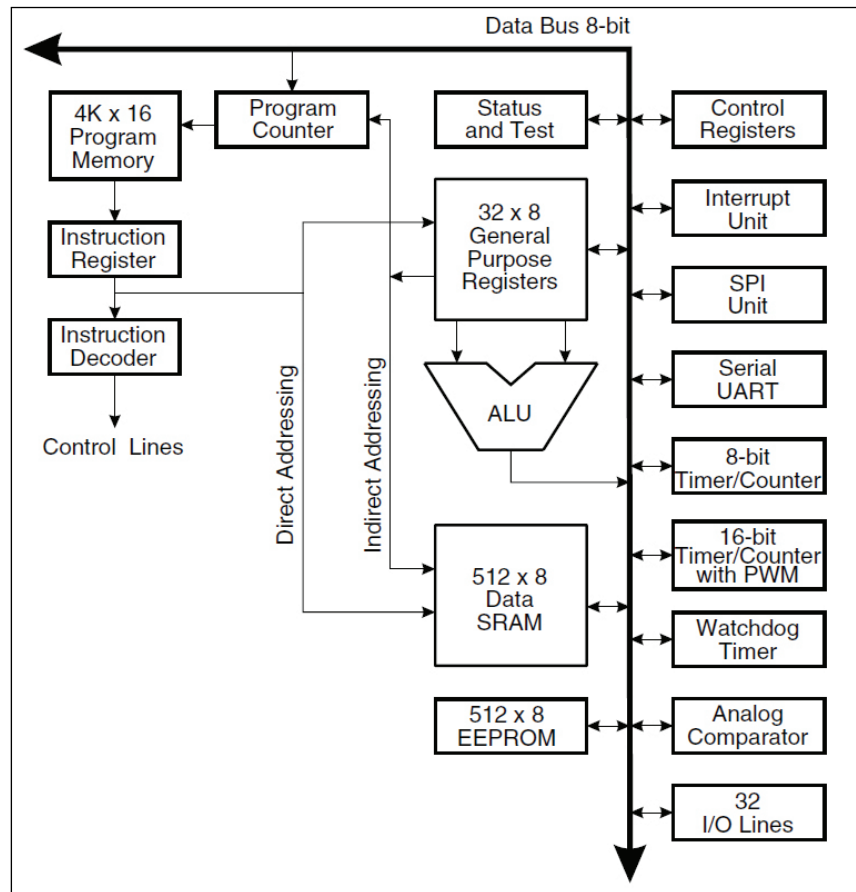
Ιστορικά ο όρος AVR αναφέρεται στην αρχιτεκτονική που αναπτύχθηκε από τους Alf-Egil Bogen και Vegard Wollan ως φοιτητές του Norwegian Institute of Technology. Η αρχιτεκτονική αυτή (η οποία βασίστηκε στην προγενέστερη αρχιτεκτονική “Harvard Architecture” και στην στρατηγική σχεδίασης CPU **Reduced Instruction Set Computing**) αφού αγοράστηκε από την εταιρία ATMEL και αποτέλεσε τη βάση για μια ευρεία γκάμα οικογενειών μικροελεγκτών οι οποίοι έγιναν ιδιαίτερα δημοφιλείς και έτυχαν ευρείας χρήσης τόσο σε επαγγελματικό επίπεδο όσο σε εκπαιδευτικό αλλά και σε απλά ερασιτεχνικό. Αξίζει να αναφερθεί ότι μετά την απήχηση που είχαν, ο όρος AVR έχει συνδεθεί με του 8bit RISC microcontrollers της ATMEL.

Κύριοι εκπρόσωποι των οικογενειών είναι οι **tinyAVR**, **megaAVR**, **90SAVR** κ.α. ενώ στα πλαίσια κάθε μιας αναπτύχθηκαν μια σειρά από συγκεκριμένα μοντέλα. Οι κυριότερες διαφορές εστιάζονται σε θέματα όπως διαθέσιμη μνήμη προγράμματος, είδος συσκευασίας το οποίο επηρεάζει το πλήθος των διαθέσιμων I/O ακροδεκτών και τις δυνατότητες επικοινωνίας με άλλες περιφερειακές συσκευές (η οποία επικοινωνία είναι από τις κύριες αποστολές ενός μικροελεγκτή). Έτσι λοιπόν η οικογένεια tinyAVR προσανατολίζεται σε εφαρμογές μικρής κλίμακας με μοντέλα που παρέχουν περιορισμένη μνήμη προγράμματος, συσκευασίες με λίγους ακροδέκτες και κατά συνέπεια περιορισμένες δυνατότητες διασύνδεσης με περιφερειακά συστήματα. Από την άλλη η οικογένεια megaAVR παρέχει μοντέλα με δυνατότητα διαχείρισης μεγάλων προγραμμάτων (μέχρι 256 kB), συσκευασίες με μέχρι και 100 ακροδέκτες και κατά συνέπεια εκτεταμένες δυνατότητες διασύνδεσης. Ενδιάμεσες δυνατότητες παρέχει η οικογένεια 90SAVR της οποίας το μοντέλο 90S8515 θα χρησιμοποιήσουμε στα πλαίσια του εργαστηρίου καθώς τα τεχνικά χαρακτηριστικά του (τα οποία εν συντομία θα αναλύσουμε παρακάτω) καλύπτουν πλήρως τις απαιτήσεις των ασκήσεων.

Σημαντική παρατήρηση είναι ότι όλοι οι AVR μικροελεγκτές μοιράζονται το ίδιο ρεπερτόριο εντολών, οργάνωση μνήμης και γενικά αρχιτεκτονικά χαρακτηριστικά και κατά συνέπεια η μετάβαση από την μια οικογένεια στην άλλη είναι ιδιαίτερα εύκολη και χωρίς μεγάλες απαιτήσεις από τον προγραμματιστή.

Στόχος του κεφαλαίου αυτού είναι να παρουσιάσει με συνοπτικό αλλά και μεστό τρόπο τα βασικά αρχιτεκτονικά χαρακτηριστικά του AT90S8515 ATMEL 8-bit AVR Microcontroller ως το βασικό εργαλείο διεξαγωγής των εργαστηριακών ασκήσεων του μαθήματος Μικροϋπολογιστές II. Έμφαση δίνεται σε βασικά στοιχεία τα οποία θα πρέπει να κατανοεί ο φοιτητής και θα χρειαστεί στα πλαίσια των ασκήσεων. Σε καμία περίπτωση δεν υποκαθιστά τα επίσημα τεχνικά εγχειρίδια που η ίδια η εταιρία παρέχει δωρεάν σε κάθε ενδιαφερόμενο.

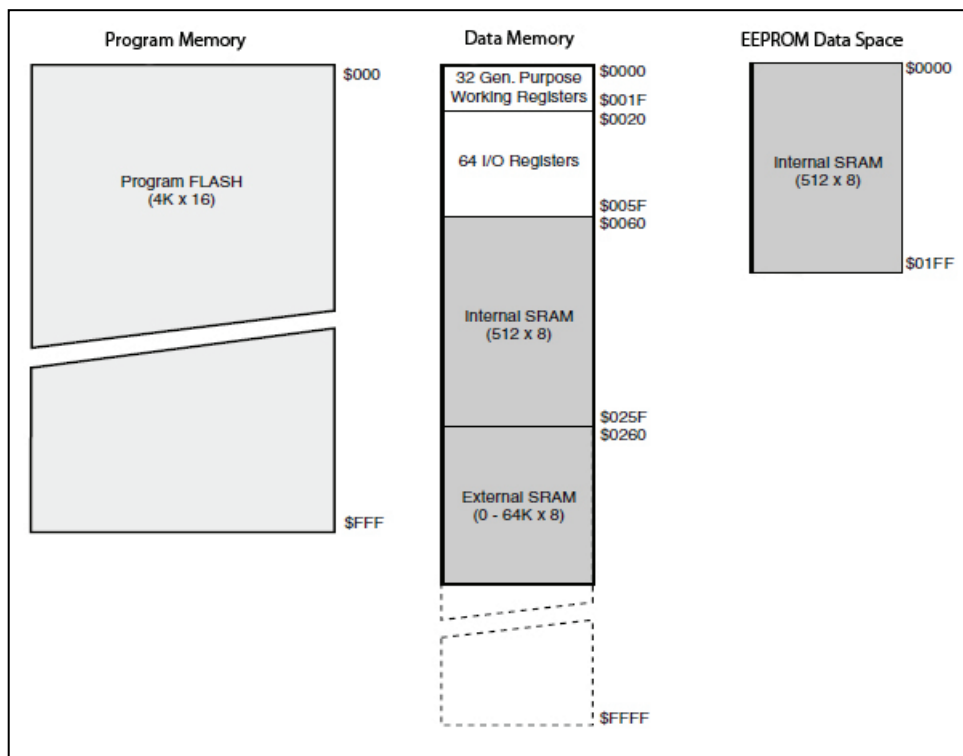
Στο σχήμα που ακολουθεί παρουσιάζονται τα βασικά τμήματα της αρχιτεκτονικής του 8515.



Εικόνα 2.1: Η AT90S8515 AVR RISC Αρχιτεκτονική (προέλευση από το επίσημο εγχειρίδιο της ATMEL)

2.1. Διαθέσιμοι Καταχωρητές

Ένα από τα κύρια χαρακτηριστικά κάθε μικροελεγκτή, καθώς επηρεάζει πολλά σημεία της γενικότερης αρχιτεκτονικής του είναι το εύρος του διαύλου δεδομένων (Data Bus όπως δηλώνεται και στην εικόνα 2.1). Όπως υποδηλώνεται από το όνομά του (8-bit AVR Microcontroller) και φαίνεται στην Εικόνα 2.1 αυτό το εύρος είναι 8 bit. Άμεση συνέπεια είναι ότι και οι καταχωρητές γενικού σκοπού που έχει στη διάθεσή του ο προγραμματιστής είναι επίσης εύρους 8 bit (1 Byte). Επιπλέον όπως φαίνεται και στο αντίστοιχο μπλοκ της εικόνας 2.1 υπάρχουν 32 διαθέσιμοι καταχωρητές. Στην Εικόνα 2.2 απεικονίζονται τα ονόματα αλλά και η χαρτογράφηση των καταχωρητών στο χώρο μνήμης δεδομένων.



Εικόνα 2.3: Οι υποστηριζόμενοι χώροι μνήμης του AT90S8515

Μνήμη Προγράμματος: Ο AT90S8515 παρέχει 8kByte On-Chip In-system Programmable Flash Memory για την αποθήκευση των προγραμμάτων. Καθώς όλες οι εντολές είναι εύρους 16 ή 32 bit η μνήμη έχει οργανωθεί σε θέσεις εύρους 16bit. Κατά συνέπεια η αντίστοιχη μνήμη έχει 4K θέσεις. Το τελευταίο χαρακτηριστικό καθορίζει και το εύρος του Program Counter (PC) το οποίο είναι 12 bit καθώς η μέγιστη θέση που πρέπει να μπορεί να δείξει είναι η \$FFF.

Μνήμη Δεδομένων: Ο χώρος διευθυνσιοδότησης δεδομένων όπως φαίνεται και από την Εικόνα 2.3 αποτελείται από τέσσερα διαφορετικά κομμάτια. Το εύρος κάθε θέσης είναι 8 bit. Τα τέσσερα τμήματα έχουν ως εξής

Οι 32 πρώτες θέσεις (\$0000-\$001F) αντιστοιχούν στους 32 γενικού σκοπού καταχωρητές.

Οι επόμενες 64 θέσεις (\$0020-\$005F) περιέχουν τις διευθύνσεις για περιφερειακά συστήματα όπως τις θύρες επικοινωνίας (I/O Ports στα οποία θα αναφερθούμε και αργότερα), Καταχωρητές ελέγχου, Χρονιστές, Μετρητές κ.α.

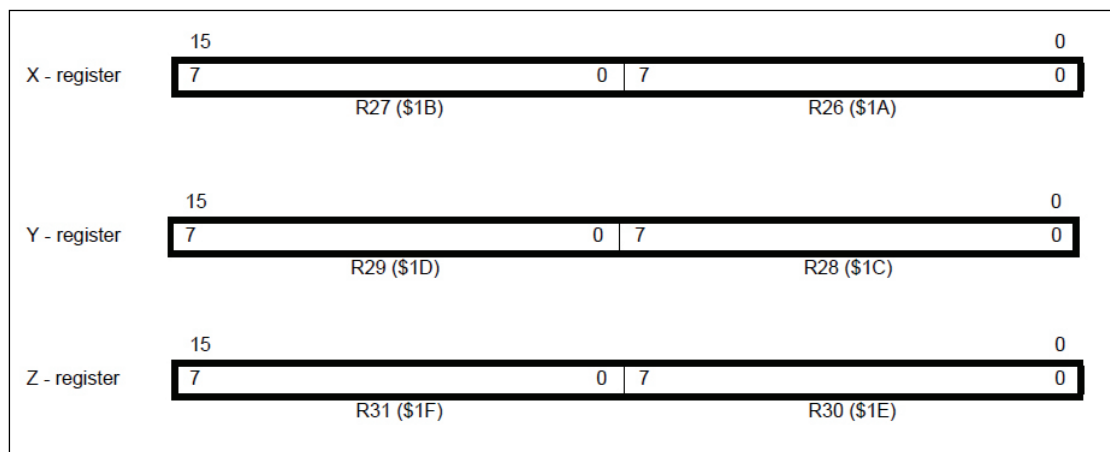
Κατόπιν ακολουθούν 512 θέσεις (\$0060-\$025F) ανήκουν στην εσωτερική SRAM μνήμη.

Τέλος σε περίπτωση που υπάρχει, μπορούν να διευθυνσιοδοτηθούν άλλες 64k θέσεις εύρους 8bit εξωτερικής SRAM μνήμης. Πρόσβαση στη μνήμη αυτή απαιτεί έναν επιπλέον κύκλο ρολογιού επεξεργαστή σε σχέση με την εσωτερική μνήμη.

Σημαντική Παρατήρηση: Από τα παραπάνω γίνεται εύκολα κατανοητό ότι για να μπορούμε να προσπέλασουμε όλο το εύρος της Data Memory πρέπει να έχουμε ένα τρόπο να ορίσουμε ένα αριθμό από το \$0000 μέχρι το \$FFFF ή με άλλα λόγια ένα

αριθμό εύρους 16bit (ή αλλιώς 2 Byte). Πως μπορούμε να ορίσουμε έναν τέτοιο αριθμό; Η πρώτη σκέψη κάποιου θα ήταν να χρησιμοποιήσει ένα από τους καταχωρητές γενικού σκοπού. Κάτι τέτοιο δεν είναι σωστό καθώς το εύρος του αριθμού που μπορεί να ορίσει ένας καταχωρητής του 8515 είναι μόνο 8bit (ή αλλιώς 1 Byte). Η αμέσως επόμενη σκέψη είναι να χρησιμοποιηθεί ένα ζευγάρι καταχωρητών καθώς μαζί μπορούν να εκφράσουν αριθμό εύρους 2 Byte. Αυτή είναι και η σωστή προσέγγιση.

Συγκεκριμένα οι τελευταίοι 6 καταχωρητές (R26-R31) έχουν μια επιπρόσθετη κρίσιμη λειτουργικότητα καθώς ανά ζευγάρια σχηματίζουν τους λογικούς 16bit καταχωρητές X, Y, Z οι οποίοι επιτρέπουν στον προγραμματιστή να προσπελάζει όλο το εύρος της μνήμης δεδομένων. Η λειτουργία αυτή απεικονίζεται στην Εικόνα 2.4.



Εικόνα 2.4: Η δημιουργία των λογικών 16bit καταχωρητών X, Y, Z από ζευγάρια των καταχωρητών γενικού σκοπού R26-R31

Μνήμη EEPROM: Ο AT90S8515 παρέχει 512 byte για δεδομένα σε μνήμη τύπου EEPROM. Η μνήμη αυτή έχει οργανωθεί σε ξεχωριστό χώρο διευθυνσιοδότησης και η προσπέλασή της γίνεται με τη χρήση συγκεκριμένων καταχωρητών διεύθυνσης, δεδομένων και ελέγχου (EEPROM address, data και control registers). Η πρόσβαση στη περιοχή διευθυνσιοδότησης της μνήμης αυτής γίνεται ένα byte τη φορά. Λόγω αυτών αλλά και επιπλέον λόγων η πρόσβαση στη μνήμη αυτή είναι πολύ αργή (απαιτεί πολλούς κύκλους ρολογιού επεξεργαστή) αλλά σαν μνήμη είναι ιδιαίτερα ανθεκτική σε επανεγγραφές.

2.3. Υποστήριξη Στοιβάς

Όπως πρέπει να είναι γνωστό και από προηγούμενα μαθήματα η στοίβα αποτελεί μια ιδιαίτερη περιοχή μνήμης ιδανική για την προσωρινή αποθήκευση δεδομένων (π.χ. σε περιπτώσεις που θέλουμε να προστατεύσουμε τα δεδομένα καταχωρητών από πιθανή αλλοίωση ή σε περιπτώσεις περάσματος παραμέτρων σε υπορουτίνες κ.α.). Επίσης ο ρόλος της είναι ιδιαίτερα κρίσιμος σε περιπτώσεις διαχείρισης interrupts και καλεσμάτων υπορουτινών καθώς εκεί αποθηκεύονται η διευθύνσεις επιστροφής του Program Counter (PC). Στην περίπτωση του AT90S8515 ακολουθείται μια πολύ

ευέλικτη τακτική. Η περιοχή αυτή μνήμης δηλώνεται στα πλαίσια της SRAM Data Memory. Κατά αυτό τον τρόπο το μέγεθος της στοίβας περιορίζεται μόνο από το μέγεθος του SRAM μνήμης καθώς και τη χρήση σε εκάστοτε περίπτωση. Επιπλέον είναι κατανοητό ότι κάθε πρόγραμμα που επιθυμεί να χρησιμοποιήσει στοίβα θα πρέπει να δηλώσει και το αντίστοιχο χώρο κατάλληλα. Τέλος ο αντίστοιχος Stack Pointer είναι εύρους 16bit και προσπελάσιμος τόσο για διάβασμα όσο και γράψιμο μέσα από το I/O χώρο διευθύνσεων.

2.4. Αριθμητική και Λογική Μονάδα

Η Αριθμητική και Λογική Μονάδα (ALU Unit) αποτελεί την καρδιά κάθε αρχιτεκτονικής μικροεπεξεργαστή ή μικροελεγκτή καθώς εκεί είναι που γίνονται οι πραγματικές πράξεις (αριθμητικές και λογικές) που στο τέλος την ημέρας υλοποιούν την επιθυμητή λειτουργία. Γίνεται, λοιπόν, εύκολα κατανοητό ότι τεχνικές που εφαρμόζονται στην συγκεκριμένη μονάδα επηρεάζουν δραστικά την απόδοση όλου του συστήματος. Κάτι τέτοιο γίνεται και στην περίπτωση του 8515 με την ALU να συνδέεται με όλους τους καταχωρητές γενικού σκοπού (όπως φαίνεται και στην Εικόνα 2.1). Αυτό το χαρακτηριστικό δίνει μεγάλη ευελιξία στον προγραμματιστή καθώς μπορούν να γίνουν πράξεις μεταξύ οποιονδήποτε 2 καταχωρητών αλλά και πράξεις μεταξύ οποιουδήποτε καταχωρητή και απόλυτου αριθμού κάτι που δεν υποστηριζόταν σε παλιότερες αρχιτεκτονικές. Επιπλέον το χαρακτηριστικό αυτό σε συνδυασμό με τις εντολές τύπου RISC δίνουν τη δυνατότητα να γίνει μια πλήρης λειτουργία της ALU σε έναν και μόνο κύκλο ρολογιού επεξεργαστή κάτι που αυξάνει δραματικά την απόδοση της μονάδας. Για λόγους πληρότητας πρέπει να αναφερθεί ότι για να επιτευχθεί η παραπάνω λειτουργία χρησιμοποιείται ακόμα μια βασική τεχνική που κληρονομήθηκε από την Harvard Architecture. Σύμφωνα με αυτή τη προσέγγιση χρησιμοποιούνται διαφορετικές μνήμες και δίαυλοι για δεδομένα και πρόγραμμα. Το πρόγραμμα εκτελείται σε pipeline δύο σταδίων. Αυτό το χαρακτηριστικό επιτρέπει στον 8515 να προ-φορτώνει μια εντολή ενώ η προηγούμενη εκτελείται και έτσι είναι δυνατόν μια πλήρης ενέργεια της ALU να γίνει σε έναν και μόνο κύκλο ρολογιού του επεξεργαστή. Οι τρεις βασικές ομάδες λειτουργιών που εκτελεί η ALU είναι αριθμητικές πράξεις, λογικές πράξεις και πράξεις ανά bit.

2.5. Θύρες Επικοινωνίας με Περιφερειακές Συσκευές

Σε έναν μικροελεγκτή σε πολλές περιπτώσεις οι δυνατότητες επικοινωνίας που παρέχει είναι το ίδιο σημαντικό με την επεξεργαστική ικανότητα ή και την διαθέσιμη μνήμη. Είναι άλλωστε χαρακτηριστικό ότι μια σημαντική διαφοροποίηση μεταξύ των διαφόρων οικογενειών των AVR μικροελεγκτών είναι ακριβώς το πλήθος των διαθέσιμων I/O ακροδεκτών και κατά συνεπεία η δυνατότητα επικοινωνία με περιφερειακές συσκευές. Σε αυτό το τομέα ο AT90S8515, όπως φαίνεται και στην Εικόνα 2.1, παρέχει 32 I/O γραμμές παρέχοντας έτσι μεγάλες δυνατότητες διασύνδεσης. Οι γραμμές αυτές ομαδοποιούνται σε 4 Θύρες Εισόδου/Εξόδου

(Input/Output Ports) και κατά συνέπεια κάθε θύρα (Port A, B, C, D) παρέχει 8 δικατευθυντήριες γραμμές εισόδου εξόδου.

Η προσπέλαση στις θύρες αυτές γίνεται μέσα από τρεις καταχωρητές ελέγχου που αντιστοιχούν σε συγκεκριμένες διευθύνσεις.

Ο πρώτος καταχωρητής συμβολίζεται ως DDRx (όπου x η θύρα που θέλουμε να ελέγξουμε) και κάθε ένα από τα 8 bit του καθορίζουν την κατεύθυνση (είσοδος ή έξοδος) που θα έχει ο αντίστοιχος ακροδέκτης της θύρας που ελέγχει. Θέτοντας 1 σε κάποιο bit ο αντίστοιχος ακροδέκτης του PORTx διαμορφώνεται ως έξοδος ενώ με 0 λειτουργεί ως είσοδος.

Ο δεύτερος καταχωρητής συμβολίζεται ως PORTx και περιέχει τα δεδομένα που πρόκειται να εγγραφούν στους ακροδέκτες τις αντίστοιχης θύρας που έχουν διαμορφωθεί ως έξοδοι.

Ο τρίτος καταχωρητής συμβολίζεται ως PINx και περιέχει τα δεδομένα που διαβάζονται από τους ακροδέκτες τις αντίστοιχης θύρας που έχουν διαμορφωθεί ως είσοδοι.

Στα πλαίσια των ασκήσεων θα χρησιμοποιήσουμε τις θύρες επικοινωνίας τόσο σας εισόδους για να διαβάζουμε την κατάσταση των διακοπών του STK500 αλλά και σαν εξόδους για να οδηγούμε τα αντίστοιχα LEDS. Στις αντίστοιχες ασκήσεις θα δοθούν και περισσότερες λεπτομέρειες για να τη χρήση τους.

2.6. Επίλογος

Όπως αναφέρθηκε και στην εισαγωγή στόχος του κεφαλαίου είναι να κάνει μια συνοπτική και κατανοητή εισαγωγή με έμφαση στις μονάδες και τα χαρακτηριστικά που σχετίζονται άμεσα με τις ασκήσεις που θα καλεστούν να εκτελέσουν οι φοιτητές στα πλαίσια του μαθήματος Μικροϋπολογιστές II. Σε μια τέτοια προσπάθεια είναι αναπόφευκτο ότι σημαντικά τμήματα και σημαντικές δυνατότητες ενός ισχυρού και ευέλικτου μικροελεγκτή όπως είναι ο AT90S8515 δεν θα παρουσιαστούν με ανάλογη λεπτομέρεια. Για λόγους πληρότητας κάποια τέτοια χαρακτηριστικά αναφέρονται επιγραμματικά εδώ.

Υποστήριξη Διακοπών (Interrupt Handling – Interrupt Unit): Ο AT90S8515 υποστηρίζει τόσο εσωτερικές όσο και εξωτερικές διακοπές (interrupts) έτσι ώστε να αντιδρά άμεσα σε μια ευρεία γκάμα γεγονότων προερχόμενα τόσο από τα εσωτερικά του υποσυστήματα όσο και από τον εξωτερικά περιφερειακά. Υποστηρίζονται 12 διαφορετικές πηγές διακοπών επιπλέον της κύρια διακοπής επανέναρξης (RESET). Κάθε μια από τις διακοπές αυτές αντιστοιχίζεται σε ένα διάνυσμα το οποίο με τη σειρά του οδηγεί στη εκτέλεση της αντίστοιχης υπο-ρουτίνας εξυπηρέτησης της διακοπής (ISR).

Υποστήριξη μετρητών χρόνου και γεγονότων (Timer/Counters Unit): Ο AT90S8515 παρέχει δύο γενικού σκοπού μετρητές (ο ένας είναι εύρους 8bit και ο άλλος 16bit) οι οποίοι μπορούν να μετρούν χρόνο με βάση το εσωτερικό ρολόι του μικροελεγκτή ή γεγονότα ενεργοποιούμενος από εξωτερικό ερέθισμα μέσα από αντίστοιχο ακροδέκτη.

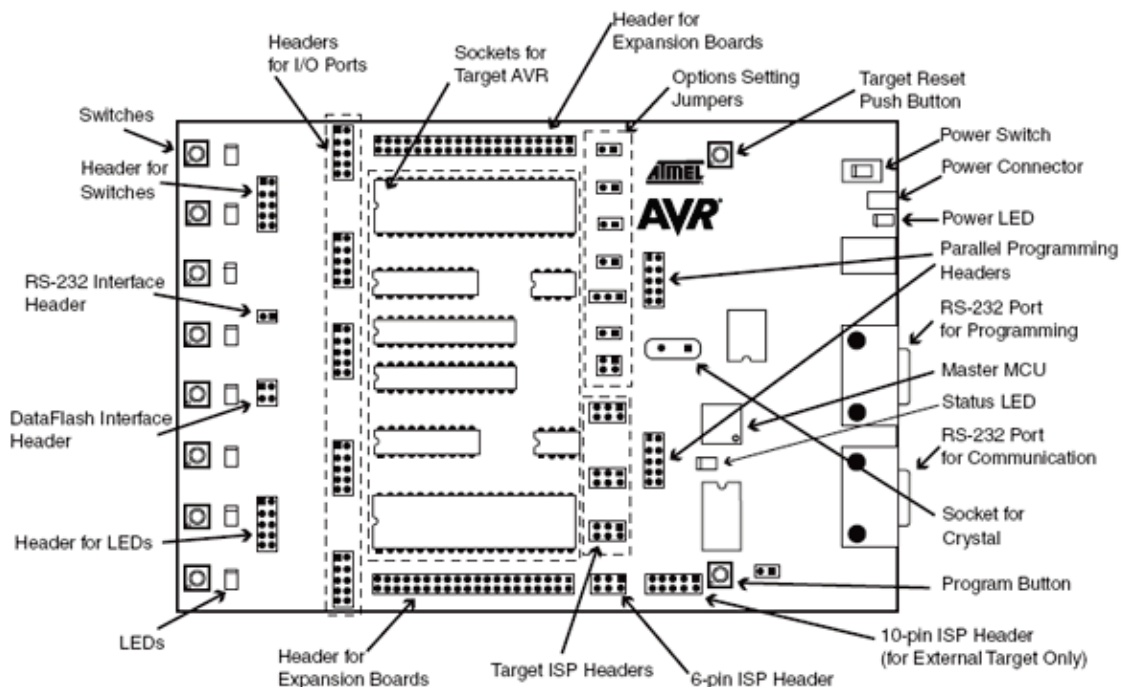
Υποστήριξη Σειριακής Επικοινωνίας (SPI Unit – Serial UART): Ο AT90S8515 παρέχει ένα Serial Peripheral Interface (SPI) το οποίο επιτρέπει σύγχρονη επικοινωνία υψηλών ταχυτήτων μεταξύ του μικροελεγκτή και περιφερειακών συσκευών ή μεταξύ AVR μικροελεγκτών και χρησιμοποιείται για το προγραμματισμό του μικροελεγκτή. Επίσης παρέχει υποστήριξη σε τυπικές σειριακές θύρες UART που χρησιμοποιούνται για επικοινωνία.

Κεφάλαιο 3 - ΠΕΡΙΓΡΑΦΗ ΤΟΥ AVR STK500 ΜΙΚΡΟΚΙΤ

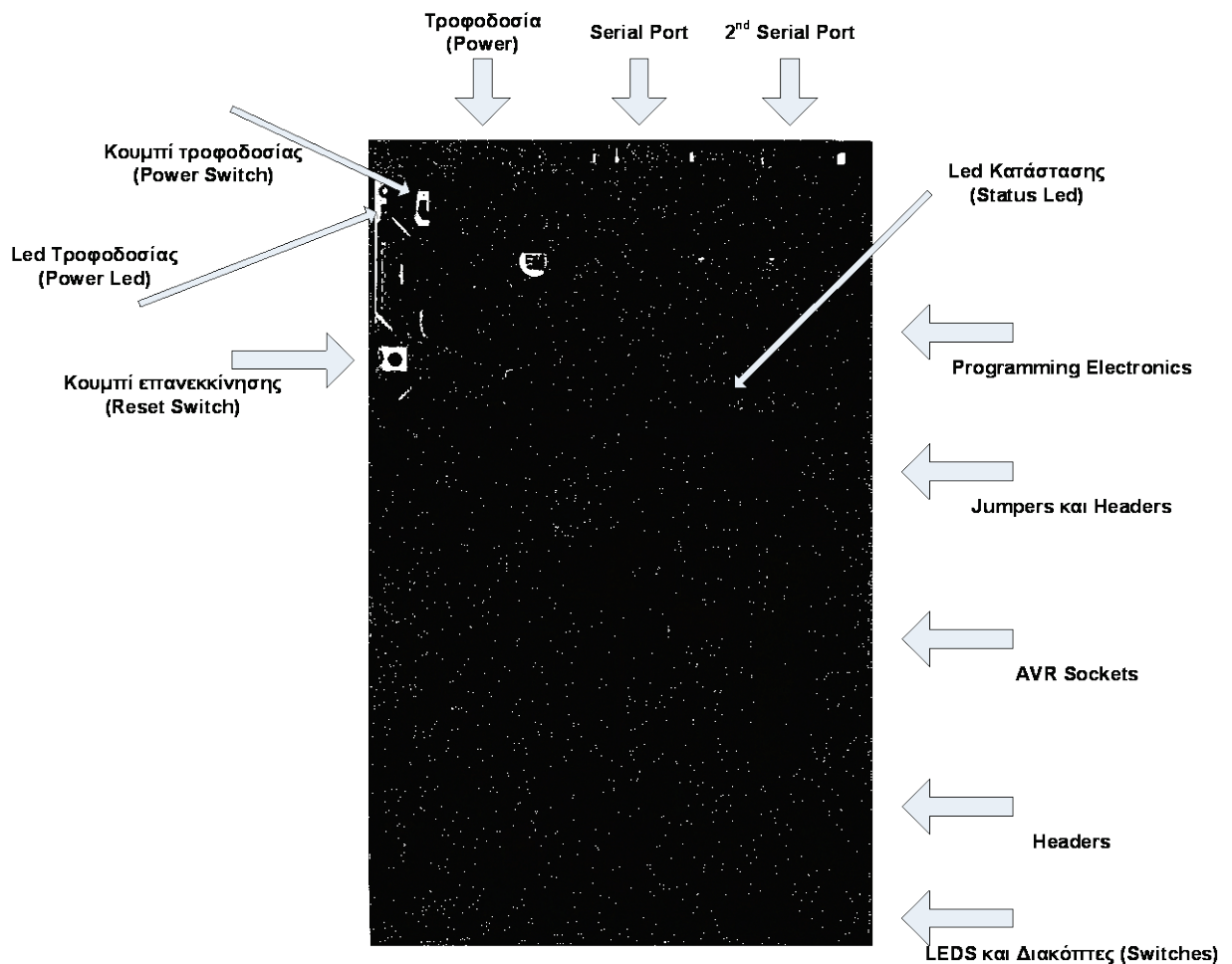
3.1 Γενικά

Το AVR STK500 ΜικροΚΙΤ είναι μία ολοκληρωμένη αναπτυξιακή εκπαιδευτική πλατφόρμα (αναπτυξιακό εκπαιδευτικό ΚΙΤ) για τον μικροελεγκτή AVR της Atmel Corporation. Έχει σχεδιαστεί για να δώσει στους εκπαιδευόμενους/σχεδιαστές την δυνατότητα γρήγορης εκπαίδευσης / ανάπτυξης κώδικα στον μικροελεγκτή AVR και την δυνατότητα ανάπτυξης πρωτοτύπων και ελέγχου νέων προγραμμάτων. Πλήρες εγχειρίδιο για το AVR STK500 ΜικροΚΙΤ ακολουθεί στα παραρτήματα. Παρακάτω ακολουθούν λίγες οδηγίες για την τροφοδοσία, τα leds και τους διακόπτες του ΜικροΚΙΤ.

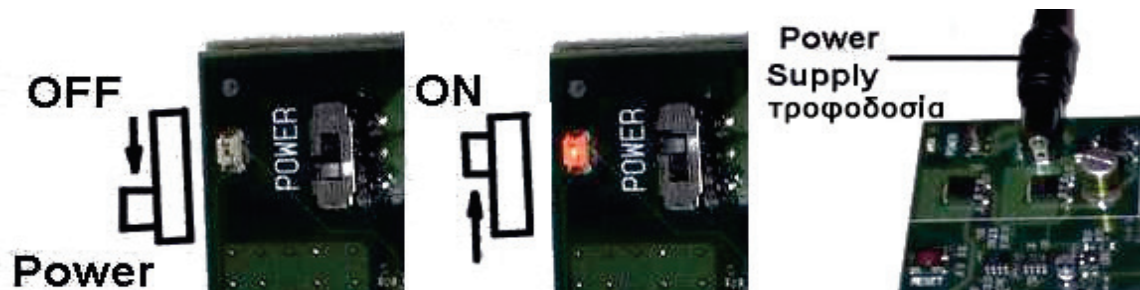
Απαιτείται μία εξωτερική 10-15V DC τροφοδοσία. Το κύκλωμα εισόδου είναι μια πλήρης ανορθωτική γέφυρα, οπότε το STK500 χειρίζεται αυτόματα τόσο θετικές ή αρνητικές συνδέσεις. Συνδέστε το καλώδιο τροφοδοσίας μεταξύ μια παροχή ηλεκτρικού ρεύματος και το STK500 (Εικόνα 3.1 και 3.2). Εφαρμόστε 10-15V DC στην υποδοχή τροφοδοσίας. Ο διακόπτης τροφοδοσίας (power switch) ανάβει ή σβήνει το ΜικροΚΙΤ (τροφοδοσία ή μη με ρεύμα) (Εικόνα 3.3). Όταν τροφοδοτείται με ρεύμα το ΜικροΚΙΤ τότε το Led τροφοδοσίας (Power Led) γίνεται κόκκινο. Ενώ το LED κατάστασης (Status Led) θα γίνει από κόκκινο, μέσω κίτρινου, πράσινο. Το πράσινο LED δηλώνει ότι η τροφοδοσία VCC είναι παρούσα (Εικόνα 3.1 και 3.2).



Εικόνα 3.1 (προέλευση: AVR STK500, AVR User guide, ATMEL)



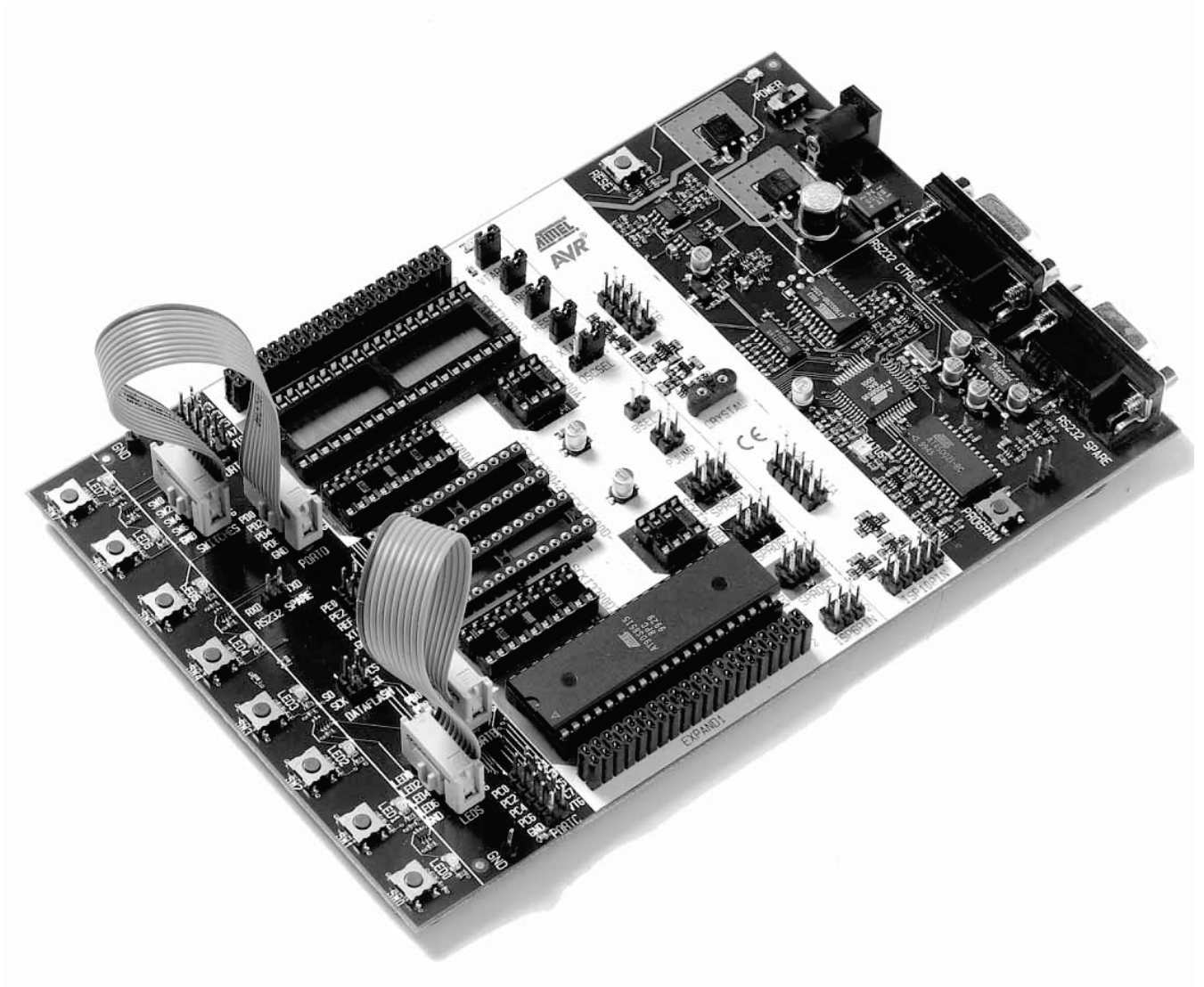
Εικόνα 3.2 (τροποποίηση από AVR STK500, AVR User guide, ATMEL)



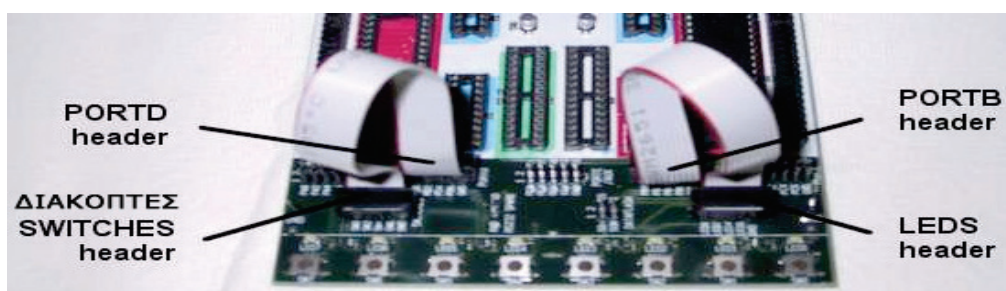
Εικόνα 3.3 (προέλευση: AVR STK500, AVR User guide, ATMEL)

3.2 Διακόπτες και Leds

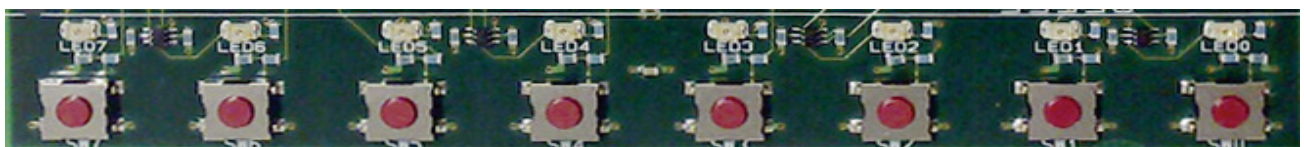
Χρησιμοποιήστε τα παρεχόμενα 10-pin καλώδια για σύνδεση στην επαφή (header) με ένδειξη "PORTB" με τον header που φέρουν την ένδειξη "LEDs", ύστερα συνδέστε την επαφή (header) με την ένδειξη "PORTD" με την επαφή (header) που φέρουν την ένδειξη "Switches - Διακόπτες". Οι συνδέσεις φαίνονται στις Εικόνες 3.4 και 3.5. Το STK500 MicroKIT περιλαμβάνει 8 κίτρινα LEDs και 8 διακόπτες (switches) (Εικόνα 3.6).



Εικόνα 3.4 (προέλευση: AVR STK500, AVR User guide, ATMEL)



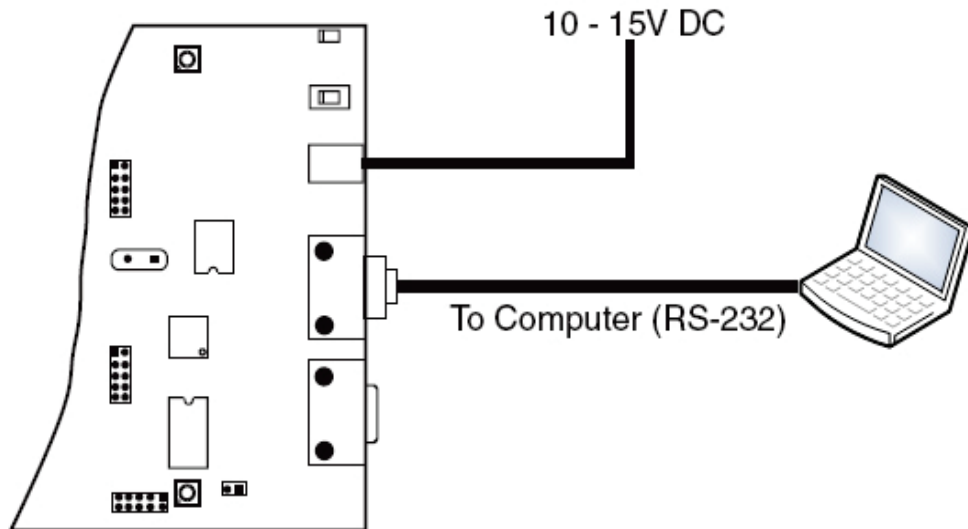
Εικόνα 3.5 (προέλευση: AVR STK500, AVR User guide, ATMEL)



Εικόνα 3.6 (προέλευση: AVR STK500, AVR User guide, ATMEL)

3.3 Σύνδεση του ΜικροKIT με τον υπολογιστή

Όπως φαίνεται και στην παρακάτω εικόνα (Εικόνα 3.7 και 3.8) θα πρέπει να συνδέσετε ένα σειριακό καλώδιο στην υποδοχή με την ένδειξη RS-232 (Serial Port) του ΜικροKIT με μία σειριακή θύρα (COM) του υπολογιστή.



Εικόνα 3.7 (τροποποίηση από AVR STK500, AVR User guide, ATMEL)

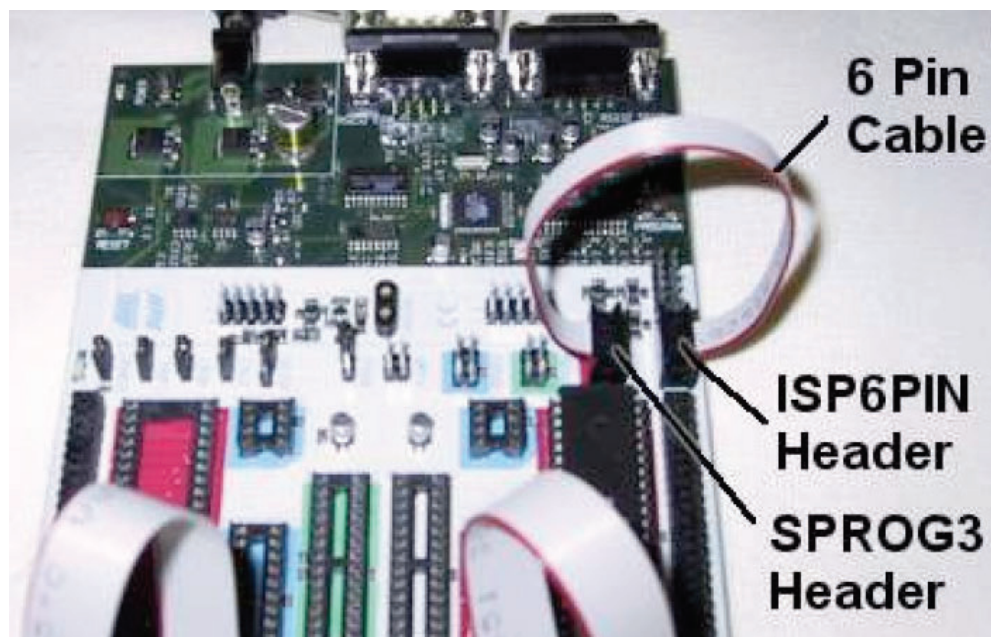


Εικόνα 3.8 (προέλευση: AVR STK500, AVR User guide, ATMEL)

3.4 Downloading του προγράμματος στο ΜικροKIT

Για να εκτελέσετε το πρόγραμμα, πρέπει να το μεταφέρετε στο STK500, και να το κάνουμε αυτό πρέπει να κάνετε τις κατάλληλες διασυνδέσεις στο ΜικροKIT. Αρχικά θα πρέπει το ΜικροKIT να μην έχει τροφοδοσία.

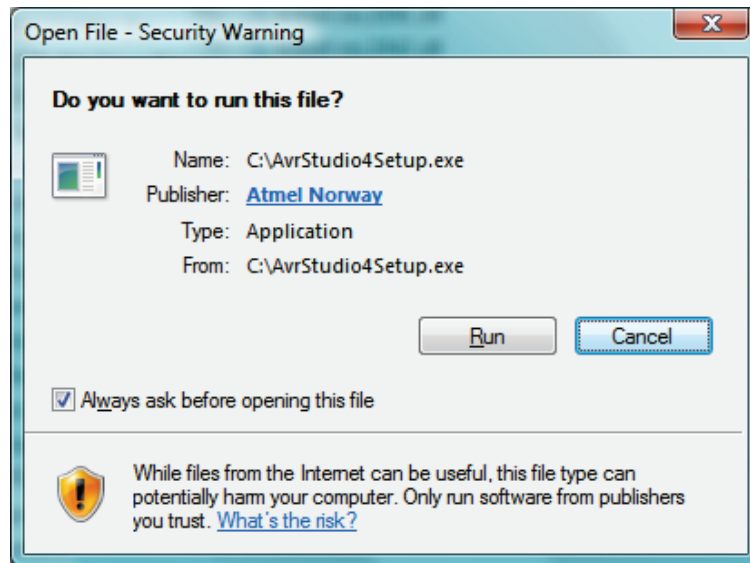
Βρείτε την καλωδιο-ταινία των 6-Pin και συνδέστε μεταξύ τους τα Pin ISP6PIN και SPROG3, τα οποία βρίσκονται πίσω ακριβώς από τον επεξεργαστή (Εικόνα 3.8). Προσοχή, διασφαλίστε ότι το καλώδιο δεν έχει συστραφεί.



Εικόνα 3.8 (προέλευση: AVR STK500, AVR User guide, ATMEL)

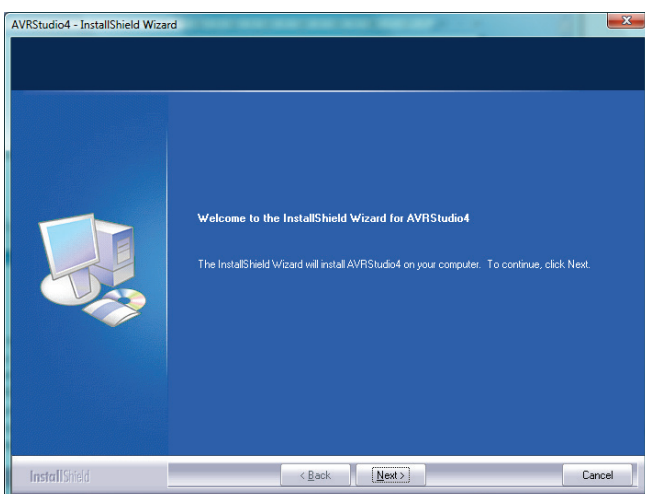
4.1 Εγκατάσταση του Λογισμικού AVR Studio 4

Για την ανάπτυξη μιας εφαρμογής στον μικροελεγκτή είναι απαραίτητος ο σχεδιασμός και η υλοποίηση του πηγαίου κώδικα. Αυτό μπορεί να γίνει με την χρήση του λογισμικού AVR Studio 4 αφού πρώτα το εγκαταστήσουμε στον Η/Υ μας. Το λογισμικό αυτό μπορείτε να το προμηθευτείτε από τον δικτυακό τόπο της εταιρίας ATMEL, <http://www.atmel.com/products/avr/> (επιλογή Tools & Software, Design Software, AVR Studio 4). Αφού κατεβάσετε το λογισμικό AVR Studio 4, τρέχετε το αρχείο “AvrStudio4Setup.exe και εμφανίζεται η παρακάτω φόρμα (Εικόνα 4.1), στην οποία πατάτε “Run”.

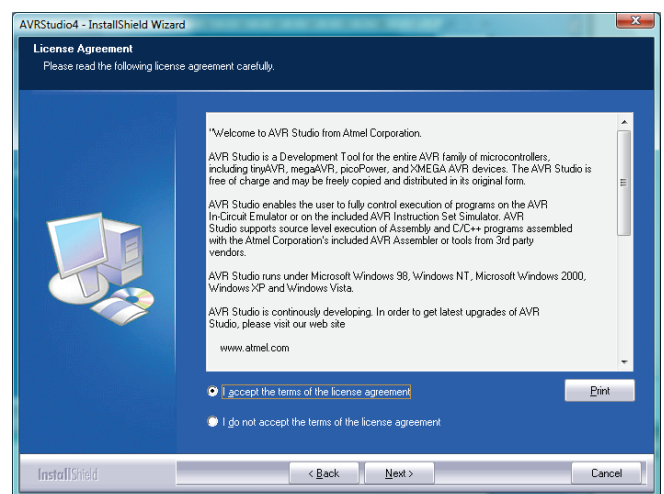


Εικόνα 4.1

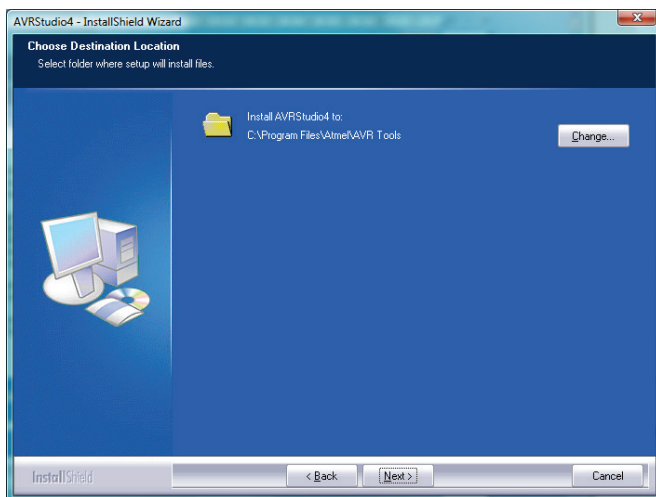
Παρακάτω ακολουθούν όλα τα βήματα (όλες οι φόρμες που παρουσιάζονται) κατά την διαδικασία εγκατάστασης του προγράμματος (Εικόνες 4.2 – 4.7). Ακολουθήστε τα βήματα ένα-ένα μέχρι να καταλήξετε στην εγκατάσταση του λογισμικού.



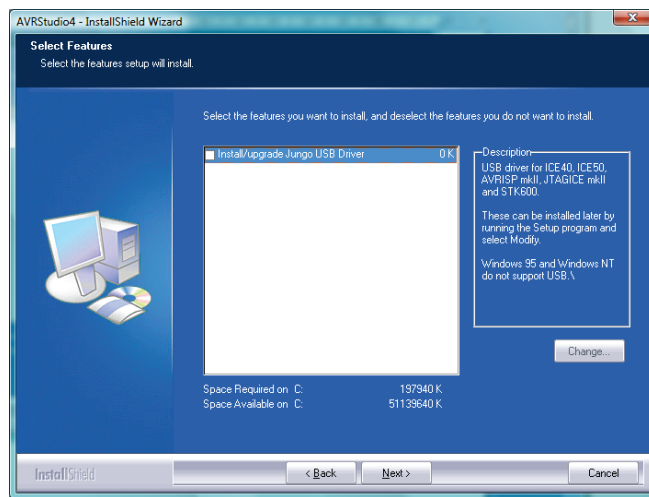
Εικόνα 4.2



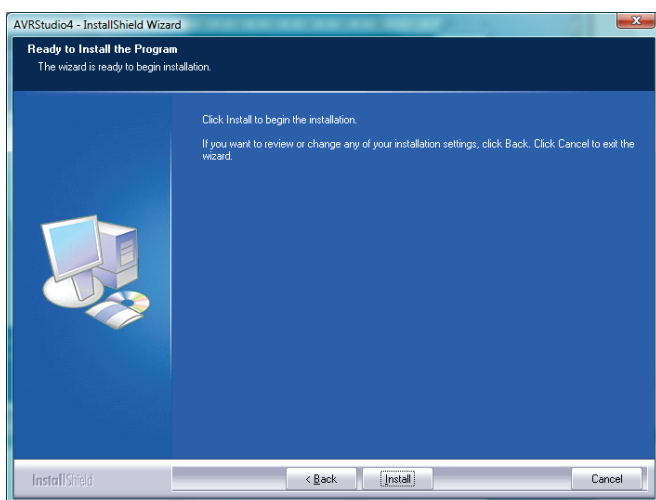
Εικόνα 4.3



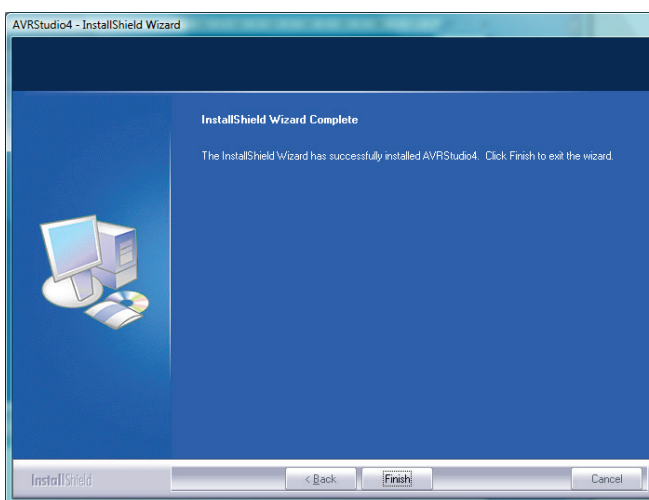
Εικόνα 4.4



Εικόνα 4.5



Εικόνα 4.6



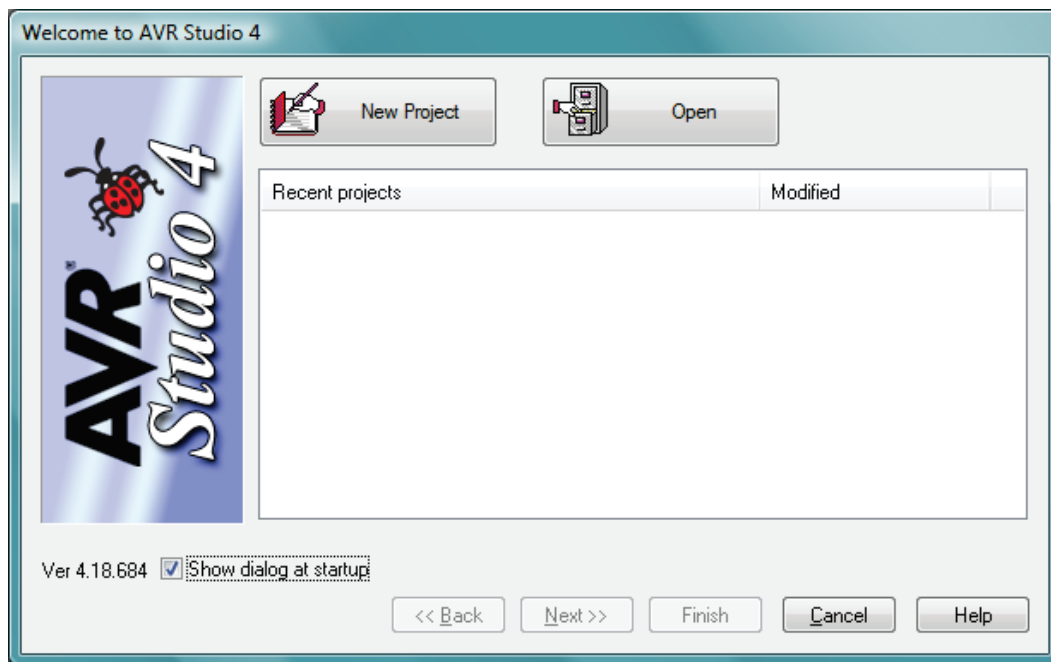
Εικόνα 4.7

4.2 Περιβάλλον του Λογισμικού AVR Studio 4

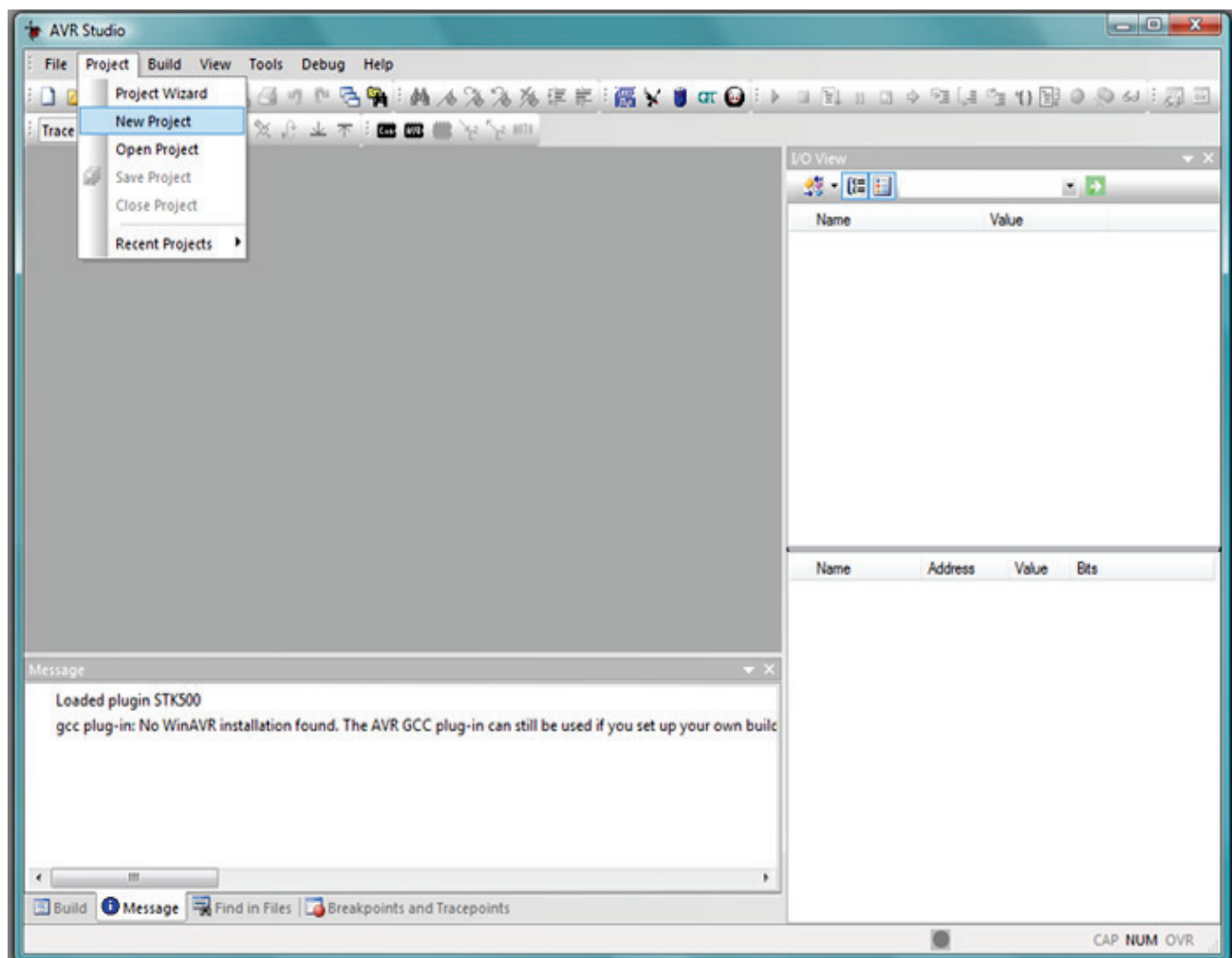
Παρακάτω θα αναφέρουμε βήμα προς βήμα την διαδικασία εκείνη που πρέπει να ακολουθηθεί για την συγγραφή και εκτέλεση ενός προγράμματος για τον AVR.

Δημιουργία νέου έργου (project)/Ορισμός παραμέτρων

Η φιλοσοφία λειτουργίας του λογισμικού AVR Studio είναι ότι κάθε πρόγραμμα αποθηκεύεται σε ένα φάκελο μαζί με όλα τα αρχεία που προκύπτουν κατά την μεταγλώττιση του. Στις εικόνες 4.8 και 4.9 βλέπουμε διαδικασία δημιουργίας Νέου Έργου (New Project). Μέσω της εικόνας 4.8, μπορούμε να επιλέξουμε κατευθείαν την επιλογή *"New Project"*. Στο παράθυρο αυτό μπορείτε δηλαδή να δημιουργήσετε ένα νέο έργο ή να επιλέξετε δουλέψετε ένα έργο που ήδη υπάρχει με το πλήκτρο *"Open"*. Μέσω της εικόνας 4.9, επιλέγουμε την επιλογή *"Project"* του μενού του AVR Studio όπου περιέχονται όλες οι λειτουργίες για την διαχείριση των έργων (project). Στην περίπτωση μας αφού θέλουμε να δημιουργήσουμε ένα νέο έργο θα επιλέξουμε την επιλογή *"New Project"*. Στο παράθυρο αυτό μπορείτε να επιλέξετε δουλέψετε ένα έργο που ήδη υπάρχει με το πλήκτρο *"Open"* ή να δημιουργήσετε ένα νέο έργο. Επιλέξτε το *"New Project"* ώστε να εμφανιστεί η οθόνη επιλογών του project .



Εικόνα 4.8

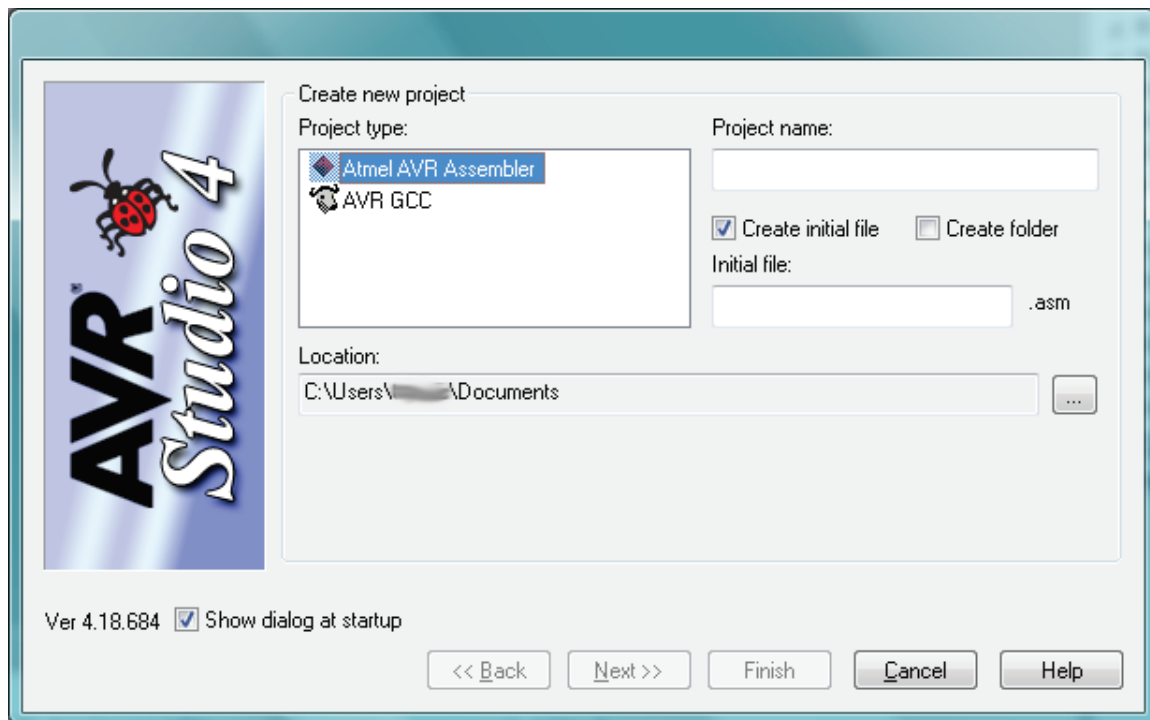


Εικόνα 4.9

Αφού επιλέξετε την δημιουργία Νέου Έργου (New Project), ύστερα θα πρέπει να ορίσετε τις παραμέτρους του έργου (project) στο οποίο θα δουλέψετε. Στην οθόνη (Εικόνα 4.10) που θα εμφανιστεί θα πρέπει να ορίσετε:

Τον τύπο του έργου (Project Type)

Σε αυτό το σημείο θα πρέπει να ορίσετε το είδος του έργου (project type) που θέλετε να δημιουργήσετε. Στην περίπτωση μας θα επιλέγουμε πάντα τύπο έργου για συμβολομεταφραστή, δηλαδή **“Atmel AVR Assembler”**. (Εικόνα 4.10)



Εικόνα 4.10

Το όνομα του έργου (Project Name)

Σε αυτό το σημείο θα πρέπει να ορίσετε το όνομα του έργου (project name) που δημιουργήσατε. Στα πλαίσια των εργαστηριακών ασκήσεων σαν όνομα έργου (project) θα δίνετε την λέξη “lab” και τον αριθμό της άσκησης που κάνετε (π.χ. Αν κάνετε την άσκηση 1 το όνομα του έργου (project) που θα δημιουργήσετε θα είναι “lab1”). (Εικόνα 4.11).

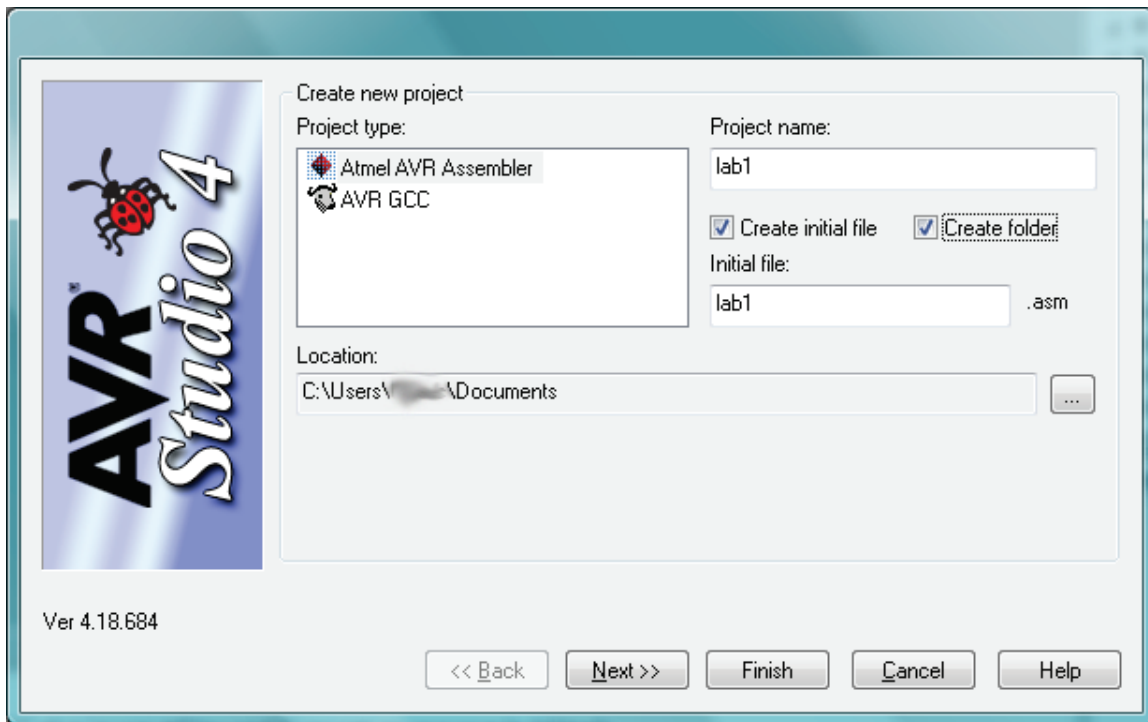
Επιλογή Initial File και Location

Όταν ορίσετε το όνομα έργου (project name), θα πρέπει να ορίσετε και όνομα στο “Initial file”, το οποίο είναι το αρχείο που θα περιέχει τον κώδικα που θα αναπτύξετε. Αν θέλετε να δημιουργεί το AVR Studio αυτόματα το αρχείο αυτό θα πρέπει να έχετε επιλέξει το κουτάκι “Create Initial File”. Σε αυτήν την περίπτωση θα βάζει στο “Initial file”, το ίδιο όνομα με το όνομα του project (Εικόνα 4.11).

Ύστερα θα πρέπει να ορίσετε τον φάκελο (directory location) στον οποίο θα δημιουργηθεί το νέο έργο (new project). Για την διεξαγωγή των εργαστηρίων οι φάκελοι των έργων (projects) θα έχουν όνομα την λέξη “lab” και το νούμερο της άσκησης. π.χ. Για την άσκηση 1 το όνομα του φακέλου θα είναι “lab1”

και θα βρίσκεται πάντα στον φάκελο “c:\Micro\MicroII”. Για να μπορέσετε να βρείτε τον φάκελο θα πρέπει να πατήσετε το κουμπί με τις τρεις τελείες, έργου (Εικόνα 4.12).

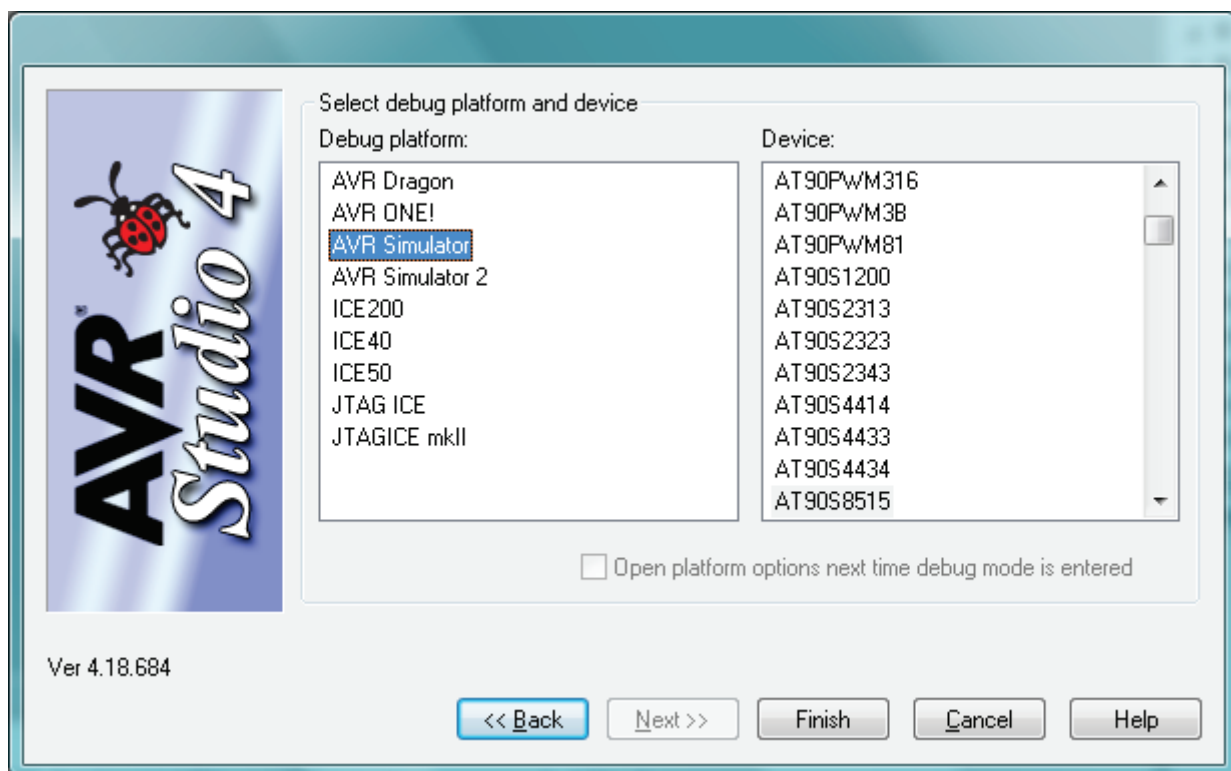
Χρησιμοποιείτε ελληνικούς χαρακτήρες στο όνομα του project, του initial file και του location.



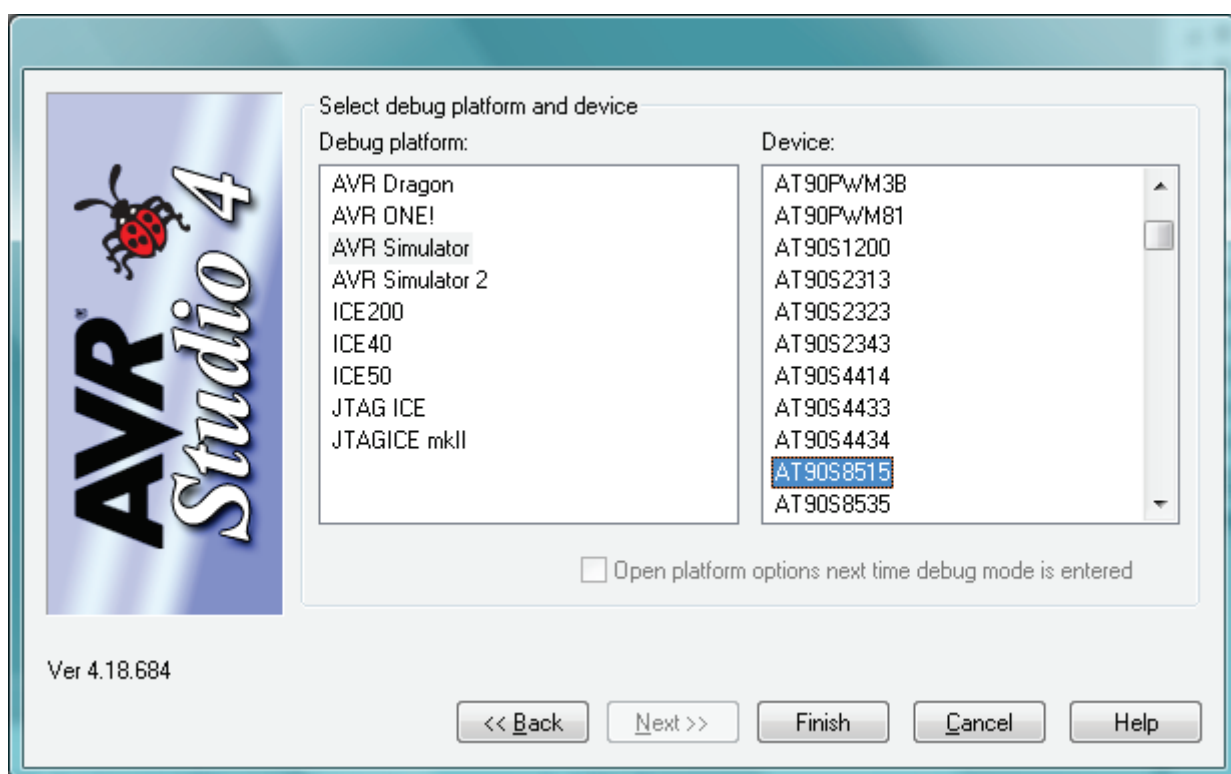
Εικόνα 4.12

Ορισμός άλλων παραμέτρων

Αφού συμπληρώσετε σωστά όλα τα στοιχεία μπορείτε να πατήσετε την επιλογή “Next” για να εμφανιστεί η οθόνη επιλογών αποσφαλμάτωσης (Debug Platform) και μοντέλου μικροελεγκτή (Device) (Εικόνα 4.13 και 4.14).. Το AVR Studio 4 μπορεί να υποστηρίξει και να χρησιμοποιηθεί από ένα ευρύ φάσμα εργαλείων αποσφαλμάτωσης. Επειδή όμως χρησιμοποιούμε τη βασική του έκδοση θα πρέπει να επιλέξετε τον εξομοιωτή “AVR simulator”(Εικόνα 4.13). και να επιλέξετε το μοντέλο AVR με το οποίο θα δουλέψετε, το οποίο στα πλαίσια του εργαστηρίου θα είναι ο AT90S8515) (Εικόνα 4.14).



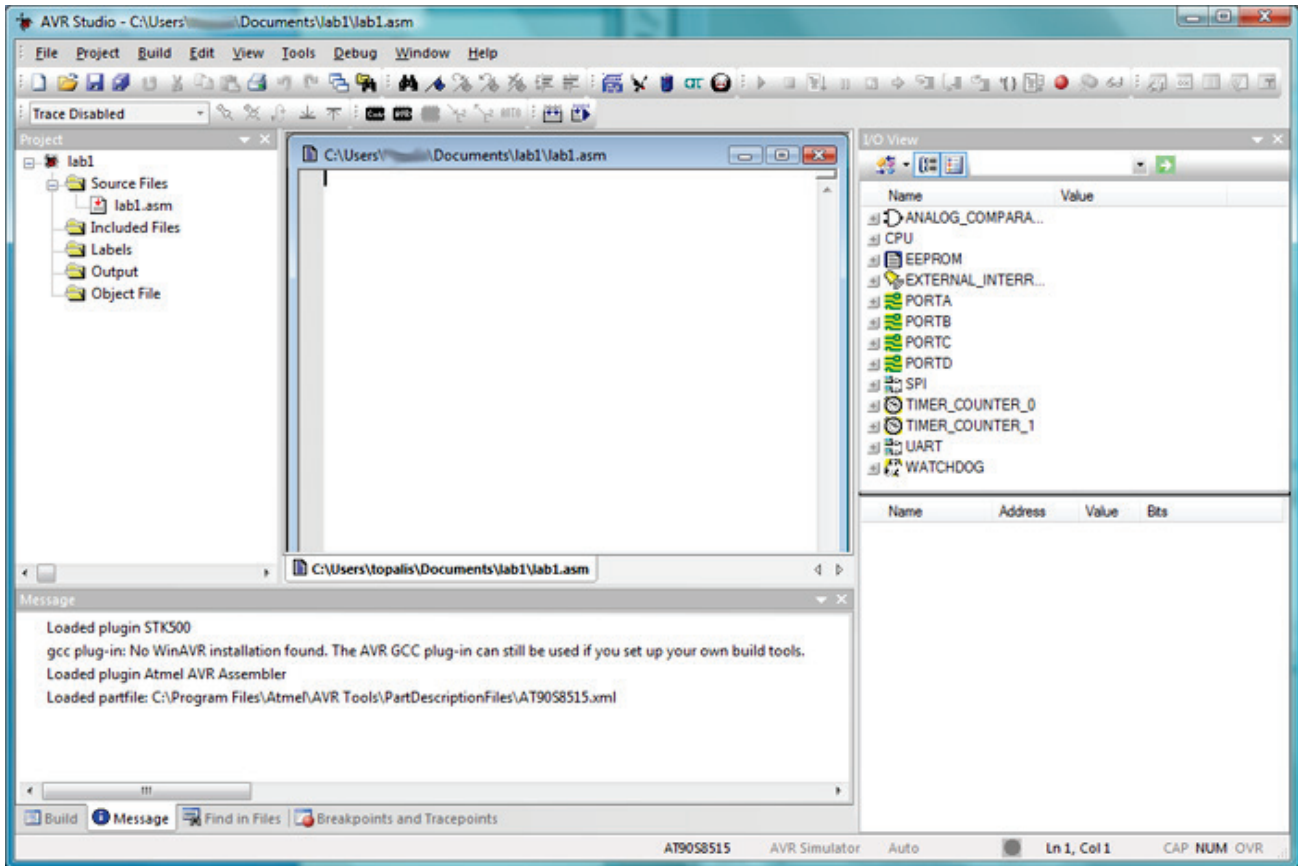
Εικόνα 4.13



Εικόνα 4.14

Αφού έχουμε ξανά ελέγξει όλες τις παραπάνω επιλογές, ύστερα πατάμε την επιλογή “Finish” και δημιουργείται το έργο (project). Οπότε εμφανίζεται η οθόνη εργασίας του

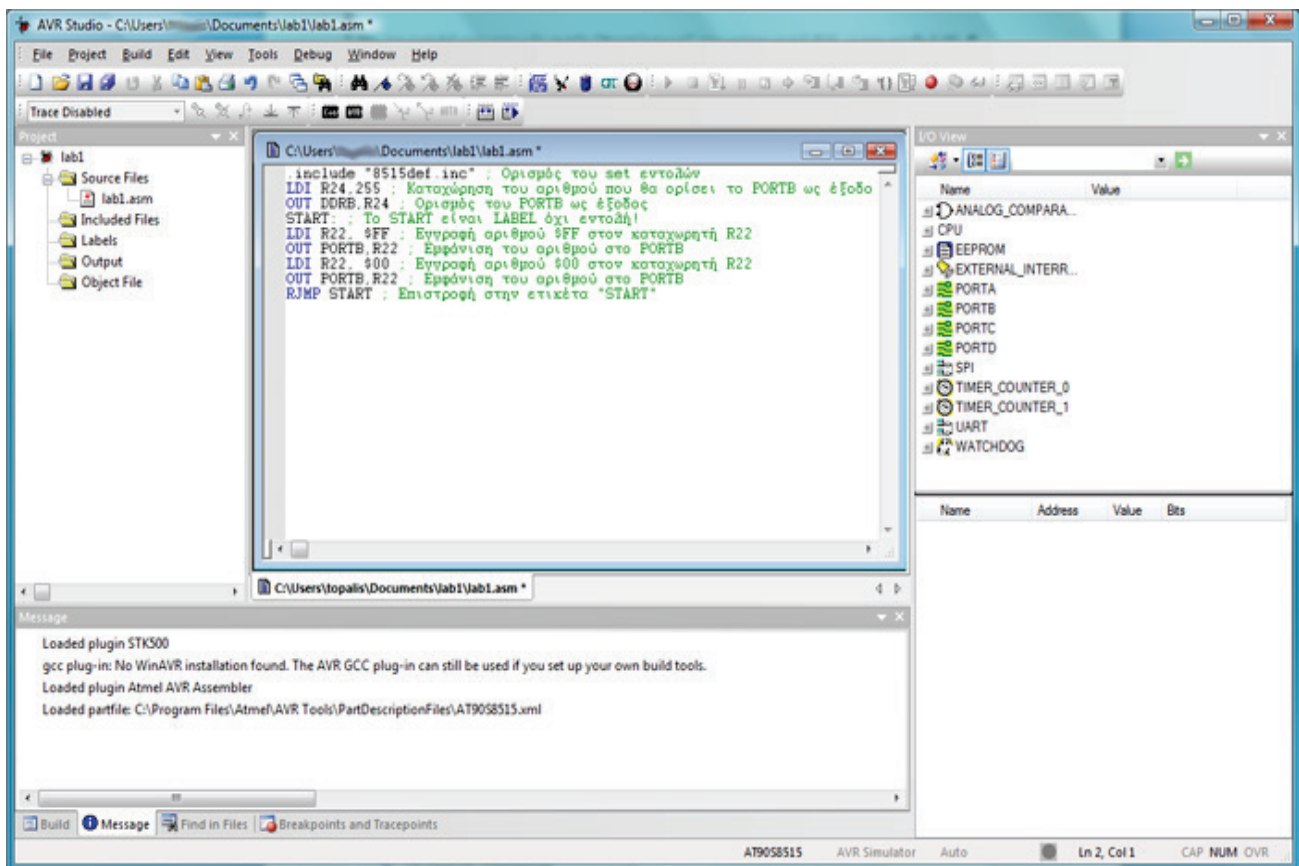
AVR Studio 4 όπου μπορείτε να γράψετε το πρόγραμμα για το συμβολομεταφραστή (Εικόνα 4.15).



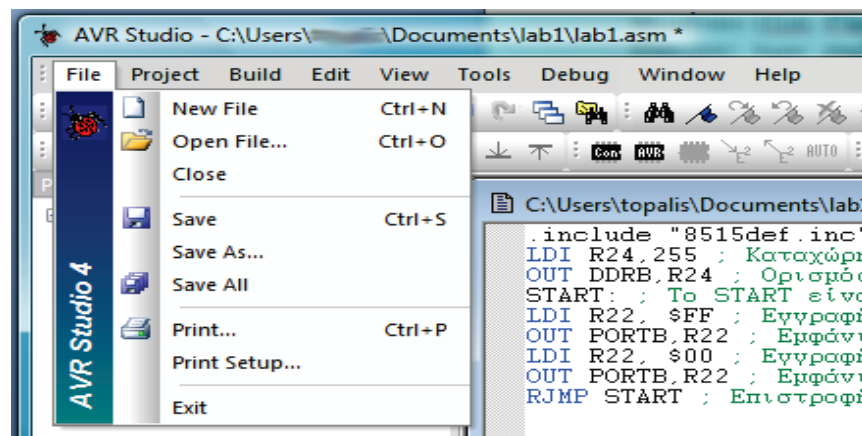
Εικόνα 4.15

Περιγραφή οθόνης εργασίας

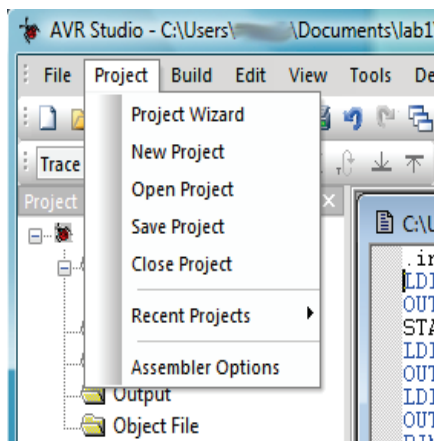
Η οθόνη εργασίας (Εικόνα 4.16) του AVR Studio 4 περιλαμβάνει γραμμή των menus όπου υπάρχουν ομαδοποιημένες όλες οι παρεχόμενες λειτουργίες του. Έτσι λοιπόν κάτω από την επιλογή "File" (Εικόνα 4.17) υπάρχουν λειτουργίες σχετικές με την διαχείριση των αρχείων (new, open, save, print, close κ.α.), στην επιλογή "Project" (Εικόνα 4.18) λειτουργίες για την διαχείριση των έργων (project), στην "Edit" (Εικόνα 4.19) για την επεξεργασία του κειμένου (Cut, Paste, κ.α.), στην "Tools" (Εικόνα 4.20) για την ενεργοποίηση εργαλείων εξομοίωσης του AVR όπως έλεγχος των register κ.α. και την επιλογή "Debug" (Εικόνα 4.21) όπου έχουμε λειτουργίες όπως η έναρξη μεταγλώττισης Build, Run, Debug κ.α.).



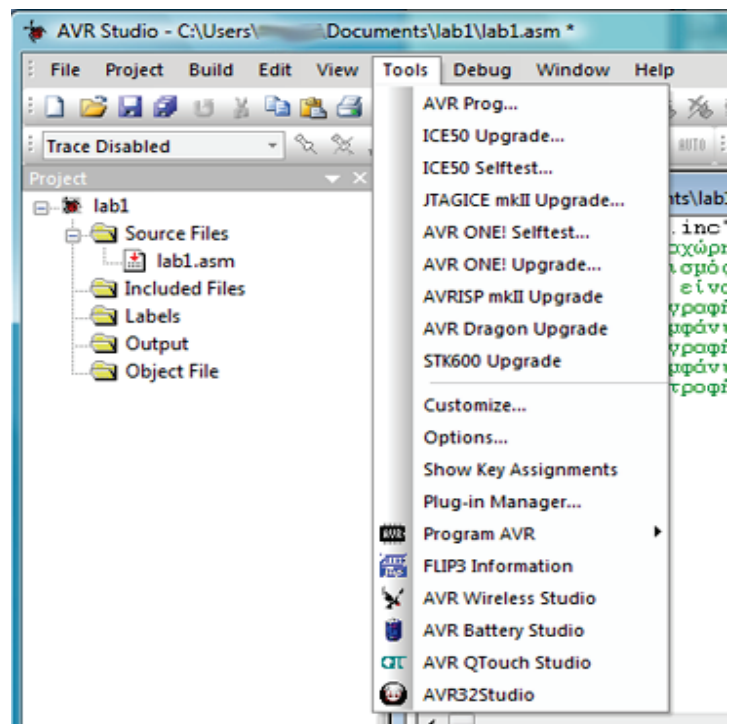
Εικόνα 4.16



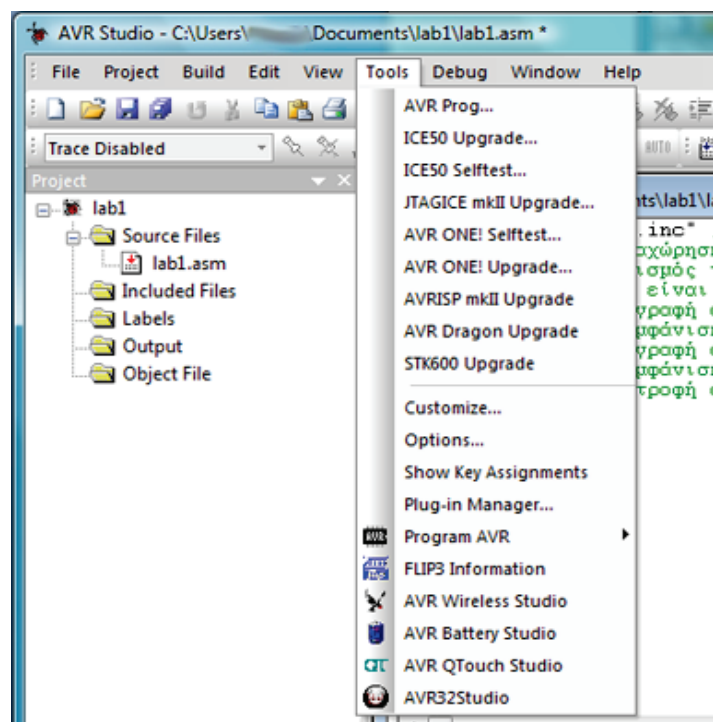
Εικόνα 4.17



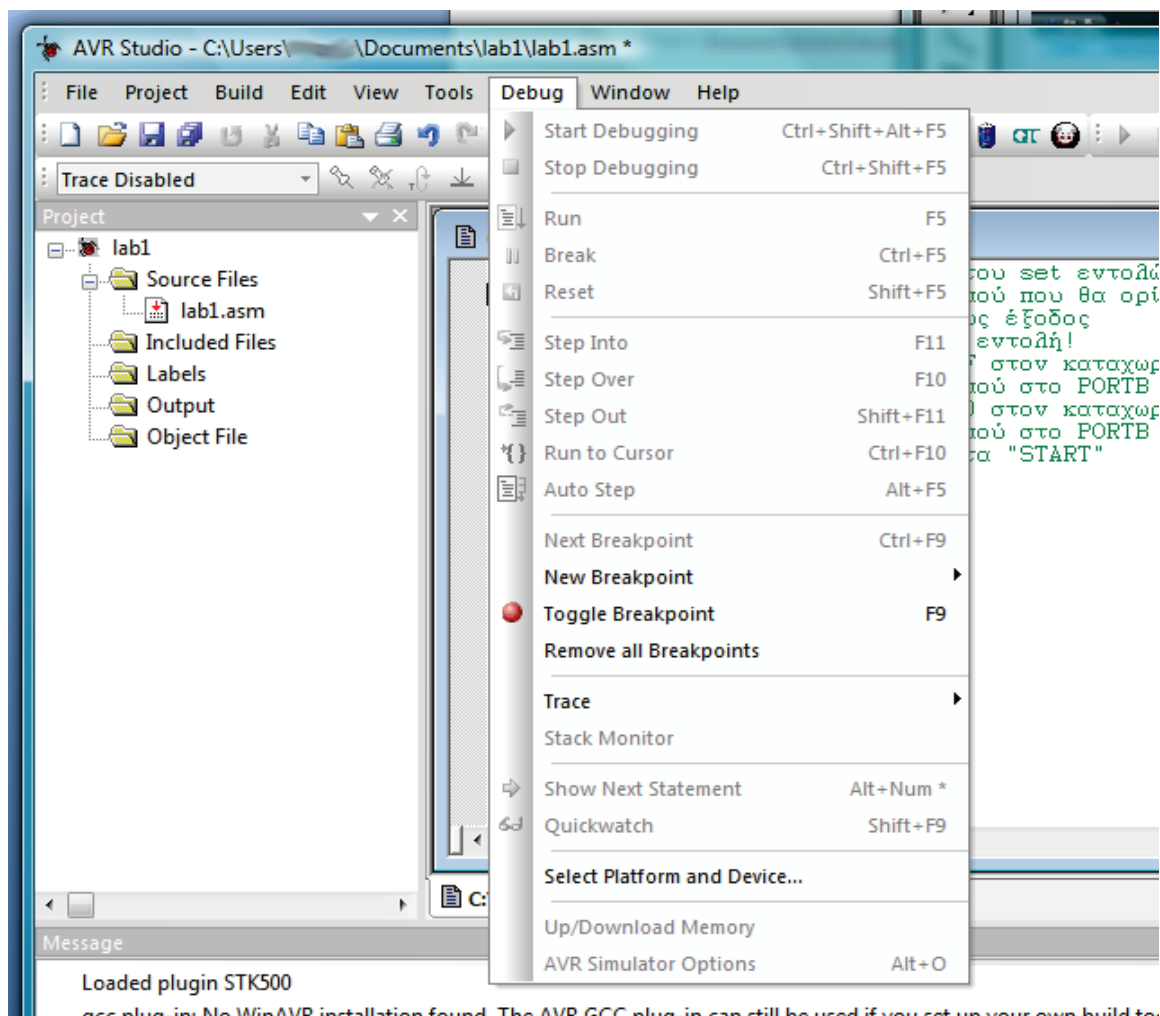
Εικόνα 4.18



Εικόνα 4.19



Εικόνα 4.20



Εικόνα 4.21

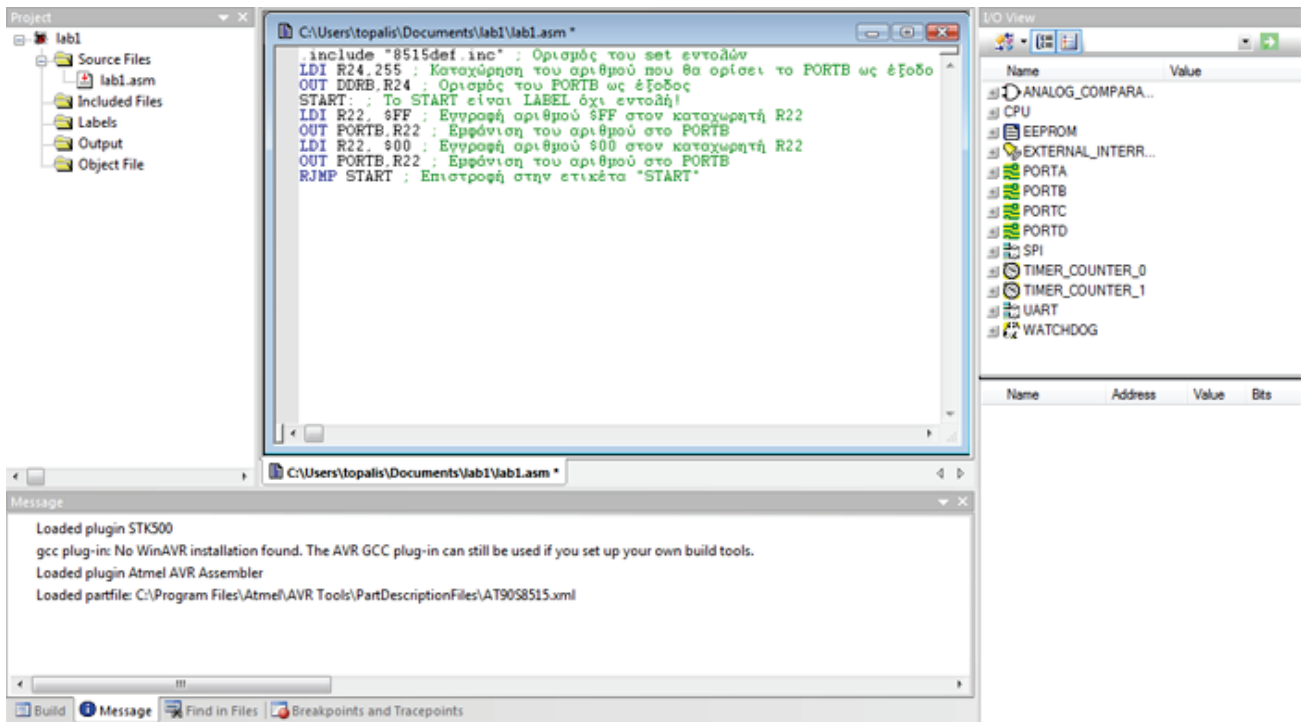
Κάτω από το μενού έχουμε τις μπάρες εργασίας “Toolbars” (Εικόνα 4.22), οι οποίες είναι συντομεύσεις για εργασίες που εκτελούνται συχνά, όπως. Build & Run, memory window.



Εικόνα 4.22

Πέραν των παραπάνω έχουμε και την εμφάνιση παραθύρων για την εισαγωγή του προγράμματος και τη λειτουργία του μικροελεγκτή. Συγκεκριμένα αριστερά του περιβάλλοντος εργασίας βλέπουμε την στήλη με το όνομα “Workspace” (Εικόνα 4.23) η οποία αποτελείται από τρία παράθυρα. Το παράθυρο με το όνομα “project” (Εικόνα 4.23) περιέχει πληροφορίες για τα αρχεία του έργου. Το βασικό κομμάτι της

επιφάνειας εργασίας καταλαμβάνεται από το παράθυρο (Εικόνα 4.23) που έχει το όνομα του αρχείου assembly. Εκεί λαμβάνει χώρα η εισαγωγή του προγράμματος (Text editor). Στο κάτω μέρος υπάρχει το παράθυρο εξόδου “output window” (Εικόνα 4.23) όπου δίνονται διάφορες πληροφορίες για την κατάσταση του AVR Studio ανάλογα με την εργασία που εκτελείται κάθε φορά και ανάλογα με την επιλογή που έχουμε κάνει (build, message κλπ). Έτσι εκεί θα εμφανίζονται τα λάθη του συμβολομεταφραστή (Assembler) και με διπλό κλικ πάνω στο μήνυμα λάθους ο δρομέας (cursor) πηγαίνει αυτόματα στο παράθυρο εισαγωγής κειμένου στη γραμμή που παρουσιάζεται το σφάλμα.



Εικόνα 4.23

Τέλος, το παράθυρο “I/O” (Εικόνα 4.23) δίνει πληροφορίες για την κατάσταση του μικροελεγκτή κατά τη διάρκεια εκτέλεσης του προγράμματος, μας δίνει τη δυνατότητα να δούμε με λεπτομέρεια τις τιμές όλων των καταχωρητών του μικροελεγκτή και να προσδιορίσουμε έτσι τυχόν σφάλματα. Στην τελευταία γραμμή της οθόνης (Εικόνα 4.23) εμφανίζονται μερικές επιπλέον πληροφορίες για τη λειτουργία του AVR Studio και την εκτέλεση του προγράμματος.

Εισαγωγή προγράμματος, Αποσφαλμάτωση (Debug) Εξομοίωση (Simulator)

Ακολουθώντας τα βήματα που περιγράψαμε προηγουμένως δημιουργήστε ένα έργο (project) με όνομα lab1 και στο παράθυρο του editor, (Εικόνα 4.23), γράψτε το παρακάτω πηγαίο κώδικα (Εικόνα 4.24) και σώστε τον.

```

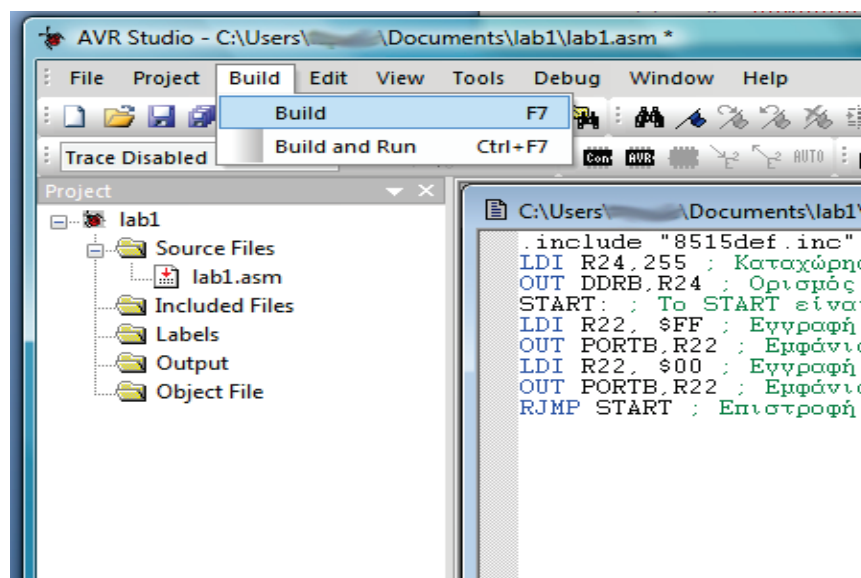
.include "8515def.inc" ; Ορισμός του set εντολών
LDI R24,255 ; Καταχώρηση του αριθμού που θα ορίσει το PORTB ως έξοδο
OUT DDRB,R24 ; Ορισμός του PORTB ως έξοδος
START: ; Το START είναι LABEL όχι εντολή!
LDI R22, $FF ; Εγγραφή αριθμού $FF στον καταχωρητή R22
OUT PORTB,R22 ; Εμφάνιση του αριθμού στο PORTB
LDI R22, $00 ; Εγγραφή αριθμού $00 στον καταχωρητή R22
OUT PORTB,R22 ; Εμφάνιση του αριθμού στο PORTB
RJMP START ; Επιστροφή στην ετικέτα "START"

```

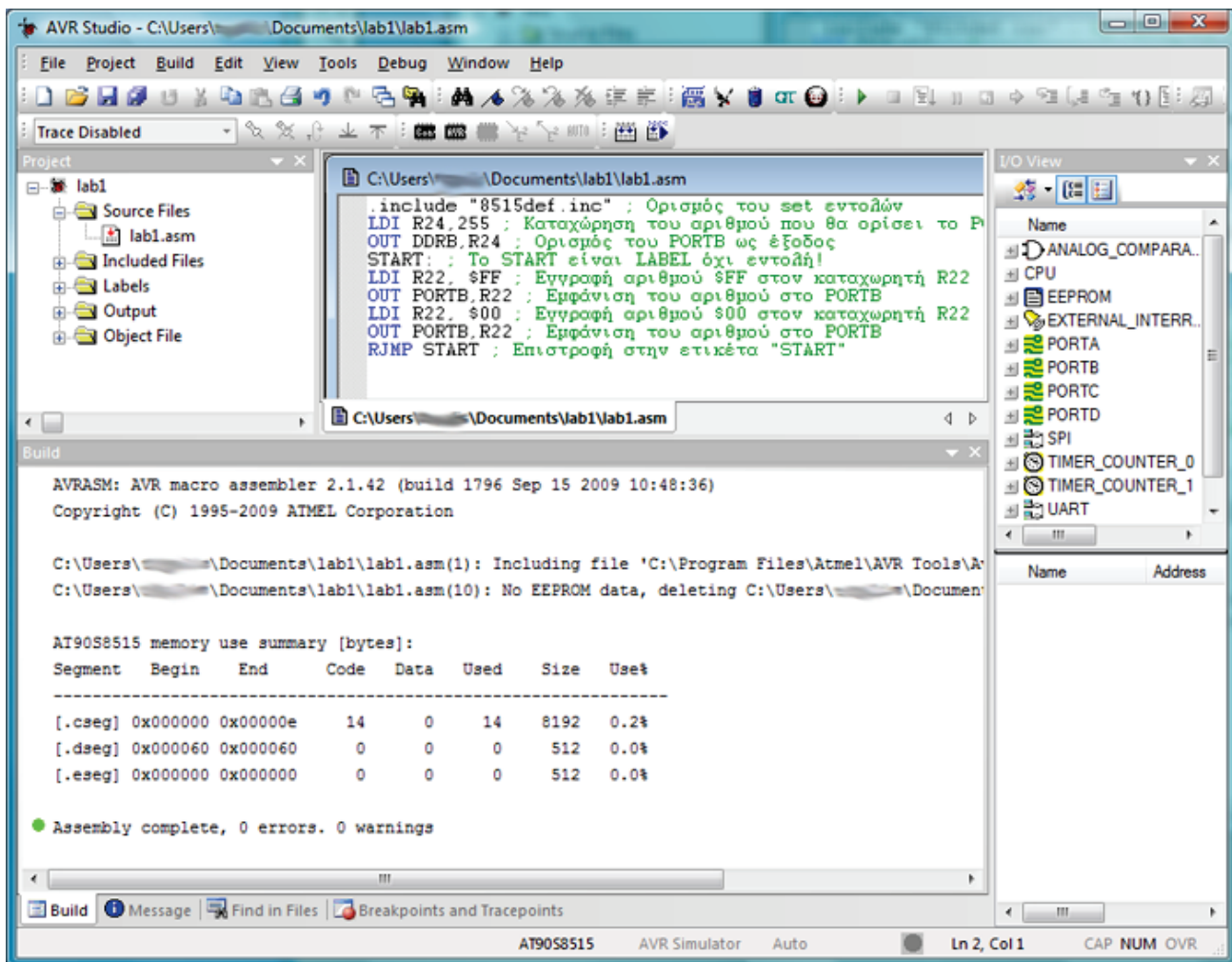
Εικόνα 4.24

Προσοχή: Η γλώσσα assembly δεν κάνει διάκριση μεταξύ μικρών και κεφαλαίων γραμμάτων (δεν είναι case sensitive). Για την εισαγωγή σχολίων χρησιμοποιούμε τον χαρακτήρα “;” στην αρχή αυτών.

Αφού περάσουμε το πρόγραμμα στον editor του AVR Studio από την επιλογή “Project” επιλέξτε την εντολή “Build” (Εικόνα 4.25) για να δημιουργηθεί το εκτελέσιμο αρχείο του προγράμματός σας. Αν ο κώδικας μας έχει συντακτικά σφάλματα αυτά θα φανούν στο παράθυρο εξόδου του μεταγλωττιστή όπως φαίνεται στο κάτω μέρος της Εικόνας 4.26.

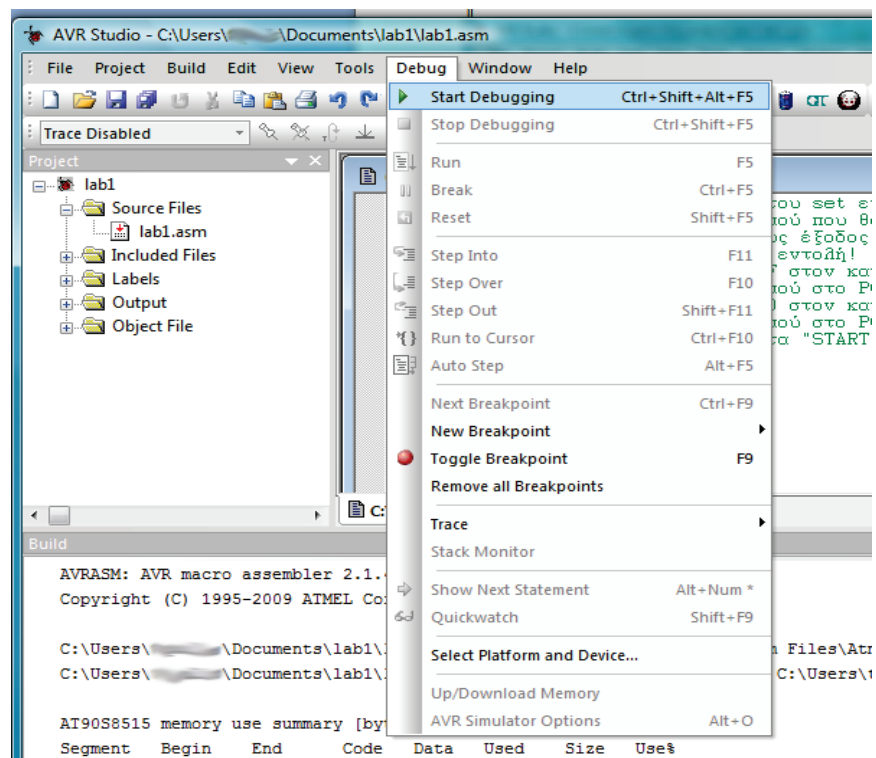


Εικόνα 4.25

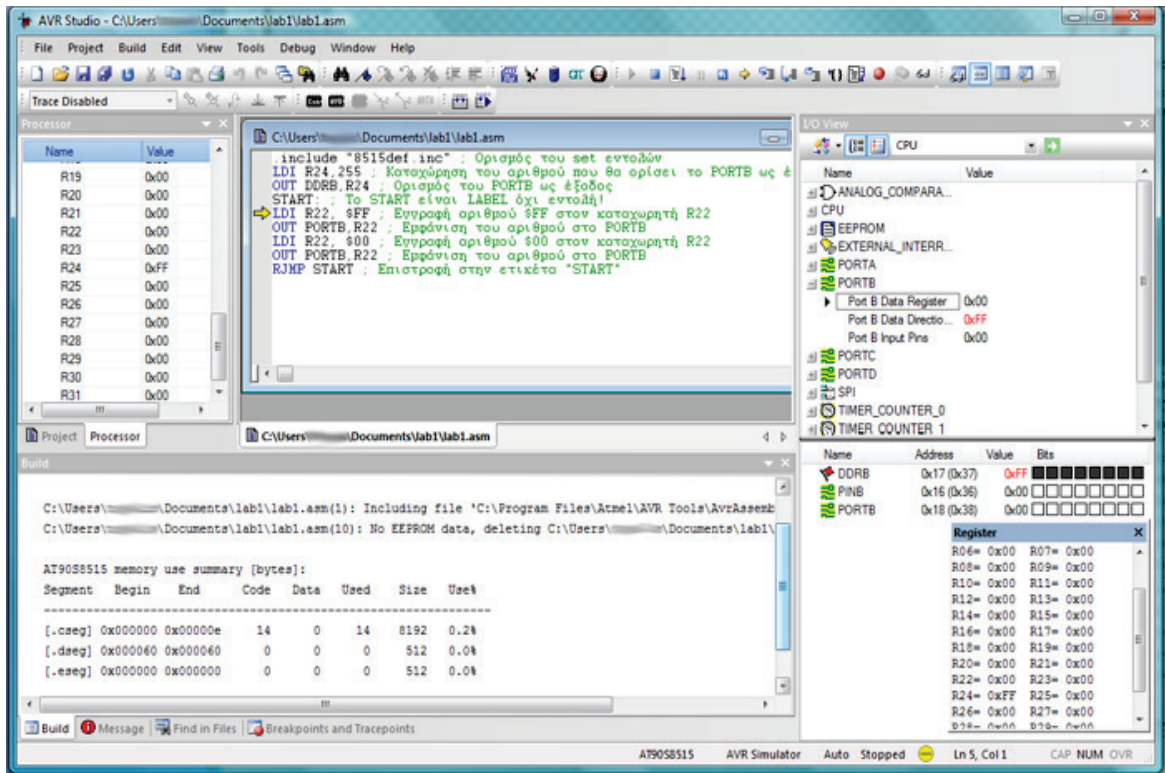


Εικόνα 4.26

Από την επιλογή “Debug” (Εικόνα 4.27) επιλέξτε “Start debugging”. Στο παράθυρο I/O view επιλέξτε να βλέπετε τους καταχωρητές του PORTB και από την επιλογή “View” επιλέξτε το “Register” (Εικόνα 4.28) ώστε να βλέπετε και τις τιμές των καταχωρητών γενικού σκοπού R16-R31, όπως φαίνεται στην παρακάτω Εικόνα 4.28. Τρέξτε τον εξομοιωτή (simulator) με εκτέλεση βήμα – βήμα (F11) και παρατηρήστε ποιοι καταχωρητές αλλάζουν περιεχόμενο και σε ποια εντολή συμβαίνει αυτό (Εικόνα 4.28).



Εικόνα 4.27



Εικόνα 4.28

Καταχωρητής Κατάστασης - Status REGISTER (SREG)

Στις εντολές πρόσθεσης και αφαίρεσης συμμετέχει και το κρατούμενο (C flag) το οποίο είναι ένα bit του καταχωρητή κατάστασης SREG (Status REGISTER). Το κρατούμενο, κάθε φορά που εκτελείται μια αριθμητική εντολή, ανανεώνεται αυτόματα. Έτσι αν το αποτέλεσμα μιας πράξης βγάλει κρατούμενο τότε θα έχουμε C = "1" ενώ αν δε βγάλει τότε θα έχουμε C = "0", ανεξάρτητα από την τιμή που είχε πριν την εκτέλεση της πράξης. Το κρατούμενο μπορούμε να το κάνουμε και κατ' απαίτηση "1" ή "0" χρησιμοποιώντας τις εντολές SEC και CLC αντίστοιχα.

Bit	7	6	5	4	3	2	1	0	
\$3F (\$5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Ο καταχωρητής κατάστασης (SREG) είναι ένας 8-bit καταχωρητής που κάθε φορά που εκτελούνται αριθμητικές πράξεις, ανανεώνεται (refresh) με σημαίες που έχουν να κάνουν με το αποτέλεσμα της πράξης.

I (Global Interrupt Enable): Επιτρέπει τις διακοπές (Interrupts) στον επεξεργαστή.

T (Target) : Οι εντολές αντιγραφής ενός Bit από μια θέση μνήμης έχουν σαν θέση αποθήκευσης το συγκεκριμένο πεδίο.

H (Half Carry): Κρατούμενο που προκύπτει από τα τέσσερα λιγότερο σημαντικά ψηφία.

S (Sign): Δείκτης προσήμου, αποτέλεσμα του «αποκλειστικού OR» EXOR μεταξύ των πεδίων V και N.

V (Overflow): Δείχνει υπερχείλιση σε πράξεις συμπληρώματος ως προς 2

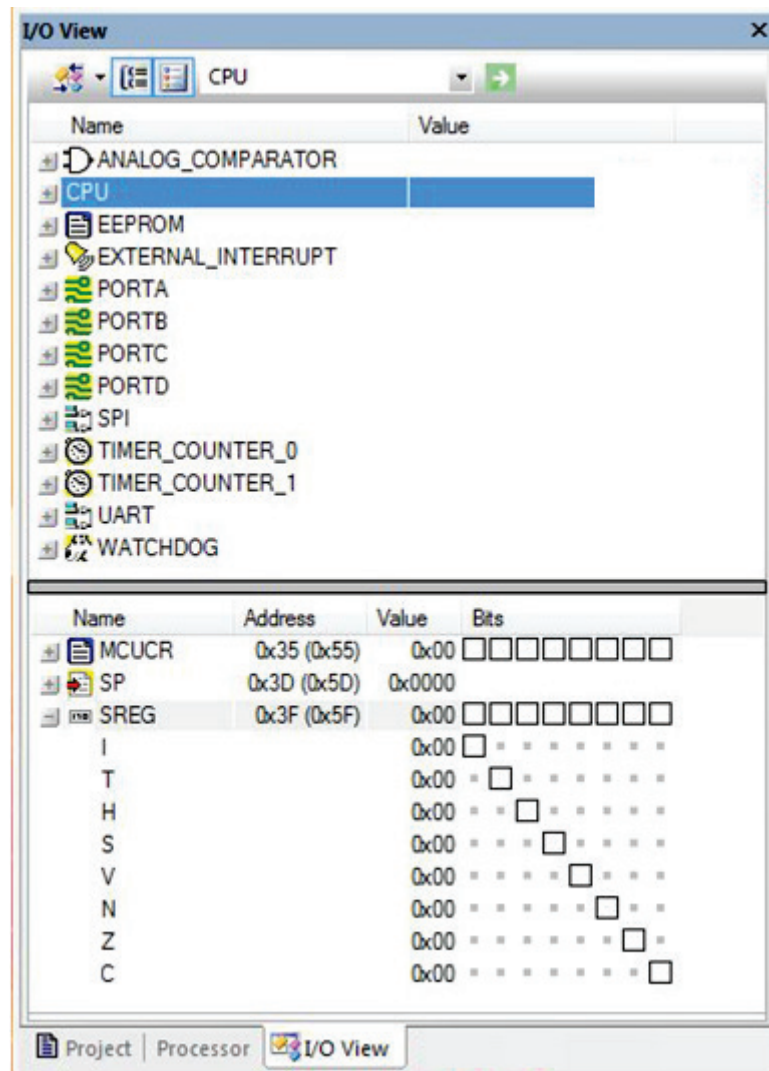
N (Negative): Δηλώνει ότι ο αριθμός που προκύπτει από μια αριθμητική ή λογική πράξη είναι αρνητικός

Z (Zero): Δηλώνει ότι το αποτέλεσμα μιας πράξης είναι 0

C (Carry): Δείκτης που δηλώνει ότι προέκυψε κρατούμενο από μια αριθμητική ή λογική πράξη.

Στην πράξη οι (πολύ) περισσότερο χρησιμοποιούμενες - λαμβανόμενες υπ όψιν σημαίες είναι οι I , N , Z , C ενώ οι T , H , S , V έχουν δευτερεύουσα σημασία.

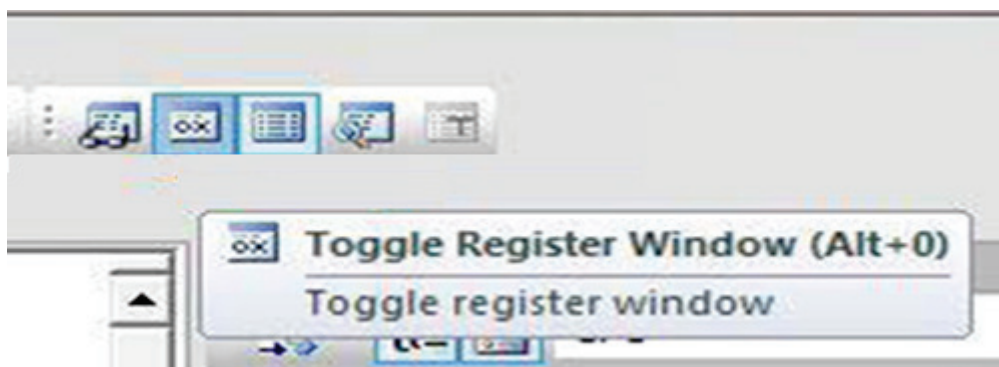
Ο έλεγχος του καταχωρητή κατάστασης SREG γίνεται από την περιοχή "I/O View" του περιβάλλοντος εργασίας του AVR Studio (Εικόνα 4.29). Στον ίδιο χώρο φαίνονται συνοπτικά όλες οι τιμές των bit του SREG.



Εικόνα 4.29

Έλεγχος των καταχωρητών γενικής χρήσης 0-31

Ο έλεγχος των καταχωρητών γενικής χρήσης 0-31 γίνεται επιλέγοντας το εικονίδιο "Toggle Register Window" ή "Alt+0" (Εικόνα 4.30). Στο παράθυρο που ενεργοποιείται (Εικόνα 4.31), φαίνονται οι τιμές των καταχωρητών γενικής χρήσης 0-31.



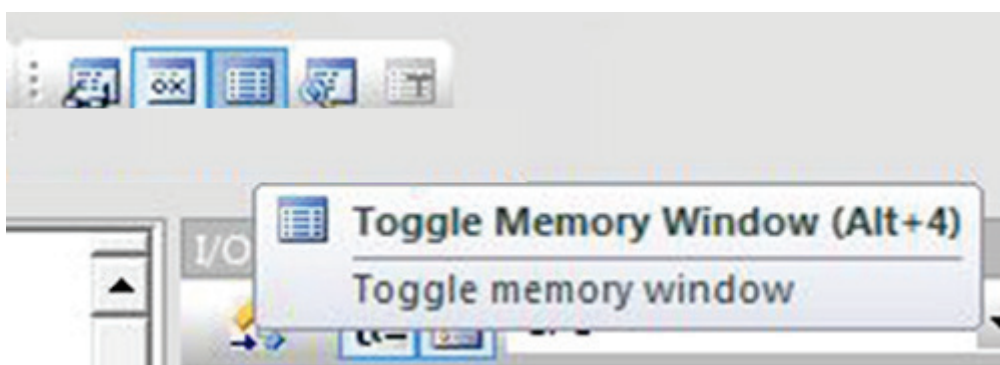
Εικόνα 4.30

Register			X	
R00= 0x00	R01= 0x00	R02= 0x00		
R03= 0x00	R04= 0x00	R05= 0x00		
R06= 0x00	R07= 0x00	R08= 0x00		
R09= 0x00	R10= 0x00	R11= 0x00		
R12= 0x00	R13= 0x00	R14= 0x00		
R15= 0x00	R16= 0x00	R17= 0x00		
R18= 0x00	R19= 0x00	R20= 0x00		
R21= 0x00	R22= 0x00	R23= 0x00		
R24= 0x00	R25= 0x00	R26= 0x00		
R27= 0x00	R28= 0x00	R29= 0x00		
R30= 0x00	R31= 0x00			

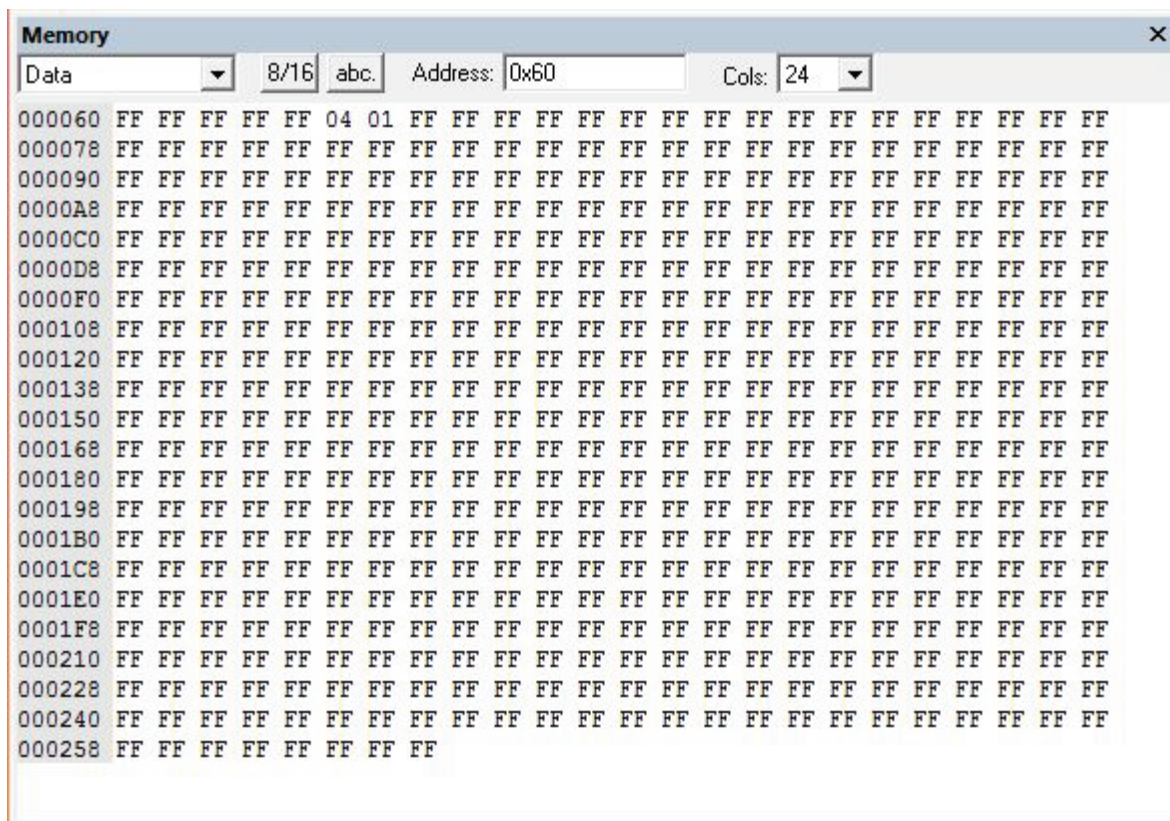
Εικόνα 4.31

Έλεγχος της μνήμης SRAM

Ο έλεγχος της μνήμης SRAM του μικροεπεξεργαστή, γίνεται επιλέγοντας το εικονίδιο "Toggle Memory Window" ή "Alt+4" (Εικόνα 4.32). Στο παράθυρο που ενεργοποιείτε (Εικόνα 4.33), φαίνονται οι τιμές της μνήμης SRAM.



Εικόνα 4.32



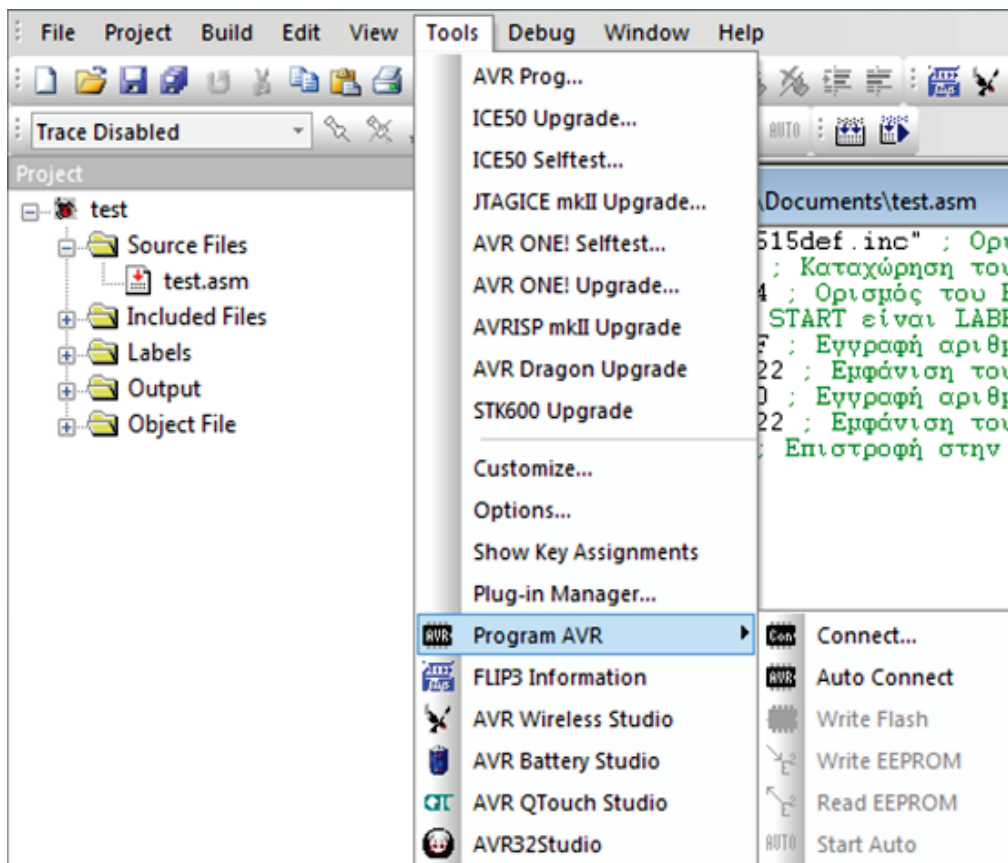
Εικόνα 4.33

Σύνδεση (connect) με το KIT και κατέβασμα/τρέξιμο (download/run) του κώδικα στο KIT

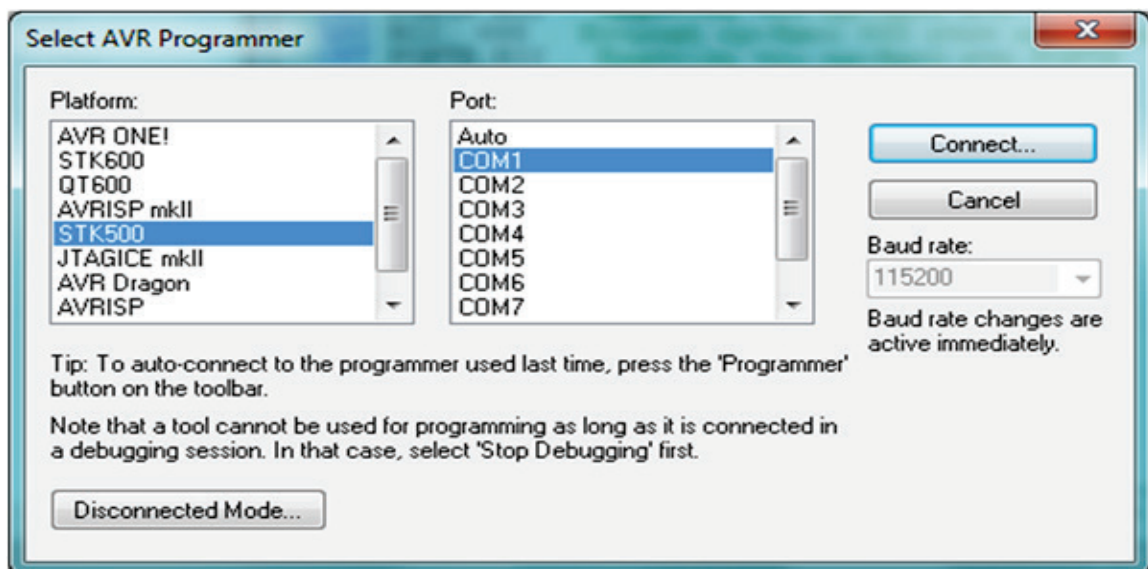
Ακολουθώντας τα βήματα που περιγράψαμε προηγουμένως δημιουργήσαμε ένα έργο (project) με όνομα lab1, γράψαμε ένα πρόγραμμα με όνομα lab1.asm, το περάσαμε από την αποσφαλμάτωση και το εξομοιώσαμε, μπορούμε να το κατεβάσουμε στο KIT.

Μέσω της επιλογής “Tools/Programm AVR” (Εικόνα 4.34) μπορούμε να επιλέξουμε την σύνδεση με το KIT, διαλέγοντας την επιλογή “Connect” (Εικόνα 4.34). Οπότε εμφανίζεται η φόρμα της Εικόνας 4.35, στην οποία πρέπει να επιλέξουμε την πλατφόρμα (Platform) STK500 και θύρα σειριακής επικοινωνίας (port), στις περισσότερες περιπτώσεις την com1 (Εικόνα 4.35). Έπειτα πρέπει να πατήσουμε την επιλογή “Connect”, μέσω της οποίας γίνεται η σύνδεση του KIT με το υπολογιστή που δουλεύουμε και εμφανίζεται η φόρμα της Εικόνας 4.36, στην οποία πατάμε την 2^η επιλογή “Program” (Εικόνα 4.36).

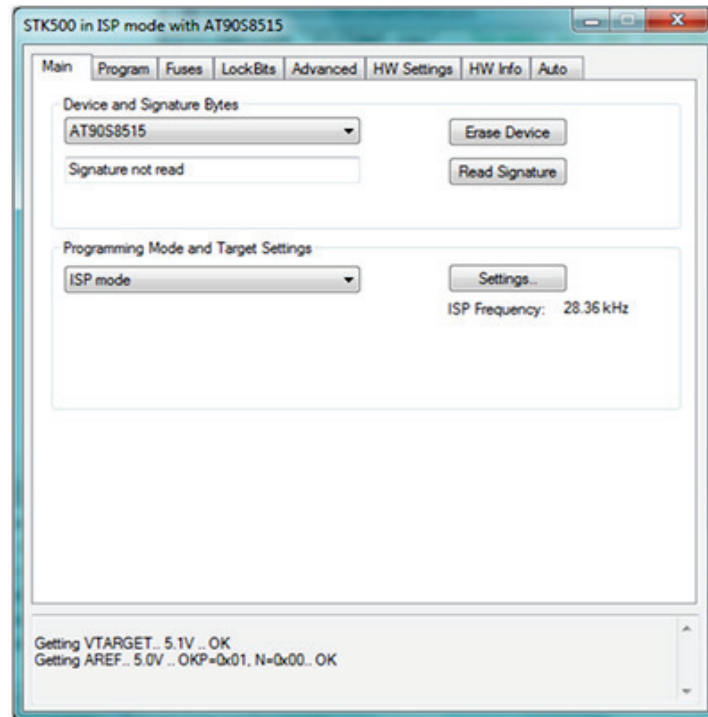
Ακολουθώντας τα βήματα που περιγράψαμε προηγουμένως, μπορούμε τελικά να κατεβάσουμε/τρέξουμε το πρόγραμμα που φτιάξαμε στο KIT. Συνήθως θα δουλεύουμε με την flash, θα διαλέγουμε το αρχείο hex (ψάχνοντας το στον φάκελο του έργου που δουλεύουμε) και με την επιλογή “Program” μπορούμε να κατεβάσουμε/τρέξουμε το πρόγραμμα που φτιάξαμε στο KIT (Εικόνα 4.37).



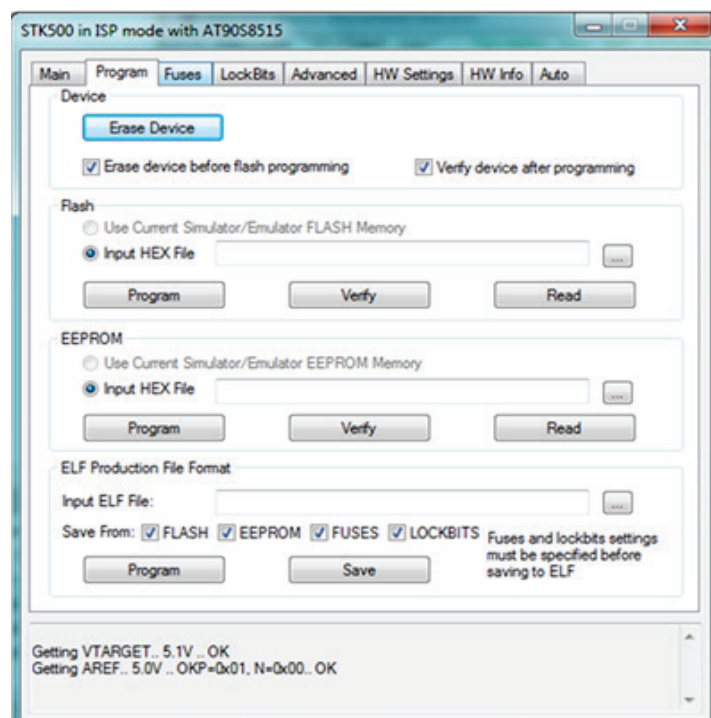
Εικόνα 4.34



Εικόνα 4.35



Εικόνα 4.36



Εικόνα 4.37

Κεφάλαιο 5 - ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΑΣΚΗΣΗ 1 – Γενικά για τους Μικροϋπολογιστές, Εξοικείωση με το ΚΙΤ και το Περιβάλλον Εργασίας

Σε αυτό το εργαστήριο γίνεται μία γνωριμία με τους μικροϋπολογιστές/ μικροελεγκτές και πιο συγκεκριμένα με τον AVR.

Παράλληλα, γίνεται μία γνωριμία με το μικροΚΙΤ AVR STK500 καθώς και με το περιβάλλον εργασίας του AVR Studio 4.0.

Ορισμοί δεδομένων:

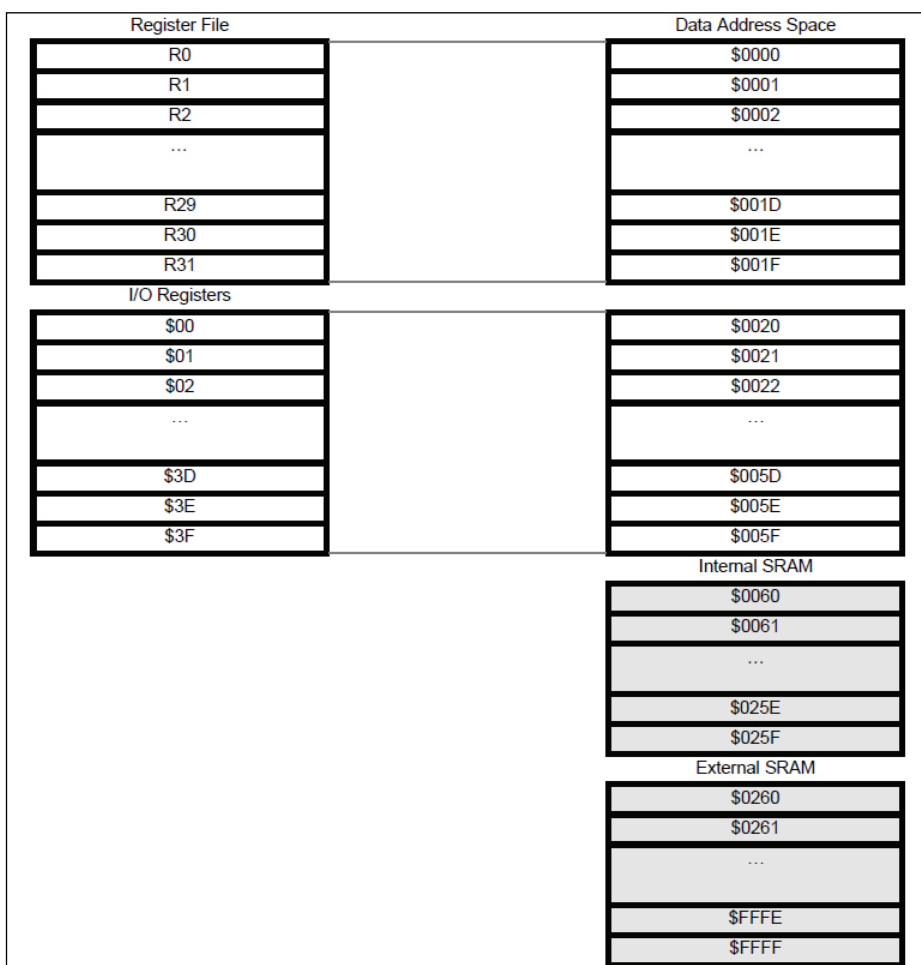
Δεκαδικό	11
Δυαδικό	0b010101
Δεκαεξαδικό	\$12

ΑΣΚΗΣΗ 2 – Εντολές Μεταφοράς Δεδομένων σε Καταχωρητές και Μνήμη

Στόχος της παρούσας εργαστηριακής άσκησης είναι να παρουσιάσει και να αναλύσει όλους του τρόπους που ο μικροελεγκτής AT90S8515 παρέχει στον προγραμματιστή για την μεταφορά δεδομένων από/προς καταχωρητές και μνήμη.

Μέσα από εκτενή παραδείγματα και ασκήσεις στόχος είναι ο φοιτητής να εξοικειωθεί τουλάχιστον με τους κυριότερους από αυτούς.

Για να γίνουν κατανοητοί οι διαθέσιμοι τρόποι διευθυνσιοδότησης πρώτα από όλα πρέπει να έχουμε στο μυαλό μας την χαρτογράφηση της μνήμης δεδομένων όπως αυτή φαίνεται στο ακόλουθο σχήμα.



Εικόνα 1: Η δομή της μνήμης δεδομένων του AT90S8515

Παρατηρήσεις: Οι καταχωρητές γενικού σκοπού αντιστοιχίζονται και σε συγκεκριμένη διεύθυνση, άρα πρόσβαση σε αυτούς μπορούμε να έχουμε τόσο αναφερόμενοι στο όνομά τους (R0...R31) όσο και μέσω της διεύθυνσής τους (\$0000 \$001F). Το κομμάτι μνήμης από \$0020 μέχρι \$005F αντιστοιχίζονται σε καταχωρητές I/O, κατά συνέπεια πρέπει να είμαστε προσεκτικοί όταν κάνουμε πρόσβαση στο κομμάτι αυτό. Τέλος στο κομμάτι μνήμης από \$0260 μέχρι \$FFFF μπορούμε να έχουμε πρόσβαση

μόνο αν υπάρχει εξωτερική μνήμη SRAM (στα πλαίσια του εργαστηρίου δεν χρειάζεται και κατά συνέπεια δεν υπάρχει).

Αφού έχουμε καλή αντίληψη για την διαθέσιμη μνήμη θα παρουσιάσουμε τους υποστηριζόμενους τρόπους διευθυνσιοδότησης τις αντίστοιχες εντολές που παρέχονται για κάθε τρόπο, αντίστοιχα παραδείγματα και σημαντικές λεπτομέρειες.

1. Άμεσος Τρόπος Διευθυνσιοδότησης (Immediate)

Ο τρόπος αυτός μας δίνει τη δυνατότητα να μεταφέρουμε σε έναν καταχωρητή μια συγκεκριμένη αριθμητική ποσότητα. Η εντολή που υποστηρίζει τον τρόπο αυτό είναι η **LDI**. Η κατεύθυνση μεταφοράς, όπως και σε όλες τις εντολές, είναι από το δεύτερο όρισμα προς το πρώτο.

LDI Rd,K Rd ← K	Περιγραφή: Άμεση αποθήκευση στον καταχωρητή προορισμού Rd της αριθμητικής ποσότητας K. Ενημέρωση σημαίων: Καμία Παράδειγμα: Η εντολή LDI R20, \$16 θα έχει σαν αποτέλεσμα την μεταφορά του αριθμού 16H στα περιεχόμενα του καταχωρητή R20. Περίοδοι ρολογιού: 1
----------------------------------	---

2. Μεταφορά από καταχωρητή σε καταχωρητή

Η εντολή **MOV** μας δίνει τη δυνατότητα μεταφοράς δεδομένων μεταξύ δύο καταχωρητών.

MOV Rd, Rr Rd ← Rr	Περιγραφή: Αντιγραφή των δεδομένων του καταχωρητή Rr στον καταχωρητή Rd. Ενημέρωση σημαίων: Καμία Παράδειγμα: Η εντολή MOV R20, R16 θα έχει σαν αποτέλεσμα να αντιγραφούν τα δεδομένα του καταχωρητή R16 στον R20 . Με άλλα λόγια, μετά την εκτέλεση και οι δύο καταχωρητές θα περιέχουν ότι περιείχε ο R16 πριν την εκτέλεση της εντολής. Περίοδοι ρολογιού: 1
-------------------------------------	---

3. Απ' ευθείας διευθυνσιοδότηση (Direct)

Με τις αντίστοιχες εντολές ο AVR μας επιτρέπει να διαβάζουμε και να γράφουμε απ' ευθείας (Direct) σε μια θέση μνήμης SRAM. Οι εντολές αυτές είναι οι **LDS** (Load Direct from SRAM) και **STS** (Store Direct to SRAM)

LDS Rd, k Rd ←(k)	Περιγραφή: Μεταφορά των δεδομένων από τη διεύθυνση μνήμης SRAM k στον καταχωρητή Rd. Ενημέρωση σημαίων: Καμία Παράδειγμα: Η εντολή LDS R20, \$70 θα έχει σαν αποτέλεσμα να διαβαστούν τα περιεχόμενα της διεύθυνσης \$70 αν αντιγραφούν στα περιεχόμενα του καταχωρητή R20. Με την ίδια λογική η εντολή STS \$70, R20 θα έχει σαν αποτέλεσμα τα περιεχόμενα του καταχωρητή R20 να γραφτούν στην διεύθυνση μνήμης \$70. Περίοδοι ρολογιού: 3
------------------------------------	--

STS k, Rr (k) ←Rr	Περιγραφή: Μεταφορά των δεδομένων από τον καταχωρητή Rr στην διεύθυνση μνήμης SRAM k. Ενημέρωση σημαίων: Καμία Παράδειγμα: Η εντολή STS \$70, R20 θα έχει σαν αποτέλεσμα τα περιεχόμενα του καταχωρητή R20 να γραφτούν στην διεύθυνση μνήμης \$70. Με την ίδια λογική η εντολή. Περίοδοι ρολογιού: 3
------------------------------------	--

4. Έμμεση διευθυνσιοδότηση (Indirect)

Όπως έχει αναφερθεί και στην εισαγωγή στην αρχιτεκτονική των AVR υπάρχουν τρεις λογικοί καταχωρητές X, Y, Z εύρους 16bit . Για να μπορέσουμε να εκφράσουμε 16bit αριθμούς χρησιμοποιούμε συγκεκριμένα ζευγάρια των 8bit καταχωρητών γενικού σκοπού.

Συγκεκριμένα

- ο X ορίζεται από το ζευγάρι R27,R26
- ο Y ορίζεται από το ζευγάρι R29,R28
- ο Z ορίζεται από το ζευγάρι R31,R30

Ο όρος έμμεση διευθυνσιοδότηση αφορά ακριβώς τη χρήση αυτών των λογικών καταχωρητών για να επιτύχουμε πρόσβαση στο address space. Οι εντολές που χρησιμοποιούν το συγκεκριμένο τρόπο διευθυνσιοδότηση είναι οι **LD**, **LDD**, **ST**, **STD**. Για λόγους πληρότητας αναφέρουμε την ύπαρξη σύνθετων εντολών έμμεσης προσπέλασης μνήμης σε συνδυασμό με αύξηση ή μείωση των περιεχομένων των

καταχωρητών X,Y,Z. Αν και οι εν λόγω εντολές αυξάνουν τις δυνατότητες του προγραμματιστή οι συγκεκριμένες λειτουργίες είναι σχετικά εξειδικευμένες για τα πλαίσια του εργαστηρίου και κατά συνέπεια για περαιτέρω πληροφορίες οι φοιτητές μπορούν αν απευθυνθούν στο παράρτημα.

1) LD Rd, X $Rd \leftarrow (X)$	Περιγραφή: Στην περίπτωση 1) μεταφορά των δεδομένων από τη διεύθυνση μνήμης SRAM X στον καταχωρητή Rd. Στην περίπτωση 2) μετά τη μεταφορά γίνεται αύξηση των δεδομένων του X κατά 1. Στην περίπτωση 3) πριν τη μεταφορά γίνεται μείωση των δεδομένων του X κατά 1 Ενημέρωση σημαίων: Καμία Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 2
2) LD Rd, X+ $Rd \leftarrow (X), X \leftarrow X+1$	
3) LD Rd, -X $X \leftarrow X-1, Rd \leftarrow (X),$	
LDD Rd, Y+q $Rd \leftarrow (Y+q)$	Περιγραφή: Μεταφορά των δεδομένων από τη διεύθυνση μνήμης SRAM (Y+q) στον καταχωρητή Rd. Ενημέρωση σημαίων: Καμία Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 2
1) ST X, Rr $(X) \leftarrow Rr$	Περιγραφή: Στην περίπτωση 1) μεταφορά των δεδομένων από τον καταχωρητή Rr στην διεύθυνση μνήμης SRAM X. Στην περίπτωση 2) μετά τη μεταφορά γίνεται αύξηση των δεδομένων του X κατά 1. Στην περίπτωση 3) πριν τη μεταφορά γίνεται μείωση των δεδομένων του X κατά 1 Ενημέρωση σημαίων: Καμία Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 2
2) ST X+, Rr $(X) \leftarrow Rr, X \leftarrow X+1$	
3) ST -X, Rr $X \leftarrow X-1, (X) \leftarrow Rr$	
STD Z+q, Rr $(Z+q) \leftarrow Rr$	Περιγραφή: Μεταφορά των δεδομένων από τον καταχωρητή Rr στην διεύθυνση μνήμης SRAM (Z+q). Ενημέρωση σημαίων: Καμία Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 2

π.χ. Θεωρήσουμε ότι

R27=\$00 και R26=\$70 άρα ο X «δείχνει» στην διεύθυνση \$0070

R29=\$00 και R28=\$75 άρα ο Y «δείχνει» στην διεύθυνση \$0075

R31=\$00 και R30=\$80 άρα ο Z «δείχνει» στην διεύθυνση \$0080

η **LD R20, X** μεταφέρει να δεδομένα από την διεύθυνση μνήμης **\$0070** στο καταχωρητή **R20**

η **LDD R20, Y+2** μεταφέρει να δεδομένα από την διεύθυνση μνήμης **\$0077** στο καταχωρητή **R20**

(**Προσοχή:** η **LDD** δεν μπορεί να χρησιμοποιηθεί σε συνδυασμό με τον καταχωρητή **X**)

η **ST X, R20** μεταφέρει τα δεδομένα από τον καταχωρητή **R20** στην διεύθυνση μνήμης **\$0070**.

η **STD Z+2, R20** μεταφέρει να δεδομένα από τον καταχωρητή **R20** στην διεύθυνση μνήμης **\$0082**

(**Προσοχή:** η **STD** δεν μπορεί να χρησιμοποιηθεί σε συνδυασμό με τον καταχωρητή **X**)

Σημαντικές Παρατηρήσεις:

Η εντολή άμεσης φόρτωσης LDI (LoaD Immediate) δεν μπορεί να χρησιμοποιηθεί συνδυασμό με ένα από τους πρώτους 16 καταχωρητές (R0-R15), μπορεί να χρησιμοποιηθεί μόνο έναν από τους R16-R32. Έτσι λοιπόν η εντολή **LDI R15, \$16** είναι λάθος.

Αν και οι καταχωρητές δεν είναι SRAM μνήμες σε φυσικό επίπεδο το γεγονός ότι μοιράζονται το ίδιο address space μας δίνει τη δυνατότητα να έχουμε πρόσβαση στους καταχωρητές μέσω των εντολών απ' ευθείας και έμμεσης διευθυνσιοδότησης. π.χ. η εντολή **STS \$03, R24** θα έχει σαν αποτέλεσμα τα περιεχόμενα του **R24** να μεταφέρονται στον **R3**.

Οι εντολές φόρτωσης και ανάγνωσης δεδομένων από την μνήμη με μετατόπιση (LDD, STD) δεν μπορούν να συνδυαστούν με τον καταχωρητή X.

Πολλές φορές η εξορισμού ονομασία των καταχωρητών R1.....R32 δυσκολεύει τη χρήση τους σε μεγάλα και πολύπλοκα προγράμματα ή όταν ο προγραμματιστής δεν είναι ιδιαίτερα εξοικιωμένος. Για το λόγο αυτό παρέχεται μια πολύ χρήσιμη ψευδοεντολή με την οποία μπορούμε να αντιστοιχήσουμε έναν συγκεκριμένο καταχωρητή με ένα συγκεκριμένο όνομα που μας είναι πιο εύκολο να το θυμόμαστε ή δίνει στον καταχωρητή ένα όνομα συναφές με το τη χρήση του.

π.χ. η ψευδοεντολή **.DEF myregister = R20** δίνει εντολή στο μεταφραστή του προγράμματος όπου υπάρχει το αναγνωριστικό **myregister** να το αντικαταστήσει με **R20**. Έτσι ο προγραμματιστής μπορεί να γράψει όλο το πρόγραμμά του χρησιμοποιώντας το **myregister** και να μην χρειάζεται να θυμάται ποιόν ακριβώς καταχωρητή έχει αντιστοιχήσει σε αυτό.

Παράδειγμα

Γράψτε πρόγραμμα που να εκτελεί τις ακόλουθες λειτουργίες.

Να δίνει στους καταχωρητές R16, R17, R18, R19 το ονόματα A,B,C,D αντίστοιχα.

Να φορτώνει στους παραπάνω καταχωρητές τις απόλυτες τιμές \$3E, \$0A, \$1F, \$55

Μεταφέρατε τα δεδομένα των A,B,C,D στους R10, R11, R12, R13 αντίστοιχα.

Μεταφέρατε τα δεδομένα των διευθύνσεων \$75 και \$76 στους καταχωρητές R14, R15

Χρησιμοποιώντας τον απ' ευθείας τρόπο διευθυνσιοδότησης μεταφέρατε τα περιεχόμενα των καταχωρητών R14 και R15 στους καταχωρητές R1 και R2.

Μεταφέρατε τα περιεχόμενα του R19 στη διεύθυνση \$0278

Λύση:

```
.include "8515def.inc"
```

;Ερώτημα 1

```
.DEF A=R16
```

```
.DEF B=R17
```

```
.DEF C=R18
```

```
.DEF D=R19
```

;Ερώτημα 2

```
LDI A,$3E
```

```
LDI B,$0A
```

```
LDI C,$1F
```

```
LDI D,$55
```

;Ερώτημα 3

```
MOV R10,A
```

```
MOV R11,B
```

```
MOV R12,C
```

```
MOV R13,D
```

;Ερώτημα 4

```
LDS R14,$75
```

```
LDS R15,$76
```

;Ερώτημα 5

```
STS $01,R14
```

```
STS $02,R15
```

;Ερώτημα 6

LDI R27,\$02

LDI R26,\$78

ST X,R19

ΑΣΚΗΣΗ 3 – Αριθμητικές Πράξεις δεκαεξαδικών 8-bit/16-bit αριθμών

Γενικά

Οι αριθμητικές πράξεις όπως πρόσθεση και αφαίρεση είναι θεμελιώδης σε πολλά υπολογιστικά συστήματα. Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζουν διάφορους αριθμητικούς τύπους δεδομένων τα οποία χρησιμοποιούνται για την πραγματοποίηση αριθμητικών υπολογισμών.

Για την εκτέλεση αριθμητικών οι μικροελεγκτές AVR διαθέτουν ένα ευρύ φάσμα εντολών. Οι εντολές αυτές έχουν ένα ή δύο ορίσματα (operands) τα οποία είναι πάντοτε οι καταχωρητές γενικού σκοπού R0 έως R31.

Δεν υπάρχει δυνατότητα να γίνει απευθείας πράξη μεταξύ του περιεχομένου δύο θέσεων μνήμης. Επομένως για να γίνει μία αριθμητική ή λογική πράξη πρέπει πρώτα να μεταφερθούν τα ορίσματα από τις θέσεις μνήμης που είναι αποθηκευμένα να γίνει η πράξη και το αποτέλεσμα να γραφεί σε μία άλλη θέση μνήμης.

Στην άσκηση αυτή για λόγους απλότητας χρησιμοποιείται για αποθήκευση των δεδομένων και των αποτελεσμάτων μόνο η εσωτερική SRAM. Οι εντολές αριθμητικών πράξεων διακρίνονται σε εντολές πρόσθεσης και αφαίρεσης με ή χωρίς κρατούμενο μεταξύ δύο καταχωρητών ή ενός καταχωρητή και μιας σταθεράς ή ζεύγους καταχωρητών και μιας σταθεράς.

Εντολές Πρόσθεσης

ADD Rd,Rr Rd ← Rd + Rr	Περιγραφή: Πρόσθεση 2 καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd. Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: ADD R4,R5 Περίοδοι ρολογιού: 1
ADC Rd,Rr Rd ← Rd + Rr+C	Περιγραφή: Πρόσθεση 2 καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd. Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: ADC R5,R6 Περίοδοι ρολογιού: 1
ADIW Rd,K Rd+1:Rd ← Rd+1:Rd + K	Περιγραφή: Πρόσθεση μιας τιμής K(0-63) σε ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος καταχωρητών Ενημέρωση σημαίων: Z,C,N,S,V Παράδειγμα: ADIW R23,5 Περίοδοι ρολογιού: 2

Εντολές Αφαίρεσης

SUB Rd,Rr Rd ← Rd - Rr	Περιγραφή: Αφαίρεση 2 καταχωρητών χωρίς χρήση κρατούμενου και αποθήκευση στον καταχωρητή προορισμού Rd Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: SUB R4,R5 Περίοδοι ρολογιού: 1
SUBI Rd,K Rd ← Rd - K	Περιγραφή: Αφαίρεση μιας τιμής χωρίς χρήση κρατούμενου από έναν καταχωρητή και αποθήκευση στον καταχωρητή Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: SUBI R4,15 Περίοδοι ρολογιού: 1
SBC Rd,Rr Rd ← Rd - Rr - C	Περιγραφή: Αφαίρεση 2 καταχωρητών με χρήση κρατούμενου και αποθήκευση στον καταχωρητή προορισμού Rd. Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: SBC R4,R5 Περίοδοι ρολογιού: 1
SBCI Rd,K Rd ← Rd - K - C	Περιγραφή: Αφαίρεση μιας τιμής με χρήση κρατούμενου από έναν καταχωρητή και αποθήκευση στον καταχωρητή Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: SBCI R9,2 Περίοδοι ρολογιού: 1

Καταχωρητής Κατάστασης - Status REGISTER (SREG)

Στις εντολές πρόσθεσης και αφαίρεσης συμμετέχει και το κρατούμενο (C flag) το οποίο είναι ένα bit του καταχωρητή κατάστασης SREG (Status REGISTER). Το κρατούμενο, κάθε φορά που εκτελείται μια αριθμητική εντολή, ανανεώνεται αυτόματα. Έτσι αν το αποτέλεσμα μιας πράξης βγάλει κρατούμενο τότε θα έχουμε C = "1" ενώ αν δε βγάλει τότε θα έχουμε C = "0", ανεξάρτητα από την τιμή που είχε πριν την εκτέλεση της πράξης. Το κρατούμενο μπορούμε να το κάνουμε και κατ' απαίτηση "1" ή "0" χρησιμοποιώντας τις εντολές SEC και CLC αντίστοιχα.

Bit	7	6	5	4	3	2	1	0	
\$3F (\$5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Ο καταχωρητής κατάστασης (SREG) είναι ένας 8-bit καταχωρητής που κάθε φορά που εκτελούνται αριθμητικές πράξεις, ανανεώνεται (refresh) με σημαίες που έχουν να κάνουν με το αποτέλεσμα της πράξης.

I (Global Interrupt Enable): Επιτρέπει τις διακοπές (Interrupts) στον επεξεργαστή.

T (Target) : Οι εντολές αντιγραφής ενός Bit από μια θέση μνήμης έχουν σαν θέση αποθήκευσης το συγκεκριμένο πεδίο.

H (Half Carry): Κρατούμενο που προκύπτει από τα τέσσερα λιγότερο σημαντικά ψηφία.

S (Sign): Δείκτης προσήμου, αποτέλεσμα του «αποκλειστικού OR» EXOR μεταξύ των πεδίων V και N.

V (Overflow): Δείχνει υπερχείλιση σε πράξεις συμπληρώματος ως προς 2

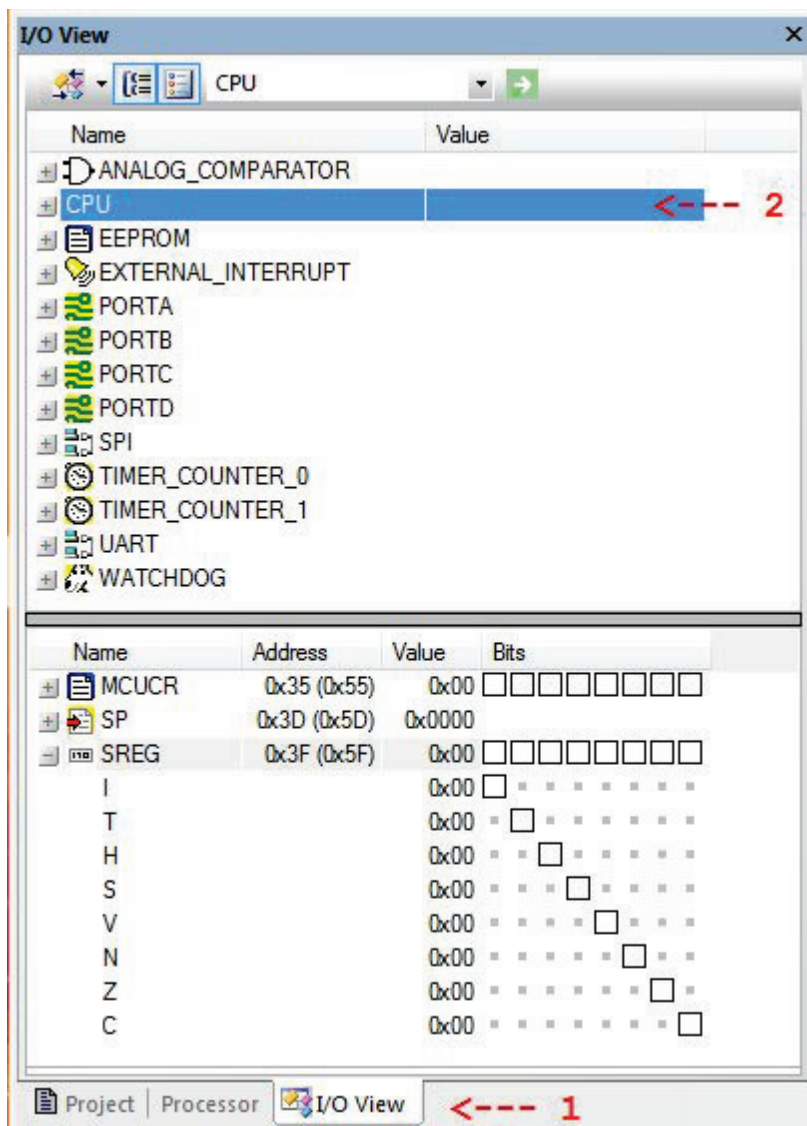
N (Negative): Δηλώνει ότι ο αριθμός που προκύπτει από μια αριθμητική ή λογική πράξη είναι αρνητικός

Z (Zero): Δηλώνει ότι το αποτέλεσμα μιας πράξης είναι 0

C (Carry): Δείκτης που δηλώνει ότι προέκυψε κρατούμενο από μια αριθμητική ή λογική πράξη.

Στην πράξη οι (πολύ) περισσότερο χρησιμοποιούμενες - λαμβανόμενες υπ όψιν σημαίες είναι οι I , N , Z , C ενώ οι T , H , S , V έχουν δευτερεύουσα σημασία.

Ο έλεγχος του καταχωρητή κατάστασης SREG γίνεται από την περιοχή "I/O View" του περιβάλλοντος εργασίας του AVR Studio (Εικόνα 5.1). Στον ίδιο χώρο φαίνονται συνοπτικά όλες οι τιμές των bit του SREG.



Εικόνα 5.1

Παραδείγματα

Παράδειγμα 1ο - Πρόσθεση 8bit αριθμών

Το αποτέλεσμα μιας πρόσθεσης 2 αριθμών 8bit (1 byte) είναι πιθανόν να είναι 2 bytes. Έτσι όταν προσθέτουμε δυο αριθμούς του ενός byte θα το κάνουμε σύμφωνα με το παρακάτω παράδειγμα:

Γράψτε ένα πρόγραμμα για να προσθέσετε τους αριθμούς \$FF και \$05. Τοποθετήστε το άθροισμά τους στην θέση μνήμης \$65 (Low Byte) και \$66 (High Byte). Υπολογίστε το άθροισμα μόνοι σας και δείτε στον Status Register (SREG) αν δημιουργήθηκε κρατούμενο. Αιτιολογείστε την απάντησή σας.

```
.include "8515def.inc"
```

```
; Πρόσθεση των δύο 8bit αριθμών και αποθήκευση στην θέση μνήμης $65
```

LDI R24,\$FF

LDI R25,\$05

ADD R24,R25

STS \$65,R24

; Αποθήκευση της θέσης μνήμης \$65 στον καταχωρητή R28

LDS R28,\$65

; Ενδεχομένως να έχουμε κρατούμενο μετά την πρόσθεση των 4 high Byte των

; προσθετέων για να καταφέρουμε να αποθηκεύσουμε το κρατούμενο που είναι

; σε ένα bit θα πρέπει να προσθέσουμε το κρατούμενο με δύο καταχωρητές που

; έχουν περιεχόμενο 0.

; Δηλαδή να γίνει η πράξη $0 + 0 + C$ έτσι ώστε αν η προηγούμενη πράξη έχει

; δώσει κρατούμενο να προκύψει ο αριθμός 01

; διαφορετικά ο 00.

; Μετατροπή του τυχόν κρατούμενου σε Byte και μεταφορά στον καταχωρητή R26

LDI R26,\$00

LDI R27,\$00

ADC R26,R27

; Αποθήκευση του κρατούμενου στη θέση μνήμης \$66

STS \$66,R26

; Αποθήκευση της θέσης μνήμης \$66 στον καταχωρητή R29

LDS R29,\$66

Παράδειγμα 2ο - Πρόσθεση 16bit αριθμών

Το αποτέλεσμα μιας πρόσθεσης 2 αριθμών 16bit (2 byte) είναι πιθανόν να είναι 3 bytes. Έτσι όταν προσθέτουμε δυο αριθμούς του ενός byte θα το κάνουμε σύμφωνα με το παρακάτω παράδειγμα:

Γράψτε ένα πρόγραμμα για να αποθηκεύσετε τον 16-bit αριθμό F1A2H στις θέσεις μνήμης 130H, 131H. Στη συνέχεια να προσθέσετε στον αριθμό που αποθηκεύσατε τον 16-bit αριθμό 1284H και τοποθετήστε το αποτέλεσμα στις θέσεις μνήμης 140H, 141H και το κρατούμενο (αν υπάρχει) στην διεύθυνση 142H. Υπολογίστε το άθροισμα μόνοι σας και δείτε στον Status Register (SREG) αν δημιουργήθηκε κρατούμενο. Αιτιολογήστε την απάντησή σας.

.include "8515def.inc"

; Αποθήκευση του αριθμού F1A2H στις θέσεις 130H και \$131H

LDI R18,\$A2

STS \$130,R18

LDI R19,\$F1

STS \$131,R19

; Πρόσθεση των δύο λιγότερο σημαντικών Bytes των δύο αριθμών με ADD για να
; αγνοηθεί τυχών προηγούμενο κρατούμενο και αποθήκευση του αποτελέσματος
; στη θέση \$140

LDS R20,\$130

LDI R21,\$84

ADD R20,R21

STS \$140,R20

; Πρόσθεση των δύο περισσότερο σημαντικών Bytes των δύο αριθμών με ADC για να
; ληφθεί υπόψη τυχών κρατούμενο από την προηγούμενη πρόσθεση και
; αποθήκευση του αποτελέσματος στη θέση \$141

LDS R20,\$131

LDI R21,\$12

ADC R20,R21

STS \$141,R20

; Μετατροπή του τυχών τελικού κρατούμενου σε Byte και αποθήκευση του στη
; θέση μνήμης \$142

LDI R24,\$00

LDI R25,\$00

ADC R24,R25

STS \$142,R24

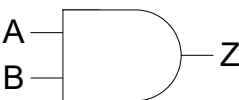
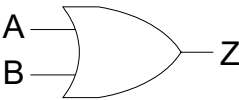
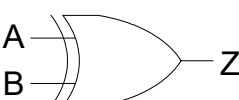
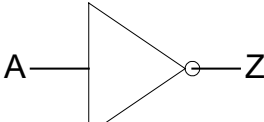
ΑΣΚΗΣΗ 4 – Λογικές Πράξεις, Εντολές Αύξησης Μείωσης, Εντολή NOP

Μαζί με τις αριθμητικές πράξεις οι αντίστοιχες λογικές πράξεις είναι οι κυριότερες λειτουργίες που μπορούν να εκτελεστούν από έναν μικροελεγκτή και συγκεκριμένα από την ALU (Arithmetic and Logic Unit) μονάδα του.

Κατά συνέπεια οι αντίστοιχες εντολές που υποστηρίζονται αποτελούν ένα ιδιαίτερα σημαντικό υποσύνολο του ρεπερτορίου εντολών κάθε μικροελεγκτή ή μικροεπεξεργαστή.

Σύμφωνα με την μικρή αυτή εισαγωγή στόχος στην παρούσα άσκηση θα γίνει μια συνοπτική αλλά πλήρης παρουσίαση των εντολών αυτών. Τελικός στόχος είναι με βάση τη παρουσίαση αυτή και μέσα από εκτενή παραδείγματα και εργαστηριακά ερωτήματα ο φοιτητής να κατανοήσει και να εξοικειωθεί με την λειτουργία των εντολών αυτών ώστε να μπορεί να τις χρησιμοποιήσει όταν και αν αυτό απαιτείται.

Στο παρακάτω πίνακα απεικονίζονται οι βασικές πύλες (ή αλλιώς πράξεις άλγεβρας BOOLE) που υποστηρίζονται με συγκεκριμένες εντολές.

ΟΝΟΜΑΣΙΑ	ΣΥΜΒΟΛΟ ΠΥΛΗΣ	ΣΧΕΣΗ ΑΛΓΕΒΡΑΣ BOOLE	ΠΙΝΑΚΑΣ ΑΛΗΘΕΙΑΣ															
AND		Z=A•B	<table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Z	0	0	0	0	1	0	1	0	0	1	1	1
			A	B	Z													
			0	0	0													
			0	1	0													
			1	0	0													
1	1	1																
OR		Z=A+B	<table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	1
			A	B	Z													
			0	0	0													
			0	1	1													
			1	0	1													
1	1	1																
XOR		Z=A ⊕ B	<table><tr><th>A</th><th>B</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Z	0	0	0	0	1	1	1	0	1	1	1	0
			A	B	Z													
			0	0	0													
			0	1	1													
			1	0	1													
1	1	0																
NOT		Z=A' = $\overline{A}$	<table><tr><th>A</th><th>Z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Z	0	1	1	0									
			A	Z														
			0	1														
1	0																	

Βασικές Λογικές Πράξεις

Επίσης θυμίζουμε ότι οι πύλες AND, OR, EXOR σε συνδυασμό με την NOT υλοποιούν τις παράγωγες πύλες NAND, NOR, XNOR αντίστοιχα οι οποίες, φυσικά, περιγράφονται

με αντίστροφους πίνακες αληθείας σε σχέση με τις βασικές πύλες από τις οποίες προέρχονται.

Οι παράγωγες πύλες δεν υποστηρίζονται από συγκεκριμένες εντολές του ρεπερτορίου εντολών του AVR και κατά συνέπεια όταν πρέπει να παραχθεί το αντίστοιχο αποτέλεσμα στο κώδικάς μας θα πρέπει να χρησιμοποιήσουμε συνδυαστικά τις υπόλοιπες εντολές.

Συγκεκριμένα οι εντολές που υποστηρίζονται είναι οι εξής.

1. Λογική Πράξη AND

AND Rd,Rr Rd $\leftarrow$ Rd•Rr	Περιγραφή: Εκτέλεση της λογικής πράξης AND μεταξύ των περιεχομένων των καταχωρητών Rd,Rr και αποθήκευση του αποτελέσματος στον Rd. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
---	---

ANDI Rd,K Rd $\leftarrow$ Rd•K	Περιγραφή: Εκτέλεση της λογικής πράξης AND μεταξύ των περιεχομένων του καταχωρητή Rd και της ποσότητας K και αποθήκευση του αποτελέσματος στον Rd. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
--	---

Σημαντική Παρατήρηση: Στην περίπτωση της AND σαν Rd,Rr μπορούν να χρησιμοποιηθούν οποιοδήποτε καταχωρητές εκ των 32 γενικού σκοπού (R1-R31). Στην περίπτωση της ANDI σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε εκ των 16 τελευταίων καταχωρητών γενικού σκοπού (R16-R31).

2. Λογική Πράξη OR

OR Rd,Rr Rd $\leftarrow$ Rd+Rr	Περιγραφή: Εκτέλεση της λογικής πράξης OR μεταξύ των περιεχομένων των καταχωρητών Rd,Rr και αποθήκευση του αποτελέσματος στον Rd. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
--	--

ORI Rd,K Rd ← Rd+K	Περιγραφή: Εκτέλεση της λογικής πράξης OR μεταξύ των περιεχομένων του καταχωρητή Rd και της ποσότητας K και αποθήκευση του αποτελέσματος στον Rd. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
-------------------------------------	---

Σημαντική Παρατήρηση: Στην περίπτωση της OR σαν Rd,Rr μπορούν να χρησιμοποιηθούν οποιοιδήποτε καταχωρητές εκ των 32 γενικού σκοπού (R1-R31). Στην περίπτωση της ORI σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε εκ των 16 τελευταίων καταχωρητών γενικού σκοπού (R16-R31).

3. Λογική Πράξη XOR

EOR Rd,Rr Rd ← Rd ⊕ Rr	Περιγραφή: Εκτέλεση της λογικής πράξης XOR μεταξύ των περιεχομένων των καταχωρητών Rd,Rr και αποθήκευση του αποτελέσματος στον Rd. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
---	--

Σημαντική Παρατήρηση: Σαν Rd,Rr μπορούν να χρησιμοποιηθούν οποιοιδήποτε καταχωρητές εκ των 32 γενικού σκοπού (R1-R31).

4. Λογική Πράξη NOT

COM Rd Rd ← Rd'	Περιγραφή: Υπολογίζει τον αρνητικό αριθμό Rd σε μορφή του συμπληρώματος του 1. Ως γνωστό αυτό γίνεται αντιστρέφοντας όλα τα bit της ποσότητας που είναι υποθηκευμένη στον Rd. Γίνεται λοιπόν κατανοητό ότι η ενέργεια αυτή είναι ισοδύναμη με τη εφαρμογή της πράξης NOT στη ποσότητα του Rd. Ενημέρωση σημαίων: Z,C,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
----------------------------------	---

Σημαντική Παρατήρηση: Σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε καταχωρητής εκ των 32 γενικού σκοπού (R1-R31).

Μια ιδιαίτερα σημαντική χρήση των εντολών αυτών είναι ότι βασιζόμενοι στους πίνακες αληθείας των λογικών πυλών οι αντίστοιχες εντολές μπορούν να χρησιμοποιηθούν για να θέσουν ή να καθαρίσουν συγκεκριμένα bit σε μια 8bit ποσότητα. Η συγκεκριμένη λειτουργία συναντάται και ως «εφαρμογή μάσκας» σε μια αριθμητική ποσότητα. Κάποιοι σχετικοί κανόνες που μπορούν εύκολα να εξαχθούν από τους πίνακες αληθείας είναι.

Όταν ένα από τα δύο ορίσματα σε μία πράξη AND είναι 0 τότε το αποτέλεσμα είναι πάντα 0 ανεξαρτήτως του δεύτερου ορίσματος.

Όταν ένα από τα δύο ορίσματα σε μία πράξη AND είναι 1 τότε το αποτέλεσμα είναι πάντα η τιμή του δεύτερου ορίσματος.

Όταν ένα από τα δύο ορίσματα σε μία πράξη OR είναι 1 τότε το αποτέλεσμα είναι πάντα 1 ανεξαρτήτως του δεύτερου ορίσματος.

Όταν ένα από τα δύο ορίσματα σε μία πράξη OR είναι 0 τότε το αποτέλεσμα είναι πάντα η τιμή του δεύτερου ορίσματος.

Το αποτέλεσμα μιας πράξης XOR μεταξύ της ίδια ποσότητας είναι πάντα 0.

Επιπλέον πολύ σημαντικές και ευρέως χρησιμοποιούμενες είναι οι εντολές αύξησης των περιεχομένων των καταχωρητών. Αυτές είναι:

INC Rd Rd ← Rd+1	Περιγραφή: Αυξάνει τα περιεχόμενα του καταχωρητή Rd κατά 1. Ενημέρωση σημαίων: Z,N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1
DEC Rd Rd ← Rd-1	Περιγραφή: Μειώνει τα περιεχόμενα του καταχωρητή Rd κατά 1 Ενημέρωση σημαίων: Z, N,V Παράδειγμα: Αναλυτικά παρακάτω. Περίοδοι ρολογιού: 1

Παραδείγματα

Παράδειγμα 1^ο: Χρήση των εντολών λογικών πράξεων του AVR

Γράψτε πρόγραμμα το οποίο θα αποθηκεύει στις διευθύνσεις \$70, \$71, \$72, \$73 του αριθμούς 0xEE, 0xA6, 0x03 και 0x3B αντίστοιχα. Κατόπιν να γίνουν οι ακόλουθες ενέργειες

Να γίνει η λογική πράξη AND μεταξύ της θέσης \$70 και του αριθμού 0xCB και το αποτέλεσμα να αποθηκευτεί στην \$74

Να γίνει η λογική πράξη OR μεταξύ της θέσης \$71 και \$72 και το αποτέλεσμα να αποθηκευτεί στην \$75

Να γίνει η λογική πράξη XOR μεταξύ της θέσης \$74 και του αριθμού 0xAA και το αποτέλεσμα να αποθηκευτεί στην \$76

Να αντιστραφεί το περιεχόμενο της θέσης \$73

Να γίνει η λογική πράξη NAND μεταξύ των θέσεων μνήμης \$75 και \$76 το αποτέλεσμα να αποθηκευτεί στην \$76

Χρησιμοποιώντας τις εντολές INC και DEC αυξήστε τα περιεχόμενα της θέσης μνήμης \$76 κατά 3 και μειώστε τα περιεχόμενα της \$75 κατά 2.

Λύση:

```
.include "8515def.inc"
```

```
;Αρχικοποίηση θέσεων μνήμης
```

```
LDI R16, $EE
```

```
STS $70, R16
```

```
LDI R16, $A6
```

```
STS $71, R16
```

```
LDI R16, $03
```

```
STS $72, R16
```

```
LDI R16, $3B
```

```
STS $73, R16
```

```
; Ερώτημα I
```

```
LDS R17, $70
```

```
ANDI R17, $CB
```

```
STS $74, R17
```

```
; Ερώτημα II
```

```
LDS R17, $71
```

```
LDS R18, $72
```

```
OR R17, R18
```

```
STS $75, R17
```

```
; Ερώτημα III
```

```
LDS R17, $74
```

```
LDI R18, $AA
```

```
EOR R17, R18
```

```
STS $76, R17
```

```
; Ερώτημα IV
```

```
LDS R17, $73
```

```
COM R17
STS $73, R17
; Ερώτημα V
LDS R17, $75
LDS R18, $76
AND R17, R18
COM R17
STS $76, R17
; Ερώτημα VI
LDS R17, $75
LDS R18, $76
INC R18
INC R18
INC R18
DEC R17
DEC R17
STS $75, R17
STS $76, R18
```

Παράδειγμα 2º: Χρήση των εντολών λογικών πράξεων του AVR για θέση ή καθαρισμό συγκεκριμένων bit σε αριθμητικές ποσότητες

Να γραφτεί πρόγραμμα το οποίο να εκτελεί τα ακόλουθα:

Να μηδενίζονται τα 2 τελευταία bit της θέσης μνήμης \$80 ενώ τα υπόλοιπα να μένουν αμετάβλητα.

Να τίθεται τα bit 0 και 5 της θέσης μνήμης \$81 ενώ τα υπόλοιπα να μένουν αμετάβλητα.

Να μηδενίσετε τη τιμή της θέσης \$82 χρησιμοποιώντας της εντολή EOR.

Λύση:

```
.include "8515def.inc"
; Ερώτημα I
LDS R17, $80
ANDI R17, 0b00111111
STS $80, R17
; Ερώτημα II
LDS R17, $81
ORI R17, 0b00100001
```

STS \$81, R17

; Ερώτημα III

LDS R17, \$82

EOR R17, R17

STS \$82, R17

ΑΣΚΗΣΗ 5 – Εντολές διακλάδωσης ή μεταφοράς ελέγχου

Γενικά

Οι εντολές διακλάδωσης ή μεταφοράς ελέγχου χρησιμεύουν στη δημιουργία διακλαδώσεων μέσα σ' ένα πρόγραμμα, αν θέλουμε να παρακάμψουμε μια σειρά από εντολές όταν ικανοποιείται ή όταν δεν ικανοποιείται κάποια συνθήκη. Επίσης χρησιμεύουν στις διαδικασίες ανακυκλώσεων και επανεκτελέσεων τμημάτων του προγράμματος. Στις εντολές διακλάδωσης ανήκουν και οι εντολές κλήσης και επιστροφής από υπορουτίνα. Μερικές από τις εντολές αυτές δεν έχουν όρισμα. Όσες όμως έχουν, έχουν σαν όρισμα την ετικέτα της εντολής, στην οποία θα γίνει η διακλάδωση. Οι εντολές διακλάδωσης αλλάζουν την ροή του προγράμματος αντικαθιστώντας τα περιεχόμενα του PC (Program Counter) με την διεύθυνση της εντολής που θα γίνει η διακλάδωση.

Οι εντολές διακλάδωσης μπορούν να χωριστούν σε δύο κατηγορίες: τις εντολές διακλάδωσης χωρίς συνθήκη και τις εντολές διακλάδωσης υπό συνθήκη.

Διακλάδωση χωρίς συνθήκη

Με τις εντολές αυτές μπορεί να γίνει “άνευ όρων” διακλάδωση, δηλαδή διακλάδωση χωρίς συνθήκη. Οι παρακάτω εντολές μας παρέχουν τη δυνατότητα να εκτελέσουμε άλμα σε επιθυμητή διεύθυνση, να καλέσουμε μια ρουτίνα και να επιστρέψουμε από αυτήν.

RJMP k PC ← PC + k + 1	Περιγραφή: Relative Jump. Σχετικό άλμα σε μια διεύθυνση της μνήμης προγράμματος μεταξύ PC - 2K + 1 και PC + 2K. Το k είναι μία ετικέτα (label) στην αριστερή στήλη του προγράμματος. Σαν ετικέτα μπορούμε να βάλουμε ότι θέλουμε αρκεί να είναι σε λατινικούς χαρακτήρες, να μην είναι το όνομα κάποιας εντολής και να μην έχει κενά. Ενημέρωση σημαίων: καμία Παράδειγμα: rjmp ok error: add r16,r17 inc r16 ok: nop Περίοδοι ρολογιού: 2
IJMP PC ← Z(15:0)	Περιγραφή: Indirect Jump to (Z). Άλμα στην διεύθυνση μνήμης flash που ορίζεται από τον δείκτη Z (Z = R31,R30). Χρησιμοποιείται σπάνια σε περιπτώσεις που θέλουμε να κάνουμε μεγάλο άλμα μέσα στη μνήμη. Ενημέρωση σημαίων: καμία Παράδειγμα: ldi r30,low ldi r31,high ijmp

	Περίοδοι ρολογιού: 2
--	----------------------

JMP $PC \leftarrow k$	Περιγραφή: Direct Jump. Απλό άλμα σε μια θέση της μνήμης προγράμματος. Ενημέρωση σημαιών: καμία Παράδειγμα: jmp farplc ; άλμα χωρίς συνθήκη ; farplc: nop ;
---------------------------------	--

Εντολές Κλίσης Υπορουτινών

Με τις εντολές αυτές μπορεί να γίνει κλήση σε μία ρουτίνα.

CALL k $PC \leftarrow k$	Περιγραφή: Direct Subroutine Call. Κλήση υπορουτίνας εντός της μνήμης προγράμματος . Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα. Ενημέρωση σημαιών: καμία Παράδειγμα: call delay Περίοδοι ρολογιού: 4/5
------------------------------------	--

RET $PC \leftarrow Stack$	Περιγραφή: Subroutine Return. Επιστροφή από υπορουτίνα. Η διεύθυνση επιστροφής φορτώνεται από τη στοίβα. Ενημέρωση σημαιών: καμία Παράδειγμα: pop sreg ret Περίοδοι ρολογιού: 4/5
-------------------------------------	---

RCALL k $PC \leftarrow PC + k + 1$	Περιγραφή: Σχετική κλήση ρουτίνας σε μια διεύθυνση μεταξύ $PC - 2K + 1$ και $PC + 2K$. Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα. Ενημέρωση σημαιών: καμία Παράδειγμα: rcall routine ; κλήση υπορουτίνας ; routine: push r14 ; αποθήκευση καταχωρητή στη στοίβα ; pop r14 ; ανάκτηση καταχωρητή από τη στοίβα ret ; επιστροφή από υπορουτίνα Περίοδοι ρολογιού: 3/4
--	---

Διακλάδωση υπό συνθήκη

α) Διακλαδώσεις που γίνεται έλεγχος byte:

Οι εντολές με τις οποίες γίνεται “υπό συνθήκη” διακλάδωση μπορούν να ελέγχουν είτε ένα bit είτε ένα byte αν ικανοποιεί κάποια συνθήκη που όταν αληθεύει θα λάβει χώρα το άλμα. Στην ουσία εξετάζουν μία από τις επτά σημαίες του καταχωρητή κατάστασης (SREG) για να αποφασίσουν αν πρέπει να γίνει το άλμα ή όχι. Οι σημαίες αυτές επηρεάζονται από τις πράξεις και από κάποιες άλλες εντολές που θα δούμε παρακάτω. Πριν εξετάσουμε αυτές τις περιπτώσεις ας δούμε εν συντομία τον SREG:

Bit	7	6	5	4	3	2	1	0	
\$3F (\$5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Ο καταχωρητής κατάστασης (SREG) είναι ένας 8-bit καταχωρητής που κάθε φορά που εκτελούνται αριθμητικές πράξεις, ανανεώνεται (refresh) με σημαίες που έχουν να κάνουν με το αποτέλεσμα της πράξης.

I (Global Interrupt Enable): Επιτρέπει τις διακοπές (Interrupts) στον επεξεργαστή.

T (Target) : Οι εντολές αντιγραφής ενός Bit από μια θέση μνήμης έχουν σαν θέση αποθήκευσης το συγκεκριμένο πεδίο.

H (Half Carry): Κρατούμενο που προκύπτει από τα τέσσερα λιγότερο σημαντικά ψηφία.

S (Sign): Δείκτης προσήμου, αποτέλεσμα του «αποκλειστικού OR» EXOR μεταξύ των πεδίων V και N.

V (Overflow): Δείχνει υπερχείλιση σε πράξεις συμπληρώματος ως προς 2

N (Negative): Δηλώνει ότι ο αριθμός που προκύπτει από μια αριθμητική ή λογική πράξη είναι αρνητικός

Z (Zero): Δηλώνει ότι το αποτέλεσμα μιας πράξης είναι 0

C (Carry): Δείκτης που δηλώνει ότι προέκυψε κρατούμενο από μια αριθμητική ή λογική πράξη.

Στην πράξη οι (πολύ) περισσότερο χρησιμοποιούμενες - λαμβανόμενες υπ όψιν σημαίες είναι οι I , N , Z , C ενώ οι T , H , S , V έχουν δευτερεύουσα σημασία.

Δεν θα δούμε όλες τις εντολές διακλάδωσης υπό συνθήκη, αλλά αυτές που χρησιμοποιούνται πιο συχνά:

CP Rd,Rr Rd - Rr	Περιγραφή: Compare. Σύγκριση με υπολογισμό της διαφοράς και χρήση κρατουμένου μεταξύ του δυο καταχωρητών. Ενημέρωση σημαιών: Z,C,N,S,V,H Παράδειγμα: cp r4,r19
-----------------------------------	--

	brne noteq ; διακλάδωση σε περίπτωση ανισότητας Περίοδοι ρολογιού: 1
--	---

CPC Rd,Rr Rd – Rr-C	Περιγραφή: Compare with Carry. Σύγκριση με υπολογισμό της διαφοράς και χρήση κρατουμένου μεταξύ των καταχωρητών Rd,Rr. Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: cpr r2,r0 cpc r3,r1 breq equal ; διακλάδωση σε περίπτωση ισότητας Περίοδοι ρολογιού: 1
--------------------------------------	---

CPI Rd,K Rd - K	Περιγραφή: Compare Register with Immediate. Σύγκριση με υπολογισμό της διαφοράς και χρήση κρατουμένου μεταξύ του καταχωρητή Rd και μιας σταθεράς. Ενημέρωση σημαίων: Z,C,N,S,V,H Παράδειγμα: cpi r19,3 brne error ; διακλάδωση σε περίπτωση ανισότητας Περίοδοι ρολογιού: 1
----------------------------------	---

CPSE Rd, Rr Av Rd = Rr τότε PC ← PC + 2 (ή 3) αλλιώς PC ← PC + 1	Περιγραφή: Compare, Skip if Equal. Σύγκριση μεταξύ των καταχωρητών Rd, Rr. Σε περίπτωση ισότητας παρακάμπτεται η επόμενη εντολή. Ενημέρωση σημαίων: καμία Παράδειγμα: cpse r4,r0 neg r4 ;εκτέλεση όταν r4<>r0 nop Περίοδοι ρολογιού: 1/2/3
---	---

Οι εντολές CP, CPI χρησιμοποιούνται απλά για να κάνουν μια σύγκριση. Για το πώς θα εκμεταλλευτούμε το αποτέλεσμα αυτής της σύγκρισης χρησιμοποιούμε τις εντολές "BRANCH":

BRXX k: όπου XX μπορεί να είναι EQ, LO, NE, GE κλπ. Αφού έχουν συγκριθεί 2 Registers, ανάλογα με την κατάσταση που έχουν οι σημαίες στον SREG γίνεται διακλάδωση στο ανάλογο label (στην διεύθυνση PC+k+1 της μνήμης flash με $-64 < k < +63$)

BRNE k Av Rd <> Rr (Z = 0) τότε PC ← PC + k + 1, αλλιώς PC ← PC + 1	Περιγραφή: Branch if Not Equal. Σχετικό άλμα σε περίπτωση ανισότητας. Όταν η σημαία μηδενός ισούται με 0 (Z=0) τότε εκτελείται σχετική διακλάδωση. Ενημέρωση σημαίων: καμία Παράδειγμα: cpr r1,5 brne notequal
--	---

	... notequal: nop ; προορισμός διακλάδωσης Περίοδοι ρολογιού: ½
--	---

BREQ k Αν $R_d = R_r$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$	Περιγραφή: Branch if Equal. Σχετικό άλμα σε περίπτωση ισότητας. Ενημέρωση σημαιών: Z,C,N,S,V,H Ενημέρωση σημαιών: καμία Παράδειγμα: cp r1,r0 breq equal ... equal: nop ; προορισμός διακλάδωσης Περίοδοι ρολογιού: ½
--	---

BRGE k Αν $R_d \geq R_r$ ($S = N \oplus V = 0$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$	Περιγραφή: Branch if Greater or Equal, Signed. Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μεγαλύτερο ή ίσο ($S = 0$) τότε εκτελείται σχετική διακλάδωση. Ενημέρωση σημαιών: καμία Παράδειγμα: cp r21,r19 ; brge greateq ... greateq: Περίοδοι ρολογιού: 1/2
---	--

BRLO k Αν $R_d < R_r$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$	Περιγραφή: Branch if Lower. Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μικρότερο ($S = 0$) τότε εκτελείται σχετική διακλάδωση. Ενημέρωση σημαιών: καμία Παράδειγμα: loop: inc r19 cpi r19,\$10 ; brlo loop Περίοδοι ρολογιού: 1/2
---	---

Άλλες εντολές "BRANCH":

BRBS s, k	Branch if Status Flag Set if ($SREG(s) = 1$) then $PC \leftarrow PC + k + 1$
BRBC s, k	Branch if Status Flag Cleared if ($SREG(s) = 0$) then $PC \leftarrow PC + k + 1$
BRCS k	Branch if Carry Set if ($C = 1$) then $PC \leftarrow PC + k + 1$
BRCC k	Branch if Carry Cleared if ($C = 0$) then $PC \leftarrow PC + k + 1$
BRSH k	Branch if Same or Higher if ($C = 0$) then $PC \leftarrow PC + k + 1$
BRMI k	Branch if Minus if ($N = 1$) then $PC \leftarrow PC + k + 1$
BRPL k	Branch if Plus if ($N = 0$) then $PC \leftarrow PC + k + 1$

BRLT	k	Branch if Less Than Zero, Signed	if (NV = 1) then PC ← PC + k + 1
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1

β) Διακλαδώσεις που γίνεται έλεγχος ενός bit:

Κατ' ουσίαν όλες οι παραπάνω εντολές Branch εξετάζουν κάποιο bit στον SREG και ανάλογα μεταβαίνει ο PC. Όμως υπάρχει η δυνατότητα να εξεταστεί ένα bit και να πραγματοποιηθεί διακλάδωση χωρίς τη μεσολάβηση της εντολής CP, CPI κλπ. Γίνεται έλεγχος bit κατ' ευθείαν σε κάποιο από τα 8 bit ενός register ή μιας I/O πόρτας. Χρησιμοποιείται η εντολή Skip if Bit in (Register είτε I/O) is (Set είτε Clear)

SBRS Rr,b Rr, b	Περιγραφή: Skip if Bit in Register is Set. Αγνοείται η επόμενη εντολή αν το bit b με $0 < b < 7$ σε έναν καταχωρητή είναι σε λογικό 1. Ενημέρωση σημαίων: καμία Παράδειγμα: loop: in r4,PINB sbrs r4,2 rjmp loop clr r4 Περίοδοι ρολογιού: 1/2/3
----------------------------------	---

SBRC Rd,b Rr, b	Περιγραφή: Skip if Bit in Register Cleared. Αγνοείται η επόμενη εντολή αν το bit b με $0 < b < 7$ σε έναν καταχωρητή είναι σε λογικό 0. Ενημέρωση σημαίων: Z,C,N,S,V,H Ενημέρωση σημαίων: καμία Παράδειγμα: loop: in r4,PINB sbrc r4,1 rjmp loop ser r4 Περίοδοι ρολογιού: 1/2/3
----------------------------------	--

SBIS A,b	Περιγραφή: Skip if Bit in I/O Register is Set. Αγνοείται η επόμενη εντολή αν το bit b με $0 < b < 7$ σε μία I/O θέση μνήμης A είναι σε λογικό 1.
-----------------	--

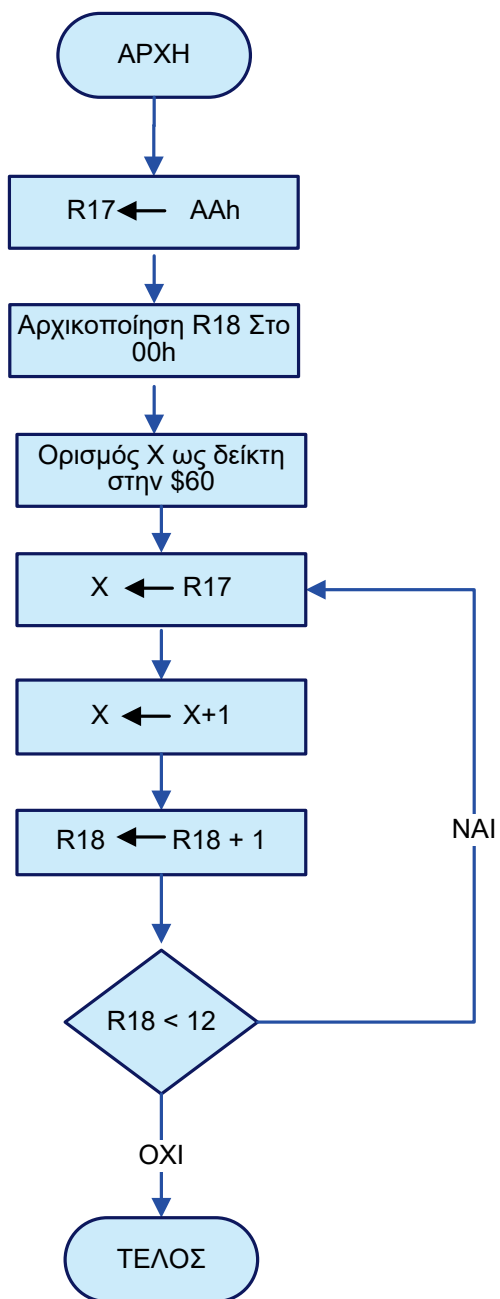
Αν $I/O(A,b) = 1$ τότε $PC \leftarrow PC + 2$ (ή 3) αλλιώς $PC \leftarrow PC + 1$	Ενημέρωση σημαιών: καμία Παράδειγμα: wait: sbis PINB,0 rjmp wait Περίοδοι ρολογιού: 1/2/3
---	--

SBIC A,b Αν $Rd < Rr$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$	Περιγραφή: Skip if Bit in I/O Register Cleared. Αγνοείται η επόμενη εντολή αν το bit b με $0 < b < 7$ σε μία I/O θέση μνήμης A είναι σε λογικό 0. Ενημέρωση σημαιών: καμία Παράδειγμα: e2wait: sbic \$1C,1 ; Αν το E2WE είναι 0,παρέκαμψε επόμενη εντολή rjmp e2wait ;η εγγραφή της EEPROM δεν ολοκληρώθηκε nop ; Περίοδοι ρολογιού: 1/2/3
---	---

Παράδειγμα 1ο - Αντιγραφή αριθμού σε συνεχόμενες θέσεις μνήμης

Γράψτε ένα πρόγραμμα στο οποίο θα αποθηκεύεται ο αριθμός AAh στις θέσεις μνήμης από \$60 έως και \$6A. Επιβεβαιώστε την εγγραφή στις συγκεκριμένες θέσεις μνήμης. Υπόδειξη: Αρχικά να γίνει διάγραμμα

```
.include "8515def.inc"
; R17 ← $AA, αυτό που θα αποθηκεύεται κάθε φορά
LDI R17,$AA
; R18 ← $00, χρήση ως μετρητή
LDI R18,$00
; X = $0060
LDI XL,$60
LDI XH,$00
LOOP:
; (X) ← R17 και X = X+1
ST X+,R17
; R18 = R18 + 1
INC R18
; Εάν R18 <= 10 επιστροφή στο LOOP
CPI R18,11
BRLO LOOP
```



ΑΣΚΗΣΗ 6 – Εύρεση Μεγίστου / Ελαχίστου

Μετά την εισαγωγή στην έννοια του βρόχου (την επαναληπτική εκτέλεση ενός κομματιού κώδικα σύμφωνα με ένα συγκεκριμένο κριτήριο) στόχος της παρούσας άσκησης είναι η ανάπτυξη, εκτέλεση και κατανόηση ενός ρεαλιστικού αλγορίθμου στον οποίον βρίσκουν χρήση τα αντίστοιχα στοιχεία.

Ο αλγόριθμος εύρεσης μέγιστου ή/και ελάχιστου αποτελεί έναν ιδανικό τέτοιο αλγόριθμο. Μπορεί να αποτελέσει μέρος ενός σε μεγαλύτερου προγράμματος σε αναρίθμητες περιπτώσεις ή να και να χρησιμοποιηθεί ως αυτόνομη εφαρμογή.

Όπως γίνεται εύκολα κατανοητό στόχος του συγκεκριμένου αλγορίθμου είναι η εύρεση της μέγιστης ή/και ελάχιστης αριθμητικής τιμής σε μια συγκεκριμένη και σαφώς οριοθετημένη περιοχή μνήμης.

Αν θεωρήσουμε λοιπόν μια περιοχή μνήμης με αρχική θέση μνήμης την \$70 και τελική την \$7A και στόχο την εύρεση του μέγιστου αποθηκεμένου αριθμού και μεταφορά αυτού στην θέση μνήμης \$80 το αντίστοιχο διάγραμμα φαίνεται στην Εικόνα 4.1. Στο διάγραμμα αυτό χρησιμοποιούνται είναι ψευδώνυμα που έχουν δοθεί σε συγκεκριμένους καταχωρητές (με χρήση της ψευδοεντολής .DEF) έτσι ώστε να διευκολυνθεί η ανάπτυξη αλλά και η κατανόηση του κώδικα.

Συγκεκριμένα:

MEGALYTERO → Περιέχει την μέγιστη τιμή που έχει βρεθεί ανά πάσα στιγμή και φυσικά θα περιέχει τη μέγιστη τιμή της περιοχής μνήμης μετά το τέλος του προγράμματος.

COUNTER → Βοηθητικός μετρητής ο οποίος περιέχει το πλήθος των θέσεων μνήμης που έχουν ελεγχθεί και κατά συνέπεια αποτελεί το κριτήριο για το πότε έχει ολοκληρωθεί η διαδικασία.

MEM_UNDER_TEST → Βοηθητικός καταχωρητής ο οποίος περιέχει το περιεχόμενο της μνήμης που ελέγχεται αν πάσα στιγμή.

Επιπλέον μια πιθανή υλοποίηση του αντίστοιχου προγράμματος είναι η ακόλουθη:

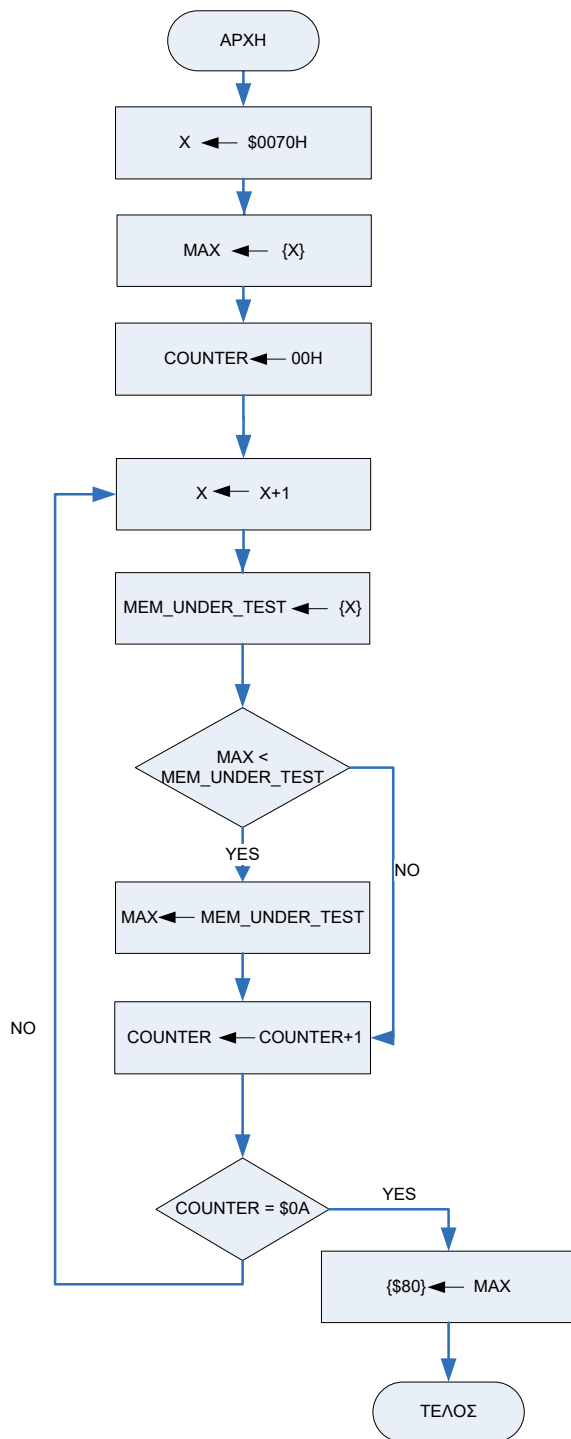
```
.include "8515def.inc"
.DEF MEGALYTERO=R20
.DEF COUNTER=R18
.DEF MEM_UNDER_TEST=R17
LDI R27, $00
LDI R26,$70
LD MEGALYTERO,X
LDI COUNTER, 0
NEXT_CHECK: INC R26
LD MEM_UNDER_TEST,X
CP MEGALYTERO, MEM_UNDER_TEST
```



```

BRCC NO_MAX_UPDATE
MOV MEGALYTERO, MEM_UNDER_TEST
NO_MAX_UPDATE: INC COUNTER
CPI COUNTER, $0A
BRNE NEXT_CHECK
STS $80, MAX

```



ΑΣΚΗΣΗ 7 – Συγκρίσεις, Διακλαδώσεις & Βρόγχοι - Διαγράμματα Ροής

Παραδείγματα:

Παράδειγμα 1ο - Συγκρίσεις, Διακλαδώσεις & Βρόγχοι - Διαγράμματα Ροής

Να γραφεί πρόγραμμα το οποίο αρχικά να αποθηκεύει δύο τυχαίους αριθμούς στις θέσεις μνήμης \$130h και \$131h. Στην συνέχεια να τοποθετεί τον μεγαλύτερο στην θέση μνήμης \$132h και τον μικρότερο στην \$133h. Στην περίπτωση όμως που είναι ίσοι οι δύο αριθμοί, να το τοποθετεί το 00H στην θέση μνήμης \$132H και στη θέση μνήμης \$133h τον αριθμό. Τρέξτε το πρόγραμμα δύο φορές, μια με άνισους αριθμούς για να επιβεβαιώσετε την εύρεση του μέγιστου / ελάχιστου και μια με ίσους για να επιβεβαιώσετε την λειτουργία του στην ισότητα. Υπόδειξη: Να γράψετε το διάγραμμα ροής.

```
.include "8515def.inc"
LDI R18,$A2
STS $130,R18
LDI R19,$F1
STS $131,R19
CP R18,R19
BREQ ISO
BRLO MIKROTERO
STS $132,R18
STS $133,R19
RJMP END
ISO: STS $133,R19
LDI R20,$00
STS $132,R20
RJMP END
MIKROTERO: STS $132,R19
STS $133,R18
END:
```

ΑΣΚΗΣΗ 8 – I/O (Ports, Push Buttons, Leds), Υπορουτίνες Καθυστερήσης

Σε αυτή την εργαστηριακή άσκηση γίνεται μία γνωριμία με τα I/O (Inputs/Outputs), στην συγκεκριμένη περίπτωση θα χρησιμοποιήσουμε ως outputs τα leds, και ως inputs τους διακόπτες. Στόχος της παρούσας άσκησης είναι η χρήση των I/O δυνατοτήτων του 8515 και του STK500 kit, χρησιμοποιώντας τα leds και τους διακόπτες. Παράλληλα θα γίνει και μία εισαγωγή στις υπορουτίνες και πιο συγκεκριμένα στην κατασκευή μίας υπορουτίνας καθυστέρησης (delay).

Όλοι οι μικροελεγκτές της οικογένειας AVR διαθέτουν μεταξύ άλλων και κάποιο αριθμό ακροδεκτών εισόδου – εξόδου, ξεκινώντας από 3 που συναντούμε στον μικροελεγκτή AT90S2323, μέχρι και 48, που συναντούμε στον μικροελεγκτή Mega103. Όλοι οι ακροδέκτες εισόδου – εξόδου των μικροελεγκτών AVR έχουν τη δυνατότητα να παρέχουν ρεύμα μέχρι 20 mA, ποσότητας ικανής για απευθείας οδήγηση διόδων LED χωρίς την ανάγκη χρήσης κατάλληλης εξωτερικής μονάδας απομόνωσης (buffer).

Για όλες τις θύρες εισόδου – εξόδου του συστήματος υπάρχουν συνολικά τρεις διευθύνσεις στη μνήμη εισόδου – εξόδου, οι οποίες σχετίζονται με κάθε μία από αυτές.

α) Η μία από τις διευθύνσεις αυτές απαιτείται για τον καθορισμό της κατεύθυνσης των ακροδεκτών, δηλαδή καθορίζει ποιοι από τους ακροδέκτες θα λειτουργούν ως είσοδοι και ποιοι ως έξοδοι και συμβολίζονται ως DDRx. Ο καταχωρητής DDRx είναι ο λεγόμενος καταχωρητής κατεύθυνσης. Τοποθετώντας ένα bit του καταχωρητή αυτού σε λογικό 1, ο ακροδέκτης της θύρας PORTx στον οποίο αντιστοιχεί, θα λειτουργεί ως έξοδος. Μετά από αυτή την ενέργεια μπορούμε να οδηγούμε κατά βούληση την λογική στάθμη του αντίστοιχου ακροδέκτη εξόδου χρησιμοποιώντας τις εντολές CBI ή SBI, ή ακόμη και τη εντολή OUT.

β) Η δεύτερη διεύθυνση αφορά στα δεδομένα που πρόκειται να εγγραφούν σε εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως έξοδοι και συμβολίζονται ως PORTx.

γ) Τέλος, η τρίτη διεύθυνση αφορά στα δεδομένα που διαβάζονται από εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως είσοδοι συμβολίζονται ως PINx.

Οι πιο απλές εντολές εισόδου-εξόδου είναι η **'in'** και **'out'**.

Η **'in'** διαβάζει την τιμή μιας θύρας ή του καταχωρητή ενός εσωτερικού περιφερειακού (χρονιστή, UART) και τη σώζει σε έναν καταχωρητή.

Σύνταξη:	Καταχωρητές:	Διευθύνσεις:
IN Rd,A	$0 \leq d \leq 31$	$0 \leq A \leq 63$

Παράδειγμα:

```

.include "m8515def.inc"
in r25,PORTB ; Διάβασε την Port B
cpi r25,4 ; Σύγκρινε την τιμή με την σταθερά "4"
breq exit ; Διακλάδωσε αν r25=4
...
exit: nop ; Προορισμός διακλάδωσης (κενός κύκλος)

```

Η **‘out’** γράφει την τιμή ενός κατ/τη σε μια θύρα ή στον κατ/τη ενός εσωτερικού περιφερειακού.

Σύνταξη:	Καταχωρητές:	Διευθύνσεις:
OUT A,Rr	$0 \leq r \leq 31$	$0 \leq A \leq 63$

Παράδειγμα:

```

.include "m8515def.inc"
clr r16 ; Καθάρισε τον r16
ser r17 ; Όλα "1" στον r17
out PORTB,r16 ; Γράψε "0" στην Port B
nop ; Περίμενε (κενος κύκλος)
out PORTB,r17 ; Γράψε "1" στην Port B

```

Επίσης, υπάρχουν οι εντολές **‘sbi’** και **‘cbi’** για χειρισμό μεμονωμένων ψηφίων μιας θύρας και οι **sbic**,

sbis για τον έλεγχο των ψηφίων μιας θύρας I/O.

Μετατροπή θύρας σε είσοδο:

```

LDI R12, 0b00000000 ; τα ψηφία της θύρας PORTD
OUT DDRD, R12 ; γίνονται είσοδοι

```

Εναλλακτικά η PORTD μετατρέπεται σε είσοδο :

```

LDI R12, 0x00 ; τα ψηφία της θύρας PORTD
OUT DDRD, R12 ; γίνονται είσοδοι

```

ή

CLR R21 ; τα ψηφία της θύρας PORTD γίνονται είσοδοι

OUT DDRD, R21 ;μετατροπή θύρας σε έξοδο

LDI R18, 0b11111111 ; τα ψηφία της θύρας PORTB

OUT DDRB, R18 ; γίνονται έξοδοι

Εναλλακτικά η PORTB μετατρέπεται σε έξοδο

LDI R22, 0xFF ; τα ψηφία της θύρας PORTB

OUT DDRB, R22 ; γίνονται έξοδοι

ή

SER R19 ; τα ψηφία της θύρας PORTB

OUT DDRB, R19 ; γίνονται έξοδοι

Σημαντικές Παρατηρήσεις:

Για να κατανοήσουμε τον κώδικα που ακολουθεί αλλά και τις ενδείξεις που έχουμε στα LEDs πρέπει να έχουμε πάντα υπόψη ότι το LED και οι διακόπτες στο STK 500 kit ακολουθούν ACTIVE LOW λογική. Δηλαδή για να φωτοβολήσει ένα LED στο STK500 στο αντίστοιχο PIN εξόδου πρέπει να στείλουμε 0 ενώ αν στείλουμε 1 θα παραμείνει σβηστό. Αντίστοιχα ένας πατημένος διακόπτης θα εμφανίσει στο PIN εισόδου 0 ενώ όσο ένας διακόπτης δεν πατιέται το αντίστοιχο PIN εισόδου θα εμφανίζει 1.

Η επιθυμητή καθυστέρηση δημιουργείται καλώντας την αντίστοιχη υπορουτίνα. Κατά συνέπεια θα πρέπει να ορίσουμε και αντίστοιχη, κατάλληλη μνήμη στοίβας όπως φαίνεται στο κώδικα που ακολουθεί.

Παράδειγμα

Χειρισμός LED με χρήση θυρών – Υποπρόγραμμα καθυστέρησης

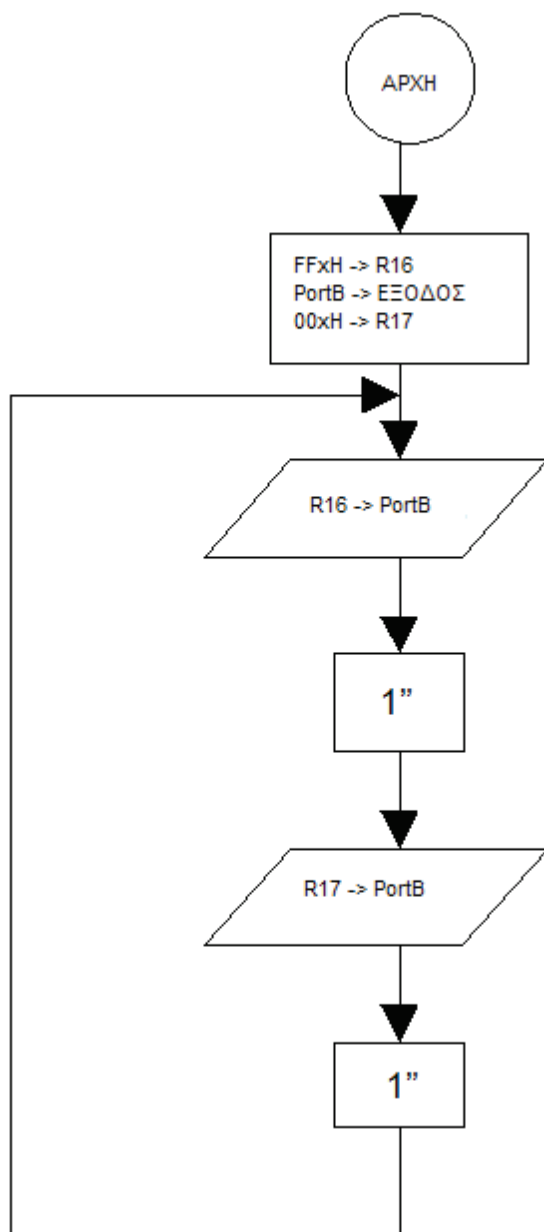
(α) Σχεδιάστε το διάγραμμα ροής και το πρόγραμμα με το οποίο θα αναβοσβήνουν και τα 8 LED με

περιοδικότητα περίπου ένα (1) δευτερόλεπτο. Δηλαδή θα είναι όλα «ON» για 1 δευτερόλεπτο και

όλα «OFF» για επίσης 1 δευτερόλεπτο κ.ο.κ.

(β) Κάνετε το ίδιο με τους αριθμούς 55 και AA , με περιοδικότητα 2 δευτερόλεπτα.

Απάντηση:



```

.include "m8515def.inc"
sp_init: ; Αρχικοποίηση του Stack Pointer
LDI R16, LOW(RAMEND)
OUT SPL,R16
LDI R16, HIGH(RAMEND)
OUT SPH,R16
LDI R16, $FF
OUT DDRB, R16 ; Ορίζουμε την Port B ως έξοδο
CLR R17 ; Όλα "0" στον r17
SER R18 ; Όλα "1" στον r18
loop: OUT PORTB,R17 ; Γράφουμε στην Port B τα περιεχόμενα του R17
  
```

```
RCALL delay ; Καλούμε την υπορουτίνα "delay"
OUT PORTB,R18 ; Γράφουμε στην Port B τα περιεχόμενα του R18
RCALL delay ; Καλούμε την υπορουτίνα "delay"
RJMP loop ; Καλούμε την υπορουτίνα "loop", επιστροφή στην αρχή!
delay:
LDI R20,$0A ; Φόρτωση του μέγιστου αριθμού επαναλήψεων
d1: DEC R20 ; Μείωση (κατά 1) του περιεχομένου του R20
BRNE d1 ; Έως ότου γίνει "0", επιστροφή στο "d1"
RET ; Επιστροφή στην επόμενη γραμμή (από την κλήση)
```

Παράδειγμα Μεγάλης Χρονοκαθυστέρησης (τριπλός βρόγχος):

```
delay:
ldi r26,$FF
d3:ldi r27,$FF
d2: ldi r28,$FF
d1: dec r28
brne d1
dec r27
brne d2
dec r26
brne d3
ret
```

Η άσκηση ξαναπαρουσιάζεται με κάποιες αλλαγές για να τρέξει και στο KIT (αλλάζουμε την delay).

```
.include "m8515def.inc"
sp_init: ; Αρχικοποίηση του Stack Pointer
LDI R16, LOW(RAMEND)
OUT SPL,R16
LDI R16, HIGH(RAMEND)
OUT SPH,R16
LDI R16, $FF
OUT DDRB, R16 ; Ορίζουμε την Port B ως έξοδο
CLR R17 ; Όλα "0" στον r17
SER R18 ; Όλα "1" στον r18
loop: OUT PORTB,R17 ; Γράφουμε στην Port B τα περιεχόμενα του R17
```



```

RCALL delay ; Καλούμε την υπορουτίνα "delay"
RCALL delay ; Καλούμε την υπορουτίνα "delay"
OUT PORTB,R18 ; Γράφουμε στην Port B τα περιεχόμενα του R18
RCALL delay ; Καλούμε την υπορουτίνα "delay"
RCALL delay ; Καλούμε την υπορουτίνα "delay"
RJMP loop ; Καλούμε την υπορουτίνα "loop", επιστροφή στην αρχή!
delay:
; Αρχή εξωτερικού βρόγχου
LDI R22, $FF
LOOP1:
; Αρχή εσωτερικού βρόγχου
LDI R23, $FF
LOOP2:
DEC R23
BRNE LOOP2
; Τέλος εσωτερικού βρόγχου
DEC R22
BRNE LOOP1
; Τέλος εξωτερικού βρόγχου
RET

```

ΑΣΚΗΣΗ 9 – I/O (Ports, Push Buttons, Leds)

Στόχος της παρούσας άσκησης είναι η χρήση των I/O δυνατοτήτων του 8515 και του STK500 kit στα πλαίσια ενός ευρύτερου, ολοκληρωμένου προγράμματος. Μέσα από ένα συνδυαστικό παράδειγμα (και αντίστοιχο εργαστηριακό μέρος) θα καταδειχθεί πως η χρήση των δυνατοτήτων εισόδου και εξόδου πληροφορίας μπορεί να αυξήσει σημαντικά τον έλεγχο, την ευελιξία και γενικότερα τις δυνατότητες που έχει ο χρήστης.

Σαν βάση για το παράδειγμα θα χρησιμοποιήσουμε το κύκλωμα λογικών πυλών, η λειτουργία του οποίου εξομοιώθηκε σε προηγούμενη άσκηση. Επιπρόσθετα, στη παρούσα άσκηση, γίνεται η παραδοχή ότι ο χρήστης επιθυμεί η εκτέλεση του προγράμματος να γίνεται τμηματικά και ο ίδιος να ελέγχει την εκτέλεση του κάθε τμήματος.

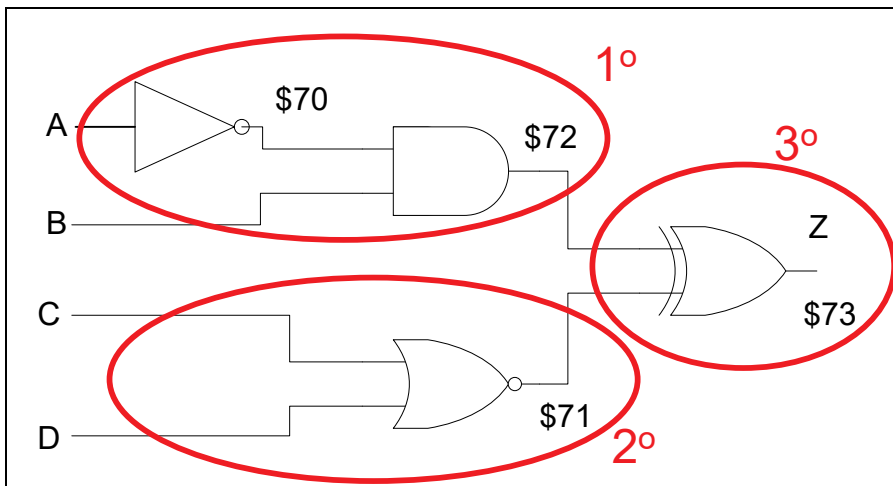
Έτσι λοιπόν και με βάση το παρακάτω σχήμα σκοπός του παραδείγματος είναι να αναπτυχθεί πρόγραμμα το οποίο να εκτελεί τα εξής.

Όσο δεν έχει πατηθεί το δεξί πλήκτρο/διακόπτης το πρόγραμμα δεν κάνει τίποτα.

Με το πρώτο πάτημα του δεξί πλήκτρου να εξομοιώνεται το 1^ο τμήμα του κυκλώματος ενώ το αποτέλεσμα να αποθηκεύεται στη διεύθυνση \$72 και να εμφανίζεται σε LEDs.

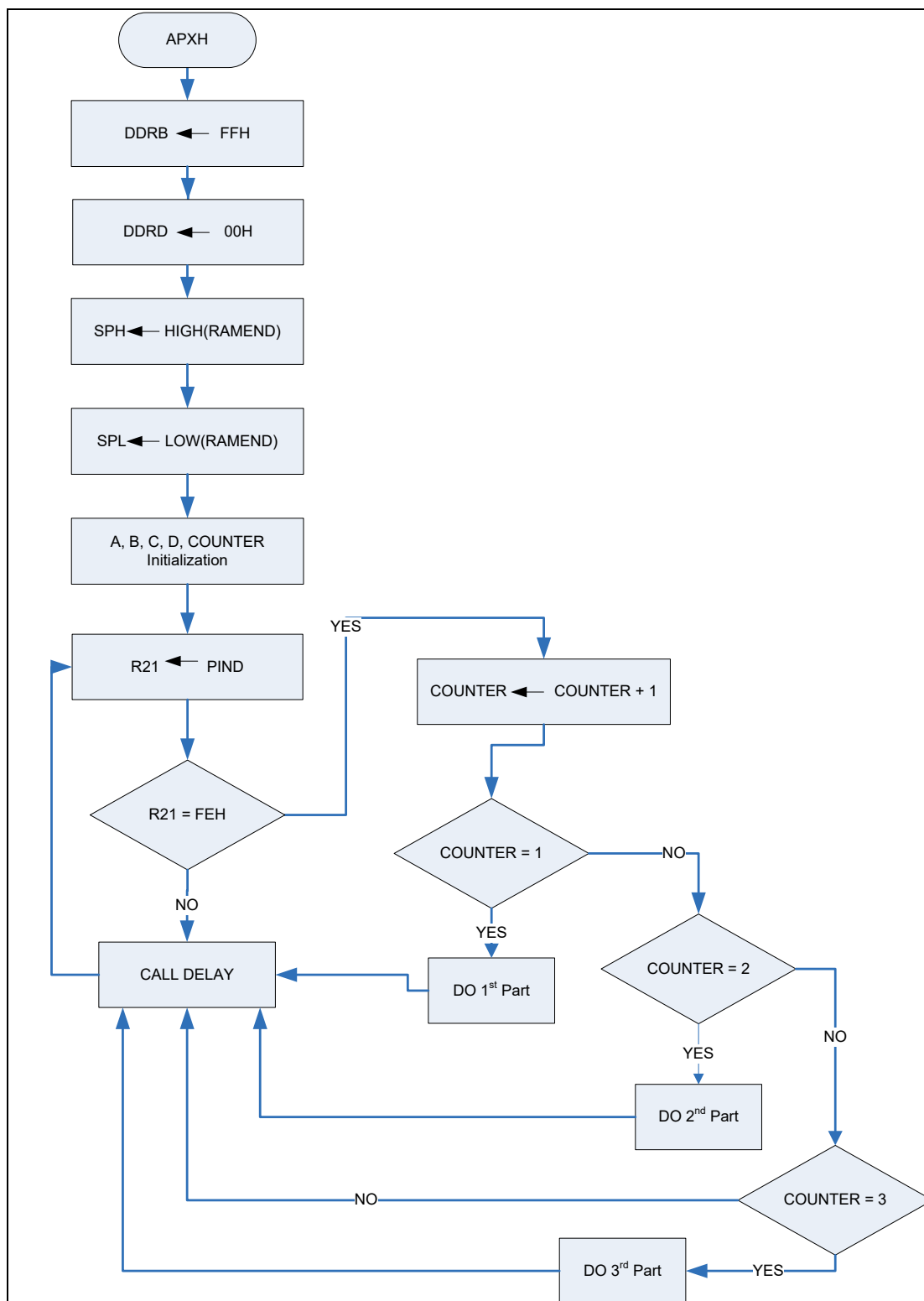
Με το δεύτερο πάτημα του δεξί πλήκτρου να εξομοιώνεται το 2^ο τμήμα του κυκλώματος ενώ το αποτέλεσμα να αποθηκεύεται στη διεύθυνση \$71 και να εμφανίζεται σε LEDs.

Με το τρίτο πάτημα του δεξί πλήκτρου να εξομοιώνεται το 3^ο τμήμα του κυκλώματος ενώ το αποτέλεσμα να αποθηκεύεται στη διεύθυνση \$73 και να εμφανίζεται σε LEDs.



Σε κάθε περίπτωση μετά το πάτημα ενός πλήκτρου να ακολουθεί μια καθυστέρηση που θα δίνει αρκετό χρόνο στο πλήκτρο/διακόπτη να γυρίσει σε κατάσταση ηρεμίας.

Το αντίστοιχο διάγραμμα ροής φαίνεται στο ακόλουθο διάγραμμα.



Τα μπλοκ “DO 1st Part”, “DO 2nd Part”, “DO 3rd Part” αποτελούνται από απλές εντολές λογικών πράξεων τις οποίες έχουμε αναλύσει και χρησιμοποιήσει σε προηγούμενη άσκηση.

Ακολουθεί μια πιθανή υλοποίησης του συγκεκριμένου αλγορίθμου με τα αντίστοιχα επεξηγηματικά σχόλια.

Σημαντικές Παρατηρήσεις:

Για να κατανοήσουμε τον κώδικα που ακολουθεί αλλά και τις ενδείξεις που έχουμε στα LEDs πρέπει να έχουμε πάντα υπόψη ότι το LED και οι διακόπτες στο STK 500 kit ακολουθούν ACTIVE LOW λογική. Δηλαδή για να φωτοβολήσει ένα LED στο STK500 στο αντίστοιχο PIN εξόδου πρέπει να στείλουμε 0 ενώ αν στείλουμε 1 θα παραμείνει σβηστό. Αντίστοιχα ένας πατημένος διακόπτης θα εμφανίσει στο PIN εισόδου 0 ενώ όσο ένας διακόπτης δεν πατιέται το αντίστοιχο PIN εισόδου θα εμφανίζει 1.

Η επιθυμητή καθυστέρηση δημιουργείται καλώντας την αντίστοιχη υπορουτίνα. Κατά συνέπεια θα πρέπει να ορίσουμε και αντίστοιχη, κατάλληλη μνήμη στοίβας όπως φαίνεται στο κώδικα που ακολουθεί.

```
.include "8515def.inc"
.ORG 5
; ορίζω την θύρα B που έχω συνδέσει στα
; LEDs ως έξοδο
LDI R24, 255
OUT DDRB, R24
; ορίζω την θύρα D που έχω συνδέσει στους
; διακόπτες ως είσοδο
LDI R24, 00
OUT DDRD, R24
; ορίζω την στοίβα μου για να μπορώ να καλέσω
; υπορουτίνες
LDI R24, HIGH(RAMEND)
OUT SPH, R24
LDI R24, LOW(RAMEND)
OUT SPL, R24
; για διευκόλυνση αντιστοιχίζω τους 4 καταχωρητές
; εισόδου του κυκλώματος σε πιο κατανοητά ονόματα
.DEF A=R16
.DEF B=R17
.DEF C=R18
.DEF D=R19
.DEF COUNTER=R20
; αρχικοποιώ τους καταχωρητές
LDI A,$3E
LDI B,$0A
LDI C,$1F
LDI D,$55
```

```

LDI COUNTER,0
AGAIN:
; διαβάζω τους διακόπτες
IN R21, PIND
CPI R21,$FE
; αν ΔΕΝ έχει πατηθεί ο επιθυμητός διακόπτης θα
; ξαναδιβάσω μετά από μια συγκεκριμένη καθυστέρηση
BRNE REPEAT_AFTER_DELAY
INC COUNTER
CPI COUNTER,1
BRNE CHECK_2ND_STEP
; αν ο COUNTER = 1 εκτελώ το πρώτο βήμα
COM A
AND A,B
STS $72,A
OUT PORTB,A
CHECK_2ND_STEP:
CPI COUNTER,2
BRNE CHECK_3RD_STEP
; αν ο COUNTER = 2 εκτελώ το δεύτερο βήμα
OR C,D
COM C
STS $71,C
OUT PORTB,C
CHECK_3RD_STEP:
CPI COUNTER,3
BRNE REPEAT_AFTER_DELAY
; αν ο COUNTER = 3 εκτελώ το τρίτο βήμα
EOR A,C
STS $73,A
OUT PORTB,A

REPEAT_AFTER_DELAY:
RCALL DELAY
RJMP AGAIN

; Αρχή υπορουτίνας καθυστέρησης
DELAY:
; Αρχή εξωτερικού βρόγχου

```

```

LDI R22, $FF
LOOP1:
; Αρχή εσωτερικού βρόγχου
LDI R23, $FF
LOOP2:
DEC R23
BRNE LOOP2
; Τέλος εσωτερικού βρόγχου
DEC R22
BRNE LOOP1
; Τέλος εξωτερικού βρόγχου
RET

```

Παρατήρηση/Ερώτηση: Δοκιμάστε να εκτελέσετε το πρόγραμμα χωρίς την παρεμβολή της καθυστέρησης μεταξύ των πατημάτων. Τι παρατηρείται όσο αφορά τη συμπεριφορά του συστήματος και που οφείλεται η συμπεριφορά αυτή ?

Δοκιμάστε να τρέξετε το παρακάτω πρόγραμμα στο KIT.

```

.include "8515def.inc"
.ORG 5
; ορίζω την θύρα B που έχω συνδέσει στα
; LEDs ως έξοδο
LDI R24, 255
OUT DDRB, R24
; ορίζω την θύρα D που έχω συνδέσει στους
; διακόπτες ως είσοδο
LDI R24, 00
OUT DDRD, R24
; ορίζω την στοίβα μου για να μπορώ να καλέσω
; υπορουτίνες
LDI R24, HIGH(RAMEND)
OUT SPH, R24
LDI R24, LOW(RAMEND)
OUT SPL, R24
; για διευκόλυνση αντιστοιχίζω τους 4 καταχωρητές
; εισόδου του κυκλώματος σε πιο κατανοητά ονόματα

```

```

.DEF A=R16
.DEF B=R17
.DEF C=R18
.DEF D=R19
.DEF COUNTER=R20
; αρχικοποιώ τους καταχωρητές
LDI A,$3E
LDI B,$0A
LDI C,$1F
LDI D,$55
LDI COUNTER,0
AGAIN:
; διαβάζω τους διακόπτες
IN R21, PIND
CPI R21,$01
; αν ΔΕΝ έχει πατηθεί ο επιθυμητός διακόπτης θα
; ξαναδιβάσω μετά από μια συγκεκριμένη καθυστέρηση
BRNE REPEAT_AFTER_DELAY
INC COUNTER
CPI COUNTER,1
BRNE CHECK_2ND_STEP
; αν ο COUNTER = 1 εκτελώ το πρώτο βήμα
COM A
AND A,B
STS $72,A
OUT PORTB,A
CHECK_2ND_STEP:
CPI COUNTER,2
BRNE CHECK_3RD_STEP
; αν ο COUNTER = 2 εκτελώ το δεύτερο βήμα
OR C,D
COM C
STS $71,C
OUT PORTB,C
CHECK_3RD_STEP:
CPI COUNTER,3
BRNE REPEAT_AFTER_DELAY
; αν ο COUNTER = 3 εκτελώ το τρίτο βήμα
EOR A,C

```

STS \$73,A

OUT PORTB,A

REPEAT_AFTER_DELAY:

RCALL DELAY

RJMP AGAIN

; Αρχή υπορουτίνας καθυστέρησης

DELAY:

; Αρχή εξωτερικού βρόγχου

LDI R22, \$FF

LOOP1:

; Αρχή εσωτερικού βρόγχου

LDI R23, \$FF

LOOP2:

DEC R23

BRNE LOOP2

; Τέλος εσωτερικού βρόγχου

DEC R22

BRNE LOOP1

; Τέλος εξωτερικού βρόγχου

RET

ΑΣΚΗΣΗ 10 – Αλγόριθμοι Ταξινόμησης (Επαναληπτική Άσκηση)

Στην παρούσα άσκηση θα παρουσιάσουμε, αναλύσουμε και αναπτύξουμε έναν ευρέως χρησιμοποιούμενο αλγόριθμο ταξινόμησης. Όπως δηλώνει και το όνομα τους οι συγκεκριμένοι αλγόριθμοι στόχο έχουν να ταξινομήσουν τα δεδομένα μιας συγκεκριμένης περιοχής μνήμης σύμφωνα με ένα συγκεκριμένο κριτήριο.

Στην βιβλιογραφία κάποιος μπορεί να βρει μια μεγάλη ποικιλία ανάλογων αλγορίθμων οι οποίοι διαφοροποιούνται με βάση την πολυπλοκότητά τους και την απόδοση του, δηλαδή την ταχύτητα με την οποία επιτυγχάνουν την επιθυμητή ταξινόμηση.

Ο αλγόριθμος στο οποίον θα εστιάσουμε συναντάνται στην βιβλιογραφία ως Bubble Short χαρακτηρίζεται από μέτρια απόδοση (ειδικά αυξανόμενων των δεδομένων προς ταξινόμηση) και χαμηλή πολυπλοκότητα η οποία των καθιστά ιδανικό για τους εκπαιδευτικούς λόγους του εργαστηρίου καθώς είναι ένα ολοκληρωμένο πρόγραμμα που απαιτεί την εφαρμογή πολλών από των γνώσεων που ο φοιτητής έχει λάβει από τις προηγούμενες εργαστηριακές ασκήσεις.

Εν συντομία ο αλγόριθμος βασίζεται σε συνεχή εξέταση όλης της περιοχής υπό εξέταση όπου σε κάθε πέρασμα γίνονται διαδοχικοί έλεγχοι αν δύο όλων των γειτονικών δεδομένων της περιοχής υπό εξέταση και γίνεται αλλαγή θέσης των δεδομένων όταν ικανοποιείται το κριτήριο της ταξινόμησης. Η όλη διαδικασία ολοκληρώνεται όταν σε ένα πέρασμα δεν υπάρξει καμία περίπτωση αλλαγής δεδομένων.

Το ακόλουθο παράδειγμα θα καταστήσει τον αλγόριθμο απόλυτα σαφή.

Θεωρούμε ότι έχουμε μια περιοχή μνήμης στην οποία έχουν αποθηκευτεί διαδοχικά τα ακόλουθα δεδομένα “5, 1, 4, 2, 8”. Εφαρμόζοντας τον Bubble Short αλγόριθμο έτσι ώστε να ταξινομηθούν κατά αύξουσα φορά θα γίνουν οι ακόλουθες ενέργειες. Σε κάθε βήμα με **έντονη** γραφή επισημαίνονται τα δεδομένα που ελέγχονται.

Πρώτο πέρασμα ελέγχου της περιοχής μνήμης.

(**5** 1 4 2 8) → (1 **5** 4 2 8) : 1^η αλλαγή

(1 **5** 4 2 8) → (1 4 **5** 2 8) : 2^η αλλαγή

(1 4 **5** 2 8) → (1 4 2 **5** 8) : 3^η αλλαγή

(1 4 2 **5** 8) → (1 4 2 5 **8**) : δεν γίνεται αλλαγή καθώς τα δεδομένα είναι σωστά ταξινομημένα.

Εδώ τελείωσε το πρώτο πέρασμα ελέγχου τη περιοχής μνήμης υπό έλεγχο. Το ότι απαιτήθηκαν τρεις αλλαγές υποδηλώνει ότι πρέπει να γίνει και νέο πέρασμα.

Δεύτερο πέρασμα ελέγχου της περιοχής μνήμης.

(**1** 4 2 5 8) → (1 4 2 5 8) : Δεν χρειάζεται αλλαγή

(1 4 2 5 8) → (1 2 4 5 8) : 1^η αλλαγή

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

Εδώ τελείωσε το δεύτερο πέρασμα ελέγχου τη περιοχής μνήμης υπό έλεγχο. Το ότι απαιτήθηκε να γίνει αλλαγή δεδομένων (ανεξαρτήτως του ότι ήταν μόνο μια) υποδηλώνει ότι υπάρχει πιθανότητα να πρέπει να γίνει και άλλη αλλαγή άρα πρέπει να γίνει νέο πέρασμα ελέγχου της μνήμης.

Τρίτο πέρασμα ελέγχου της περιοχής μνήμης.

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

(1 2 4 5 8) → (1 2 4 5 8) : Δεν χρειάζεται αλλαγή

Εδώ τελείωσε το τρίτο πέρασμα ελέγχου τη περιοχής μνήμης υπό έλεγχο. Το ότι δεν απαιτήθηκε να γίνει αλλαγή δεδομένων σημαίνει ότι τα δεδομένα μας είναι ταξινομημένα σωστά και το πρόγραμμα μπορεί να τελειώσει.

Για περισσότερες πληροφορίες μπορείτε να διαβάσετε εδώ.

http://en.wikipedia.org/wiki/Bubble_sort

Παράδειγμα

Αναπτύξτε πρόγραμμα το οποίο να ταξινομεί κατά αύξουσα σειρά τα δεδομένα που βρίσκονται στις θέσεις μνήμης \$70-\$74 εφαρμόζοντας τον Bubble Short αλγόριθμο.

```
.include "8515def.inc"
```

```
.DEF SWAP_COUNTER=R20
```

```
; ορίζω την θύρα B που έχω συνδέσει στα
```

```
; LEDs ως έξοδο
```

```
LDI R24, 255
```

```
OUT DDRB, R24
```

```
; ορίζω την στοίβα μου για να μπορώ να καλέσω
```

```
; υπορουτίνες
```

```
LDI R24, HIGH(RAMEND)
```

```
OUT SPH, R24
```

```
LDI R24, LOW(RAMEND)
```

```
OUT SPL, R24
```

```
NEXT_PASS:
```

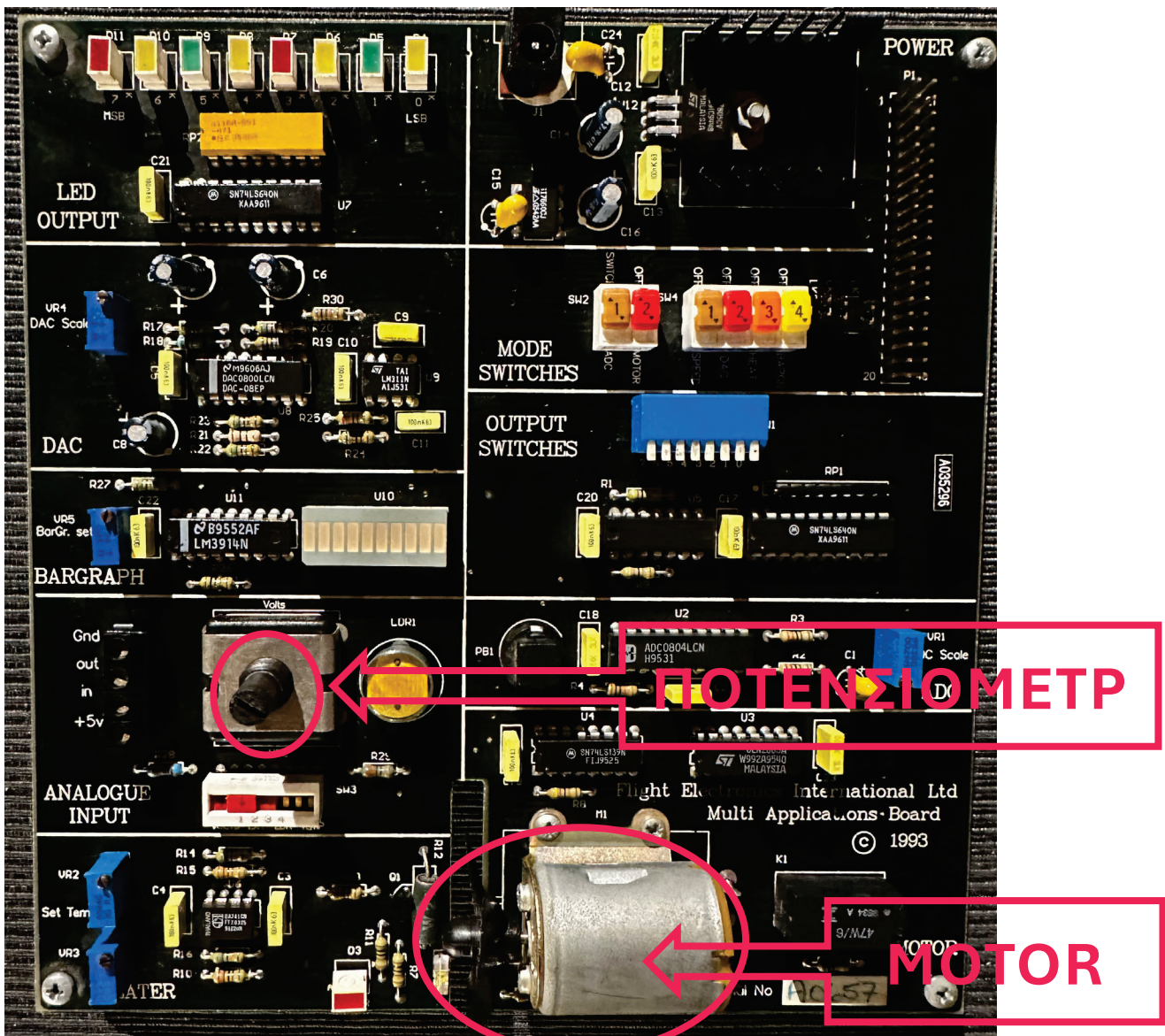
```
LDI SWAP_COUNTER, $00
```

```
LDI R29,$00
LDI R28,$70
NEXT_MEM_CHECK:
LD R16, Y
LDD R17,Y+1
CP R17, R16
BRGE NO_SWAP
ST Y, R17
STD Y+1, R16
INC SWAP_COUNTER
NO_SWAP:
INC R28
CPI R28, $74
BRNE NEXT_MEM_CHECK
CPI SWAP_COUNTER, $00
BRNE NEXT_PASS
OUT PORTB,SWAP_COUNTER
HALT:
RJMP HALT
```

ΑΣΚΗΣΗ 11 – ΕΛΕΓΧΟΣ ΣΤΗΝ ΤΑΧΥΤΗΤΑ ΚΑΙ ΣΤΗ ΦΟΡΑ ΕΝΟΣ ΚΙΝΗΤΗΡΑ ΜΕ ΓΡΑΜΜΙΚΟ ΠΟΤΕΝΣΙΟΜΕΤΡΟ

(KIT και APPLICATION BOARD)

Χρησιμοποιήστε κατάλληλα το γραμμικό ποτενσιόμετρο έτσι ώστε να επιτευχθεί έλεγχος στην ταχύτητα αλλά και στη φορά περιστροφής του κινητήρα. Στο κέντρο της πλήρους διαδρομής του ποτενσιομέτρου, ο κινητήρας θα σταματά να περιστρέφεται. Η φοράς περιστροφής θα καθορίζεται από το αν η τιμή που επιστρέφει ο ADC μετατροπέας βρίσκεται πριν ή μετά το παραπάνω σημείο ακινησίας του κινητήρα. Γνωρίζοντας ότι οι ADC μετατροπείς δεν παρέχουν πάντοτε σταθερή τιμή στην έξοδό τους, συνυπολογίστε την ύπαρξη μιας μικρής ανοχής στο σημείο ακινησίας του κινητήρα.

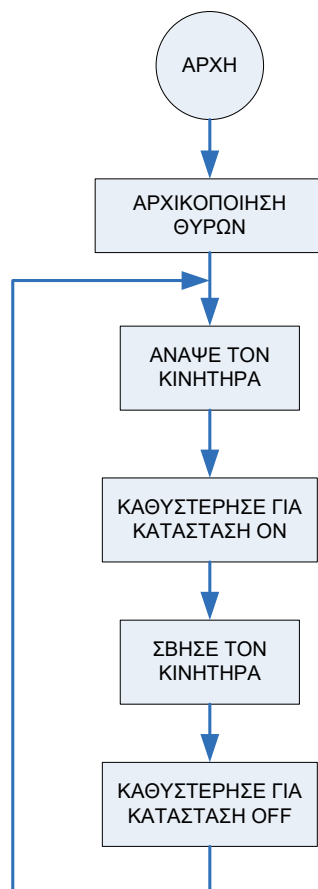


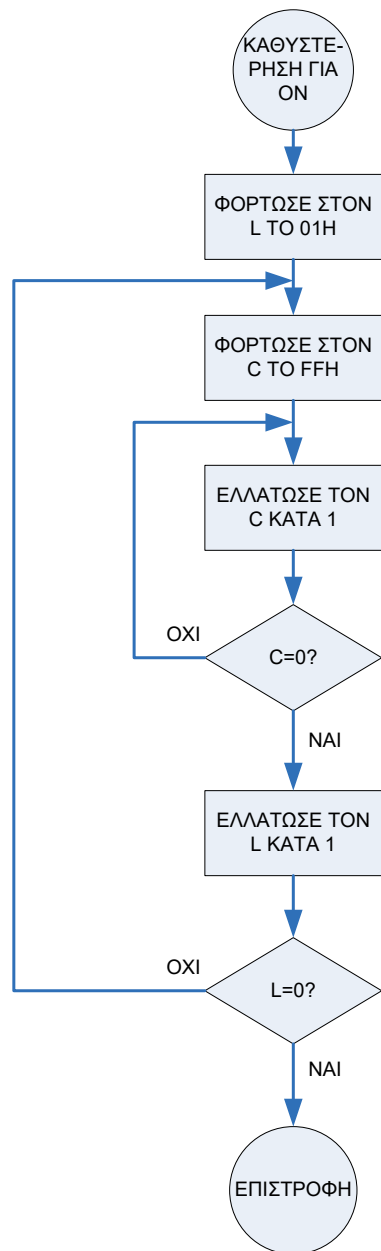
Η τάση τροφοδοσίας του κινητήρα προέρχεται απευθείας από την τάση τροφοδοσίας του Flight Application Board χωρίς κάποια σταθεροποίηση, και αυτό για να αποφευχθεί η δημιουργία θορύβου στα κυκλώματα που τροφοδοτούνται από σταθεροποιημένη τάση. Η αλλαγή της φοράς περιστροφής επιτυγχάνεται εφαρμόζοντας ανεστραμμένη τάση στον κινητήρα, με τη βοήθεια του ρελαί K1.

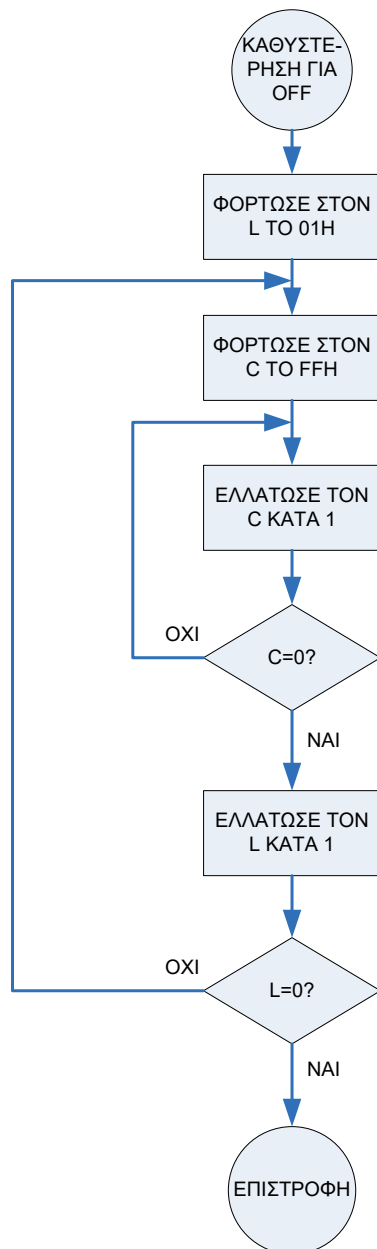
Η διαφορετική ταχύτητα περιστροφής επιτυγχάνεται, ανάβοντας και σβήνοντας τον κινητήρα πολύ γρήγορα, και αλλάζοντας την αναλογία χρόνου που ο κινητήρας είναι ανοιχτός και σβηστός.

Αρχικά γράφουμε ένα δοκιμαστικό πρόγραμμα το οποίο θα μας βοηθήσει να βρούμε τον μέγιστο χρόνο που μπορεί να παραμείνει σβηστός ο κινητήρας, εφόσον έχει περιστραφεί αρχικά για κάποιον συγκεκριμένο χρόνο όπως για παράδειγμα 10msec, έτσι ώστε η περιστροφή του να γίνεται χωρίς διακοπές. Αυτό τον μέγιστο χρόνο που παραμένει σβηστός ο κινητήρας θα τον χρησιμοποιήσουμε για την υλοποίηση του προβλήματος. Η περιστροφή του κινητήρα θα πρέπει να είναι αδιάκοπη και στις δύο φορές περιστροφής, έτσι ο χρόνος θα πρέπει να είναι τέτοιος ώστε να μπορεί να ανταποκρίνεται και το ρελαί.

Παρακάτω ακολουθεί το διάγραμμα ροής του παραπάνω δοκιμαστικού προγράμματος:







Στις δύο υπορουτίνες καθυστέρησης που φαίνονται παραπάνω, αλλάζουμε τις τιμές που θέτουμε στους καταχωρητές L και C, έτσι ώστε να επιτύχουμε το επιθυμητό αποτέλεσμα στην ταχύτητα περιστροφής του κινητήρα.

Το listing του παραπάνω δοκιμαστικού προγράμματος είναι:

LOOP:

```

LDI A,$40      ;Φόρτωσε στον A το 40H
OUT PORTC,A    ;Άναψε τον κινητήρα
CALL DELAY_ON  ;Καθυστέρηση για κατάσταση ON
LDI A,$00      ;Φόρτωσε στον A το 00H
OUT PORTC,A    ;Σβήσε τον κινητήρα
CALL DELAY_OFF ;Καθυστέρηση για κατάσταση OFF
  
```

`JMP LOOP` ;Επανάλαβε

`DELAY_ON:`

`LDI L,$0A`

`LOADC:`

`LDI C,$CC`

`DECRC:`

`DEC C`

`BRNE DECRC`

`DEC L`

`BRNE LOADC`

`RET`

`DELAY_OFF:`

`LDI L,$01`

`LOADC2:`

`LDI C,$CC`

`DECRC2:`

`DEC C`

`BRNE DECRC2`

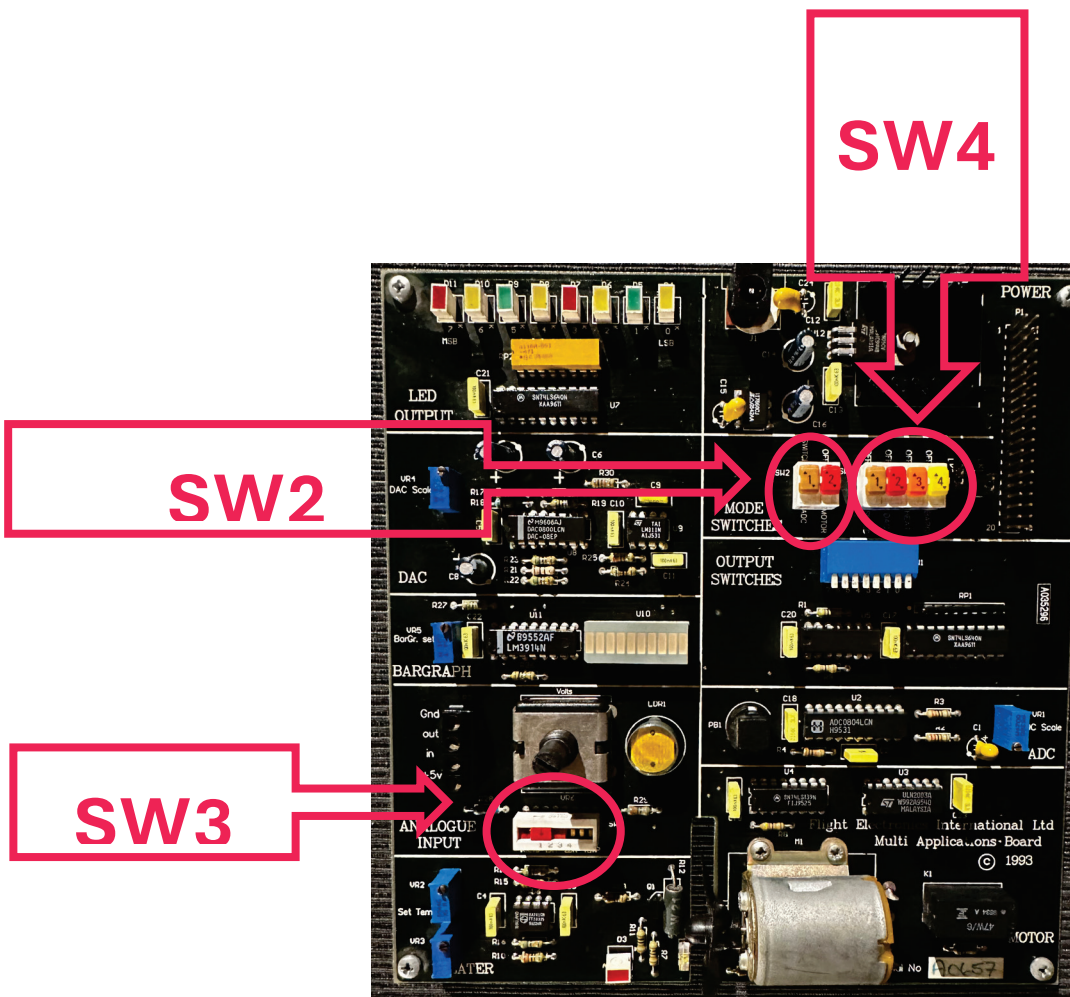
`DEC L`

`BRNE LOADC2`

`RET`

Κατάσταση Διακοπών:

Διακόπτης	SW2A	SW2B	SW3	SW4A	SW4B	SW4C	SW4D
Θέση	ADC	MOTOR	VOLTS	OFF	OFF	OFF	OFF



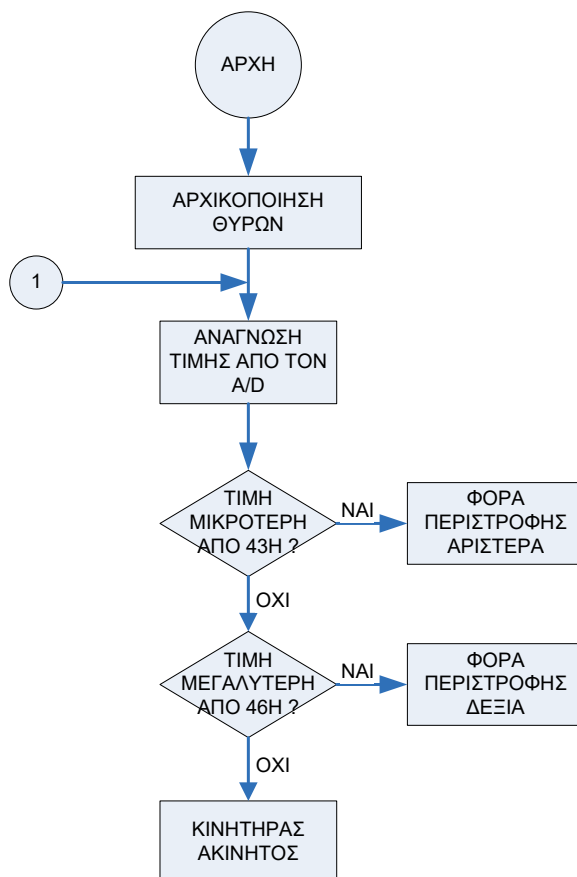
Μετά από αρκετές δοκιμές, και αλλάζοντας συνεχώς και στις δύο υπορουτίνες καθυστέρησης, τις τιμές των καταχωρητών L και C, βρέθηκε πως ο χρόνος που πρέπει να παραμείνει ο κινητήρας αναμένος είναι 100msec και ο μέγιστος χρόνος που μπορεί να παραμείνει σβήστός είναι 45msec. Οι χρόνοι αυτοί είναι τέτοιοι ώστε ο κινητήρας να περιστρέφεται αργά, χωρίς όμως να διακόπτεται η περιστροφή του.

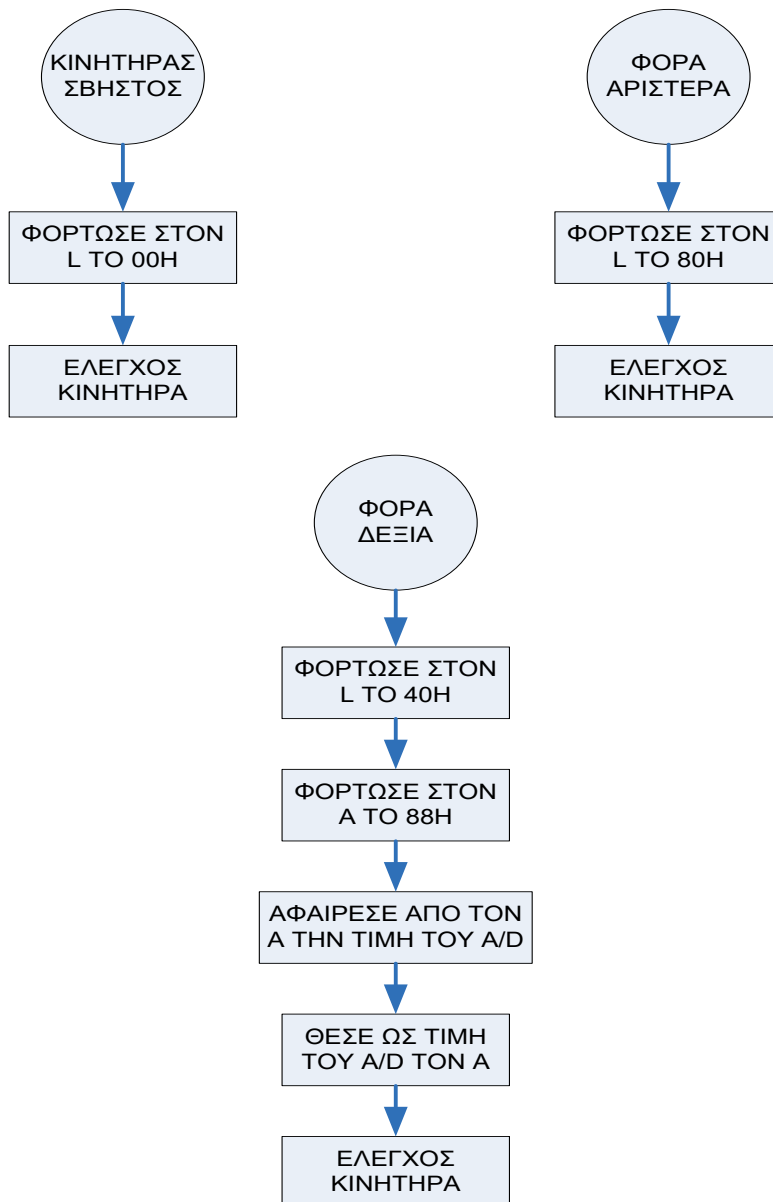
Έτσι λοιπόν για να λυθεί το αρχικό πρόβλημα, που ζητά τον έλεγχο των στροφών του κινητήρα με τη βοήθεια του γραμμικού ποτενσιόμετρου, θα ελέγχουμε το χρόνο που ο κινητήρας παραμένει σβηστός, ανάλογα με τη θέση του γραμμικού ποτενσιόμετρου. Λαμβάνοντας υπόψιν την ανοχή που πρέπει να δωθεί στο σημείο ακινησίας του κινητήρα, βλέπουμε πως οι εκμεταλλεύσιμες τιμές για τον έλεγχο των στροφών του κινητήρα είναι από 0 ως 42H, μιας και το σημείο ακινησίας ορίστηκε να είναι από την τιμή 43H ως και την τιμή 45H. Ο χρόνος λοιπόν των 45msec, που μπορεί ο κινητήρας να παραμείνει σβηστός, θα διαιρεθεί έτσι ώστε ο χρόνος αυτός να επιτυγχάνεται με τη μέγιστη εκμεταλλεύσιμη τιμή.

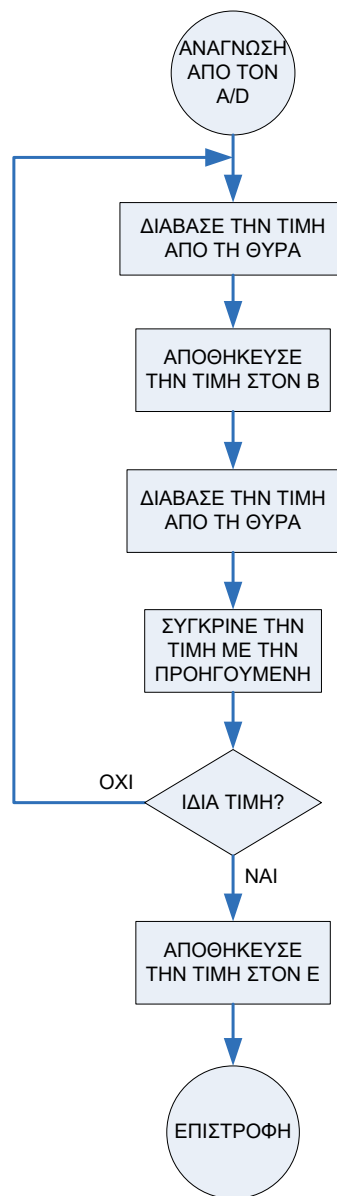
Ιδιαίτερη προσοχή πρέπει να δωθεί στην περίπτωση που το ποτενσιόμετρο βρίσκεται στην αρχή της διαδρομής του ή στο τέλος. Σ'αυτή την περίπτωση, μιας και ο χρήστης ζητά τον μέγιστο αριθμό στροφών, τότε η υπορουτίνα καθυστέρησης κατά την κατάσταση OFF, δε θα πρέπει να κληθεί διότι δε θα λειτουργήσει όπως είναι επιθυμητό και θα δημιουργήσει καθυστέρηση μεγαλύτερη των 45msec.

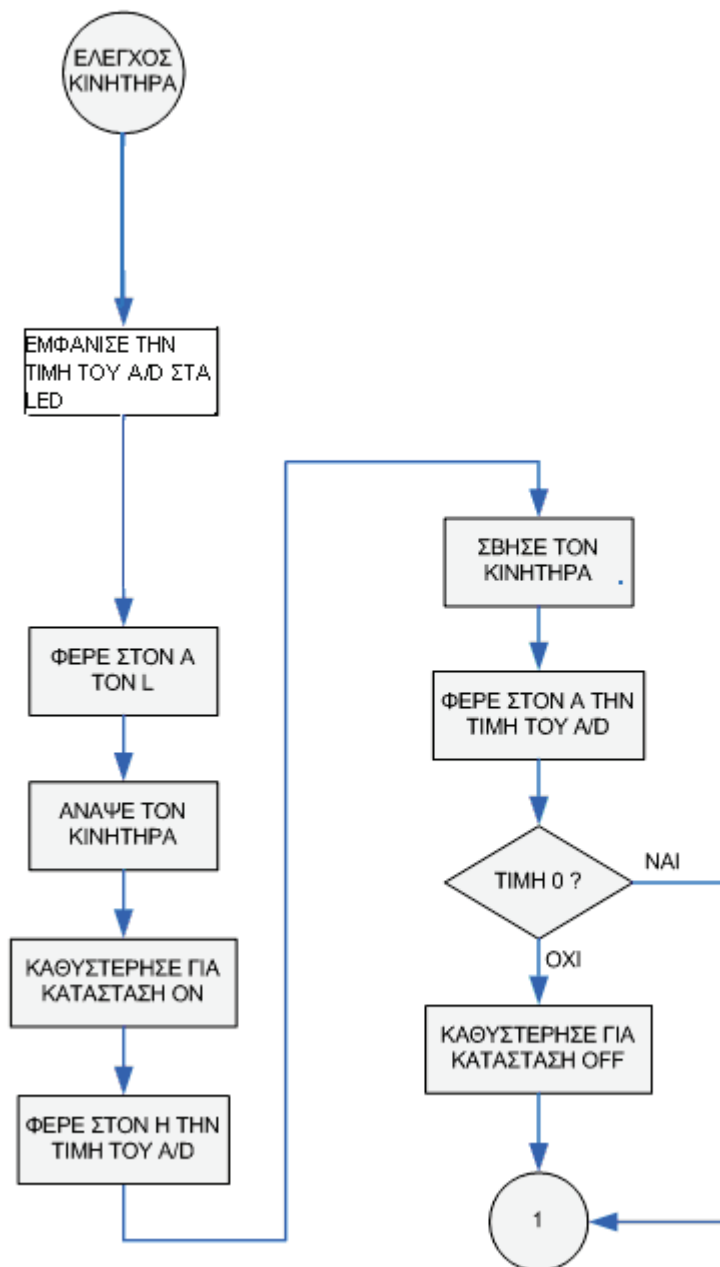
Για τη σωστή λειτουργία του προγράμματος ο διακόπτης SW4A θα πρέπει να βρίσκεται στη θέση 'ADC' και ο SW4B στη θέση 'MOTOR'.

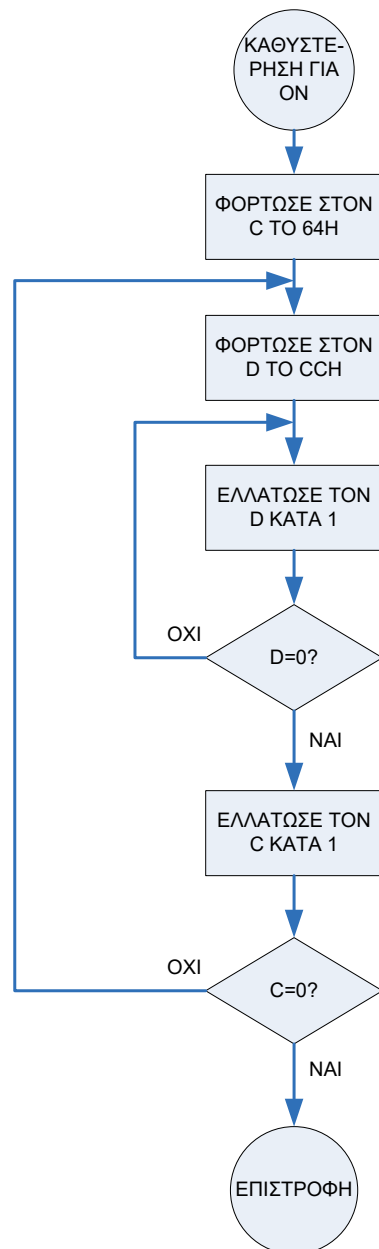
Αρχικά σχεδιάζουμε το διάγραμμα ροής του προγράμματος:

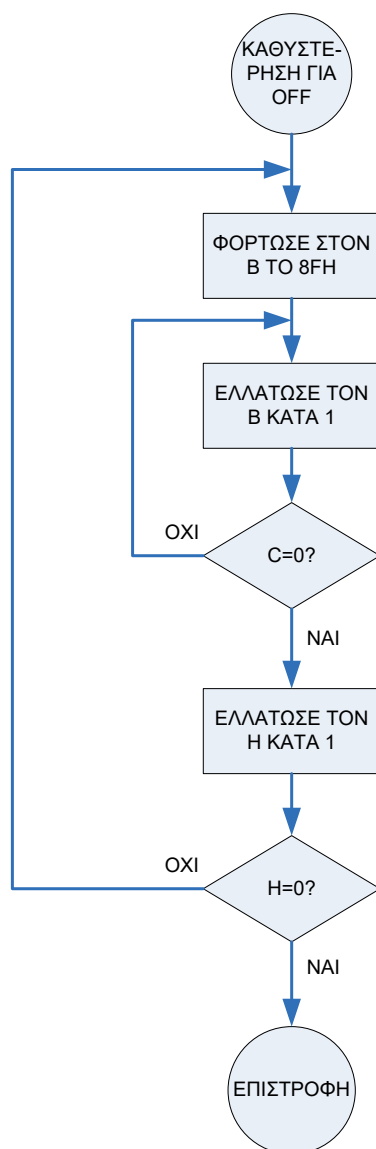












Ακολουθεί το listing του κώδικα:

```

.include "m8515def.inc"
sp_init:
LDI R16, LOW(RAMEND)
OUT SPL,R16
LDI R16, HIGH(RAMEND)
OUT SPH,R16
LDI R16, $FF
OUT DDRC, R16 ;ορίζω την θύρα C σαν έξοδο
CLR R12 ;μηδενίζω τον καταχωρητή R12
OUT DDRD,R12 ;ορίζω την θύρα D σαν είσοδο

```

OUT DDRB,R16 ;ορίζω την θύρα B που έχω συνδέσει στα LEDS ως έξοδο

.DEF A=R16

.DEF B=R17

.DEF C=R18

.DEF D=R19

.DEF E=R20

.DEF H=R21

.DEF L=R22

LOOP:

RCALL ADCC ;Διάβασε την τιμή από τον A/D

CPI A,\$43 ;Σύγκρινε την τιμή με το 43H

BRMI MOTOR_LEFT ;Τιμή μικρότερη από 43H. Φορά περιστροφής αριστερά

CPI A,\$46 ;Σύγκρινε την τιμή με το 46H

BRPL MOTOR_RIGHT ;Τιμή μεγαλύτερη από 46H. Φορά περιστροφής δεξιά.

RJMP MOTOR_OFF ;Κινητήρας σβηστός

MOTOR_OFF:

LDI L,\$00 ;Κινητήρας σβηστός

RJMP MOTOR_CONTROL ;Πήγαινε στον έλεγχο του κινητήρα

MOTOR_LEFT:

LDI L,\$80 ;Φορά περιστροφής αριστερά

RJMP MOTOR_CONTROL ;Πήγαινε στον έλεγχο του κινητήρα

MOTOR_RIGHT:

LDI L,\$40 ;Φορά περιστροφής δεξιά

LDI A,\$88 ;Φόρτωσε στον A το 88H

SUB A,E;Αφαίρεσε την τιμή του A/D

MOV E,A ;Φέρε στον E τον A

RJMP MOTOR_CONTROL;Πήγαινε στον έλεγχο του κινητήρα

MOTOR_CONTROL:

OUT PORTB,E ;Εμφάνισε την τιμή του A/D

MOV A,L ;Άναψε τον κινητήρα

OUT PORTC,A

RCALL DELAY_ON ;Καθυστέρηση για κατάσταση ON


```

LDI A,$00 ;Σβήσε τον κινητήρα
OUT PORTC,A
MOV H,E ;Φέρε στον H τον E
MOV A,E ;Φέρε στον A τον E
CPI A,$00 ;Σύγκρινε με το 00H
BREQ LOOP ;Τιμή 0. Πήγαινε στην αρχή
RCALL DELAY_OFF ;Καθυστερήση για κατάσταση OFF
RJMP LOOP ;Πήγαινε στην αρχή

```

ADCC:

```

IN A,PIND ;Διάβασε την τιμή
MOV B,A ;Αποθήκευσε την τιμή στον B
IN A,PIND ;Διάβασε την τιμή
CP A,B ;Σύγκρινε την τιμή με τον B
BRNE ADCC ;Διαφορετικές τιμές? Πήγαινε στην αρχή
MOV E,A ;Ίδιες τιμές. Αποθήκευσε την τιμή στον E
RET ;Επιστροφή

```

DELAY_ON:

```
LDI C,$64
```

LOADD:

```
LDI D,$0CC
```

DECRD:

```

DEC D
CPI D,0
BRNE DECRD
DEC C
CPI C,0
BRNE LOADD
RET

```

DELAY_OFF:

```
LDI B,$8F
```

DCRCB:

```

DEC B
CPI B,0
BRNE DCRCB
DEC H
CPI H,0

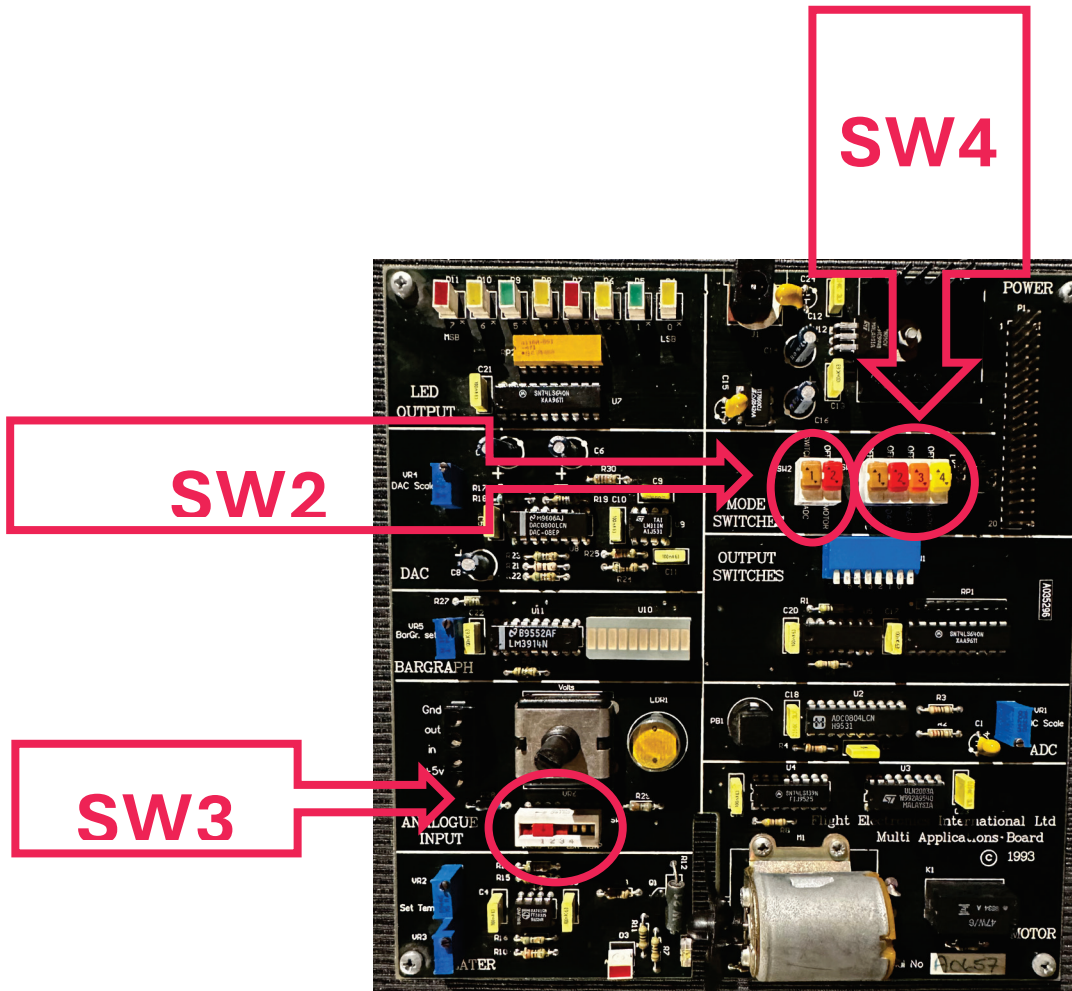
```

BRNE DELAY_OFF

RET

Κατάσταση Διακοπών:

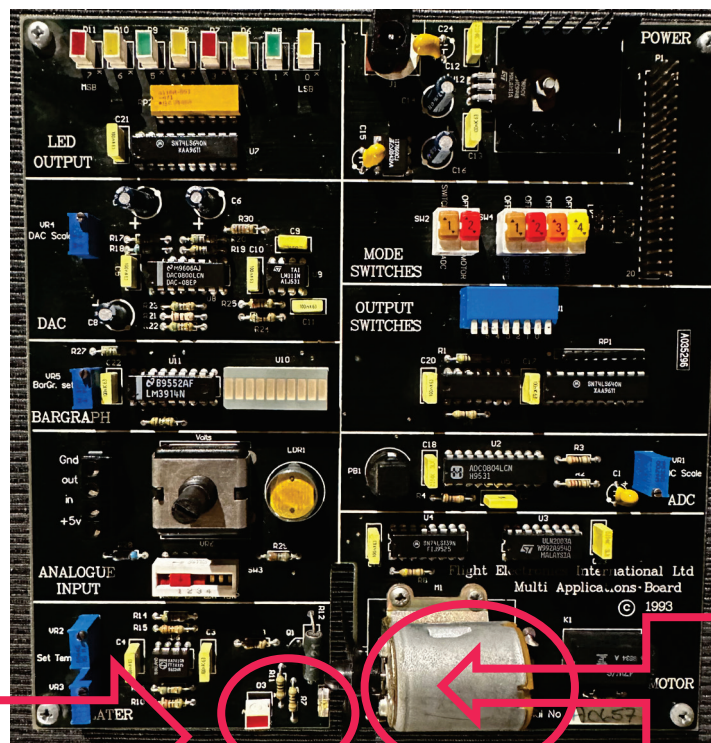
Διακόπτης	SW2A	SW2B	SW3	SW4A	SW4B	SW4C	SW4D
Θέση	ADC	MOTOR	VOLTS	OFF	OFF	OFF	OFF



ΑΣΚΗΣΗ 12 – ΔΙΑΤΗΡΗΣΗ ΣΤΑΘΕΡΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΕΝΤΟΣ ΟΡΙΩΝ ΣΤΗ ΘΕΡΜΟΑΝΤΙΣΤΑΣΗ ΜΕ ΤΗΝ ΒΟΗΘΕΙΑ ΤΟΥ ΚΙΝΗΤΗΡΑ

(KIT και APPLICATION BOARD)

Σκοπός αυτής της άσκησης είναι να διατηρηθεί σταθερή θερμοκρασία εντός ορίων στη θερμοαντίσταση με τη βοήθεια του κινητήρα που βρίσκεται στο Flight Application Board. Η θερμοαντίσταση θα βρίσκεται συνεχώς σε λειτουργία, και με τη βοήθεια του κινητήρα θα γίνει προσπάθεια να διατηρηθεί η θερμοκρασία της σταθερή εντός των ορίων που έχουμε ορίσει. Η θερμοκρασία μετράτε με τη βοήθεια του A/D μετατροπέα που δέχεται ως είσοδο την έξοδο κατάλληλου κυκλώματος που μετρά απ' ευθείας την θερμοκρασία της αντίστασης. Η θερμοαντίσταση αρχικά θερμαίνεται μέχρι ενός μέγιστου σημείου, και μετά μπαίνει σε λειτουργία ο κινητήρας μέχρι την επιθυμητή τιμή θερμοκρασίας. Αυτό θα γίνει ελέγχοντας τη λειτουργία του κινητήρα. Η επιθυμητή τιμή, είναι η τιμή που διαβάζεται από τον A/D και όχι θερμοκρασία σε °C. Η τρέχουσα τιμή που διαβάζεται από τον A/D να εμφανίζεται στα LED.



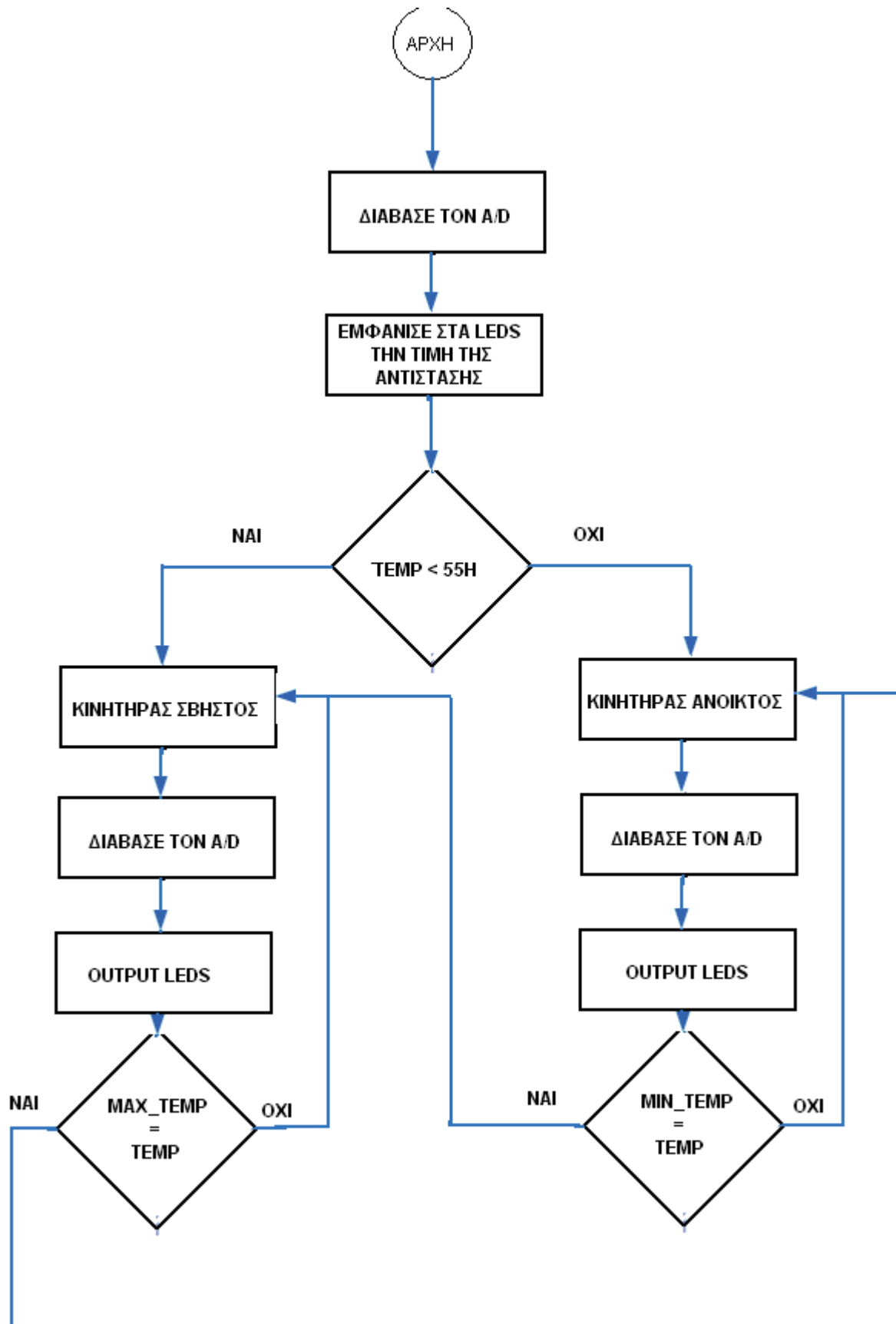
Θερμο
αντίσταση
και Led

MOTOR

Σημείωση: Μετά από αρκετές δοκιμές, παρατηρήθηκε πως η τιμή που επιστρέφει ο A/D, δεν μπορεί να κατέβει κάτω από την τιμή 4FH. Αποκλίσεις προς τα πάνω από την προαναφερθείσα τιμή, οφείλονται στη θερμοκρασία του περιβάλλοντος.

Ανάπτυξη

Σχεδιάζουμε το διάγραμμα ροής του προγράμματος:



Ακολουθεί το listing του κώδικα:

```
.include "8515def.inc"

sp_init:
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,HIGH(RAMEND)
OUT SPH,R16

LDI R16, $FF
OUT DDRC, R16
CLR R12
OUT DDRD,R12
OUT DDRB,R16

.DEF MAX_TEMP=R18
.DEF MIN_TEMP=R19
.DEF START_MOTOR=R20
.DEF STOP_MOTOR=R21
.DEF TEMP=R16

LDI MAX_TEMP,$55
LDI MIN_TEMP,$50
LDI START_MOTOR,$A0;Βάζει δε λειτουργία τον κινητήρα μαζί με την θερμοαντίσταση
LDI STOP_MOTOR,$20;Σταματάει τον κινητήρα και ανάβει την θερμοαντίσταση
INIT:
RCALL ADCREAD'Διάβασε την τιμή του A/D
RCALL OUTPUT_LEDS Εμφάνισε την τιμή στα LEDS
CP TEMP,MAX_TEMP Σύγκρινε την μέγιστη θερμοκρασία με την θερμοκρασία αυτή την στιγμή
BRLO STOP ;Εάν είναι μικρότερη από 55H σταματά τον κινητήρα
START:
RCALL MOTOR_START ;Ξεκίνα τον κινητήρα
RCALL ADCREAD ;Διάβασε την τιμή του A/D
RCALL OUTPUT_LEDS ;Εμφάνισε την τιμή στα LEDS
CP MIN_TEMP,TEMP ;Σύγκρινε την ελάχιστη θερμοκρασία με την θερμοκρασία αυτή την στιγμή
BRNE START ;Εάν δεν είναι ίδιες τότε κράτα τον κινητήρα σε λειτουργία
```

STOP:

RCALL MOTOR_STOP ;Σταμάτα τον κινητήρα

RCALL ADCREAD ;Διάβασε την τιμή του A/D

RCALL OUTPUT_LEDS ;Εμφάνισε την τιμή στα LEDS

CP MAX_TEMP,TEMP ;Σύγκρινε την μέγιστη θερμοκρασία με την θερμοκρασία αυτή την στιγμή

BRNE STOP ;Εάν δεν είναι ίδιες τότε κράτα τον κινητήρα σβηστό

RJMP START ;Εάν είναι διαφορετικές τότε βάλε σε λειτουργία τον κινητήρα

ADCREAD:

IN TEMP,PIND

RET

MOTOR_START:

OUT PORTC,START_MOTOR

RET

MOTOR_STOP:

OUT PORTC,STOP_MOTOR

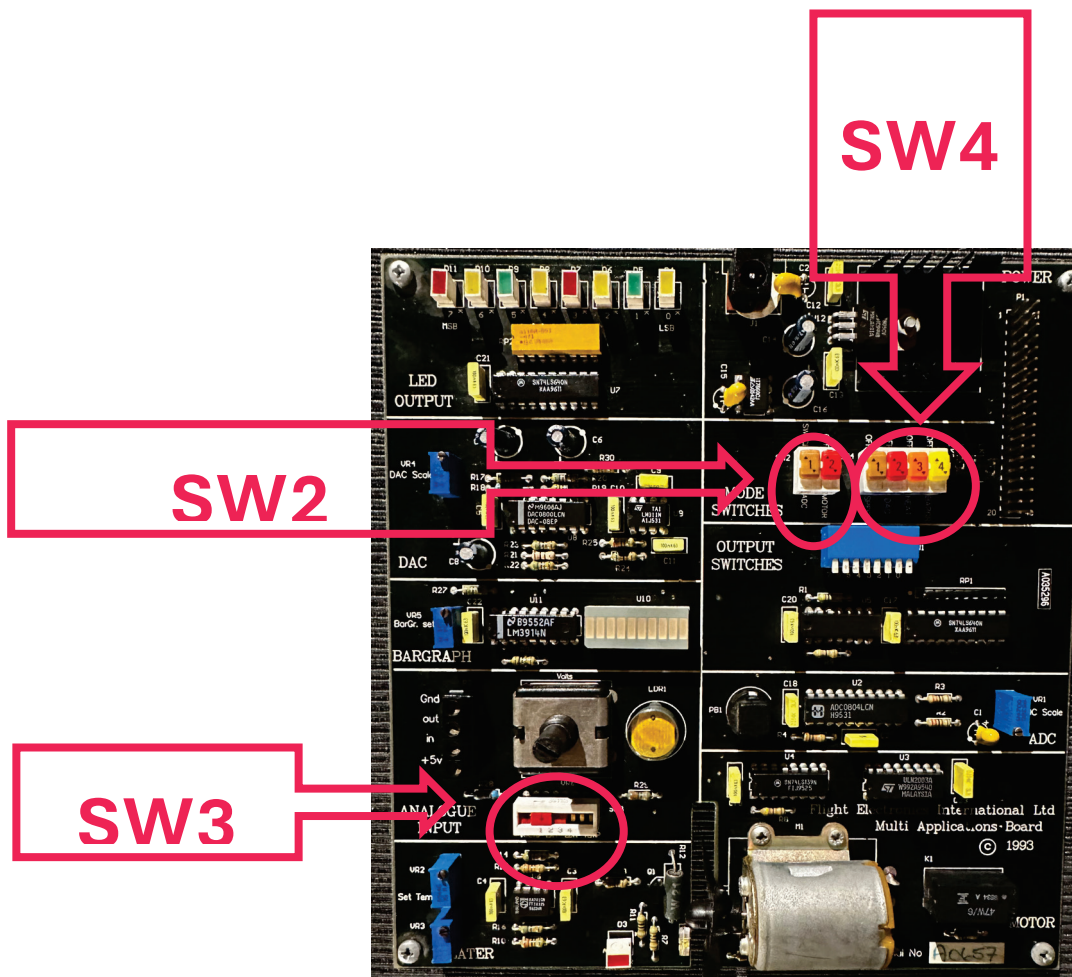
RET

OUTPUT_LEDS:

OUT PORTB,TEMP

RET

Διακόπτης	SW2A	SW2B	SW3	SW4A	SW4B	SW4C	SW4D
Θέση	ADC	MOTOR	TEMP	OFF	OFF	HEAT	OFF



ΑΣΚΗΣΗ 13 – ΕΛΕΓΧΟΣ ΤΩΝ LEDS ΜΕ ΦΩΤΟΔΙΟΔΟ

(KIT και APPLICATION BOARD)

Ο σκοπός της άσκησης είναι να ανάβουμε τα LEDS ανάλογα με το ποσοστό της φωτεινότητας που υπάρχει στο περιβάλλον. Δηλαδή όσο πιο πολύ ποσοστό φωτεινότητας έχουμε όσο και λιγότερα LEDS να ανάβουν. Αυτό θα γίνει με την βοήθεια της φωτοδιόδου και των LEDS. Η φωτοδιόδος όταν δεν δέχεται φως η τάση στην έξοδο της γίνεται μηδέν ενώ όταν δέχεται το μέγιστο ποσοστό φωτεινότητας η τάση στην έξοδο της γίνεται 2.55. Η έξοδος της φωτοδιόδου οδηγείται σε ένα A/D ο οποίος έχει τα εξής χαρακτηριστικά :

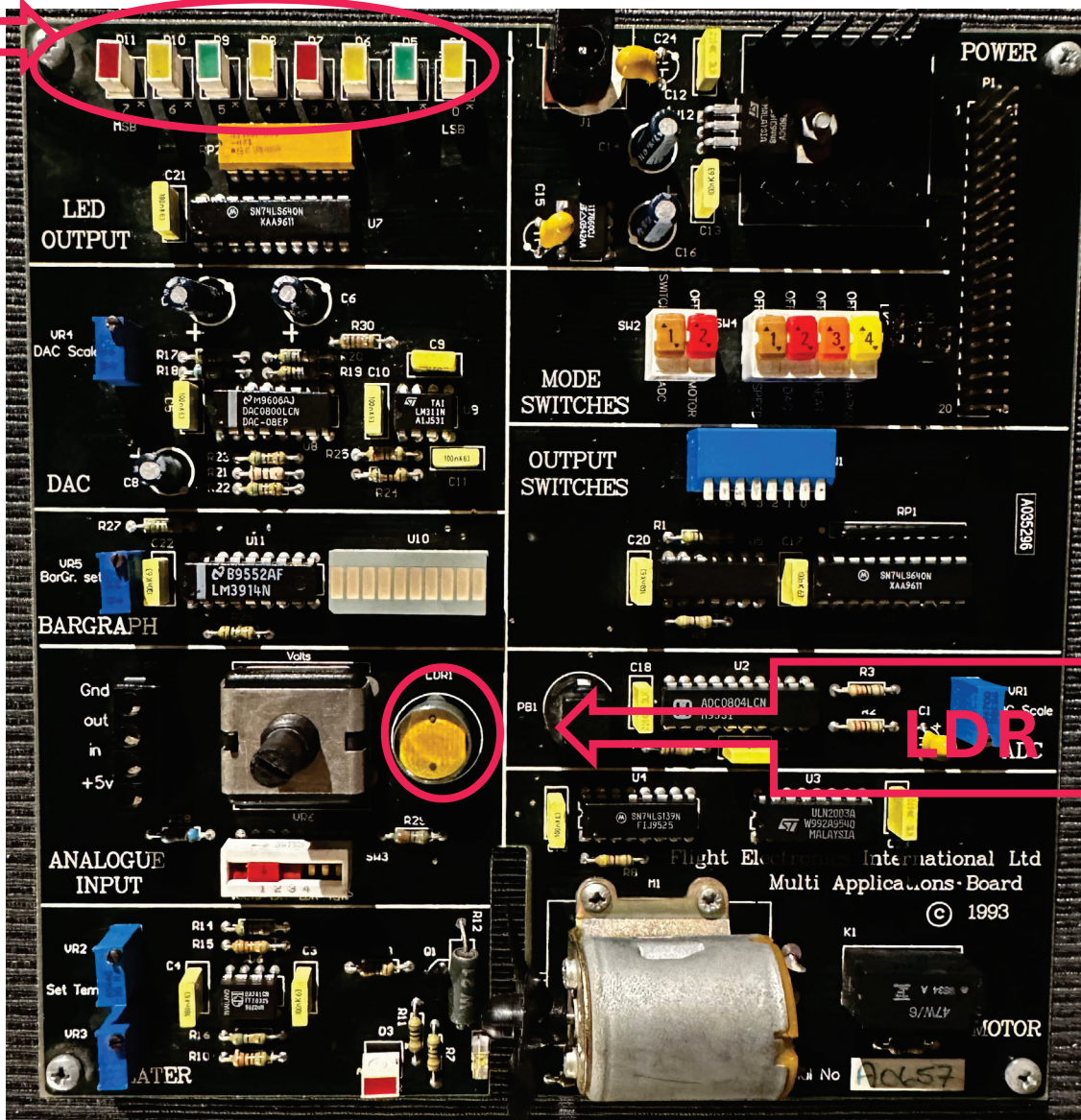
είσοδος 0.00 V για 00 hex έξοδος

είσοδος 2.50 V για 80 hex έξοδος

είσοδος 5.00 V για FF hex έξοδο.

Άρα η μέγιστη τιμή που μπορούμε να πάρουμε στην έξοδο του A/D για την μέγιστη τιμή της εξόδου της φωτοδιόδου είναι 80 hex. Λόγω όμως της μικρής ακριβείας της φωτοδιόδου και του A/D η ελάχιστη έξοδος που μπορούμε να πάρουμε από τον A/D είναι 10 hex και η μέγιστη είναι 39 hex. Χωρίζουμε το διάστημα 10 – 39 σε οκτώ ίσα διαστήματα όσα είναι και τα LEDs.

LED



Παρακάτω ακολουθεί το listing του κώδικα:

```
.include "8515def.inc"
sp_init:
LDI R16,LOW(RAMEND)
OUT SPL,R16
```



```

LDI R16,HIGH(RAMEND)
OUT SPH,R16
;INIT PORTS
LDI R16,00
OUT DDRD,R16 ;INPUT ;θέτω την PORTD σαν είσοδο
LDI R16,$FF
OUT DDRC,R16 ;OUTPUT ;θέτω την PORTB σαν έξοδο
LOOP:
LDI R17,$11
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL1 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
όλα τα leds
LDI R16,0B11111111
OUT PORTC,R16
BRLO LOOP ;πήγαινε στην αρχή
LABEL1:
LDI R17,$15 ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL2 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 7 leds
LDI R16,0B11111110
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL2:
LDI R17,$1A ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL3 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 6 leds
LDI R16,0B11111100
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL3:
LDI R17,$1F ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL4 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 5 leds
LDI R16,0B11111000

```

```

OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL4:
LDI R17,$24 ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL5 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 4 leds
LDI R16,0B11110000
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL5:
LDI R17,$29 ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL6 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 3 leds
LDI R16,0B11100000
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL6:
LDI R17,$2E ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL7 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
τα 2 leds
LDI R16,0B11000000
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL7:
LDI R17,$33 ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND ;διαβάζω την έξοδο του A/D
CPC R16,R17
BRGE LABEL8 ;εάν η τιμή του A/D είναι μεγαλύτερη πήγαινε στην επόμενη σύγκριση αλλιώς άναψε
το 1 led
LDI R16,0B10000000
OUT PORTC,R16
BRLO LOOP ;πήγαινε πάλι στην αρχή
LABEL8:
LDI R17,$39 ;φορτώνω στον R17 το επόμενο όριο
IN R16,PIND; διαβάζω την έξοδο του A/D

```

CPC R16,R17

BRLO LABEL9 ;εάν η τιμή του A/D είναι μικρότερη πήγαινε πάλι στην αρχή αλλιώς σβήσε όλα τα leds

LDI R16,0B00000000

OUT PORTC,R16

RJMP LABEL8 ;κράτησε όλα τα leds σβηστά όσο η τιμή του A/D είναι μεγαλύτερη ή ίση του 39 Η

LABEL9:

RJMP LOOP

Στο τέλος αντί για BRLO LOOP μπει η BRLO LABEL9 λόγω ότι υπάρχει ένας περιορισμός στο hardware που δεν επιτρέπει στις εντολές σύγκρισης να κάνουν άλμα πάνω από έναν ορισμένο αριθμό θέσεων μνήμης. Η εντολή RJMP δεν έχει αυτόν τον περιορισμό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Κ.Ζ. ΠΕΚΜΕΤΖΗ, ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ, ΤΟΜΟΣ 2: Μικροελεγκτές AVR, ΑΘΗΝΑ 2012.
- AVR STK500, AVR User guide, ATMEL.
- ATMEL, <http://www.atmel.com>
- Sean Ellis, Introduction to the STK500, 2001.

Παράρτημα 1 - ΡΕΠΕΡΤΟΡΙΟ ΕΝΤΟΛΩΝ ΤΟΥ AVR

Προέλευση/μετάφραση από το επίσημο εγχειρίδιο της ATMEL.

Π1.1 Συμβολισμοί περιγραφής εντολών

Καταχωρητής κατάστασης (SREG)

SREG: Καταχωρητής κατάστασης, C: Σημαία κρατουμένου, Z: Σημαία μηδενισμού, N: Σημαία αρνητικού αποτελέσματος

V: Σημαία για υπερχείλιση σε συμπλήρωμα ως προς 2 S: N xor V, για έλεγχο προσημασμένων

H: Σημαία ενδιάμεσου κρατουμένου

T: Μεταφερόμενο bit, χρησιμοποιείται από τις εντολές BLD and BST I: Καθολική Σημαία ενεργοποίησης/απενεργοποίησης Διακοπών

Καταχωρητές και Ορίσματα

Rd, Rr: Καταχωρητής Εργασίας Προορισμού και Προέλευσης

R: Result after instruction is executed

K: Σταθερό Δεδομένο, k: Σταθερή Διεύθυνση

b: Θέση Bit Καταχωρητή Εργασίας ή Καταχωρητή Θυρών I/O (3-bit) s: Θέση Bit στον Καταχωρητή κατάστασης (3-bit)

X, Y, Z: Καταχωρητές έμμεσης διευθυνσιοδότησης (X=R27:R26, Y=R29:R28 and Z=R31:R30)

A: Διεύθυνση θυρών I/O

q: Μετατόπιση σε έμμεση διευθυνσιοδότηση (6-bit)

Π1.2 Πλήρης περίληψη του συνόλου εντολών

Π1.2.1 ΕΝΤΟΛΕΣ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ

Mnemonics	Operands	Description	Operation	Flags	Clock
MOV	Rd, Rr	Move Between Registers	$Rd \leftarrow Rr$	None	1
MOVW	Rd, Rr	Copy Register Word	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	$Rd \leftarrow (X), X \leftarrow X+1$	None	2
LD	Rd, -X	Load Indirect and Pre-Dec.	$X \leftarrow X-1, Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y+1$	None	2
LD	Rd, -Y	Load Indirect and Pre-Dec.	$Y \leftarrow Y-1, Rd \leftarrow (Y)$	None	2
LDD	Rd, Y+q	Load Indirect + Displacement	$Rd \leftarrow (Y+q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	2

LD	Rd, - Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect + Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STD	Y+q,Rr	Store Indirect + Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
ST	- Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
STD	Z+q,Rr	Store Indirect + Displacement	$(Z + q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct to SRAM	$(k) \leftarrow Rr$	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
LPM	Rd, Z	Load Program Memory	$Rd \leftarrow (Z)$	None	3
LPM	Rd, Z+	Load Program Memory	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	3
SPM		Store Program Memory	$(Z) \leftarrow R1:R0$	None	-
IN	Rd, P	In Port	$Rd \leftarrow P$	None	1
OUT	P, Rr	Out Port	$P \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2

Π1.2.2 ΕΝΤΟΛΕΣ ΑΡΙΘΜΗΤΙΚΩΝ ΚΑΙ ΛΟΓΙΚΩΝ ΠΡΑΞΕΩΝ

Mnemonics	Operands	Description	Operation	Flags	Clock
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Regs	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	RdI,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Regs	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Const from Reg	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract & C two Regs	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract & C Const from Reg	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	RdI,K	Subtract Imm from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \wedge Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Reg & Const	$Rd \leftarrow Rd \wedge K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Reg & Const	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF \oplus Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd (\$FF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \wedge Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow \$FF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2

Π1.2.3 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΔΙΑΚΛΑΔΩΣΗΣ

Εντολές άλματος και κλήσης ρουτινών

Mnemonics	Operands	Description	Operation	Flags	Clock
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3

CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{STACK}$	None	4
RETI		Interrupt Return	$PC \leftarrow \text{STACK}$	I	4

Εντολές παράκαμψης

Mnemonics	Operands	Description	Operation	Flags	Clock
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) *	None	1/ 2/ 3
SBRSC	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) *	None	1/ 2/ 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) *	None	1/ 2/ 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) *	None	1/ 2/ 3

* Σε όλες τις περιπτώσεις που ισχύει η συνθήκη φορτώνεται η νέα τιμή του μετρητή Προγράμματος: $PC \leftarrow PC + 2 \text{ or } 3$.

Εντολές σύγκρισης

Mnemonics	Operands	Description	Operation	Flags	Clock
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) $PC \leftarrow PC + 2 \text{ or } 3$	None	1/2/3
CP	Rd,Rr	Compare	$Rd < Rr$	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd < Rr + C$	Z, N,V,C,H	1
CPI	Rd,K	Compare Reg with Immediate	$Rd < K$	Z,N,V,C,H	1

* Μεταξύ Καταχωρητών Εργασίας ή και Καταχωρητή Εργασίας με σταθερά.

Εντολές διακλάδωσης

Mnemonics	Operands	Description	Operation	Flags	Clock
BRBS	s,k	Branch if Status Flag Set	if (SREG(s) = 1) *	None	1/2
BRBC	s,k	Branch if Status Flag Cleared	if (SREG(s) = 0) *	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then *	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then *	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then *	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then *	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then *	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then *	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then *	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then *	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N⊕V = 0) then *	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N⊕V = 1) then *	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then *	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then *	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then *	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then *	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then *	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then *	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then *	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then *	None	1/2

* Σε όλες τις περιπτώσεις που ισχύει η συνθήκη φορτώνεται η νέα τιμή του μετρητή Προγράμματος: $PC \leftarrow PC + k + 1$

Π1.3 Περιγραφή Εντολών

Π1.3.1 ΕΝΤΟΛΕΣ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ

MOV Rd, Rr

Λειτουργία: $Rd \leftarrow Rr$

Περιγραφή: Το περιεχόμενο του καταχωρητή Rr αντιγράφεται στον Rd. Ο Rr παραμένει αναλλοίωτος.

Ενημέρωση σημαιών: καμία

MOVW Rd, Rr

Λειτουργία: $Rd+1:Rd \leftarrow Rr+1:Rr$ Περιγραφή: Το περιεχόμενο του ζεύγους καταχωρητών Rr+1:Rr αντιγράφεται στο ζεύγος καταχωρητών Rd+1:Rd.

Ενημέρωση σημαιών: καμία

LDI Rd, K

Λειτουργία: $Rd \leftarrow K$

Περιγραφή: Άμεση φόρτωση μιας σταθεράς 8-bit στον καταχωρητή Rd. Ο καταχωρητής Rd είναι ένας από τους R16-R31.

Ενημέρωση σημαιών: καμία

LDS Rd, k

Λειτουργία: $Rd \leftarrow (k)$

Περιγραφή: Τα περιεχόμενα της θέσης μνήμης (k) φορτώνονται στον καταχωρητή Rd.

Περίοδοι ρολογιού: 2

Ενημέρωση σημαιών: καμία

LD Rd, X (επίσης LD Rd, Y ή Z)

Λειτουργία: $Rd \leftarrow (X)$

Περιγραφή: Έμμεση φόρτωση του καταχωρητή Rd με τα περιεχόμενα της θέσης μνήμης (X). Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27 και Y=R28:R29, Z= R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

LD Rd, X+ (επίσης LD Rd, Y+ ή Z+)

Λειτουργία: $Rd \leftarrow (X)$, $X \leftarrow X + 1$ Περιγραφή: Έμμεση φόρτωση του καταχωρητή Rd με τα περιεχόμενα της θέσης μνήμης (X). Αύξηση του καταχωρητή X κατά μία μονάδα. Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27 και Y=R28:R29, Z= R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

LD Rd, -X (επίσης LD Rd, -Y ή -Z)

Λειτουργία: $X \leftarrow X - 1$, $Rd \leftarrow (X)$

Περιγραφή: Μείωση του καταχωρητή X κατά μία μονάδα. Έμμεση φόρτωση του καταχωρητή Rd με τα περιεχόμενα της θέσης μνήμης (X). Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27 και $Y=R28:R29, Z= R30:R31$.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

LDD Rd, Y+q (επίσης LDD Rd, Z+q)

Λειτουργία: $Rd \leftarrow (Y+q)$

Περιγραφή: Έμμεση φόρτωση του καταχωρητή Rd με τα περιεχόμενα της θέσης μνήμης (Y+q). Ο καταχωρητής Y είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R28:R29 και $Z= R30:R31$.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

STS k,Rr

Λειτουργία: $(k) \leftarrow Rr$ Περιγραφή: Άμεση αποθήκευση καταχωρητή. Το περιεχόμενο του καταχωρητή Rr αποθηκεύεται στη θέση μνήμης (k).

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

ST X, Rr

Λειτουργία: $(X) \leftarrow Rr$ Περιγραφή: Έμμεση αποθήκευση καταχωρητή. Το περιεχόμενο του καταχωρητή Rr αποθηκεύεται στη θέση μνήμης (X). Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

ST X+, Rr (επίσης ST Rd, Y+ ή Z+)

Λειτουργία: $(X) \leftarrow Rr$, $X \leftarrow X+1$ Περιγραφή: Έμμεση αποθήκευση καταχωρητή. Το περιεχόμενο του καταχωρητή Rr αποθηκεύεται στη θέση μνήμης (X). Αύξηση του καταχωρητή X κατά μία μονάδα. Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

ST -X, Rr (επίσης ST Rd, -Y ή -Z)

Λειτουργία: $X \leftarrow X-1$, $(X) \leftarrow Rr$

Περιγραφή: Έμμεση αποθήκευση καταχωρητή. Μείωση του καταχωρητή X κατά μία μονάδα. Το περιεχόμενο του καταχωρητή Rr αποθηκεύεται στη θέση μνήμης (X). Ο καταχωρητής X είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R26:R27.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

STD Y+q, Rr (επίσης STD Rd, Z+q)

Λειτουργία: $(Y+q) \leftarrow Rr$

Περιγραφή: Έμμεση αποθήκευση καταχωρητή με χρήση μετατόπισης. Το περιεχόμενο του καταχωρητή Rr αποθηκεύεται στη θέση μνήμης (Y+q). Ο καταχωρητής Y είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R28:R29 και Z= R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

LPM

Λειτουργία: $R0 \leftarrow (Z)$

Περιγραφή: Φόρτωση των περιεχομένων μιας θέσης μνήμης. Το περιεχόμενο της θέσης μνήμης προγράμματος (Z) φορτώνεται στον καταχωρητή R0. Ο καταχωρητής Z είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 3

LPM Rd, Z

Λειτουργία: $Rd \leftarrow (Z)$

Περιγραφή: Φόρτωση των περιεχομένων μιας θέσης μνήμης. Το περιεχόμενο της θέσης μνήμης προγράμματος (Z) φορτώνεται στον καταχωρητή Rd. Ο καταχωρητής Z είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 3

LPM Rd, Z+

Λειτουργία: $Rd \leftarrow (Z), Z \leftarrow Z + 1$ Περιγραφή: Το περιεχόμενο της θέσης μνήμης προγράμματος (Z) φορτώνεται στον καταχωρητή Rd. Αύξηση του καταχωρητή Z κατά μία μονάδα. Ο καταχωρητής Z είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 3

SPM

Λειτουργία: $(Z) \leftarrow R1:R0$

Περιγραφή: (Store Program Memory) Έμμεση αποθήκευση ζεύγους καταχωρητών. Το περιεχόμενο των καταχωρητών R1:R0 αποθηκεύεται στη θέση μνήμης προγράμματος (Z). Ο καταχωρητής Z είναι ένας καταχωρητής δείκτη που αντιστοιχεί στο ζεύγος καταχωρητών εργασίας R30:R31.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: -

IN Rd, A

Λειτουργία: $Rd \leftarrow I/O(A)$

Περιγραφή: Εισαγωγή δεδομένων από καταχωρητή εισόδου-εξόδου (θύρες, χρονιστές κλπ) σε καταχωρητή Rd.

Ενημέρωση σημαιών: καμία

OUT A, Rr

Λειτουργία: $I/O(A) \leftarrow Rr$

Περιγραφή: Εξαγωγή δεδομένων από καταχωρητή Rr σε καταχωρητή εισόδου-εξόδου (θύρες, χρονιστές, καταχ/τες κλπ).

Ενημέρωση σημαιών: καμία

PUSH Rr

Λειτουργία: $STACK \leftarrow Rr$

Περιγραφή: Αποθήκευση περιεχομένων καταχωρητή Rr στη στοίβα. Μείωση του δείκτη στοίβας κατά μία μονάδα.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

POP Rd

Λειτουργία: $Rd \leftarrow STACK$

Περιγραφή: Αύξηση του δείκτη στοίβας κατά μία μονάδα. Αποθήκευση περιεχομένων της στοίβας στον καταχωρητή Rd.

Ενημέρωση σημαιών: καμία

Περίοδοι ρολογιού: 2

Π1.3.2 ΕΝΤΟΛΕΣ ΑΡΙΘΜΗΤΙΚΩΝ ΚΑΙ ΛΟΓΙΚΩΝ ΠΡΑΞΕΩΝ

ADD Rd, Rr

Λειτουργία: $Rd \leftarrow Rd + Rr$

Περιγραφή: Πρόσθεση 2 καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.

Ενημέρωση σημαιών: Z, C, N, S, V, H

ADC Rd, Rr

Λειτουργία: $Rd \leftarrow Rd + Rr + C$

Περιγραφή: Πρόσθεση 2 καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.

Ενημέρωση σημαιών: Z, C, N, S, V, H

ADIW Rd, K

Λειτουργία: $Rd+1:Rd \leftarrow Rd+1:Rd + K$ Περιγραφή: Πρόσθεση μιας τιμής K (0-63) σε ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος καταχωρητών

Ενημέρωση σημαιών: Z, C, N, S, V

Περίοδοι ρολογιού: 2

SUB Rd, Rr

Λειτουργία: $Rd \leftarrow Rd - Rr$

Περιγραφή: Αφαίρεση 2 καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd

Ενημέρωση σημαιών: Z, C, N, S, V, H

SUBI Rd, K

Λειτουργία: $Rd \leftarrow Rd - K$

Περιγραφή: Αφαίρεση μιας τιμής χωρίς χρήση κρατουμένου από έναν καταχωρητή και αποθήκευση στον καταχωρητή

Ενημέρωση σημαίων: Z, C, N, S, V, H

SUBCI Rd, K

Λειτουργία: $Rd \leftarrow Rd - K - C$

Περιγραφή: Αφαίρεση μιας τιμής με χρήση κρατουμένου από έναν καταχωρητή και αποθήκευση στον καταχωρητή

Ενημέρωση σημαίων: Z, C, N, S, V, H

SBC Rd, Rr

Λειτουργία: $Rd \leftarrow Rd - Rr - C$

Περιγραφή: Αφαίρεση 2 καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd

Ενημέρωση σημαίων: Z, C, N, S, V, H

SBIW Rd, K

Λειτουργία: $Rd+1:Rd \leftarrow Rd+1:Rd - K$

Περιγραφή: Αφαίρεση μιας τιμής από ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος καταχωρητών

Ενημέρωση σημαίων: Z, C, N, S, V

Περίοδοι ρολογιού: 2

AND Rd, Rr

Λειτουργία: Λογικό AND $Rd \leftarrow Rd \text{ AND } Rr$

Περιγραφή: Λογικό AND μεταξύ των περιεχομένων των καταχωρητών Rd, Rr

Παράδειγμα: AND R26,R8

ANDI Rd, K

Λειτουργία: Λογικό AND $Rd \leftarrow Rd \text{ AND } K$

Περιγραφή: Λογικό AND μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K .Ο καταχωρητής Rd είναι ένας από τους R16-R31. Σταθερά K είναι 8bits

Ενημέρωση σημαίων: Z, N, S,V

OR Rd, Rr

Λειτουργία: Λογικό OR $Rd \leftarrow Rd \text{ OR } Rr$

Περιγραφή: Λογικό OR μεταξύ των περιεχομένων των καταχωρητών Rd, Rr

Ενημέρωση σημαίων: Z, N, S,V

ORI Rd, K

Λειτουργία: $Rd \leftarrow Rd \text{ OR } K$

Περιγραφή: Λογικό OR μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K. Ο καταχωρητής Rd είναι ένας από τους R16-R31. Η σταθερά K είναι 8 bit

Ενημέρωση σημαιών: Z, N, S,V

EOR Rd, Rr

Λειτουργία: $Rd \leftarrow Rd \text{ XOR } Rr$

Περιγραφή: Λογικό XOR (αποκλειστικό 'ή') μεταξύ των περιεχομένων των καταχωρητών Rd, Rr.

Ενημέρωση σημαιών: Z, N, S,V

COM Rd

Λειτουργία: $Rd \leftarrow \$FF - Rd$

Περιγραφή: Αντιστροφή των περιεχομένων του καταχωρητή Rd με χρήση αριθμητικής συμπληρώματος 1.

Ενημέρωση σημαιών: Z, C, N, S,V

NEG Rd

Λειτουργία: $Rd \leftarrow \$00 - Rd$

Περιγραφή: Αντιστροφή των

περιεχομένων του καταχωρητή Rd με χρήση αριθμητικής συμπληρώματος 2.

Ενημέρωση σημαιών: Z,C, N, S,V, H

SBR Rd, K

Λειτουργία: $Rd \leftarrow Rd \text{ OR } K$

Περιγραφή: Τοποθέτηση λογικού '1' στα bits του καταχωρητή Rd εκτελώντας λογικό OR μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K. Ο καταχωρητής Rd είναι ένας από τους R16- R31.

Ενημέρωση σημαιών: Z, N, S,V

CBR Rd, K

Λειτουργία: $Rd \leftarrow Rd \text{ AND } (\$FFh - K)$ Περιγραφή: Τοποθέτηση λογικού '0' στα bits του καταχωρητή Rd εκτελώντας λογικό AND μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K. Ο καταχωρητής Rd είναι ένας από τους R16- R31.

Ενημέρωση σημαίων: Z, N, S,V

INC Rd

Λειτουργία: $Rd \leftarrow Rd + 1$

Περιγραφή: Αύξηση των περιεχομένων του καταχωρητή Rd κατά μία μονάδα.

Ενημέρωση σημαίων: Z, N, S,V

DEC Rd

Λειτουργία: $Rd \leftarrow Rd - 1$

Περιγραφή: Μείωση των περιεχομένων του καταχωρητή Rd κατά μία μονάδα.

Ενημέρωση σημαίων: Z, N, S,V

TST Rd

Λειτουργία: $Rd \leftarrow Rd \wedge Rd$

Περιγραφή: Έλεγχος μηδενικής ή αρνητικής τιμής των περιεχομένων του καταχωρητή Rd.

Ενημέρωση σημαίων: Z, N,S,V Παράδειγμα: TST R3

CLR Rd

Λειτουργία: $Rd \leftarrow \$00$

Περιγραφή: Μηδενισμός καταχωρητή Rd.

Ενημέρωση σημαίων: Z, N, S,V

SER Rd

Λειτουργία: $Rd \leftarrow \$FF$

Περιγραφή: Τοποθέτηση '1' σε όλα τα bits του καταχωρητή (τιμή \$FF).

Ενημέρωση σημαίων: Z, N, S,V

MUL Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$

Περιγραφή: Μη προσημασμένος πολλαπλασιασμός μεταξύ των περιεχομένων των καταχωρητών Rd, Rr. Το αποτέλεσμα αποθηκεύεται στο ζεύγος καταχωρητών R1, R0.

Ενημέρωση σημαίων: Z, C

Περίοδοι ρολογιού: 2

MULS Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$ Περιγραφή: Προσημασμένος πολλαπλασιασμός Rd, Rr . Το αποτέλεσμα στο ζεύγος καταχωρητών $R1, R0$.

Ενημέρωση σημαιών: Z, C

Περίοδοι ρολογιού: 2

MULSU Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$

Περιγραφή: Προσημασμένος πολλαπλασιασμός μεταξύ των καταχωρητών Rd (προσημασμένου), Rr (μη-προσημασμένου). Το αποτέλεσμα στο ζεύγος καταχωρητών $R1, R0$.

Ενημέρωση σημαιών: Z, C

Περίοδοι ρολογιού: 2

FMUL Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$

Περιγραφή: Μη προσημασμένος κλασματικός πολλαπλασιασμός μεταξύ των περιεχομένων των καταχωρητών Rd (μορφοποίηση 1.7) και Rr (μορφοποίηση 1.7). Το αποτέλεσμα αποθηκεύεται στ ζεύγος καταχωρητών $R1, R0$ (μορφοποίηση 1.15).

Ενημέρωση σημαιών: Z, C

Περίοδοι ρολογιού: 2

FMULS Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$

Περιγραφή: Προσημασμένος κλασματικός πολλαπλασιασμός μεταξύ των περιεχομένων των καταχωρητών Rd (μορφοποίηση 1.7) και Rr (μορφοποίηση 1.7). Το αποτέλεσμα αποθηκεύεται στο ζεύγος καταχωρητών $R1, R0$ (μορφοποίηση 1.15).

Ενημέρωση σημαιών: Z, C

Περίοδοι ρολογιού: 2

FMULSU Rd, Rr

Λειτουργία: $R1, R0 \leftarrow Rd \times Rr$

Περιγραφή: Προσημασμένος κλασματικός πολλαπλασιασμός μεταξύ των περιεχομένων των καταχωρητών Rd (προσημασμένος με μορφοποίηση 1.7) και Rr (μη-προσημασμένος με μορφοποίηση 1.7). Το αποτέλεσμα αποθηκεύεται στο ζεύγος καταχωρητών $R1, R0$ (μορφοποίηση 1.15).

Ενημέρωση σημαιών: Z, C

Περίοδοι ρολογιού: 2

Π1.3.3 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΚΑΙ ΔΙΑΚΛΑΔΩΣΗΣ

RJMP k

Λειτουργία: $PC \leftarrow PC + k + 1$

Περιγραφή: Σχετικό άλμα σε μια διεύθυνση της μνήμης προγράμματος μεταξύ $PC - 2K + 1$ και $PC + 2K$.

Περίοδοι ρολογιού: 2

IJMP

Λειτουργία: $PC \leftarrow Z(15:0)$

Περιγραφή: Έμμεση εκτέλεση άλματος σε θέση της μνήμης προγράμματος η διεύθυνση της οποίας αντιστοιχεί στο περιεχόμενο του καταχωρητή δείκτη Z.

Περίοδοι ρολογιού: 2

JMP k

Λειτουργία: $PC \leftarrow k$

Περιγραφή: Απλό άλμα σε μια θέση της μνήμης προγράμματος.

Περίοδοι ρολογιού: 3

RCALL k

Λειτουργία: $PC \leftarrow PC + k + 1$

Περιγραφή: Σχετική κλήση ρουτίνας σε μια διεύθυνση μεταξύ $PC - 2K + 1$ και $PC + 2K$. Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα.

Περίοδοι ρολογιού: 3/4

ICALL k

Λειτουργία: $PC \leftarrow Z(15:0)$ Περιγραφή: Έμμεση κλήση ρουτίνας η διεύθυνση της οποίας αντιστοιχεί στο περιεχόμενο του καταχωρητή δείκτη Z.

Περίοδοι ρολογιού: 3/4

CALL k

Λειτουργία: $PC \leftarrow k$

Περιγραφή: Κλήση υπορουτίνας εντός της μνήμης προγράμματος. Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα.

Περίοδοι ρολογιού: 4/5

RET

Λειτουργία: $PC \leftarrow \text{Stack}$

Περιγραφή: Επιστροφή από υπορουτίνα. Η διεύθυνση επιστροφής φορτώνεται από τη στοίβα.

Περίοδοι ρολογιού: 4/5

RETI

Λειτουργία: $PC \leftarrow \text{Stack}$ Περιγραφή: Επιστροφή από ρουτίνα εξυπηρέτησης διακοπής. Η διεύθυνση επιστροφής φορτώνεται από τη στοίβα και η σημαία ολικών διακοπών τίθεται.

Ενημέρωση σημαιών: I

Περίοδοι ρολογιού: 4/5

CPSE Rd, Rr

Λειτουργία: Αν $Rd = Rr$ τότε $PC \leftarrow PC + 2$ (ή 3) αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σύγκριση μεταξύ των καταχωρητών Rd, Rr . Σε ισότητα παρακάμπτεται η επόμενη εντολή.

Περίοδοι ρολογιού: 1/2/3

CP Rd, Rr

Λειτουργία: $Rd - Rr$

Περιγραφή: Σύγκριση με υπολογισμό της διαφοράς μεταξύ των καταχωρητών Rd, Rr .

Ενημέρωση σημαιών: Z, C, N, S, V, H

CPC Rd, Rr

Λειτουργία: $Rd - Rr - C$

Περιγραφή: Σύγκριση με υπολογισμό της διαφοράς και χρήση κρατουμένου μεταξύ των καταχωρητών Rd, Rr .

Ενημέρωση σημαιών: Z, C, N, S, V, H

CPI Rd, K

Λειτουργία: $Rd - K$

Περιγραφή: Σύγκριση με υπολογισμό της διαφοράς και χρήση κρατουμένου μεταξύ του καταχωρητή Rd και μιας σταθεράς. Ο καταχωρητής Rd είναι ένας από τους R16-R31

Ενημέρωση σημαιών: Z, C, N, S, V, H

SBRC Rr, b

Λειτουργία: Αν $Rr(b) = 0$ τότε $PC \leftarrow PC + 2$ (ή 3) αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Αν το bit (b) του καταχωρητή είναι 0 ($Rr(b) = 0$) τότε παρακάμπτεται η επόμενη εντολή.

Περίοδοι ρολογιού: 1/2/3

SBRS Rr, b

Λειτουργία: Αν $Rr(b) = 1$ τότε $PC \leftarrow PC + 2$ (ή 3) αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Αν το bit (b) του καταχωρητή είναι 1 ($Rr(b) = 1$) τότε παρακάμπτεται η επόμενη εντολή.

Περίοδοι ρολογιού: 1/2/3

SBIC A, b

Λειτουργία: Αν $I/O(A,b) = 0$ τότε $PC \leftarrow PC + 2$ (or 3) αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Αν το bit (b) του καταχωρητή εισόδου-εξόδου είναι 0 ($A(b) = 0$) τότε παρακάμπτεται η επόμενη εντολή.

Περίοδοι ρολογιού: 1/2/3

SBIS A, b

Λειτουργία: Αν $I/O(A,b) = 1$ τότε $PC \leftarrow PC + 2$ (ή 3) αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Αν το bit (b) του καταχωρητή εισόδου-εξόδου είναι 1 ($A(b) = 1$) τότε παρακάμπτεται η επόμενη εντολή.

Περίοδοι ρολογιού: 1/2/3

BRBS s, k

Λειτουργία: Αν $SREG(s) = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Αν το bit s του καταχωρητή $SREG(s) = 1$ τότε εκτελείται άλμα στη διεύθυνση της μνήμης προγράμματος που βρίσκεται k θέσεις μετά σε σχέση με το μετρητή προγράμματος.

Περίοδοι ρολογιού: 1/2

BRBC s, k

Λειτουργία: Αν $SREG(s) = 0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Αν το bit s του καταχωρητή κατάστασης είναι 0 ($SREG(s) = 0$) τότε εκτελείται άλμα στη διεύθυνση της μνήμης προγράμματος που βρίσκεται k θέσεις μετά σε σχέση με το μετρητή προγράμματος.

Περίοδοι ρολογιού: 1/2

BREQ k

Λειτουργία: Αν $R_d = R_r$ ($Z = 1$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Σχετικό άλμα σε περίπτωση ισότητας. Όταν η σημαία μηδενός ισούται με 1 ($Z = 1$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 1, k.

Περίοδοι ρολογιού: 1/2

BRNE k

Λειτουργία: Αν $R_d \neq R_r$ ($Z = 0$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Σχετικό άλμα σε περίπτωση ανισότητας. Όταν η σημαία μηδενός ισούται με 0 ($Z = 0$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 1, k.

Περίοδοι ρολογιού: 1/2

BRCS k

Λειτουργία: Αν $C = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία κρατουμένου ισούται με 1 ($C = 1$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 0, k.

Περίοδοι ρολογιού: 1/2

BRCC k

Λειτουργία: Αν $C = 0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία κρατουμένου ισούται με 0 ($C = 0$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 0, k.

Περίοδοι ρολογιού: 1/2

BRSH k

Λειτουργία: Αν $R_d \geq R_r$ ($C = 0$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση δώσει μεγαλύτερο ή ίσο ($C = 0$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 0, k.

Περίοδοι ρολογιού: 1/2

BRLO k

Λειτουργία: Αν $R_d < R_r$ ($C = 1$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$ Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση δώσει μικρότερο ($C = 1$) τότε εκτελείται σχετική διακλάδωση.

Ισοδυναμεί στην εντολή BRBS 0, k.

Περίοδοι ρολογιού: 1/2

BRMI k

Λειτουργία: Αν $N = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση δώσει ($N = 1$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 2, k.

Περίοδοι ρολογιού: 1/2

BRPL k

Λειτουργία: Αν $N = 0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση δώσει ($N = 0$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 2, k.

Περίοδοι ρολογιού: 1/2

BRGE k

Λειτουργία: Αν $Rd \geq Rr$ ($S=N < V=0$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μεγαλύτερο ή ίσο ($S = 0$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 4, k.

Περίοδοι ρολογιού: 1/2

BRLT k

Λειτουργία: Αν $Rd < Rr$ ($S=N < V=1$) τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μικρότερο ($S = 1$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 4, k.

Περίοδοι ρολογιού: 1/2

BRHS k

Λειτουργία: Αν $H = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία δεκαδικού κρατουμένου ισούται με 1 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 5, k.

Περίοδοι ρολογιού: 1/2

BRHC k

Λειτουργία: Αν $H = 0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία δεκαδικού κρατουμένου ισούται με 0 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 5,k.
Περίοδοι ρολογιού: $\frac{1}{2}$

BRTS k

Λειτουργία: Αν $T = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία T ισούται με 1 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 6, k.

Περίοδοι ρολογιού: $\frac{1}{2}$

BRTC k

Λειτουργία: Αν $T = 0$ τότε $PC \leftarrow PC + k +$

1, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία T ισούται με 0 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 6,k.

Περίοδοι ρολογιού: $\frac{1}{2}$

BRVS k

Λειτουργία: Αν $V = 1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία υπερχείλισης V ισούται με 1 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 3, k.

Περίοδοι ρολογιού: $\frac{1}{2}$

BRVC k

Λειτουργία: Αν $V = 0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία υπερχείλισης V ισούται με 0 τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 3, k.

Περίοδοι ρολογιού: $\frac{1}{2}$

BRIE k

Λειτουργία: Αν $I=1$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία ολικών διακοπών ισούται με 1($I=1$) τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBS 7,k.

Περίοδοι ρολογιού: $\frac{1}{2}$

BRID k

Λειτουργία: Αν $I=0$ τότε $PC \leftarrow PC + k + 1$, αλλιώς $PC \leftarrow PC + 1$

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Όταν η σημαία ολικών διακοπών ισούται με 0 ($I=0$), δηλαδή οι διακοπές είναι απενεργοποιημένες, τότε εκτελείται σχετική διακλάδωση. Ισοδυναμεί στην εντολή BRBC 7, k.

Περίοδοι ρολογιού: 1/2

Π1.3.4 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ ΚΑΙ ΕΝΤΟΛΕΣ ΣΕ ΕΠΙΠΕΔΟ BIT

LSL Rd

Λειτουργία: $Rd(n+1) \leftarrow Rd(n)$, $Rd(0) \leftarrow 0$, $C \leftarrow Rd(7)$

Περιγραφή: Ολίσθηση των bits του καταχωρητή Rd μία θέση προς τα αριστερά. Το bit0 μηδενίζεται και το bit7 φορτώνεται στη σημαία C του SREG.

Ενημέρωση σημαιών: Z, C, N, S, V, H

LSR Rd

Λειτουργία: $Rd(n) \leftarrow Rd(n+1)$, $Rd(7) \leftarrow 0$, $C \leftarrow Rd(0)$

Περιγραφή: Ολίσθηση των bits του καταχωρητή Rd μία θέση προς τα δεξιά. Το bit7 μηδενίζεται και το bit0 φορτώνεται στη σημαία C του SREG.

Ενημέρωση σημαιών: Z, C, N, S, V

ROL Rd

Λειτουργία: $Rd(0) \leftarrow C$, $Rd(n+1) \leftarrow Rd(n)$, $C \leftarrow Rd(7)$

Περιγραφή: Περιστροφή των bits του καταχωρητή Rd μία θέση προς τα αριστερά μέσω της σημαίας κρατουμένου. Η σημαία C ολισθαίνει στο bit0 και το bit7 ολισθαίνει στη σημαία C.

Ενημέρωση σημαιών: Z, C, N, S, V, H

ROR Rd

Λειτουργία: $Rd(7) \leftarrow C$, $Rd(n) \leftarrow Rd(n+1)$, $C \leftarrow Rd(0)$

Περιγραφή: Περιστροφή των bits του καταχωρητή Rd μία θέση προς τα δεξιά μέσω της σημαίας κρατουμένου. Η σημαία C ολισθαίνει στο bit7 και το bit0 ολισθαίνει στη σημαία C.

Ενημέρωση σημαιών: Z, C, N, S, V

ASR Rd

Λειτουργία: $Rd(n) \leftarrow Rd(n+1)$ $n=0..6$, $C \leftarrow Rd(0)$

Περιγραφή: Ολίσθηση των bits του καταχωρητή Rd μία θέση προς τα δεξιά. Το bit7 διατηρείται σταθερό και το bit0 ολισθαίνει στη σημαία C.

Ενημέρωση σημαιών: Z, C, N, S, V

SWAP Rd

Λειτουργία: $R(7:4) \leftarrow Rd(3:0)$, $R(3:0) \leftarrow Rd(7:4)$

Περιγραφή: Εναλλαγή των τμημάτων high και low ενός byte.

BSET s

Λειτουργία: $SREG(s) \leftarrow 1$ Περιγραφή: Ενεργοποίηση σημαίας του καταχωρητή κατάστασης.

Ενημέρωση σημαιών: SREG(s)

Παράδειγμα: bset 5

BCLR s

Λειτουργία: $SREG(s) \leftarrow 0$ Περιγραφή: Μηδενισμός σημαίας του καταχωρητή κατάστασης.

Ενημέρωση σημαιών: SREG(s)

SBI A, s

Λειτουργία: $I/O(A,b) \leftarrow 1$

Περιγραφή: Ενεργοποίηση του bit s ενός καταχωρητή εισόδου-εξόδου.

CBI A, s

Λειτουργία: $I/O(A,b) \leftarrow 0$ Περιγραφή: Μηδενισμός του bit s ενός καταχωρητή εισόδου-εξόδου.

BST Rd, b

Λειτουργία: $T \leftarrow Rd(b)$

Περιγραφή: Αποθήκευση του bit b του καταχωρητή Rd στη σημαία T του SREG.
Ενημέρωση σημαιών: T

BLD Rd, b

Λειτουργία: $Rd(b) \leftarrow T$

Περιγραφή: Αποθήκευση της σημαίας T του SREG στο bit b του καταχωρητή Rd.

SEC

Λειτουργία: $C \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας κρατουμένου του SREG.

Ενημέρωση σημαιών: C

CLC

Λειτουργία: $C \leftarrow 0$

Περιγραφή: Μηδενισμός της σημαίας κρατουμένου του SREG.

Ενημέρωση σημαιών: C

SEN

Λειτουργία: $N \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας αρνητικού προσήμου του καταχωρητή κατάστασης.

Ενημέρωση σημαιών: N

CLN

Λειτουργία: $N \leftarrow 0$

Περιγραφή: Μηδενισμός της σημαίας αρνητικού προσήμου.

Ενημέρωση σημαιών: N

SEZ

Λειτουργία: $Z \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας μηδενισμού του καταχωρητή κατάστασης.

Ενημέρωση σημαιών: Z

CLZ

Λειτουργία: $Z \leftarrow 0$

Περιγραφή: Απενεργοποίηση της σημαίας μηδενισμού.

Ενημέρωση σημαιών: Z

SEI

Λειτουργία: $I \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας διακοπών του καταχωρητή κατάστασης.

Ενημέρωση σημαιών: I

CLI

Λειτουργία: $I \leftarrow 0$

Περιγραφή: Απενεργοποίηση της σημαίας διακοπών.

Ενημέρωση σημαιών: I

SES

Λειτουργία: $S \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας προσήμου του SREG.

Ενημέρωση σημαιών: S

CLS

Λειτουργία: $S \leftarrow 0$

Περιγραφή: Απενεργοποίηση της σημαίας προσήμου.

SEV

Λειτουργία: $V \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας υπερχείλισης στην αριθμητική συμπληρώματος 2.

Ενημέρωση σημαιών: V

CLV

Λειτουργία: $V \leftarrow 0$

Περιγραφή: Απενεργοποίηση της σημαίας υπερχείλισης.

Ενημέρωση σημαιών: V

SET

Λειτουργία: $T \leftarrow 1$

Περιγραφή: Ενεργοποίηση της σημαίας αντιγραφής και αποθήκευσης.

Ενημέρωση σημαιών: T

SHE

Λειτουργία: $H \leftarrow 1$

Περιγραφή: Απενεργοποίηση της σημαίας αντιγραφής και αποθήκευσης.

Ενημέρωση σημαιών: T

CLH

Λειτουργία: $H \leftarrow 0$

Περιγραφή: Απενεργοποίηση της σημαίας δεκαδικού κρατουμένου.

Ενημέρωση σημαιών: H

SLEEP

Περιγραφή: Θέτει το κύκλωμα σε ανενεργό κύκλο που προσδιορίζεται από τον καταχωρητή ελέγχου της MCU.

WDR

Περιγραφή: Επαναθέτει το χρονιστή επιτήρησης.

NOP

Περιγραφή: Καμία λειτουργία

ISBN 978-618-00-5262-6