ΤΕΙ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

Μαρινάκης Μανώλης, Χανιά Μάρτιος 2003

ΕΙΣΑΓΩΓΗ ΣΤΟ ΠΕΡΙΒΑΛΛΟΝ ΕΡΓΑΣΙΑΣ

Περιεχόμενο

- Διαδικασία ανάπτυξης προγραμμάτων. Το πρόγραμμα του συμβολομεταφραστή (assembler), εξομοιωτή (simulator/debugger) της Keil.
- Γνωριμία με τους καταχωρητές του μικροελεγκτή και ο τρόπος λειτουργίας τους.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να αναπτύξουν ένα πρόγραμμα
- Να εκτελούν ένα πρόγραμμα στον εξομοιωτή.
- Να απαριθμούν τα είδη των καταχωρητών
- Να φορτώνουν τους καταχωρητές με δεδομένα

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Keil (μVision)

Διαδικασία ανάπτυξης προγραμμάτων. Το πρόγραμμα της Keil.

Η ανάπτυξη εφαρμογών με μικροελεγκτές και μικροεπεξεργαστές είναι μια σύνθετη εργασία που απαιτεί εμπειρία και καλή γνώση των εντολών και των δυνατοτήτων του συγκεκριμένου επεξεργαστή.
Η ανάπτυξη της εφαρμογής αποτελείται από δύο τμήματα:

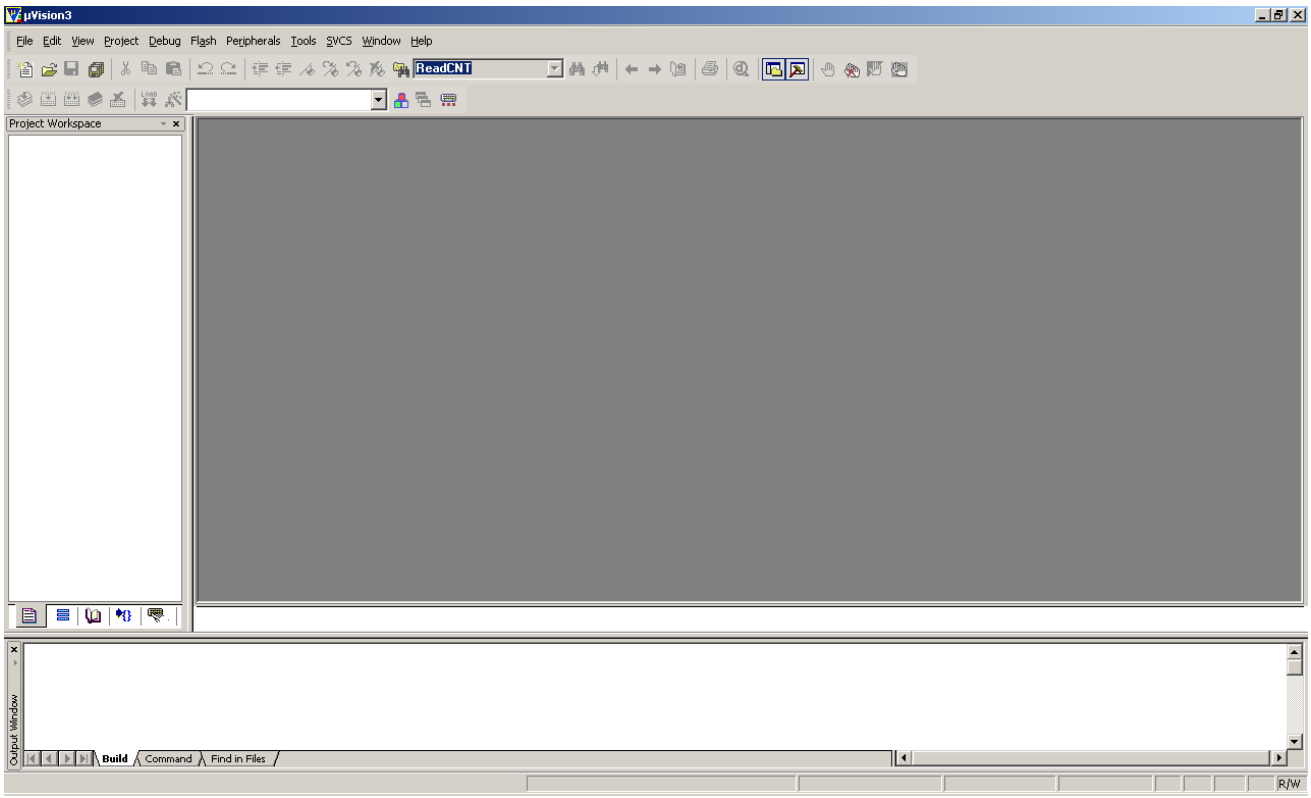
- Την ανάπτυξη του λογισμικού, δηλαδή του προγράμματος που θα τρέξει ο συγκεκριμένος μικροεπεξεργαστής ή μικροελεγκτής και
- Την ανάπτυξη του απαραίτητου υλικού (hardware) δηλαδή όλων των απαραίτητων κυκλωμάτων που χρειάζονται για να λειτουργήσει ο μικροεπεξεργαστής ή ο μικροελεγκτής.

Η κατασκευή του υλικού είναι μια χρονοβόρα και ακριβή διαδικασία. Για να βοηθηθούν οι σπουδαστές στην ανάπτυξη του λογισμικού και στην εκπαίδευση πάνω σε ένα συγκεκριμένο μικροελεγκτή διατίθεται στο εργαστήριο μια πλακέτα με κυκλώματα γενικής χρήσης και το μικροελεγκτή αυτό, ή οποία ονομάζεται **αναπτυξιακό σύστημα.** Η πλακέτα αυτή διαθέτει ότι χρειάζεται (διακόπτες, push buttons, leds, LCD display, keypad, μικροελεγκτή) για να μπορεί ο ενδιαφερόμενος να ελέγχει τα προγράμματα του.
Εκτός από το αναπτυξιακό κύκλωμα, σήμερα πολλοί κατασκευαστές διαθέτουν και έναν εξομοιωτή (simulator). Ο εξομοιωτής είναι ένα πρόγραμμα που τρέχει σε προσωπικούς υπολογιστές και το οποίο μιμείται τη συμπεριφορά του μικροελεγκτή στην οθόνη. Με άλλα λόγια στον εξομοιωτή βάζουμε το πρόγραμμα που θέλουμε να εκτελέσουμε στο μικροελεγκτή και βλέπουμε πως αυτό εκτελείται. Έτσι βλέπουμε στην οθόνη τα περιεχόμενα των καταχωρητών, της μνήμης και των περιφερειακών και καθώς

τρέχουμε το πρόγραμμα μας οι τιμές των περιεχομένων των διαφόρων στοιχείων του μικροελεγκτή αλλάζουν όπως θα άλλαζαν και στον πραγματικό μικροελεγκτή.

Η εταιρεία Keil έχει δημιουργήσει για τον 8051 και τους συμβατούς με αυτόν ένα αναπτυξιακό περιβάλλον που τρέχει σε περιβάλλον Windows και περιέχει ένα εξομοιωτή και ένα συμβολομεταφραστή (assembler) και ένα συνδέτη (linker).

Στην άσκηση αυτή θα δούμε πως γράφουμε ένα πρόγραμμα στο περιβάλλον της Keil. Αρχικά τρέχουμε το πρόγραμμα Keil μVision και ανοίγει το παράθυρο που φαίνεται στην εικόνα 1.1

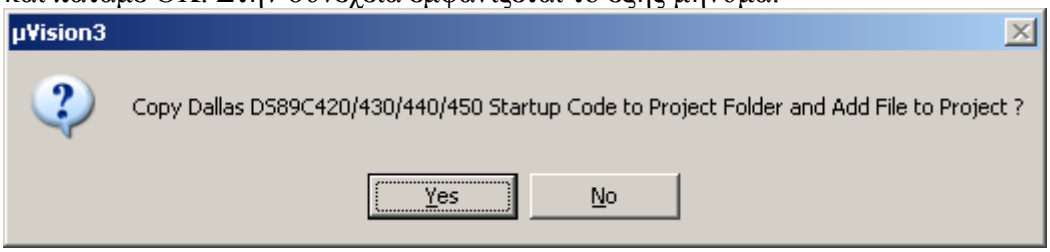


Εικόνα 1.1 Το κεντρικό παράθυρο του προγράμματος μVision

Κάθε φορά που θέλουμε να γράψουμε ένα πρόγραμμα στο περιβάλλον μVision ακολουθούμε τα παρακάτω βήματα:

Βήμα 1^{ο}: Από το μενού Project → New μVision Project

Από το παράθυρο που εμφανίζεται επιλέγουμε τον φάκελο στον οποίο θα αποθηκευθεί το project, δίνουμε όνομα στο project και πατάμε **save**. Στην συνέχεια στο παράθυρο που εμφανίζεται κάνουμε διπλό κλικ στο Dallas Semiconductor, επιλέγουμε DS89C450 και πατάμε OK. Στην συνέχεια εμφανίζεται το εξής μήνυμα:

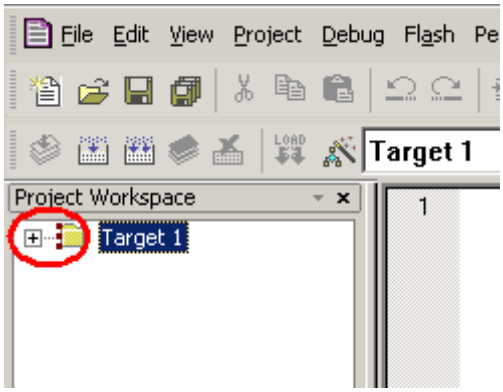


Εικόνα 1.2

Πατάμε «No».

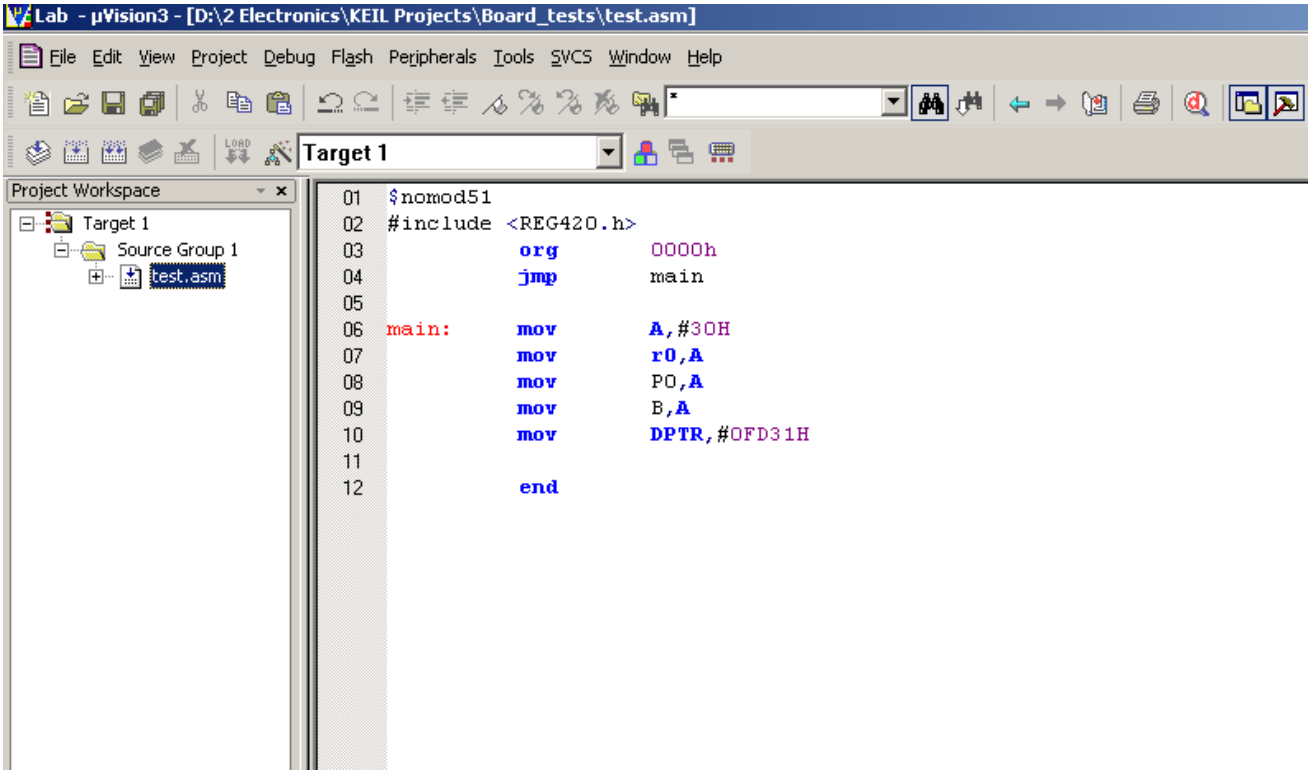
Βήμα 2^{ο}: Από το μενού File επιλέγουμε New και στην συνέχεια File →Save As. Δίνουμε όνομα στο αρχείο με κατάληξη asm (π.χ. test.asm)

Βήμα 3^{ο}: Πατάμε το «+» αριστερά από τον φάκελο Target 1 (Εικόνα 1.3). Στην συνέχεια, δεξί κλικ στον φάκελο που θα εμφανιστεί (Source Group 1) και επιλέγουμε «Add files to group 'Source Group 1'». Από εκεί επιλέγουμε το αρχείο που δημιουργήσαμε (π.χ. test.asm).



Εικόνα 1.3

Το αρχείο που δημιουργήσαμε (test.asm) είναι ένα αρχείο κειμένου στο οποίο θα συντάξουμε το πρόγραμμά μας.
Το πρόγραμμα θα ξεκινάει πάντα με την με τα εξής:
$nomod51
#include <REG420.h>
Και θα τελειώνει πάντα με ***end***. (Εικόνα 1.4)



Εικόνα 1.4

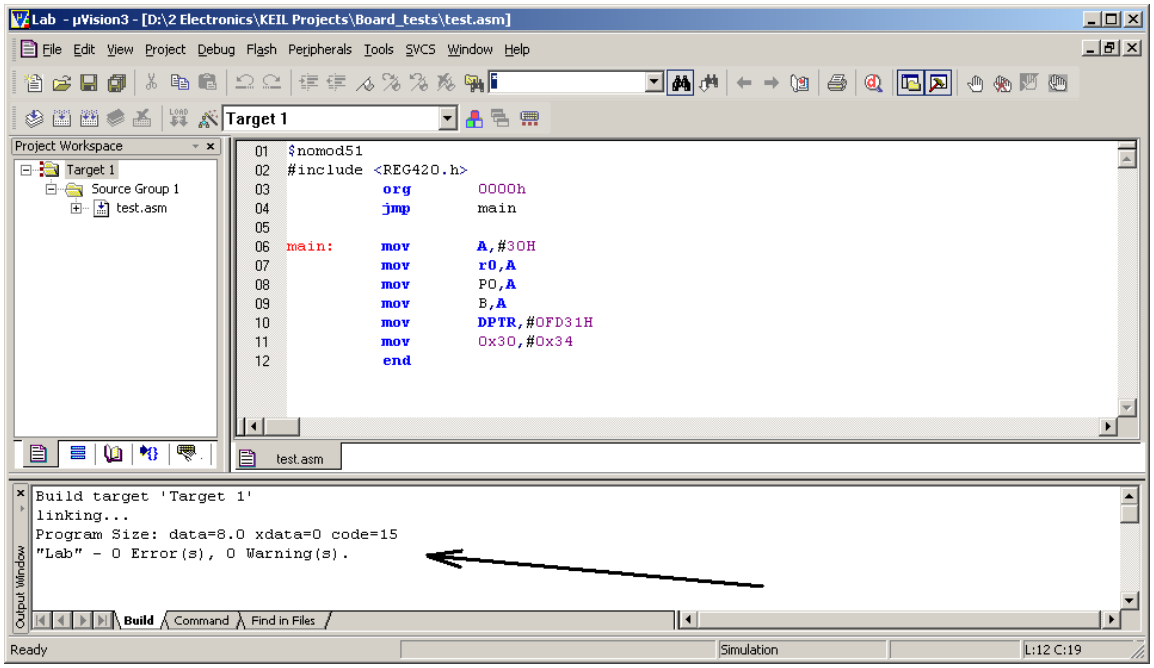
Στην συνέχεια κάνουμε δεξί κλικ στον φάκελο Target 1 (εικόνα 1.3) και επιλέγουμε «options for target 'Target 1'».

1. Επιλέγουμε την καρτέλα «Target» και στο πεδίο «XTAL(MHz)» δηλώνουμε την συχνότητα λειτουργίας του μικροελεγκτή (π.χ. 11.0592)
2. Επιλέγουμε την καρτέλα «Output» και ενεργοποιούμε την επιλογή «Create HEX file».
3. Επιλέγουμε την καρτέλα «Debug» και ενεργοποιούμε την επιλογή «Limit Speed to Real-Time».

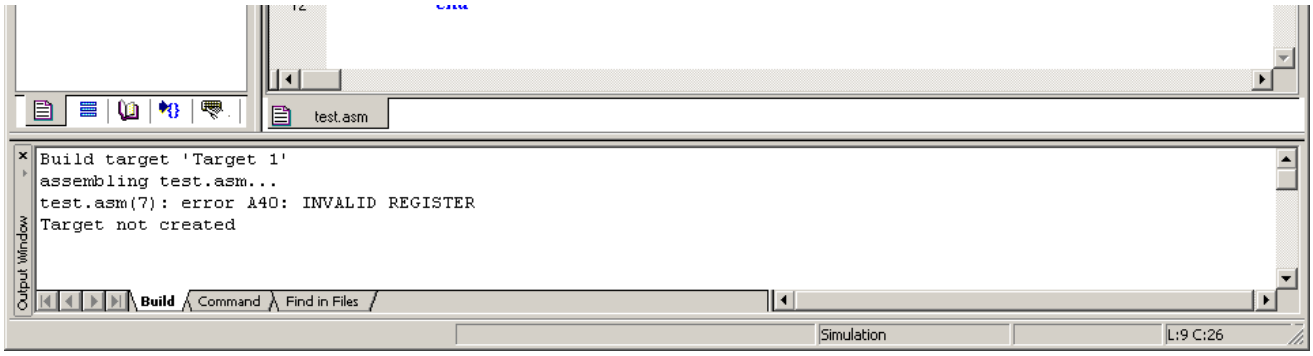
Πατάμε OK και η καρτέλα κλείνει.

Τώρα είμαστε έτοιμοι να συντάξουμε το πρόγραμμά μας (εικόνα 1.4).

Αφού συντάξουμε το πρόγραμμα πρέπει να δούμε αν υπάρχουν λάθη στην σύνταξη αυτού, και να κάνουμε την μετάφραση του προγράμματος. Αυτό γίνεται αυτόματα πατώντας από το κεντρικό μενού "Project"→ "Build Target". Αν δεν υπάρχουν λάθη στην σύνταξη του προγράμματος θα δούμε το μήνυμα, στην καρτέλα build, που εμφανίζεται στην εικόνα 1.5. Αν υπάρχουν λάθη αυτά θα εμφανιστούν όπως φαίνεται στην εικόνα 1.6. Εκεί θα δούμε τον κωδικό του λάθους και την γραμμή στην οποία υπάρχει το λάθος. Για να προχωρήσουμε παρακάτω πρέπει πρώτα να το διορθώσουμε.



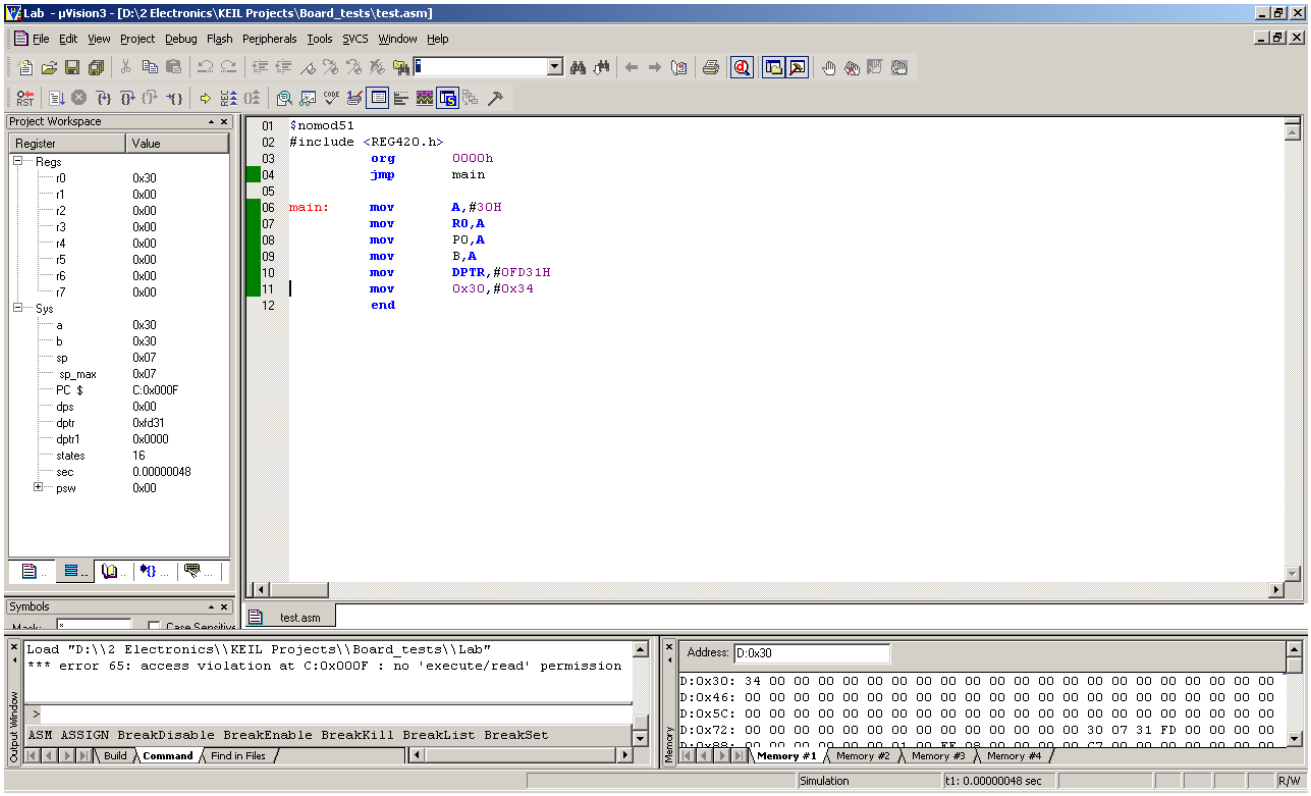
Εικόνα 1.5



Εικόνα 1.6

ΕΞΟΜΟΙΩΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

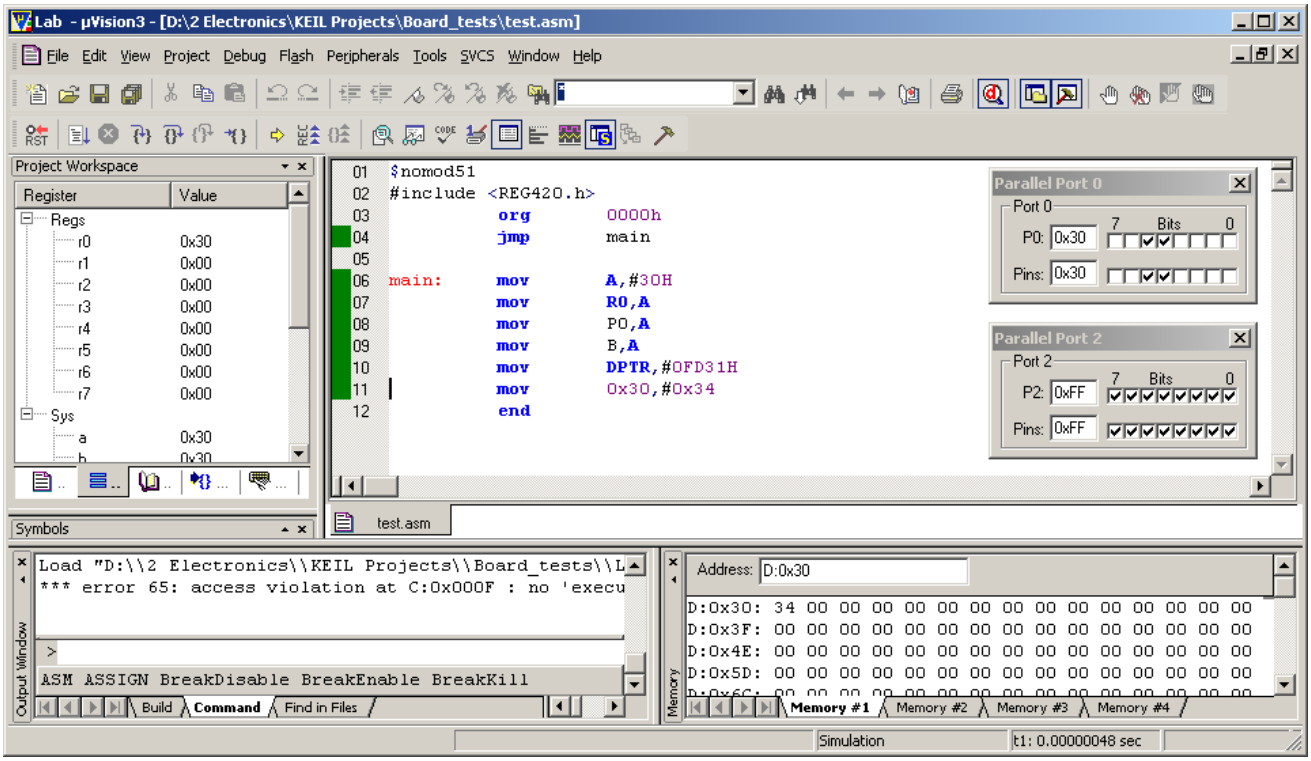
Αφού έχουμε κάνει build το πρόγραμμα είμαστε έτοιμοι για την εξομοίωση. Αυτό γίνεται πατώντας από το κεντρικό μενού “Debug” → “Start/Stop debug session” οπότε εμφανίζεται η εικόνα 1.7



Εικόνα 1.7

Αριστερά, στο παράθυρο project workspace, βλέπουμε τους καταχωρητές του 8051. Πάνω εμφανίζονται οι καταχωρητές R (R0 έως R7) και παρακάτω εμφανίζονται κάποιοι καταχωρητές ειδικών λειτουργιών (SFR) (A, B, SP…).
Κάτω δεξιά εμφανίζεται η μνήμη του 8051. Εκεί εμφανίζεται πλήθος διευθύνσεων. Για να δούμε τι τιμή υπάρχει σε μια συγκεκριμένη διεύθυνση, π.χ. 30H, την εισάγουμε στο πεδίο address και πατάμε ENTER. Αυτή η εισαγωγή γίνεται στην μορφή D:0x30. Μόλις πατήσουμε ENTER θα εμφανιστεί το περιεχόμενο της διεύθυνσης πάνω αριστερά (στην περίπτωσή μας 34 HEX).

Για να δούμε τα περιφεριακά του 8051, π.χ. τις πόρτες του, πατάμε από το κεντρικό μενού “peripherals” → I/O-ports → και την πόρτα που επιθυμούμε να δούμε. Εικόνα 1.8



Εικόνα 1.8

ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για να τρέξουμε το πρόγραμμα έχουμε δύο επιλογές:
Με το πλήκτρο **F11** όπου κάθε φορά που το πατάμε εκτελείται μία εντολή
Με το πλήκτρο **F10** όπου κάθε φορά που το πατάμε εκτελείται μία εντολή αλλά αν γίνει κλήση υπορουτίνας τότε όλη η υπορουτίνα θα εκτελεστεί σαν μία εντολή.
Με το πλήκτρο **F5** όπου εκτελείται όλο το πρόγραμμα.

ΑΣΚΗΣΗ 2

Περιεχόμενο

- Σημασία των διαφόρων τρόπων διευθυνσιοδότησης. Παρουσίαση των εντολών μεταφοράς και παραδείγματα.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να κατανοούν τους τρόπους διευθυνσιοδότησης
- Να χρησιμοποιούν, κάθε φορά, την κατάλληλη διευθυνσιοδότηση.

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Avocet

Μέρος 1^{ο}

Σαν τρόπους διευθυνσιοδότησης (Addressing modes) ορίζουμε τους τρόπους εκείνους με τους οποίους καθορίζεται η διεύθυνση των δεδομένων που παίρνουν μέρος στις εντολές. Διακρίνουμε τρεις τρόπους διευθυνσιοδότησης όπως φαίνεται παρακάτω:

1. **Άμεσος (Immediate).** Στον τρόπο αυτό τα δεδομένα βρίσκονται μέσα στην εντολή του προγράμματος αμέσως μετά το μνημονικό της εντολής.
   Π.χ. MOV A, #30H
   Μετά την εκτέλεση της εντολής, ο συσσωρευτής Α θα περιέχει τον αριθμό 30Η.
2. **Καταχωρητή (Register).** Στη διευθυνσιοδότηση καταχωρητή, τόσο η πηγή όσο και ο προορισμός, είναι καταχωρητές της CPU. Κατά τη φάση της εκτέλεση δεν έχουμε προσπέλαση στη μνήμη.
   Π.χ. MOV A, R1
   Μετά την εκτέλεση της εντολής ο καταχωρητής Α θα έχει περιεχόμενο αυτό του καταχωρητή R1 δηλαδή αντιγράφεται το περιεχόμενο του καταχωρητή R1 (καταχωρητής πηγής) στον καταχωρητή Α (καταχωρητής προορισμού).
3. **Απ' ευθείας (Direct).** Η διεύθυνση μνήμης του δεδομένου αναγράφεται αμέσως μετά από το μνημονικό της εντολής.
   Π.χ. MOV A,20H
   Μετά από την εκτέλεση της εντολής, ο καταχωρητής Α θα έχει περιεχόμενο το περιεχόμενο της διεύθυνσης μνήμης RAM 20Η. Αν δηλαδή η θέση μνήμης 20Η έχει περιεχόμενο 1ΑΗ τότε ο Α θα έχει και αυτός περιεχόμενο την ίδια τιμή 20Η.
4. **Έμμεσος καταχωρητή (register indirect).** Η διεύθυνση μνήμης του δεδομένου δίνεται από την τιμή κάποιου καταχωρητή δείκτη (R0,R1)
   Π.χ. MOV @R1,#35H

Αν ο R1 έχει την τιμή 20H μετά την εκτέλεση της εντολής η διεύθυνση μνήμης 20H θα αποκτήσει περιεχόμενο 35H.

Μεταφορά δεδομένων
Οι μεταφορές δεδομένων μεταξύ καταχωρητών και μνήμης, γίνονται κυρίως με την εντολή MOV. Η σύνταξη της εντολής γίνεται ως εξής:

Ετικέτα : **MOV καταχωρητής/μνήμη, καταχωρητής/μνήμη/άμεσα**
— Προορισμός Πηγή —

Η ετικέτα είναι προαιρετική και ο προορισμός και η πηγή πρέπει να έχουν τον ίδιο αριθμό bits. Η μνήμη στον προορισμό και την πηγή μπορεί να είναι

1. Εσωτερική RAM
2. Καταχωρητές ειδικού σκοπού
3. Εξωτερική RAM
4. Εσωτερική και εξωτερική ROM

Οι μεταφορές δεδομένων στον μικροελεγκτή 8051, μπορούν να διαιρεθούν στους ακόλουθους βασικούς τύπους:

- MOV προορισμός, πηγή
- PUSH πηγή ή POP προορισμός
- XCH προορισμός, πηγή

Οί τρόποι διευθυνσιοδότησης που είδαμε παραπάνω υποστηρίζονται από τις εντολές μεταφοράς δεδομένων. Για κάθε ένα τρόπο διευθυνσιοδότησης έχουμε τις παρακάτω περιπτώσεις σύνταξης:

1. Άμεσος και καταχωρητή
   - MOV A, #n όπου n ένα δεδομένο 8 bit
   - MOV A, Rr όπου Rr ένας από τους 8 καταχωρητές R0-R7
   - MOV Rr, A
   - MOV Rr,#n
   - MOV DPTR,#nn όπου nn ένα δεδομένο 16 bits
2. Απευθείας
   - MOV A,add όπου add μια διεύθυνση
   - MOV add,A
   - MOV Rr,add
   - MOV add,Rr
   - MOV add,#n
   - MOV add1,add2
3. Έμμεσος
   - MOV @Rp,#n όπου Rp ένας από τους καταχωρητές R0,R1
   - MOV @Rp,add
   - MOV @Rp,A
   - MOV add,@Rp
   - MOV A,@Rp

4. Ορισμένες φορές απαιτείται η προσπέλαση ενός προκαθορισμένου πλήθους δεδομένων που φυλάσσονται σε διαδοχικές θέσεις μνήμης ROM, σε μορφή ενός

πίνακα δεδομένων. Για την μόνιμη εγγραφή των δεδομένων στη μνήμη ROM χρησιμοποιούνται κατάλληλες ψευδοεντολές που διαθέτει ο Assembler (assembler directives) όπως η ψευδοεντολή DB (define byte). Η ανάγνωση των δεδομένων αυτών από τη μνήμη ROM γίνεται με έμμεσο τρόπο διευθυνσιοδότησης με χρήση τόσο του καταχωρητή Α όσο και ενός καταχωρητή δείκτη 16bit που είναι ή ο PC ή ο DPTR. Συγκεκριμένα η διεύθυνση μνήμης ROM βρίσκεται από το άθροισμα της τιμής του καταχωρητή δείκτη και του Α και το περιεχόμενο της θέσης αυτής αντιγράφεται στον Α. Η σύνταξη της εντολής είναι:
MOVC A,@A+DPTR που αντιγράφει το περιεχόμενο της θέσης μνήμης rom, που δείχνει το άθροισμα του Α και του DPTR σαν περιεχόμενο του Α

MOVC A, @A+PC , που αντιγράφει το περιεχόμενο της θέσης μνήμης rom, που δείχνει το άθροισμα του Α και του PC σαν περιεχόμενο του Α.

Σωρός

Οι εντολές διαχείρισης του σωρού χρησιμοποιούν απευθείας διευθυνσιοδότηση και είναι οι εξής:

push add Αυξάνει τον SP, Αντιγράφει τα δεδομένα της διεύθυνσης add στην διεύθυνση μνήμης της RAM που δείχνει ο SP

pop add Αντιγράφει τα δεδομένα από την διεύθυνση της εσωτερικής RAM που δείχνει ο SP στην διεύθυνση add, Μειώνει τον SP

- SP τίθεται στην διεύθυνση 07 της εσωτερικής RAM όταν γίνεται reset στον μικροελεγκτή όμως από τον προγραμματιστή τοποθετείται συνήθως σε διευθύνσεις που είναι πάνω από τις registers banks
- Όταν ο SP φτάσει την διεύθυνση FFH ξεκινά πάλι από την 00H
- Η εσωτερική RAM τελειώνει στη διεύθυνση 7FH. Αν κάνετε push πάνω απ' αυτή τη διεύθυνση θα προκύψει λάθος

Οι εντολές ανταλλαγής δεδομένων (xch) χρησιμοποιούν όλους τους τρόπους διευθυνσιοδότησης εκτός από τον άμεσο και είναι οι εξής:

xch a,Rr
xch a,add
xch a,@Rp
xchd a,@Rp Ανταλλάσσει τα λιγότερο 4 σημαντικά bits στον Α και στο περιεχόμενο της διεύθυνσης στον Rp

Μέρος Β

1. Γράψτε πρόγραμμα σε γλώσσα assembly που να θέτει στους αντίστοιχους καταχωρητές τα παρακάτω δεδομένα: A=30H, R0=2AH, P0=FFH, SP=2AH,B=3DH, DPTR=453EH. Με το βοήθεια των assembler, linker, και simulator του ολοκληρωμένου περιβάλλοντος ED95 να τρέξετε το πρόγραμμα αυτό.

2. Γράψτε πρόγραμμα που να βάζει στη θέση μνήμης 40H το περιεχόμενο 89H με απευθείας διευθυνσιοδότηση και στη θέση μνήμης 41H και 42H το περιεχόμενο D9H με έμμεση διευθυνσιοδότηση.

3. Να βάλετε τον αριθμό 8DH στις θέσεις μνήμης RAM 30H μέχρι 34H με τη χρήση και εντολών διαχείρισης σωρού.

4. Να αντιγράψετε το περιεχόμενο της θέσης μνήμης 45H στη θέση μνήμης 7DH με χρήση έμμεσου τρόπου διευθυνσιοδότησης.

5. Γράψτε πρόγραμμα που να ανταλλάσσει τα περιεχόμενα των καταχωρητών A, P3 και B, P2

6. Γράψτε πρόγραμμα που να μεταφέρει στον καταχωρητή DPTR το περιεχόμενο της διεύθυνσης 20H και της επόμενης, στη συνέχεια να ανταλλάσσεται ο DPH με τον DPL και τέλος στη θέση μνήμης 22H και την επόμενη της να μεταφέρεται η τιμή του DPTR. Πριν τρέξετε το πρόγραμμα με τη βοήθεια του εξομοιωτή να βάλετε στη διεύθυνση μνήμης 20H και της επόμενης δύο τυχαίες τιμές.

7. Γράψτε πρόγραμμα που να φορτώνει του καταχωρητές R0 μέχρι R7 της register bank 3 με τα περιεχόμενα των θέσεων μνήμης 00H – 07H.

ΑΣΚΗΣΗ 3

Περιεχόμενο

- Παρουσίαση των εντολών αριθμητικών και λογικών πράξεων με παραδείγματα.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν τις εντολές αριθμητικών πράξεων και λογικών πράξεων

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Avocet

Μέρος 1^{ο}

Θεωρητική εισαγωγή

Αριθμητικές εντολές

Στην κατηγορία των αριθμητικών πράξεων ανήκουν εντολές που εκτελούν τις βασικές αριθμητικές πράξεις μεταξύ των περιεχομένων του Α και ενός δεδομένου που προκύπτει με τους τρόπους διευθυνσιοδότησης που είδαμε προηγουμένως δηλαδή άμεσο, καταχωρητή, απευθείας και έμμεσο. Οι εντολές ομαδοποιούνται ως εξής: εντολές αύξησης ή μείωσης κατά ένα του περιεχομένου καταχωρητή ή μνήμης, εντολές πρόσθεσης και αφαίρεσης, εντολές πολλαπλασιασμού και διαίρεσης και εντολή αριθμητικής τροποποίησης. Τα μνημονικά των εντολών αυτών φαίνονται παρακάτω:

INC προορισμός
DEC προορισμός
ADD/ADDC προορισμός, πηγή
SUBB προορισμός, πηγή προορισμός ← προορισμός- πηγή-carry
MUL AB
DIV AB
DA A decimal adjust τον καταχωρητή Α

Παρακάτω φαίνονται τα μνημονικά των δυνατών προσθέσεων:

ADD A,#n
ADD A,Rr
ADD A,add
ADD A,@Rp

Επίσης τα μνημονικά των αντίστοιχων προσθέσεων με κρατούμενο :

ADDC A,#n
ADDC A,add
ADDC A,Rr
ADDC A,@Rp

Αντίστοιχα για την αφαίρεση θα έχουμε:

SUBB A,#n
SUBB A,add
SUBB A,Rr
SUBB A,@Rp

Για τις εντολές αύξησης και μείωσης κατά ένα ενός καταχωρητή ή μιας θέσης μνήμης έχουμε:

INC A
INC Rr
INC add
INC @Rp
INC DPTR
DEC A
DEC Rr
DEC add
DEC @Rp

Στην πρόσθεση και την αφαίρεση μπορούμε να χρησιμοποιήσουμε είτε αριθμούς με πρόσημο, είτε χωρίς πρόσημο.
Ένας βασικός καταχωρητής του μικροελεγκτή είναι ο PSW. Τα bit που διαθέτει δείχνουν ανάλογα με την τιμή τους (0 ή 1) κάθε στιγμή την κατάσταση της CPU
Η σημασία των bits του PSW φαίνεται παρακάτω:

CY	AC	F0	RS1	RS0	OV	------	P

Σχήμα 1.1 Ο καταχωρητής σημαιών PSW (Program Status Word)

Η σημαία Carry Flag (CY)

Η σημαία του κρατούμενου γίνεται 1 όταν υπάρχει κρατούμενο μετά από πρόσθεση ή αφαίρεση ή σύγκριση αλλιώς γίνεται μηδέν.

Η σημαία Auxiliary Carry (AC):

Η σημαία αυτή γίνεται 1 όταν υπάρχει κρατούμενο από το bit 3 του A (δηλαδή την πρώτη τετράδα – nibble), αλλιώς γίνεται 0

Η σημαία Overflow (OV)

Η σημαία υπερχείλισης γίνεται 1, όταν το αποτέλεσμα είναι μεγαλύτερο από 8 bits (με αριθμούς συμπληρώματος του 2). Επίσης όταν το αποτέλεσμα πολλαπλασιασμού (χωρίς πρόσημο) ξεπερνά το ένα byte και όταν επιχειρείται διαίρεση δια του 0.

Η σημαία Parity Flag (P)
Η σημαία ισοτιμίας τίθεται στο 1 εάν το πλήθος των 1 στον Α είναι μονός αριθμός. Εάν είναι ζυγός τότε τίθεται στο 0. Η ενημέρωση της σημαίας γίνεται μετά από κάθε εντολή.

Η σημαία F0 χρησιμοποιείται από τον προγραμματιστή όπως θέλει αυτός και οι σημαίες RS0 και RS1 επιλέγουν register bank.

Οι σημαίες που επηρεάζονται στον καταχωρητή σημαιών PSW είναι οι CY, AC, OV, P. Οι σημαίες αυτές τίθενται σε λογικό 1 ή σε 0 αυτόματα, ανάλογα με το αποτέλεσμα μιας αριθμητικής πράξης.

Πρέπει να θυμόμαστε ότι η σημαία CY επηρεάζεται από την πρόσθεση αριθμών χωρίς πρόσημο ενώ η σημαία OV από τους αριθμούς με πρόσημο (συμπλήρωμα του 2).
Η σημαία υπερχείλισης OV γίνεται 1 όταν έχουμε κρατούμενο από το bit 6 στο bit 7 και CY=0 ή όταν δεν έχουμε κρατούμενο από το bit 6 στο bit 7 αλλά CY=1. Έτσι προκειμένου περί προσημασμένων αριθμών εάν OV=1 έχουμε υπερχείλιση, αυτό σημαίνει ότι η περιοχή –128, 127 των προσημασμένων αριθμών 8 bit έχει ξεπεραστεί, άρα το αποτέλεσμα είναι λάθος. Οι ίδιες εντολές πρόσθεσης (και αφαίρεσης) χρησιμοποιούνται για προσημασμένους ή μη αριθμούς. Η διαφορά έγκειται στο εάν το πρόγραμμα ελέγχει το CY ή το OV για υπερχείλιση αποτελέσματος και στο πώς το πρόγραμμα χρησιμοποιεί το αποτέλεσμα.

Η εντολή **MUL AB** πολλαπλασιάζει τον μη προσημασμένο αριθμό του καταχωρητή Α με τον αντίστοιχο του Β, και βάζει το λιγότερο σημαντικό byte του γινομένου στον Α και το περισσότερο σημαντικό byte στον Β. Το overflow flag (OV) γίνεται 1 εάν το αποτέλεσμα είναι μεγαλύτερο του FFH. Το CY είναι πάντα 0 και το AC δεν επηρεάζεται.

Η εντολή **DIV AB** διαιρεί το byte στον καταχωρητή Α με το byte στον καταχωρητή Β και βάζει το πηλίκο στον Α και το υπόλοιπο στον Β. Το overflow flag γίνεται 1, εάν το Β=00Η πριν από την διαίρεση. Το CY είναι πάντα 0 και το AC δεν επηρεάζεται.

Λογικές εντολές

Με τις εντολές λογικών πράξεων μπορούν να γίνουν όλες οι βασικές λογικές πράξεις είτε σε επίπεδο byte, ή σε επίπεδο bit.
Επίσης υπάρχουν εντολές περιστροφής που ενεργούν είτε σε ένα byte και το κρατούμενο.

Οι εντολές λογικών πράξεων είναι:

ANL (AND)
ORL (OR)
XRL (XOR)
CPL (ΣΥΜΠΛΗΡΩΜΑ ΩΣ ΠΡΟΣ ΕΝΑ)

Οι εντολές αυτές υποστηρίζουν σε επίπεδο byte, και τους τέσσερις τρόπους διευθυνσιοδότησης που είδαμε πριν για την πηγή, και ο προορισμός είναι είτε ο καταχωρητής A ή μια απευθείας διεύθυνση.

Έτσι έχουμε για την λογική πράξη AND τις παρακάτω περιπτώσεις:

ANL A,#n
ANL A,add
ANL A,Rr
ANL A,@Rp

ANL add,A
ANL add,#n

Αντίστοιχα ισχύουν και για τις άλλες λογικές πράξεις .
Η εντολή CLR A μηδενίζει όλα τα bit του καταχωρητή A, ενώ η CPL A βρίσκει το συμπλήρωμα ως προς 1 του A.

Υπάρχουν και οι λογικές εντολές που δρουν σε επίπεδο bit, σε bit addressable διευθύνσεις RAM και καταχωρητές ειδικών λειτουργιών (sfr)
Το carry flag (CY) είναι ο προορισμός των περισσοτέρων από τις εντολές αυτές, διότι μπορεί εύκολα να ελεγχθεί η κατάσταση του και να αλλάξει η ροή του προγράμματος με εντολές άλματος υπό συνθήκη.
Οι εντολές αυτές είναι οι παρακάτω:

ANL C,b	C ←C and b
ANL C,/b	C ← C and b’
ORL C,b	
ORL C,/b	
CPL C	
CPL b	
CLR C	
CLR b	
MOV C,b	C← bit
MOV b,C	
SETB C	
SETB b	

Με τις εντολές περιστροφής ο A μπορεί να περιστραφεί κατά μια θέση δεξιά ή αριστερά με το κρατούμενο να συμπεριλαμβάνεται στην περιστροφή ή όχι.

Οι εντολές αυτές είναι:

RL A
RLC A
RR A
RRC A
SWAP A

Η τελευταία εναλλάσσει τα 4 λιγότερο σημαντικά ψηφία του A με τα 4 περισσότερο σημαντικά ψηφία (nibbles) και μπορεί να θεωρηθεί σαν μια περιστροφή των nibbles του A.

Μέρος 2^{ο}

1. Να γίνει η πρόσθεση των περιεχομένων της θέσης μνήμης 34H και του καταχωρητή B και να μπει το αποτέλεσμα στους καταχωρητές R6 (LSB) και στον R7 (MSB).

2. Αφαιρέστε το περιεχόμενο του καταχωρητή R1 από περιεχόμενο της θέσης μνήμης 42H και βάλτε το αποτέλεσμα στον καταχωρητή B.

3. Αυξήστε τα περιεχόμενα των θέσεων μνήμης 13H, 14H, και 15H χρησιμοποιώντας μόνο έμμεσο τρόπο.

4. Πολλαπλασιάστε τα περιεχόμενα της θέσης μνήμης RAM 22H με τα περιεχόμενα της θέσης μνήμης RAM 15H και βάλτε το αποτέλεσμα στη θέση μνήμης 19H (low byte) και 1AH (high byte).

5. Υψώστε στο τετράγωνο την τιμή του R5 και βάλτε το αποτέλεσμα στους R0 και R1.

6. Διαιρέστε το περιεχόμενο της θέσης μνήμης 13H με το περιεχόμενο της θέσης μνήμης 14H, και μετά ανακτήσατε το αρχικό περιεχόμενο της 13H πολλαπλασιάζοντας την απάντηση με το δεδομένο στην 14H.

7. Συμπληρώστε τα 4 λιγότερα σημαντικά bits της θέσης μνήμης 2AH.

8. Μηδενίστε το bit 3 της θέσης μνήμης 22H χωρίς να επηρεαστεί κανένα άλλο bit.

9. Αντιγράψτε το bit 6 του R0 στο bit 3 της πόρτας P0.

10. Περιστρέψτε τον καταχωρητή DPTR μια θέση προς τα αριστερά δηλαδή το bit 15 γίνεται bit 0.

11. Να γίνει ολίσθηση του καταχωρητή B μια θέση στα αριστερά.

12. Αποθηκεύστε τα 4 λιγότερα σημαντικά bits του A και στα δύο nibble (4 δυαδικά ψηφία) του περιεχομένου της θέσης 3CH.

ΑΣΚΗΣΗ 4

Περιεχόμενο

- Παρουσίαση των εντολών διακλάδωσης και κλήσης υπορουτινών με παραδείγματα.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν τις εντολές διακλάδωσης και κλήσης υπορουτινών

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Avocet

Μέρος 1^{ο}

Θεωρητική εισαγωγή
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 με την διεύθυνση της εντολής που θα γίνει η διακλάδωση. Τα άλματα έχουν τις ακόλουθες τιμές εύρους διευθύνσεων

Σχετικά(relative): Από PC+127 bytes, μέχρι PC-128 bytes μακριά από τον PC
Απόλυτα κοντινά (absolute short): οπουδήποτε σε μια σελίδα 2K bytes
Απόλυτα μακρινά (absolute long) : οπουδήποτε στην μνήμη κώδικα

Οι εντολές διακλάδωσης είναι δύο γενικών κατηγοριών: Οι διακλαδώσεις **χωρίς συνθήκη** και οι διακλαδώσεις **υπό συνθήκη.**

Διακλάδωση χωρίς συνθήκη

Οι εντολές με τις οποίες μπορεί να γίνει η "άνευ όρων¨" διακλάδωση είναι:

JMP @A+DPTR : άλμα στην διεύθυνση που δείχνεται από το άθροισμα A+DPTR

AJMP sadd : άλμα στην απόλυτα κοντινή διεύθυνση sadd (εντολή 2 bytes)
LJMP ladd : άλμα στην απόλυτα μακρινή διεύθυνση ladd (εντολή 3 bytes)
SJMP radd : σχετικό άλμα στην διεύθυνση radd (εντολή 2 bytes)
NOP : (μην κάνεις τίποτα και πήγαινε στην επόμενη εντολή)

Η διακλάδωση γίνεται σε εκείνη την εντολή που καθορίζεται από το όρισμα τους.

Διακλάδωση υπό συνθήκη με έλεγχο ενός bit

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. Τα άλματα που ελέγχουν ένα bit είναι τα παρακάτω:

JC radd : κάνε διακλάδωση στη διεύθυνση radd αν το C=1 αλλιώς συνέχισε
JNC radd : το ίδιο αν το C=0
JB b,radd : κάνε διακλάδωση αν το bit b τεθεί 1
JNB b, radd : κάνε διακλάδωση αν το bit b γίνει 0
JBC b,radd : κάνε διακλάδωση αν το bit b γίνει 1 και κάνε το b μηδέν

Ένα παράδειγμα εφαρμογής των παραπάνω εντολών είναι το εξής:

```
Loop:  mov A,#10h
       Mov r0,A
Loop1:Add A,R0
       JNC Loop1
```

Τι κάνει το παρακάτω πρόγραμμα;

Τα άλματα που ελέγχουν ένα byte είναι τα παρακάτω:

CJNE προορισμός, πηγή, διεύθυνση -Αν προορισμός <> πηγή, άλμα στη διεύθυνση

1. CJNE A, add,radd
2. CJNE A,#n,radd
3. CJNE Rn,#n,radd
4. CJNE @Rp,#n,radd

DJNZ προορισμός, διεύθυνση - Μείωση προορισμός κατά 1 και άλμα στη διεύθυνση αν το αποτέλεσμα δεν είναι μηδέν

1. DJNZ Rn,radd
2. DJNZ add,radd

JZ radd Αν A=00H άλμα στην διεύθυνση radd

JNZ radd Αν A>00H άλμα στην διεύθυνση radd

Κλήση υπορουτινών

Η κλήση υπορουτινών γίνεται με τις εντολές που ακολουθούν οι οποίες υποστηρίζουν απόλυτα κοντινά και απόλυτα μακρινά άλματα.

ACALL sadd
LCALL ladd
RET επιστροφή από υπορουτίνα
RETI επιστροφή από ρουτίνα εξυπηρέτησης διακοπής

Ένα παράδειγμα εφαρμογής των παραπάνω εντολών είναι το ακόλουθο:

```
Main:   mov  81h,#30h
        Lcall sub
        Nop
        …..
sub:    mov a,#45h
        ret
```

<u>Μέρος 2ο</u>
Ασκήσεις

1. Να γίνει πρόγραμμα που να μηδενίζει τις θέσεις μνήμης ram από 30H μέχρι 4FH

2. Να γραφεί πρόγραμμα που να μεταφέρει τον πίνακα θέσεων μνήμης από την 00H μέχρι 0FH σε μια άλλη περιοχή που αρχίζει από την 30H

3. Να γίνει πρόγραμμα που να βρίσκει τον μεγαλύτερο ενός συνόλου δεδομένων που βρίσκονται στις θέσεις από την 50H μέχρι την 70H

4. Τοποθετήστε ένα τυχαίο αριθμό στην διεύθυνση 20H και αυξήστε τον μέχρι να γίνει ίσος με ένα τυχαίο αριθμό που βάζετε στην θέση R5

5. Να γραφεί πρόγραμμα που να αθροίζει δύο αριθμούς των 3 bytes που βρίσκονται στις θέσεις μνήμης 30H-32H και 33H –35H και να βάζει το αποτέλεσμα της πρόσθεσης στις θέσεις μνήμης από 36H-39H

6. Να γίνει μια χρονοκαθυστέρηση διάρκειας 1msec και στη συνέχεια μια χρονοκαθυστέρηση 100msec

7. Να τεθεί κάθε τρίτο byte της εσωτερικής RAM από την 20h έως 7FH σε FFH

8. Μετρείστε τον αριθμό των 0 οποιουδήποτε αριθμού στον καταχωρητή R3 και βάλτε το πλήθος στον R5

ΑΣΚΗΣΗ 5

1. Περιεχόμενο

Αυτή η άσκηση έχει σαν σκοπό, την εξοικείωση με τις βασικές λειτουργίες εισόδου - εξόδου, και την κλήση υπορουτινών.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν τις πόρτες του μικροελεγκτή ώστε να κάνουν διάφορους ελέγχους.

2. Περιγραφή

Τα 32 pins εισόδου εξόδου στον 89C51, είναι οργανωμένα σε τέσσερις πόρτες 8-bit δικατευθυντήριες, που ονομάζονται P0-P3. Όπως όλοι οι SFR έτσι και οι πόρτες έχουν δικές τους διευθύνσεις και είναι bit addressable. Συγκεκριμένα οι διευθύνσεις αυτές είναι:

Ονομασία πόρτας	Διεύθυνση (HEX)
P0	80
P1	90
P2	A0
P3	B0

Κάθε πόρτα έχει ένα 8-bit latch, οι έξοδοι του οποίου οδηγούν τα pins της πόρτας αυτής. Τα περιεχόμενα των latches μπορούν να γραφούν ή να διαβαστούν σε ένα SFR. Ποιο συγκεκριμένα η τιμή που γράφεται στον αντίστοιχο SFR, προωθείται στο latch , το οποίο εξακολουθεί να εκπέμπει το σήμα και μετά το πέρας της λειτουργίας εγγραφής. Η τιμή μιας πόρτας εξόδου αλλάζει μόνον όταν μια νέα τιμή καταχωρείται latch. Η πόρτα P0 χρειάζεται pull up αντιστάσεις για να λειτουργήσει. Όταν μια πόρτα πρόκειται να χρησιμοποιηθεί σαν πόρτα εισόδου, η τιμή FFH πρέπει πρώτα να γραφεί στην πόρτα. Πιο συγκεκριμένα , διαβάζοντας τον αντίστοιχο SFR, διαβάζεται η τιμή των pins της πόρτας. Οι πόρτες 0,2 και 3 διαθέτουν εναλλακτικές λειτουργίες. Δηλαδή τα διάφορα pins αυτών των θυρών, μπορούν να χρησιμοποιηθούν σαν γραμμές ψηφιακής εισόδου - εξόδου, ή εναλλακτικά για άλλες δευτερεύουσες λειτουργίες. Συγκεκριμένα οι πόρτες 0 και 2 χρησιμεύουν για διασύνδεση με την εξωτερική μνήμη. Οι εναλλακτικές λειτουργίες για την πόρτα 3, περιλαμβάνουν εισόδους για τον timer, και τις διακοπές, πόρτα σειριακής εισόδου και εξόδου και σήματα ελέγχου για διασύνδεση με την εξωτερική μνήμη. Οι εναλλακτικές λειτουργίες της πόρτας 3 φαίνονται στον παρακάτω πίνακα:

PIN	ΧΡΗΣΗ	SFR
P3.0-RXD	Σειριακή είσοδος	SBUF
P3.1-TXD	Σειριακή έξοδος	SBUF
P3.2-INT0	Εξωτερική διακοπή 0	TCON.1
P3.3-INT1	Εξωτερική διακοπή 1	TCON.3
P3.4-T0	Timer 0 -εξωτερική είσοδος	TMOD
P3.5-T1	Timer 1-εξωτερική είσοδος	TMOD
P3.6-WR	Write για εξωτερική μνήμη	-
P3.7-RD	Read για εξωτερική μνήμη	-

3. ΣΥΝΔΕΣΗ ΤΩΝ ΘΥΡΩΝ ΜΕ ΕΞΩΤΕΡΙΚΕΣ ΣΥΣΚΕΥΕΣ

3.1.1 Οδηγώντας leds

Θεωρώντας ότι μια πόρτα του μικροελεγκτή μπορεί να δώσει το αναγκαίο ρεύμα, υπάρχουν δύο τρόποι οδήγησης ενός led, με πηγή ρεύματος ή με καταβόθρα ρεύματος. Οδήγηση με πηγή ρεύματος έχουμε όταν η άνοδος του led συνδέεται στο pin της πόρτας και η κάθοδος γειώνεται. Το led θα ανάψει όταν το pin αυτό γίνει λογικό 1 (+5V).
Αντίστοιχα όταν η άνοδος τίθεται στα 5V και για να ανάψει το led, η κάθοδος τίθεται σε λογικό 0 (0V) από το pin της πόρτας στο οποίο συνδέεται, έχουμε οδήγηση με καταβόθρα ρεύματος. Το σχήμα που ακολουθεί δείχνει τις δύο δυνατότητες.

Πηγή

Καταβόθρα

Ακολουθούν τα κατάλληλα προγράμματα.

Το πρώτο δείχνει πως μπορούμε να αναβοσβήνουμε ένα led με οδήγηση καταβόθρας ρεύματος. Το led συνδέεται στο ποδαράκι p1.1

```
        org 0000
        sjmp s1
        org 30h
s1:     clr  p1.1
        acall del
        setb p1.1
        acall del
        sjmp s1

del:    mov r3,#0ffh
```

```
lp2:    mov r2,#0ffh
lp1:    mov r1,#0ffh
lp3:    djnz r1,lp3
        djnz r2,lp1
        djnz r3,lp2
        ret
        end
```

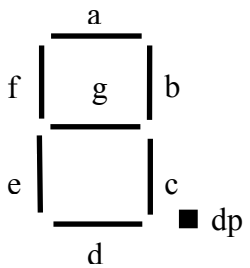
Το δεύτερο πρόγραμμα οδηγεί 8 leds συνδεδεμένα στον τρόπο λειτουργίας πηγής ρεύματος στην πόρτα 1 του μικροελεγκτή.

```
        Org 0000
        Sjmp s1
        Org 30h
s1:     mov acc,#0
        Mov p1,#0
ag:     acall del
        Inc acc
        Mov p1,acc
        Sjmp ag
del:    mov r3,#0ffh
lp2:    mov r2,#0ffh
lp1:    mov r1,#0ffh
lp3:    djnz r1,lp3
        djnz r2,lp1
        djnz r3,lp2
        ret
        end
```

Τι θα συμβεί όταν ο απαριθμητής φτάσει στο FF στο παραπάνω πρόγραμμα;

3.1.2 Οδηγώντας seven segment displays

Όπως είναι γνωστό ένα seven segment display διαθέτει 8 leds. Το όγδοο led είναι συνήθως η τελεία της υποδιαστολής. Η διάταξη φαίνεται στο σχήμα που ακολουθεί.



Τα LEDs των τμημάτων του display συνδέονται στην πόρτα 1 στον τρόπο λειτουργίας καταβόθρας ρεύματος θεωρώντας ότι το display είναι κοινής ανόδου. Δηλαδή συνδέεται η άνοδος του κάθε led σε ένα ποδαράκι της πόρτας, όπως φαίνεται στον πίνακα που ακολουθεί:

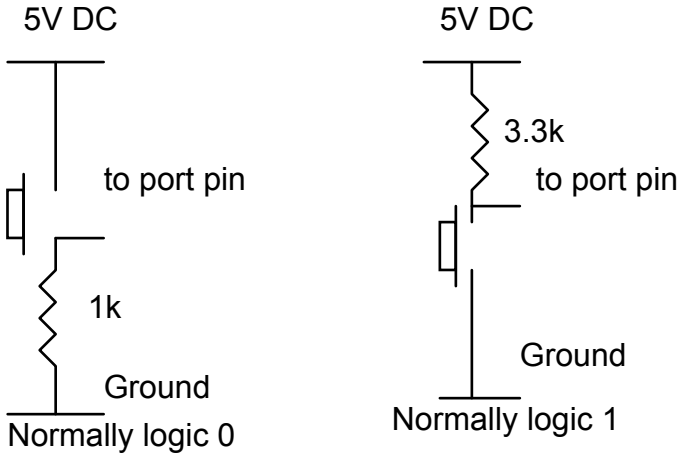
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Dp	G	F	E	D	C	B	A

Το παρακάτω πρόγραμμα απαριθμεί συνεχώς από το 0 μέχρι το 4 σε ένα seven segment display

```
        Org 0
        Sjmp s1
        Org 30h
S1:     mov p1,#11000000B ; Binary for 0
        Acall del
        Mov p1,#11111001B ; Binary for 1
        Acall del
        Mov p1,#10100100B ; Binary for 2
        Acall del
        Mov p1,#10110000Bh ; Binary for 3
        Acall del
        Mov p1,#10011001B  ; Binary for 4
        Acall del
        Sjmp s1
del:    mov r3,#0ffh
lp2:    mov r2,#0ffh
lp1:    mov r1,#0ffh
lp3:    djnz r1,lp3
        djnz r2,lp1
        djnz r3,lp2
        ret
        end
```

3.2 Διακόπτες

Τυπικά ένας διακόπτης μπορεί να είναι είτε απλός on/off ή push button ή μικροδιακόπτης (dip switch)



Μπορεί να συνδεθεί σε ένα dc τροφοδοτικό και να δώσει ένα συγκεκριμένο λογικό επίπεδο. Το σχήμα παραπάνω δείχνει τυπικές συνδεσμολογίες για διακόπτες push buttons για λογικό 0 και λογικό 1 με τον διακόπτη ανοικτό.
Ακολουθεί ένα μικρό πρόγραμμα σε assembly που περιμένει όντας σε ένα συνεχή βρόγχο μέχρι να πατηθεί ο διακόπτης. Το πρόγραμμα υποθέτει ότι η έξοδος του διακόπτη συνδέεται στο ποδαράκι p1.0

```
        org 0
        sjmp start
        org 30h
start:  jnb p1.0, start
        end
```

Όταν πατηθεί ο διακόπτης το πρόγραμμα θα βγει από το βρόγχο και θα σταματήσει.
Εάν χρησιμοποιείται ένας διακόπτης που όντας ανοικτός είναι σε λογικό 1 τότε η εντολή JNB στο πρόγραμμα παραπάνω πρέπει να αλλάξει με την εντολή JB.

Ο διακόπτης ενός πληκτρολογίου είναι push button σε κατάσταση normal open. Το σχήμα που ακολουθεί δείχνει το αποτέλεσμα του κωδωνισμού (bouncing) όταν ένας διακόπτης κλείνεται ή ανοίγεται.

Διάγραμμα χρονισμού που παράγεται από το άνοιγμα και κλείσιμο ενός διακόπτη…

Όπως φαίνεται από το σχήμα ο κωδωνισμός φαίνεται σαν πολλαπλά κλεισίματα ενός διακόπτη. Εάν θέλουμε να ελαχιστοποιήσουμε τα προβλήματα από το κωδωνισμό ενός διακόπτη συνήθως χρησιμοποιούμε μια χρονοκαθυστέρηση. Εάν περιμένουμε για 10-15 ms μετά το κλείσιμο ή άνοιγμα του διακόπτη, θα έχει σταματήσει το φαινόμενο και θα έχει περιέλθει στη σταθερή του κατάσταση. Τότε μιλάμε για software debouncing του διακόπτη. Το πρόγραμμα που ακολουθεί κάνει software debounce ενός διακόπτη συνδεμένου στο ποδαράκι 0 της πόρτας p1. Όταν ο διακόπτης είναι ανοικτός είναι σε κατάσταση λογικού 1 και όταν κλείνει δίνει στη πόρτα λογικό 0.

```
        org  30h
switch  data p1
debounce: mov switch, #0ffh
loop1:  jb switch.0, loop1     ; αναμονή μέχρι να κλείσει
        acall delay            ; αναμονή παύσης κωδωνισμού
loop2:  jnb switch.0, loop2    ; αναμονή μέχρι να ανοίξει
        acall delay            ; αναμονή παύσης κωδωνισμού
        ret
```

4. Εργασίες

1. Γράψτε ένα απλό πρόγραμμα που να ανάβει τα leds της πόρτας 1 των οποίων οι αντίστοιχοι διακόπτες στην πόρτα 2 είναι ανοικτοί.

2. Να γίνει πρόγραμμα που να παράγει στη πόρτα P1.0 τετραγωνική κυματομορφή. Ο χρόνος ON και OFF να καθορίζεται από τη ρουτίνα χρονοκαθυστέρησης που είδαμε στην προηγούμενη άσκηση.

3. Γράψτε υποπρόγραμμα που αναβοσβήνει τα leds κατά τέτοιο τρόπο, ώστε να δημιουργείται η αίσθηση ότι ένα φως περιστρέφεται από αριστερά προς τα δεξιά ή αντίστροφα. Το υποπρόγραμμα να καλείται από στοιχειώδες κυρίως πρόγραμμα. Η χρονοκαθυστέρηση θα γίνεται με την βοήθεια του κατάλληλου προγράμματος. Η θέση ενός διακόπτη συνδεμένου σε ένα ποδαράκι μιας πόρτας θα καθορίζει, την φορά περιστροφής αν είναι σε λογικό 0 (από αριστερά προς δεξιά) ή 1 (από δεξιά προς τα αριστερά).

4. Γράψτε ένα πρόγραμμα που να προσθέτει το High Nibble με το Low Nibble της P1 και το άθροισμα να εμφανίζεται σε leds που συνδέονται στην πόρτα P3.

5. Να γίνει πρόγραμμα που να μετρά σε ένα seven segment display από το 0 μέχρι το 9 και η κάθε ένδειξη να εμφανίζεται για 1 sec. Να γίνει η κατάλληλη συνδεσμολογία στο αναπτυξιακό.

ΑΣΚΗΣΗ 6

1. Περιεχόμενο

Αυτή η άσκηση έχει σαν σκοπό, την εξοικείωση με τις διακοπές (Interrupts) του μικροελεγκτή 8051.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν τις διακοπές του μικροελεγκτή ώστε να κάνουν διάφορους ελέγχους.

2. Περιγραφή

Ο 8051 έχει μια αρκετά ευέλικτη δομή διακοπών. Έχει δύο ακροδέκτες στη θύρα 3 (P3.2 και P3.3) που έχουν σαν εναλλακτική χρήση τις εξωτερικές διακοπές. Επίσης διακοπές μπορούν να προκληθούν από τα εσωτερικά περιφερειακά και συγκεκριμένα από τους χρονιστές, από την σειριακή θύρα και από το RESET. Όταν ο επεξεργαστής ανιχνεύσει μια διακοπή (π.χ. ο ακροδέκτης P3.3 (INT1) πάει σε χαμηλή στάθμη) και ισχύουν οι προϋποθέσεις που θα δούμε πιο κάτω, τότε εγκαταλείπει το κυρίως πρόγραμμα και πηγαίνει να εκτελέσει την αντίστοιχη ρουτίνα εξυπηρέτησης διακοπής. Μόλις μπει στη ρουτίνα αυτή τότε:

1. Αποταμιεύει στην στοίβα την διεύθυνση επιστροφής.
2. Η εκτέλεση συνεχίζεται στη διεύθυνση εκκίνησης της ρουτίνας εξυπηρέτησης διακοπής η οποία βρίσκεται ενσωματωμένη στο υλικό.

Όταν τελειώσει ρουτίνα εξυπηρέτησης διακοπής η τελευταία εντολή της που πρέπει να είναι η RETI, παίρνει τη διεύθυνση επιστροφής από την στοίβα και την βάζει στον PC και η εκτέλεση του προγράμματος συνεχίζεται από εκεί που σταμάτησε στο κυρίως πρόγραμμα. Όπως είδαμε η εκτέλεση των διακοπών αρχίζει σε συγκεκριμένες διευθύνσεις του κώδικα που λέγονται ανύσματα διακοπών (interrupt vector). Επειδή οι διευθύνσεις αυτές είναι κοντά η μία στην άλλη (απέχουν 8 bytes) συνήθως χρησιμοποιείται jmp για να πάμε σε άλλη περιοχή της μνήμης. Οι ανυσματικές διευθύνσεις διακοπών είναι οι εξής:

Ετικέτα	Διάνυσμα Διακοπής	Περιφερειακό
RESET	00H	POWER ON RESET
EXT INT 0	03H	Εξωτ. Διακοπή 0
TIMER 0	0BH	TIMER 0
EXT INT 1	13H	Εξωτ. Διακοπή 1
TIMER 1	1BH	TIMER 1
SER INT	23H	Σειριακή θύρα

Έτσι αν γίνει διακοπή από τον χρονιστή 0 (timer 0) τότε η εκτέλεση θα διακλαδωθεί στην διεύθυνση 0BH. Το reset είναι η πρώτη λειτουργία που γίνεται, μόλις δοθεί τροφοδοσία στον ελεγκτή, και η εκτέλεση αρχίζει από τη διεύθυνση 00 διότι μηδενίζεται ο απαριθμητής προγράμματος ενώ παίρνουν τις αρχικές τους τιμές οι SFR. Έτσι όλο το πρόγραμμα μπορεί να θεωρηθεί μια ειδική ρουτίνα εξυπηρέτησης διακοπής μα άνυσμα διακοπής στη θέση 0.
Για να εκτελεστεί μια διακοπή πρέπει να ισχύουν διάφορες προϋποθέσεις δηλαδή:

- Η συγκεκριμένη διακοπή να έχει ενεργοποιηθεί δηλαδή το συγκεκριμένο bit που της αντιστοιχεί στον καταχωρητή IE να είναι 1. Ο IE βρίσκεται στους καταχωρητής ειδικού σκοπού όπως ξέρουμε.
- Οι διακοπές να έχουν γενικά ενεργοποιηθεί, δηλαδή το bit EA του IE να είναι 1.
- Να μην έχει ταυτόχρονα ζητηθεί άλλη διακοπή υψηλότερης προτεραιότητας. Εάν υπάρχει θα εκτελεστεί πρώτη η υψηλής προτεραιότητας.

Εάν όλες οι πιο πάνω προϋποθέσεις πληρούνται τότε το υλικό επιτρέπει να αρχίσει η εκτέλεση της ρουτίνας εξυπηρέτησης διακοπής

Τα bit του καταχωρητή IE είναι τα εξής:

EA	-	-	ES	ET1	EX1	ET0	EX0

Τα bit του καταχωρητή IP είναι τα εξής:

-	-	-	PS	PT1	PX1	PT0	PX0

Ο πρώτος ενεργοποιεί τις διακοπές και δεύτερος καθορίζει την προτεραιότητα τους.
Εάν δύο διακοπές είναι της ίδιας προτεραιότητας και ζητηθούν ταυτόχρονα, τότε ο ελεγκτής καθορίζει ποια θα εξυπηρετηθεί πρώτη βάσει μιας εσωτερικής σειράς εξέτασης η οποία είναι η εξής(από υψηλή σε χαμηλή):
External 0,Timer 0, External 1, Timer 1, Serial.

3. Εργασίες

1. Να γίνει πρόγραμμα που με τη βοήθεια εξωτερικών διακοπών να ανάβει ένα LED το οποίο είναι συνδεμένο στο pin P1.0 για 1sec με το πάτημα ενός push button.

2. Γράψτε υποπρόγραμμα που αναβοσβήνει τα leds κατά τέτοιο τρόπο, ώστε να δημιουργείται η αίσθηση ότι ένα φως περιστρέφεται από αριστερά προς τα δεξιά ή αντίστοιχα. Το υποπρόγραμμα να καλείται από στοιχειώδες κυρίως πρόγραμμα. Η χρονοκαθυστέρηση θα γίνεται με την βοήθεια του κατάλληλου προγράμματος. Ένα Button στον ακροδέκτη P3.2 θα αντιστρέφει την φορά περιστροφής κάθε φορά που το πατάμε. Να χρησιμοποιήσετε διακοπές.

3. Να γίνει πρόγραμμα που να μετρά από το 0 μέχρι το 9 σε μια οθόνη επτά ψηφίων. Με τη βοήθεια διακόπτη συνδεδεμένου στο INT0 να γίνεται αντιστροφή της αρίθμησης κάθε φορά που τον πατάμε.

4. Να γραφεί πρόγραμμα που να ενεργοποιεί ένα βομβητή που είναι συνδεμένος στο P1.5 όταν πατάμε ένα Push Button που είναι συνδεμένο στο INT0 και να τον απενεργοποιεί όταν πατάμε ένα Push Button που είναι συνδεμένο στο INT1.

ΑΣΚΗΣΗ 7

Περιεχόμενο

- Χρονιστές - Απαριθμητές (Timers - Counters)

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν τους χρονιστές για χρονοκαθυστερήσεις και απαρίθμηση παλμών

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Avocet

Μέρος 1^{ο}

Θεωρητική εισαγωγή

Ο μικροελεγκτής 8051 και οι συμβατοί με αυτόν διαθέτουν ενσωματωμένες δύο περιφερειακές βαθμίδες χρονιστών / απαριθμητών. Οι βαθμίδες αυτές επιτρέπουν να υλοποιούμε χρονοκαθυστερήσεις, απαρίθμηση παλμών κ.λ.π. Όπως έχουμε ήδη αναφέρει χρονοκαθυστερήσεις μπορούν να γίνουν και με τη βοήθεια απλών προγραμμάτων όμως στην περίπτωση αυτή ο μικροϋπολογιστής είναι δεσμευμένος σ' αυτή τη λειτουργία και δεν μπορεί να κάνει κάτι άλλο παράλληλα. Έτσι αναλώνεται ο μικροελεγκτής σε απλές λειτουργίες. Οι χρονιστές απαριθμητές αξιοποιούνται καλύτερα σε συνδυασμό με τις διακοπές που μπορούν να προκαλέσουν, διότι έτσι ο μικροελεγκτής μπορεί να κάνει απερίσπαστος κάποια άλλη εργασία, έως ότου ο χρονιστής / απαριθμητής τον ειδοποιήσει με την διακοπή για να στραφεί στην εξυπηρέτηση του. Οι χρονιστές απαριθμητές έχουν διάφορους τρόπους (modes) λειτουργίας που προγραμματίζονται από τους καταχωρητές TMOD και TCON καθώς και τον IE όσον αφορά τις διακοπές.

Ας δούμε λίγο πιο αναλυτικά τους καταχωρητές TMOD και TCON.

Ο καταχωρητής TMOD (Timer/Counter Mode Control Register), μας βοηθάει να καθορίσουμε τον τρόπο λειτουργίας του Timer και τα bit του τα βλέπουμε στο παρακάτω σχήμα:

TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0

Τα πρώτα 4 bit του καταχωρητή (b0 – b3) αφορούν τον Timer 0 και τα υπόλοιπα (b4 – b7) αφορούν τον Timer 1, γι' αυτό έχουν και κοινή ονομασία.

Αναλυτικά η σημασία και η χρήση των bit αυτών είναι:

GATE: Όταν το bit αυτό είναι 1 έχουμε έλεγχο gating (Gating Control). Αυτό σημαίνει ότι οι Timers ενεργοποιούνται **μόνο** όταν τα INT0 και INT1 είναι 1 και τα bit TR0 ή/και TR1 θέτονται σε 1. Όταν το GATE είναι 0, τότε οι Timers ενεργοποιούνται και απενεργοποιούνται μότο με τα bit TR0 και TR1.

C/T: Αν το bit αυτό είναι 1 τότε ο χρονιστής δουλεύει σαν μετρητής (Counter) και παίρνει είσοδο από τους ακροδέκτες T0 ή/και T1(αν πρόκειται για το χρονιστή 0 ή/και χρονιστή 1 αντίστοιχα. Αν το bit αυτό είναι 0 τότε ο χρονιστής δουλεύει σαν Timer και παίρνει είσοδο από εσωτερικό ρολόι του συστήματος.

Τα bit M1και M0 καθορίζουν τον τρόπο λειτουργίας (Mode) του Timer όπως φαίνεται στον παρακάτω πίνακα:

M1	M0	Mode	Descreption
0	0	0	8-bit Timer/Counter. THx with TLx as 5-bit prescaler
0	1	1	16-bit Timer/Counter. THx and TLx are cascaded; there is no prescaler.
1	0	2	8-bit auto-reload Timer/Counter. THx holds a value which is to be reloaded into TLx each time it overflows.
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer only controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter stopped

Ο καταχωρητής TCON (Timer/Counter Control Register) μας βοηθάει στον έλεγχο των Timer. Τα bit αυτού του καταχωρητή είναι:

TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: Σημαία υπερχείλισης του Timer 1. Η σημαία αυτή γίνεται αυτόματα 1 όταν υπερχειλίσει ο Timer. Καθαρίζεται αυτόματα όταν ο επεξεργαστής πάει να εκτελέσει τη ρουτίνα εξυπηρέτησης διακοπής του Timer. Όταν δουλεύουμε τον Timer χωρίς τη χρήση διακοπών τότε πρέπει η σημαία αυτή να καθαρίζεται από το πρόγραμμα.

TR1: Ενεργοποίηση/ απενεργοποίηση του Timer 1. Θα μπορούσαμε να πούμε ότι αυτό το bit είναι ο διακόπτης On/Off του Timer. Έτσι όταν αυτό το bit είναι 1 ο Timer μετράει, ενώ, όταν το bit αυτό είναι 0, ο Timer δε μετράει.

Η λειτουργία των bit TF0 και TR0 είναι η ίδια με των TF1 και TR1 που περιγράψαμε παραπάνω, αλλά τα TF0 και TR0 αφορούν τον Timer 0.

Τα υπόλοιπα bit του καταχωρητή TCON δεν αφορούν τους χρονιστές, έτσι δεν θα τα δούμε σε αυτή την ενότητα.

Μέρος 2ο

Ασκήσεις

1. Να γραφεί πρόγραμμα το οποίο με τη χρήση του χρονιστή 0 να δημιουργεί τετραγωνικό παλμό 10KHz στο P1.0

2. Να γραφεί πρόγραμμα το οποίο με τη χρήση του χρονιστή 0 να δημιουργεί τετραγωνικό παλμό 1KHz στο P1.0

3. Ένας βομβητής συνδέεται στον ακροδέκτη P1.7 και ένα διακόπτης (pulse button) στο P1.6. Να γραφεί πρόγραμμα που να διαβάζει το λογικό επίπεδο του διακόπτη και να ενεργοποιεί τον βομβητή για 1 δευτερόλεπτο κάθε φορά που ανιχνεύεται μετάβαση από 1 σε 0.

4. Να γραφεί πρόγραμμα που, με τη χρήση χρονιστών, να δημιούργει ταυτόχρονα τετραγωνικούς παλμούς 7KHz και 500Hz στους ακροδέκτες P1.7 και P1.6 αντίστοιχα.

ΑΣΚΗΣΗ 8

Περιεχόμενο

- Έλεγχος σειριακής θύρας του 8051 (Serial Port)

.

Μετά την εκτέλεση της άσκησης οι σπουδαστές πρέπει να μπορούν…

- Να χρησιμοποιούν την σειριακή θύρα για αποστολή και λήψη χαρακτήρων.

Προτεινόμενος εργαστηριακός εξοπλισμός

- Ένας προσωπικός υπολογιστής με λειτουργικό windows
- Το πρόγραμμα της Avocet

Μέρος 1^{ο}

Θεωρητική εισαγωγή

Ο 8051 περιλαμβάνει μία σειριακή θύρα η οποία μπορεί να λειτουργήσει με διάφορους τρόπους (Modes) σε μια μεγάλη γκάμα συχνοτήτων. Η ουσιαστική λειτουργία της σειριακής θύρας είναι να παρέχει παράλληλη σε σειριακή μετατροπή των εξερχόμενων δεδομένων (output data) και σειριακή σε παράλληλη μετατροπή των εισερχόμενων δεδομένων (input data). Η πρόσβαση του hardware στην σειριακή θύρα γίνεται μέσω των ακροδεκτών RxD (Receive Data) και TxD (Transmit Data). Αυτά τα pins είναι το 11 (P3.1) για το TxD και το 10 (P3.0) για το RxD. Η σειριακή θύρα παρέχει πλήρως αμφίδρομη (full duplex) επικοινωνία, δηλαδή ταυτόχρονη αποστολή και λήψη δεδομένων.

Για τον προγραμματισμό και την χρήση της σειριακής χρησιμοποιούμε δύο καταχωρητές ειδικής λειτουργίας (SFR) τον SBUF και τον SCON. Στον SBUF καταχωρούνται τα δεδομένα προς αποστολή ή/και λήψη. Ο SCON καθορίζει τους τρόπους λειτουργίας της σειριακής θύρας. Το baud rate της θύρας ρυθμίζεται με την βοήθεια του Timer1, όπως θα δούμε παρακάτω.

Τα bit του καταχωρητή SCON φαίνονται στο παρακάτω σχήμα:

SCON	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

Τα bit SM0 και SM1 καθορίζουν σε τι Mode θα δουλεύει η σειριακή σύμφωνα με τον πίνακα που φαίνεται παρακάτω:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	FOSC/12
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	FOSC/64 or FOSC/32
1	1	3	9-bit UART	variable

UART : Universal Asynchronous Receiver/Transmitter
FOSC : Συχνότητα Κρυστάλλου

Το Baud Rate στα Mode 1 και 3, ρυθμίζεται με τη βοήθεια του Timer 1, τον οποίο δουλεύομε σε Mode2 (auto-reload), και η τιμή που θα βάλουμε στον TH1 την υπολογίζουμε με τη βοήθεια της σχέσης:

$$BaudRate = \frac{2^{SMOD1} \cdot Fosc}{32 \cdot 12 \cdot [256 - (TH1)]}$$

Το SMOD1 είναι το bit 6 του καταχωτητή PCON, και με τη βοήθεια του μπορούμε να διπλασιάζουμε το Baud Rate θέτοντας το απλά σε 1. Η εξ' ορισμού (default) τιμή αυτού του bit είναι 0.
Λύνοντας αυτή τη σχέση ως προς TH1 έχουμε:

$$TH1 = 256 - \frac{2^{SMOD1} \cdot Fosc}{384 \cdot BaudRate}$$

Από την τελευταία σχέση παρατηρούμε ότι γνωρίζοντας τη συχνότητα κρυστάλλου (Fosc) και το επιθυμητό Baud Rate, μπορούμε εύκολα να υπολογίσουμε την τιμή του TH1.

Το bit REN ενεργοποιεί τη λήψη της σειριακής θύρας. Έτσι εάν θέλουμε να είναι ενεργοποιημένη η λήψη τότε κάνουμε αυτό το bit 1, διαφορετικά το κάνουμε 0.

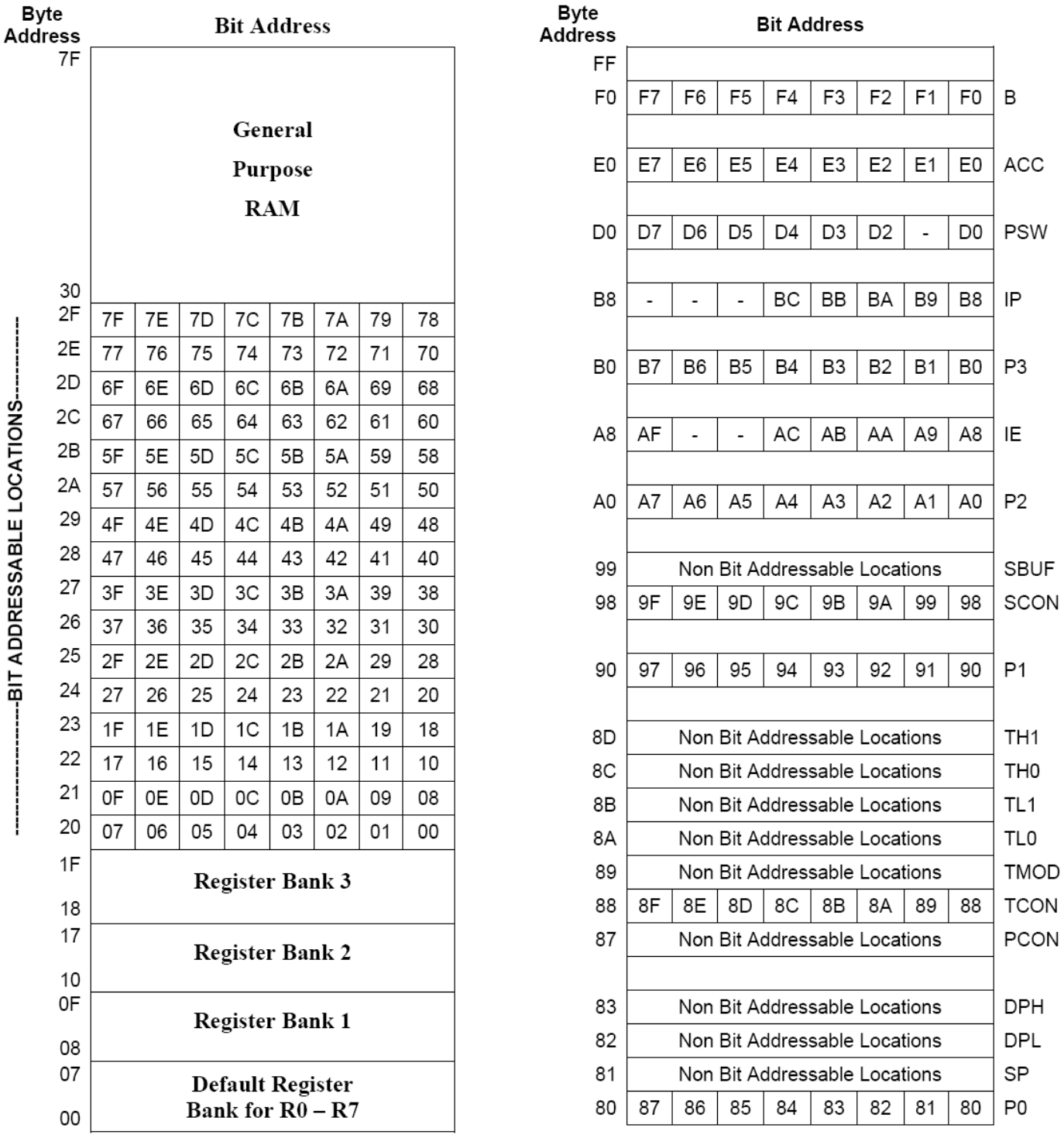
Το bit TI είναι η σημαία της διακοπής αποστολής (Transmit interrupt flag). Η σημαία αυτή γίνεται αυτόματα 1 μόλις ολοκληρωθεί η αποστολή ενός χαρακτήρα από τη σειριακή θύρα. Η σημαία αυτή δεν γίνεται αυτόματα 0 και έτσι αυτό πρέπει να γίνεται από το πρόγραμμα.

Το bit RI είναι η σημαία της διακοπής λήψης (Receive interrupt flag). Η σημαία αυτή γίνεται αυτόματα 1 μόλις ολοκληρωθεί η λήψη ενός χαρακτήρα από τη σειριακή θύρα. Η σημαία αυτή δεν γίνεται αυτόματα 0 και έτσι αυτό πρέπει να γίνεται από το πρόγραμμα.

Μέρος 2ο

Πειραματικό μέρος:

Προγραμματίστε τον Timer 1 του 8051 να παράγει τον κατάλληλο χρονισμό για να στέλνουμε χαρακτήρες με ταχύτητα 2400 bps. Στην συνέχεια να γίνει πρόγραμμα το οποίο να αποστέλλει στην σειριακή θύρα το μήνυμα "TEI HLEKTRONIKHS".



ΧΑΡΤΗΣ ΜΝΗΜΗΣ RAM ΤΟΥ 8051

Register Bank 0		Register Bank 1		Register Bank 2		Register Bank 3	
RS0:0 RS1:0		RS0:1 RS1:0		RS0:0 RS1:1		RS0:1 RS1:1	
Κατ/τής	Διεύθυνση	Κατ/τής	Διεύθυνση	Κατ/τής	Διεύθυνση	Κατ/τής	Διεύθυνση
R0	00	R0	08	R0	10	R0	18
R1	01	R1	09	R1	11	R1	19
R2	02	R2	0A	R2	12	R2	1A
R3	03	R3	0B	R3	13	R3	1B
R4	04	R4	0C	R4	14	R4	1C
R5	05	R5	0D	R5	15	R5	1D
R6	06	R6	0E	R6	16	R6	1E
R7	07	R7	0F	R7	17	R7	1F

Οι 32 καταχωρητές γενικού σκοπού που υπάρχουν στη RAM και οι διευθύνσεις τους

SFR	διεύθυνση(HEX)
A	E0
B	F0
DPL	82
DPH	83
IE	A8
IP	B8
P0	80
P1	90
P2	A0
P3	B0
PCON	87
PSW	D0
SBUF	99
SCON	98
SP	81
TCON	88
TMOD	89
TH0	8C
TL0	8A
TH1	8D
TL1	8B

Οι Καταχωρητές ειδικού σκοπού του μικροελεγκτή 8051 και οι διευθύνσεις τους