Άσκηση 6 - Απαντήσεις

Β. ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ

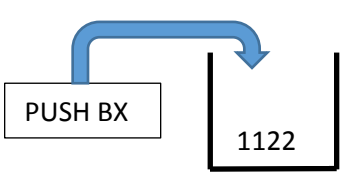
1. Πρόγραμμα που φορτώνει το περιεχόμενο του BX στο CX μέσω του σωρού, θέτοντας όμως αρχικά τη διεύθυνση κορυφής του σωρού σε 007E

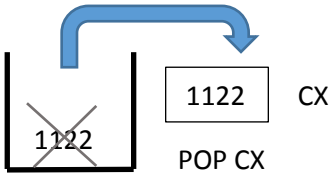
mov sp,007E
mov BX,1122
push BX
pop CX

α) Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος
β) Πώς μεταβάλλεται το περιεχόμενο του sp;

Ερώτημα 1 - Απαντήσεις

Αρχικά, θέτουμε περιεχόμενο στον καταχωρητή *sp*, καθορίζουμε δηλαδή την κορυφή του σωρού. Μόλις εκτελεστεί αυτή η εντολή, θα πρέπει απλά να παρατηρήσουμε την αντίστοιχη αλλαγή στο σωρό που εμφανίζεται στον debugger (περιοχή κάτω δεξιά). Στη συνέχεια, φορτώνουμε το 1122 στον BX, ενώ με την εντολή *push BX* προωθούμε το περιεχόμενό του στο σωρό. Εφόσον αρχικά sp=007E, μετά την προώθηση θα πρέπει να ισχύει ότι sp=007C. Θυμηθείτε ότι, η αλλαγή του *sp* θα είναι κατά δύο θέσεις μνήμης, αφού πάντα αποθηκεύονται 2 byte. Το πρόγραμμα ολοκληρώνεται με την εντολή *pop CX*. Έτσι, αυτό που αποθηκεύτηκε τελευταίο στο σωρό (τιμή 1122) θα ανακτηθεί πρώτο. Αυτό σημαίνει ότι τώρα, CX=1122, αφού το CX βάλαμε ως προορισμό της τιμής που ανακτήθηκε από το σωρό. Ο καταχωρητής *sp* περιέχει τώρα ξανά την τιμή 007E.





2. Πρόγραμμα που ανταλλάσσει τα περιεχόμενα των καταχωρητών AX και BX μέσω του σωρού

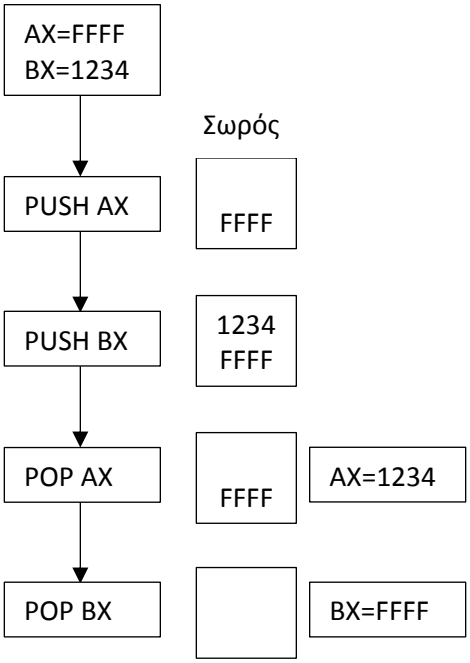
mov AX,0ffff
mov BX,1234
push AX
push BX
pop AX
pop BX

α) Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος
β) Σχεδιάστε μπλοκ διάγραμμα που να δείχνει τη λειτουργία του προγράμματος

Ερώτημα 2 - Απαντήσεις

Στην αρχή, φορτώνουμε με κάποια τιμή τους καταχωρητές AX και BX. Οι δύο διαδοχικές προωθήσεις καταχωρητών (Push AX, Push BX), θα διαμορφώσουν το περιεχόμενο του σωρού ως εξής:

Μετά τα διαδοχικά pop, AX=1234 και BX=FFFF



3. Πρόγραμμα που θέτει 1 την τιμή όλων των bit του καταχωρητή κατάστασης και μετά επαναφέρει την αρχική

```
mov AX,0ffff        Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος
pushf
push AX
popf
popf
```

Ερώτημα 3 - Απαντήσεις

Αρχικά, φορτώνουμε την τιμή FFFF στον καταχωρητή AX (1111 1111 1111 1111). Με την εντολή *pushf* αποθηκεύουμε το περιεχόμενο του καταχωρητή κατάστασης στο σωρό. Θυμηθείτε ότι, οποιαδήποτε προώθηση στο σωρό γίνεται με δεδομένα 16bit. Από την άλλη μεριά όμως, έχουμε αναφέρει ότι, ο καταχωρητής κατάστασης είναι 8bit. Ποια είναι όμως τα υπόλοιπα bit που αποθηκεύονται στο σωρού, αφού πρέπει να προωθηθούν 16bit και όχι 8bit; Η απάντηση έρχεται από τον κατασκευαστή του μικροεπεξεργαστή που χρησιμοποιούμε. Όπως φαίνεται στο σχήμα που ακολουθεί, ο καταχωρητής κατάστασης σε επίπεδο κυκλώματος, έχει υλοποιηθεί με 16bit, απλά κάποια από αυτά δεν χρησιμοποιούνται.

Bit→	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	--	--	--	--	OF	DF	IF	TF	SF	ZF	--	AF	--	PF	--	CF

Επομένως, εστιάζουμε μόνο στα bit που μας ενδιαφέρουν. Άρα, με την εντολή *pushf*, προωθούνται στο σωρό τα 16bit που φαίνονται στο παραπάνω σχήμα. Στη συνέχεια, προωθούμε στην κορυφή του σωρού το περιεχόμενο του AX, δηλαδή το FFFF. Με την εντολή *popf*, ανακτούμε 2 byte από την κορυφή του σωρού τα οποία θα γίνουν το νέο περιεχόμενο του καταχωρητή κατάστασης. Αφού όμως στην κορυφή ήταν το FFFF, όλα τα bit του καταχωρητή κατάστασης, θα γίνουν 1. Το σχήμα που ακολουθεί είναι χαρακτηριστικό.

c = 1
z = 1
s = 1
o = 1
p = 1
a = 1
i = 1
d = 1

Ανακτώντας όμως το FFFF, τώρα έχει μείνει στην κορυφή του σωρού το παλιό περιεχόμενο του καταχωρητή κατάστασης. Έτσι, εκτελώντας τώρα την εντολή *popf* (δεύτερη φορά), ο καταχωρητής κατάστασης ξαναπαίρνει το αρχικό του περιεχόμενο, πριν δηλαδή από την κατάσταση που φαίνεται στο παραπάνω σχήμα.

4. Πρόγραμμα που ενεργοποιεί το CF του καταχωρητή κατάστασης (θεωρώντας ότι, το CF αποτελεί το λιγότερο σημαντικό ψηφίο του καταχωρητή κατάστασης)

```
pushf
pop AX
or AX,1
push AX
popf
```

α) Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος
β) Εξηγήστε τη λειτουργία της λογικής πράξης ως προς το επιθυμητό αποτέλεσμα

Ερώτημα 4 - Απαντήσεις

Για να τροποποιηθεί το περιεχόμενο του καταχωρητή κατάστασης, θα πρέπει αυτό να φορτωθεί προσωρινά σε κάποιο καταχωρητή και στη συνέχεια να γίνει η αντίστοιχη ενημέρωση. Για αυτό το λόγο, αρχικά προωθούμε στο σωρό το περιεχόμενο του καταχωρητή κατάστασης (*pushf*). Με την εντολή *pop AX*, φορτώνονται 2 byte από την κορυφή του σωρού, δηλαδή το περιεχόμενο του καταχωρητή κατάστασης. Η πράξη OR γίνεται με το περιεχόμενο του AX και το 0000 0000 0000 0001, αφού το AX είναι 16bit. Πράξη OR με μηδέν, δεν μεταβάλει την υπάρχουσα τιμή του αντίστοιχου bit. OR όμως με 1, θα κάνει σίγουρα το συγκεκριμένο bit 1, ανεξάρτητα από την υπάρχουσα τιμή του. Από το σχήμα που ακολουθεί φαίνεται ότι, πρέπει να παρέμβουμε στο λιγότερο σημαντικό bit που αντιπροσωπεύει το CF.

Bit→	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	--	--	--	--	OF	DF	IF	TF	SF	ZF	--	AF	--	PF	--	CF

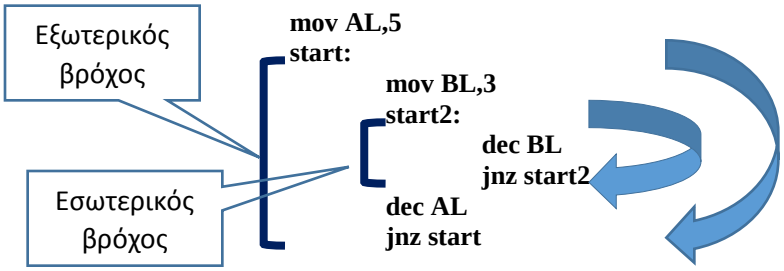
Μετά την παρέμβαση στο λιγότερο σημαντικό bit, προωθούμε το νέο περιεχόμενο του AX στο σωρό. Τέλος, ενημερώνουμε το περιεχόμενο του καταχωρητή κατάστασης με την εντολή *popf*, στην ουσία δηλαδή από το τελευταίο περιεχόμενο του AX.

5. Διπλός βρόχος με χρήση δύο καταχωρητών (5x3=15 επαναλήψεις)

```
mov AL,5
start:
        mov BL,3
        start2:
                dec BL
                jnz start2
        dec AL
        jnz start
```

Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος

Ερώτημα 5 - Απαντήσεις



Κάθε τρεις κύκλους του εσωτερικού βρόχου, ολοκληρώνεται ένας κύκλος του εξωτερικού βρόχου. Άρα, 5 κύκλοι του εξωτερικού επί 3 κύκλοι του εσωτερικού, μας δίνουν συνολικά 5X3=15 επαναλήψεις (εντός του εσωτερικού βρόχου). Οι βρόχοι έχουν υλοποιηθεί με δομή τύπου do-while, ενώ έχουν αναπτυχθεί ακριβώς με τον ίδιο τρόπο (αλλάζει μόνο ο καταχωρητής που λειτουργεί ως μετρητής).

6. Διπλός βρόχος με χρήση ενός καταχωρητή (5x3=15 επαναλήψεις)

```
mov AX,5                    Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος
start:
        push AX
        mov AX,3
        start2:
                dec AX
                jnz start2
        pop AX
        dec AX
        jnz start
```

Ερώτημα 6 - Απαντήσεις

Όπως θα έχετε ήδη παρατηρήσει, οι βασικοί καταχωρητές γενικής χρήσης είναι μόλις τέσσερις (AX, BX, CX, DX) και μπορούν να διπλασιαστούν αν χρησιμοποιηθούν τμηματικά ως 8bit (κάθε καταχωρητής 16bit προσφέρει δύο 8bit καταχωρητές, π.χ. ο AX αποτελείται από τους AH και AL των 8bit, ώστε συνολικά να σχηματίζονται τα 16bit). Στις πραγματικές εφαρμογές, τέσσερις μόνο καταχωρητές περιορίζουν σημαντικά τον προγραμματιστή στις επιλογές του. Αυτό το πρόβλημα λύνεται αν απλά ο προγραμματιστής χρησιμοποιήσει την κεντρική μνήμη του υπολογιστή που εκτείνεται σε χιλιάδες ή εκατομμύρια θέσεις. Για αυτό το λόγο και ο μικροεπεξεργαστής υποστηρίζει πολλούς τρόπους διευθυνσιοδότητης, πολλούς τρόπους δηλαδή για την προσπέλαση της μνήμης. Όπως είδαμε σε προηγούμενα ερωτήματα, ο σωρός είναι ένας τρόπος οργάνωσης δεδομένων στη μνήμη του υπολογιστή, η λειτουργία του οποίου υποστηρίζεται από μερικές μόνο απλές εντολές (προώθηση και ανάκτηση). Στις περιπτώσεις που το πλήθος των καταχωρητών είναι περιορισμένο, μπορεί να χρησιμοποιηθεί ο σωρός, αφού και αυτός υλοποιείται στη μνήμη. Στο πρόγραμμα αυτού του ερωτήματος, θέλουμε να χρησιμοποιήσουμε τον ίδιο καταχωρητή ως μετρητή και για τους δύο βρόχους. Φυσικά, αυτό δεν θα μπορούσε να γίνει χωρίς προσωρινή αποθήκευση του ενός μετρητή. Αν το κάναμε αυτό χωρίς την ενδιάμεση προσωρινή αποθήκευση, τότε θα καταστρέφαμε την τιμή του μετρητή του εξωτερικού βρόχου, με αποτέλεσμα να μην γίνεται το επιθυμητό πλήθος επαναλήψεων.
Το πρόγραμμα ξεκινά κανονικά με τον εξωτερικό βρόχο και την αρχικοποίηση του μετρητή (AX=5) πριν από αυτόν. Προχωρώντας στο εσωτερικό, αποθηκεύουμε στο σωρό την τιμή του μετρητή, προκειμένου να προχωρήσουμε στην αρχικοποίηση του μετρητή του εσωτερικού βρόχου, χρησιμοποιώντας πάντα τον ίδιο καταχωρητή. Μόλις ολοκληρωθούν οι κύκλοι του εσωτερικού βρόχου, τότε ανακτούμε από την κορυφή του σωρού την τρέχουσα τιμή του μετρητή που χρησιμοποιείται στον εξωτερικό βρόχο. Έτσι, τώρα τον μειώνουμε και ελέγχουμε αν έχει φτάσει στο μηδέν και οδηγούμε κατά περίπτωση τη ροή εκτέλεσης στο σημείο start.

7. Πρόγραμμα που υπολογίζει το άθροισμα των στοιχείων ενός πίνακα 5 θέσεων που ξεκινά από τη διεύθυνση [0000]. Το άθροισμα αποθηκεύεται στη θέση [0005].

```
mov CL,0
mov BX,0
start:
        mov AL,[BX]
        add CL,AL
        inc BX
        cmp BX,4
        jle start
mov [0005],CL
```

α) Καταχωρήστε στο Data Segment αριθμούς της επιλογής σας (θέσεις [0000] έως [0004])
β) Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος

Ερώτημα 7 - Απαντήσεις

Τόσο ο υπολογισμός του αθροίσματος, όσο και ο τρόπος υλοποίησης του βρόχου παραμένει ίδιος, όπως το έχουμε ήδη αναλύσει. Η μόνη διαφοροποίηση είναι στους αριθμούς που συμμετέχουν στον υπολογισμό του αθροίσματος και προέρχονται από την κεντρική μνήμη. Η ανάγνωση κάθε αριθμού γίνεται χρησιμοποιώντας την εντολή mov AL,[BX], ελέγχοντας τις τιμές του BX. Ο καταχωρητής BX παίρνει τις τιμές 0 έως 4, άρα διαβάζονται τα αντίστοιχα byte, ως εξής:

```
mov AL,[0]
mov AL,[1]
mov AL,[2]
mov AL,[3]
mov AL,[4]
```

Η αλλαγή της τιμής στην αγκύλη, εξασφαλίζεται χρησιμοποιώντας το BX και μέσω του βρόχου εναλλάσσονται οι τιμές. Έτσι, το άθροισμα θα είναι:

Περιεχόμενο θέσης [0] + Περιεχόμενο θέσης [1] + Περιεχόμενο θέσης [2] + Περιεχόμενο θέσης [3] + Περιεχόμενο θέσης [4]

8. Πρόγραμμα που μετρά το πλήθος των μηδενικών στοιχείων ενός πίνακα 16 θέσεων που ξεκινά από τη διεύθυνση [0000]. Το πλήθος των μηδενικών αριθμών αποθηκεύεται στη διεύθυνση [0005]

```
mov CL,0
mov BX,0
start:
        mov AL,[BX]
        cmp AL,0
        jne cont (διεύθυνση 010Dh)
        inc CL
cont:
        inc BX
        cmp BX,0F
        jle start (διεύθυνση 0105)
mov [0005],CL
```

α) Καταχωρήστε στο Data Segment αριθμούς της επιλογής σας (θέσεις [0000] έως [0004])
β) Δοκιμάστε και σχολιάστε τη λειτουργία του προγράμματος

Ερώτημα 8 - Απαντήσεις

Το τμήμα προγράμματος που αποτελεί τον πυρήνα του ελέγχου για την εύρεση των μηδενικών στοιχείων, είναι:

cmp AL,0

jne cont (διεύθυνση 010Dh)

inc CL

cont:

Για κάθε τιμή που διαβάζουμε από την τρέχουσα θέση μνήμης, πραγματοποιούμε έλεγχο ισότητας με το μηδέν. Αν ο αριθμός δεν είναι ίσος με το μηδέν (not equal), τότε γίνεται άλμα στο σημείο *cont*, όπου συνεχίζεται ο βρόχος για το επόμενο στοιχείο. Αν η συνθήκη είναι ψευδής (άρα ο αριθμός είναι μηδέν), τότε δεν γίνεται άλμα στο σημείο *cont* και επομένως εκτελείται η αύξηση του CL που φιλοξενεί το πλήθος των μηδενικών αριθμών και συνεχίζεται ο βρόχος (δεν σταματά η ροή στο *cont*, αφού πρόκειται για ετικέτα).