Αρχιτεκτονική Υπολογιστών

Ενότητα 4: Πολλαπλασιασμός (MUL,IMUL) . Διαίρεση (DIV,IDIV). Εμφάνιση αλφαριθμητικού. Εμφάνιση χαρακτήρα. Είσοδος χαρακτήρα. Μεταφορές 8bit<=>16bit.

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.ece.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Εντολή Πολλαπλασιασμού 8bit MUL

- Η εντολή MUL χρησιμοποιείται για πολ/σμό μη προσημασμένων αριθμών και συντάσσεται:

```
MUL (παράμετρος1)
```

- Όπου (παράμετρος1) είναι ένας τελεστής, είτε καταχωρητής γενικής χρήσης, είτε μεταβλητή μνήμης.
- Αν ο τελεστής είναι ενός byte τότε η (παράμετρος1) **πολλαπλασιάζει το περιεχόμενο του AL και το αποτέλεσμα αποθηκεύεται στον AX.**

Παράδειγμα:

Αν AL=10 και DL=10 μετά την εντολή **MUL DL** θα γίνει ο πολλαπλασιασμός του DL με τον AL και το αποτέλεσμα θα μπει στον AX. Δηλαδή, ο AX=100.

Εντολή Πολλαπλασιασμού 16bit MUL

- Η εντολή MUL χρησιμοποιείται για πολ/σμό μη προσημασμένων αριθμών και συντάσσεται:

```
MUL (παράμετρος1)
```

- Όπου (παράμετρος1) είναι ένας τελεστής, είτε καταχωρητής γενικής χρήσης, είτε μεταβλητή μνήμης.
- Αν ο τελεστής είναι δύο Byte τότε η (παράμετρος1) **πολλαπλασιάζει το περιεχόμενο του AX και το αποτέλεσμα αποθηκεύεται στον DX:AX.** Ο πολλαπλασιασμός θεωρείται μη προσημασμένος.

Παράδειγμα:

- Αν AX=10h και CX=1Fh μετά την εντολή **MUL CX** θα γίνει ο πολλαπλασιασμός του CX με τον AX και το αποτέλεσμα θα μπει στον DX:AX. Δηλαδή, ο DX=01h, AX=F0h.

Οι προσημασμένοι και οι μη προσημασμένοι αριθμοί (1/2)

- Ένα εύρος bit, μπορεί να χρησιμοποιηθεί για να αναπαραστήσει ένα συγκεκριμένο πλήθος αριθμών. Για παράδειγμα τα 8bit μπορεί να αναπαραστήσουν 256 διαφορετικές τιμές.
- Αν θεωρήσουμε ότι τα bit, αναπαριστούν μη προσημασμένες τιμές (χωρίς πρόσημο), τότε εύκολα βρίσκεται ο αριθμός που αναπαριστούν αφού είναι ο ίδιος με τη δυαδική αναπαράσταση. Για παράδειγμα, ο δυαδικός αριθμός 00110011 είναι ο δεκαδικός αριθμός 51, ενώ ο δυαδικός αριθμός 11001100 είναι ο δεκαδικός αριθμός 204.
  - Οι αριθμοί αυτοί δεν έχουν πρόσημο και είναι λάθος να τους θεωρούμε θετικούς.

Οι προσημασμένοι και οι μη προσημασμένοι αριθμοί (2/2)

- Αν θεωρήσουμε ότι τα bit, αναπαριστούν προσημασμένες τιμές *(τιμές που έχουν είτε θετικό, είτε αρνητικό πρόσημο),* τότε το πλήθος των διαφορετικών αναπαραστάσεων χωρίζεται “σχεδόν” στη μέση, ως εξής:
  - Το πρώτο τμήμα (από 0 έως το μισό περίπου του πλήθους) αντιστοιχεί στις θετικές τιμές. Η θετική τιμή αυτών είναι ίδια με την αντίστοιχη τιμή που θα υπήρχε αν θεωρούσαμε τον αριθμό χωρίς πρόσημο. Για παράδειγμα, αν έχουμε 8 bit, ο δυαδικός αριθμός 00110011 είναι ο δεκαδικός αριθμός 51, αφού ανήκει στο πρώτο τμήμα *(από 0 – 00000000b έως 127 – 01111111b).*
  - Το δεύτερο τμήμα *(από το μισό περίπου του πλήθους έως τη μέγιστη τιμή)* αντιστοιχεί στις αρνητικές τιμές. Η αρνητική τιμή αυτών δεν είναι η ίδια με την αντίστοιχη τιμή χωρίς πρόσημο. Για να βρεθεί η τιμή, θα πρέπει να υπολογιστεί το συμπλήρωμα ως προς 2.
- Ο χωρισμός είναι “σχεδόν” στη μέση, επειδή υπάρχει το μηδέν. Ως εκ τούτου οι αρνητικοί αριθμοί έχουν ένα περισσότερο μέλος *(π.χ. για 8bit είναι από -128 έως 0, ενώ οι θετικοί από 0 έως +127).*

Υπολογισμός συμπληρώματος ως προς 2

- Στην περίπτωση που ο αριθμός ανήκει στο 2ο μισό του πλήθους, δηλαδή έχει ως πιο σημαντικό bit (τέρμα αριστερά, most significant bit, **MSB**), το '1' και το θεωρήσουμε ως προσημασμένο, τότε είναι αρνητικός αριθμός.
- Ο υπολογισμός του αρνητικού αριθμού γίνεται με το συμπλήρωμα ως προς 2.
- Η διαδικασία μπορεί να γίνει με 2 τρόπους:
  - Είτε αντιστροφή όλων των bit, πρόσθεση 1 και τοποθέτηση προσήμου.π.χ. ο 11001100b (=NOT=>) 00110011 (=ADD + 1) 00110100, δηλαδή είναι ο 52 και επειδή το MSB ήταν 1 γίνεται -52.
  - Είτε αφαίρεση του αριθμού (αν το υπολογίσουμε ως μη προσημασμένο) από το πλήθος των στοιχείων που αντιστοιχούν στα συγκεκριμένα bit. Για παράδειγμα τα 8bit έχουν 256 διαφορετικές αναπαραστάσεις. Οπότε το 11001100b έχει τη μη προσημασμένη τιμή στο δεκαδικό σύστημα 204. Άρα γίνεται η πράξη 256 – 204 = 52 και επειδή ο αριθμός μας είχε MSB 1, τότε τοποθετείται το αρνητικό πρόσημο και έχουμε -52.

Οι θετικοί αριθμοί δεν υπολογίζονται με συμπλήρωμα ως προς 2

- Ένας θετικός αριθμός, δηλαδή που έχει το MSB=0 έχει ίδια τιμή είτε είναι προσημασμένος, είτε είναι χωρίς πρόσημο.
  - Για παράδειγμα, ο δυαδικός αριθμός 00110011 είναι ο δεκαδικός αριθμός 51 ως μη προσημασμένος.
  - Επίσης, ο δυαδικός αριθμός 00110011 είναι ο δεκαδικός +51 ως προσημασμένος.
- (**ΠΡΟΣΟΧΗ**) Ένα λάθος που εμφανίζεται κάποιες φορές, είναι να γίνεται ο υπολογισμός του συμπληρώματος ως προς 2 σε θετικούς αριθμούς, προκειμένου να βρεθεί η προσημασμένη δεκαδική τιμή του.

Η εντολή IMUL (1/2)

- Εκτός από την εντολή πολλαπλασιασμού MUL η οποία συνδέεται μόνο με **μη προσημασμένους αριθμούς**, υπάρχει και η εντολή πολλαπλασιασμού **προσημασμένων αριθμών** IMUL.

- Η σύνταξη των εντολών είναι η ίδια, και το αποτέλεσμα αποθηκεύεται στους ίδιους καταχωρητές.

- Η διαφορά είναι στο αποτέλεσμα της πράξης.

- Αν και οι δυο αριθμοί που πολλαπλασιάζονται είναι θετικοί, δηλαδή έχουν το MSB=0, τότε το αποτέλεσμα της IMUL είναι το ίδιο με το αποτέλεσμα της MUL στις ίδιες τιμές.

Η εντολή IMUL (2/2)

- Αν τουλάχιστον ένας αριθμός είναι αρνητικός, τότε το αποτέλεσμα είναι διαφορετικό.
- Υπολογίζεται η τιμή του αρνητικού αριθμού, το συμπλήρωμα ως προς 2, και στη συνέχεια γίνεται ο υπολογισμός.
  - Αν το αποτέλεσμα που προκύπτει είναι θετικό (γιατί πολλαπλασιάστηκε ένας αρνητικός αριθμός με έναν αρνητικό αριθμό), τότε δεν γίνεται κάποια άλλη αλλαγή και αποθηκεύεται στους αντίστοιχους καταχωρητές.
  - Αν το αποτέλεσμα που προκύπτει είναι αρνητικό (γιατί πολλαπλασιάστηκε ένας θετικός αριθμός με έναν αρνητικό αριθμό), τότε υπολογίζεται το συμπλήρωμα ως προς 2 του αποτελέσματος, και αυτό αποθηκεύεται στους αντίστοιχους καταχωρητές.

Παράδειγμα της IMUL για θετικούς αριθμούς

- Αν AH=02Fh, και AL=01Bh, και έχουμε IMUL AH, τότε θα γίνει η 8bit πράξη πολλαπλασιασμού AH*AL και το αποτέλεσμα θα αποθηκευτεί στον καταχωρητή AX (δες προηγούμενες διαφάνειες με τη σύνταξη της MUL).
- Χρησιμοποιείται η IMUL, οπότε οι αριθμοί χρησιμοποιούνται ως προσημασμένοι:
  - το AH=02F είναι το 8bit 00101111b (+47).
  - το AL=01B είναι το 8bit 00011011b (+27).
  - Οι αριθμοί είναι θετικοί, άρα και το αποτέλεσμα θετικό.
  - 02Fh * 01Bh = 04F5h και αυτό αποθηκεύεται στον AX.
  - Αν αντί για προσημασμένο **IMUL AH** είχαμε το απροσήμαστο **MUL AH** θα **είχαμε το ίδιο αποτέλεσμα.**

Παράδειγμα της IMUL για αρνητικό και θετικό αριθμό

- Αν AH=AFh, και AL=01Bh, και έχουμε IMUL AH, τότε θα γίνει η 8bit πράξη πολλαπλασιασμού AH*AL και το αποτέλεσμα θα αποθηκευτεί στον καταχωρητή AX (δες προηγούμενες διαφάνειες με τη σύνταξη της MUL).
- Χρησιμοποιείται η IMUL, οπότε οι αριθμοί χρησιμοποιούνται ως προσημασμένοι:
  - το AH=0AFh είναι το 8bit 10101111b, δηλαδή αρνητικός και έχει τιμή 100h-AFh= 256 - 175 = 51h (Οπότε το 0AFh είναι ο -81 ή -51h),
  - το AL=01B είναι το 8bit 00011011b (+27).
  - Ένας αριθμός είναι αρνητικός, το αποτέλεσμα θα είναι αρνητικό, άρα μετά το τέλος της πράξης θα υπολογιστεί το συμπλήρωμα ως προς 2.
  - 051h * 01Bh = 88Bh (στο δεκαδικό (81) * (27) = 2187).
  - Το 88Bh πρέπει να γίνει αρνητικός. Αυτό έχει συμπλήρωμα ως προς 2 για 2Byte: 10000h – 88Bh = F775, και αυτή η τιμή θα αποθηκευτεί στο AX.
  - Αν αντί για προσημασμένο **IMUL AH** είχαμε το απροσήμαστο **MUL AH** θα είχαμε το αποτέλεσμα AFh * 1Bh = 1275h.

Παράδειγμα της IMUL για δυο αρνητικούς αριθμούς

- Αν AH=AFh, και AL=0B1h, και έχουμε IMUL AH, τότε θα γίνει η 8bit πράξη πολλαπλασιασμού AH*AL και το αποτέλεσμα θα αποθηκευτεί στον καταχωρητή AX (δες προηγούμενες διαφάνειες με τη σύνταξη της MUL).
- Χρησιμοποιείται η IMUL, οπότε οι αριθμοί χρησιμοποιούνται ως προσημασμένοι:
  - το AH=0AFh είναι το 8bit 10101111b, δηλαδή αρνητικός και έχει τιμή 100h-AFh= 256 - 175 = 51h (Οπότε το 0AFh είναι ο -81 ή -51h),
  - το AL=0B1 είναι το 8bit 10110001b, δηλαδή αρνητικός και έχει τιμή 100h-0B1=256-177 = 4Fh (Οπότε το 0B1h είναι ο -79 ή -4Fh).
  - Δυο αριθμοί είναι αρνητικοί, το αποτέλεσμα θα είναι θετικό, άρα δε θα χρειαστεί κάποια άλλη αλλαγή.
  - 051h * 04Fh = 018FFh (στο δεκαδικό (81) * (79) = 6399).
  - Επειδή πολλαπλασιάζουμε δυο αρνητικοί αριθμούς, το αποτέλεσμα είναι θετικός αριθμός, οπότε θα αποθηκευτεί η τιμή 18FFh στο AX.
  - Αν αντί για προσημασμένο **IMUL AH** είχαμε το απροσήμαστο **MUL AH** θα είχαμε το αποτέλεσμα AFh * B1h = 78FFh.

Ο συμβολισμός DX:AX

- Το DX:AX είναι ένας συμβολισμός που δείχνει ότι έχουμε ένα αποτέλεσμα 4Byte ως εξής: Τα 2 πιο σημαντικά Byte βρίσκονται στον DX και τα 2 λιγότερο σημαντικά στον AX.
- Αν θέλουμε να τα μεταφέρουμε σε μια θέση μνήμης που έχει δηλωθεί ως **dd** (4 Bytes), αυτό θα πρέπει να γίνει προσέχοντας το endianess (little endian). Παράδειγμα:
  - Η θέση μνήμης έχει δηλωθεί ως dd (π.χ. **res dd 0**).
  - Στις θέσεις res[0] και res[1] θα υπάρχει το AX και στις θέσεις res[2] και res[3] θα υπάρχει το DX.
  - Αυτό γίνεται ως:

```
mov res[2],DX
mov res[0],AX
```

Παράδειγμα

```
mov  ax,3Ah
mov  cx,0FFAh
mul  cx
mov  res[2],dx
mov  res[0],ax
```

Εντολή Διαίρεσης 8bit DIV

- Η εντολή DIV χρησιμοποιείται για διαίρεση μη προσημασμένων αριθμών και συντάσσεται:

```
DIV (παράμετρος1)
```

- Όπου (παράμετρος1) είναι ένας τελεστής είτε καταχωρητής γενικής χρήσης είτε μεταβλητή μνήμης.
- Αν ο τελεστής είναι ενός Byte τότε η προέλευση διαιρεί το περιεχόμενο του AX και το **μεν πηλίκο της διαίρεσης αποθηκεύεται στον AL το δε υπόλοιπο στον AH**.

Παράδειγμα:

Αν AX=10 και CL=3 μετά την εντολή **DIV CL** θα γίνει η διαίρεση AX/CL και το πηλίκο θα μπει στον AL, το υπόλοιπο στον AH. Δηλαδή, ο AL=3, AH=1.

Εντολή Διαίρεσης 16bit DIV

- Η εντολή DIV χρησιμοποιείται για διαίρεση μη προσημασμένων αριθμών και συντάσσεται:

```
DIV (παράμετρος1)
```

- Όπου (παράμετρος1) είναι ένας τελεστής είτε καταχωρητής γενικής χρήσης είτε μεταβλητή μνήμης.
- Αν ο τελεστής είναι δύο Byte τότε η προέλευση διαιρεί το περιεχόμενο της DX:AX και το **μεν πηλίκο της διαίρεσης αποθηκεύεται στον AX το δε υπόλοιπο στον DX**.

Παράδειγμα:

Αν DX=0Fh, AX=10h και BX=30h μετά την εντολή **DIV BX** θα γίνει η διαίρεση DX:AX/BX (δλδ 0F10h/30h) και το πηλίκο θα μπει στον AX, το υπόλοιπο στον DX. Δηλαδή, ο AX=50h, DX=0Fh.

Προσημασμένη Διαίρεση / Πολλαπλασιασμός

- Για προσημασμένο πολλαπλασιασμό χρησιμοποιούμε την εντολή **IMUL** (ίδια σύνταξη).
- Για προσημασμένη διαίρεση χρησιμοποιούμε την εντολή **IDIV** (ίδια σύνταξη).

Εμφάνιση Αλφαριθμητικού

- Δηλώνουμε στο τμήμα δεδομένων το αλφαριθμητικό που θέλουμε να εμφανίσουμε ως εξής:

```
message db 'The result is:', 10, 13, '$'
```

- Προσοχή: Ο τελευταίος χαρακτήρας του μηνύματος είναι το $.
- Στο τμήμα του κώδικα μεταφέρουμε στον καταχωρητή DX την ενεργό διεύθυνση του αλφαριθμητικού που θέλουμε να τυπώσουμε.

```
LEA DX, message ή MOV DX, offset message
```

- Στο τμήμα του κώδικα καλούμε την 9η κλήση του DOS αμέσως μετά την προηγούμενη εντολή.

```
MOV AH, 09h
INT 21h
```

Εμφάνιση μονοψήφιου αριθμού

- Στο τμήμα του κώδικα μεταφέρουμε στον καταχωρητή DL το περιεχόμενο του τελεστής είτε είναι καταχωρητής γενικής χρήσης είτε μεταβλητή μνήμης.

```
mov dl,num1
```

- Στο τμήμα του κώδικα μετατρέπουμε τη δυαδική τιμή του αριθμού σε τιμή που αντιστοιχεί στον χαρακτήρα του πίνακα ASCII (μετατροπή καθαρού αριθμού => ASCII).

```
add dl,30h
```

- Στο τμήμα του κώδικα καλούμε την 2η κλήση του DOS αμέσως μετά για εμφάνιση χαρακτήρα.

```
mov ah,02h
int 21h
```

Εμφάνιση διψήφιου αριθμού (1/2)

- Ο προηγούμενος τρόπος εμφάνισης (`mov ah,02h` και `int 21h`) εκτυπώνει στην οθόνη μόνο ένα χαρακτήρα.
- Αν ο αριθμός μας αποτελείται από πολλά ψηφία, τότε θα πρέπει να τα απομονώσουμε.
- Η απομόνωση επιτυγχάνεται με διάφορους τρόπους: είτε με διαίρεση, είτε με λογική πράξη με μάσκα, είτε με ολίσθηση.
- Ο πιο εύκολος τρόπος είναι η διαίρεση:
  - Εκτιμούμε πόσα ψηφία το μέγιστο στο δεκαδικό σύστημα μπορεί να δημιουργηθούν.
  - Διαιρούμε διαδοχικά με το 10, και κάθε φορά κρατάμε το πηλίκο, το οποίο το τοποθετούμε σε μια θέση μνήμης.
  - Μεταφέρουμε το υπόλοιπο στο διαιρετέο και επαναλαμβάνουμε τη διαίρεση με το 10.
  - Διαιρούμε τόσες φορές με το 10, όσα τα ψηφία του αριθμού.

Εμφάνιση διψήφιου αριθμού (2/2)

- Αφού υπολογίσουμε τα ψηφία και τα αποθηκεύσουμε σε διαφορετικές θέσεις μνήμης, αρχίζει η εκτύπωση.
- Μεταφέρουμε κάθε φορά το πιο σημαντικό ψηφίο στον DL και το εκτυπώνουμε με τη 02 κλήση του int 21h.

Για παράδειγμα:

```
mov dl,ekatontada
add dl,30h
mov ah,02
int 21h
mov dl,dekada
add dl,30h
mov ah,02 ....
```

Αν δε γνωρίζουμε το μέγιστο αριθμό ψηφίων, τότε μπορούμε να κάνουμε διαδοχικές διαιρέσεις με το 10, έως ότου το πηλίκο γίνει 0 (δες επόμενη διάλεξη για εντολή σύγκρισης cmp), και εκτυπώνουμε όλα τα πηλίκα που έχουμε βρει.

Εισαγωγή χαρακτήρα
(με ταυτόχρονη εμφάνιση)

- Η εισαγωγή ενός χαρακτήρα από το πληκτρολόγιο με ταυτόχρονη εμφάνιση στην οθόνη, γίνεται με την 1η κλήση του DOS.
- Στο τμήμα του κώδικα καλούμε την 1η κλήση του DOS:

```
mov ah,01h
    int 21h
```

- Η ASCII τιμή του χαρακτήρα που θα διαβαστεί από το πληκτρολόγιο θα τοποθετηθεί στον καταχωρητή AL.
- Αν θέλουμε να εισάγουμε αριθμό θα πρέπει να μετατρέψουμε την ASCII τιμή στην καθαρή τιμή ως εξής:

```
SUB AL,30h
```
(μόνο για εισαγωγή αριθμού).

Εισαγωγή χαρακτήρα *(χωρίς ταυτόχρονη εμφάνιση)*

- Η εισαγωγή ενός χαρακτήρα από το πληκτρολόγιο χωρίς εμφάνιση στην οθόνη, γίνεται με την 8η κλήση του DOS.
- Στο τμήμα του κώδικα καλούμε την 1η κλήση του DOS:

```
mov ah,08h
    int 21h
```

- Η ASCII τιμή του χαρακτήρα που θα διαβαστεί από το πληκτρολόγιο θα τοποθετηθεί στον καταχωρητή AL.
- Αν θέλουμε να εισάγουμε αριθμό θα πρέπει να μετατρέψουμε την ASCII τιμή στην καθαρή τιμή ως εξής:

```
SUB AL,30h
```

(μόνο για εισαγωγή αριθμού).

Σχόλια στην assembly

- Ο χαρακτήρας που υποδηλώνει ότι από εκείνο το σημείο έως το τέλος της γραμμής είναι σχόλιο είναι

ο χαρακτήρας ;

- Μπορεί να βρίσκεται σε οποιοδήποτε σημείο της γραμμής.

Παραδείγματα

```
; Program  hello world
 MOV AH,02 ;etoimasia gia thn 02  klhsh
```

Μεγέθη μεταβλητών

Μπορούμε να δηλώνουμε το μέγεθος που θα καταλαμβάνει κάθε στοιχείο μιας μεταβλητής στο τμήμα δεδομένων ως εξής:

- DB (define Byte) κάθε στοιχείο είναι 1Byte.
- DW (define word) κάθε στοιχείο είναι 2Byte.
- DD (define double Byte) κάθε στοιχείο είναι 4Byte.

Παραδείγματα:

```
msg db "H","E","LLO","$"        (6Byte)
buf1 dw 1                       (2*1 = 2Byte)
buf2      dd  1,4,5,10          (4*4 = 16 Byte)
```

Ο ειδικός καταχωρητής IP

- Είναι 2Byte.
- Κάθε φορά δείχνει **την επόμενη** προς εκτέλεση εντολή.
- Κάθε εντολή σε ASSEMBLY μετατρέπεται σε έναν αριθμό από bytes. Για παράδειγμα:

```
Loc: Machine Code          Source
0028: B4 09                MOV AH,9
002A: BA 00 00          LEA DX,MSG
```

- Αν Το IP=28h τότε στον επόμενο κύκλο θα εκτελεστεί η εντολή στο LOC=028h. Η εντολή είναι 2 Byte, οπότε ο IP θα αυξηθεί κατά 2 μόλις αρχίσει η εκτέλεση αυτής. Δηλαδή 28h+2=2Ah.
- Το IP=2Ah τότε στον επόμενο κύκλο θα εκτελεστεί η εντολή στο LOC=02Ah. Η εντολή είναι 3 Byte, οπότε ο IP θα αυξηθεί κατά 3 μόλις αρχίσει η εκτέλεση αυτής. Δηλαδή 2Ah+3=2Dh.

Πότε αλλάζουν οι τιμές στους γενικούς καταχωρητές

- Συνήθως ένας καταχωρητής διατηρεί την τιμή που του έχουμε δώσει έως να του δώσουμε άλλη τιμή με το MOV.
- Όμως υπάρχουν περιπτώσεις που ένας καταχωρητής αλλάζει τιμή και ίσως δεν το αντιλαμβανόμαστε.
- Πότε αλλάζει ένας καταχωρητής τιμή μπορούμε να το δούμε μέσω του emulator.
- Γενικές περιπτώσεις:
  - Οι καταχωρητές AX (AH,AL) αλλάζουν ύστερα από την κλήση INT 21h.
  - Μερικές κλήσεις int 21h επιστρέφουν μια τιμή σε συγκεκριμένους καταχωρητές.
  - Οι αριθμητικές πράξεις αλλάζουν τιμές σε συγκεκριμένους καταχωρητές.

Μετατροπή 8bit σε 16bit

- Ένας καταχωρητής 8bit δε μπορεί να μεταφερθεί σε έναν καταχωρητή 16bit.
  - π.χ. Απαγορεύεται το MOV AX,DL.
- Όμως γνωρίζουμε ότι ένας καταχωρητής 16bit αποτελείται από το Low Byte και το High Byte.
- Αν θέλουμε λοιπόν να κάνουμε μετατροπή θα πρέπει να μεταφέρουμε την τιμή μας στο Low Byte και να μηδενίσουμε το High Byte. Ο αριθμός θα είναι ο ίδιος, μόνο που θα τον έχουμε μετατρέψει σε 16bit.
- Παράδειγμα (μεταφορά DL στον AX):

```
MOV AL,DL
MOV AH,0
```

Μετατροπή 16bit σε 8bit

- Ένας καταχωρητής 16bit δε μπορεί να μεταφερθεί σε έναν καταχωρητή 8bit.
  - π.χ. Απαγορεύεται το MOV DL,AX
- Όμως γνωρίζουμε ότι ένας καταχωρητής 16bit αποτελείται από το Low Byte και το High Byte
- Επίσης γνωρίζουμε ότι αν ένας αριθμός 16bit έχει τιμή από 0 έως 255 (μη προσημασμένος), τότε χρησιμοποιείται μόνο το Low Byte, ενώ το High Byte έχει τιμή 0.
- Για αυτό το λόγο, αν είμαστε σίγουροι ότι ο καταχωρητής έχει τιμή από 0 έως 255, μπορούμε να χρησιμοποιήσουμε από εκείνο το σημείο μόνο το Low Byte

  Παράδειγμα (θέλω να μεταφέρω τον AX (0-255) στο DL):

```
MOV DL,AL
```

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης