ΤΕΙ ΣΕΡΡΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΣΗΜΕΙΩΣΕΙΣ ΕΡΓΑΣΤΗΡΙΟΥ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Εκπαιδευτικό Σύστημα BGC-8088

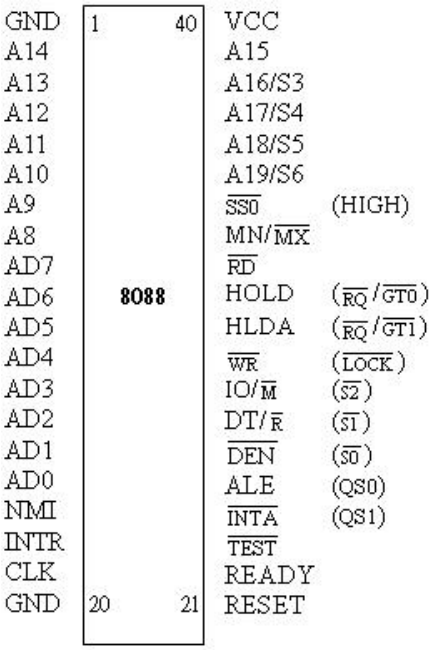
Δρ. Σπύρος Α. Καζαρλής
Επίκουρος Καθηγητής

Σέρρες, Σεπτέμβριος 2004

Κεφάλαιο 1 : Τα Χαρακτηριστικά του BGC-8088

Ο Εκπαιδευτικός Αναπτυξιακός Υπολογιστής BGC-8088 MicroEngineer V.3 είναι ένα σύστημα βασισμένο στον μικροεπεξεργαστή INTEL 8088-2, που αποτελεί την έκδοση διπλού χρονισμού (λειτουργεί στα 4.77 MHz και τα 8 MHz) του απλούστερου INTEL 8088 (4.77 MHz) που ήταν ο επεξεργαστής του πρώτου IBM PC, στον οποίο βασίζονται οι σύγχρονοι υπολογιστές με επεξεργαστές της σειράς x86 και Pentium της INTEL
Ο BGC-8088 έχει 32K RAM και 16K ROM που μπορούν να επεκταθούν με άλλα 16 K ROM με προγράμματα του χρήστη.
Η επικοινωνία με τον χρήστη γίνεται μέσω ενός πληκτρολογίου 56 πλήκτρων σε διάταξη QWERTY που περιλαμβάνει όλους τους εκτυπώσιμους χαρακτήρες και σύμβολα, καθώς και πλήκτρα λειτουργιών και ελέγχου, και μίας LCD οθόνης 2 γραμμών και 40 χαρακτήρων.
Διαθέτει 2 υποδοχές ISA των 62 pins για σύνδεση καρτών επέκτασης ISA των 8 bit.
Διαθέτει μία σειριακή θύρα RS232-C και μία παράλληλη θύρα.
Διαθέτει επίσης έναν Προγραμματιζόμενο Χρονιστή (Prorammable Interval Timer – PIT) 8254 της INTEL που χρησιμοποιείται για παραγωγή σημάτων χρονισμού και ως μετρητής χρόνου (counter), ένα chip Προγραμματιζόμενης Διεπαφής Περιφερειακών (Programmable Peripheral Interface – PPI) 8255 της INTEL που παρέχει 3 θύρες Εισόδου / Εξόδου των 8-bit πλήρως προγραμματιζόμενες, και έναν Προγραμματιζόμενο Ελεγκτή Διακοπών (Programmable Interrupt Controller – PIC) 8259A της INTEL που παρέχει 8 γραμμές διακοπών interrupts.
Τέλος διαθέτει ειδική θύρα σύνδεσης εκπαιδευτικών πλακετών των 50 pin όπου συνδέονται όλες οι γραμμές του data bus, του address bus και οι γραμμές ελέγχου (control bus) του BGC-8088.

Ο επεξεργαστής
Ο 8088 είναι ένας επεξεργαστής με εσωτερική αρχιτεκτονική των 16 bit (χρησιμοποιεί καταχωρητές των 16 bit) αλλά έχει Δίαυλο Δεδομένων (Data Bus) των 8 bit για επικοινωνία με την μνήμη και τις περιφερειακές συσκευές. Επίσης έχει Δίαυλο Διευθύνσεων (Address Bus) των 20 bit, οπότε και μπορεί να απευθυνθεί σε 2^{20} = 1048576 = 1MB μνήμη (RAM και ROM).



Σχήμα 1. Ο Επεξεργαστής 8088 και η σημασία των (40) pins του

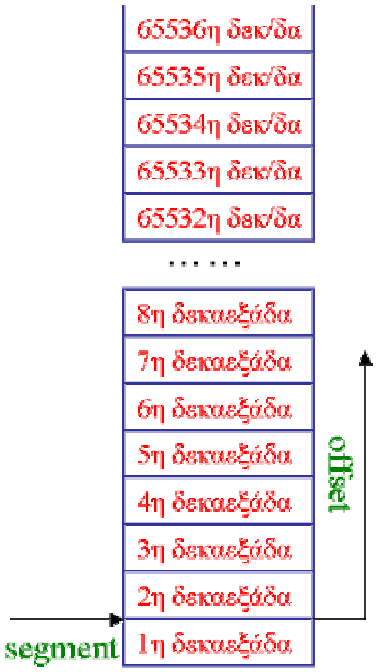
Έχει συχνότητες λειτουργίας τα 4.77 MHz και τα 8 MHz. Στον BGC-8088 ο επεξεργαστής λειτουργεί στα 4.77 MHz.
Διαθέτει 14 καταχωρητές των 16 bit για γενικές και ειδικές λειτουργίες, μερικοί από τους οποίους μπορούν να χωρισθούν σε δύο καταχωρητές των 8-bit.
Διαθέτει σετ εντολών με 90 συνολικά εντολές που έχουν 24 συνολικά διαφορετικούς τρόπους σύνταξης (addressing modes). Περιλαμβάνουν αριθμητικές πράξεις στα 8 ή 16 bit, με πρόσημο ή χωρίς, συμπεριλαμβανομένου του πολλαπλασιασμού και της διαίρεσης.

Οι καταχωρητές του 8088
Διαθέτει 14 καταχωρητές των 16 bit για γενικές και ειδικές λειτουργίες, μερικοί από τους οποίους μπορούν να χωρισθούν σε δύο καταχωρητές των 8-bit. Οι καταχωρητές του 8088 είναι οι ακόλουθοι :

Ονομα Καταχωρητή	Δομή		Χρήση
	16 bits 8 bits 8 bits		
AX	AH	AL	Accumulator (Συσσωρευτής)
BX	BH	BL	Base (Καταχωρητής Βάσης)
CX	CH	CL	Counter (Μετρητής)
DX	DH	DL	Data (Δεδομένα)

BP		Base Pointer (Δείκτης Βάσης)
SI		Source Index (Δείκτης Πηγής)
DI		Destination Index (Δείκτης Κατεύθυνσης)
SP		Stack Pointer (Δείκτης Στοίβας)
IP		Instruction Pointer (Δείκτης Εντολής)
CS		Code Segment (Τμήμα Κώδικα)
DS		Data Segment (Τμήμα Δεδομένων)
SS		Stack Segment (Τμήμα Στοίβας)
ES		Extra Segment (Εξτρα Τμήμα)
FG		FlaG Register (Καταχωρητής Σημαιών)

Οι **καταχωρητές AX, BX, CX, DX** μπορούν να θεωρηθούν και ως διπλοί καταχωρητές των 2x8bit. Για παράδειγμα ο χρήστης μπορεί να προσπελάσει τον καταχωρητή AX και μέσω των δύο 8-bit καταχωρητών AH και AL. Ο AH αντιστοιχεί στο υψηλότερης τάξης τμήμα του AX (δηλαδή τα bits 8-15) ενώ ο AL αντιστοιχεί στο χαμηλότερης τάξης τμήμα του AX (δηλαδή τα bits 0-7). Το ίδιο συμβαίνει αντίστοιχα και για τους καταχωρητές BX, CX και DX. Οι καταχωρητές αυτοί είναι γενικής χρήσης, δηλαδή μπορούν να μεταφέρουν δεδομένα προς και από τη μνήμη, να εκτελέσουν αριθμητικές πράξεις κλ.π. Ο Συσσωρευτής όμως (AX) είναι πιο σημαντικός από τους υπόλοιπους καθώς συγκεκριμένες εντολές και τρόποι προσπέλασης μνήμης εκτελούνται μόνο με αυτόν, όπως ο πολλαπλασιασμός και η διαίρεση.
Οι καταχωρητές **CS, DS, SS, ES είναι καταχωρητές τμημάτων** (segment registers). Η σημασία τους είναι η ακόλουθη :
Ο επεξεργαστής 8088 μπορεί να προσπελάσει 1MB μνήμης έχοντας address bus με 20 γραμμές διευθύνσεων (2^{20}=1048576). Για συμβατότητα όμως με τον προηγούμενο επξεργαστή της INTEL, τον 8080, κατασκευάστηκε με καταχωρητές διευθύνσεων των 16 bits (2^{16}=65536). Έτσι επινοήθηκε η μέθοδος των παραγράφων (paragraphs), σύμφωνα με την οποία, κάθε διεύθυνση των 16 bits δεν αναφέρεται σε απόλυτη διεύθυνση αλλά στην αρχή μίας δεκαεξάδας από bytes (π.χ. η διεύθυνση 0001_{16} αναφέρεται στην αρχή της πρώτης δεκαεξάδας μνήμης : 00010_{16}=16, η 0002_{16} στην αρχή της δεύτερης δεκαεξάδας μνήμης : 00020_{16}=32 κ.λ.π.). Η τελευταία δεκαεξάδα είναι η $FFFF_{16}$ που αναφέρεται στην διεύθυνση $FFFF0_{16}$ = 1048560.
Για την προσπέλαση οποιασδήποτε διεύθυνσης μνήμης εκτός από την διεύθυνση παραγράφου (segment) χρειάζεται και μία διεύθυνση μετατόπισης (offset). Έτσι οι διευθύνσεις σχηματίζονται ως segment:offset. Η μετατόπιση (offset) θα αρκούσε να είναι των 4 bits (2^4=16) έτσι ώστε να προσδιορίζει το byte της δεκαεξάδας. Για λόγους ομοιογένειας όμως το offset είναι επίσης των 16 bits και μπορεί να δώσει 2^{16}=65536 διαφορετικές διευθύνσεις πάνω από την αρχή της παραγράφου, και όχι μόνο 16. Έτσι έχοντας το segment σταθερό και αλλάζοντας το offset, προφανώς «μπαίνουμε» και στον χώρο των επόμενων δεκαεξάδων της μνήμης. Από τα παραπάνω είναι κατανοητό ότι δεν είναι μονοσήμαντη η διευθυνσιοδότηση μνήμης με την μέθοδο segment:offset, δηλαδή για κάθε απόλυτη διεύθυνση μνήμης υπάρχουν πολλοί συνδυασμοί segment:offset που την προσδιορίζουν (π.χ. το 17ο byte μνήμης έχει διεύθυνση 0001:0001 αλλά και 0000:0011).



Για να βρούμε την απόλυτη διεύθυνση μνήμης ακολουθούμε τον απλό τύπο :

Απόλυτη διεύθυνση = segment X 16 + offset.

Έτσι οι καταχωρητες τμημάτων (CS,DS,SS,ES) φτιάχτηκαν για να καταχωρούν το κομμάτι 'segment' από την έκφραση «segment:offset» των διευθύνσεων μνήμης που σχηματίζονται στον επεξεργαστή κατά την διάρκεια της λειτουργίας του και της εκτέλεσης εντολών.

Πιο συγκεκριμένα, κάθε πρόγραμμα αποτελείται από 3 τμήματα που είναι :

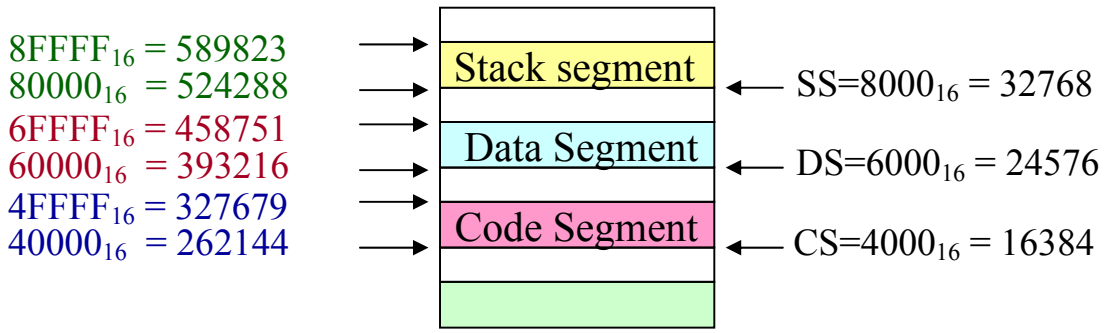
1. Το τμήμα κώδικα (code segment) όπου βρίσκεται το ίδιο το πρόγραμμα,
2. Το τμήμα δεδομένων (data segment) όπου βρίσκονται οι μεταβλητές του προγράμματος και,
3. Το τμήμα στοίβας (stack segment) που χρησιμοποιείται για την στοίβα του προγράμματος (stack) που είναι ένα τμήμα μνήμης το οποίο προσπελαύνεται από τον 8088 με δομή LIFO και χρησιμοποιείται για προσωρινή αποθήκευση δεδομένων (π.χ. προσωρινή αποθήκευση καταχωρητών, πέρασμα παραμέτρων σε υπορουτίνες, κ.λ.π)

Έτσι ο 8088 έχει έναν καταχωρητή τμήματος για κάθε ένα από τα 3 αυτά τμήματα, που «δείχνει» την δεκαεξάδα στην οποία ξεκινάει το κάθε τμήμα. Οι καταχωρητές αυτοί είναι :

1. Ο CS (code segment register) που δείχνει στο τμήμα κώδικα,
2. Ο DS (data segment register) που δείχνει στο τμήμα δεδομένων, και
3. Ο SS (stack segment register) που δείχνει στο τμήμα στοίβας.

Ο τέταρτος καταχωρητής τμήματος που είναι ο ES (Extra segment register) χρησιμοποιείται ως «γενικής χρήσης» καταχωρητής τμήματος που μπορεί να «δείξει» σε οποιοδήποτε άλλο τμήμα (δεκαεξάδα) της μνήμης ώστε να προσπελάσει οποιαδήποτε διεύθυνση χωρίς να «χαλάσει» τις τιμές των CS,DS,SS που είναι αφιερωμένοι για άλλο σκοπό.

Το κάθε τμήμα (κώδικα, δεδομένων, στοίβας, έξτρα) έχοντας σταθερή διεύθυνση segment, μπορεί μέσω του μεταβλητού 16μπιτου offset να έχει μέγεθος μέχρι 64K (0000..FFFF). Αλλάζοντας τιμή στον segment register κάποιου τμήματος είναι δυνατή η επανατοποθέτηση (relocation) του τμήματος σε οποιαδήποτε δεκαεξάδα μνήμης.



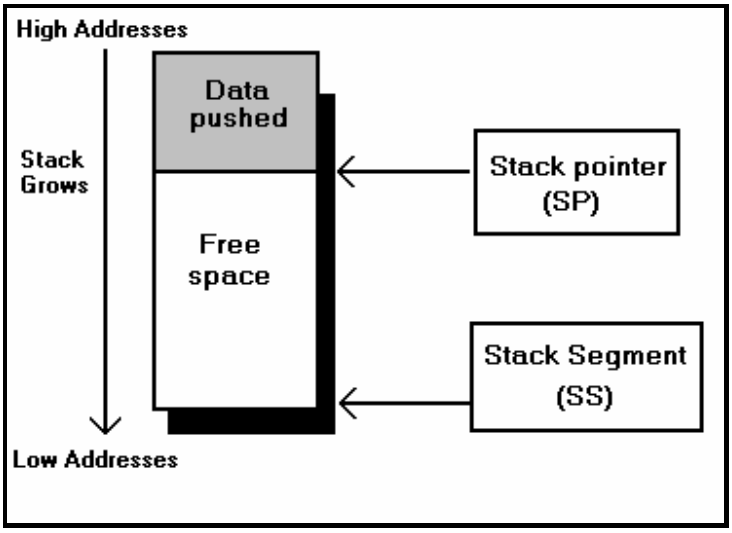
Σχήμα 2. Παράδειγμα τοποθέτησης των 3 τμημάτων ενός προγράμματος στη μνήμη και αντίστοιχες τιμές των καταχωρητών τμημάτων.

Για την αποθήκευση του "offset" κομματιού της διεύθυνσης μνήμης που είναι σε μορφή «segment:offset» χρησιμοποιούνται άλλοι καταχωρητές του 8088. Για παράδειγμα, ο καταχωρητής IP (Instruction Pointer) κρατάει το 'offset' κομμάτι της διεύθυνσης που δείχνει πάντα την επόμενη εντολή που θα εκτελεστεί. Το 'segment' κομμάτι της διεύθυνσης το κρατάει ο καταχωρητής CS (code segment register). Έτσι η διεύθυνση μνήμης που σχηματίζεται από τους καταχωρητές CS:IP είναι πάντα η διεύθυνση της επόμενης εντολής κώδικα μηχανής που θα εκτελεστεί.
Σημείωση: στον BGC-8088 οι εξ'ορισμού τιμές για τους καταχωρητές τμημάτων είναι :

Α/Α	Καταχωρητής	Αρχική Τιμή
1	CS	0100
2	DS	0100
3	SS	0040
4	ES	0100

Λαμβάνοντας υπ'όψιν ότι η αρχική τιμή του καταχωρητή IP είναι 0000, συμπεραίνουμε ότι τα προγράμματα που γράφουμε στον BGC-8088 ξεκινούν εξ'ορισμού από την διεύθυνση : 0100:0000 που μεταφράζεται σε απόλυτη διεύθυνση : $01000_{16}=4096_{10}$.
Παράδειγμα 2: Η εντολή CMPSB συγκρίνει ένα byte ενός string από την διεύθυνση DS:SI με ένα byte ενός άλλου string στην διεύθυνση ES:DI, επηρεάζοντας τις σημαίες (flags). Εδώ οι SI (Source Index) και DI (Destination Index) έχουν καταχωρημένα τα offset.

Στο σετ των καταχωρητών του 8088 υπάρχουν και ειδικοί καταχωρητές όπως :
Ο καταχωρητής IP (Instruction Pointer) που δείχνει την επόμενη εντολή που θα εκτελεστεί (μέσα στο Τμήμα Κώδικα – Code Segment)
Ο καταχωρητής SP (Stack Pointer) ο οποίος δείχνει το σημείο μέχρι το οποίο έχει γεμίσει η στοίβα (stack) μέσα στο Τμήμα Στοίβας (Stack Segment). Ο καταχωρητής ξεκινά με μεγάλη τιμή (0B3F) και όσο γεμίζει το stack αυτός μειώνεται (το stack γεμίζει από πάνω προς τα κάτω, δηλαδή από μεγάλες διευθύνσεις προς μικρότερες).



Σχήμα 3. Ο τρόπος λειτουργίας του σωρού stack.

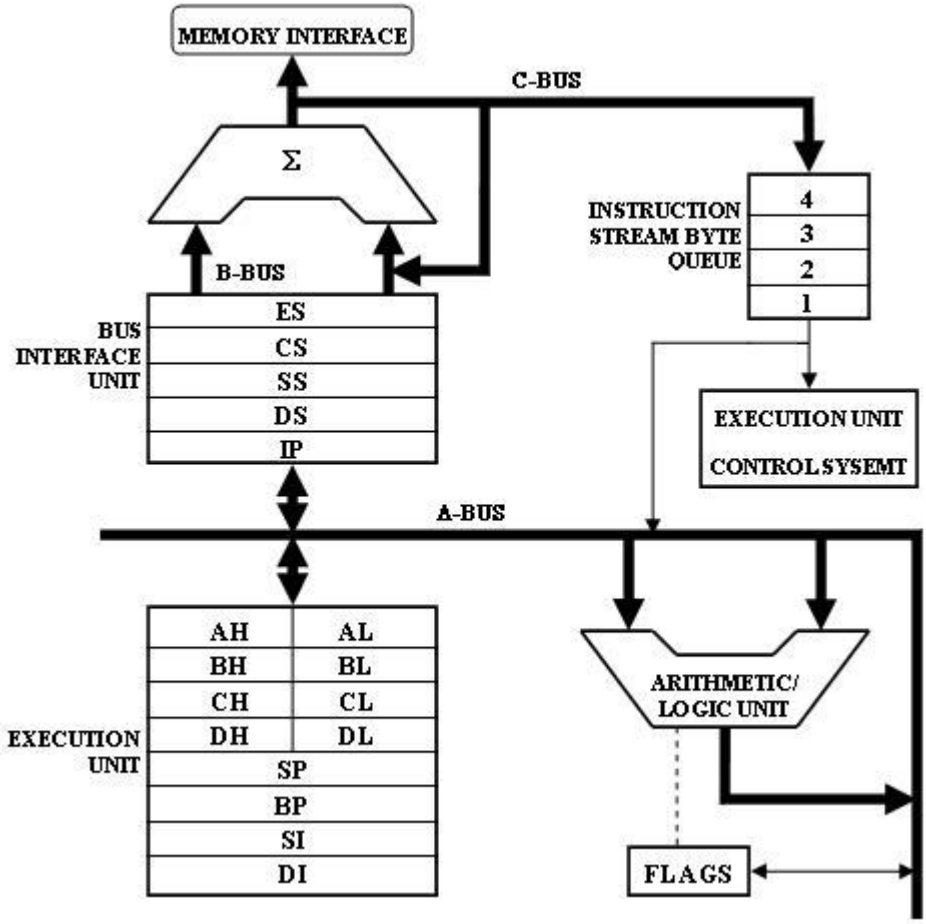
Ο καταχωρητής FG (Flag Register) ο οποίος περιέχει σημαίες (flags) που δείχνουν ή καθορίζουν την κατάσταση του επεξεργαστή. Στον καταχωρητή FG δεν έχει σημασία η τιμή του ως 16-bit δυαδικό νούμερο, αλλά το κάθε bit ξεχωριστά και ανεξάρτητα από τα υπόλοιπα, που έχει και την δική του ξεχωριστή σημασία. Η σημασία των bits του Καταχωρητή Σημαιών του 8088 είναι η εξής :

Ως γνωστόν από τα 16 bits του FG χρησιμοποιούνται μόνο τα 9 που είναι :

Bit	**Ονομασία**	**Συντόμευση**	**Εξήγηση**
0	Carry Flag	CY	Κρατούμενο, γίνεται 1 όταν αποτελέσματα πράξεων ξεπερνούν το όριο των 16 bits.
1	*Δεν χρησιμοποιείται*		
2	Parity Flag	PF	Σημαία Ισοτιμίας, γίνεται 1 όταν το αποτέλεσμα μίας πράξης είναι δυαδικός αριθμός με ζυγό αριθμό μονάδων.
3	*Δεν χρησιμοποιείται*		
4	Auxiliary Carry Flag	AF	Βοηθητικό Κρατούμενο, γίνεται 1 όταν κατά την εκτέλεση μίας πράξης μεταφέρεται κρατούμενο από το byte χαμηλής τάξης στο byte υψηλής τάξης (low nibble carry)
5	*Δεν χρησιμοποιείται*		
6	Zero Flag	ZF	Σημαία Μηδενός, γίνεται 1 όταν το αποτέλεσμα μίας πράξης π.χ. ADD σε οποιοδήποτε καταχωρητή, είναι ίσο με 0.
7	Sign Flag	SF	Σημαία Προσήμου, γίνεται 1 όταν το αποτέλεσμα μίας

			πράξης είναι αρνητικός αριθμός.
8	Trap Flag	TF	Σημαία Βηματικής Εκτέλεσης, όταν είναι 1 εκτελεί τις εντολές βήμα-βήμα για debugging.
9	Interrupt Flag	IF	Σημαία Αποδοχής Διακοπών, όταν είναι 1 επιτρέπεται η διακοπή του προγράμματος από interrupts.
10	Direction Flag	DF	Σημαία Κατεύθυνσης, όταν είναι 1, οι εντολές των string εκτελούνται από υψηλές διευθύνσεις προς χαμηλές.
11	Overflow Flag	OF	Σημαία Υπερχείλισης, γίνεται 1 όταν το αποτέλεσμα μίας πράξης ξεπερνά το όριο των προσημασμένων αριθμών δηλαδή -32768…+32767
12	*Δεν χρησιμοποιείται*		
13	*Δεν χρησιμοποιείται*		
14	*Δεν χρησιμοποιείται*		
15	*Δεν χρησιμοποιείται*		

Στο Σχήμα 3 φαίνεται ένα Λειτουργικό Διάγραμμα των καταχωρητών του 8088, καθώς και οι τρείς εσωτερικοί δίαυλοι (A-bus, B-bus, C-bus) που τους συνδέουν μεταξύ τους.

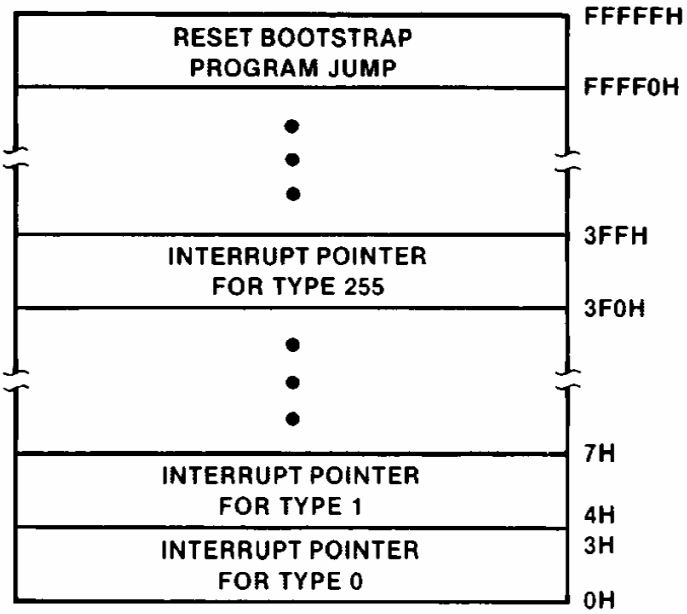


Σχήμα 3. Λειτουργικό διάγραμμα καταχωρητών και εσωτερικοί δίαυλοι καταχωρητών

Η μνήμη του BGC-8088

Ο BGC-8088 έχει 32K RAM και 16K ROM που μπορούν να επεκταθούν με άλλα 16 K ROM με προγράμματα του χρήστη. Θεωρητικά θα μπορούσε να φιλοξενήσει έως και 1MB μνήμη (RAM + ROM).

Η RAM μνήμη είναι χαρτογραφημένη στις διευθύνσεις 00000_{16} … $07FFF_{16}$ (0…32767). Οι πρώτες 4096 διευθύνσεις είναι δεσμευμένες και χρησιμοποιούνται από το πρόγραμμα MONITOR (00000_{16} … $00FFF_{16}$). Από αυτές οι πρώτες 1024 διευθύνσεις (00000_{16} … $003FF_{16}$) περιέχουν τις διευθύνσεις των ρουτινών εξυπηρέτησης των διακοπών (vectors of interrupt handler routines – interrupt vectors). Οι διακοπές μπορεί να είναι μέχρι και 256 και για κάθε ρουτίνα η διεύθυνσή της αποτελείται από 4 byte (2 segment + 2 offset). Για παράδειγμα η διεύθυνση της ρουτίνας εξυπηρέτησης της διακοπής 85 (INT 85), που όταν εκτελεστεί επανεκκινεί το πρόγραμμα MONITOR, βρίσκεται στην θέση 00214_{16} = 85_{16} x 4. Εκεί υπάρχουν διαδοχικά τα bytes 9F 01 00 FC. Τα πρώτα 2 είναι low και high bytes του offset, και τα 2 επόμενα τα low και high bytes του segment. Έτσι η πραγματική διεύθυνση της ρουτίνας εξυπηρέτησης της διακοπής 85 είναι η FC00:019F. Η διεύθυνση αυτή είναι στην ROM του μηχανήματος. Η περιοχή των interrupt vectors φαίνεται στο σχήμα 4.



Σχήμα 4. Η περιοχή των interrupt vectors του 8088 και το Jump εκκίνησης.

Οι υπόλοιπες 28672 θέσεις μνήμης είναι διαθέσιμες για προγράμματα του χρήστη (01000_{16} … $07FFF_{16}$). Για τον λόγο αυτό ο καταχωρητής CS έχει την εξ'ορισμού τιμή 0100 (0100:0000 = απόλυτη διεύθυνση 01000).
Τα 16 KB της μνήμης ROM που περιλαμβάνουν το πρόγραμμα MONITOR και τους drivers είναι χαρτογραφημένα στις διευθύνσεις $FC000_{16}$ … $FFFFF_{16}$. Τα 16K ROM που μπορεί να προσθέσει ο χρήστης (ελεύθερο slot μνήμης) χαρτογραφούνται στις διευθύνσεις $F8000_{16}$ … $FBFFF_{16}$. Κατά την εκκίνηση (εφαρμογή τροφοδοσίας) και κατά την επανεκκίνηση (reset) του μηχανήματος, ο 8088 αλλά και όλοι οι απόγονοί του στην σειρά X86 και Pentium της INTEL, ξεκινούν με την εκτέλεση της εντολής που βρίσκεται στην διεύθυνση $FFFF0_{16}$, που είναι στην ROM. Η εντολή αυτή συνήθως είναι ένα JMP στην ρουτίνα εκκίνησης του Λειτουργικού Συστήματος (του προγράμματος MONITOR στην περίπτωση του BGC-8088) που είναι σε μόνιμη μνήμη ROM (BIOS για τα PC). Η περιοχή αυτή μνήμης φαίνεται στο Σχήμα 4.
Τέλος στο παρακάτω σχήμα φαίνεται η πλήρης χαρτογράφηση της μνήμης του BGC-8088:

Περιοχή Μνήμης	**Διευθύνσεις**	**Χώρος (bytes)**
Εντολή εκκίνησης του συστήματος	FFFFF FFFF0	00010_{16} 16_{10}
ROM που περιέχει το πρόγραμμα MONITOR και τους drivers	FFFEF FC000	$03FF0_{16}$ 16368_{10}
Χώρος για ROM με προγράμματα του χρήστη (ελεύθερο slot μνήμης)	FBFFF F8000	04000_{16} 16384_{10}
Χώρος διευθύνσεων που δεν αντιστοιχεί σε chip μνήμης	F7FFF 08000	$F0000_{16}$ 983040_{10}
Ελεύθερος χώρος για προγράμματα του χρήστη	07FFF 01000	07000_{16} 28672_{10}
Περιοχή Μεταβλητών του προγράμματος MONITOR	00FFF 00400	$00C00_{16}$ 3072_{10}
Διευθύνσεις των ρουτινών εξυπηρέτησης των διακοπών – Interrupt Vectors	003FF 00000	00400_{16} 1024_{10}

Σχήμα 5. Η πλήρης χαρτογράφηση της μνήμης του BGC-8088

Οι drivers και ο τρόπος χειρισμού του hardware

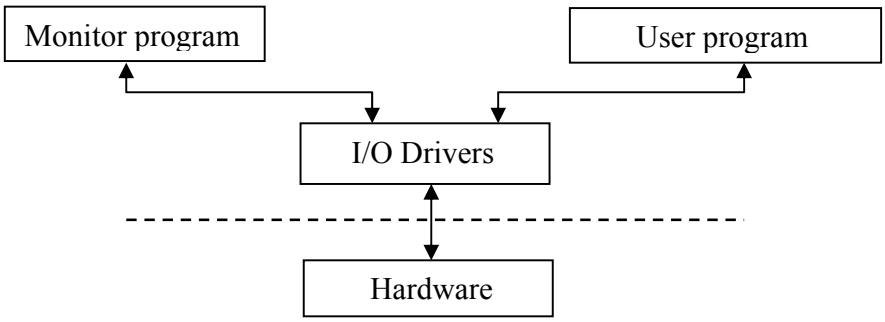
Εκτός από το πρόγραμμα MONITOR που επιτρέπει την διαχείριση του μηχανήματος και την συγγραφή και εκτέλεση προγραμμάτων, στην ROM υπάρχουν και οι οδηγοί των συσκευών Εισόδου/Εξόδου του μηχανήματος (I/O drivers). Οι οδηγοί αυτοί είναι στην ουσία υπορουτίνες που τις καλούμε με μία εντολή δακοπής (INT xx) και όχι με την εντολή CALL που είναι η κατ'εξοχήν εντολή για κλήση υπορουτινών. Καλώντας αυτές τις υπορουτίνες μπορούμε να πραγματοποιήσουμε διαδικασίες Εισόδου/Εξόδου με την αντίστοιχη συσκευή. Η κάθε υπορουτίνα δέχεται και παραμέτρους που καθορίζουν το ποιά ενέργεια Εισόδου/Εξοδου θα πραγματοποιήσουν και πώς. Οι παράμετροι δίνονται ως τιμές σε συγκεκριμένους καταχωρητές πριν από την εκτέλεση της ρουτίνας του οδηγού.

Για παράδειγμα, η εντολή INT 84 εκτελεί την υπορουτίνα του οδηγού για την οθόνη LCD. Η ρουτίνα αυτή το μόνο που μπορεί να κάνει είναι να εμφανίσει στην οθόνη (στην τρέχουσα θέση του κέρσορα) ένα χαρακτήρα και να προχωρήσει τον κέρσορα στην επόμενη θέση. Το ποιος χαρακτήρας θα εμφανιστεί καθορίζεται από την τιμή που θα βάλουμε στον καταχωρητή AL πριν την εκτέλεση της εντολής INT 84. Η τιμή αυτή αντιστοιχεί στον ASCII κωδικό του χαρακτήρα που θέλουμε να εμφανιστεί (π.χ. 'A'=41, 'B'=42, …, 'a'=61,…). Δηλαδή ο κώδκας :

```
MOV AL,41
INT 84
```

εμφανίζει στην οθόνη LCD το γράμμα 'A'.

Στο Σχήμα 6 φαίνεται ο τρόπος με τον οποίο το MONITOR και τα προγράμματα του χρήστη μπορούν να απευθυνθούν στο hardware.



Σχήμα 6. Η προσπέλαση του hardware από τα προγράμματα (Monitor και χρήστη) μέσω των οδηγών συσκευών Εισόδου/Εξόδου (I/O Drivers)

Πρέπει εδώ να σημειωθεί ότι ο χρήστης μπορεί να συντάξει προγράμματα γλώσσας μηχανής τα οποία θα χρησιμοποιούν το hardware χωρίς την χρήση των οδηγών συσκευών που παρέχονται στην μνήμη ROM. Αυτό όμως απαιτεί τόσο τη γνώση για το πώς ακριβώς μπορεί να προγραμματιστεί η κάθε συσκευή Εισόδου / Εξόδου σε χαμηλό επίπεδο, όσο και αρκετό προγραμματισμό (και debugging) από την πλευρά του χρήστη.
Οι οδηγοί συσκευών που βρίσκονται στην ROM και οι αντίστοιχοι κωδικοί διακοπών είναι οι εξής :

Οδηγός Σειριακής Θύρας RS-232 INT 80

Μέσω του οδηγού αυτού μπορούμε να κάνουμε τα ακόλουθα :

1. Να αρχικοποιήσουμε την Σειριακή Θύρα (initialize).
2. Να μεταδώσουμε δεδομένα (transmit).
3. Να λάβουμε δεδομένα (receive).
4. Να διαβάσουμε την κατάσταση της σειριακής πόρτας (read status).

Οδηγός Πληκτρολογίου INT 81

Μέσω του οδηγού αυτού μπορούμε να κάνουμε τα ακόλουθα :

1. Να διαβάσουμε ένα χαρακτήρα από το πληκτρολόγιο (read caracter).
2. Να διαβάσουμε μία ολόκληρη γραμμή από το πληκτρολόγιο (read command line)
3. Να διαβάσουμε την κατάσταση του πληκτρολογίου (read status).

Οδηγός Εκτυπωτή (Παράλληλης) INT 82

Μέσω του οδηγού αυτού μπορούμε να κάνουμε τα ακόλουθα :

1. Να αρχικοποιήσουμε τον εκτυπωτή (initialize).
2. Να τυπώσουμε δεδομένα (print data).
3. Να διαβάσουμε την κατάσταση του εκτυπωτή (read status).
4. Να προγραμματίσουμε την θύρα ελέγχου του εκτυπωτή (write control port)

Οδηγός Κάρτας Γραφικών Hercules INT 83

Μέσω του οδηγού αυτού μπορούμε να κάνουμε τα ακόλουθα :

1. Να καθορίσουμε το σχήμα του κέρσορα (set cursor type).
2. Να εμφανίσουμε έναν χαρακτήρα στην τρέχουσα θέση του κέρσορα (display character)
3. Να καθορίσουμε τα χαρακτηριστικά (attribute) ενός χαρακτήρα (set attribute)

Οδηγός Οθόνης LCD INT 84

Μέσω του οδηγού αυτού μπορούμε μόνο να εμφανίσουμε έναν οποιοδήποτε χαρακτήρα στην θέση του κέρσορα.

Επιστροφή στο MONITOR INT 85

Με την διακοπή αυτή επανεκκινείται το πρόγραμμα MONITOR. Μπορούμε να την θέτουμε ως τελευταία εντολή ενός προγράμματος. Μηδενίζει τις τιμές των καταχωρητών, γι' αυτό δεν ενδείκνυται για DEBUGGING ενός προγράμματος.

Διακοπή Debugging INT 3

Η διακοπή αυτή χρησιμοποιείται έξυπνα από το πρόγραμμα MONITOR, κατά την εκτέλεση ενός προγράμματος, με καθορισμό διευθύνσεων παύσης (breakpoints). Μπορεί να χρησιμοποιηθεί ως η τελευταία εντολή ενός προγράμματος, καθώς μόλις εκτελείται, εμφανίζει τις τιμές των καταχωρητών και επιστρέφει στο MONITOR.

Κεφάλαιο 2 : Το Πρόγραμμα MONITOR και οι εντολές του.

Το Πρόγραμμα MONITOR

Το πρόγραμμα MONITOR του BGC-8088 αποτελεί στην ουσία ένα μίνι λειτουργικό σύστημα που επιτρέπει στον χρήστη του μηχανήματος να διαχειριστεί το μηχάνημα και να εισάγει και να εκτελέσει προγράμματα Assembly. Το πρόγραμμα MONITOR περιέχεται στην ROM του μηχανήματος και εκτελείται μόλις εφαρμοστεί η τάση τροφοδοσίας στην κεντρική πλακέτα και στον επεξεργαστή. Όταν εκτελεστεί εμφανίζει το μήνυμα :

```
BGC-8088 MICROENGINEER MONITOR V 3.5
*
```

Το αστεράκι «*» αποτελεί στην ουσία τον χαρακτήρα προτροπής (prompt) της γραμμής εντολών του MONITOR που υποδηλώνει ότι το MONITOR είναι έτοιμο για εισαγωγή εντολών. Ακριβώς δίπλα εμφανίζεται ένας κέρσορας που αναβοσβήνει, υποδεικνύοντας στον χρήστη την θέση που θα εμφανιστεί ο επόμενος χαρακτήρας που θα πληκτρολογηθεί. Αφού εμφανίσει την προτροπή και τον κέρσορα, το MONITOR περιμένει την εισαγωγή εντολής από τον χρήστη. Αν ο χρήστης δώσει μία έγκυρη εντολή και πατήσει το πλήκτρο CR (Carriage Return) τότε το MONITOR αναλύει την εντολή σε εντολή-παραμέτρους και κάνει τις απαραίτητες ενέργειες, εκτελώντας τον κατάλληλο κώδικα ανά περίπτωση. Αν δοθεί λάθος εντολή ή σωστή εντολή με λάθος παραμέτρους τότε το MONITOR εμφανίζει την φράση :

```
←ERROR !
```

δίπλα στην εσφαλμένη εντολή.
Μετά από την ολοκλήρωση της κάθε εντολής το MONITOR εμφανίζει και πάλι τον χαρακτήρα προτροπής «*» και τον κέρσορα, περιμένοντας την επόμενη εντολή.
Να σημειωθεί εδώ ότι μετά από την εκτέλεση μίας εντολής μπορούμε να την ξανα-εκτελέσουμε (χωρίς να την ξανα-γράψουμε) απλά πατώντας το βελάκι επάνω «↑». Επίσης μπορούμε πριν πατήσουμε το CR (Carriage Return) μπορούμε να διορθώσουμε ή να συμπληρώσουμε την εντολή κατά βούληση. Επίσης αντί για το βελάκι επάνω ο χρήστης μπορεί να πατήσει επαναληπτικά το βελάκι κάτω «↓» οπότε και εμφανίζεται η προηγούμενη εντολή χαρακτήρα – χαρακτήρα.
Επίσης, σε περίπτωση λάθους μας μπορούμε να κάνουμε Soft-Reset στο σύστημα, δηλαδή επανεκκίνηση του συστήματος μέσω λογισμικού, πατώντας απλά το πλήκτρο “RESET”.

Οι Εντολές του MONITOR

Οι εντολές που δέχεται το πρόγραμμα MONITOR χωρίζονται σε 6 κατηγορίες που είναι :

1. Εντολές Διαχείρισης Μνήμης
2. Εντολές Συμβολομετάφρασης
3. Εντολές Αριθμητικών Πράξεων
4. Εντολές Ελέγχου Εκτέλεσης Προγράμματος
5. Εντολές Εισόδου/Εξόδου
6. Εντολές Επικοινωνίας

Οι εντολές του MONITOR αποτελούνται από ένα ή δύο γράμματα και 0-3 παραμέτρους. Όλες οι αριθμητικές παράμετροι δίνονται στο δεκαεξαδικό σύστημα. Για παράδειγμα η εντολή :

```
F10,20,FF
```

γεμίζει (**F**ill) τις θέσεις μνήμης 0100:0010 έως και 0100:0020 με τον αριθμό FF (255 στο δεκαδικό σύστημα)

Παράμετροι των Εντολών

Παρακάτω αναλύονται η σύνταξη και η σημασία συχνά χρησιμοποιούμενων παραμέτρων στις εντολές του MONITOR :

<Διεύθυνση> : σε παραμέτρους εντολών όπου απαιτείται <Διεύθυνση> ο χρήστης πρέπει να εισάγει μία έγκυρη διεύθυνση μνήμης του 8088. Ως γνωστόν οι διευθύνσεις μνήμης του 8088 αποτελούνται από δύο συστατικά segment : offset (τμήμα : μετατόπιση). Κατά συνέπεια και οι παράμετροι τύπου <Διεύθυνση> θα πρέπει να γράφονται ομοίως. Και το segment και το offset έχουν μέγιστο 4 δεκαεξαδικά ψηφία (από 0000 έως FFFF). Έτσι η μικρότερη διεύθυνση είναι 0000:0000 και η μεγαλύτερη FFFF:FFFF. Να υπενθυμίσουμε εδώ ότι η απόλυτη διεύθυνση προκύπτει από τον τύπο :

Απόλυτη Διεύθυνση = segment * 16 + offset

Εάν στην παράμετρο διεύθυνσης υπάρχουν και segment και offset, τότε αυτά χωρίζονται με άνω και κάτω τελεία «:». Το segment μπορεί και να παραληφθεί σε μία διεύθυνση αλλά όχι και το offset. Όταν παραλείπεται το segment, τότε εννοείται το εξ'ορισμού segment το οποίο διαφέρει σε κάθε εντολή. Επίσης οι Καταχωρητές Τμημάτων (Segment Registers) CS, DS, ES, SS μπορούν να χρησιμοποιηθούν στη θέση του segment της διευθυνσης.

Παραδείγματα έγκυρων διευθύνσεων :	**Παραδείγματα μη έγκυρων διευθύνσεων :**
0100:5E3C	01001:5E3C
CS:5E3C	CS5E3C
ES:5E3C	DS
5E3C	01005E3C
3C	FG

Έτσι η εντολή :

```
F10,20,FF
```

μπορεί να γραφτεί σε πλήρη μορφή ως :

```
F0100:0010,0020,FF
```

Να σημειωθεί εδώ ότι γενικά οι εντολές εκτελούνται για διευθύνσεις που ανήκουν στο ίδιο segment και διαφέρουν στο offset. Έτσι γενικά δεν επιτρέπεται στην δεύτερη διεύθυνση να έχει segment. Εννοείται το ίδιο με αυτό της πρώτης διεύθυνσης.

<Περιοχή Διευθύνσεων> : χρησιμοποιείται σε εντολές που εφαρμόζονται σε περιοχή συνεχόμενων διευθύνσεων και όχι σε μεμονωμένες διευθύνσεις. Μπορεί να έχει 3 διαφορετικές συντάξεις :

1. <Διεύθυνση1> <Διεύθυνση2> : όπου η <Διεύθυνση1> δηλώνει την αρχή της περιοχής διευθύνσεων και η <Διεύθυνση2> το τέλος. Οι δύο διευθύνσεις πρέπει να ανήκουν στο ίδιο segment και διαφέρουν στο offset. Έτσι γενικά δεν

επιτρέπεται στην δεύτερη διεύθυνση να έχει segment. Εννοείται το ίδιο με αυτό της πρώτης διεύθυνσης.

2. <Διεύθυνση> L <Αριθμός> : όπου η <Διεύθυνση> δηλώνει την αρχή της περιοχής διευθύνσεων και το L <Αριθμός> το μέγεθος της περιοχής σε bytes. Έτσι η περιοχή διευθύνσεων 0010 0020 μπορεί να γραφτεί και ως 0010 L11.
3. <Διεύθυνση> : που δηλώνει την αρχή της περιοχής διευθύνσεων ενώ το μέγεθος της περιοχής είναι ένα εξ'ορισμού μέγεθος ανάλογα με την εντολή. Π.χ. στην εντολή D το εξ'ορισμού μέγεθος περιοχής είναι 80, και έτσι η εντολή D0100:0200 ισοδυναμεί με την εντολή D0100:0200 L80 αλλά και με την εντολή D0100:0200 027F.

Παραδείγματα έγκυρων περιοχών :

0100:0100 0200
CS:0100 0200
DS:0100 L100
SS:0100
0100

Παραδείγματα μη έγκυρων περιοχών :

0100:0100 0100:0200
0100 L0100:0200
DS L20
0100-0200
0100..0200

<Αριθμός> : όπου πρέπει να δώσουμε έναν αριθμό με μέγιστο τα 4 δεκαεξαδικά ψηφία, δηλαδή στην περιοχή 0000-FFFF (0..65535)

<Byte> : όπου πρέπει να δώσουμε έναν αριθμό με μέγιστο τα 2 δεκαεξαδικά ψηφία, δηλαδή στην περιοχή 00-FF (0.255).

<String> : αλφαριθμητικές σταθερές, δηλαδή σειρές χαρακτήρων (γραμμάτων-αριθμών-συμβόλων) που περικλείονται μέσα σε μονά ή διπλά εισαγωγικά. Να σημειωθεί εδώ ότι αν ένα string αρχίζει με μονό εισαγωγικό θα πρέπει και να κλείνει με μονό εισαγωγικό, ενώ αν αρχίζει με διπλό εισαγωγικό θα πρέπει και να κλείνει με διπλό εισαγωγικό. Επίσης, ένα string που περικλείεται από μονά εισαγωγικά μπορεί να περιέχει εσωτερικά χαρακτήρες διπλών εισαγωγκών π.χ. 'This is a "typical" string', οι οποίοι θεωρούνται τότε κοινοί χαρακτήρες. Ομοίως ένα string που περικλείεται από διπλά εισαγωγικά μπορεί να περιέχει εσωτερικά χαρακτήρες μονών εισαγωγκών π.χ. "This is a 'typical' string", οι οποίοι θεωρούνται τότε κοινοί χαρακτήρες.

<Σειρά> : που αποτελείται από μία ακολουθία από <Byte> ή <String> που χωρίζονται μεταξύ τους με κόμμα ",". Παραδείγματα σειρών είναι :

01,02,03,04,05,06,07,08,09,0A
'ABC','DEF','GHI','JKL'
"12",34,'56',78,"9A",BC
"The ",'number ', "is : " , 1C

Εντολές Διαχείρισης Μνήμης

Οι εντολές διαχείρισης μνήμης του προγράμματος MONITOR είναι (κατά αλφαβητική σειρά) :

C	(Compare)	: Συγκρίνει τα περιεχόμενα της μνήμης
D	(Display)	: Εμφανίζει τα περιεχόμενα της μνήμης
E	(Edit)	: Τοποθέτηση και μεταβολή δεδομένων στην μνήμη

F (Fill) : Γεμίζει την μνήμη με συγκεκριμένα δεδομένα
M (Move) : Μετακινεί δεδομένα σε άλλη περιοχή μνήμης

Εντολή C (Compare)

Σύνταξη Εντολής : C <Περιοχή Διευθύνσεων> <Διεύθυνση>

Η <Περιοχή Διευθύνσεων> καθορίζει τις διευθύνσεις μνήμης που θα συγκριθούν και η <Διεύθυνση> καθορίζει την αρχή της δεύτερης περιοχής δεδομένων με την οποία θα γίνει η σύγκριση byte προς byte. Αν κατά την σύγκριση δεν προκύψει καμία διαφορά τότε δεν εμφανίζεται τίποτα. Αν κάποιο ή κάποια bytes διαφέρουν τότε για κάθε τέτοιο byte εμφανίζεται η διεύθυνση του byte της πρώτης περιοχής, το ίδιο το byte της πρώτης περιοχής, το αντίστοιχο byte της δεύτερης περιοχής και η αντίστοιχη διεύθυνση της δεύτερης περιοχής. Παράδειγμα :

```
C 0 F 20        ή     C0,F,20
```

(που σημαίνει σύγκρινε τα περιεχόμενα των διευθύνσεων μνήμης από την διεύθυνση 0 (0100:0000) έως και την διεύθυνση μνήμης F (0100:000F) με τα περιεχόμενα των αντίστοιχων διευθύνσεων που ξεκινούν από την διεύθυνση 20 (0100:0020) δηλαδή με τα περιεχόμενα των διευθύνσεων 0100:0020 εως 0100:002F. Το αποτέλεσμα θα είναι κάτι σαν το επόμενο :

```
0100:0000 01 FF 0100:0020
0100:0001 02 FE 0100:0021
0100:0002 03 FD 0100:0022
0100:0003 04 FC 0100:0023
0100:0004 05 FB 0100:0024
0100:0005 06 FA 0100:0025
0100:0006 07 F9 0100:0026
0100:0007 08 F8 0100:0027
0100:0008 09 F7 0100:0028
0100:0009 0A F6 0100:0029
0100:000A 0B F5 0100:002A
0100:000B 0C F4 0100:002B
0100:000C 0D F3 0100:002C
0100:000D 0E F2 0100:002D
0100:000E 0F F1 0100:002E
0100:000F 10 F0 0100:002F
```

με την προϋπόθεση βέβαια ότι όλα τα bytes διαφέρουν.

Εντολή D (Display)

Σύνταξη Εντολής : D <περιοχή διευθύνσεων>

Η Εντολή D (Display) εμφανίζει τα περιεχόμενα των διευθύνσεων μνήμης που καθορίζονται από την <περιοχή διευθύνσεων>. Εάν η εντολή δοθεί χωρίς παράμετρο τότε εμφανίζει 128 συνεχόμενα bytes (80_H) από την διεύθυνση που είχε σταματήσει η προηγούμενη εντολή D (Display). Εάν η εντολή εκτελείται για πρώτη φορά και χωρίς παράμετρο τότε εμφανίζει τα πρώτα 128 bytes από την διεύθυνση 0100:0000. Τα δεδομένα της μνήμης εμφανίζονται σε οκτάδες bytes ανά γραμμή οθόνης ενώ εμφανίζεται και η διεύθυνση του πρώτου byte της οκτάδας αλλά και η ASCII αναπαράσταση των bytes. Παράδειγμα :

```
D 0 1F              ή     D0,1F
```

(που σημαίνει εμφάνισε τα περιεχόμενα των διευθύνσεων από την διεύθυνση 0 (0100:0000) έως και την διεύθυνση 1F (0100:001F)). Το αποτέλεσμα θα είναι κάτι σαν το ακόλουθο :

```
0100:0000  41424344-45464748 ABCDEFGH
0100:0008  494A4B4C-4D4E4F50 IJKLMNOP
0100:0010  51525354-55565758 QRSTUVWX
0100:0018  595A5B5C-5D5E5F60 YZ[\]^_`
```

Τα δεδομένα εμφανίζονται δυο-δύο γραμμές και πηγαίνουμε παρακάτω πατώντας ένα οποιοδήποτε πλήκτρο.
Εάν στην περιοχή διευθύνσεων δεν καθοριστεί segment (όπως και στο παραπάνω παράδειγμα) τότε εννοείται το segment που δείχνει ο καταχωρητής DS (που έχει εξ'ορισμού την τιμή 0100).

Εντολή E (Edit)

Σύνταξη Εντολής : E <Διεύθυνση> [<Σειρά>]
Η εντολή E (Edit) επιτρέπει στον χρήστη να δει αλλά και να μεταβάλλει τα δεδομένα της μνήμης. Η προβολή και μεταβολή των δεδομένων αρχίζει από την <Διεύθυνση> που δίνεται ως παράμετρος και συνεχίζεται μέχρι ο χρήστης να δώσει σε κάποια διεύθυνση "Q" και πατήσει το πλήκτρο CR (Carriage Return), οπότε η εντολή σταματά και επανερχόμαστε στην προτροπή του MONITOR. Η πρβολή/μεταβολή των δεδομένων γίνεται ως εξής :
1. Εμφανίζεται η διεύθυνση των δεδομένων και δίπλα το περιεχόμενο byte με 2 δεκαεξαδικά ψηφία (προβολή).
2. Εάν θέλουμε να μεταβάλλουμε το περιεχόμενο αυτής της διεύθυνσης μνήμης τότε ακριβώς δίπλα πληκτρολογούμε το νέο περιεχόμενο με το πολύ 2 δεκαεξαδικά ψηφία και πατάμε το πλήκτρο CR (Carriage Return).
3. Εάν δεν θέλουμε να μεταβάλλουμε το περιεχόμενο τότε απλά πατάμε το CR (Carriage Return) για να πάμε στην επόμενη διεύθυνση μνήμης.
Παράδειγμα :

```
E 0
```

(που σημαίνει εκκίνηση της προβολής/μεταβολής των δεδομένων της μνήμης από την διεύθυνση 0 ή 0100:0000). Το αποτέλεσμα θα είναι κάτι σαν το ακόλουθο :

```
0100:0000  41 FF     (έγινε αλλαγή από 41 σε FF)
0100:0001  42        (δεν εγινε αλλαγή)
0100:0002  43        (δεν εγινε αλλαγή)
0100:0003  44 FE     (έγινε αλλαγή από 44 σε FE)
0100:0002  43        (δεν εγινε αλλαγή)
0100:0002  q         (τερματισμός της εντολής)
```

Εάν δοθεί ως δεύτερη παράμετρος και μία <Σειρά> τότε η εντολή E (Edit) αυτόματα τοποθετεί τα δεδομένα της <Σειράς> σε συνεχόμενες διευθύνσεις, ξεκινώντας από την <Διεύθυνση> που δόθηκε ως πρώτη παράμετρος. Παράδειγμα :

```
E 0 1,2,3,4,5
```

(τοποθετεί τα νούμερα 1..5 σε συνεχόμενες διευθύνσεις ξεκινώντας από την διεύθυνση 0 ή 0100:0000). Έτσι εάν στην συνέχεια δώσουμε :

```
D 0,F
```

(δηλαδή να μας εμφανίσει τα περιεχόμενα των διευθύνσεων 0 ή 0100:0000 έως και F ή 0100:000F) θα πάρουμε κάτι σαν το ακόλουθο :

```
0100:0000  01020304-05464748 .....FGH
0100:0008  494A4B4C-4D4E4F50 IJKLMNOP
```

όπου φαίνεται η τοποθέτηση των αριθμών 1..5 στις πρώτες 5 διευθύνσεις αρχίζοντας από την διεύθυνση 0100:0000.

Εντολή F (Fill)

Σύνταξη Εντολής : F <Περιοχή Διευθύνσεων> <Σειρά>

Η εντολή F (Fill) γεμίζει τις διευθύνσεις μνήμης που καθορίζονται από την <Περιοχή Διευθύνσεων> με τα δεδομένα που βρίσκονται στην <Σειρά>. Εάν η <περιοχή διευθύνσεων> είναι μεγαλύτερη από τα δεδομένα της <Σειράς> τότε τα δεδομένα της <Σειράς> τοποθετούνται επαναληπτικά στην μνήμη (σαν pattern) μέχρι να γεμίσουν ολόκληρη την <Περιοχή Διευθύνσεων>. Παράδειγμα :

```
F 0,7,FF
```

(δηλαδή γέμισε τις διευθύνσεις από 0 ή 0100:0000 έως και 7 ή 0100:0007 με τον αριθμό FF). Εάν στη συνέχεια δώσουμε :

```
D 0,F
```

(δηλαδή να μας εμφανίσει τα περιεχόμενα των διευθύνσεων 0 ή 0100:0000 έως και F ή 0100:000F) θα πάρουμε κάτι σαν το ακόλουθο :

```
0100:0000  FFFFFFFF-FFFFFFFF ........
0100:0008  494A4B4C-4D4E4F50 IJKLMNOP
```

όπου φαίνεται ότι οι πρώτες 8 θέσεις μνήμης έχουν γεμίσει με τον αριθμό FF.

Παράδειγμα δεύτερο :

```
F 0,F,41,42
```

(δηλαδή γέμισε τις διευθύνσεις από 0 ή 0100:0000 έως και F ή 0100:000F με τους αριθμούς 41 και 42 επαναληπτικά). Εάν στη συνέχεια δώσουμε :

```
D 0,F
```

(δηλαδή να μας εμφανίσει τα περιεχόμενα των διευθύνσεων 0 ή 0100:0000 έως και F ή 0100:000F) θα πάρουμε το ακόλουθο :

```
0100:0000  41424142-41424142 ABABABAB
0100:0008  41424142-41424142 ABABABAB
```

όπου φαίνεται ότι οι πρώτες 16 θέσεις μνήμης έχουν γεμίσει με το pattern 41,42.

Εντολή M (Move)

Σύνταξη Εντολής : M <Περιοχή Διευθύνσεων> <Διεύθυνση>

Η εντολή M (Move) μετακινεί (αντιγράφει) τα δεδομένα της <περιοχής διευθύνσεων> σε μία άλλη περιοχή διευθύνσεων που ξεκινά από την <Διεύθυνση> που δόθηκε ως $2^{η}$ παράμετρος. Οι δύο περιοχές διευθύνσεων μπορεί και να επικαλύπτονται μερικώς. Σε αυτή την περίπτωση η αντιγραφή γίνεται σωστά αλλά τα δεδομένα της <Περιοχής Διευθύνσεων> καταστρέφονται μερικώς αφού επικαλύπτονται κατά την αντιγραφή. Παράδειγμα :

```
E 0 1,2,3,4,5
```

(τοποθετεί τα νούμερα 1..5 σε συνεχόμενες διευθύνσεις ξεκινώντας από την διεύθυνση 0 ή 0100:0000). Έτσι εάν στην συνέχεια δώσουμε :

```
D 0,F
```

(δηλαδή να μας εμφανίσει τα περιεχόμενα των διευθύνσεων 0 ή 0100:0000 έως και F ή 0100:000F) θα πάρουμε κάτι σαν το ακόλουθο :

```
0100:0000  01020304-05464748 .....FGH
0100:0008  494A4B4C-4D4E4F50 IJKLMNOP
```

όπου φαίνεται η τοποθέτηση των αριθμών 1..5 στις πρώτες 5 διευθύνσεις αρχίζοντας από την διεύθυνση 0100:0000. Εάν τότε δώσουμ :

```
M 0,4,8
```

(δηλαδή να μετακινήσει τα δεδομένα της περιοχής διευθύνσεων από 0 ή 0100:0000 έως και 4 ή 0100:0004 στην περιοχή που ξεκινά από την διεύθυνση 8 ή 0100:0008). Έτσι εάν στην συνέχεια δώσουμε :

```
D 0,F
```

(δηλαδή να μας εμφανίσει τα περιεχόμενα των διευθύνσεων 0 ή 0100:0000 έως και F ή 0100:000F) θα πάρουμε κάτι σαν το ακόλουθο :

```
0100:0000  01020304-05464748 .....FGH
0100:0008  01020304-054E4F50 .....NOP
```

όπου φαίνεται η αντιγραφή των αριθμών 1..5 στις διευθύνσεις από την 8 ή 0100:0008 και μετά.

Εντολές Συμβολομετάφρασης

Οι εντολές Συμβολομετάφρασης του προγράμματος MONITOR επιτρέπουν την εισαγωγή εντολών προγράμματος γλώσσας μηχανής του 8088 σε μορφή συμβολικής γλώσσας Assembly (Assembly Language). Οι εντλές αυτές είναι (κατά αλφαβητική σειρά) :

A	(Assemble)	: Επιτρέπει την εισαγωγή προγράμματος γλώσσας μηχανής σε μορφή συμβολικού κώδικα assembly
I	(Insert)	: Επιτρέπει την παρεμβολή εντολών προγράμματος γλώσσας μηχανής σε μορφή συμβολικού κώδικα assembly, ανάμεσα σε εντολές ήδη καταχωρημένου προγράμματος (παρεμβολή εντολής).
U	(Un-Assemble)	: Εμφανίζει ένα ήδη καταχωρημένο πρόγραμμα γλώσσας μηχανής σε μορφή συμβολικού κώδικα assembly (Disassemble).

Εντολή A (Assemble)

Σύνταξη Εντολής : A [<Διεύθυνση>]

Επιτρέπει στον χρήστη την εισαγωγή προγράμματος γλώσσας μηχανής του 8088 στην μνήμη, σε μορφή συμβολικού κώδικα assembly. Η καταχώρηση του προγράμματος ξεκινά από την <Διεύθυνση>. Εάν η εντολή A (Assemble) δοθεί χωρίς παράμετρο τότε ξεκινά την καταχώρηση του προγράμματος από την εξ'ορισμού διεύθυνση 0100:0000.

Να σημειωθεί εδώ ότι μία εντολή σε μορφή **γλώσσας μηχανής** εκφράζεται καθαρά αριθμητικά, για παράδειγμα :

```
01 D8
```

(σημαίνει πρόσθεσε τον καταχωρητή AX με τον καταχωρητή BX και βάλε το αποτέλεσμα στον AX). Η ίδια εντολή σε **συμβολική γλώσσα Assembly** είναι :

```
ADD AX,BX
```

Όπως γίνεται εύκολα αντιληπτό στην Assembly χρησιμοποιούνται μνημονικές λέξεις (mnemonics) για τις εντολές καθώς και ονόματα καταχωρητών αλλα και στοιχεία

σύνταξης όπως το κόμμα "," για να γίνεται ευκολότερα και η σύνταξη ενός προγράμματος αλλά και η ανάγνωση ενός ήδη καταχωρημένου προγράμματος.
Πρέπει να ξεκαθαριστεί από την αρχή επίσης ότι :

1. Ο Assembler του MONITOR μεταφράζει κατευθείαν τις εντολές Assembly σε μορφή γλώσσας μηχανής και τοποθετεί τα νούμερα στην μνήμη.
2. Δεν έχει την δυνατότητα χρήσης Ετικετών – Labels για την μακρο-αντικατάσταση διευθύνσεων.
3. Δεν έχει την δυνατότητα να κάνει link με βιβλιοθήκες ρουτινών
4. Δεν έχει την δυνατότητα χρήσης ετικέτας για το Τμήμα Κώδικα (Code Segment) ή το Τμήμα Δεδομένων (Data Segment).

Οι εντολές της Assembly του 8088 και οι τρόποι σύνταξης των εντολών αναφέρονται αναλυτικά στο Κεφάλαιο 3.
Η εντολή A (Assemble) λειτουργεί ως εξής : Εμφανίζει την διεύθυνση όπου θα εισαχθεί η πρώτη εντολή που είναι ίδια με την <Διεύθυνση>.Π.χ

```
0100:0000 _
```

Εκεί ο χρήστης πληκτρολογεί την πρώτη εντολή του προγράμματος και πατάει CR (Carriage Return). Αυτόματα εμφανίζεται η διεύθυνση της επόμενης εντολής, αφού βέβαια ληφθεί υπ'όψιν το μέγεθος της εντολής που δώσαμε, σε bytes. Π.χ. :

```
0100:0000 MOV AX,0001
0100:0003 _
```

Κατ' αυτόν τον τρόπο ο χρήστης μπορεί να πληκτρολογήσει όλες τις εντολές του προγράμματος. Όταν πληκτρολογήσει και την τελευταία, τότε στην επόμενη γραμμή πατάει CR (Carriage Return) και τερματίζεται η εντολή A (Assemble), ξαναγυρνώντας στην προτροπή του MONITOR.

Εντολή I (Insert)
Σύνταξη Εντολής : I [<Διεύθυνση>]
Επιτρέπει στον χρήστη την εισαγωγή προγράμματος γλώσσας μηχανής του 8088 στην μνήμη, σε μορφή συμβολικού κώδικα assembly, αλλά σε Insert Mode. Αυτό σημαίνει ότι οι εντολές που θα πληκτρολογηθούν εισάγονται ανάμεσα στις εντολές του προγράμματος που υπήρχε ήδη στην μνήμη σε εκείνη τη διεύθυνση. Για παράδειγμα έστω το παρακάτω πρόγραμμα :

```
0100:0000 MOV AX,0001
0100:0003 MOV BX,0002
0100:0006 ADD AX,BX
0100:0008 MOV [0050],AX
```

Το οποίο προσθέτει τους αριθμούς 1 και 2 τοποθετώντας τους στους καταχωρητές AX και BX αντίστοιχα, και αποθηκεύει το αποτέλεσμα στη θέση μνήμης 50 ή 0100:0050.
Εάν θέλουμε να προσθέσουμε μία εντολή ακόμα (π.χ. μία εντολή NOP) πρίν από την εντολή ADD, θα πρέπει να δώσουμε την εντολή :

```
I 6
```

Οπότε το MONITOR θα μας επιτρέψει να εισάγουμε εντολή/ές πριν από την ADD. Κατά την εισαγωγή των νέων εντολών, αυτές που υπήρχαν από την ADD και κάτω, μετατοπίζονται προς τα κάτω αφήνοντας χώρο για τις νέες εντολές που εισάγονται. Έτσι αν εισάγουμε μόνο την NOP και στην επόμενη απλά πατήσουμε CR ο κώδικας θα έχει γίνει ως εξής :

```
0100:0000 MOV AX,0001
```

```
0100:0003 MOV BX,0002
0100:0006 NOP
0100:0007 ADD AX,BX
0100:0009 MOV [0050],AX
```

Ο χρήστης θα παρατηρήσει ότι κατά την εισαγωγή εντολών σε Insert Mode ανάβει το LED με το διακριτικό "INS" που βρίσκεται πάνω από την LCD οθόνη (δίπλα στο LED με το διακριτικό "POWER" που ανάβει όσο το μηχάνημα έχει τροφοδοσία).

Ο χρήστης μπορεί να γυρίσει σε Insert Mode ακόμα και όταν έχει ξεκινήσει με εντολή A (Assemble) η οποία επικαλύπτει τις θέσεις μνήμης με τις νέες εντολές, και όχι από I (Insert). Αυτό γίνεται αν κατά την εκτέλεση της εντολής A (Assemble) ο χρήστης πατήσει το πλήκτρο "INS" οπότε ανάβει το LED "INS" και γυρνάμε σε Insert Mode.

Όσο βρισκόμαστε σε Insert Mode, οι εντολές που πληκτρολογούμε εισάγονται στο πρόγραμμα μετακινώντας τις άλλες εντολές προς τα κάτω. Όσο δεν βρισκόμαστε σε Insert Mode οι εντολές που πληκτρολογούμε επικαλύπτουν τις εντολές που υπήρχαν εκεί.

Πρέπει επίσης να τονιστεί ότι η μετακίνηση των εντολών προς τα κάτω, που λαμβάνει χώρα κατά την πληκτρολόγηση εντολών σε Insert Mode, μπορεί να καταστήσει μη έγκυρες ή απλά λανθασμένες τις διευθύνσεις εντολών μετακίνησης όπως η εντολή JMP (Jump). Για παράδειγμα, έστω το πρόγραμμα :

```
0100:0000 MOV BX,0000
0100:0003 INC BX
0100:0004 ADD AX,BX
0100:0006 JMP 0003
```

Το οποίο προσθέτει στον AX τους αριθμούς 1+2+3+… και η εντολή JMP 0003 υλοποιεί τον βρόχο επανάληψης (επ'άπειρον). Αν πριν την εντολή INC εισάγουμε μία εντολή μηδενισμού του AX (π.χ. μία MOV AX,0000) η οποία λείπει και πρέπει να μπεί τότε ο κώδικας θα γίνει ως εξής :

```
0100:0000 MOV BX,0000
0100:0003 MOV AX,0000
0100:0006 INC BX
0100:0007 ADD AX,BX
0100:0009 JMP 0003
```

Στον κώδικα αυτό αμέσως φαίνεται η εσφαλμένη διεύθυνση της εντολής JMP η οποία έχει παραμείνει 0003 ενώ τώρα το σωστό είναι JMP 0006. Αυτό είναι κάτι που πρέπει να το προσέχει ιδιαίτερα ο χρήστης όταν χρησιμοποιεί το Insert Mode.

Εντολή U (Unassemble)

Σύνταξη Εντολής : U [<Περιοχή Διευθύνσεων>]

Η εντολή αυτή διαβάζει τα bytes που βρίσκονται αποθηκευμένα στην <Περιοχή Διευθύνσεων> , τα εκλαμβάνει ως εντολές γλώσσας μηχανής και τα εμφανίζει στην οθόνη σε μορφή συμβολικής γλώσσας Assembly που είναι πιο κατανοητή στην ανάγνωση ενός προγράμματος από την γλώσσα μηχανής.

Εάν η εντολή δοθεί χωρίς παράμετρο τότε μεταφράζει μόνο 32 συνεχόμενα bytes (20_H) από την διεύθυνση που είχε σταματήσει η προηγούμενη εντολή U (Unassemble). Εάν η εντολή εκτελείται για πρώτη φορά και χωρίς παράμετρο τότε εμφανίζει τα πρώτα 32 bytes από την διεύθυνση CS:IP που θεωρώντας τις εξ'ορισμού τιμές των καταχωρητών που είναι για τον CS 0100 και για τον IP 0000 είναι η διεύθυνση 0100:0000. Κατά την

εμφάνιση των εντολών εμφανίζεται μία εντολή σε κάθε γραμμή (διεύθυνση εντολής – εντολή σε μορφή κώδικα μηχανής – εντολή assembly) κάνοντας παύση για πάτημα πλήκτρου κάθε δύο γραμμές (όσες χωράει η LCD οθόνη). Παράδειγμα :

U 0

(που σημαίνει εμφάνισε τις εντολές Assembly από την διεύθυνση 0100:0000 και για τα πρώτα 32 bytes) θα εμφανιστεί :

```
0100:0000 BB0000    MOV BX,0000
0100:0003 B80000    MOV AX,0000
0100:0006 43        INC BX
0100:0007 01D8      ADD AX,BX
0100:0009 EBF8      JMP 0003
0100:000B FEFD      ???
0100:000D 7D7C      JNL 008B
.............................
```

Παρατηρούμε ότι μετά την εντολή JMP 0003 που εισάγαμε εμείς και έχει μεταφραστεί από το MONITOR στον αντίστοιχο κώδικα μηχανής EBF8 ακολουθούν τα δεδομένα FEFD τα οποία έτυχε να βρίσκονται σε εκείνες τις θέσεις μνήμης και για τα οποία η εντολή U εμφανίζει ερωτηματικά ??? Αυτό γίνεται γιατί η τυχαία ακολουθία bytes FE-FD ΔΕΝ ΑΝΤΙΣΤΟΙΧΕΙ ΣΕ ΚΑΜΙΑ ΕΝΤΟΛΗ ΤΟΥ 8088. Γι αυτό θα πρέπει να σιγουρευόμαστε ότι στην <Περιοχή Διευθύνσεων> που δίνουμε στην εντολή U, υπάρχουν όντως εντολές προγράμματος και όχι data.

Επίσης είναι σημαντικό να ξεκινάμε την εντολή U (Unassemble) από την σωστή διεύθυνση. Αν για παράδειγμα δώσουμε αντί για U 0 την εντολή U 1 τότε θα πάρουμε το εξής :

```
0100:0001 0000      ADD [BX+SI],AL
0100:0003 B80000    MOV AX,0000
0100:0006 43        INC BX
0100:0007 01D8      ADD AX,BX
0100:0009 EBF8      JMP 0003
0100:000B FEFD      ???
0100:000D 7D7C      JNL 008B
.............................
```

δηλαδή μας εμφανίζει «εσφαλμένα» την πρώτη εντολή (θα μπορούσε βέβαια το σφάλμα να συνεχιστεί και στην μετάφραση των επόμενων εντολών). Αυτό συμβαίνει γιατί η εντολή U 1 ξεκινά την μετάφραση από την μνήμη 0100:0001 όπου βρίσκει 2 συνεχόμενα μηδενικά 00-00 που αντιστοιχούν στην εντολή ADD [BX+SI],AL !

Ένα άλλο παράδειγμα είναι το εξής : αν η πρώτη εντολή ήταν MOV BX,0043 τότε δίνοντας U 0 θα παίρναμε :

```
0100:0000 BB4300    MOV BX,0043
0100:0003 B80000    MOV AX,0000
0100:0006 43        INC BX
0100:0007 01D8      ADD AX,BX
0100:0009 EBF8      JMP 0003
0100:000B FEFD      ???
0100:000D 7D7C      JNL 008B
.............................
```

Αν όμως δώσουμε U 1 θα εμφανιστεί το εξής :

```
0100:0001 43         INC BX
0100:0003 00B80000   ADD [BX+SI+0000],BH
0100:0006 43         INC BX
0100:0007 01D8       ADD AX,BX
0100:0009 EBF8       JMP 0003
0100:000B FEFD       ???
0100:000D 7D7C       JNL 008B
.............................
```

όπου φαίνεται ότι εκλαμβάνονται εσφαλμένα οι δύο πρώτες εντολές.

Εντολές Αριθμητικών Πράξεων

Οι εντολές Αριθμητικών Πράξεων του προγράμματος MONITOR επιτρέπουν την μετατροπή αριθμών μεταξύ διαφορετικών συστημάτων αρίθμησης (δυαδικό – δεκαδικό – δεκαεξαδικό) καθώς και στοιχειώδεις πράξεις. Οι εντολές αυτές είναι (κατά αλφαβητική σειρά) :

B	(dec to Bin)	: Μετατρέπει έναν δεκαδικό αριθμό σε δυαδικό
H	(+ & - of 2 Hex)	: Υπολογίζει το άθροισμα και την διαφορά 2 δεκαεξαδκών αριθμών.
J	(dec to hex)	: Μετατρέπει έναν δεκαδικό αριθμό σε δεκαεξαδικό.
S	(hex to dec)	: Μετατρέπει έναν δεκαεξαδικό αριθμό σε δεκαδικό.
V	(bin to dec)	: Μετατρέπει έναν δυαδικό αριθμό σε δεκαδικό.

Εντολή B (dec to Bin)

Σύνταξη Εντολής : B <Δεκαδικός Αριθμός>

Μετατρέπει τον δεκαδικό αριθμό σε δυαδικό. Ο δεκαδικός αριθμός που δίνεται ως παράμετρος μπορεί να είναι στο διάστημα (0..4294967295) ή (0..2^{32}-1). Αρνητικοί αριθμοί δεν γίνονται δεκτοί. Το αποτέλεσμα εμφανίζεται σαν ένας αριθμός των 32 bits (μέγιστο). Τυχόν πλεονάζοντα μηδενικά στα αριστερά δεν εμφανίζονται. Ο αριθμός ακολουθείται από το γράμμα "B" ώστε να γίνεται σαφές ότι είναι δυαδικός. Παραδείγματα :

```
B 4294967295
→11111111111111111111111111111111B
B 341
→101010101B
B 129
→10000001B
```

Εντολή H (+ & - of Hex)

Σύνταξη Εντολής : H <Αριθμός1>, <Αριθμός2>

Υπολογίζει το άθροισμα και την διαφορά δύο δεκαεξαδικών αριθμών. Οι <Αριθμοί> μπορεί να είναι το πολύ έως FFFF (65535_{10}) δηλαδή 16 bit. Αν το άθροισμα είναι μεγαλύτερο από FFFF τότε το κρατούμενο (carry) χάνεται. Αν ο πρώτος <Αριθμός> είναι μικρότερος από τον δεύτερο τότε για την αφαίρεση χρησιμοποιείται ένα έξτρα «δανεικό» έτσι ώστε η διαφορά να είναι θετικός αριθμός. Παραδείγματα :

```
H 5,3
SUM=0008  DIFF.=0002      (5+3=8, 5-3=2)
```

```
H 123,123
SUM=0246  DIFF.=0000     (123+123=246, 123-123=0)
H FFFF,1
SUM=0000  DIFF.=FFFE     (FFFF+1=10000 το κρατούμενο
                         χάνεται, FFFF-1=FFFE)
H 0,1
SUM=0001  DIFF.=FFFF     (0+1=1, 10000 - 1 = FFFF
                         χρησιμοποιείται δανεικό για
                         την αφαίρεση: 10000+0000-1 =
                         FFFF)
H 100,200
SUM=0300  DIFF.=FF00     (100+200=300, 10100-200=FF00
                         χρησιμοποιείται δανεικό για
                         την αφαίρεση: 10000+0100-
                         0200=FF00)
```

Εντολή J (dec to hex)

Σύνταξη Εντολής : J <Δεκαδικός Αριθμός>

Μετατρέπει τον δεκαδικό αριθμό σε δεκαεξαδικό. Ο δεκαδικός αριθμός που δίνεται ως παράμετρος μπορεί να είναι στο διάστημα (0..4294967295) ή (0..2^{32}-1). Αρνητικοί αριθμοί δεν γίνονται δεκτοί. Το αποτέλεσμα εμφανίζεται σαν ένας αριθμός των 8 δεκαεξαδικών ψηφίων (μέγιστο). Τυχόν πλεονάζοντα μηδενικά στα αριστερά δεν εμφανίζονται. Ο αριθμός ακολουθείται από το γράμμα “H” ώστε να γίνεται σαφές ότι είναι δεκαεξαδικός. Παραδείγματα :

```
J 12345
→3039H
J 65535
→FFFFH
J 11259375
→ABCDEFH
```

Εντολή S (hex to dec)

Σύνταξη Εντολής : S <Δεκαεξαδικός Αριθμός>

Μετατρέπει τον δεκαεξαδικό αριθμό σε δεκαδικό. Ο δεκαεξαδικός αριθμός που δίνεται ως παράμετρος μπορεί να είναι στο διάστημα (0..FFFFFFFF) ή (0..2^{32}-1) και το αποτέλεσμα είναι στο διάστημα (0..4294967295). Αρνητικοί αριθμοί δεν γίνονται δεκτοί. Τυχόν πλεονάζοντα μηδενικά στα αριστερά δεν εμφανίζονται. Ο αριθμός ακολουθείται από το γράμμα “D” ώστε να γίνεται σαφές ότι είναι δεκαδικός. Παραδείγματα :

```
S 3039
→12345D
S FFFF
→65535D
S ABCDEF
→11259375D
```

Εντολή V (bin to dec)

Σύνταξη Εντολής : V <Δυαδικός Αριθμός>

Μετατρέπει τον δυαδικό αριθμό σε δεκαδικό. Ο δυαδικός αριθμός που δίνεται ως παράμετρος μπορεί να είναι οποιοσδήποτε αριθμός με μέγιστο πλήθος ψηφίων 32 bits(0..2^{32}-1). Το αποτέλεσμα θα είναι στο διάστημα (0..4294967295). Αρνητικοί αριθμοί δεν γίνονται δεκτοί. Τυχόν πλεονάζοντα μηδενικά στα αριστερά δεν εμφανίζονται. Ο αριθμός ακολουθείται από το γράμμα "D" ώστε να γίνεται σαφές ότι είναι δεκαδικός. Παραδείγματα :

```
V 11111111111111111111111111111111
→4294967295D
V 101010101
→341D
V 10000001
→129D
```

Εντολές Ελέγχου Εκτέλεσης Προγράμματος

Οι εντολές αυτές επιτρέπουν την εκτέλεση ενός προγράμματος (ολόκληρου η εντολή-εντολή) και τον έλεγχο των τιμών των καταχωρητών του 8088. Οι εντολές αυτές είναι (κατά αλφαβητική σειρά) :

G	(Go)	: Εκτελεί ένα πρόγραμμα που είναι στη μνήμη μέχρι τέλους ή μέχρι συγκεκριμένη εντολή (breakpoint)
T	(Trace)	: Εκτελεί ένα πρόγραμμα εντολή - εντολή.
R	(Registers)	: Εμφανίζει τα περιεχόμενα των καταχωρητών (registers) του 8088.

Εντολή G (Go)

Σύνταξη Εντολής : G [=<Διεύθυνση Εκκίνησης>] [<Διεύθυνση Παύσης 1>, <Διεύθυνση Παύσης 2>, …, <Διεύθυνση Παύσης 10>]

Ξεκινά την εκτέλεση ενός προγράμματος γλώσσας μηχανής. Το πρόγραμμα ξεκινά από την <Διεύθυνση Εκκίνησης> και εκτελείται συνέχεια. Ο χρήστης μπορεί προαιρετικά να καθορίσει μέχρι 10 <Διευθύνσεις Παύσεων> (Breakpoints) όπου το πρόγραμμα θα σταματά προσωρινά και θα έχουμε την δυνατότητα να βλέπουμε τα αποτελέσματα (καταχωρητές, μνήμη κ.λ.π.). Εάν δεν δοθεί <Διεύθυνση Εκκίνησης> τότε το πρόγραμμα ξεκινά από την τρέχουσα διεύθυνση που έχουν αποθηκευμένη οι καταχωρητές CS : IP. Επειδή ο καταχωρητής IP (Instruction Pointer) ενημερώνεται (αυξάνει) αυτόματα κατά την εκτέλεση του προγράμματος, αν μετά από σταμάτημα σε breakpoint δώσουμε ξανά την εντολή "G" το πρόγραμμα απλά θα συνεχίσει την εκτέλεσή του.

Πρέπει να σημειωθεί εδώ το εξής : Το πρόγραμμα MONITOR είναι επίσης ένα πρόγραμμα σε γλώσσα μηχανής το οποίο όταν δώσουμε την εντολή "G" κάνει απλά JMP στην <διεύθυνση εκκίνησης> που δώσαμε σαν παράμετρο. Έτσι, όταν εκτελεστεί ένα δικό μας πρόγραμμα χωρίς <Διευθύνσεις Παύσης> (δηλαδή απλά με G=<Διεύθυνση Εκκίνησης> ή σκέτο G) ΔΕΝ υπάρχει τρόπος να επανέλθουμε στο πρόγραμμα MONITOR. Ο 8088 θα εκτελέσει το δικό μας πρόγραμμα και θα συνεχίσει να εκτελεί τις επόμενες εντολές που υπάρχουν στη μνήμη (ή τις υποθετικές εντολές στις οποίες αντιστοιχούν τα δεδομένα που τυχαίνει να βρίσκονται στις επόμενες θέσεις μνήμης). Έτσι το αποτέλεσμα θα είναι είτε να πέσει σε κάποιον ατέρμονα βρόχο (συνεχή επανάληψη) ή να σταματήσει την εκτέλεση εντολών όταν βρεί μία εντολή HLT (Halt). Σε κάθε περίπτωση όμως δεν θα είμαστε σε θέση να δούμε τι τιμές που πήραν οι

καταχωρητές ως αποτέλεσμα της εκτέλεσης του δικού μας προγράμματος, καθώς αυτοί θα έχουν επηρεαστεί και από άλλες εντολές. Έτσι το σωστό είναι όταν εκτελούμε ένα δικό μας πρόγραμμα το οποίο για παράδειγμα ξεκινά στη διεύθυνση 0100:0000 και τελειώνει στην διεύθυνση 0100:0200 να το εκτελέσουμε ως εξής :

G=0,200

(που σημαίνει ξεκίνα την εκτέλεση προγράμματος από την διεύθυνση 0 ή 0100:0000 και όταν φτάσεις στην διεύθυνση 200 ή 0100:0200 σταμάτα την εκτέλεση και εμφάνισέ μας τους καταχωρητές του 8088).

Προσοχή χρειάζεται στο ότι η <Διεύθυνση Παύσης> θα πρέπει όντως να αντιστοιχεί στην ΑΡΧΗ μίας εντολής και όχι σε ένα τυχαίο ενδιάμεσο byte, γιατί η εκτέλεση του προγράμματος θα σταματήσει προσωρινά μόνο όταν μετά από την εκτέλεση μίας εντολής βρεθεί ο καταχωρητής IP να ισούται ακριβώς με την <Διεύθυνση Παύσης>.

Σημείωση : υπάρχει τρόπος επιστροφής στο πρόγραμμα MONITOR χρησιμοποιώντας το Software Interrupt «INT 85» σαν τελευταία εντολή του προγράμματος. Όμως το INT 85 καλεί τμήμα κώδικα στην ROM που κάνει reset στο πρόγραμμα MONITOR μηδενίζοντας τους καταχωρητές. Έτσι, τυχόν αποτελέσματα που θέλετε να κρατήσετε ή να ελέγξετε θα πρέπει να τα αποθηκεύσετε σε θέσεις μνήμης.

Η καλύτερη λύση είναι να βάζουμε σαν τελευταία εντολή του προγράμματός μας την εντολή "**INT 3**" η οποία καλεί τμήμα κώδικα στην ROM που προκαλεί την εμφάνιση των καταχωρητών και αμέσως μετά επιστρέφει τον έλεγχο στο MONITOR. Το Software Interrupt "INT 3" χρησιμοποιείται με έξυπνο τρόπο από το πρόγραμμα MONITOR για την υλοποίηση των Breakpoints.

Εντολή R (Registers)

Σύνταξη Εντολής : R [<Ονομα Καταχωρητή>]

Εμφανίζει τις τιμές των καταχωρητών του 8088 και επιτρέπει την μεταβολή τους.

1. Εάν δοθεί χωρίς παράμετρο τότε η εντολή "R" απλά εμφανίζει τις τιμές των 14ων καταχωρητών του 8088 (5 ανά γραμμή) και στο τέλος εμφανίζει και την εντολή (σε μορφή assembly) που βρίσκεται στην διεύθυνση που δείχνουν οι καταχωρητές CS:IP Για παράδειγμα :

```
R
AX=0000 CX=0000 DX=0000 BX=0000 SP=0B3F
BP=0000 SI=0000 DI=0000 ES=0100 CS=0100
SS=0040 DS=0100 IP=0000 FG=0000
0100:0000 B80100    MOV AX,0001
```

2. Αν δοθεί με παράμετρο το <όνομα καταχωρητή> τότε εμφανίζει τα περιεχόμενα του συγκεκριμένου καταχωρητή και μας δίνει την δυνατότητα να αλλάξουμε την τιμή του. Αν λοιπόν δώσουμε μία νέα τιμή και πατήσουμε CR (Carriage Return) τότε ο καταχωρητής παίρνει την τιμή που δώσαμε. Αν δεν θέλουμε να αλλάξουμε την τιμή του καταχωρητή απλά παταμε CR. Μετά το CR εμφανίζεται η τιμή του επόμενου καταχωρητή για εποπτεία/αλλαγή. Η σειρά εμφάισης των καταχωρητών είναι AX,CX,DX,BX,SP,BP,SI,DI,ES,CS,SS,DS,IP,FG. Αν θέλουμε να τερματίσουμε την εντολή "R" μπορούμε είτε να δώσουμε "q" και CR ή να πατάμε CR μέχρι και τον Flag Register FG.

3. Ο καταχωρητής FG (Flag Register) είναι των 16 bit αλλά τα bit του αποτελούν σημαίες κατάστασης (flags) του 8088 και το καθένα έχει την δική του σημασία όπως περιγράφεται στο Κεφάλαιο 1. Μπορούμε να του αλλάξουμε τιμή, όπως και στους

υπόλοιπους καταχωρητές, αλλά στην συνολική δεκαεξαδική μορφή των 4ων ψηφίων Οπότε θα πρέπει ο χρήστης να υπολογίζει το συνολικό αριθμό που προκύπτει επηρεάζοντας συγκεκριμένα bits, πράγμα που δεν είναι γενικά εύκολο. Γι'αυτό το λόγο υπάρχει και μία ειδική εντολή η "**RF**" που μας επιτρέπει να θέσουμε τιμές στα bits του καταχωρητή FG με πιο εύκολο τρόπο και σε ένα-ένα bit ξεχωριστά.
Ως γνωστόν από τα 16 bits του FG χρησιμοποιούνται μόνο τα 9 που είναι :

Bit	**Ονομασία**	**Συντόμευση**	**Εξήγηση**
0	Carry Flag	CY	Κρατούμενο
2	Parity Flag	PF	Σημαία Ισοτιμίας
4	Auxiliary Carry Flag	AF	Βοηθητικό Κρατούμενο
6	Zero Flag	ZF	Σημαία Μηδενός
7	Sign Flag	SF	Σημαία Προσήμου
8	Trap Flag	TF	Σημαία Βηματικής Εκτέλεσης
9	Interrupt Flag	IF	Σημαία Αποδοχής Διακοπών
10	Direction Flag	DF	Σημαία Κατεύθυνσης
11	Overflow Flag	OF	Σημαία Υπερχείλισης

Από τις παραπάνω σημαίες η εντολή "RF" μας επιτρέπει να τις μεταβάλλουμε όλες εκός από το Trap Flag, το οποίο είναι πιθανό να προκαλέσει απρόβλεπτα αποτελέσματα αν μεταβληθεί από τον χρήστη. Για τα υπόλοιπα 8 flags το MONITOR έχει προβλέψει διαφορετική ονομασία για το κάθε flag και για το αν έχει την τιμή 0 ή 1. Οι ονομασίες αυτές για κάθε σημαία και για κάθε τιμή (0,1) φαίνονται στον παρακάτω πίνακα :

Σημαία\Bits	**0**	**1**
OF	NV	OV
DF	UP	DN
IF	DI	EI
SF	PL	NG
ZF	NZ	ZR
AF	NA	AC
PF	PO	PE
CY	NC	CY

Έτσι, για παράδειγμα, αν δώσουμε την εντολή "RF" και μας εμφανίζει το παρακάτω αποτέλεσμα :

```
FLAG STATUS=NV UP EI PL ZR AC PE NC
NEW STATUS=
```

αυτό σημαίνει ότι οι σημαίες είναι :

OF=0
DF=0
IF=1
SF=0
ZF=1
AF=1
PF=1
CY=0

Στην προτροπή "NEW STATUS=" ο χρήστης μπορεί να δώσει νέες τιμές σε όσα και όποια flags θέλει. Για παράδειγμα, αν δώσουμε :

```
FLAG STATUS=NV UP EI PL ZR AC PE NC
```

```
NEW STATUS=DI NG NA
```
Είναι σαν να αλλάζουμε μόνο τα παρακάτω flags και με τις εξής τιμές :

IF=0
SF=1
AF=0

Εντολή T (Trace)

Σύνταξη Εντολής : T [=<Διεύθυνση Εκκίνησης>] [<Αριθμός Εντολών>]

Η εντολή αυτή εκτελεί ένα πρόγραμμα που βρίσκεται στην μνήμη αλλά βήμα – βήμα (η καλύτερα εντολή-εντολή). Το πρόγραμμα ξεκινά από την <Διεύθυνση Εκκίνησης> και εκτελείται μία εντολή κάθε φορά. Μετά από κάθε εντολή που εκτελείται εμφανίζονται τα περιεχόμενα των καταχωρητών καθώς και η εντολή που υπάρχει στην διεύθυνση CS:IP (η επόμενη εντολή που θα εκτελεστεί). Αν παραλείψουμε την <Διεύθυνση Εκκίνησης> τότε το πρόγραμμα ξεκινά από την διεύθυνση που δείχνουν οι καταχωρητές CS:IP. Ο <Αριθμός Εντολών> καθορίζει πόσες εντολές του προγράμματος θα εκτελεστούν βήμα – βήμα. Αν η παράμετρος αυτή παραλειφθεί, τότε η εντολή "T" εκτελεί μία μόνο εντολή του προγράμματος και επιστρέφει στην προτροπή του MONITOR (αφού βέβαια μας δείξει τους καταχωρητές). Έστω για παράδειγμα το εξής πρόγραμμα :

```
0100:0000 MOV AX,0001
0100:0003 MOV BX,0002
0100:0006 ADD AX,BX
0100:0008 MOV [0050],AX
```

Το οποίο προσθέτει τους αριθμούς 1 και 2 τοποθετώντας τους στους καταχωρητές AX και BX αντίστοιχα, και αποθηκεύει το αποτέλεσμα στη θέση μνήμης 50 ή 0100:0050.

Δίνοντας :

T=0

Παίρνουμε :

```
AX=0001 CX=0000 DX=0000 BX=0000 SP=0B3F
BP=0000 SI=0000 DI=0000 ES=0100 CS=0100
SS=0040 DS=0100 IP=0003 FG=F002
0100:0003 BB0200    MOV BX,0002
T
AX=0001 CX=0000 DX=0000 BX=0002 SP=0B3F
BP=0000 SI=0000 DI=0000 ES=0100 CS=0100
SS=0040 DS=0100 IP=0006 FG=F002
0100:0006 01D8      ADD AX,BX
T
AX=0003 CX=0000 DX=0000 BX=0002 SP=0B3F
BP=0000 SI=0000 DI=0000 ES=0100 CS=0100
SS=0040 DS=0100 IP=0008 FG=F006
0100:0008 A35000    MOV [0050],AX
κ.λ.π.
```

Εντολές Εισόδου/Εξόδου

Οι εντολές αυτές επιτρέπουν την μεταφορά δεδομένων από και προς περιφερειακές συσκευές χωρίς να χρειάζεται να αναπτύξει ο χρήστης κατάλληλα προγράμματα γλώσσας μηχανής. Οι εντολές αυτές είναι (κατά αλφαβητική σειρά) :

N (iNput data) : Μπορεί να διαβάζει την τιμή κάποιας θύρας Εισόδου και να μας εμφανίζει την τιμή εισόδου.

O (Output data) : Στέλνει τιμές σε θύρες εξόδου.

Εντολή N (iNput data)

Σύνταξη Εντολής : N <Διεύθυνση I/O> [,]

Η εντολή αυτή διαβάζει την θύρα Εισόδου με την <Διεύθυνση I/O> που δόθηκε και μας εμφανίζει την τιμή εισόδου με την μορφή :

INPUT VALUE =60

Εάν δεν δοθεί το κόμμα ‘,’ τότε μόνο μία τιμή διαβάζεται από την θύρα και εμφανίζεται στην οθόνη. Αν δοθεί και το κόμμα τότε η θύρα διαβάζεται επαναληπτικά και εμφανίζεται η τιμή εισόδου στην οθόνη κάθε φορά που ο χρήστης πατάει το πλήκτρο ‘,’

Για να τερματιστεί η επανάληψη αρκεί να πατήσουμε το CR (Carriage Return). Για παράδειγμα :

N FF6D

(που σημαίνει διάβασε την τιμή του Line Status Register του chip 8250 που υλοποιεί την σειριακή θύρα RS-232)

INPUT VALUE =60

(που σημαίνει ότι τα bits 5 και 6 του καταχωρητή είναι 1, δηλαδή ο 8250 είναι έτοιμος να δεχτεί δεδομένα και ότι ο καταχωρητής αποστολής δεδομένων είναι άδειος)

Εντολή O (Output data)

Σύνταξη Εντολής : O <Διεύθυνση I/O> <Byte> [,]

Η εντολή αυτή εξάγει την τιμή <Byte> στην συσκευή εισόδου / εξόδου που καθορίζεται από την <Διεύθυνση I/O>. Εάν δεν δοθεί το κόμμα ‘,’ τότε μόνο μία τιμή (<Byte>) εξάγεται προς την <Διεύθυνση I/O> που καθορίστηκε. Αν δοθεί και το κόμμα τότε το πρόγραμμα MONITOR ζητάει επαναληπτικά από τον χρήστη να δώσει νέες τιμές για έξοδο προς την <Διεύθυνση I/O> που καθορίστηκε, με μια προτροπή της μορφής :

OUTPUT VALUE =

όπου ο χρήστης δίνει την τιμή εξόδου. Αν η τιμή συνοδευτεί από ένα κόμμα ‘,’ τότε η διαδικασία επαναλαμβάνεται, δηλαδή :

O FF41,41,

OUTPUT VALUE = 42,

OUTPUT VALUE = κ.λ.π.

Αν όμως η τιμή δεν συνοδεύεται από κόμμα τότε η διαδικασία σταματάει και ξαναγυρνάμε στην προτροπή του MONITOR.

Η παραπάνω εντολή O FF41,41 εξάγει στην διεύθυνση I/O FF41 η οποία αντιστοιχεί στον Καταχωρητή Δεδομένων (Data Register) του LCD τον αριθμό 41_H που αντιστοιχεί στο γράμμα ‘A’ βάσει του κώδικα ASCII. Το αποτέλεσμα είναι το γράμμα ‘A’ να εμφανιστεί στην LCD οθόνη.

Εντολές Επικοινωνίας

Οι εντολές αυτές επιτρέπουν την μεταφορά δεδομένων μέσω της σειριακής θύρας προς ή από έναν άλλο υπολογιστή που επίσης έχει σειριακή θύρα και τρέχει το ίδιο ή κατάλληλο software (όπως το PCV3 για PC). Η βασική ευκολία που προσφέρουν οι εντολές αυτές είναι η μεταφορά προγραμμάτων από το BGC-8088 προς ένα PC, μέσω της σειριακής θύρας, για αποθήκευση στο PC, αλλά και αντίστροφα, δηλαδή η μεταφορά προγραμμάτων από το PC προς το BGC-8088 για τοποθέτηση στη μνήμη και εκτέλεση. Έτσι ο χρήστης μπορεί να αποθηκεύει σε PC τα προγράμματα που αναπτύσσει στο BGC-8088, αλλά και να εκτελεί προγράμματα στο BGC-8088 χωρίς να χρειάζεται να τα πληκτρολογήσει (μεταφέροντάς τα από το PC). Οι εντολές αυτές είναι (κατά αλφαβητική σειρά) :

L (downLoad data) : Μεταφέρει δεδομένα από το PC προς το BGC-8088 και τα αποθηκεύει στην μνήμη του BGC-8088.

Z (upload data) : Μεταφέρει δεδομένα από το BGC-8088 προς το PC όπου και αποθηκεύονται με τη μορφή αρχείων.

Εντολή L (downLoad data)

Σύνταξη Εντολής : L [<Διεύθυνση>]

Η εντολή αυτή επιτρέπει την μεταφορά δεδομένων από PC στο BGC-8088 μέσω της σειριακής θύρας. Το PC θα πρέπει να τρέχει κατάλληλο software όπως το PCV3.COM για να γίνει εφικτή η επικοινωνία των δύο μηχανημάτων. Τα δεδομένα στο PC (π.χ. κώδικας μηχανής 8088) βρίσκονται σε μορφή αρχείων binary. Τα δεδομένα θα αποθηκευτούν στον BGC-8088 στην περιοχή μνήμης που ξεκινά από την <Διεύθυνση> που δόθηκε. Αν δεν δοθεί <Διεύθυνση> τότε τα δεδομένα φορτώνονται στην εξ'ορισμού διεύθυνση 0100:0000. Πρέπει εδώ να σημειωθεί ότι θα πρέπει το αρχείο binary στο PC να έχει κατάλληλο μέγεθος ώστε να χωράει στην μνήμη του BGC-8088 από την <Διεύθυνση> αρχής και μετά. Μόλις εκτελέσουμε την εντολή τότε εμφανίζεται στην οθόνη το εξής :

TRANSFER DATA FROM HOST ----->
TO: 0100:0000
STRIKE <CR> WHEN READY ……

Σε αυτό το σημείο θα πρέπει να τρέξουμε το πρόγραμμα PCV3.COM στο PC (αφού βέβαια έχουμε συνδέσει το σειριακό καλώδιο) και να επιλέξουμε το αρχείο που θα μεταφέρουμε. Τότε πατάμε πρώτα το πλήκτρο CR στον BGC-8088 ώστε να μπεί σε κατάσταση αναμονής λήψης, οπότε και εμφανίζει τη φράση :

DOWN LOAD ……

και αμέσως μετά πατάμε το τελευταίο πλήκτρο στο πρόγραμμα PCV3 ώστε να αρχίσει η μεταφορά των δεδομένων. Όταν ολοκληρωθεί η μεταφορά των δεδομένων, το πρόγραμμα MONITOR θα εμφανίσει τη φράση :

TRANSFER xxxxxH BYTES

Όπου «xxxxx» είναι το πλήθος των bytes που ελήφθησαν από τον BGC-8088.

Εντολή Z (upload data)

Σύνταξη Εντολής : Z [<Περιοχή Διεθύνσεων>]

Η εντολή αυτή επιτρέπει την μεταφορά δεδομένων από το BGC-8088 στο PC μέσω της σειριακής θύρας. Το PC θα πρέπει να τρέχει κατάλληλο software όπως το PCV3.COM

για να γίνει εφικτή η επικοινωνία των δύο μηχανημάτων. Η εντολή 'Z' μεταφέρει στο PC τα δεδομένα που βρίσκονται στην <Περιοχή Διευθύνσεων> που δίνεται ως παράμετρος. Αν δεν δοθεί <Περιοχή Διευθύνσεων> τότε η εντολή 'Z' θα μεταφέρει στο PC 100_H bytes που ξεκινούν από την εξ'ορισμού διεύθυνση 0100:0000. Τα δεδομένα θα αποθηκευτούν στο PC με την μορφή αρχείου binary, το όνομα του οποίου καθορίζει ο χρήστης. Μόλις εκτελέσουμε την εντολή τότε εμφανίζεται στην οθόνη το εξής :

TRANSFER DATA TO HOST ----->
FROM: 0100:0000 LENGTH: 00100H
STRIKE <CR> WHEN READY ……

Σε αυτό το σημείο θα πρέπει να τρέξουμε το πρόγραμμα PCV3.COM στο PC (αφού βέβαια έχουμε συνδέσει το σειριακό καλώδιο) να επιλέξουμε το όνομα του αρχείου στο οποίο θα σωθούν τα δεδομένα και να πατήσουμε το τελικό πλήκτρο ώστε το PC να μπεί σε κατάσταση αναμονής λήψης. Τότε πατάμε το CR στο BGC-8088 οπότε και εμφανίζεται η φράση :

UP LOAD ……

και αρχίζει η μεταφορά των δεδομένων. Όταν ολοκληρωθεί η μεταφορά των δεδομένων, το πρόγραμμα MONITOR θα εμφανίσει τη φράση :

TRANSFER COMPLETELY

Κεφάλαιο 3 : Οι εντολές της γλώσσας Assembly του 8088

Το σετ εντολών του 8088 περιέχει 90 εντολές οι οποίες μπορούν να χωριστούν σε 6 κατηγορίες που είναι :

1. Εντολές Μεταφοράς δεδομένων (Data Transfer Commands)
2. Εντολές Αριθμητικών Πράξεων (Arithmetic Commands)
3. Εντολές Λογικών Πράξεων (Logic Commands)
4. Εντολές χειρισμού αλφαριθμητικών (String Manipulation Commands)
5. Εντολές Ελέγχου Ροής Προγράμματος (Program Flow Control Commands)
6. Εντολές Ελέγχου του Επεξεργαστή (Processor Control Commands)

Οι εντολές των 6 αυτών κατηγοριών αναλύονται στη συνέχεια :

Εντολές Μεταφοράς δεδομένων (Data Transfer Commands)

Εντολή **MOV** (Move – Μεταφορά δεδομένων μεταξύ καταχωρητών και μνήμης)
Σύνταξη Εντολής :

MOV καταχωρητής1, καταχωρητής2

Αντιγράφει τα περιεχόμενα του καταχωρητή 2 στον καταχωρητή 1. Παράδειγμα :
MOV AX,BX
(βάλε στον AX το περιεχόμενο του BX, δηλαδή AX=BX)

MOV καταχωρητής, [θέση μνήμης]

Αντιγράφει τα περιεχόμενα της θέσης μνήμης στον καταχωρητή. Παράδειγμα :
MOV AX,[200]
(βάλε στον AX το περιεχόμενο της θέσης μνήμης 200 και 201 ή καλύτερα 0100:0200 και 0100:0201, όπου 0100 η τιμή του καταχωρητή DS)

MOV [θέση μνήμης], καταχωρητής

Αντιγράφει τα περιεχόμενα του καταχωρητή στη θέση μνήμης. Παράδειγμα:
MOV [200],CX
(βάλε στη θέση μνήμης 200 και 201 ή καλύτερα 0100:0200 και 0100:0201, το περιεχόμενο του καταχωρητή CX, όπου 0100 η τιμή του καταχωρητή DS)

MOV καταχωρητης, τιμή

Βάζει στον καταχωρητή την τιμή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:
MOV DX, 1AF4
(βάλε στον καταχωρητή DX την τιμή 1AF4)

MOV BY/WO[διεύθυνση μνήμης], τιμή

Βάζει στην διεύθυνση μνήμης την τιμή που δώσαμε. Παράδειγμα:
MOV BY[200], 7A
(βάλε στην θέση μνήμης 0200 τον αριθμό 7A. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα μεταφερθεί). Αν δώσουμε:
MOV WO[200],7A
τότε θα μπούν στις θέσεις μνήμης 200 και 201 ή 0100:0200 και 0100:0201 τα 2 byte του αριθμού 007A, δηλαδή το 7A στην 200 και το 00 στη 201).

Εντολή **PUSH** (Push – Αποθήκευση στη στοίβα)
Σύνταξη Εντολής :

PUSH καταχωρητής

Αντιγράφει τα περιεχόμενα του καταχωρητή (2 bytes) στην στοίβα, ενημερώνοντας τον καταχωρητή SP. Ο καταχωρητής που η τιμή του πηγαίνει στην στοίβα, μπορεί να είναι και καταχωρητής τμήματος (segment register). Παράδειγμα :

PUSH AX

(βάλε στην στοίβα τον AX)

PUSH [θέση μνήμης]

Αντιγράφει τα περιεχόμενα της θέσης μνήμης και της επόμενης (2 bytes) στην στοίβα, ενημερώνοντας τον καταχωρητή SP. Παράδειγμα:

PUSH [200]

(βάλε στην στοίβα το περιεχόμενο της θέσης μνήμης 200 και 201 ή 0100:0200 και 0100:0201).

Εντολή **PUSHF** (Push Flags – Αποθήκευση σημαιών στη στοίβα)

Σύνταξη Εντολής :

PUSHF

Αντιγράφει τα περιεχόμενα του καταχωρητή σημαιών (Flags Register – FG) (2 bytes) στην στοίβα, ενημερώνοντας τον καταχωρητή SP.

Παράδειγμα:

PUSHF

(βάλε στην στοίβα τον καταχωρητή σημαιών FG)

Εντολή **POP** (Pop – Ανάκτηση από τη στοίβα)

Σύνταξη Εντολής :

POP καταχωρητής

Ανακτά από την στοίβα τα περιεχόμενα του καταχωρητή (2 bytes), ενημερώνοντας τον καταχωρητή SP. Ο καταχωρητής που η τιμή του ανακτάται από την στοίβα, μπορεί να είναι και καταχωρητής τμήματος (segment register). Παράδειγμα :

POP BX

(ανέκτησε από την στοίβα την τιμή του BX)

POP [θέση μνήμης]

Ανακτά από την στοίβα τα περιεχόμενα μίας θέσης μνήμης, και της επόμενης (2 bytes), ενημερώνοντας τον καταχωρητή SP. Παράδειγμα:

POP [200]

(ανέκτησε από την στοίβα το περιεχόμενο της θέσης μνήμης 200 και 201 ή 0100:0200 και 0100:0201).

Εντολή **POPF** (Pop Flags – Ανάκτηση Σημαιών από τη στοίβα)

Σύνταξη Εντολής :

POPF

Ανακτά από την στοίβα τα περιεχόμενα του καταχωρητή σημαιών (Flags Register – FG) (2 bytes), ενημερώνοντας τον καταχωρητή SP.

Παράδειγμα:

POPF

(ανέκτησε από την στοίβα τον καταχωρητή σημαιών FG)

Εντολή **XCHG** (Exchange – Ανταλλαγή δεδομένων μεταξύ καταχωρητών και μνήμης)
Σύνταξη Εντολής :
XCHG καταχωρητής1, καταχωρητής 2

Ανταλλάσσει τα δεδομένα των δύο καταχωρητών. Παράδειγμα:
XCHG CX,DX
(ανταλλάσσει τις τιμές των καταχωρητών CX και DX έτσι ώστε μετά την εντολή ο CX έχει την τιμή που είχε ο DX και ο DX έχει την τιμή που είχε ο CX)

XCHG καταχωρητής , [διεύθυνση μνήμης] ή
XCHG [διεύθυνση μνήμης] , καταχωρητής

Ανταλλάσσει τα δεδομένα του καταχωρητή και της θέσης μνήμης. Αν ο καταχωρητής είναι των 16 bit τότε 2 bytes ανταλλάσσονται. Αν ο καταχωρητής είναι των 8 bit τότε 1 byte ανταλλάσσεται. Παράδειγμα:
XCHG AX,[200]
(ανταλλάσσει την τιμη του καταχωρητή AX – 2 bytes με το περιεχόμενο της θέσης μνήμης 200 και 201 ή 0100:0200 και 0100:0201 έτσι ώστε μετά την εντολή ο AX έχει την τιμή που είχε η μνήμη 200 και 201 και η μνήμη 200 και 201 έχει την τιμή που είχε ο AX)

Εντολή **IN** (Input from port – Είσοδος δεδομένων από θύρα εισόδου)
Σύνταξη Εντολής :
IN AL,<αριθμός θύρας>

Διαβάζει το byte που βρίσκεται στην θύρα εισόδου που δίνουμε και το τοποθετεί στον καταχωρητή AL. Χρησιμοποιείται όταν έχουμε μέχρι 256 θύρες I/O στο σύστημα. Παράδειγμα :
IN AL, 5C
(Διαβάζει την τιμή της θύρας 5C).

IN AL,<Καταχωρητής DX που περιέχει αριθμό θύρας>

Διαβάζει το byte που βρίσκεται στην θύρα εισόδου που δίνουμε ως τιμή του καταχωρητή DX και το τοποθετεί στον καταχωρητή AL. Χρησιμοποιείται όταν έχουμε μέχρι 65536 θύρες I/O στο σύστημα (όπως στα PC και τον BGC-8088). Παράδειγμα :
MOV DX, FF6D
IN AL, DX
(Διαβάζει την τιμή της θύρας FF6D που αντιστοιχεί στην τιμή του Line Status Register του chip 8250 που υλοποιεί την σειριακή θύρα RS-232. Η τιμή αυτή είναι συνήθως 60 που σημαίνει ότι τα bits 5 και 6 του καταχωρητή κατάστασης είναι 1, δηλαδή ο 8250 είναι έτοιμος να δεχτεί δεδομένα και ότι ο καταχωρητής αποστολής δεδομένων είναι άδειος).

Εντολή **OUT** (Output to port – Έξοδος δεδομένων προς θύρα Εξόδου)
Σύνταξη Εντολής :
OUT <αριθμός θύρας>,AL

Εξάγει το byte που βρίσκεται στον καταχωρητή AL στην θύρα εξόδου που δίνουμε. Χρησιμοποιείται όταν έχουμε μέχρι 256 θύρες I/O στο σύστημα. Παράδειγμα :
OUT A4, AL
(Εξάγει το περιεχόμενο του καταχωρητή AL στην θύρα A4).

OUT <Καταχωρητής DX που περιέχει αριθμό θύρας>, AL

Εξάγει το byte που βρίσκεται στο καταχωρητή AL στην θύρα εξόδου που δίνουμε ως τιμή του καταχωρητή DX. Χρησιμοποιείται όταν έχουμε μέχρι 65536 θύρες I/O στο σύστημα (όπως στα PC και τον BGC-8088). Παράδειγμα :
MOV AL, FE
MOV DX, FF70
OUT DX, AL
(Εξάγει το νούμερο FE στην θύρα εξόδου FF70 που αντιστοιχεί στον καταχωρητή κατάστασης συστήματος (LED πάνω από το πληκτρολόγιο και μεγαφωνάκι). Ο αριθμός FF_{16} είναι ο 11111110_2 και επομένως κάνει το bit 0 ίσο με 0 που σημαίνει ότι ανάβει το LED του Caps Lock.).

Εντολή **XLAT** (Translate byte to AL – Μετατροπή του AL βάσει πίνακα αντιστοίχισης)
Σύνταξη Εντολής :
XLAT

Εντοπίζει ένα byte σε πίνακα στην μνήμη, την διεύθυνση βάσης του οποίου κρατά ο καταχωρητής BX (DS:BX) ενώ η θέση στον πίνακα καθορίζεται από τον καταχωρητή AL. Το byte του πίνακα καταχωρείται πάλι στον AL
Παράδειγμα :
MOV BX,0200
MOV AL,05
XLAT
(βάζει στον AL το 5ο στοιχείο του πίνακα που αρχίζει στην διεύθυνση 0200).

Εντολή **LEA** (Load effective address to register – Φόρτωσε τελική διεύθυνση)
Σύνταξη Εντολής :
LEA καταχωρητής, [διεύθυνση]

Βάζει στον καταχωρητή την διεύθυνση που δίνουμε. Η διεύθυνση μπορεί να εκφραστεί με έναν από 24 διαφορετικούς τρόπους (βλέπε Τέλος Κεφαλαίου – Τρόποι Διευθυνσιοδότησης Μνήμης)
Παράδειγμα :
LEA AX, [200]
(βάλε στον καταχωρητή AX την διεύθυνση 0200. Ισοδυναμεί με MOV AX,200)
LEA BX, [BP+DI+1234]
(βάλε στον καταχωρητή BX την διεύθυνση που προκύπτει αν στο περιεχόμενο του BP προσθέσουμε το περιεχόμενο του DI και σε αυτό τον αριθμό μετατόπισης 1234_{16}).

Εντολή **LDS** (Load pointer to DS register – Φόρτωσε δείκτη στον καταχωρητή DS)
Σύνταξη Εντολής :
LDS καταχωρητής, [διεύθυνση]

Βάζει στον καταχωρητή τα 2 πρώτα byte που βρίσκονται στην διεύθυνση που δώσαμε (offset) και τα 2 επόμενα byte τα βάζει στον καταχωρητή DS (segment). Η διεύθυνση μπορεί να εκφραστεί με έναν από 24 διαφορετικούς τρόπους (βλέπε Τέλος Κεφαλαίου – Τρόποι Διευθυνσιοδότησης Μνήμης)
Παράδειγμα :
0100:0200 FF A9 3C 7B …….
…..
LDS AX, [200]
(βάλε στον καταχωρητή AX τα byte FF και A9 (A9FF) και στον DS τα byte 3C και 7B (7B3C)).

Εντολή **LES** (Load pointer to ES register – Φόρτωσε δείκτη στον καταχωρητή ES)
Σύνταξη Εντολής :
LES καταχωρητής, [διεύθυνση]

Βάζει στον καταχωρητή τα 2 πρώτα byte που βρίσκονται στην διεύθυνση που δώσαμε (offset) και τα 2 επόμενα byte τα βάζει στον καταχωρητή ES (segment). Η διεύθυνση μπορεί να εκφραστεί με έναν από 24 διαφορετικούς τρόπους (βλέπε Τέλος Κεφαλαίου – Τρόποι Διευθυνσιοδότησης Μνήμης)
Παράδειγμα :
0100:0200 FF A9 3C 7B …….
…..
LES AX, [200]
(βάλε στον καταχωρητή AX τα byte FF και A9 (A9FF) και στον ES τα byte 3C και 7B (7B3C)).

Εντολή **LAHF** (Load AH with Flags – Φόρτωσε τις σημαίες στον AH)
Σύνταξη Εντολής :
LAHF

Βάζει στον καταχωρητή AH, δηλαδή στο υψηλής τάξης byte του AX τα πρώτα 8 bits του καταχωρητή σημαιών. Σε αυτά περιλαμβάνονται οι σημαίες Carry (CY), Parity (PF), Auxiliary Carry (AF), Zero (ZF), και Sign (SF). Δεν περιλαμβάνονται οι σημαίες Trap (TF), Interrupt (IF), Direction (DF) και Overflow (OF). Παράδειγμα :
LAHF
(βάλε στον καταχωρητή AH το πρώτο byte του καταχωρητή σημαιών).

Εντολή **SAHF** (Store AH into Flags – Αντέγραψε τον AH στις σημαίες)
Σύνταξη Εντολής :
SAHF

Αντιγράφει τον καταχωρητή AH, δηλαδή το υψηλής τάξης byte του AX στα πρώτα 8 bits του καταχωρητή σημαιών. Σε αυτά περιλαμβάνονται οι σημαίες Carry (CY), Parity (PF), Auxiliary Carry (AF), Zero (ZF), και Sign (SF). Δεν περιλαμβάνονται οι σημαίες Trap (TF), Interrupt (IF), Direction (DF) και Overflow (OF). Παράδειγμα :
SAHF
(αντέγραψε τον καταχωρητή AH στο πρώτο byte του καταχωρητή σημαιών).

Εντολές Αριθμητικών Πράξεων (Arithmetic Commands)

Εντολή **ADD** (Add – Πρόσθεση χωρίς κρατούμενο)

Σύνταξη Εντολής :

ADD καταχωρητής1, καταχωρητής2

Αθροίζει τα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2 χωρίς να συμπεριλάβει τυχόν κρατούμενο (Carry) και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :

ADD AX,BX

(άθροισε τον AX και τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX+BX)

ADD καταχωρητής, [θέση μνήμης]

Αθροίζει τα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), χωρίς κρατούμενο, και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η πρόσθεση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

ADD AX,[200]

(πρόσθεσε τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)

Παράδειγμα 2ο :

ADD AL, [200]

(πρόσθεσε τον AL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

ADD [θέση μνήμης], καταχωρητής

Αθροίζει τα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), χωρίς κρατούμενο, και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη – low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η πρόσθεση γίνεται με 1 byte από τη μνήμη και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :

ADD [200], BX

(πρόσθεσε τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

ADD [200], CL

(πρόσθεσε τον CL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στην θέση μνήμης 200 (1 byte))

ADD καταχωρητης, τιμή

Αθροίζει τον καταχωρητή και την τιμή που δίνουμε, χωρίς κρατούμενο, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους

καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

ADD DX, 1AF4

(άθροισε ττην τιμή του καταχωρητή DX και την τιμή 1AF4 και βάλε το αποτέλεσμα στον DX)

ADD BY/WO[διεύθυνση μνήμης], τιμή

Αθροίζει το περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

ADD BY[200], 7A

(πρόσθεσε το περιεχόμενο της θέσης μνήμης 0200 και τον αριθμό 7A και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα προστεθεί). Αν δώσουμε:

ADD WO[200],7A

τότε θα προστεθούν τα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 007A, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **ADC** (Add with Carry – Πρόσθεση με κρατούμενο)

Σύνταξη Εντολής :

ADC καταχωρητής1, καταχωρητής2

Αθροίζει τα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2 και το κρατούμενο (Carry) και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :

ADC DX,CX

(άθροισε τον DX και τον CX και τυχόν κρατούμενο και βάλε το αποτέλεσμα στον DX, δηλαδή DX=DX+CX+Carry)

ADC καταχωρητής, [θέση μνήμης]

Αθροίζει τα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), με κρατούμενο, και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η πρόσθεση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

ADC AX,[200]

(πρόσθεσε τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, με κρατούμενο, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)

Παράδειγμα 2ο :

ADC BL, [200]

(πρόσθεσε τον BL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, με κρατούμενο, και βάλε το αποτέλεσμα στον BL, όπου 0100 η τιμή του καταχωρητή DS)

ADC [θέση μνήμης], καταχωρητής

Αθροίζει τα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), με κρατούμενο, και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη, low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η πρόσθεση

γίνεται με 1 byte από τη μνήμη και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :

ADC [200], BX

(πρόσθεσε τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, με κρατούμενο, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

ADC [200], DL

(πρόσθεσε τον DL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, με κρατούμενο, και βάλε το αποτέλεσμα στην θέση μνήμης 200)

ADC καταχωρητης, τιμή

Αθροίζει τον καταχωρητή και την τιμή που δίνουμε, με κρατούμενο, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

ADC BP, 2B68

(άθροισε ττην τιμή του καταχωρητή BP και την τιμή 2B68 και βάλε το αποτέλεσμα στον BP)

ADC BY/WO[διεύθυνση μνήμης], τιμή

Αθροίζει το περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε και το κρατούμενο, και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

ADD BY[300], F0

(πρόσθεσε το περιεχόμενο της θέσης μνήμης 0300 και τον αριθμό F0 και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 300. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα προστεθεί). Αν δώσουμε:

ADD WO[300], F0

τότε θα προστεθούν τα περιεχόμενα των θέσεων μνήμης 300 και 301 ή 0100:0300 και 0100:0301, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 00F0, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **INC** (Increment - Αύξηση)

Σύνταξη Εντολής :

INC καταχωρητής

Αυξάνει την τιμή του καταχωρητή κατά 1. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

INC SI

(αύξησε την τιμή του καταχωρητή SI κατά 1)

Εντολή **AAA** (ASCII Adjust for Add - Προσαρμογή δεκαδικής πρόσθεσης upacked BCD)

Σύνταξη Εντολής :

AAA

Διορθώνει το αποτέλεσμα στον καταχωρητή AL από μία προηγούμενη πρόσθεση unpacked BCD αριθμών. Τα περιεχόμενα του AL μεταβάλλονται ωστε να περιέχουν ένα δεκαδικό ψηφίο 0..9 (unpacked BCD). Αν υπάρξει δεκαδική υπερχείλιση στον AL, αυξάνει ο AH κατά 1. Παράδειγμα:

```
MOV AX,09
MOV CX,01
ADD AX,CX  (αποτέλεσμα AL=0A)
AAA        (αποτέλεσμα AL=00, αυξάνει ο AH σε 01, οπότε AH-AL=01-00
           δηλαδή αποτέλεσμα σε unpacked BCD = 10  )
```

Εντολή **DAA** (Decimal Adjust for Add – Προσαρμογή δεκαδικής πρόσθεσης packed BCD)

Σύνταξη Εντολής :

DAA

Διορθώνει το αποτέλεσμα στον καταχωρητή AL από μία προηγούμενη πρόσθεση packed BCD αριθμών. Τα περιεχόμενα του AL μεταβάλλονται σε ένα ζευγάρι δεκαδικών ψηφίων που το καθένα είναι 0..9 (packed BCD). Αν υπάρξει δεκαδική υπερχείλιση στον AL, ανάβει το Carry. Παράδειγμα:

```
MOV AX,19
MOV CX,01
ADD AX,CX  (αποτέλεσμα AL=1A)
DAA        (αποτέλεσμα AL=20, δεν ανάβει το Carry)
Παράδειγμα 2
MOV AX,99
MOV CX,01
ADD AX,CX  (αποτέλεσμα AL=9A)
DAA        (αποτέλεσμα AL=00, ανάβει το carry)
```

Εντολή **SUB** (Subtract – Αφαίρεση χωρίς δανεικό)

Σύνταξη Εντολής :

SUB καταχωρητής1, καταχωρητής2

Αφαιρεί από το περιεχόμενο του καταχωρητή1 το περιεχόμενο του καταχωρητή2 χωρίς να συμπεριλάβει τυχόν δανεικό (Carry-Borrow) και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :

```
SUB AX,BX
```

(αφαίρεσε από τον AX τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX-BX)

SUB καταχωρητής, [θέση μνήμης]

Αφαιρεί από το περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης (2 byte), χωρίς δανεικό, και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η αφαίρεση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

```
SUB AX,[200]
```

(αφαίρεσε από τον AX το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στον

AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)

Παράδειγμα 2ο :

SUB AL, [200]

(αφαίρεσε από τον AL το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

SUB [θέση μνήμης], καταχωρητής

Αφαιρεί από το περιεχόμενο της θέσης μνήμης (2 byte), το περιεχόμενο του καταχωρητή, χωρίς δανεικό, και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη – low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η αφαίρεση γίνεται με 1 byte από τη μνήμη και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :

SUB [200], BX

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, το περιεχόμενο του καταχωρητή BX, χωρίς δανεικό, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

SUB [200], CL

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, το περιεχόμενο του καταχωρητή CL, χωρίς δανεικό, και βάλε το αποτέλεσμα στην θέση μνήμης 200 – 1 byte)

SUB καταχωρητης, τιμή

Αφαιρεί από τον καταχωρητή την τιμή που δίνουμε, χωρίς δανεικό, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

SUB DX, 1AF4

(αφαιρεί από την τιμή του καταχωρητή DX την τιμή 1AF4 και βάζει το αποτέλεσμα στον DX)

SUB BY/WO[διεύθυνση μνήμης], τιμή

Αφαιρεί από το περιεχόμενο της διεύθυνσης μνήμης την τιμή που δώσαμε και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

SUB BY[200], 7A

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 0200 τον αριθμό 7A και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα αφαιρεθεί). Αν δώσουμε:

SUB WO[200],7A

τότε θα αφαιρεθούν από τα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού, ο αριθμός 007A, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **SBB** (Subtract with Borrow – Αφαίρεση με δανεικό)

Σύνταξη Εντολής :

SBB καταχωρητής1, καταχωρητής2

Αφαιρεί από το περιεχόμενο του καταχωρητή1 το περιεχόμενο του καταχωρητή2 συμπεριλάμβάνοντας και τυχόν δανεικό (Carry-Borrow) και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :

SBB AX,BX

(αφαίρεσε από τον AX τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX-BX-Carry)

SBB καταχωρητής, [θέση μνήμης]

Αφαιρεί από το περιεχόμενο του καταχωρητή το περιεχόμενο της θέσης μνήμης (2 byte), με δανεικό, και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η αφαίρεση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

SBB AX,[200]

(αφαίρεσε από τον AX το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, με δανεικό, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)

Παράδειγμα 2ο :

SBB AL, [200]

(αφαίρεσε από τον AL το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, με δανεικό, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

SBB [θέση μνήμης], καταχωρητής

Αφαιρεί από το περιεχόμενο της θέσης μνήμης (2 byte), το περιεχόμενο του καταχωρητή, με δανεικό, και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη – low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η αφαίρεση γίνεται με 1 byte από τη μνήμη και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :

SBB [200], BX

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, το περιεχόμενο του καταχωρητή BX, με δανεικό, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

SBB [200], CL

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, το περιεχόμενο του καταχωρητή CL, με δανεικό, και βάλε το αποτέλεσμα στην θέση μνήμης 200 – 1 byte)

SBB καταχωρητης, τιμή

Αφαιρεί από τον καταχωρητή την τιμή που δίνουμε, με δανεικό, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

SBB DX, 1AF4

(αφαιρεί από την τιμή του καταχωρητή DX την τιμή 1AF4, με δανεικό, και βάζει το αποτέλεσμα στον DX)

SBB BY/WO[διεύθυνση μνήμης], τιμή

Αφαιρεί από το περιεχόμενο της διεύθυνσης μνήμης την τιμή που δώσαμε, με δανεικό, και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

SBB BY[200], 7A

(αφαίρεσε από το περιεχόμενο της θέσης μνήμης 0200 τον αριθμό 7A, με δανεικό, και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα αφαιρεθεί). Αν δώσουμε:

SBB WO[200],7A

τότε θα αφαιρεθούν από τα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού, ο αριθμός 007A, με δανεικό, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **DEC** (Decrement - Μείωση)

Σύνταξη Εντολής :

DEC καταχωρητής

Μειώνει την τιμή του καταχωρητή κατά 1. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

DEC DI

(μείωσε την τιμή του καταχωρητή DI κατά 1)

Εντολή **NEG** (Negative - Αρνητικό)

Σύνταξη Εντολής :

NEG καταχωρητής

Αλλάζει πρόσημο στην τιμή του καταχωρητή με βάση το συμπλήρωμα ως προς 2. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

NEG BX

(αλλάζει πρόσημο στην τιμή του καταχωρητή BX, δηλ. BX=-BX ή BX=0-BX)

Εντολή **CMP** (Compare - Σύγκριση)

Σύνταξη Εντολής :

CMP καταχωρητής1, καταχωρητής2

Συγκρίνει τα περιεχόμενα του καταχωρητή1 και του καταχωρητή2 επηρεάζοντας τις σημαίες (flags). Στην ουσία πραγματοποιεί την αφαίρεση (καταχωρητής1-καταχωρητής2) χωρίς όμως να αλλοιώνει το περιεχόμενο του καταχωρητή 1 όπως οι εντολές αφαίρεσης (SUB,SBB). Παράδειγμα :

CMP AX,BX

(σύγκρινε τα περιεχόμενα των καταχωρητών AX και BX επηρεάζοντας τις σημαίες.)

Παρακάτω δίνονται παραδείγματα τιμών των AX,BX και σημαιών που προκύπτουν με την σύγκριση CMP :

A/A	AX	BX	Σημαίες
1	0002	0001	-
2	0004	0001	Parity
3	0001	0001	Zero, Parity
4	0001	0002	Carry, AuxCarry, Negative, Parity
5	0000	0002	Carry, AuxCarry, Negative

CMP καταχωρητής, [θέση μνήμης]

Συγκρίνει το περιεχόμενο του καταχωρητή και το περιεχόμενο της θέσης μνήμης (2 byte), επηρεάζοντας τις σημαίες, κάνοντας εικονική αφαίρεση (καταχωρητής-μνήμη). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η σύγκριση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

CMP AX,[200]

(σύγκρινε τον AX το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201)

Παράδειγμα 2ο :

CMP AL, [200]

(Σύγκρινε τον AL με το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200)

CMP [θέση μνήμης], καταχωρητής

Συγκρίνει το περιεχόμενο της θέσης μνήμης (2 byte) και του καταχωρητή, επηρεάζοντας τις σημαίες, κάνοντας εικονική αφαίρεση (μνήμη-καταχωρητής). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η σύγκριση γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

CMP [200],AX

(σύγκρινε το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και το περιεχόμενο του καταχωρητή AX)

Παράδειγμα 2ο :

CMP [200],AL

(Σύγκρινε το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και του AL)

CMP καταχωρητης, τιμή

Συγκρινει το περιεχόμενο του καταχωρητή με την τιμή που δίνουμε, επηρεάζοντας τις σημαίες. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

CMP DX, 1AF4(Συγκρίνει τον DX με την τιμή 1AF4)

CMP BY/WO[διεύθυνση μνήμης], τιμή

Συγκρίνει το περιεχόμενο της διεύθυνσης με την τιμή που δώσαμε, επηρεάζοντας τις σημαίες. Παράδειγμα:

CMP BY[200], 7A

(Συγκρίνει το περιεχόμενο της θέσης μνήμης 0200 με τον αριθμό 7A. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα συγκριθεί). Αν δώσουμε:

CMP WO[200],7A

τότε θα συγκριθούν τα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού, με τον αριθμό 007A).

Εντολή **AAS** (ASCII Adjust for Subtract - Προσαρμογή δεκαδικής αφαίρεσης unpacked BCD)

Σύνταξη Εντολής :

AAS

Διορθώνει το αποτέλεσμα στον καταχωρητή AL από μία προηγούμενη αφαίρεση unpacked BCD αριθμών. Τα περιεχόμενα του AL μεταβάλλονται ώστε να περιέχουν ένα δεκαδικό ψηφίο 0..9. Αν προκύψει δεκαδικό δανεικό, τότε μειώνεται ο AH κατά 1. Παράδειγμα:

```
MOV AX, 0300
MOV BX, 0001
SUB AX,BX   (Αποτέλεσμα AL=FF, AH=02, AX=02FF)
AAS         (Αποτέλεσμα AL=09, μείωση του AH=1, AX=0109)
```

Εντολή **DAS** (Decimal Adjust for Subtract – Προσαρμογή δεκαδικής αφαίρεσης packed BCD)

Σύνταξη Εντολής :

DAS

Διορθώνει το αποτέλεσμα στον καταχωρητή AL από μία προηγούμενη αφαίρεση packed BCD αριθμών. Τα περιεχόμενα του AL μεταβάλλονται σε ένα ζευγάρι δεκαδικών ψηφίων. Παράδειγμα:

```
MOV AX, 35
MOV BX, 47
SUB AX,BX   (Αποτέλεσμα AL=EE)
DAS         (Αποτέλεσμα AL=88, και υπάρχει δανεικό, 88-100=-12)
```

Εντολή **MUL** (Multiply unsigned – Πολλαπλασιασμός αριθμών χωρίς πρόσημο)

Σύνταξη Εντολής :

MUL καταχωρητής

Πολλαπλασιάζει τον καταχωρητή AX με τον καταχωρητή που δίνουμε και βάζει το αποτέλεσμα στο 32-bit ζευγάρι καταχωρητών DX:AX, όπου ο DX κρατά το τμήμα υψηλότερης τάξης και ο AX το τμήμα χαμηλότερης ταξης του αποτελέσματος(DX:AX = AX * καταχωρητής). Αν ο καταχωρητής που δίνουμε είναι των 8-bit π.χ. BL τότε πολλαπλασιάζεται ο AL με αυτόν που δίνουμε και το αποτέλεσμα πηγαίνει στον AX (AX = AL * καταχωρητής). Ο πολλαπλασιασμός γίνεται θεωρώντας ότι οι αριθμοί δεν έχουν πρόσημο (unsigned). Παράδειγμα :

```
MUL BX
```

(πολλαπλασίασε τον AX με τον BX και βάλε το αποτέλεσμα στους DX:AX)

Παρακάτω δίνονται παραδείγματα τιμών των AX, BX και του αποτελέσματος του πολλαπλασιασμού τους με την MUL :

A/A	AX	BX	Αποτέλεσμα DX:AX
1	0002	0003	0000:0006 (2x3=6)
2	0002	0006	0000:000C (2x6=12)
3	0100	0100	0001:0000 (256x256=65536)
4	FFFF	FFFF	FFFE:0001 (65535x65535=4294836225)
5	FFFF	0002	0001:FFFE (65535x2=131070)

Εντολή **IMUL** (Integer Multiply signed – Πολλαπλασιασμός αριθμών με πρόσημο)
Σύνταξη Εντολής :
IMUL καταχωρητής

Πολλαπλασιάζει τον καταχωρητή AX με τον καταχωρητή που δίνουμε και βάζει το αποτέλεσμα στο 32-bit ζευγάρι καταχωρητών DX:AX, όπου ο DX κρατά το τμήμα υψηλότερης τάξης και ο AX το τμήμα χαμηλότερης τάξης του αποτελέσματος(DX:AX = AX * καταχωρητής). Αν ο καταχωρητής που δίνουμε είναι των 8-bit π.χ. BL τότε πολλαπλασιάζεται ο AL με αυτόν που δίνουμε και το αποτέλεσμα πηγαίνει στον AX (AX = AL * καταχωρητής). Ο πολλαπλασιασμός γίνεται θεωρώντας ότι οι αριθμοί είναι προσημασμένοι (signed). Παράδειγμα :
MUL BX
(πολλαπλασίασε τον AX με τον BX και βάλε το αποτέλεσμα στους DX:AX)
Παρακάτω δίνονται παραδείγματα τιμών των AX, BX και του αποτελέσματος του πολλαπλασιασμού τους με την IMUL :

A/A	AX	BX	Αποτέλεσμα DX:AX
1	0002	0003	0000:0006 (2x3=6)
2	0002	0006	0000:000C (2x6=12)
3	0100	0100	0001:0000 (256x256=65536)
4	FFFF	FFFF	0000:0001 (-1 x -1 = 1)
5	FFFF	0001	FFFF:FFFF (-1 x 1 = -1)
6	FFFF	0002	FFFF:FFFE (-1 x 2 = -2)

Εντολή **AAM** (ASCII Adjust for Multiply – Προσαρμογή δεκαδικού πολλαπλασιασμού unpacked BCD)
Σύνταξη Εντολής :
AAM

Διορθώνει το αποτέλεσμα στον καταχωρητή AX από έναν προηγούμενο πολλαπλασιασμό (MUL) unpacked BCD αριθμών. Τα περιεχόμενα του AX μεταβάλλονται ώστε να περιέχουν δύο δεκαδικά ψηφία 0..9 (unpacked BCD). Αν προκύψει δεκαδικό δανεικό, τότε μειώνεται ο AH κατά 1. Παράδειγμα:

```
MOV AX, 0004
MOV BX, 0003
MUL BX      (Αποτέλεσμα AX=000C)
AAM         (Αποτέλεσμα AX=0102)
```

Εντολή **DIV** (Divide unsigned – Διαίρεση αριθμών χωρίς πρόσημο)
Σύνταξη Εντολής :
DIV καταχωρητής

Διαιρεί το 32-bit ζευγάρι καταχωρητών DX:AX με τον καταχωρητή που δίνουμε και βάζει το πηλίκο στον AX και το υπόλοιπο στον DX (AX = DX:AX / καταχωρητής, DX=υπόλοιπο). Αν ο καταχωρητής που δίνουμε είναι των 8-bit π.χ. BL τότε διαιρείται ο AX με αυτόν που δίνουμε και το πηλίκο πηγαίνει στον AL, ενώ το υπόλοιπο στον AH (AL = AX / καταχωρητής, AH=υπόλοιπο). Η διαίρεση γίνεται θεωρώντας ότι οι αριθμοί δεν έχουν πρόσημο (unsigned). Παράδειγμα :

DIV BX

(διαίρεσε τους DX:AX με τον BX και βάλε το πηλίκο στον AX και το υπόλοιπο στον DX)

Παρακάτω δίνονται παραδείγματα τιμών των DX:AX, BX και του αποτελέσματος της διαίρεσής τους με την DIV :

A/A	DX:AX	BX	Πηλίκο AX, Υπόλοιπο DX	
1	0000:000C	0003	0004	0000 (12 / 3 = 4, υπόλ.=0)
2	0000:000D	0003	0004	0001 (13 / 3 = 4, υπόλ.=1)
3	0001:0000	0100	0100	0000 (65536 / 256=256, υπόλ.=0)
4	FFFE:0001	FFFF	FFFF	0000 (4294836225/65535=65535)
5	0001:FFFE	0002	FFFF	0000 (131070 / 2 = 65535)

Εντολή **IDIV** (Integer Divide signed – Διαίρεση αριθμών με πρόσημο)

Σύνταξη Εντολής :

IDIV καταχωρητής

Διαιρεί το 32-bit ζευγάρι καταχωρητών DX:AX με τον καταχωρητή που δίνουμε και βάζει το πηλίκο στον AX και το υπόλοιπο στον DX (AX = DX:AX / καταχωρητής, DX=υπόλοιπο). Αν ο καταχωρητής που δίνουμε είναι των 8-bit π.χ. BL τότε διαιρείται ο AX με αυτόν που δίνουμε και το πηλίκο πηγαίνει στον AL, ενώ το υπόλοιπο στον AH (AL = AX / καταχωρητής, AH=υπόλοιπο). Η διαίρεση γίνεται θεωρώντας ότι οι αριθμοί δεν έχουν πρόσημο (unsigned). Παράδειγμα :

DIV BX

(διαίρεσε τους DX:AX με τον BX και βάλε το πηλίκο στον AX και το υπόλοιπο στον DX)

Παρακάτω δίνονται παραδείγματα τιμών των DX:AX, BX και του αποτελέσματος της διαίρεσής τους με την IDIV :

A/A	DX:AX	BX	Πηλίκο AX, Υπόλοιπο DX	
1	0000:000C	0003	0004	0000 (12 / 3 = 4, υπόλ.=0)
2	0000:000D	0003	0004	0001 (13 / 3 = 4, υπόλ.=1)
3	0001:0000	0100	0100	0000 (65536 / 256=256, υπόλ.=0)
4	FFFF:FFFF	FFFF	0001	0000 (-1 / -1 = 1)
5	F000:0000	8000	2000	0000 (-268435456 / -32768 = 8192)

Εντολή **AAD** (ASCII Adjust for Divide – Προσαρμογή δεκαδικής διαίρεσης unpacked BCD)

Σύνταξη Εντολής :

AAD

Διορθώνει το αποτέλεσμα στον καταχωρητή AX αμέσως πρίν από μία επικείμενη διαίρεση (DIV) με έναν άλλο unpacked BCD αριθμό. Τα περιεχόμενα του AX μεταβάλλονται έτσι ώστε η επικείμενη διαίρεση να δώσει το σωστό αποτέλεσμα (unpacked BCD).

Εντολή **CBW** (Convert Byte to Word – Μετατροπή Byte σε Word)

Σύνταξη Εντολής :

CBW

Μετατρέπει τον προσημασμένο 8-bit αριθμό που βρίσκεται στον καταχωρητή AL σε προσημασμένο 16-bit αριθμό που αποθηκεύεται στον AX. Παραδείγματα μετατροπών με την εντολή CBW:

A/A	AL	AX	Σχόλια
1	01	0001	1 → 1
2	7F	007F	127 → 127
3	80	FF80	-128 → -128
4	D4	FFD4	-44 → -44
5	FF	FFFF	-1 → -1

Εντολή **CWD** (Convert Word to Double Word – Μετατροπή Word σε Double Word)
Σύνταξη Εντολής :
CWD

Μετατρέπει τον προσημασμένο 16-bit αριθμό που βρίσκεται στον καταχωρητή AX σε προσημασμένο 32-bit αριθμό που αποθηκεύεται στους καταχωρτές DX: AX. Παραδείγματα μετατροπών με την εντολή CWD:

A/A	AX	DX:AX	Σχόλια
1	0001	0000:0001	1 → 1
2	7FFF	0000:7FFF	32767 → 32767
3	8000	FFFF:8000	-32768 → -32768
4	C5A8	FFFF:C5A8	-14936 → -14936
5	FFFF	FFFF:FFFF	-1 → -1

Εντολές Λογικών Πράξεων (Logic Commands)
Εντολή **AND** (And – Λογικό ΚΑΙ)
Σύνταξη Εντολής :
AND καταχωρητής1, καταχωρητής2

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2 και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :
AND AX,BX
(κάνε λογικό AND τον AX και τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX AND BX)

AND καταχωρητής, [θέση μνήμης]

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη ΚΑΙ γίνεται με 1 byte από τη μνήμη. Παράδειγμα :
AND AX,[200]
(κάνε λογικό ΚΑΙ τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)
Παράδειγμα 2ο :
AND AL, [200]

(Κάνε λογικό ΚΑΙ στον AL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

AND [θέση μνήμης], καταχωρητής

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη, low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη ΚΑΙ γίνεται με 1 byte από τη μνήμη, και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :

AND [200], BX

(κάνε λογικό ΚΑΙ τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

AND [200], CL

(κάνε λογικό ΚΑΙ στον CL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στην θέση μνήμης 200 (1 byte))

AND καταχωρητης, τιμή

Πραγματοποιεί λογικό ΚΑΙ (AND) στον καταχωρητή και την τιμή που δίνουμε, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

AND DX, 1AF4

(κάνε λογικό ΚΑΙ στην τιμή του καταχωρητή DX και στην τιμή 1AF4 και βάλε το αποτέλεσμα στον DX)

AND BY/WO[διεύθυνση μνήμης], τιμή

Πραγματοποιεί λογικό ΚΑΙ (AND) στο περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

AND BY[200], 7A

(κάνε λογικό ΚΑΙ στο περιεχόμενο της θέσης μνήμης 0200 και τον αριθμό 7A και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα συμμετέχει στην πράξη). Αν δώσουμε:

AND WO[200],7A

τότε θα γίνει λογικό ΚΑΙ στα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 007A, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **OR** (Or – Λογικό Ή)

Σύνταξη Εντολής :

OR καταχωρητής1, καταχωρητής2

Πραγματοποιεί λογικό Ή (OR) στα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2 και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :
OR AX,BX
(κάνε λογικό OR τον AX και τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX OR BX)

OR καταχωρητής, [θέση μνήμης]

Πραγματοποιεί λογικό Ή (OR) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη OR γίνεται με 1 byte από τη μνήμη. Παράδειγμα :
OR AX,[200]
(κάνε λογικό OR τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)
Παράδειγμα 2ο :
OR AL, [200]
(Κάνε λογικό OR στον AL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

OR [θέση μνήμης], καταχωρητής

Πραγματοποιεί λογικό Ή (OR) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη, low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη OR γίνεται με 1 byte από τη μνήμη, και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :
OR [200], BX
(κάνε λογικό OR τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)
Παράδειγμα 2ο :
OR [200], CL
(κάνε λογικό OR στον CL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στην θέση μνήμης 200 (1 byte))

OR καταχωρητης, τιμή

Πραγματοποιεί λογικό Ή (OR) στον καταχωρητή και την τιμή που δίνουμε, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:
OR DX, 1AF4
(κάνε λογικό OR στην τιμή του καταχωρητή DX και στην τιμή 1AF4 και βάλε το αποτέλεσμα στον DX)

OR BY/WO[διεύθυνση μνήμης], τιμή

Πραγματοποιεί λογικό Ή (OR) στο περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:
OR BY[200], 7A
(κάνε λογικό OR στο περιεχόμενο της θέσης μνήμης 0200 και τον αριθμό 7A και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα συμμετέχει στην πράξη). Αν δώσουμε:
OR WO[200],7A
τότε θα γίνει λογικό OR στα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 007A, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **XOR** (Exclusive Or – Αποκλειστική Διάζευξη)
Σύνταξη Εντολής :
XOR καταχωρητής1, καταχωρητής2

Πραγματοποιεί λογικό Αποκλειστικό Ή (XOR) στα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2 και βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :
XOR AX,BX
(κάνε λογικό XOR τον AX και τον BX και βάλε το αποτέλεσμα στον AX, δηλαδή AX=AX XOR BX)

XOR καταχωρητής, [θέση μνήμης]

Πραγματοποιεί λογικό Αποκλειστικό Ή (XOR) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη XOR γίνεται με 1 byte από τη μνήμη. Παράδειγμα :
XOR AX,[200]
(κάνε λογικό XOR τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)
Παράδειγμα 2ο :
XOR AL, [200]
(Κάνε λογικό XOR στον AL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στον AL, όπου 0100 η τιμή του καταχωρητή DS)

XOR [θέση μνήμης], καταχωρητής

Πραγματοποιεί λογικό Αποκλειστικό Ή (XOR) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), και βάζει το αποτέλεσμα στη θέση μνήμης (και στην επόμενη, low-high bytes). Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη XOR γίνεται με 1 byte από τη μνήμη, και το αποτέλεσμα αποθηκεύεται σε 1 byte στην μνήμη. Παράδειγμα :
XOR [200], BX
(κάνε λογικό XOR τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, και βάλε το αποτέλεσμα

στην θέση μνήμης 200 και 201 (low-high bytes), όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 θα έχει το low byte του αποτελέσματος και η 201 το high byte)

Παράδειγμα 2ο :

XOR [200], CL

(κάνε λογικό XOR στον CL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, και βάλε το αποτέλεσμα στην θέση μνήμης 200 (1 byte))

XOR καταχωρητης, τιμή

Πραγματοποιεί λογικό Αποκλειστικό Ή (XOR) στον καταχωρητή και την τιμή που δίνουμε, και βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX,BX,CX,DX,BP,SI,DI,SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

XOR DX, 1AF4

(κάνε λογικό XOR στην τιμή του καταχωρητή DX και στην τιμή 1AF4 και βάλε το αποτέλεσμα στον DX)

XOR BY/WO[διεύθυνση μνήμης], τιμή

Πραγματοποιεί λογικό Αποκλειστικό Ή (XOR) στο περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε και βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

XOR BY[200], 7A

(κάνε λογικό XOR στο περιεχόμενο της θέσης μνήμης 0200 και τον αριθμό 7A και βάλε το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα συμμετέχει στην πράξη). Αν δώσουμε:

XOR WO[200],7A

τότε θα γίνει λογικό XOR στα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 007A, και το αποτέλεσμα θα μπεί στις θέσεις μνήμης 200 και 201, στην 200 το low byte και στην 201 το high byte).

Εντολή **TEST** (Test – Λογικό ΚΑΙ χωρίς καταχώρηση αποτελέσματος)

Σύνταξη Εντολής :

TEST καταχωρητής1, καταχωρητής2

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή1 και του καταχωρητή 2, επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάζει το αποτέλεσμα στον καταχωρητή 1. Παράδειγμα :

TEST AX,BX

(κάνε λογικό AND τον AX και τον BX, επηρεάζοντας τις σημαίες αλλά χωρίς να βάλεις το αποτέλεσμα στον AX, δηλαδή AX AND BX με αλλαγή των σημαιών)

TEST καταχωρητής, [θέση μνήμης]

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάζει το αποτέλεσμα στον καταχωρητή. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη ΚΑΙ γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

TEST AX,[200]

(κάνε λογικό ΚΑΙ τον AX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte) ή καλύτερα 0100:0200 και 0100:0201, επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάλεις το αποτέλεσμα στον AX, όπου 0100 η τιμή του καταχωρητή DS. Η θέση μνήμης 200 έχει το low byte και η 201 το high byte)

Παράδειγμα 2ο :

TEST AL, [200]

(Κάνε λογικό ΚΑΙ στον AL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200, όπου 0100 η τιμή του καταχωρητή DS)

TEST [θέση μνήμης], καταχωρητής

Πραγματοποιεί λογικό ΚΑΙ (AND) στα περιεχόμενα του καταχωρητή και της θέσης μνήμης (2 byte), επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάζει το αποτέλεσμα στη θέση μνήμης. Αν ο καταχωρητής είναι του ενός byte, π.χ. AL, τότε η λογική πράξη ΚΑΙ γίνεται με 1 byte από τη μνήμη. Παράδειγμα :

TEST [200], BX

(κάνε λογικό ΚΑΙ τον BX και το περιεχόμενο της θέσης μνήμης 200 και 201 (2 byte, low-high) ή καλύτερα 0100:0200 και 0100:0201, επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάλεις το αποτέλεσμα στην θέση μνήμης 200 και 201.)

Παράδειγμα 2ο :

TEST [200], CL

(κάνε λογικό ΚΑΙ στον CL και το περιεχόμενο της θέσης μνήμης 200 (1 byte) ή καλύτερα 0100:0200)

TEST καταχωρητης, τιμή

Πραγματοποιεί λογικό ΚΑΙ (AND) στον καταχωρητή και την τιμή που δίνουμε, επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάζει το αποτέλεσμα στον καταχωρητή. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα:

TEST DX, 1AF4

(κάνε λογικό ΚΑΙ στην τιμή του καταχωρητή DX και στην τιμή 1AF4 επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάλεις το αποτέλεσμα στον DX)

TEST BY/WO[διεύθυνση μνήμης], τιμή

Πραγματοποιεί λογικό ΚΑΙ (AND) στο περιεχόμενο της διεύθυνσης μνήμης και την τιμή που δώσαμε, επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάζει το αποτέλεσμα στην διεύθυνση μνήμης. Παράδειγμα:

TEST BY[200], 7A

(κάνε λογικό ΚΑΙ στο περιεχόμενο της θέσης μνήμης 0200 και τον αριθμό 7A επηρεάζοντας τις σημαίες, αλλά ΧΩΡΙΣ να βάλεις το αποτέλεσμα στην διεύθυνση μνήμης 200. Ο προσδιορισμός BY μπροστά από την διεύθυνση καθορίζει ότι ένα byte θα συμμετέχει στην πράξη). Αν δώσουμε:

TEST WO[200],7A

τότε θα γίνει λογικό ΚΑΙ στα περιεχόμενα των θέσεων μνήμης 200 και 201 ή 0100:0200 και 0100:0201, ως low και high bytes ενός 16bit αριθμού με τον αριθμό 007A).

Εντολή **NOT** (Not – Λογική Άρνηση)

Σύνταξη Εντολής :

NOT καταχωρητής

Αντιστρέφει τα bit του καταχωρητή, δηλαδή κάνει τα 0 να γίνουν 1 και τα 1 να γίνουν 0. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
NOT DX
(Αντέστρεψε τα bits του καταχωρητή DX)

Εντολή **SHL/SAL** (Shift Logical/Arithmetic Left – Μετακίνηση των bits αριστερά)
Σύνταξη Εντολής :
SHL/SAL καταχωρητής,1

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα αριστερά κατά 1 θέση. Ο καταχωρητής συμπληρώνεται από δεξιά με μηδενικά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
SHL BX,1
(Μετακίνησε τα bits του καταχωρητή BX κατά μια θέση προς τα αριστερά)

SHL/SAL καταχωρητής,CL

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα αριστερά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Ο καταχωρητής συμπληρώνεται από δεξιά με μηδενικά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
SHL BX,CL
(Μετακίνησε τα bits του καταχωρητή BX προς τα αριστερά, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **SHR** (Shift Logical Right – Λογική Μετακίνηση των bits Δεξιά)
Σύνταξη Εντολής :
SHR καταχωρητής,1

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα δεξιά κατά 1 θέση. Ο καταχωρητής συμπληρώνεται από αριστερά με μηδενικά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
SHR BX,1
(Μετακίνησε τα bits του καταχωρητή BX κατά μια θέση προς τα δεξιά)

SHR καταχωρητής,CL

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα δεξιά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Ο καταχωρητής συμπληρώνεται από αριστερά με μηδενικά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
SHR BX,CL
(Μετακίνησε τα bits του καταχωρητή BX προς τα δεξιά, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **SAR** (Shift Arithmetic Right – Αριθμητική Μετακίνηση των bits Δεξιά)
Σύνταξη Εντολής :
SAR καταχωρητής,1

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα δεξιά κατά 1 θέση. Το σημαντικότερο ψηφίο του αριθμού (most significant bit), που έχει σημασία προσήμου στους προσημασμένους αριθμούς, παραμένει. Χρησιμοποιείται για διαίρεση προσημασμένων αριθμών δια 2. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

SAR BX,1

(Μετακίνησε τα bits του καταχωρητή BX κατά μια θέση προς τα δεξιά, με διατήρηση προσήμου)

SAR καταχωρητής,CL

Μετακινεί (ολισθαίνει) τα bit του καταχωρητή προς τα δεξιά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL, με διατήρηση προσήμου. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

SHR BX,CL

(Μετακίνησε τα bits του καταχωρητή BX προς τα δεξιά, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **ROL** (Rotate Left – Περιστροφή των bits αριστερά)

Σύνταξη Εντολής :

ROL καταχωρητής,1

Περιστρέφει τα bit του καταχωρητή προς τα αριστερά κατά 1 θέση. Το bit που υπερχειλίζει επανεισάγεται από δεξιά και αντιγράφεται στο carry. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

ROL AX,1

(Περίστρεψε τα bits του καταχωρητή AX κατά μια θέση προς τα αριστερά)

ROL καταχωρητής,CL

Περιστρέφει τα bit του καταχωρητή προς τα αριστερά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Τα bit που υπερχειλίζουν επανεισάγονται από δεξιά και αντιγράφονται στο carry. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

ROL AX,CL

(Περίστρεψε τα bits του καταχωρητή AX προς τα αριστερά, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **ROR** (Rotate Right – Περιστροφή των bits Δεξιά)

Σύνταξη Εντολής :

ROR καταχωρητής,1

Περιστρέφει τα bit του καταχωρητή προς τα δεξιά κατά 1 θέση. Το bit που υπερχειλίζει επανεισάγεται από αριστερά και αντιγράφεται στο carry. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

ROR DX,1

(Περίστρεψε τα bits του καταχωρητή DX κατά μια θέση προς τα δεξιά)

ROR καταχωρητής,CL

Περιστρέφει τα bit του καταχωρητή προς τα δεξιά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Τα bit που υπερχειλίζουν επανεισάγονται από αριστερά και αντιγράφονται στο carry. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

ROL DX,CL

(Περίστρεψε τα bits του καταχωρητή DX προς τα δεξιά, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **RCL** (Rotate Through Carry Flag Left – Περιστροφή των bits αριστερά μέσω του κρατούμενου)

Σύνταξη Εντολής :

RCL καταχωρητής,1

Περιστρέφει τα bit του καταχωρητή προς τα αριστερά κατά 1 θέση. Το bit που υπερχειλίζει πηγαίνει στο Carry και το Carry (αρχική τιμή) επανεισάγεται από δεξιά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

RCL AX,1

(Περίστρεψε τα bits του καταχωρητή AX κατά μια θέση προς τα αριστερά μέσω του κρατούμενου)

RCL καταχωρητής,CL

Περιστρέφει τα bit του καταχωρητή προς τα αριστερά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Τα bit που υπερχειλίζουν πηγαίνουν στο carry και το carry επανεισάγεται από δεξιά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

RCL AX,CL

(Περίστρεψε τα bits του καταχωρητή AX προς τα αριστερά μέσω του carry, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολή **RCR** (Rotate Through Carry Flag Right – Περιστροφή των bits δεξιά μέσω του κρατούμενου)

Σύνταξη Εντολής :

RCR καταχωρητής,1

Περιστρέφει τα bit του καταχωρητή προς τα δεξιά κατά 1 θέση. Το bit που υπερχειλίζει πηγαίνει στο Carry και το Carry (αρχική τιμή) επανεισάγεται από αριστερά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :

RCR AX,1

(Περίστρεψε τα bits του καταχωρητή AX κατά μια θέση προς τα δεξιά μέσω του κρατούμενου)

RCR καταχωρητής,CL

Περιστρέφει τα bit του καταχωρητή προς τα δεξιά κατά τόσες θέσεις όσο η τιμή του καταχωρητή CL. Τα bit που υπερχειλίζουν πηγαίνουν στο carry και το carry επανεισάγεται από αριστερά. Προσοχή, λειτουργεί μόνο για τους καταχωρητές

AX, BX, CX, DX, BP, SI, DI, SP και όχι για καταχωρητές τμημάτων και σημαίες. Παράδειγμα :
RCR AX,CL
(Περίστρεψε τα bits του καταχωρητή AX προς τα δεξιά μέσω του carry, τόσες θέσεις όσο η τιμή του καταχωρητή CL)

Εντολές Χειρισμού Αλφαριθμητικών (String Manipulation Commands)
Εντολή **REP** (Repeat – Επανέλαβε)
Σύνταξη Εντολής :
REP <Εντολή Αλφαριθμητικών>

Επαναλαμβάνει την επόμενη εντολή αλφαριθμητικών τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Προσοχή, λειτουργεί μόνο όταν η επόμενη εντολή είναι μία από τις εντολές αλφαριθμητικών, δηλαδή τις MOVS, CMPS, SCAS, LODS, STOS. Παράδειγμα :
MOV AX,41
MOV CX,5
REP STOSB
(Αποθηκεύει 5 bytes 41_{16} ή "A" στην θέση ES:DI)

Εντολή **MOVS** (Move String Byte/Word – Αντιγραφή byte/word από String σε String)
Σύνταξη Εντολής :
MOVSB

Αντιγράφει ένα byte ενός string από την διεύθυνση DS:SI στην διεύθυνση ES:DI. Αμέσως μετά αυξάνει ή μειώνει τους καταχωρητές SI και DI κατά 1, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Σημαία Κατεύθυνσης	Σημασία	Μεταβολή των SI, DI
0	UP	Αύξηση
1	DOWN	Μείωση

Συνδυάζεται με την εντολή "REP" η οποία εκτελεί την MOVSB τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Παράδειγμα :
MOV SI, 200
MOV DI, 300
MOV CX, 4
REP MOVSB
(Αντιγράφει 4 bytes από το string στην θέση DS:SI στο string που είναι στη θέση ES:DI, αυξάνοντας τους SI, DI σε κάθε επανάληψη και μειώνοντας τον CX)

MOVSW

Αντιγράφει δύο byte (word) ενός string από την διεύθυνση DS:SI στην διεύθυνση ES:DI. Αμέσως μετά αυξάνει ή μειώνει τους καταχωρητές SI και DI κατά 2, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Εντολή **CMPS** (Compare String Byte/Word – Σύγκρινε byte/word από δύο string)
Σύνταξη Εντολής :
CMPSB

Συγκρίνει ένα byte ενός string από την διεύθυνση DS:SI με ένα byte ενός άλλου string στην διεύθυνση ES:DI, επηρεάζοντας τις σημαίες (flags). Αμέσως μετά αυξάνει ή μειώνει τους καταχωρητές SI και DI κατά 1, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Σημαία Κατεύθυνσης	Σημασία	Μεταβολή των SI, DI
0	UP	Αύξηση
1	DOWN	Μείωση

Συνδυάζεται με την εντολή "REP" η οποία εκτελεί την CMPSB τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Πιό συχνά όμως συνδυάζεται με την εντολή LOOP ώστε να υπάρχει περιθώριο λήψης αποφάσεων σε κάθε σύγκριση. Παράδειγμα :

MOV SI, 200
MOV DI, 300
CMPSB

(Συγκρίνει 1 byte από το string στην θέση DS:SI με 1 byte από το string που είναι στη θέση ES:DI, επηρεάζοντας τις σημαίες, και αυξάνοντας τους SI, DI)

CMPSW

Συγκρίνει δύο byte (word) ενός string από την διεύθυνση DS:SI με δύο byte ενός άλλου string στην διεύθυνση ES:DI, επηρεάζοντας τις σημαίες (flags). Αμέσως μετά αυξάνει ή μειώνει τους καταχωρητές SI και DI κατά 2, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Εντολή **SCAS** (Scan String Byte/Word – Αναζήτηση byte/word σε string)
Σύνταξη Εντολής :

SCASB

Συγκρίνει το περιεχόμενο του καταχωρητή AL με ένα byte ενός string στην διεύθυνση ES:DI, επηρεάζοντας τις σημαίες (flags). Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή DI κατά 1, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Σημαία Κατεύθυνσης	Σημασία	Μεταβολή του DI
0	UP	Αύξηση
1	DOWN	Μείωση

Συνδυάζεται με την εντολή "REP" η οποία εκτελεί την SCASB τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Πιό συχνά όμως συνδυάζεται με την εντολή LOOP ώστε να υπάρχει περιθώριο λήψης αποφάσεων σε κάθε σύγκριση. Παράδειγμα :

MOV AL, 41
MOV DI, 300
SCASB

(Συγκρίνει τον AL με 1 byte από το string στην θέση ES:DI, επηρεάζοντας τις σημαίες, και αυξάνοντας τον DI)

SCASW

Συγκρίνει το περιεχόμενο του καταχωρητή AX δύο byte (word) με δύο byte ενός string που βρίσκεται στην διεύθυνση ES:DI, επηρεάζοντας τις σημαίες (flags). Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή DI κατά 2, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Εντολή **LODS** (Load String Byte/Word – Φόρτωσε byte/word ενός string)
Σύνταξη Εντολής :
LODSB

Φορτώνει στον καταχωρητή AL (1 byte) το περιεχόμενο της θέσης μνήμης που είναι αποθηκευμένη στους καταχωρητές DS:SI. Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή SI κατά 1, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Σημαία Κατεύθυνσης	Σημασία	Μεταβολή του SI
0	UP	Αύξηση
1	DOWN	Μείωση

Παράδειγμα :
MOV SI,0500
LODSB
MOV [0200],AL
LODSB
MOV [0201],AL
(Φορτώνει 2 bytes από την θέση DS:SI (DS:0500), αυξάνοντας τον SI σε κάθε επανάληψη και τα αποθηκεύει στις θέσεις μνήμης DS:0200 και DS:0201)
Συνδυάζεται με την εντολή "REP" η οποία εκτελεί την LODSB τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Πιό συχνά όμως συνδυάζεται με την εντολή LOOP ώστε να υπάρχει περιθώριο επεξεργασίας των bytes που φορτώνονται πριν το επόμενο φόρτωμα.

LODSW

Φορτώνει στον καταχωρητή AX (2 byte) το περιεχόμενο της θέσης μνήμης που είναι αποθηκευμένη στους καταχωρητές DS:SI. Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή SI κατά 2, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Εντολή **STOS** (Store String Byte/Word – Αποθήκευσε byte/word σε string)
Σύνταξη Εντολής :
STOSB

Γράφει το περιεχόμενο του καταχωρητή AL (1 byte) στην θέση μνήμης που είναι αποθηκευμένη στους καταχωρητές ES:DI. Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή DI κατά 1, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Σημαία Κατεύθυνσης	Σημασία	Μεταβολή του DI
0	UP	Αύξηση
1	DOWN	Μείωση

Συνδυάζεται με την εντολή "REP" η οποία εκτελεί την STOSB τόσες φορές όσο η τιμή του καταχωρητή CX, ο οποίος μειώνεται σε κάθε επανάληψη, με τελική τιμή = 0. Παράδειγμα :
MOV AX,41
MOV CX,5
REP STOSB

(Αποθηκεύει 5 bytes 41_{16} ή "A" στην θέση ES:DI, αυξάνοντας τον DI σε κάθε επανάληψη ώστε τα 'A' να αποθηκεύονται σε διαδοχικές θέσεις)

STOSW

Γράφει το περιεχόμενο του καταχωρητή AX (2 byte) στην θέση μνήμης που είναι αποθηκευμένη στους καταχωρητές ES:DI. Αμέσως μετά αυξάνει ή μειώνει τον καταχωρητή DI κατά 2, ανάλογα με την τιμή της σημαίας κατεύθυνσης (Direction Flag – DF).

Εντολές Ελέγχου Ροής Προγράμματος (Program Flow Control Commands)

Εντολή **CALL** (Call – Κλήση υπορουτίνας)

Σύνταξη Εντολής :

CALL <διεύθυνση μνήμης>

Μεταφέρει την εκτέλεση του προγράμματος στην υπορουτίνα που βρίσκεται στην <διεύθυνση μνήμης> που δόθηκε, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή CALL που εκτελείται (Near Call - ο CS παραμένει ο ίδιος). Παράδειγμα :

CALL 0300 (Near Call Direct Relative)

(πήγαινε στην υπορουτίνα που βρίσκεται στη διεύθυνση μνήμης 0300 ή CS:0300)

CALL καταχωρητής

Μεταφέρει την εκτέλεση του προγράμματος στην υπορουτίνα που βρίσκεται στην διεύθυνση μνήμης που υπάρχει σαν περιεχόμενο του <καταχωρητή> που δόθηκε, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή CALL που εκτελείται (ο CS παραμένει ο ίδιος). Παράδειγμα :

MOV BX,0300

CALL BX (Near Call Register Indirect Absolute)

(πήγαινε στην υπορουτίνα που βρίσκεται στη διεύθυνση μνήμης 0300 ή CS:0300)

CALL [διεύθυνση μνήμης] ή CALL NE [διεύθυνση μνήμης] (NE=NEAR)

Διαβάζει από την [διεύθυνση μνήμης] 2 byte (low-high) τα οποία σχηματίζουν μία διεύθυνση και μεταφέρει την εκτέλεση του προγράμματος στην υπορουτίνα που βρίσκεται σε αυτή τη διεύθυνση, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή CALL που εκτελείται (Near Call - ο CS παραμένει ο ίδιος). Παράδειγμα :

MOV WO[0300], 0500

CALL [0300] (Near Call Memory Indirect Absolute)

(διάβασε από την διεύθυνση 0300 2 byte (low-high, low=00, high=05, address=0500) και πήγαινε στην υπορουτίνα που βρίσκεται σε αυτή τη διεύθυνση μνήμης 0500 ή CS:0500)

CALL <segment:offset>

Μεταφέρει την εκτέλεση του προγράμματος στην υπορουτίνα που βρίσκεται στην διεύθυνση μνήμης <segment:offset> που δόθηκε, η οποία μπορεί να είναι οπουδήποτε στη μνήμη (Far Call - ο CS αλλάζει). Παράδειγμα :

CALL F000:0100 (Far Call Direct Absolute)

(πήγαινε στην υπορουτίνα που βρίσκεται στη διεύθυνση μνήμης F000:0100)

CALL FAR [διεύθυνση μνήμης]

Διαβάζει από την [διεύθυνση μνήμης] 4 byte (offset low, offset high, segment low, segment high) τα οποία σχηματίζουν μία διεύθυνση και μεταφέρει την εκτέλεση του προγράμματος στην υπορουτίνα που βρίσκεται σε αυτή τη διεύθυνση, η οποία μπορεί να βρίσκεται οπουδήποτε στη μνήμη (Far Call - ο CS μεταβάλλεται). Παράδειγμα :

MOV WO[0300], 0570
MOV WO[0302],C400
CALL FAR [0300] (Far Call Memory Indirect Absolute)

(διάβασε από την διεύθυνση 0300 4 byte (offset low=70, offset high=05, segment low=00, segment high=C4, address=C400:0570) και πήγαινε στην υπορουτίνα που βρίσκεται σε αυτή τη διεύθυνση μνήμης C400:0570)

Εντολή **RET** (Return from Call – Επιστροφή από υπορουτίνα)
Σύνταξη Εντολής :
RET

Επιστρέφει στην επόμενη εντολή μετά την CALL η οποία κάλεσε την υπορουτίνα. Τοποθετείται ως τελευταία εντολή των υπορουτινών, ώστε μετά την εκτέλεσή τους να επιστρέφει η εκτέλεση στο κυρίως πρόγραμμα.

Εντολή **JMP** (Jump – Μεταπήδηση προγράμματος σε άλλη διεύθυνση)
Σύνταξη Εντολής :
JMP <διεύθυνση μνήμης>

Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στην <διεύθυνση μνήμης> που δόθηκε, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή JMP που εκτελείται (Near Jump - ο CS παραμένει ο ίδιος). Παράδειγμα :

JMP 0300 (Near Jump Direct Relative)

(πήγαινε στην εντολή που βρίσκεται στη διεύθυνση μνήμης 0300 ή CS:0300)

JMP καταχωρητής

Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στην διεύθυνση μνήμης που υπάρχει σαν περιεχόμενο του <καταχωρητή> που δόθηκε, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή JMP που εκτελείται (ο CS παραμένει ο ίδιος). Παράδειγμα :

MOV BX,0300
JMP BX (Near Jump Register Indirect Absolute)

(πήγαινε στην εντολή που βρίσκεται στη διεύθυνση μνήμης 0300 ή CS:0300)

JMP [διεύθυνση μνήμης] ή JMP NE [διεύθυνση μνήμης] (NE=NEAR)

Διαβάζει από την [διεύθυνση μνήμης] 2 byte (low-high) τα οποία σχηματίζουν μία διεύθυνση και μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται σε αυτή τη διεύθυνση, η οποία ανήκει στο ίδιο Τμήμα Κώδικα (code segment) με την εντολή JMP που εκτελείται (Near Jump - ο CS παραμένει ο ίδιος). Παράδειγμα :

MOV WO[0300], 0500
JMP [0300] (Near Jump Memory Indirect Absolute)

(διάβασε από την διεύθυνση 0300 2 byte (low-high, low=00, high=05, address=0500) και πήγαινε στην εντολή που βρίσκεται σε αυτή τη διεύθυνση μνήμης 0500 ή CS:0500)

JMP <segment:offset>

Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στην διεύθυνση μνήμης <segment:offset> που δόθηκε, η οποία μπορεί να είναι οπουδήποτε στη μνήμη (Far Jump - ο CS αλλάζει). Παράδειγμα :

JMP F000:0100 (Far Jump Direct Absolute)

(πήγαινε στην εντολή που βρίσκεται στη διεύθυνση μνήμης F000:0100)

JMP FAR [διεύθυνση μνήμης]

Διαβάζει από την [διεύθυνση μνήμης] 4 byte (offset low, offset high, segment low, segment high) τα οποία σχηματίζουν μία διεύθυνση και μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται σε αυτή τη διεύθυνση, η οποία μπορεί να βρίσκεται οπουδήποτε στη μνήμη (Far Jump - ο CS μεταβάλλεται). Παράδειγμα :

MOV WO[0300], 0570
MOV WO[0302], C400
JMP FAR [0300] (Far Jump Memory Indirect Absolute)

(διάβασε από την διεύθυνση 0300 4 byte (offset low=70, offset high=05, segment low=00, segment high=C4, address=C400:0570) και πήγαινε στην εντολή που βρίσκεται σε αυτή τη διεύθυνση μνήμης C400:0570)

Εντολές **Jxx** (Jump conditional – Διακλάδωση υπό συνθήκη)

Σύνταξη Εντολής :

Jxx <διεύθυνση>

Ανάλογα με το είδος της (xx) ελέγχει την κατάσταση κάποιας σημαίας (flag) και υπό συνθήκη μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται σε αυτή τη διεύθυνση. Προσοχή, η διεύθυνση θα πρέπει να είναι κοντινή και στο διάστημα –128..+127 bytes από την αρχή της επόμενης εντολής.

Είδη εντολών :

JE ή JZ (Jump on Equal/Zero)

Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης είχε αποτέλεσμα 0 (ZF=1)

JNE ή JNZ (Jump on Not Equal/Not Zero)

Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης είχε αποτέλεσμα διάφορο του 0 (ZF=0)

JL ή JNGE (Jump on Less/Not Greater or Equal)

Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μικρότερος του δεύτερου (SF <> OF)

JLE ή JNG (Jump on Less or Equal/Not Greater)

Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μικρότερος ή ίσος του δεύτερου (SF <> OF ή ZF=1)

JNL ή JGE (Jump on Not Less/Greater or Equal)

Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μεγαλύτερος ή ίσος του δεύτερου (SF = OF)

JNLE ή JG (Jump on Not Less or Equal/Greater)
Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μεγαλύτερος του δεύτερου (ZF=0 και SF = OF)

JB ή JNAE (Jump on Below/Not Above or Equal)
Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ μη προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μικρότερος του δεύτερου (CF = 1)

JBE ή JNA (Jump on Below or Equal/Not Above)
Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ μη προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μικρότερος ή ίσος του δεύτερου (CF = 1 ή ZF = 1)

JNB ή JAE (Jump on Not Below /Above or Equal)
Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ μη προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μεγαλύτερος ή ίσος του δεύτερου (CF = 0)

JNBE ή JA (Jump on Not Below or Equal/Above)
Διακλάδωση προγράμματος αν η προηγούμενη σύγκριση μεταξύ μη προσημασμένων αριθμών είχε αποτέλεσμα ο πρώτος να είναι μεγαλύτερος του δεύτερου (CF = 0 και ZF=0)

JP ή JPE (Jump on Parity/Parity Even)
Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης έδωσε αριθμό με ζυγό αριθμό μονάδων (PF=1).

JNP ή JPO (Jump on Not Parity/Parity Odd)
Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης έδωσε αριθμό με μονό αριθμό μονάδων (PF=0).

JO (Jump on Overflow)
Διακλάδωση προγράμματος αν κατά την προηγούμενη πράξη δημιουργήθηκε υπερχείλιση των προσημασμένων αριθμών (OF=1).

JNO (Jump on Not Overflow)
Διακλάδωση προγράμματος αν κατά την προηγούμενη πράξη δεν δημιουργήθηκε υπερχείλιση των προσημασμένων αριθμών (OF=0).

JS (Jump on Sign)
Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης ήταν αρνητικός αριθμός (SF=1).

JNS (Jump on Not Sign)
Διακλάδωση προγράμματος αν το αποτέλεσμα της προηγούμενης πράξης ήταν θετικός αριθμός (SF=0).

JCXZ (Jump on CX Zero)
Διακλάδωση προγράμματος αν η τιμή του καταχωρητή CX είναι 0 (CX=0).

Εντολή **LOOP** (Loop – Βρόχος με απαριθμητή τον CX)
Σύνταξη Εντολής :

LOOP <διεύθυνση>

Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στη διεύθυνση που δόθηκε, τόσες φορές όσες η τιμή του καταχωρητή CX. Σε κάθε επανάληψη μειώνεται ο CX κατά 1 και ελέγχεται αν έφτασε στο 0. Αν δεν έχει φτάσει τότε πηγαίνει πάλι στη διεύθυνση, που υποτίθεται ότι πρέπει να δείχνει την αρχή του βρόχου. Αν ο CX βρεθεί ίσος με 0, τότε ο βρόχος σταματά και το πρόγραμμα συνεχίζεται στην επόμενη εντολή μετά την LOOP. Προσοχή, η διεύθυνση θα πρέπει να είναι κοντινή και στο διάστημα –128..+127 bytes από την αρχή της επόμενης εντολής. Παράδειγμα :

0100:0000 MOV AX,0
0100:0003 MOV CX, BX
0100:0005 ADD AX, DX
0100:0007 LOOP 0005

(Επαναλαμβάνει την εντολή ADD AX,DX τόσες φορές όσο η τιμή του CX (BX), δηλαδή CX φορές πρόσθεση του DX : DX+DX+DX…+DX, δηλαδή πολλαπλασιάζει τους DX και BX)

Εντολή **LOOPZ ή LOOPE** (Loop – Βρόχος με απαριθμητή τον CX και συνθήκη ισότητας)

Σύνταξη Εντολής :

LOOPZ <διεύθυνση> ή LOOPE <διεύθυνση>

Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στη διεύθυνση που δόθηκε, τόσες φορές όσες η τιμή του καταχωρητή CX και εφόσον η σημαία μηδενός ZF έχει ανάψει (ZF=1). Σε κάθε επανάληψη μειώνεται ο CX κατά 1 και ελέγχεται αν έφτασε στο 0. Αν δεν έχει φτάσει τότε πηγαίνει πάλι στη διεύθυνση, που υποτίθεται ότι πρέπει να δείχνει την αρχή του βρόχου. Αν ο CX βρεθεί ίσος με 0, ή η σημαία μηδενός δεν είναι αναμμένη (ZF=0) τότε ο βρόχος σταματά και το πρόγραμμα συνεχίζεται στην επόμενη εντολή μετά την LOOP. Προσοχή, η διεύθυνση θα πρέπει να είναι κοντινή και στο διάστημα –128..+127 bytes από την αρχή της επόμενης εντολής. Παράδειγμα :

0100:0000 MOV CX,5
0100:0003 MOV BX,CX
0100:0005 MOV AX, [BX+0200]
0100:0009 CMP AX, [BX+0300]
0100:000D LOOPZ 0005

(Συγκρίνει 2 string 5 θέσεων που βρίσκονται στις θέσεις 0200 και 0300 από το τέλος προς την αρχή, και σταματάει είτε όταν συγκρίνει πλήρως τα string είτε όταν βρει την πρώτη διαφορά. Αν συγκρίνει πλήρως τα string τότε η LOOPZ τερματίζει λόγω μηδενισμού του CX. Αν βρεθεί διαφορά με την CMP τότε η LOOP τερματίζει επειδή η σημαία μηδενός γίνεται 0, ZF=0)

Εντολή **LOOPNZ ή LOOPNE** (Loop – Βρόχος με απαριθμητή τον CX και συνθήκη ανισότητας)

Σύνταξη Εντολής :

LOOPNZ <διεύθυνση> ή LOOPNE <διεύθυνση>

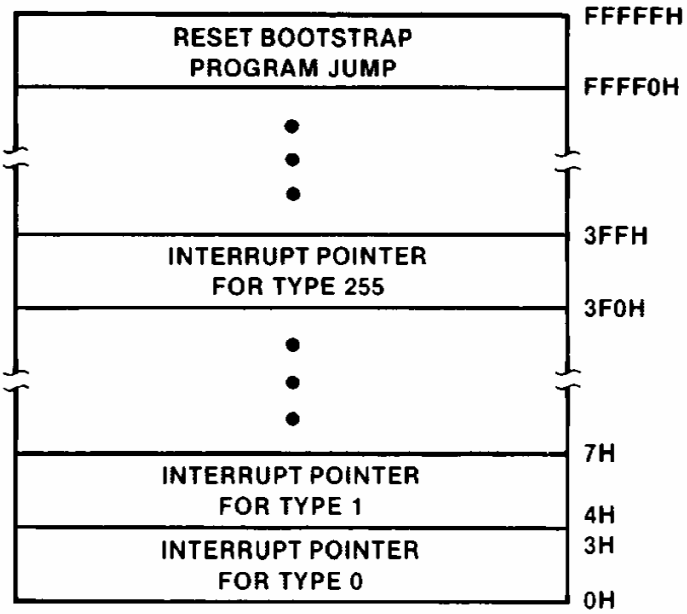
Μεταφέρει την εκτέλεση του προγράμματος στην εντολή που βρίσκεται στη διεύθυνση που δόθηκε, τόσες φορές όσες η τιμή του καταχωρητή CX και εφόσον η σημαία μηδενός ZF είναι σβηστή (ZF=0). Σε κάθε επανάληψη μειώνεται ο CX κατά 1 και ελέγχεται αν έφτασε στο 0. Αν δεν έχει φτάσει τότε πηγαίνει πάλι στη διεύθυνση, που υποτίθεται ότι πρέπει να δείχνει την αρχή του βρόχου. Αν ο CX βρεθεί ίσος με 0, ή η σημαία μηδενός είναι αναμμένη (ZF=1) τότε ο βρόχος σταματά και το πρόγραμμα συνεχίζεται στην επόμενη εντολή μετά την LOOP. Προσοχή, η διεύθυνση θα πρέπει να είναι κοντινή και στο διάστημα –128..+127 bytes από την αρχή της επόμενης εντολής. Παράδειγμα :

```
0100:0000 MOV BX,FFFF
0100:0003 MOV CX,20
0100:0006 INC BX
0100:0007 MOV AL, [BX+0200]
0100:000B CMP AL, 00
0100:000D LOOPNZ 0005
```

(Μετράει στον BX πόσους χαρακτήρες έχει πραγματικά ένα string το οποίο έχει μέγιστο τους 20 χαρακτήρες και χαρακτήρα τερματισμού το /0 (00_{16}). Ο βρόχος με την LOOPNZ σταματάει είτε όταν διαβάσει και τους 20 χαρακτήρες, δηλαδή όταν μηδενιστεί ο CX, είτε όταν η σύγκριση ενός χαρακτήρα με το 00 δώσει αποτέλεσμα true δηλαδή ZF=1)

Εντολή **INT** (Interrupt – Κλήση ρουτίνας διακοπής)
Σύνταξη Εντολής :
INT <αριθμός διακοπής>

Παράγει μία κλήση προς την ρουτίνα χειρισμού της διακοπής της οποίας ο αριθμός δίνεται ως παράμετρος. Ο αριθμός της διακοπής μπορεί να είναι στην περιοχή 00..FF (0..255 στο δεκαδικό). Κάθε αριθμός διακοπής αντιστοιχεί σε μία διεύθυνση υπορουτίνας η οποία θα εκτελεστεί. Οι διευθύνσεις των ρουτινών χειρισμού των διακοπών είναι των 4ων bytes και είναι αποθηκευμένες στις πρώτες 1024 διευθύνσεις μνήμης (00000_{16} … $003FF_{16}$). Για παράδειγμα η διεύθυνση της ρουτίνας εξυπηρέτησης της διακοπής 85 (INT 85), που όταν εκτελεστεί επανεκκινεί το πρόγραμμα MONITOR, βρίσκεται στην θέση 00214_{16} = 85_{16} x 4. Εκεί υπάρχουν διαδοχικά τα bytes 9F 01 00 FC. Τα πρώτα 2 είναι low και high bytes του offset, και τα 2 επόμενα τα low και high bytes του segment. Έτσι η πραγματική διεύθυνση της ρουτίνας εξυπηρέτησης της διακοπής 85 είναι η FC00:019F. Η διεύθυνση αυτή είναι στην ROM του μηχανήματος. Η περιοχή των interrupt vectors φαίνεται στο σχήμα 1.



Σχήμα 1. Η περιοχή των interrupt vectors του 8088 και το Jump εκκίνησης.

Οι πρώτες 32 διακοπές (00..1F, ή $0..31_{10}$) είναι δεσμευμένες από την Intel και πολλές από αυτές αντιστοιχούν σε εσωτερικά σφάλματα του επεξεργαστή (π.χ. divide by zero).

INTO

Αντιστοιχεί στην διακοπή με αριθμό 4 (Overflow Interrupt). Όταν κληθεί ελέγχει την τιμή της σημαίας υπερχείλισης (OF) και αν αυτή είναι 1 τότε εκτελεί την ρουτίνα διαχείρισης της υπερχείλισης.

INT 3

Η διακοπή αυτή χρησιμοποιείται έξυπνα από το πρόγραμμα MONITOR, κατά την εκτέλεση ενός προγράμματος, με καθορισμό διευθύνσεων παύσης (breakpoints). Μπορεί να χρησιμοποιηθεί ως η τελευταία εντολή ενός προγράμματος, καθώς μόλις εκτελείται, εμφανίζει τις τιμές των καταχωρητών και επιστρέφει στο MONITOR (trap to debugger).

Εντολή **IRET** (Return from Interrupt – Επιστροφή από ρουτίνα διακοπής)
Σύνταξη Εντολής :

IRET

Τοποθετείται ως τελευταία εντολή μιας ρουτίνας εξυπηρέτησης διακοπής (interrupt handler routine), και επιστρέφει την εκτέλεση του προγράμματος στην επόμενη εντολή από αυτή που κάλεσε την διακοπή (INT).

Εντολές Ελέγχου του Επεξεργαστή (Processor Control Commands)

Εντολή **CLC** (Clear Carry – Μηδενίζει το κρατούμενο)
Σύνταξη Εντολής :

CLC

Καθιστά την σημαία του κρατούμενου (Carry Flag – CF) ίση με 0.

Εντολή **STC** (Set Carry – Θέτει το κρατούμενο)
Σύνταξη Εντολής :
STC

Καθιστά την σημαία του κρατούμενου (Carry Flag – CF) ίση με 1.

Εντολή **CMC** (Complement Carry – Συμπληρωματικό κρατούμενο)
Σύνταξη Εντολής :
CMC

Καθιστά την σημαία του κρατούμενου (Carry Flag – CF) ίση με το συμπλήρωμα της προηγούμενης τιμής. Έτσι αν το carry είναι 0 γίνεται 1, και αν είναι 1 γίνεται 0.

Εντολή **CLD** (Clear Direction – Μηδενίζει τη σημαία κατεύθυνσης)
Σύνταξη Εντολής :
CLD

Καθιστά την σημαία κατεύθυνσης (Direction Flag – DF) ίση με 0. Η σημαία κατεύθυνσης καθορίζει αν οι εντολές των strings θα εκτελούνται από μικρές διευθύνσεις προς μεγάλες ή ανάποδα. Όταν είναι 0, τότε οι εντολές των string εκτελούνται από μικρές διευθύνσεις προς μεγάλες.

Εντολή **STD** (Set Direction – Θέτει τη σημαία κατεύθυνσης)
Σύνταξη Εντολής :
STD

Καθιστά την σημαία κατεύθυνσης (Direction Flag – DF) ίση με 1. Η σημαία κατεύθυνσης καθορίζει αν οι εντολές των strings θα εκτελούνται από μικρές διευθύνσεις προς μεγάλες ή ανάποδα. Όταν είναι 1, τότε οι εντολές των string εκτελούνται από μεγάλες διευθύνσεις προς μικρές.

Εντολή **CLI** (Clear Interrupt – Μηδενίζει τη σημαία διακοπών)
Σύνταξη Εντολής :
CLI

Καθιστά την σημαία διακοπών (Interrupt Flag – IF) ίση με 0. Η σημαία διακοπών καθορίζει αν θα επιτρέπονται οι διακοπές (hardware interrupts) κατά την εκτέλεση του προγράμματος ή όχι. Όταν είναι 0, τότε απαγορεύεται η διακοπή του προγράμματος από interrupts.

Εντολή **STI** (Set Interrupt – Θέτει τη σημαία διακοπών)
Σύνταξη Εντολής :
STI

Καθιστά την σημαία διακοπών (Interrupt Flag – IF) ίση με 1. Η σημαία διακοπών καθορίζει αν θα επιτρέπονται οι διακοπές (hardware interrupts) κατά την εκτέλεση του προγράμματος ή όχι. Όταν είναι 1, τότε επιτρέπεται η διακοπή του προγράμματος από interrupts.

Εντολή **HALT** (Halt – Σταμάτημα του επεξεργαστή)

Σύνταξη Εντολής :
HALT

Σταματά την λειτουργία του επεξεργαστή και τον φέρνει σε κατάσταση στάσης (halt mode). Από την κατάσταση αυτή μπορεί να βγεί αν :

1. Γίνει Reset στον επεξεργαστή.
2. Παραχθεί επιτρεπόμενη διακοπή
3. Παραχθεί διακοπή NMI (Non-Maskable Interrupt)

Εντολή **WAIT** (Wait for pending floating point exceptions – Αναμονή για την διαχείριση σφαλμάτων κινητής υποδιαστολής)
Σύνταξη Εντολής :
WAIT

Σταματά προσωρινά την λειτουργία του επεξεργαστή και τον κάνει να ελέγξει για τυχόν σφάλματα που προέκυψαν με εντολές κινητής υποδιαστολής. Βάζοντας μία τέτοια εντολή αμέσως μετά από οποιαδήποτε εντολή κινητής υποδιαστολής εξασφαλίζει ότι τυχόν σφάλματα που θα προκύψουν κατά την εκτέλεση της εντολής floating point, θα τύχουν ολοκληρωμένης διαχείρισης πριν προχωρήσει ο επεξεργαστής σε επόμενες εντολές που θα μπορούσαν να αλλοιώσουν το αποτέλεσμα της εντολής floating point.

Εντολή **ESC** (Escape – Χορήγηση διαύλων σε άλλους επεξεργαστές)
Σύνταξη Εντολής :
ESC

Δίνει το δικαίωμα σε άλλους επεξεργαστές να διαχειριστούν τους διαύλους δεδομένων και διευθύνσεων, ενώ για τον τρέχοντα επεξεργαστή λειτουργεί σαν NP.

Εντολή **LOCK** (Lock signal assertion – Κλείδωμα των διαύλων για αποκλειστική χρήση)
Σύνταξη Εντολής :
LOCK

Προκαλεί την έναυση του σήματος LOCK του επεξεργαστή το οποίο όταν είναι σε ενέργεια κάνει ολόκληρη την επόμενη εντολή να εκτελεστεί ως ατομική ενέργεια. Αυτό σημαίνει ότι η εντολή θα έχει αποκλειστική χρήση της μνήμης που θα προσπελάσει. Σε περιβάλλοντα πολλών παράλληλων επεξεργαστών αυτό εξασφαλίζει την ακεραιότητα της εκτέλεσης των εντολών, σε περίπτωση που αυτές εκτελούνται σε ένα περιβάλλον με μνήμη κοινής χρήσης μεταξύ των επεξεργαστών. Σε περιβάλλον ενός επεξεργαστή η εντολή LOCK προκαλεί το κλείδωμα του Διαύλου Δεδομένων και Διευθύνσεων έτσι ώστε να αποκλειστούν τυχόν μεταφορές δεδομένων μέσω του DMA και βεβαίως απαγορεύονται οι διακοπές μέσω interrupts. Η εντολή LOCK μπορεί να ακολουθηθεί μόνο από τις εντολές: ADD, ADC, AND, DEC, INC, NEG, NOT, OR, SBB, SUB XOR, and XCHG, και μόνο όταν αυτές χρησιμοποιούν τη μνήμη. Παράδειγμα:
LOCK ADD AX,[200]
(Κάνει την εντολή ADD να έχει αποκλειστική χρήση της μνήμης που χρησιμοποιεί, δηλαδή των διευθύνσεων 0100:0200, 0100:0201)

Εντολή **NOP** (No Operation – Μη Λειτουργία)
Σύνταξη Εντολής :
NOP

Εντολή που δεν κάνει καμία ενέργεια, ενώ απλά καταναλώνει κύκλους μηχανής από την CPU. Χρησιμοποιείται είτε σε βρόχους καθυστέρησης είτε για να γεμίζει τμήματα προγραμμάτων τα οποία προβλέπεται ότι θα συμπληρωθούν αργότερα.

Οι κωδικοί των εντολών

Στις επόμενες σελίδες παραθέτονται οι εντολές του 8088 σε συνοπτική μορφή μαζί με τους κωδικούς των εντολών (opcodes).

8086/8088 Instruction Set Summary

Mnemonic and Description	Instruction Code			
DATA TRANSFER				
MOV = Move:	76543210	76543210	76543210	76543210
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register	1011 w reg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod 1 1 0 r/m		
Register	01010 reg			
Segment Register	000 reg 110			
POP = Pop:				
Register/Memory	10001111	mod 0 0 0 r/m		
Register	01011 reg			
Segment Register	000 reg 111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010 reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
ARITHMETIC	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
ADD = Add:				
Reg./Memory with Register to Either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w = 01
Immediate to Accumulator	0 0 0 0 0 1 0 w	data	data if w = 1	
ADC = Add with Carry:				
Reg./Memory with Register to Either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to Register/Memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w = 01
Immediate to Accumulator	0 0 0 1 0 1 0 w	data	data if w = 1	
INC = Increment:				
Register/Memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA = ASCII Adjust for Add	0 0 1 1 0 1 1 1			
BAA = Decimal Adjust for Add	0 0 1 0 0 1 1 1			
SUB = Subtract:				
Reg./Memory and Register to Either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from Register/Memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w = 01
Immediate from Accumulator	0 0 1 0 1 1 0 w	data	data if w = 1	
SSB = Subtract with Borrow				
Reg./Memory and Register to Either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from Register/Memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w = 01
Immediate from Accumulator	0 0 0 1 1 1 w	data	data if w = 1	
DEC = Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG = Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP = Compare:				
Register/Memory and Register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with Register/Memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w = 01
Immediate with Accumulator	0 0 1 1 1 1 0 w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	0 0 1 1 1 1 1 1			
DAS = Decimal Adjust for Subtract	0 0 1 0 1 1 1 1			
MUL = Multiply (Unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL = Integer Multiply (Signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM = ASCII Adjust for Multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV = Divide (Unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV = Integer Divide (Signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD = ASCII Adjust for Divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW = Convert Byte to Word	1 0 0 1 1 0 0 0			
CWD = Convert Word to Double Word	1 0 0 1 1 0 0 1			

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code			
LOGIC	76543210	76543210	76543210	76543210
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift Logical/Arithmetic Left	110100vw	mod 100 r/m		
SHR = Shift Logical Right	110100vw	mod 101 r/m		
SAR = Shift Arithmetic Right	110100vw	mod 111 r/m		
ROL = Rotate Left	110100vw	mod 000 r/m		
ROR = Rotate Right	110100vw	mod 001 r/m		
RCL = Rotate Through Carry Flag Left	110100vw	mod 010 r/m		
RCR = Rotate Through Carry Right	110100vw	mod 011 r/m		
AND = **And:**				
Reg./Memory and Register to Either	001000dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 100 r/m	data	data if w = 1
Immediate to Accumulator	0010010w	data	data if w = 1	
TEST = **And Function to Flags. No Result:**				
Register/Memory and Register	1000010w	mod reg r/m		
Immediate Data and Register/Memory	1111011w	mod 000 r/m	data	data if w = 1
Immediate Data and Accumulator	1010100w	data	data if w = 1	
OR = **Or:**				
Reg./Memory and Register to Either	000010dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 001 r/m	data	data if w = 1
Immediate to Accumulator	0000110w	data	data if w = 1	
XOR = **Exclusive or:**				
Reg./Memory and Register to Either	001100dw	mod reg r/m		
Immediate to Register/Memory	1000000w	mod 110 r/m	data	data if w = 1
Immediate to Accumulator	0011010w	data	data if w = 1	
STRING MANIPULATION				
REP = Repeat	1111001z			
MOVS = Move Byte/Word	1010010w			
CMPS = Compare Byte/Word	1010011w			
SCAS = Scan Byte/Word	1010111w			
LODS = Load Byte/Wd to AL/AX	1010110w			
STOS = Stor Byte/Wd from AL/A	1010101w			
CONTROL TRANSFER				
CALL = **Call:**				
Direct Within Segment	11101000	disp-low	disp-high	
Indirect Within Segment	11111111	mod 010 r/m		
Direct Intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect Intersegment	11111111	mod 011 r/m		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code		
JMP = Unconditional Jump:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct Within Segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct Within Segment-Short	1 1 1 0 1 0 1 1	disp	
Indirect Within Segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct Intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET = Return from CALL:			
Within Segment	1 1 0 0 0 0 1 1		
Within Seg Adding Immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ = Jump on Equal/Zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE = Jump on Below/Not Above or Equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	0 1 1 1 0 1 1 0	disp	
JP/JPE = Jump on Parity/Parity Even	0 1 1 1 1 0 1 0	disp	
JO = Jump on Overflow	0 1 1 1 0 0 0 0	disp	
JS = Jump on Sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	0 1 1 1 1 1 1 1	disp	
JNB/JAE = Jump on Not Below/Above or Equal	0 1 1 1 0 0 1 1	disp	
JNBE/JA = Jump on Not Below or Equal/Above	0 1 1 1 0 1 1 1	disp	
JNP/JPO = Jump on Not Par/Par Odd	0 1 1 1 1 0 1 1	disp	
JNO = Jump on Not Overflow	0 1 1 1 0 0 0 1	disp	
JNS = Jump on Not Sign	0 1 1 1 1 0 0 1	disp	
LOOP = Loop CX Times	1 1 1 0 0 0 1 0	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	1 1 1 0 0 0 0 1	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	1 1 1 0 0 0 0 0	disp	
JCXZ = Jump on CX Zero	1 1 1 0 0 0 1 1	disp	
INT = Interrupt			
Type Specified	1 1 0 0 1 1 0 1	type	
Type 3	1 1 0 0 1 1 0 0		
INTO = Interrupt on Overflow	1 1 0 0 1 1 1 0		
IRET = Interrupt Return	1 1 0 0 1 1 1 1		

8086/8088 Instruction Set Summary (Continued)

Mnemonic and Description	Instruction Code	
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive:
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high; disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = then EA = disp-high: disp-low.
if s:w = 01 then 16 bits of immediate data form the operand
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register
x = don't care
z is used for string primitives for comparison with ZF FLAG
SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS =
X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

DATA SHEET REVISION REVIEW

The following list represents key differences between this and the -005 data sheet. Please review this summary carefully.

1. The Intel 8088 implementation technology (HMOS) has been changed to (HMOS-II).

Τρόποι Διευθυνσιοδότησης Μνήμης - Addressing Modes

Όπως γίνεται αντιληπτό οι εντολές του 8088 συντάσσονται με αρκετούς διαφορετικούς τρόπους. Σε αυτούς τους τρόπους σύνταξης υπεισέρχονται παράμετροι που αφορούν καταχωρητές, δεδομένα, θέσεις μνήμης, σχετικές μετατοπίσεις κ.λ.π. Βέβαια ΔΕΝ συντάσσονται ΟΛΕΣ οι εντολές με ΟΛΟΥΣ τους τρόπους σύνταξης. Οι τρόποι σύνταξης των εντολών είναι οι ακόλουθοι :

Τρόποι Σύνταξης των εντολών του 8088

1. **Υπονοούμενος (Implied)** : Η σύνταξη της εντολής δεν περιέχει παράμετρο ή συντάσσεται με συγκεκριμένο τρόπο ο οποίος υποννοεί το ποιές είναι οι παράμετροι.
   Παράδειγμα : STC (Set Carry)
   LODSB ([DS:SI]→ AL)

2. **Παράμετρος Καταχωρητής (Register Operand)** : Η εντολή επενεργεί σε ένα καταχωρητή.
   Παράδειγμα : PUSH AX (AX→Stack)
   DEC BX (BX=BX-1)

3. **Παράμετρος Θέση Μνήμης (Memory Operand)** : Η εντολή επενεργεί σε δεδομένα που βρίσκονται στην μνήμη και μάλιστα στο τμήμα δεδομένων (data segment).
   Παράδειγμα : POP [0200] (Stack→ DS:0200)

4. **Παράμετρος Σχετική Μετατόπιση (Relative Offset Operand)** : Η εντολή σχετίζεται με αλλαγή στη ροή του προγράμματος (εντολές Jxx, κ.λ.π.) και μεταφέρει την εκτέλεση σε διεύθυνση του τμήματος κώδικα (code segment) που απέχει συγκεκριμένο αριθμό bytes, θετικό ή αρνητικό, από την αρχή της επόμενης εντολής.
   Παράδειγμα : JNE 0020 (IF (ZF=0) IP=0020)
   Προσοχή : Η εντολή σε επίπεδο Assembly συντάσσεται με απόλυτη διεύθυνση κώδικα (0020), και αυτό γίνεται για ευκολία στην ανάπτυξη του προγράμματος, αλλά η αντίστοιχη εντολή σε κώδικα μηχανής περιέχει την απόσταση ανάμεσα στην διεύθυνση προορισμού και στην διεύθυνση της επόμενης εντολής. Έτσι άν η επόμενη εντολή είχε διεύθυνση 0010 τότε η εντολή σε γλώσσα μηχανής θα ήταν 75 10 (75 = opcode της JNE, και 10 = 0020 – 0010)

5. **Παράμετρος Αριθμός (Numerical Operand)** : Η εντολή δέχεται ως παράμετρο ένα σταθερό νούμερο που έχει θέση δεδομένου.
   Παράδειγμα : INT 3 (Display Registers & Return to MONITOR)

6. **Έμμεση Προσπέλαση (Indirect Addressing)** : Η εντολή δέχεται ως παράμετρο μία θέση μνήμης από την οποία διαβάζει κάποια bytes (2 ή 4) τα οποία σχηματίζουν την τελική διεύθυνση στην οποία θα επενεργήσει η εντολή.
   Παράδειγμα : MOV WO[0500], 0570
   MOV WO[0502], E400
   CALL FAR [0500] (CS=E400, IP=0570)

Συνδυασμοί των παραπάνω

1) **Καταχωρητής σε Καταχωρητή (Register to Register)** : Η εντολή διαβάζει δεδομένα από ένα καταχωρητή και μετά από επεξεργασία τα αποθηκεύει σε άλλο καταχωρητή.
   Παράδειγμα : MOV CX,DX (CX=DX)
   XCHG AX,BX (AX=BX, BX=AX)

2) **Καταχωρητής και Μνήμη (Register to/from Memory)** : Η εντολή διαβάζει δεδομένα από ένα καταχωρητή ή τη μνήμη και μετά από επεξεργασία τα αποθηκεύει στη μνήμη ή σε καταχωρητή αντίστοιχα.
   Παράδειγμα : SUB AX,[0100] (AX=AX-[DS:0100])
   CMP [0200],CX ([0200]~CX ?)

3) **Καταχωρητής και Αριθμός (Register and Numerical Value)** : Η εντολή παίρνει ένα σταθερό νούμερο και το περιεχόμενο ενός καταχωρητή και μετά από κάποια επεξεργασία το αποθηκεύει σε ένα καταχωρητή.
   Παράδειγμα : TEST DX, FFFF (DX~FFFF ?)
   IN AL,5C (AL=port(5C))

4) **Μνήμη και Αριθμός (Memory and Numerical Value)** : Η εντολή επενεργεί πάνω σε ένα σταθερό νούμερο και μία διεύθυνση μνήμης.
   Παράδειγμα : ADC BY[0300], 33 ([DS:0300] += 33 + CF)
   OR WO[0400],FFFF ([DS:0400] = [DS:0400] OR FFFF)

Από τους παραπάνω τρόπους σύνταξης των εντολών, αρκετοί συντάσσονται με διεύθυνση μνήμης (π.χ. 3. Παράμετρος Θέση Μνήμης - Memory Operand). Σε αυτές τις εντολές η διεύθυνση μνήμης μπορεί να είναι στην απλούστερη περίπτωση μία σταθερή διεύθυνση (π.χ. POP [0200]). Όμως οι εντολές του 8088 μπορούν να συντάξουν διευθύνσεις με την βοήθεια και συγκεκριμένων καταχωρητών που παίζουν τον ρόλο των δεικτών. Έτσι όταν συντάσσουμε μία διεύθυνση μνήμης με τη βοήθεια και ενός καταχωρητή, τότε το περιεχόμενο του καταχωρητή προστίθεται στην διεύθυνση μνήμης που δώσαμε, ώστε να προκύψει η τελική διεύθυνση. Οι καταχωρητές που μπορούν να συμμετέχουν στον προσδιορισμό διεύθυνσης ως δείκτες είναι οι BX, SI, DI, BP, οι οποίοι μπορεί να συνταχθούν και σε συνδυασμούς, δίνοντας έτσι 24 διαφορετικούς τρόπους σύνταξης διευθύνσεων μνήμης. Οι τρόποι αυτοί είναι οι εξής :

Τρόποι διευθυνσιοδότησης μνήμης

1) Διευθυνσιοδότηση [ADDR16] Παράδειγμα : MOV AX, [0200]
2) Διευθυνσιοδότηση [BP+ADDR8] Παράδειγμα : MOV AX, [BP+27]
3) Διευθυνσιοδότηση [BP+ADDR16] Παράδειγμα : MOV AX, [BP+8765]
4) Διευθυνσιοδότηση [BP+SI] Παράδειγμα : MOV AX, [BP+SI]
5) Διευθυνσιοδότηση [BP+SI+ADDR8] Παράδειγμα : MOV AX, [BP+SI+4D]
6) Διευθυνσιοδότηση [BP+SI+ADDR16] Παράδειγμα : MOV AX, [BP+SI+900A]
7) Διευθυνσιοδότηση [BP+DI] Παράδειγμα : MOV AX, [BP+DI]
8) Διευθυνσιοδότηση [BP+DI+ADDR8] Παράδειγμα : MOV AX, [BP+DI+77]

9) Διευθυνσιοδότηση [BP+DI+ADDR16]	Παράδειγμα : MOV AX, [BP+DI+11EE]
10) Διευθυνσιοδότηση [BX]	Παράδειγμα : MOV AX, [BX]
11) Διευθυνσιοδότηση [BX+ADDR8]	Παράδειγμα : MOV AX, [BX+2B]
12) Διευθυνσιοδότηση [BX+ADDR16]	Παράδειγμα : MOV AX, [BX+1234]
13) Διευθυνσιοδότηση [BX+SI]	Παράδειγμα : MOV AX, [BX+SI]
14) Διευθυνσιοδότηση [BX+DI]	Παράδειγμα : MOV AX, [BX+DI]
15) Διευθυνσιοδότηση [BX+SI+ADDR8]	Παράδειγμα : MOV AX, [BX+SI+A8]
16) Διευθυνσιοδότηση [BX+DI+ADDR8]	Παράδειγμα : MOV AX, [BX+DI+59]
17) Διευθυνσιοδότηση [BX+SI+ADDR16]	Παράδειγμα : MOV AX, [BX+SI+4C7E]
18) Διευθυνσιοδότηση [BX+DI+ADDR16]	Παράδειγμα : MOV AX, [BX+DI+91DE]
19) Διευθυνσιοδότηση [SI]	Παράδειγμα : MOV AX, [SI]
20) Διευθυνσιοδότηση [SI+ADDR8]	Παράδειγμα : MOV AX, [SI+23]
21) Διευθυνσιοδότηση [SI+ADDR16]	Παράδειγμα : MOV AX, [SI+FEDC]
22) Διευθυνσιοδότηση [DI]	Παράδειγμα : MOV AX, [DI]
23) Διευθυνσιοδότηση [DI+ADDR8]	Παράδειγμα : MOV AX, [DI+23]
24) Διευθυνσιοδότηση [DI+ADDR16]	Παράδειγμα : MOV AX, [DI+2552]

Σπύρος Καζαρλής
Επίκουρος Καθηγητής
Τμήμα Πληροφορικής & Επικοινωνιών
ΤΕΙ Σερρών