Άσκηση 1 – Εντολή MOV

Α. ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

Η εντολή **mov** είναι η πιο συνηθισμένη και χρησιμοποιείται για τη μεταφορά-φόρτωση δεδομένων εμπλέκοντας τους καταχωρητές και τη μνήμη ανάλογα την περίπτωση. Η εντολή **mov** έχει πολλές μορφές και ο προγραμματιστής επιλέγει την κατάλληλη ανάλογα με τη λειτουργία που επιθυμεί να πραγματοποιήσει. Σε αυτή την εντολή, ο προορισμός των δεδομένων είναι πάντα το πρώτο όρισμα.

Τρόποι χρήσης

Άμεση διευθυνσιοδότηση

Άμεση φόρτωση τιμής σε καταχωρητή.

παράδειγμα: **mov ax,10h** ; ax=10h

(το σύμβολο h σημαίνει ότι ο αριθμός είναι στο δεκαεξαδικό σύστημα)

Διευθυνσιοδότηση καταχωρητή

Για την προέλευση ή τον προορισμό των δεδομένων χρησιμοποιείται καταχωρητής

Παραδείγματα:

mov ax,cx ;ax=cx

mov ax,20h ;ax=20h

mov si,es:[bx] ;si=es:[bx] (το si φορτώνεται με το περιεχόμενο δύο θέσεων μνήμης ξεκινώντας από τη διεύθυνση es:[bx], όπου es δείχνει περιοχή μνήμης και το περιεχόμενο του bx τη διεύθυνση μέσα σε αυτό το τμήμα

Άμεση Διευθυνσιοδότηση μνήμης

Η προέλευση μπορεί να είναι απευθείας μια διεύθυνση μνήμης

Παραδείγματα:

mov ax,aLabel ;όπου aLabel αντιπροσωπεύει μια διεύθυνση. Εδώ θα μεταφερθούν 2 byte

mov al,[0000] ;φόρτωση απευθείας από τη διεύθυνση 0000

Έμμεση Διευθυνσιοδότηση μέσω καταχωρητή

Η προέλευση καθορίζεται μέσω ενός καταχωρητή (π.χ. BX, SI, DI)

Παράδειγμα:

mov ax,[bx] ;πρέπει προηγουμένως το bx να περιέχει τη διεύθυνση που θα προσπελάσουμε

Σχετική Διευθυνσιοδότηση βάσης

Πρόκειται για συνδυασμό άμεσης και έμμεσης διευθυνσιοδότησης. Η διεύθυνση προέλευσης προκύπτει από την απόκλιση σε σχέση με μια σταθερή διεύθυνση.

Παράδειγμα: **mov al,[bx+4]** ;η διεύθυνση καθορίζεται από το

;περιεχόμενο του bx προσαυξημένο κατά 4

Διευθυνσιοδότηση ευρετηρίου βάσης

Σε αυτή τη μορφή, η διεύθυνση προέλευσης προκύπτει από μια διεύθυνση βάσης, προσαυξημένη κατά ένα μεταβλητό αριθμό (περιεχόμενο καταχωρητή) και ένα σταθερό αριθμό. Η ευελιξία αυτής της εντολής βασίζεται στον μεταβλητό αριθμό.

Παράδειγμα:

mov ax,[bx+si+4]

Σημείωση: Ανάλογα τον Assembler (μετατροπή από Assembly σε γλώσσα μηχανής) που χρησιμοποιούμε, η σύνταξη των εντολών μπορεί να διαφέρει (π.χ. στον τρόπο που τοποθετούνται οι αγκύλες)

Στο πειραματικό μέρος θα πρέπει να δοκιμάσετε την εκτέλεση των εντολών που ακολουθούν και να γράψετε για κάθε περίπτωση τα αντίστοιχα σχόλια.

Τα σχόλια αυτά περιλαμβάνουν:

(α) αν η εντολή που σας έχει δοθεί είναι σωστή. Αν εμφανίζεται σφάλμα, θα πρέπει να βρείτε γιατί συμβαίνει αυτό.

(β) αν η εντολή αναγράφεται στη μνήμη με διαφορετικό τρόπο από αυτόν που την πληκτρολογήσατε και γιατί

(γ) ποιο είναι το αποτέλεσμα της εντολής (μνήμη και στοιχεία του μικροεπεξεργαστή), εξήγηση του αποτελέσματος

(δ) ποιο ήταν το προηγούμενο περιεχόμενο και ποιο είναι τώρα μετά την εκτέλεση της εντολής

Η εντολή **mov**

Παράδειγμα:

Δοκιμάζουμε την εντολή **mov al,09**

Σχόλια:

(α) Η εντολή είναι σωστή (δεν εμφανίζεται σφάλμα)

(β) Εμφανίζεται με την ίδια μορφή όπως την πληκτρολογήσαμε στη μνήμη (περιβάλλον λογισμικού)

(γ) Φορτώνει στο low byte (λιγότερο σημαντικό byte) του AX τον αριθμό 09 ανεξάρτητα από το υπόλοιπο περιεχόμενο του καταχωρητή.

(δ) Προηγούμενο περιεχόμενο AX=0000, νέο περιεχόμενο AX=0009

Συμβουλή: Τυπώστε την άσκηση και συμπληρώστε με το στυλό σας τις απαντήσεις στο εργαστήριο. Αυτά τα φύλλα άσκησης θα τα χρησιμοποιήσετε για το διάβασμα της ύλης του εργαστηρίου.

Α. ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ

1. Να δοκιμάσετε και να μελετήσετε ως προς τη λειτουργία τις ακόλουθες εντολές:

Σημείωση: Μπορείτε αν θέλετε να γράψετε τη μια εντολή μετά την άλλη και να τις δοκιμάσετε συνολικά. Σε κάθε περίπτωση, θα πρέπει να ελέγχετε το περιεχόμενο του IP για να γνωρίζετε κάθε φορά από πού θα συνεχιστεί η εκτέλεση των εντολών.

1.1 **mov ax,09**

Σχόλια:

1.2 **mov ax,0ffh**

Σχόλια:

1.3 **mov ax,0ffffh**

Σχόλια:

1.4 **mov ah,05**

Σχόλια:

1.5 **mov al,01**

Σχόλια:

1.6 **mov bh,-4**

Σχόλια:

1.7 **mov al,ds:[0001]**

Σχόλια:

1.8 **mov ax,ds:[0001]**

Σχόλια:

1.9 **mov al,cs:[0100]**

Σχόλια:

1.10 **mov ax,cs:[0100]**

Σχόλια:

1.11 **mov [0000],1fh**

Σχόλια:

1.12 **mov al,1fh**
mov [0001],al

Σχόλια:

2. Αποθήκευση συμβολοσειρών στην περιοχή δεδομένων

Είναι γνωστό ότι, ο υπολογιστής χρησιμοποιεί συγκεκριμένα σετ συμβόλων προκειμένου να είναι δυνατή η αποτύπωση πληροφοριών και δεδομένων κυρίως για το χρήστη. Ένα τέτοιο σετ είναι το ASCII 7bit/8bit. Για να αποθηκευτεί η συμβολοσειρά «Panos» στην περιοχή της μνήμης θα χρησιμοποιήσουμε μια εντολή mov.

Το σύμβολο-χαρακτήρας «P» έχει κώδικα ASCII 50h. Για να αποθηκευτεί αυτό το σύμβολο στη διεύθυνση 0000 της περιοχής δεδομένων θα χρησιμοποιήσουμε τις ακόλουθες εντολές:

mov al,50h
mov ds:[0000],al

Συμπληρώστε και δοκιμάστε τις υπόλοιπες εντολές που απαιτούνται, ώστε να αποθηκευτούν στις επόμενες θέσεις μνήμης τα υπόλοιπα σύμβολα για τη συμβολοσειρά «Panos».

Μπορώ να αποφύγω τη φόρτωση τιμής στον καταχωρητή al? Γράψτε τον τρόπο

Πίνακας ASCII 7bit

Dec	Hex	Oct	Char	Binary	Dec	Hex	Oct	Char	Binary
0	00	000	NUL	0000 0000	64	40	100	@	0100 0000
1	01	001	SOH	0000 0001	65	41	101	A	0100 0001
2	02	002	STX	0000 0010	66	42	102	B	0100 0010
3	03	003	ETX	0000 0011	67	43	103	C	0100 0011
4	04	004	EOT	0000 0100	68	44	104	D	0100 0100
5	05	005	ENQ	0000 0101	69	45	105	E	0100 0101
6	06	006	ACK	0000 0110	70	46	106	F	0100 0110
7	07	007	BEL	0000 0111	71	47	107	G	0100 0111
8	08	010	BS	0000 1000	72	48	110	H	0100 1000
9	09	011	HT	0000 1001	73	49	111	I	0100 1001
10	0A	012	LF	0000 1010	74	4A	112	J	0100 1010
11	0B	013	VT	0000 1011	75	4B	113	K	0100 1011
12	0C	014	FF	0000 1100	76	4C	114	L	0100 1100
13	0D	015	CR	0000 1101	77	4D	115	M	0100 1101
14	0E	016	SO	0000 1110	78	4E	116	N	0100 1110
15	0F	017	SI	0000 1111	79	4F	117	O	0100 1111
16	10	020	DLE	0001 0000	80	50	120	P	0101 0000
17	11	021	DC1	0001 0001	81	51	121	Q	0101 0001
18	12	022	DC2	0001 0010	82	52	122	R	0101 0010
19	13	023	DC3	0001 0011	83	53	123	S	0101 0011
20	14	024	DC4	0001 0100	84	54	124	T	0101 0100
21	15	025	NAK	0001 0101	85	55	125	U	0101 0101
22	16	026	SYN	0001 0110	86	56	126	V	0101 0110
23	17	027	ETB	0001 0111	87	57	127	W	0101 0111
24	18	030	CAN	0001 1000	88	58	130	X	0101 1000
25	19	031	EM	0001 1001	89	59	131	Y	0101 1001
26	1A	032	SUB	0001 1010	90	5A	132	Z	0101 1010
27	1B	033	ESC	0001 1011	91	5B	133	[	0101 1011
28	1C	034	FS	0001 1100	92	5C	134	\	0101 1100
29	1D	035	GS	0001 1101	93	5D	135	]	0101 1101
30	1E	036	RS	0001 1110	94	5E	136	^	0101 1110
31	1F	037	US	0001 1111	95	5F	137	_	0101 1111
32	20	040	SPace	0010 0000	96	60	140	`	0110 0000
33	21	041	!	0010 0001	97	61	141	a	0110 0001
34	22	042	"	0010 0010	98	62	142	b	0110 0010
35	23	043	#	0010 0011	99	63	143	c	0110 0011
36	24	044	$	0010 0100	100	64	144	d	0110 0100
37	25	045	%	0010 0101	101	65	145	e	0110 0101
38	26	046	&	0010 0110	102	66	146	f	0110 0110
39	27	047	'	0010 0111	103	67	147	g	0110 0111
40	28	050	(	0010 1000	104	68	150	h	0110 1000
41	29	051	)	0010 1001	105	69	151	i	0110 1001
42	2A	052	*	0010 1010	106	6A	152	j	0110 1010
43	2B	053	+	0010 1011	107	6B	153	k	0110 1011
44	2C	054	,	0010 1100	108	6C	154	l	0110 1100
45	2D	055	-	0010 1101	109	6D	155	m	0110 1101
46	2E	056	.	0010 1110	110	6E	156	n	0110 1110
47	2F	057	/	0010 1111	111	6F	157	o	0110 1111
48	30	060	0	0011 0000	112	70	160	p	0111 0000
49	31	061	1	0011 0001	113	71	161	q	0111 0001
50	32	062	2	0011 0010	114	72	162	r	0111 0010
51	33	063	3	0011 0011	115	73	163	s	0111 0011
52	34	064	4	0011 0100	116	74	164	t	0111 0100

53	35	065	5	0011 0101	117	75	165	u	0111 0101
54	36	066	6	0011 0110	118	76	166	v	0111 0110
55	37	067	7	0011 0111	119	77	167	w	0111 0111
56	38	070	8	0011 1000	120	78	170	x	0111 1000
57	39	071	9	0011 1001	121	79	171	y	0111 1001
58	3A	072	:	0011 1010	122	7A	172	z	0111 1010
59	3B	073	;	0011 1011	123	7B	173	{	0111 1011
60	3C	074	<	0011 1100	124	7C	174	|	0111 1100
61	3D	075	=	0011 1101	125	7D	175	}	0111 1101
62	3E	076	>	0011 1110	126	7E	176	~	0111 1110
63	3F	077	?	0011 1111	127	7F	177	DEL	0111 1111