Άσκηση 2
Εντολές ADD, SUB, XCHG, NOP, INC, DEC, ADC

ΑΠΑΝΤΗΣΕΙΣ

1. Πρόσθεση

Δίνεται το ακόλουθο πρόγραμμα:

mov ax,[0000]
mov bx,[0002]
add ax,bx
mov [0004],ax
	α) συμπληρώστε τις διευθύνσεις (περιοχή δεδομένων) που λείπουν, ώστε να προστεθούν οι 16bit αριθμοί που είναι αποθηκευμένοι στις θέσεις 0000 (low byte),0001 (high byte) και 0002 (low byte), 0003 (high byte) και το αποτέλεσμα να αποθηκευτεί στις θέσεις 0004 (low byte),0005 (high byte).

β) Γράψτε τα περιεχόμενα της περιοχής δεδομένων (data segment) για τις ακόλουθες διευθύνσεις:

0000 = CD	0001 = 20	0002 = FF
0003 = 9F	0004 = 00	0005 = EA

* Σημειώστε τα περιεχόμενα που βλέπετε στο πρόγραμμα τα οποία μπορεί να διαφέρουν στη δική σας περίπτωση

γ) Δοκιμάστε το παραπάνω πρόγραμμα στον debugger και γράψτε τόσο τα νέα περιεχόμενα της περιοχής δεδομένων (των διευθύνσεων που επηρεάζονται), όσο και την κατάσταση του bit κρατούμενου (c) του καταχωρητή κατάστασης. Αιτιολογήστε την απάντησή σας.

AX=20CD, BX=9FFF
Μετά την πρόσθεση
AX=C0CC
[0004]=CC, [0005]=C0
c=0

δ) Δοκιμάστε το ίδιο αρχικό πρόγραμμα στον debugger αντικαθιστώντας όμως το **ax** με το **al** και το **bx** με το **bl**. Σχολιάστε τις διαφορές στη λειτουργία και τα αποτελέσματα.

Νέο πρόγραμμα	Σχολιασμός

mov al,[0000]
mov bl,[0002]
ad al,bl
mov [0004],al
	al=CD
	bl=FF
	al=CC, c=1
	[0004]=CC
	Για την αποτύπωση του τελικού αριθμού θα πρέπει να χρησιμοποιηθεί και το κρατούμενο στην αριστερότερη θέση.

2. Εντολές αύξησης και μείωσης (INC, DEC)

α) δοκιμάστε το ακόλουθο πρόγραμμα και σχολιάστε αναλυτικά τη λειτουργία του (π.χ. παλιό και νέο περιεχόμενο μνήμης και καταχωρητών)

```
mov bx,0000
mov ax, ds:[bx]
inc bx
inc bx
mov cx,ds:[bx]
add ax,cx
inc bx
inc bx
mov ds:[bx],ax
```

Σχολιασμός

mov ax,ds:[0000], άρα AX=20CD

Με τη διπλή αύξηση του BX (INC BX, INC BX), BX=0002

mov cx,ds:[bx] (CH=[0003], CL=[0002])

στην πρόσθεση

```
   20CD
+  9FFF
----------
   C0CC
```

Άρα,

[0005]=C0, [0004]=CC

β) Τι θα συμβεί αν το παραπάνω πρόγραμμα τροποποιηθεί ως εξής ;

```
mov bx,0000
mov ax, ds:[bx]
inc bx
mov cx,ds:[bx]
add ax,cx
inc bx
mov ds:[bx],ax
```

Σχολιασμός

Ax=20CD

bx=0001 (λόγω της αύξησης)

cx=FF20

μετά την πρόσθεση το bx αυξάνεται και γίνεται 02, άρα

[0003]=AH, [0002]=AL

και προσοχή στο κρατούμενο για τη σωστή αποτύπωση του κρατούμενου

γ) δίνεται το ακόλουθο πρόγραμμα:

mov bx,0004
mov ax,ds:[bx]
dec bx
dec bx
mov cx,ds:[bx]
add ax,cx
 	
dec bx
 	
dec bx
 	
mov ds:[bx],ax
	γ1) Συμπληρώστε τις εντολές που λείπουν, ακολουθώντας τη λογική του προγράμματος

γ2) Να σχολιάστε τη λειτουργία του παραπάνω προγράμματος και να το συγκρίνετε με το πρόγραμμα στο βήμα 2α)

Ίδια λογική με τα προηγούμενα, απλά εδώ ξεκινάμε από υψηλότερη διεύθυνση και καταλήγουμε στη διεύθυνση 0000.

3. Εντολή πρόσθεσης με κρατούμενο

α1) μετατρέψτε τους αριθμούς 362 και 1233 (χρησιμοποιήστε την αριθμομηχανή στον υπολογιστή) στο δεκαεξαδικό σύστημα

$362_{(10)} = 16A_{(16)}$ $1233_{(10)} = 4D1_{(16)}$

α2) να γίνει η πρόσθεση των δύο παραπάνω δεκαεξαδικών αριθμών (Υπόδειξη: θα προστεθούν ξεχωριστά τα low και high byte. Αν προκύψει κρατούμενο στα low byte, τότε προστίθεται στα high byte).

```
  01 6A
 +04 D1
-----------

 0000 0001 0110 1010
+0000 0100 1101 0001
 -------------------------
 0000 0110 0011 1011

   0     6     3     B
```

α3) Επαληθεύστε το αποτέλεσμα της παραπάνω πρόσθεσης συμπληρώνοντας και δοκιμάζοντας στον debugger το ακόλουθο πρόγραμμα:

```
mov ax,16Ah

mov bx,4D1h

add ax,bx
```

β) Δοκιμάστε και σχολιάστε (αναλυτικά αλλά και συνολικά ως προς το αποτέλεσμα που παράγεται) τη λειτουργία του ακόλουθου προγράμματος:

```
mov ax,0ffffh
mov bx,01
add ax,bx
mov [0000],ax
adc cx,0
mov [0002],cx
```

Σημείωση: στις πράξεις θα πρέπει να παρατηρείτε και την κατάσταση του bit c (κρατούμενο) του καταχωρητή κατάστασης

Σχόλια

ax=FFFF, bx=0001 add ax,bx => ax=0000, c=1 [0000]=00, [0001]=0 adc cx,0 => cx=1 mov [0002],cx [0000] [0001] [0002] 00 00 01	FFFF + 1 ---------- 010000 01 1o byte 00 2o byte 00 3o byte

4. Αφαίρεση – Εντολή SUB

α) Αποθηκεύστε τη λέξη «hello» στην περιοχή δεδομένων χρησιμοποιώντας τον δεκαεξαδικό κώδικα ASCII κάθε συμβόλου και ξεκινώντας από τη διεύθυνση 0000 (Υπόδειξη: Χρησιμοποιήστε όπου απαιτείται, καταχωρητή 16bit για να φορτώνετε μαζί δύο θέσεις μνήμης).

Κώδικας

Ο κώδικας ASCII για τα σύμβολα του μηνύματος, είναι h 68h e 65h l 6Ch

```
l       6Ch
o       6Fh

η αποθήκευση στη μνήμη θα είναι ως εξής:

[0000]  [0001]  [0002]  [0003]  [0004]
  'h'     'e'     'l'     'l'     'o'

Πρόγραμμα
mov ax,6568h
mov [0000],ax

mov ax,6C6Ch
mov [0002],ax

mov al,6Fh
mov [0004],al
```

β) Χρησιμοποιήστε την εντολή SUB για να μετατρέψετε την παραπάνω λέξη με κεφαλαία γράμματα, δηλαδή σε «HELLO» στην περιοχή δεδομένων. Συμβουλευτείτε τον πίνακα ASCII για να βρείτε τη διαφορά στον κώδικα που έχουν τα πεζά με τα κεφαλαία γράμματα.

Κώδικας

Παρατηρώντας τον πίνακα ASCII παρατηρούμε ότι ο κώδικας για κεφαλαία και πεζά γράμματα διαφέρει κατά 20h. Έτσι, αν αφαιρέσουμε το 20h από τον κώδικα κάθε πεζού γράμματος, θα προκύψει ο κώδικας του αντίστοιχου κεφαλαίου.

π.χ.

```
mov ax,[0000]
sub al,20h
sub ah,20h
mov [0000],ax
```

Έτσι, τα σύμβολα «h» και «e», έχουν μετατραπεί αντίστοιχα σε «H» και «E».

Την ίδια διαδικασία ακολουθούμε και για τα υπόλοιπα σύμβολα του μηνύματος.

5. Ανταλλαγή περιεχομένων – Εντολή XCHG

α) Να αναπτυχθεί πρόγραμμα που ανταλλάσσει το περιεχόμενο δύο καταχωρητών (θα πρέπει να φορτώσετε σε αυτούς κάποια τιμή αρχικά). Σε αυτό το πρόγραμμα δεν θα χρησιμοποιήσετε την εντολή XCHG.

```
mov al,01       ;τυχαία τιμή
mov bl,09       ;τυχαία τιμή
mov cl,al       ;προσωρινή αποθήκευση του πρώτου αριθού
mov al,bl       ;ο δεύτερος αριθμός αποθηκεύεται στη θέση του πρώτου
mov bl,cl       ;ο πρώτος αριθμός ανακτάται από την προσωρινή θέση
                ;και αποθηκεύεται στη θέση του δεύτερου αριθμού
```

β) Δοκιμάστε το ακόλουθο πρόγραμμα
mov ax,01fah
mov bx,09
xchg ax,bx

Σημειώστε τα περιεχόμενα των καταχωρητών ax,bx πριν και μετά την εκτέλεση της εντολής XCHG και σχολιάστε.

```
ax=1fa
bx=09

μετά την εκτέλεση της εντολής XCHG

ax=09
bx=1fa
```

γ) Να αναπτυχθεί πρόγραμμα που να ανταλλάσσει τα περιεχόμενα των διευθύνσεων 0000 και 0001 της περιοχής δεδομένων

```
mov al,[0000]
mov ah,[0001]
xchg al,ah
mov [0000],al
mov [0001],ah
```

δ) Να αναπτυχθεί πρόγραμμα που να ανταλλάσσει τα 16bit περιεχόμενα των θέσεων μνήμης (περιοχή δεδομένων) [0000][0001] και [0002][0003] αντίστοιχα.

```
mov ax,[0000]
mov bx,[0002]
xchg ax,bx
mov [0000],ax
mov [0002],bx
```

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