אסמבלר / שפת סף

המעבד - לב המערכת, תפקידיו:

1. לפקח על פעולות מרכיבי המערכת
2. לתאם את זמני פעולתם
3. יוזם התקשרות לסביבה בהתאם לתוכניות

המעבד מחובר לזכרון וליחידות קלט / פלט
שלושת היחידות יחד - המעבד, הזכרון ויחידות קלט פלט מהוות את המחשב.
זכרון - מחולק לשני זכרונות עיקריים:

Rom - זכרון לקריאה בלבד.

בחלק הזה שמורים תכוניות בסיסיות להפעלת המערכת וטבלאות בסיסיות
כמו טבלאות תרגום לשפת מכונה (שפת סף)

זכרון RAM - זכרון זמין, זכרון נדיף

הזכרון הזמין לא נועל כולו לשימוש המשתמש
בחלק מהזכרון נטענים תוכניות המשמשות את המערכת, מהזכרון המורחב (דיסק קשיח, או התקני I/0 אחרים) כמו תכנית הפעלה ה-Bios הגדרות חומרה, פסיקות ישרות וכד'.

חלק מזכרון ה-RAM נועד לתכנות, ישנה תוכנית מיפוי זכרון המאפשרת להשתמש גם בחלקים שלא מיועדים לתכנות (או שימוש כללי) אם הם פנויים.

אמצעי I/0 מחוברים דרך יציאות המחשב, לכל יציאה יש כתובת, וזהו כתובת הרכיב.

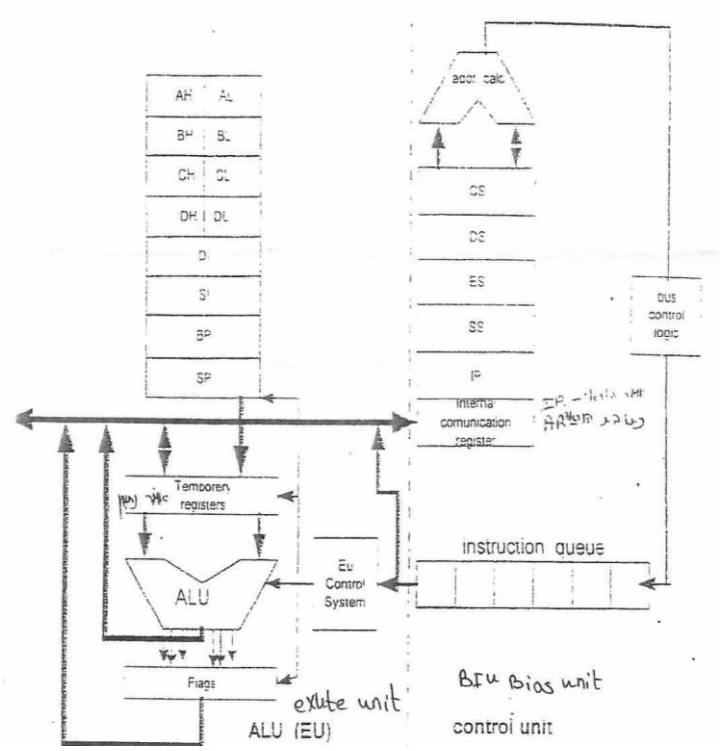
גם לכל זכרון יש כתובת, כך שכל נתון שמגיע מזוהה מהיכן הגיע ע"פ כתובתו.

המעבד מחובר לזכרון ולאמצעי קלט / פלט באמצעות Bus, פסי נתונים כתובות ובקרה כשפסי הבקרה משמשים להגדרת מצב קיים קלט / פלט נתונים כדי שהמעבד יתחיל לעבוד הוא צריך לקבל הוראות, בלחיצה על כפתור אתחול המחשב מופעלת תוכנית post השמורה ב Rom ומורה למעבד מהיכן להעתיק את מערכת ההפעלה ושאר הדברים הנצרכים לצורך תפעול, לזכרון הRAM.
המעבד יכול לעבוד בכל פעם על פקודה אחת בלבד.

time sharing - חלוקת מנות זמן בין היישומים השונים שמתבצעים בו זמנית, כך נראה כאילו הדברים מתבצעים בו זמנית, אך למעשה בכל פסיקת זמן יתבצע משהו אחד בלבד.
ישנן מערכות כמו מערכות זמן אמת בהם הדברים חייבים להתבצע בו זמנית - מערכות כאלו מכילות מס' מיקרו מעבדים.

והן תומכות ב Privigal level PL - הגדרות גישה ועדיפות להתקנים מסוימים.

מבנה המעבד: cpu



Instruction queue: תור ההוראות. גודלו 6 בתים. כאמור המעבד צריך לקרוא הוראה כדי לבצע אותה. ההוראה שהמעבד קורא נכנסת לתור זה ואז מתחיל תהליך הפענוח והביצוע.
במעבדים מתקדמים יותר, תור ההוראות גדול יותר ויש שיטות מיוחדות לניהול התור.

Address calculating unit: יחידה לחישוב כתובות. תפקידה לחשב את הכתובת הפיזית.

סוגי האוגרים השונים - אוגרי סגמנטים

זכרון ה RAM מחולק לשלוש חלקים עיקריים

מרחב תוכנה, מרחב חומרה וזכרון הרחבה.

במרחב התוכנה משתמשים לתוכניות השונות במערכת.

המרחב הזה גם הוא מחולק לארבע חלקים.

Code Segment - מקטע קוד, אזור זכרון המשמש לשמירת הוראות ופקודות.

data Segment - מקטע זכרון המשמש לשמירת נתונים עבור תוכנית או תוכנה כגון משתנים וכד'.

stack Segment - אזור זכרון המשמש 1. לשימוש זמני של נתונים

2. לקריאה וחזרה מפונק' - כתובת חזרה נשמרת במחסנית.
3. בשימוש בפסיקות שרות (יוסבר בהמשך)

extra Segment - אזור זכרון המשמש בד"כ כזכרון נתונים עודף אך יש סט פקודות, כלומר מעגלים חשמליים מסוימים שבנויים באופן שההתייחסות לנתונים המוצבעים שמורים דווקא באזור זה. בפקודות מחרוזתיות.

SI ו DI משמשים כמצביעים לכתובות בזכרון השמורות בזכרון זה.

מחרוזת מחרוזת

מקור יעד

כל תוכנית שכתובה באסמבלר, או מתורגמת לאסמבלר מסווגת את הנתונים בתוכנית כלומר קוד, ומשתנים וכד' לסגמנטים המתאימים לצורך שמירה.

ביחידת הבקרה קיימים האוגרים CS - מכיל כתובת תחילת זכרון הקוד ביחס לזכרון הכללי.

DS - מכיל כתובת תחילת זכרון הנתונים ביחס לזכרון הכללי.

SS - מכיל כתובת תחילת זכרון המחסנית ביחס לזכרון הכללי.

es - מכיל כתובת תחילת זכרון העודף ביחס לזכרון הכללי.

בתוך הזכרון פונים לכתובת יחסית, כשבתחילת אזור מחושבת הכתובת מכתובת O.

יחסית לאותו אזור, הכתובת הפיזית תחושב: כ. יחסית + 10h* (כתובת תחילת אזור)

כלומר תוכן האוגר המצב לאזור

המעבד מחולק לשני חלקים המהוים מעין שני מעבדים נפרדים שפועלים לחוד בו זמנית ומקושרים ביניהם באמצעות תור הוראות.

1. **יחידת הבקרה Bios** - מקבלת כתובת נתון והוראה,

היחידה מפענחת את ההוראה, מתרגמת כתובת מכתובת יחסית לכתובת מוחלטת (יוסבר בהמשך) תוך כדי המעבר על ההוראות, מעדכנת כל פעם את האוגר Ip. אוגר מצביע על ההוראה הבאה לביצוע, כל הוראה מתורגמת נכנסת לתור ההוראות (כמובן תלוי באורך התור), בלי תלות בביצועי ההוראות. כל קפיצה, לולאה או התניה גורמת להשהיה כי יש למלא את תור ההוראות מחדש.

2. **היחידה הביצועית Exute** - שולפת מתור ההוראות הוראה מפוענחת, חלק מהנתונים היא שומרת באוגרי נתונים לצורך ביצוע. המעבד יכול להפעיל את המעגלים החשמליים לביצוע הוראות רק על נתונים שיושבים במעבד ולכן כל גישה לזכרון הנתונים לוקח זמן שעון - פסיקת שעות, מחזור שעון נוסף ע"מ לשלוף את הנתון מהזכרון ולשמור אותו באחד מאוגרי הנתונים.

ולכן פקודות אסמבלר שמתורגמות ישר למעגלים החשמליים עובדות ויעילות ביותר בשימוש באוגרי נתונים, מהדר שפה עילית טוב, יתרגם מראש כמה שיותר פקודות המשתמשות באוגרי נתונים.

המעבד מבצע את הפעולה הדרושה ע"י הפעלת המעגל הרצוי באמצעות מפענח הוראות להוראה מתאימה.

מעדכן את התוצאה באוגר לפעמים מעביר חזרה לזכרון דרך פסי הכתובות והנתונים.

ומעדכן גם את אוגר הדגלים.

אוגר הדגלים - אוגר אחד יחיד שמתעדכן כל פעם, ניתן לשמור את ערכו מפעולה לפעולה לפי הצורך.

פעולה מעדכנת - פעולת מעבד אשר משנה את מצב הדגלים. לא כל הוראה משנה את מצב הדגלים. לא כל הוראה משנה את מצב הדגלים והוראה המשנה את מצב הדגלים אינה משנה בהכרח את כולם.

מבנה אוגר הדגלים:

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

CF - Carry Flag **דגל הנשא.** ערכו "1" כאשר בפעולת החיבור שהתבצעה היה נשא (Carry). או שבפעולת החיסור היה **לווה** (Borrow).

PF - Parity Flag **דגל הזוגיות.** ערכו של דגל זה "1" כאשר מספר הסיביות המכילות "1" בתוצאה יהיה זוגי, ואם לא - ערכו יהיה "0".

AF **דגל נשא עזר.** הדגל מגיב כמו דגל הנשא, אך מתייחס למצב שבו יש נשא או לווה בין סיבית 3 ל-4 בפעולה. משמש בעיקר בפעולות BCD.

ZF - Zero Flag **דגל אפס.** ערכו "1" כאשר תוצאת החישוב היא אפס, ואם לא - ערכו יהיה "0".

SF - Sign Flag **דגל סימן.** ערכו "1" כאשר סיבית MSB בפעולה קודמת היה "1". הוא משמש לציון מספרים שליליים וחיוביים בשיטת המשלים ל-2.

TF - Trap Flag **דגל מלכודת.** ערכו "1" כאשר נמצאים בצורת עבודה של צעד יחיד, שבה מבצעים כל פקודה בנפרד ולא ברצף. הדבר שימושי בעיקר בכלים ל**ניפוי שגיאות תוכנה** (Debuggers). הדגל מציין למעבד להפסיק ביצוע בגמר כל פקודה ולהעביר שליטה לתוכנית פסיקת Single Step.

IF - Interrupt Flag **דגל פסיקה.** כשדגל זה מופעל הוא מתיר פסיקות במערכת. הוא ניתן לשליטה בתוכנה בפקודות Cli ו-Sti.

DF - Direction Flag **דגל כיוון.** משתמשים בדגל זה בפעולות על מחרוזות. ערכו של דגל זה קובע את כיוון ההתקדמות בכתובות הזכרון שעליהן מתבצעת הפעולה (לפנים או לאחור).

OF - Overflow Flag **דגל גלישה.** ערכו "1" כאשר תוצאת הפעולה חורגת מגודל האופרנד שעליו התבצע הפעולה. הוא מציין כי התוצאה אינה מדויקת. או אינה נכונה כלל. יש לבדוק גם את המצב של OF כדי לדעת אם תוצאת הפעולה האריתמטית אמנם נכונה.

האוגר IP מיחידת הבקרה מקבל כל פעם כ.הוראה הבאה לביצוע.
כתובת כל הוראה מחושבת לפי אזור הקוד, ההוראה נשלפת מאזור הקוד.
המעבד יודע לקרוא את הנתון הנשלף בתור הוראה ולא סתם נתון, כי נשלף מסגמנט ההוראה.
כך גם כשקורא נתון מתוך אזור הנתונים הוא יקרא אותו בתור נתון ולא בתור הוראה וכד'.

מראה כל הזכרונות זהה, רצף בתים בערך הקסא (תצוגה יפה של בנארי).

במעבד 8086 בית - שתי ספרות הקסא 8 ביט

מילה - ארבע ספרות הקסא 16 ביט

```
11 FF 76 FC E8 F4 41 83 C4 02 80 3E 8D 5D 00 75
96 5E 8B E5 5D C3 55 8B EC FF 76 04 B8 F5 07 50
E8 72 FF 8B E5 5D C3 55 8B EC 83 EC 12 56 E8 A2
46 89 46 F0 F6 46 F0 40 75 03 E9 37 01 8D 46 FA
50 8D 46 FE 50 8D 46 F2 50 8D 46 F6 50 E8 0D FE
83 C4 08 83 7E F6 00 75 06 8B 46 F2 89 46 F6 8B
FD 8C DB 53 81 C3 2D 00 03 DA 8C CD 8B C2 80 E4
0F B1 04 8B F2 D3 E6 8B CE D1 E9 4E 4E 8B FE 2B
E8 2B D8 8E C5 8E DB F3 A5 FC 8E DD 07 06 BF 00
01 33 F6 AD 95 BA 10 00 EB 2B AD 95 B2 10 EB 35
AD 95 B2 10 EB 36 AD 95 B2 10 EB 3B AD 95 B2 10
EB 5D AD 95 B2 10 EB 5E AD 95 B2 10 EB 5F AD 95
B2 10 72 08 A4 D1 ED 4A 74 F4 73 F8 33 C9 33 DB
D1 ED 4A 74 C5 D1 D3 D1 ED 4A 74 C4 D1 D3 85 DB
74 17 D1 ED 4A 74 BF D1 D3 80 FB 06 72 0B D1 ED
4A 75 04 AD 95 B2 10 D1 D3 2E 8A 8F 61 01 80 F9
```

אוגרי נתונים

כל האוגרים	AX		AH AL
בגודל מילה	BX	ניתנים לחלוקה	BH BL
4 ספרות	CX	ניתנים לחלוקה	CH CL
הקסא	DX		DH DL
	SI		
	DI	לא ניתנים לחלוקה.	
	BP		

אוגרי הנתונים נמצאים ביחידה הביצועית של המעבד, הם משמשים לאחסון זמני של נתון מלבד זאת, לכל אחד מהאוגרים תפקיד אחד **לפחות** מיוחד רק לו (לא נכנס כרגע).

Decimal	Hex	Octal	Binary	Graphic Character	ASCII Meaning
0	0	0	00000000		^@ NUL (null)
1	1	1	00000001	☺	^A SOH (start-of-header)
2	2	2	00000010	☻	^B STX (start-of-transmission)
3	3	3	00000011	♥	^C ETX (end-of-transmission)
4	4	4	00000100	♦	^D EOT (end-of-text)
5	5	5	00000101	♣	^E ENQ (enquiry)
6	6	6	00000110	♠	^F ACK (acknowledge)
7	7	7	00000111	•	^G BEL (bell)
8	8	10	00001000	◘	^H BS (backspace)
9	9	11	00001001	○	^I HT (horizontal tab)
10	A	12	00001010	◙	^J LF (line feed - also ^Enter)
11	B	13	00001011	♂	^K VT (vertical tab)
12	C	14	00001100	♀	^L FF (form feed)
13	D	15	00001101	♪	^M CR (carriage return)
14	E	16	00001110	♫	^N SO
15	F	17	00001111	☼	^O SI
16	10	20	00010000	►	^P DLE
17	11	21	00010001	◄	^Q DC1
18	12	22	00010010	↕	^R DC2
19	13	23	00010011	‼	^S DC3
20	14	24	00010100	¶	^T DC4
21	15	25	00010101	§	^U NAK
22	16	26	00010110	▬	^V SYN
23	17	27	00010111	↨	^W ETB
24	18	30	00011000	↑	^X CAN (cancel)
25	19	31	00011001	↓	^Y EM
26	1A	32	00011010	→	^Z SUB (also end-
27	1B	33	00011011	←	^[ESC (Escape)
28	1C	34	00011100	∟	^\ FS (field sep
29	1D	35	00011101	↔	^] GS
30	1E	36	00011110	▲	^^ RS (record se
31	1F	37	00011111	▼	^_ US
32	20	40	00100000		Space
33	21	41	00100001	!	!
34	22	42	00100010	"	"
35	23	43	00100011	#	#
36	24	44	00100100	$	$
37	25	45	00100101	%	%
38	26	46	00100110	&	&
39	27	47	00100111	'	'
40	28	50	00101000	(	(
41	29	51	00101001	)	)
42	2A	52	00101010	*	*
43	2B	53	00101011	+	+
44	2C	54	00101100	,	,
45	2D	55	00101101	-	-
46	2E	56	00101110	.	.
47	2F	57	00101111	/	/
48	30	60	00110000	0	0
49	31	61	00110001	1	1
50	32	62	00110010	2	2
51	33	63	00110011	3	3
52	34	64	00110100	4	4
53	35	65	00110101	5	5
54	36	66	00110110	6	6
55	37	67	00110111	7	7
56	38	70	00111000	8	8
57	39	71	00111001	9	9
58	3A	72	00111010	:	:
59	3B	73	00111011	;	;
60	3C	74	00111100	<	<
61	3D	75	00111101	=	=
62	3E	76	00111110	>	>
63	3F	77	00111111	?	?
64	40	100	01000000	@	@
65	41	101	01000001	A	A
66	42	102	01000010	B	B
67	43	103	01000011	C	C
68	44	104	01000100	D	D
69	45	105	01000101	E	E
70	46	106	01000110	F	F
71	47	107	01000111	G	G
72	48	110	01001000	H	H
73	49	111	01001001	I	I
74	4A	112	01001010	J	J
75	4B	113	01001011	K	K
76	4C	114	01001100	L	L
77	4D	115	01001101	M	M
78	4E	116	01001110	N	N
79	4F	117	01001111	O	O
80	50	120	01010000	P	P
81	51	121	01010001	Q	Q
82	52	122	01010010	R	R
83	53	123	01010011	S	S
84	54	124	01010100	T	T
85	55	125	01010101	U	U
86	56	126	01010110	V	V
87	57	127	01010111	W	W
88	58	130	01011000	X	X
89	59	131	01011001	Y	Y
90	5A	132	01011010	Z	Z

Decimal	Hex	Octal	Binary	Graphic Character	ASCII Meaning
91	5B	133	01011011	[	[
92	5C	134	01011100	\	\
93	5D	135	01011101	]	]
94	5E	136	01011110	^	^
95	5F	137	01011111	_	_
96	60	140	01100000	`	`
97	61	141	01100001	a	a
98	62	142	01100010	b	b
99	63	143	01100011	c	c
100	64	144	01100100	d	d
101	65	145	01100101	e	e
102	66	146	01100110	f	f
103	67	147	01100111	g	g
104	68	150	01101000	h	h
105	69	151	01101001	i	i
106	6A	152	01101010	j	j
107	6B	153	01101011	k	k
108	6C	154	01101100	l	l
109	6D	155	01101101	m	m
110	6E	156	01101110	n	n
111	6F	157	01101111	o	o
112	70	160	01110000	p	p
113	71	161	01110001	q	q
114	72	162	01110010	r	r
115	73	163	01110011	s	s
116	74	164	01110100	t	t
117	75	165	01110101	u	u
118	76	166	01110110	v	v
119	77	167	01110111	w	w
120	78	170	01111000	x	x
121	79	171	01111001	y	y
122	7A	172	01111010	z	z
123	7B	173	01111011	{	{
124	7C	174	01111100	|	|
125	7D	175	01111101	}	}
126	7E	176	01111110	~	~
127	7F	177	01111111	Δ	Del
128	80	200	10000000	Ç	
129	81	201	10000001	ü	
130	82	202	10000010	é	
131	83	203	10000011	â	
132	84	204	10000100	ä	
133	85	205	10000101	à	
134	86	206	10000110	å	
135	87	207	10000111	ç	
136	88	210	10001000	ê	
137	89	211	10001001	ë	

Decimal	Hex	Octal	Binary	Graphic Character	ASCII Meaning
138	8A	212	10001010	è	
139	8B	213	10001011	ï	
140	8C	214	10001100	î	
141	8D	215	10001101	ì	
142	8E	216	10001110	Ä	
143	8F	217	10001111	Å	
144	90	220	10010000	É	
145	91	221	10010001	æ	
146	92	222	10010010	Æ	
147	93	223	10010011	ô	
148	94	224	10010100	ö	
149	95	225	10010101	ò	
150	96	226	10010110	û	
151	97	227	10010111	ù	
152	98	230	10011000	ÿ	
153	99	231	10011001	Ö	
154	9A	232	10011010	Ü	
155	9B	233	10011011	¢	
156	9C	234	10011100	£	
157	9D	235	10011101	¥	
158	9E	236	10011110	₧	
159	9F	237	10011111	ƒ	
160	A0	240	10100000	á	
161	A1	241	10100001	í	
162	A2	242	10100010	ó	
163	A3	243	10100011	ú	
164	A4	244	10100100	ñ	
165	A5	245	10100101	Ñ	
166	A6	246	10100110	ª	
167	A7	247	10100111	º	
168	A8	250	10101000	¿	
169	A9	251	10101001	⌐	
170	AA	252	10101010	¬	
171	AB	253	10101011	½	
172	AC	254	10101100	¼	
173	AD	255	10101101	¡	
174	AE	256	10101110	«	
175	AF	257	10101111	»	
176	B0	260	10110000	░	
177	B1	261	10110001	▒	
178	B2	262	10110010	▓	
179	B3	263	10110011	│	
180	B4	264	10110100	┤	
181	B5	265	10110101	╡	
182	B6	266	10110110	╢	
183	B7	267	10110111	╖	
184	B8	270	10111000	╕	

תו	ערך עשרוני
א	128
ב	129
ג	130
ד	131
ה	132
ו	133
ז	134
ח	135
ט	136
י	137
ך	138
כ	139
ל	140
ם	141
מ	142
ן	143
נ	144
ס	145
ע	146
ף	147
פ	148
ץ	149
צ	150
ק	151
ר	152
ש	153
ת	154

Decimal	Hex	Octal	Binary	Graphic Character	ASCII Meaning
185	B9	271	10111001	╣	
186	BA	272	10111010	║	
187	BB	273	10111011	╗	
188	BC	274	10111100	╝	
189	BD	275	10111101	╜	
190	BE	276	10111110	╛	
191	BF	277	10111111	┐	
192	C0	300	11000000	└	
193	C1	301	11000001	┴	
194	C2	302	11000010	┬	
195	C3	303	11000011	├	
196	C4	304	11000100	─	
197	C5	305	11000101	┼	
198	C6	306	11000110	╞	
199	C7	307	11000111	╟	
200	C8	310	11001000	╚	
201	C9	311	11001001	╔	
202	CA	312	11001010	╩	
203	CB	313	11001011	╦	
204	CC	314	11001100	╠	
205	CD	315	11001101	═	
206	CE	316	11001110	╬	
207	CF	317	11001111	╧	
208	D0	320	11010000	╨	
209	D1	321	11010001	╤	
210	D2	322	11010010	╥	
211	D3	323	11010011	╙	
212	D4	324	11010100	╘	
213	D5	325	11010101	╒	
214	D6	326	11010110	╓	
215	D7	327	11010111	╫	
216	D8	330	11011000	╪	
217	D9	331	11011001	┘	
218	DA	332	11011010	┌	
219	DB	333	11011011	█	
220	DC	334	11011100	▄	
221	DD	335	11011101	▌	
222	DE	336	11011110	▐	
223	DF	337	11011111	▀	
224	E0	340	11100000	α	
225	E1	341	11100001	ß	
226	E2	342	11100010	Γ	
227	E3	343	11100011	π	
228	E4	344	11100100	Σ	
229	E5	345	11100101	σ	
230	E6	346	11100110	µ	
231	E7	347	11100111	τ	

Decimal	Hex	Octal	Binary	Graphic Character	ASCII Meaning
232	E8	350	11101000	Φ	
233	E9	351	11101001	Θ	
234	EA	352	11101010	Ω	
235	EB	353	11101011	δ	
236	EC	354	11101100	∞	
237	ED	355	11101101	φ	
238	EE	356	11101110	ε	
239	EF	357	11101111	∩	
240	F0	360	11110000	≡	
241	F1	361	11110001	±	
242	F2	362	11110010	≥	
243	F3	363	11110011	≤	
244	F4	364	11110100	⌠	
245	F5	365	11110101	⌡	
246	F6	366	11110110	÷	
247	F7	367	11110111	≈	
248	F8	370	11111000	°	
249	F9	371	11111001	∙	
250	FA	372	11111010	·	
251	FB	373	11111011	√	
252	FC	374	11111100	ⁿ	
253	FD	375	11111101	²	
254	FE	376	11111110	■	
255	FF	377	11111111		

מבנה הוראה

הערה; אופרנד2, אופרנד1 שם הוראה: תוית.

תוית: אופציונלית, מתן שם לכתובת הוראה

שמים תוית בצמוד להוראה רק במקרה שרוצים לקפוץ להוראה ממקום כשלהו בתכנית.

לדוג': mov AL, 3

go: ADD AL, 2

Jmp go

:go זה תוית, שם סתמי שנתנו לכתובת ההוראה שלה הצמדנו את התוית.

קטע הקוד שבדוג' מבצע לולאה אין סופית בכל פעם.

ההוראה Jmp go תחזיר אותו להוראה הקודמת.

אופרנד: אופרנד זה תא כלשהו שמבצעים עליו פעולה מסוימת או שהוא מעורב בפעולה מסוימת (כמו משתנה, אוגר)

אופרטור זה פעולה.

בהוראה יש מקס' שני אופרנדים, אם ההוראה מכילה שני אופרנדים האופרנד **השמאלי** נחשב **כיעד**, עליו מבוצעת הפעולה ובו נשמרת התוצאה. האופרנד **הימני** נחשב **כמקור**, הוא מעורב בפעולה אך אינו משתנה מביצוע.

בהוראות שאין בהם אופרנדים, ההוראה עצמה מגדירה על מה מבוצעת הפעולה והיכן היא נשמרת התוצאה.

כללים:

1. הוראה עובדת או על בתים או על מילים.
2. מס' קבוע יכול להיות בגודל בית או מילה, אלא א"כ הוא גדול מבית או מילה.
3. מס' בבסיס 16 שמתחיל באות מצריך 0 מוביל.
4. הנתונים בתוכנית נכתבים בבסיס 10 אא"כ צוין אחרת. h - להקסא דצימאלי

   b - לבינארי

   o - לאוקטלי
5. הנתונים יושבים בזכרון בבסיסי 16.

פעולות בסיסיות

לכל פעולה יש מעגל חשמלי תואם במעבד
המעגלים החשמליים במעבד בנויים לפי בתים או מילים.
פקודות אסמבלר עובדות על בתים או מילים, כשהפקודה עובדת על שני אופרנדים
היא עובדת על שני בתים או שתי מילים בלבד.

דוג'	ביצוע	תחביר	פקודה	
ADD AX,3 AX=AX+3	op1 ← op1+op2	ADD op1, op2	ADD	בפקודות בעלות שני אופרנדים op1 כלומר האופרנד השמאלי נחשב כיעד.
ADC Bx, cx Bx=Bx+cx+CF	op1←op1+op2+cf קודם	ADC op1, op2	ADC	
SuB Bx, SI Bx=Bx-SI	Op1←op1-op2	SuB op1, op2	SuB	
SBB D, DL DL=DL-DH- SBB DL, DH DL=DL-DH-CF	op1 ←op1-op2-cf קודם	SBB op1, op2	SBB	
inc AL AL=AL+1	op1 ←op1+1	inc op1	inc	
Dec AL AL=AL-1	op1 ←op1-1	Dec	Dec	
neg DI DI=-DI	neg op1	neg op1	neg	
Xchg AX, BX AX ↔ BX	op1 ↔ op2	Xchg op1, op2	Xchg	
mov AX, 3 AX=3 AX=0003	op1 ← op2	mov op1, op2	mov	

ניתן לתת שם בתוכנית לכל כתובת שהיא, משתנה - זהו שם שנותנים לכתובת נתון
תוית - מגדירה כתובת הוראה
כל שם של פונק' וכד' זה כתובת הוראה, אליה מנחים את המעבד לבצע.
הגדרת תוית: שם הוראה בצמוד להוראה
לדוג' num: mov AX, 4

כתובת ההוראה הנתונה מסומנת עם התוית num, המעבד יתרגם את הnum לכתובת הוראה.

הוראת cmp.

הוראה המשמשת לצורך השואה בין נתונים, ההשואה מתבצעת בהתאם להוראת SuB אלא שהתוצאה לא נשמרת, היא גורמת רק לעדכון הדגלים (כפי שהוסבר בכיתה עבור חיוביים בלבד ועבור מסומנים חיוביים ושליליים)

תחביר cmp op1, op2

ביצוע: op1 - op2 ← עדכון באוגר הדגלים

העדכון עפ"י op1 לדוג', אם op1 = 0 אזי דגל האפס שוה 1 וכן הלאה.

הוראת I/mul - כפל

תחביר: I/mul op1

שידוע שהוא

בגודל בית / מילה

ביצוע: כפל בתים: AX ← op1*AL

כפל מילים: DXAX ← op1*AX

הוראת mul משמשת לכפל מס' חיוביים בלבד, במידת הצורך התוצאה מוחבת באפסים.

הוראת Imul משמשת לכפל מס' מסומנים (חלקם חיוביים וחלקם שליליים)

הוראת DIV / I - חילוק

תחביר: I/DIV op1

ביצוע:

חילוק בתים: AX — AL - שלם
op1 — AH - שארית

חילוק מילים: DXAX — AX - שלם
op1 — DX - שארית

הוראת DIV משמשת לחילוק חיוביים בלבד במידת הצורך מרחיבים את הנתון לפני חילוק על AX

או על DXAX באפסים מובילים.

הוראת IDIV משמשת לחילוק מסומנים, במידת הצורך ההרחבה של הנתון לפני חילוק על AX או

על DXAX בהתאם לסימן

CBW - הרחבה על AX

CWD - הרחבה על DXAX

תרגול על דגלים - והוראות בסיסיות

1. רשמי את ערכם של האופרנד היעד ושל דגלי הנשא, אפס, גלישה וסימן לאחר ביצוע קטעי התכנית בשאלות להלן.

א.
```
Mov  BL, OABH
Mov  CL,  90H
Mov  CL,  Bc
```

ב.
```
Mov AX, 8FH
Mov BX, AX
ADD AX, Bx
```

ג.
```
Mov DL, 50H
Mov CH, OFFN
Sub DL, CH
```

ד.
```
Mov DL, 50H
Mov CH, OFFH
SUB CH, DL
```

ה.
```
Mov DL, 50H
Mov CH, OFFH
Mov AL,  CH
```

ו.
```
Mov CX, OFF H
Mov BX, 2
SuB CX, Bx
Iwc Bx
ADD CX, Bx
DEC CX
```

2. כתבי קטע תכנית שתחלק שני מספרים ע"י חיסור, יש לרשום את התוצאה בDH והשארית בCH.
3. בדקי עבור תו מסוים אם הוא ספרה או תו אלפא בתא.
אם הוא ספרה יש להכניס ל-cx 0 ועבור תו להכניס ל-cx 1.
4. כתבי תוכנית למציאת המחלק המשותף הגדול ביותר של שני מספרים לפי האלגוריתם של אוקלידיס. התוצאה תוכנס לCH.
5. כתבי תכנית שתחשב את סכום N המספרים הזוגיים הראשונים עבור א. n=20 ב. /n=50
6. כתבי קטע של תכנית שתחשב את סכום המספרים האי זוגיים הקטנים מ-100 הסכום יכנס לDx
7. חשבי !5, היכן תמצא התוצאה?
8. סכום 6 המספרים הראשונים בפיבונצי'
9. מייני את הנתונים שב CL, BL, AL בסדר יורד AL - הכי קטן CL - הכי גדול

JCXZ

JB / JA - אפשרי לא אפשרי

JL / JG - אפשרי אפשרי

נספחים

הוראות קפיצה

ההוראה	תיאור ההוראה	התנאי הנבדק	הוראה הפוכה	סוג הקפיצה
JE	Jmp if equal (=)	ZF=1	JNE	A
JNE	Jmp if not equal	ZF=0	JE	A
JZ	Jmp if zero	ZF=1	JNZ	A
JNZ	Jmp if not zero	ZF=0	JZ	A
JC	Jmp if carry	CF=1	JNC	B
JNC	Jmp if no carry	CF=0	JC	B
JO	Jmp if overflow	OF=1	JNO	A
JNO	Jmp if not overflow	OF=0	JO	A
JP	Jmp if parity	PF=1	JNP	A
JPE	Jmp if parity even	PF=1	JPO	A
JNP	Jmp if no parity	PF=0	JP	A
JPO	Jmp if parity odd	PF=0	JPE	A
JS	Jmp if sign	SF=1	JNS	A
JNS	Jmp if no sign	SF=0	JS	A
JA	Jmp if above (>)	CF=0, ZF=0	JNA	B
JNA	Jmp if not above (not >)	CF=1 or ZF=1	JA	B
JB	Jmp if below (<)	CF=1	JNB	B
JNB	Jmp if not below (not <)	CF=0	JB	B
JNBE	Jmp if not below or equal (not <=)	CF=0, ZF=0	JBE	B
JBE	Jmp if below or equal (<=)	CF=1 or ZF=1	JNBE	B
JAE	Jmp if above or equal (>=)	CF=0	JNAE	B
JNAE	Jmp if not above or equal (not >=)	CF=1	JAE	B
JG	Jmp if greater (>)	SF=OF or ZF=0	JNG	C
JNG	Jmp if not greater than (not >)	SF, OF or ZF = 1	JG	C
JL	Jmp if less than (<)	SF<>OF	JNL	C
JNL	Jmp if not less than (not <)	SF=OF	JL	C
JNLE	Jmp if not less than or equal (not <=)	SF=OF or ZF=0	JLE	C
JGE	Jmp if greater than or equal (>=)	SF=OF	JL	C
JNGE	Jmp if not greater or equal (not >=)	SF<>OF	JGE	C
JLE	Jmp if less than or equal (<=)	SF, OF or ZF = 1	JNLE	C

JCXZ

תרגילי חזרה - הוראות בסיסיות

1.
```
mov cx, 0
mov  AX, 25
ADD AX, -25h
cmp  AX, 0
Jne  end1
mov CX, 1
end1: nop
```

2.
```
mov CL, 5
mov BL, 3
mov DL, 0
Do1: inc BL
ADD DL, BL
Dec CL
cmp CL, 0
Jne DO1
nop
```

3.
```
mov DX, 100h
mov SI, 0
do1: ADD SI, DX
Sub DX, 5
cmp DX, 5
Jae do1
```

4.
```
mov AL, 20
mov DL, -20
cmp AL, DL
Jb Do1
mov CL, 1
Jmp end1
Do1: mov AL,-20h
cmp AL, DL
Jb Do2
mov CL, 2
Jmp end1
Do2: ADD AL, DL
end1: nop
```

תרגילי חזרה

חוקי / לא חוקי

א.
1. ADD AL, BH
2. mov CL, 260
3. MOV cx, -240
4. mov SI,CL
5. ADD DL, DI
6. ADD DL, DI
7. cmp 3, CL
8. mov CL, FEh
9. mov DL, FF

ב. מייני את המס' הבאים בהקסא (בגודל בית)

א. לפי מספרים חיוביים **ב.** לפי מספרים מסומנים

OABH	90h	-30h	60h	75	-20	20	-OABH
AB	90	DO	60	B4	EC	14	55

ג. כתבי את ערכי האוגרים בביצוע הפעולות הבאות:

1. mov AL, 30
   ADD AL, -30h
   Sub AL, -30

2. mov AL, OFFH
   inc AL
   mov SI, offh
   inc SI
   mov AH, 0
   ADD AX, SI

גישה לזכרון

בכל פקודה הפונה לזכרון, יש לציין את סוג הזכרון אליו פונים (באסמבלר אין הגבלה פיזית)
ואת גודל הזכרון (בית, מילה)
נוכל לציין את
סוג זכרון: 1. בצמוד לכתובת נציין את אוגר הסגמנט

[300h]:DS - כתובת 300h בזכרון הנתונים

[20h] : SS - כתובת 20h במחסנית וכד'.

2. הכתובת לא תרשם ככתובת קבועה, אלא באמצעות אוגר.

[DI] [SI] [BX+DI] [BX+SI] [BX] - מציינים כתובת בזכרון נתונים.

[Bp+DI] [Bp+SI] [Bp] - מציינים כתובת בזכרון המחסנית

ניתן לצרף לכל סוגריים מס' קבוע בלבד.

אין אפשרות לצרופים נוספים

לדוג' ~~[SI+DI]~~

הערה: בתוך סוגריים מרובעות

מותר להשתמש רק באוגרים Bx, SI, DI, Bp

אסור להשתמש ב-DX, CX, AX

או חלק מ-Bx רק Bx בכללותו

נוכל לציין את גודל זכרון

1. ע"י שימוש באופרטור ptr

2, [400h]: mov Byte ptr DS ← לכתובת 400h יכנס 02.

5234h, [400h]: Mov word ptr DS ← לכתובת 400h יכנס 34h ולכתובת 401 יכנס 52h.

2. ע"י האופרנד הנוסף שבגוף הפעולה.

[Bx], AL- mov AL בגודל בית, ולכן יגש לכתובת זכרון בזכרון הנתונים המוצבעת ע"י Bx ויקח בית.

דוג' אם Bx = 20h
ובכתובת 20h הנתון 50h
אזי AL = 50h

[Bx+SI], AX - mov AX בגודל מילה, ולכן יגש לכתובת זכרון בזכרון הנתונים המחושבת ע"י Bx+SI ויקח מילה כלומר יקח תוכן תא ספציפי והתא הבא אחריו.

דוג' אם Bx=2 SI=3
בכתובת 5: 85h
בכתובת 6: 32h
Ax = 3285h

שימי לב!

1. אסור שני תאי זכרון בפקודה אחת.
2. כתובות הזכרון ממוספרות משמאל לימין. (הערך שבכתובת הגבוה (מבחינת מס' סידורי של כתובות) יכנס למקום הגבוה - בחלק השמאלי של אוגר, והערך בכתובת הנמוכה יכנס למקום הנמוך כלומר לצד הימני של האוגר.
בהתיחסות לכתובות בגודל מילה.

הערך הגבוה ↔ כתובת גבוהה

הערך הנמוך ↔ כתובת נמוכה

תרגילים

1. מה תהיה תמונת הזכרון בכל אחד מהתרגילים הבאים, אם ידוע שמרחב הזכרון

או תאים 400H 410H מאופס לפני כל תרגיל.

א. 1.
```
mov  cx, 5
mov AX, OFFEEH
mov Bx, 400h
Do: mov [Bx], AH FF
mov [Bx+1], AL EE
inc Bx
inc Bx
Dec cx
cmp cx, 0
Jne do
```

2.
```
mov  cx, 5
mov AX, OFFEEH
mov Bx, 400h
Do: mov [Bx], AX
inc Bx
Dec cx
cmp cx, 0
Jne do
```

3.
```
mov  cx, 5
mov AX, OFFEEH
mov Bx, 400h
Do: mov [Bx], AX
inc Bx
inc Bx
Dec cx
cmp cx, 0
Jne do
```

3.
```
mov  cx, 5
mov AX, OFFEEH
mov Bx, 400h
Do: mov [Bx], AL
inc Bx
inc Bx
Dec cx
cmp cx, 0
Jne do
```

ב. חוקי / לא חוקי - נמקי

1. mov aL, [20h]
2. mov [20h], DS:AX
3. mov [BX + SI], 50h
4. mov [Bx], 450
5. mov [Bx], ax
6. mov SI, [SI]

.2

א.	צייני = < >		ב.	צייני חוקי ולא חוקי
	-4	FC		muL 3
	-2	FFFE		DIV Bp
	FFFF	FF		CBW BL
	80h	70h		ADD DX, DL
	95h	80h		ADD DL, CF
	'a'	'A'		

.3

א. מה מבצעת התוכנית

ב. מה תבצע התוכנית אם נוריד את השורה המסומנת ב***

ג. איזו הוראה נצטרך לשנות בעקבות השינוי בסעיף ב' ומהוה שינוי...

ד. מה גודלו מקסימלי של n

```
    N equ 14
    data segment
    Date ends
    Code segment
            Assume cs: code, ds: data
    Start: mov ax, data
            Mov ds, ax
            Mov di, 100h
            Mov cx, n
    Xx1: mov bp, cx
***         mov cx, n
              Xx2:  mov ax, bp
                    Mov bx, cx
                    Imul bx
                    Mov [di], ax
                    Inc di
            Loop  xx2
            Mov cx, 16
            Sub cx, n
            Xx3: mov byte ptr[di], 0
                Inc di
            Loop xx3
            Mov cx, bp
    Loop xx1
    Nop
    Code ends
    End start
```

4. א. הראי את תמונת הזכרון בסיום הקטע הבא, כל פקודה תלויה בחברתה.

```
mov cx, 3
1.  mov Ax, 2442h
2.  mov Bx, 200h
3.  go: mov [Bx], AX
4.          inc Bx
5.  Loop go
```

ב. איך יראה הזכרון אם אחרי שורה 4 נוסיף inc AX

ג. איך יראה הזכרון אם במקום inc AX נוסיף inc BX

5.

```
SuB AX, DX
ADD DX, AX
SuB AX, DX
neg AX
```

1. כתבי מה יתבצע כתוצאה מרצף הפקודות.
2. כתבי שתי דרכים נוספות לביצוע הנ"ל.

6. נתון ? nLdw

וכן הקטע הבא:

```
mov cx , nL
mov DX, 2
mov AL, 17
Loop next
cmp AL, 1
Jne do
next: mov AX, 9
        Jmp sof
do:  mov AX, 3
Sof:
```

מה ערכו של Ax אם ידוע:

1. nL=0
2. nL=1
3. nL=3

7.

א. הסבירי מה מבצעת התוכנית

ב. מה תבצע התוכנית אם במקום inc DI פעמיים נכתוב פעם אחת.

מה המשמעות לגבי התוכנית.

ג. איך יראה מערך התוצאה מא' וב'

```
Data Segment
    Array1 DB 1 Dup (03h, 02h, 05h, 06h, 07h)
    Array2 Dw 5 Dup (?)
Data ends
Code Segment
    Assume cs: code, ds:data
start: mov Ax, Data
        mov DS, ax
        mov CX, 5
        mov SI, 0
        mov DI, 0
        Loopex:
                mov Ax, 0
                mov Bx, 1
                mov Bp, cx
                mov cx, 0


mov CL, Array [SI]
Loopin:
        ADD AX, BX
        ADD BX, 2
Loop Loopin
mov Array 2 [DI], AX
inc SI
inc DI
inc DI


mov cx, Bp
Loop Loopex
sof: code ends
        end start
```

הגדרת משתנים

משתנים מגדירים ב- Data Segments

הגדרת משתנה:	ערך	גודל	שם משתנה
דוג':	5	DB	A

ניתן להגדיר משתנים בגדלים שונים

db - בית

dw - מילה

dd - מילה כפולה, 2 מילים, 4 בתים

dq - 8 בתים

dt - 10 בתים

פקודות אסמבלר עובדות רק על בתים או מילים, גדלים אלו מתיחסים רק להכנסת נתונים לזכרון.

ניתן להגדיר משתנה מבלי לאתחלו עם הסימן ?

A db ?

ניתן להקצות מקום בזכרון, בלי שם

5 db, הוא יכניס 05 לתא הזכרון הבא.

לדוג' A db 5

db 8

B dw 6

תמונת הזכרון:	ערך:	00	06	08	05
	כתובת:	3	2	1	0

ניתן להקצות מס' ערכים באותה שורה B db ,3 ,4 ,'a' ,-3

בפניה לשם משתנה הוא פונה לתא הראשון

אם נרצה לפנות לשאר האיברים יחסית לשם המשתנה שהוגדר בשורה זו, נפנה עם כתובת יחסית

B[1] = 04

B[3] = FD (-3)

אין משמעות אמיתית להגדרת משתנה, מערך מטר' וכד'

שם משתנה הוא מעין תמרור באמצע הזכרון

שם משתנה = כתובת וגודל

כל הזכרון הוא מערך אחד גדול וניתן לפנות מכל מקום בזכרון להמשך הזכרון ממנו והלאה.

(ניתן לפנות גם אחורה, אך זה עושה לפעמים בעיות)

הגדרת המשתנים היא ביחס להקצאה הראשונית של המשתנים בזכרון.

אחרי שנתון יושב בזכרון, אין שמירת מידע מהיכן הגיע לזכרון.
ובכל הוראה, בהתאם לאופן שבו נפנה לנתון זה תהיה צורת ההתייחסות אליו.

לדוג' A db 03 ,52 ,-4 ,'A'

B dw oAB5h, 2233h

C db 5

תמונת הזכרון:

03	34	FC	41
0	1	2	3

B5	OA	33	22	05
4	5	6	7	8

A = כתובת 0 גודל בית

B = כתובת 4 גודל מילה

C = כתובת 8 גודל בית

7 = 3 + כתובת = B[3] ← פניה לכתובת 7 בזכרון בגודל מילה

0522 כלומר יפנה לכתובת 7 וכתובת 9

6 = 6 + כתובת [6] A ← פניה לכתובת 6 בזכרון בגודל בית

0 יפנה לכתובת 6 שתוכנו 33h

כך גם ביחס למשתנים בגדלים גדולים ניתן לפנות דרך משתנים אחרים בגודל בית או מילה או לחילופין ניתן לפנות אליהם דרך שימוש באופרטור ptr.

A db 20

D dd 095ABCDh

E dw 35h, 22, -1

תמונת הזכרון:

14	CD	AB	95	00	35	00
0	1	2	3	4	5	6

16	00	FF	FF
7	8	9	A

כתובת 0 = A

כתובת 1 = D

כתובת 5 = E

A[3] - פניה לכתובת 3 בזכרון, תוכנו 95

Byte ptr D[3] - פניה לכתובת 4 בזכרון תוכנו 00

word ptr D[3] - פניה לכתובת 5 בזכרון תוכנו 00_

שימי לב! גם עבור מילים - הצעדים בזכרון תמיד בבתים, ז"א האינדקס מתיחס לכתובת ספציפית בזכרון. ובהגיע לכתובת המבוקשת בוחרים האם להתיחס לבית הספציפי, או לבית ולבית העוקב.

בדוג' הנ"ל הגענו לכתובת 3 בספירה של בתים ומשם לקחנו מילה (לא ספרנו 3 מילים)

כללים:

1. הגדרת מחרוזת תמיד בבתים, כל תו תופס בית וזה נכנס לזכרון לפי הסדר שבו נכתב במחרוזת. דוגמא: A db '3AB2'
   תמונת הזכרון: 33 41 42 32
2. מחרוזת **בגודל אחר** חוקית **עד שתי תוים** וזה נחשב כערך מחרוזתי
   A dw '34' - 34 33
   A dd '34' - 34 33 00 00
   לא חוקי - ~~A dd '343'~~
3. אסור שני תאי זכרון בפקודה אחת.
4. משתנה הוא תא זכרון
5. מותר קבוע ומשתנה

ההוראה Dup - מבצעת הקצאה למס' תאי זכרון ביחד.

num DB 3 Dup (2 ,-2 ,'2' ,'-2')

יקצה את תוכן הסוגריים 3 פעמים

02 FE 32 2D 32 02 FE 32 2D 32 02 FE 32 2D 32

ניתן לבצע Dup בתוך Dup עד 8 פעמים.

num DB 3 Dup (2 Dup (5 ,12)

05 oc 05 oc

05 oc 05 oc

05 oc 05 oc

תרגילים

1. מטרת התכנית הבאה: לתרגם טקסט מאותיות קטנות לאותיות גדולות:

הטקסט נתון בסגמנט הנתונים במערך TEXT שארכו נתון במשתנה N1. טבלת התרגום מוגדרת במערך TAVLA באורך N2 מילים.

א. מהו תוכן מערך TEXT לאחר בצוע התכנית?

ב. בתכנית יש שגיאה ולכן היא אינה מבצעת את הנדרש. כיצד יש לתקן את התכנית על מנת שתבצע את הנדרש?

ג. מהו השימוש במערך TAB1 ומהי השיטה של התכנית לבצוע תרגום של טקסט (לאחר תיקונה)?

```
N=44
   DATA SEGMENT
   TEXT    DB  'This sentence is written un small letters…'
   N1      DB N
   TAVLA DW 'Aa', 'Bb', 'Cc', 'Dd', 'Ee', 'Ff', 'Gg', 'Hh', 'Ii', 'Jj', 'Kk','Ll', 'Mm', 'Nn',
                 'Oo', 'Pp', 'Qq', 'Rr', 'Ss', 'Tt', 'Uu', 'Vv', 'Ww','Xx', 'Yy', 'Zz'
   N2      DB   26
   TAB1    DB   256 DUP (?)
   DATA ENDS
   SSEG SEGMENT STACK
           DW 100H DUP (?)
   SSEG ENDS
   PROG SEGMENT
           ASSUME DS: DATA, SS:SSEG, CS:PROG
           MAIN PROC FAR
              PUSH DS
              MOV AX, 0
              PUSH AX
              START: MOV AX, DATA
              MOV DS, AX
              MOV TAB1[0],0
              MOV CX, 255
           L1:MOV DI,CX
```

```
    MOV TAB1[DI]CL
    LOOP L1
    MOV C, 0
    MOV CL, N2
    MOV BX, 0
    MOV SI, BX
L2: MOV AX, TAVLA [SI]
    MOV BL, AH
    MOV TAB1[BX], AL
    INC SI
    INC SI
    LOOP L2
    MOV AX, 0
    MOV BX, 0
    MOV CX, 0
    MOV CL, N1
    MOV SI, BX
L3: MOV AL, TEXT[SI]
    MOV BL, AL
    MOV AL, TAB1[BX]
    MOV TEXT[SI], AL
    INC SI
    LOOP L3
    RET
MAIN ENDP
PROG ENDS
END MAIN
```

.2 (סעיף א' 8 נקודות, סעיף ב' 12 נקודות)

```
Data segment
Num1 db 128, -128, -7fh
Num2 dw 446, -1bah, -42h, 0606h, 'ab'
Data ends
```

א. הראי את תמונת הזכרון

ב. הניחי כי כל אחד מהאוגרים מאופס מראש, אין קשר לוגי בין הסעיפים השונים. כתבי את האוגרים או תאי הזכרון שמושפעים מהפקודות ואת ערכם.

1.
```
Mov al, num1+1
Mov bl, num1+2
Add al, bl
Cbw
```

2.
```
Mov si, offset num1
Lea di, num2
Mov al, byte ptr num2[si]
Mov dx, num2[si+1]
```

3.
```
Mov di, offset num2
Add al, [di]
Adc al, [di-1]
Adc al,
Byte ptr num2[di+1]
```

.3 (20 נקודות)

כתבי תוכנית שתגדיר מטריצת מספרים, המטריצה תוגדר כdb והיא תכיל מס' חיובים בלבד

מ n-1, הניחי כי אברי המטריצה מאותחלים.

כתבי את התוכנית בצורה מושלמת (כולל העטיפה נדרשת)

כתבי תוכנית שתגדיר מטריצת בתים בגודל nxn

על התוכנית לבדוק ולהחזיר תשובה בResult

אם בכל שורה ועמודה קיימים כל המס' מ-n-1

לדוגמא:

result =1
1	2	3	4
2	1	4	3
3	4	2	1
4	3	1	2

.4

```
    Mov SI, 0
    Mov AL, 5
Loop next
    ADD AL, AL
    Mov [SI], AL
next: inc SI
nop
```

.5 מה יתבצע אם

א. cx=3

ב. cx=1

ג. cx=0

```
ARR  DW     34H, 89H, OFFH, 59, 144, 253
1.  LABEL1:     MOV     DI, 0
2.              MOV     CX,5
3.              MOV     SI,2
4.              LEA     BX, ARR
5.              MOV     DH,0
6.  LABEL2:     MOV     AL,[BX+DI]
7.              CMP     AL,[BX+SI]
8.              JL      NO
9.              MOV     AH, [BX+SI]
10.             MOV     [BX+SI], AL
11.             MOV     [BX+DI], AH
12.             MOV     DH;1
13. NO:         ADD     SI,2
14.             ADD     DI,2
15.             LOOP    LABEL2
16.             CMP     DH,1
17.             JE      LABEL1
```

א. כתבי את תלוכן המערך ARR לפני כל ביצוע של הפקודה JE LABEL1

ב. חזרי על א' אם מחליפים את הפקודה __ בפקודה JB NO

ג. מה תפקידו של אוגר DH

מחסנית

מחסנית היא אזור זכרון קיים, בשפות אחרות אנחנו לא נגשים אליה ישירות ובכל זאת משתמשים בה.
כל קריאה לפונק' ולפסיקה משתמשת במחסנית כפי שנראה להלן.
גם באסמבלר ניתן להתעלם ממנה ולהשאיר אותה רק לשימושים כנ"ל.
ובכל זאת משתמשים בה, מאחר והשימוש בה יעיל יותר משימוש בזכרון רגיל כך שמשתמשים בה כתאי עזר (כל מה שעושים באסמבלר, כותב הקומפיילר עבור תוכנה ספציפית) לשימוש זמני של אוגרים וכד'.

בהגדרת מחסנית לא מגדירים מחסנית חדשה, אלא הגדרת מחסנית ממקמת את האוגר SP כך שיצביע לראש המחסנית - במקום ידוע. SP בכל מקרה מצביע למקום כלשהו במחסנית (אין זכרון ריק). כך שלא חייבים להגדיר מחסנית.

פעולות המחסנית הייחודיות לה:

sp - push op1 יורד ב2 בתים ובמילה הזו (כלומר בין ההצבעה הקודמת להצבעה הנוכחית נכנס נתון בגודל מילה (גודל בית לא חוקי))

sp - push op1 עולה ב2 בתים והנתון שבין ההצבעה הקודמת להצבעה הנוכחית,
המילה הזו מועתקת לאורפנד שכתוב בגוף ההוראה.

שם פונק sp - call יורד ב2 ובמילה הזו הוא דוחף את כ.ההוראה שאחרי ההוראה call למחסנית

כלומר Ip - אוגר ההוראה הבאה לביצוע מועתק למחסנית ו-Ip מקבל את כ.הפונק' שהתקבלה משם הפונק'
(שם פונק' - כ.ההוראה הראשונה בפונק'
שם פונק' משמש כתוית להוראה זו)

Sp - Ret עולה ב2 והנתון הזה נשלף כלומר מועתק מהמחסנית לאוגר Ip כך שהתוכנית תוכל להמשיך עפ"י הנדרש.
כותב התוכנית אחראי שביציאה מהפונק', כלומר כשנרצה לחזור לתוכנית כתובת החזרה, כתובת ההוראה שאחרי ההוראה call אכן תופיע בראש המחסנית - Sp יצביע עליה.
שליחת נתונים לפונק' וחזרת נתונים מהפונק' גם היא דרך המחסנית.

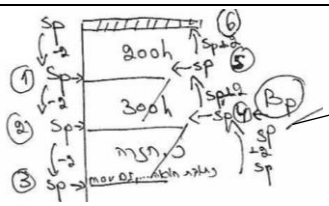
```
Start:
    mov  cx, 10
    mov  Bx, 200h
    mov SI, 300h
1   push Bx
2   push SI
3   call copybyte
    mov DI, 700h
    push Bx
    push DI
    mov cx, 10
    call copybyte
copybyte: 4 pop Dx   שמירת כ.חזרה באוגר
            mov Bp, sp
          5 popDI
            mov Bx, [Bp+2]
        go: mov aL, [DI]
            xchg aL, [Bx]
            mov [DI], aL
            inc Bx
            Dec DI
            Dec cx
            cmp cx, 0
        Jne go
          6 pop Bx
          7 push Dx
      8 ret
```

הפקודות שקשורות במחסנית ממוספרות

ניתוח התוכנית: נניח כי נתונה תמונת זכרון מכתובת 200, הבית הראשון כתובת 200 הבא 201 וכן הלאה

הכתובות מאותחלות בערך ידוע מ200h עד 20Ah	200:	OA OB OC OD OE OF O1 O2 O3 O4 O5 200 201 202 203 204 205 206 207 208 209 20A 00 90 80 70 60 50 40 30 20 2F7 2F8 2F9 2FA 2FB 2FC 2FD 2FE 2FF
גם מכתובת 2F7 עד 300 מאות שאר הזכרון לא מאותחל.	300:	10 300
	700:	

בתחילה עומד כאן
המספור עפ"י מס' הפקודות בתוכנית



תמונת המחסנית החל מהוראה 1

כ. חזרה = DX
נוכחי ב4 Bp=sp
DI=300
BX=[Bp+2]=200h
בהוראה 6 Bx=200 והמחסנית ריקה
בהוראה 7 Dx נכנס למחסנית - כ.חזרה
בהוראה 8 נשלפת כ. החזרה לאוגר IP וחוזרים להמשך תוכנית.

תרגיל

עבור שני התרגילים המחסנית מוגדרת Dw 20h Dup (?)

בכל אחד מהתרגילים הראי את תמונת המחסנית ותוכן אוגרים או תאי זכרון.

```
1.    mov ax, data
      mov ds, ax
    1 push ax
      mov Bp, 5
do1 :mov Bx, 1
      mov Dx, 3
    2 push Bp
    8 call func 1
    3 pop Bp
      ADD Bp, Bp
      cmp Bp, 20
      Jne do1
      Jmp sof
      func - 1 proc near
      mov Bp, sp
do    ADD Dx, Bx
      cmp Dx, 5
      Jne do
    4 push Dx
    5 push Bx
      mov Bp, [Bp]
    6 push Bp
    7 ret 6
      func - 1 endp
2.    A1 db 35, 35h, '35', 00, 0, 0
      A2 dw 35, 3500h, '35'
1 push word ptr A1
2 push A2
3 pop word ptr A1
4 pop A2
5 push word ptr A1+2
6 push A1+2
7 mov
8 mov Bp, offset A2
9 mov DF 4
10 push [Bx +DF]
11 push Ds: [Bp+DI]
12 pop [Bx + DI]
13 pop Ds: [Bp+DI]
```

מצורפת תכנית מקור באסמבלי. קרא אותה בעיון. מה יהיה תכנם של _____

אחת מהשגרות PROC..b, PROC...a ?

לאיזה כתובת יועבר תוכן זה?

```
DATA      SEGMENT
    D1    DB  '0987654321'
    D2    DW 2 DUP (?)
    D3    EQU 1234
    D4    DW (?)
DATA      ENDS
STACK SEGMENT STACK
    DB    8 DUP (?)
TOS LABEL WORD
STACK ENDS
CODE SEGMENT

START:    MOV AX, DATA
          MOV DS, AX
          MOV ES, AX
          JMP   MAIN
PROC_A:
          MOV BP, SP
          MOV BX, [BP+4]
          MOV AL, [BX+1]
          MOV AH, [BX+3]
          RET
PROC_B:
          CALL PROC_A
          RET
MAIN:
          MOV BX, 1
          MOV AX, 3
          PUSH AX
          PUSH BX
          CALL   PROC_A
          MOV BP, SP
          MOV DS: [BP], AX
          CALL     PROC_B
          POP      CX
          POP      DX
          MOV BP, SP
SOF:      NOP
          MOV   AH, 4CH
          INT     21H
CODE ENDS
          END START
```

1. מה מראה המחסנית

כתבי את ערכי האוגרים DI, SI, Sp, Bp

בכל פקודה המסומנת ב*

```
dseg segment
        a1  db  81h, '82h'
        a2 dw 81, '81'
        a3ddoc3248861h
dseg ends
slck Segment stack
        Dw Is Dup (?)
Stck ends

F1 proc near
        push bp
        mov bp, sp
      * mov di, [Bp+6]
      * push a2
      * pop a2[2]
      * pop bp
      * ret 4
    F1  endp
start:  mov ax, data
        mov ds, ax
        mov SI, 5
        push SI
        mov DI, 3
        push DI
        call F1
code ends
end start
```

2. את המחסנית, מה מבצעת התכנית?

```
MAIN    PROC    FAR
0       MOV     AX, M
2       MOV     BX, N
4       MOV     CX, X
6       MOV     DX, X
8       PUSH    BX
A       PUSH    DX
C       PUSH    AX
E       PUSH    CX
*10     CALL    HASHEV
12      RET
...
HASHEV  PROG    NEAR
32      POP     SI
34      CALL    SQUARE
*36     POP     BX
38      MUL     BL
3A      MOV     BX, AX
3C      CALL    SQUARE
*3E     POP     CX
40      MUL     CL
42      ADD     BX, AX
44      PUSH    SI
46      RET
...
SQUARE  PROG    NEAR
60      POP     DX
*62     POP     AX
64      MUL     AL
*66     PUSH    DX
68      RET
```

נתון m, n, x, y משתנים dw המכילים נתון קטן מ256.

יש מען במחסנית, התייחס למען.

תרגול - מבוא לקלט פלט

שאלה 1

כתבי תכנית המקבלת במילה A מספר בינארי ומחזירה במערך B בן 5 בתים את ערכו הדצימאלי בהצגה קרקטריאלית.

לדוגמא: אם המילה A מכילה את הערך ההקסא-דצימאלי 14EBH (מס' בנארי - מס' בהקסא)

שערכו הדצימאלי הוא 5355 - התכנית תחזיר במערך B.

30 35 33 35 35

A וB יוגדרו בסגמנט הנתונים בצורה הבאה:

A dw (?)

B DB 5 dup (?)

הנחיה: כתבי פונק' שתבצע המרה של מספר שכתובתו שמורה במחסנית.

נתון כי מס' הספרות המקסימלי הוא 5.

שאלה 2

כתוב תכנית המקבלת מס' A בן n ספרות בבסיס 8 בהצגה קרקטריאלית ומחזירה במשתנה B את ערכו העשרוני בהצגה קרקטריאלית.

לדוג' A = 14732 n=5 (מקס' 5) - התוכנית מקבלת A: 31 34 37 33 32

התוכנית תחזיר בB (6618) 30 36 36 31 38

העזרי בפונק', הפונק' תחזיר את הספרות הקרקטריאליות של B במחסנית, ובתוכנית תציב בערך.

קלט פלט int 21h

ע"מ להשתמש בפקודות קלט / פלט עלינו לפנות לטבלאות חיצוניות int 21h / int 10h

לשם כך צריך להשתמש עבור התוכנית הראשית במבנה הבא

```
code Seg ment
    Assme….
main proc far
    push DS
    mov Ax, 0
    push Ax
    …
ret
main ends
code ends
end main
```

בכל פעם שנשתמש במבנה הנ"ל הוא ידחוף למחסנית שתי מילים (בכתובות הגבוהות)

מילים אלו תשלפנה מהמחסנית ביציאה מהתוכנית (בהוראת ret)

הטבלה int 2/4 - טבלת פסיקות, כל פסיקה מבצעת משהו אחר.

פונים לטבלה עם מס' פסיקה (מציבים מס' פסיקה בah)

ובהתאם לכך יודע לבצע את הדרישה.

ah=1 קלט תו בודד והצגתו למסך, נכנס ל-aL

ah=2 פלט תו בודד השמור בDL

ah=7 קלט תו בודד מבלי להציג על המסך נכנס ל-aL

ah=9 פלט מחרוזת מדפיס את המחרוזת עד לזיהוי '$'

המחרוזת מוגדרת בזכרון, בסיומה דולר, מדפיס מחרוזת שכתובתה שמורה בDX

ah=10 קלט מחרוזת לשם כך יש להגדיר בזכרון למחרוזת בגודל n

string db n+1, 1+2 dup (?)

↓

מס' תוים

שרוצים לקלוט כולל enter

המחרוזת תקלט מהמקום השלישי,

במקום השני יציב את מס' התוים שנקלט בפועל לא כולל enter

בסוף המחרוזת יציב enter

אין אפשרות לקלוט מס' ניתן לקלוט ולפלוט תוים בלבד.

לביצוע הקלט, DX שוה לכתובת המקום אליו רוצים לבצע קלט (כתובת string)

תרגילים:

1. נתונה ההגדרה הבאה 17 _____

```
Data Segment
    names db 'Saiat', 'miri$', 'michal$'
    mess 1 db 'קיים$'
    mess 1 db 'לא קיים$'
Data ends
```

כתבי תוכנית שתקלוט שם מהמשתמש ותבדוק האם השם הנקלט קיים במערך השמות.
אם כן התוכנית תדפיס קיים, אחרת התוכנית תדפיס לא קיים.

2. כתבי תכנית לקלט מס' בן 5 ספרות - המספר הנקלט דצימאלי
יש להדפיס את המספר בהקסאי ובאוקטלי.
מה צריך לשנות בתוכנית אם המס' הנקלט יהיה הקסא.

3. כתבי תוכנית שתקלוט סיסמא עד 5 ספרות, סיום הקלט ב↳ הסיסמא לא תוצג, במקום
כל ספרה יוצג '*' על התוכנית להודיע בסיום הקלט אם הסיסמא שגויה או תקינה.

שאלות

א. פרסי את תמונת הזכרון

```
A  db  -127, -120, oabh, -53h, 89h, 90, 25
       81    88    ab    Ad    78  5A  19
B  dw  -127,  88h, oabh, -53h, 89h, 90, 25
       81ff   88   oo   aboo   AD FF  89  oo 5A oo 19 oo
```

ב. כתבי את הערכים העשרוניים של המס' הנתונים כמספרים חיוביים בלבד, התבססי על A

129 136 171 173 137 90 25

ג. כתבי את הערכים העשרוניים של המס' הנתונים כמספרים מסומנים, התבססי על A

-127 -120 -85 -83 -119 90 25

שפת סף - שאלות

1. לפניך תוכנית בשפת אסמבלי הכוללת שגרה רקורסיבית בשם REC.

שגרה זו מקבלת - באמצעות המחסנית - מספר כארגומנט (פרמטר) בבסיס דצימלי, ומדפיסה אותו למסך בתצוגה הקסאדצימלית.

בתוכנית חסרים **ארבעה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)- (4), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
DATA
SEGMENT
    P DW 65535, -1
DATA ENDS
SSEG SEGMENT STACK 'STACK'
    DB 100 H DUP()
SSEG ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START:  MOV     AX, DATA
        MOV     DS, AX
        CMP     P, 0
        JE      EXIT
        MOV     BX, 16
        PUSH    P+2
        __________(1)____________
        CALL    REC
EXIT:   MOV     AH, 4CH
        MOV     BP, SP
        CMP     WORD PTR [BP+4], 0
        JNE     CON
        MOV     AH, 2
        JMP     DIS
COM:    ____________(2)____________
        MOV     AX, [BP+4]
        DIV     BX
        PUSH    DX
        PUSH    AX
        CALL    REC
DIS:    CMP     WORD PTR [BP+6], -1
        JE      SOF
        MOV     DL, [BP+6]
        ADD     DL, '0'             ;(*)
        CMP     DL, '9'
        ____________(3)____________
        SUB     DL, '0'+10
        ADD     DL, 'A'
SHOW:   INT     21H
SOF:    POP     BP
        ____________(4)____________
```

2. מה יודפס על המסך כשתגיע התוכנית לתווית EXIT? הסבר מדוע.

הסעיפים 3 - 6 שלהלן אינם תלויים זה בזה.

3. אם נחליף **בתוכנית הנתונה** את השורה **P DW 65535 ,-1** בשורה P DW -1 ,-1, האם תשתנה תשובתך לסעיף ב'? ענה "כן" או "לא" ונמק בקצרה.
4. אם נחליף **בתוכנית הנתונה** את השורה **MOV DS ,AX** בשורה **MOV ES ,AX**, האם ישתנו ביצועי התוכנית? ענה "כן" או "לא" ונמק בקצרה.
5. האם **התוכנית הנתונה** מתייחסת למספר שמודפס למסך כאל מספר מסומן? ענה "כן" או "לא" ונמק בקצרה.
6. אם נחליף **בתוכנית הנתונה** את השורה המסומנת ב-(*) בשורה **ADD, DL, 30H**, האם ישתנו ביצועי התוכנית? ענה "כן" או "לא" ונמק בקצרה.

שאלה __

לפניך תוכנית בשפת אסמבלי, הכוללת שגרה **רקורסיבית** בשם REC. השגרה מקבלת באמצעות מחסנית את הארגומנט NUM, שהוא מספר עשרוני שלם חיובי הגדול מ-0. אם ספרותיו של המספר (החל מספרת האחדות) מהוות סדר עולה, שגרה זו מחזירה באוגר CL את הערך 1, אחרת - השגרה מחזירה באוגר CL את הערך 0.

לדוגמה: בעבור המספר 85541 השגרה תחזיר 1, ובעבור המספר 8154 השגרה תחזיר 0.

הנחת יסוד: ערכו של המספר שמשוכן במשתנה NUM הוא 65535 לכל היותר.

א. בשגרה חסרים **שישה** ביטויים, המסומנים במספרים בין סוגריים עגולים. רשום במחברת הבחינה את מספרי הביטויים החסרים (1)- (6), בסדר עולה, וכתוב ליד כל מספר את הביטוי החסר שהוא מייצג.

```
SSEG SEGMENT STACK 'STACK'
DB 100 DUP ( )
SSEG ENDS
DSEG SEGMENT
NUM DW 5 4 3 2 1
ZERO DW 0
DSEG ENDS
CODE SEGMENT
        ASSUME CS: CODE, DS:
        DSEG
START:  MOV     AX, DSEG
        MOV     DS, AX
        PUSH    NUM
        PUSH    ZERO
        CLC
        CALL    REC
        ADD     SP, 2           ;(**)
        MOV     AH, 4CH
        INT     21H
```

```
REC:    PUSH BP
        MOV     BP, SP
        JC      NEEXT
        MOV     AX, _______ (1)
_________
        MOV     BX, 10
        XOR     CL, CL
NEXT:   XOR     DX, DX
        _______(2)_______
        CMP     DL,
_________(3)_________
        _________(4)_________
        OR      AX, AX
        JNE     NEX
        _________(5)_________
        JMP     EXIT
NEX:    _________(6)_________
        STC
        CALL    REC
EXIT:   POP     BP
        RET     2
CODE ENDS
END START
```

ב. אם נשמיט מן התוכנית הנתונה את השורה המסומנת ב-(**), האם שינוי זה ישפיע על ביצועי התוכנית? ענה "כן" או "לא".

ג. אם נחליף בתוכנית הנתונה את השורה OR AX, AX בשורה CMP AX, 0, האם שינוי זה ישפיע על הערך שתחזיר השגרה? ענה "כן" או "לא".

שאלה

לפניך תכנית בשפת אסמבלי הכוללת שגרה **רקורסיבית** בשם FUNCI אשר מקבלת באמצעות מחסנית את NUM, שהוא מספר עשרוני שלם חיובי הגדול מאפס.

```
DATA    SEGMENT
        NUM DW 1053
DATA    ENDS
SSEG SEGMENT STACK 'STACK'
        DB 100H DUP (?)
SSEG    ENDS
CODE    SEGMENT
    ASSUME CS: CODE, DS: DATA, SS: SSEG
    START: MOV AX, DATA
        MOV     DS, AX
        PUSH    NUM
        MOV     CX, 0
        INT 21H
FUNC1   PROC
        PUSH    BP
        MOV     BP, SP
        MOV     AX, 0
        JZ      SOF
        MOV     BX, 10
        XOR     DX, DX          ;(1)
        DIV     BX
        INC     CX              ;(2)
        PUSH    AX
        CALL    FUNC1
                                ;(3)
SOF:    POP     BP
        SET 2
FUNC1   ENDP
CODE    ENDS
END     START
```

א. מה יהיה תוכנו של האוגר CX, בבסיס עשרוני, לאחר ביצוע התכנית הנתונה?

ב. אם נחליף בתכנית הנתונה את השורה NUM DW 1053 בשורה NUM DW 25612, מה יהיה תוכנו של האוגר CX, בבסיס עשרוני, לאחר ביצוע התכנית?

ג. אם נחליף בתכנית הנתונה את השורה NUM DW 1053 בשורה NUM DW -64483, האם תוכנו של האוגר CX, בבסיס עשרוני, יהיה שונה מהערך שהוחזר באוגר CX בסעיף א'? ענה "כן" או "לא".

שאלה

לפניך תכנית בשפת אסמבלי:

```
DATA    SEGMENT
                A  DB  7, 3, 4, 5, 6, 2
                LEN = $-A
                P  DW  A, LEN
DATA ENDS
    SSEG SEGMENT STACK 'STACK'
        DB  100H DUP (?)
    SSEG ENDS
    CODE SEGMENT
        ASSUME  CS: CODE, DS: DATA
START:      MOV AX, DATA
            MOV DS, AX
            PUSH P
            PUSH P+2
            CALL TROUBLE
SOF:        MOV AH, 4CH
            INT 21H
TROUBLE:    PUSH BP
            MOV BP, SP
CHAZOR:     MOV CX, [BP+4]
            DEC CX
            XOR SI, SI
            MOV BX, [BP+6]
AGAIN:      MOV AL, [BE]
            CMP AL, [BX+1]
            JBE CON
            XCHG AL, [BX+1]
            MOV [BX], AL
            INC SI
CON:        INC BX
            LOOP AGAIN
            OR SI, SI
            JNE CHAZOR
            POP BP
            RET 4
CODE ENDS
END START
```

א. רשום במחברת הבחינה את הערכים שישוכנו, לאחר הרצת התכנית, במערך A, החל באיבר הראשון וכלה באיבר האחרון.

ב. השורה 2, 6, 5, 4, 3, 7 A DB מוחלפת בשורה 14, 10, 5 A DB. רשום במחברת הבחינה את הערכים שישוכנו, לאחר הרצת התכנית, במערך A, החל באיבר הראשון וכלה באיבר האחרון.

ג. מה מבצעת התכנית? **ענה במשפט אחד בלבד.**