# חומרת מחשב

## רכיבי זיכרון:

כל שינוי במעבד הוא מבצע פעולה:
2hz – שתי פעולות בשניה.
100hz- 100 פעולות בשניה.

**ADRESS BUS – ערוץ הכתובות:**
ערוץ הכתובות הוא מערכת חוטים המחוברת לרכיבי המחשב שתפקידה להעביר קוד בינארי המשמש לאיתור כתובות.
**DATA BUS – ערוץ הנתונים:**
ערוץ הנתונים הוא מערכת חוטים המחוברת לרכיבי המחשב שתפקידה להעביר קוד בינארי המהווה אינפורמציה כלשהיא.

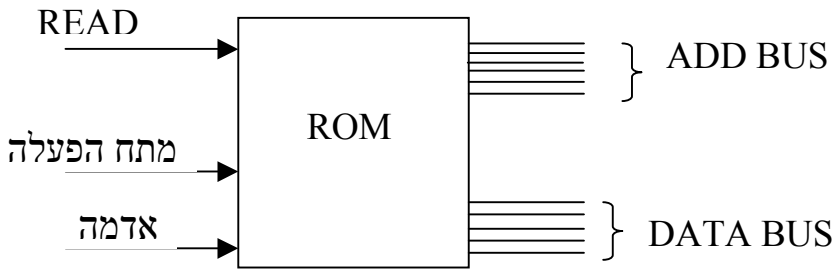
## סוגי זיכרונות:

זיכרונות חשמליים.
זיכרונות מכניים (אלקטרו-מכניים).
זיכרונות אופטים.

### זיכרונות חשמליים:

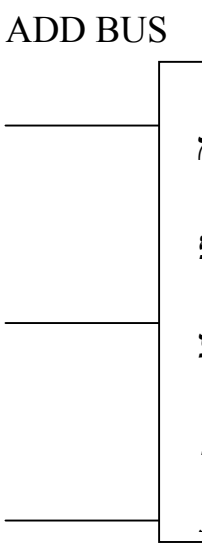
זיכרונות קבועים.
זיכרונות זמניים.

### זיכרונות קבועים:




יש קשר קבוע בין כמות הקווים לבין גודל הרכיב, כאשר יש מערכת אינטגרלית (הרכיבים משולבים) כמות ה – ADD היא לפי הרכיב הגדול ביותר.

| | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
|---|---|---|---|---|---|---|---|---|
| ADD 0 | | | | | | | | |
| ADD 1 | | | | | | | | |
| ADD 2 | | | | | | | | |
| ADD 3 | | | | | | | | |
| ADD 4 | | | | | | | | |
| ADD 5 | | | | | | | | |
| ADD 6 | | | | | | | | |
| ADD 7 | | | | | | | | |
| ADD | | | | | | | | |
| ADD | | | | | | | | |
| ADD 6 | | | | | | | | |
| ADD | | | | | | | | |




כל ריבוע בתוך המערכת מתאר בית אחד - תא היכול להכיל 1/0.
DATE – קווים המהווים את המידע שנכנס לרכיב. כל כתובת בזיכרון היא תמיד 1BATE = 8 BIT.
שלושת קווי הכתובות הם קווי ה – ADD. מכניס את קוד המפענח ואז המצביע מתייצב על הכתובת הנכונה ומוציא את המידע ע"י קווי הנתונים.
קווי הנתונים – להוצאת האינפורמציה מהכתובות או מהזיכרון.
קווי כתובות – המידע מגיע מבחוץ מהמעבד או מהרכיבים שונים. חד סטרי – מהתאים החוצה.

מתח הפעלה ואדמה מתחברים לחשמל.
– READ
כדי לקרוא צריך לתת פולס חשמלי. חוט הקריאה אמור להגיב בשינוי מצח. שלא רוצים לטפל ברכיב אז הוא צריך להיות על 0. שרוצים שהרכיב יפעל הוא צריך לקבל ADD כשיש שינוי עליה / ירידה וכשהשינוי מתייצב על המתח הוא מוכן לקבל את הקוד של ה – ADD.
כשמתבצע שינוי ברגל הקריאה הרכיב מוכן לקבל ADD הוא מתביית על הכתובת הנכונה ומוציא את המידע על גבי ה – DATE. התהליך מסתיים שהפולס מתייצב חזרה על ה – stand by.

:ROM
נפח: רכיבים ישנים מתחילת שנות השבעים. 1K-5K קטנים.
זמן גישה:

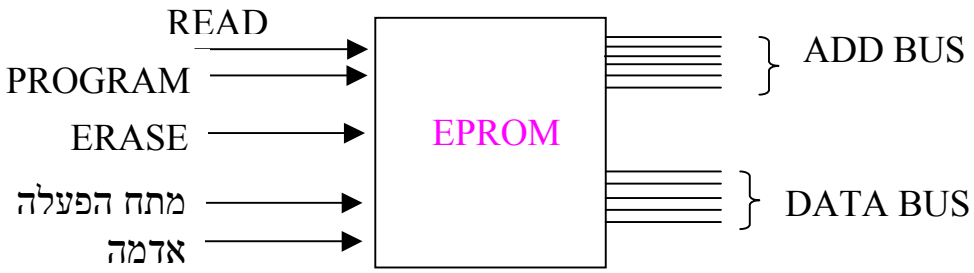

הזמן שלוקח לרכיב להגיב מבקשת הנתון עד יציאת הנתון. נמדד בשניות או בחלקי שניות.
ברכיבים אלו זה נמדד ב – 200Msec.
הנתונים והמידע הוא מידע שנוצר עם יצירת הרכיב. המזמין צריך לדעת מראש מה הוא רוצה שיהיה יצוק בתוך הרכיב. זאת אומרת הרכיב סגור, קבוע ללא שינויים.

יתרונות:
1. אין תלות במתח – אם מוצאים את המתח המידע לא נמחק.

חסרונות:
1. זמן גישה ונפח – עם הזמן היו צריכים רכיבים עם יותר נפח ומהירות גישה גדולה יותר.
2. צריך לשנות ולכתוב.

EPROM - (צריך להוסיף בריבוע)

רכיב שניתן לשנות ולכתוב בתוכו אחרי שהוא הגיע צרוב מהמפעל.



EPROM – כדי לחסוך בעלויות בקשנו רכיב ריק – לבן / חלק. רכיב ללא מידע ולכן תהליך הייצור שלו יותר זול. אפשר לקנות כמויות גדולות ולבצע הרבה שינויים.
אפשר למחוק את המידע שבפנים ע"י אור אולטרה סגול - ריאקציה לחומר הכימי ממנו הוא עשוי. המחיקה מתבצעת לאחר פרק שמן מסוים ובעוצמה מסוימת.

השינוי בין EPROM ל – EPROM:
EPROM – באמצעות אור אולטרה סגול.
EPROM – באמצעיים חשמליים.

EPROM - מקבלים חלק וצריך לצרוב פנימה. תהליך הצריבה מתחיל ע"י רגל PROGRAM המקבל פולס (עליה/ירידה) - מכניסה את הרכיב לאפשרות צריבה.
ברגע שהוא חש שינוי הוא מחכה לכתובת. קוד הכתובת מגיע מרכיב חיצוני - ADD BUS לפי הקוד הוא מתביית על הכתובת ברכיב. אח"כ הוא מחכה למידע שמגיע מגורם חיצוני – DATE BUS חיצוני שכיוון שרימת הנתונים הוא מהחוץ לבפנים.
רכיב חיצוני – מערכת כלשהיא (יכול להיות מעבד) שיש לו מערכת של שני חוטי ADD ושני חוטי DATA שמחוברים ל – DATA ו – ADD של הזיכרון.
נותנים פעם פולס ברגל PROGRAM ואז יש שינוי, יש תהליך של כתיבה. הוא מחכה לקוד כתובת הרכיב מתביית על כתובת הרכיב האקטיבי (מעבד) מספק את האינפורמציה לכתיבה על הזיכרון. רק בתהליך זה
ה – DATA BUS הוא מבחוץ לבפנים.
כמות הצריבות אחת תלויה ביכולת של הגורם החיצוני ובתדר של הזיכרון.

<u>ERASE – תהליך המחיקה</u>:
מספקים פולס חשמלי לרגל ה – ERASE (בעליה או בירידה) ומכניסים רצף של אחדים (לממרות שאם שמים רצף של אפסים עדיין יש מחיקה).
נותנים פולס חשמלי שיכנס למחיקה. הגורם החיצוני נותן את כתובת המחיקה.
תפקיד הרכיב להינעל על הכתובת ואז באמצעות הרכיב האקטיבי מקבלים רצף של אחדים כדי לאפס ולמחוק את הנתונים בכתובות הרכיב ROM את מצב REAST.

EPROM – המחיקה היא בבת אחת על כל הרכיב. או הכל או כלום.
EPROM – מחיקה סלקטיבית – אתה יכול לבחור.
קודם כל מוחקים ואח"כ כותבים את המידע החדש.

שני מחזורי עבודה – מחיקה וכתיבה.
חייבים למחוק בשביל לכתוב מחדש.
EPROM ו – EPROM
נפח – עד 256KB.
זמן גישה – עד 50Msec.

יתרונות:

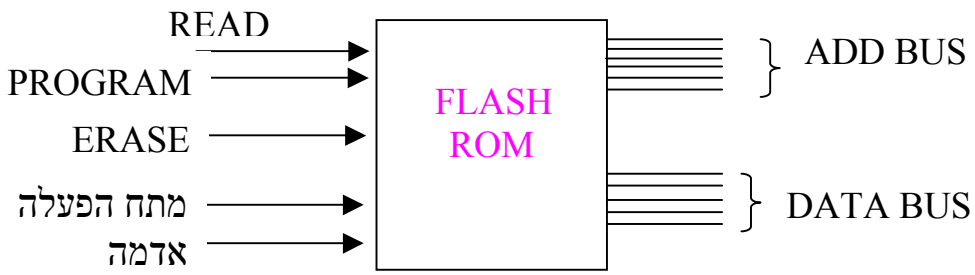
1. מחיקה / כתיבה סלקטיבית.
2. נפחים גדולים.
3. זמן גישה נמוך.

חסרונות:

1. יחסית יקר.
2. מתח המחיקה שונה ממתח הכתיבה.

כדי להכניס את רצף האחדים צריך לשנות את המתח. מתח שלא קיים במחשב. שלומר צריך מתח שמספקת יחידה חיצונית ולהשתמש במתח כדי למחוק.

FLASH ROM




המתחים לכתיבה ולקריאה הם מתחים שנמצאים במחשב.
כדי לבצע שינויים אפשר לעשות זאת תוך כשי עבודה במחשב.

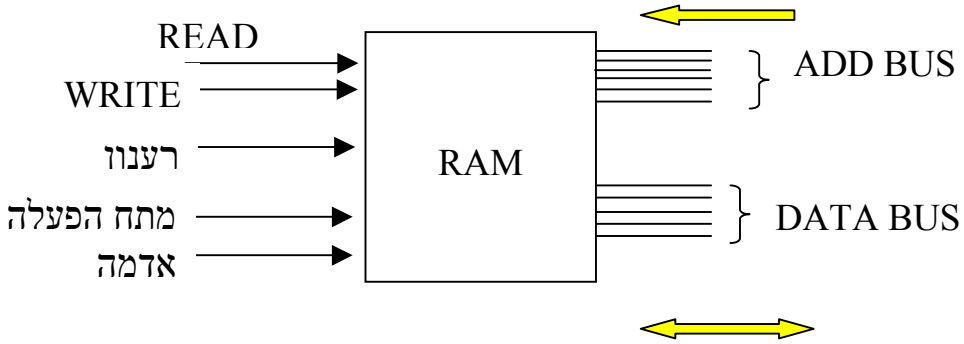
אין שינוי בזמן הגישה ובנפח, אין תלות במתח.

שימוש ב – ROM במחשב:
לאחסון תוכנות ה – BIOS – Basic Input Output System
BIOS- אוסף של שגרות (רוטינה) המאפשרות הפעלה של כתיבה או קריאה של הדיסק הקשיח, כתיבה לתצוגה, קליטת נתונים ממקלדת ועוד

EPROM - 286
EPROM - 386
486 – על כל דורותיו EPROM
פנטיום – FLASH ROM

רכיבי ה – BIOS כיום הם רכיבים שניתנים לשדרוג.

זיכרונות זמניים:
RAM – RANDOM ACCESS MEMORY
דינאמיים.
סטטיים.




– READ
כדי לקרוא צריך לתת פולס חשמלי. חוט הקריאה אמור להגיב בשינוי מצח. שלא רוצים לטפל ברכיב אז הוא צריך להיות על 0. שרוצים שהרכיב יפעל הוא צריך לקבל ADD כשיש שינוי עליה / ירידה וכשהשינוי מתייצב על המתח הוא מוכן לקבל את הקוד של ה – ADD.
כשמתבצע שינוי ברגל הקריאה הרכיב מוכן לקבל ADD הוא מתביית על הכתובת הנכונה ומוציא את המידע על גבי ה – DATE. התהליך מסתיים שהפולס מתייצב חזרה על ה – stand by.

ב – RAM אין תהליך ל מחיקה אלא של שיכתוב - אפשר לכתוב עליו.
תהליך כתיבה – פלוס ברגל WRITE.
קבלת קוד הכתובת לכתיבת הרכיב התבייתות על הכתובות והכנסת המידע.
ה – RAM תלוי במתח – אם ננתק את אספקת המתח המידע ברכיב אובד.
כשמחזירים את המתח הרכיב מקבל סט של אינפורמציה אקראית – זבל שאין להם משמעות.
DRAM – אחרי הפעלת המחשב הוא מידע לא רלוונטי עש שנשנה אותו לפי הצרכים הרלוונטים שלנו.

יתרונות:

1. נפח 32MB.
2. זמן גישה – 50-80nsec – ככל שזמן הרכיב נמוך יותר הרכיב יקר יותר.

חסרונות:

1. יש תלות במתח.
2. אחרי פרק שמן מסוים המידע מתפוגג – תכונה של החומר.
כתיבה מחודשת של האינפורמציה לוקחת הרבה זמן. חייבים לשמר מידע רלוונטי צריך לרענן אותו כל פרק זמן קבוע (לרענן – לכתוב מחדש) פרק הזמן הקבוע שמתייחס לרענון: 1/18sec.
אחרי כל פרק זמן זה המידע הולך.

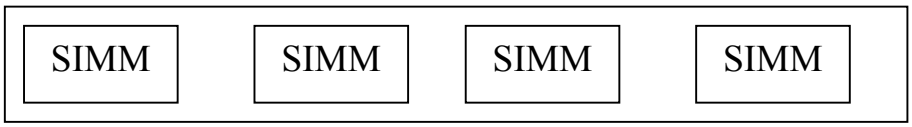
DRAM

תרגיל – זמן גישה 70nsec בפרק זמן של 1/18sec. כמה פעמים אפשר לכתוב עליו?

1/18=0.0555 = ~800,000 = ~750K

כמעט 1Mכתובות אפשר לכתוב.

SIMM – Single Inline Module Memory




כל רכיב מרעננים אותו בנפרד. בו זמנית מרעננים את כל הארבעה.

נוצר מצב ש – 1M לא מתרענן וניתן להפסיד את החומר הרשום בו. בד"כ שימוש בזיכרון זה הוא זמני.

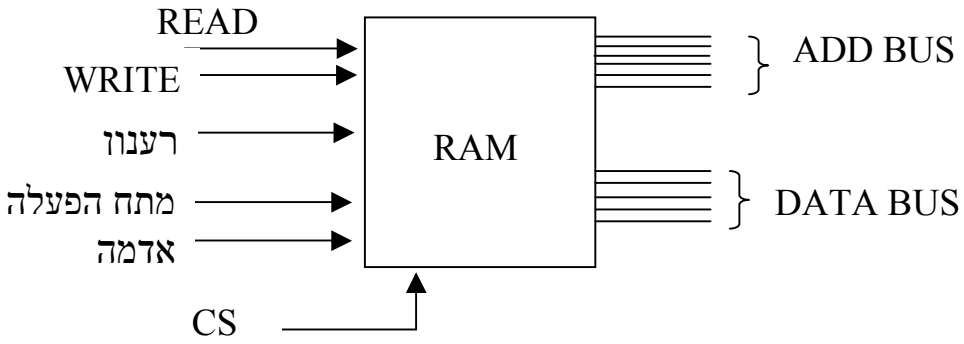
SIMM 30,72,168 קווי רגלים.

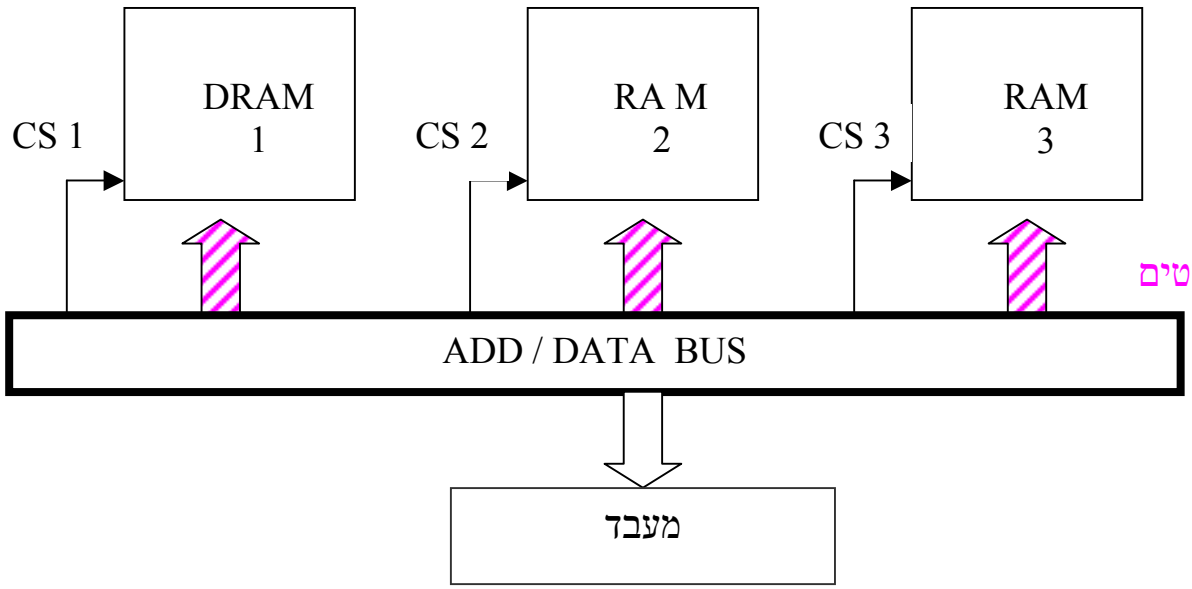
האם כמות הרגלים משפיעה על המהירות? לא.

רוחב כתובת בזיכרון הוא 8 ביט. אם שמים 16 ביט – אז הוא מעביר 2 כתובות למרות שהוא עדיין מעביר 8 ביט בכל כתובת. אין קשר בין כמות ה – DATA לבין מהירות הגישה. ADD BUS – קשור בנפח ולא מהירות.

DIMM – DUAL Inline Module Memory
יותר רכיבים על כרטיס.
4 רכיבים של 1M ב – 4M – SIMM
8 רכיבים של 1M ב – 8M – DIMM
במיקום שרק צד אחד יהיה רכיב עכשיו בשני הצדדים יש רכיבים. יותר רכיבים על הכרטיס ולכל כרטיס יש יותר זיכרון.
אין מספר רכיבים אבסלוטי בד"כ בין 4-6 רכיבים. ככל שיש יותר DATE BUB העברת הנתונים תהיה יותר מהירה ככל שיש יותר ADD BUS זה אומר שהנפח יותר גדול ואז הכרטיס עצמו יותר גדול.

CHIPS SELECT CS רגל



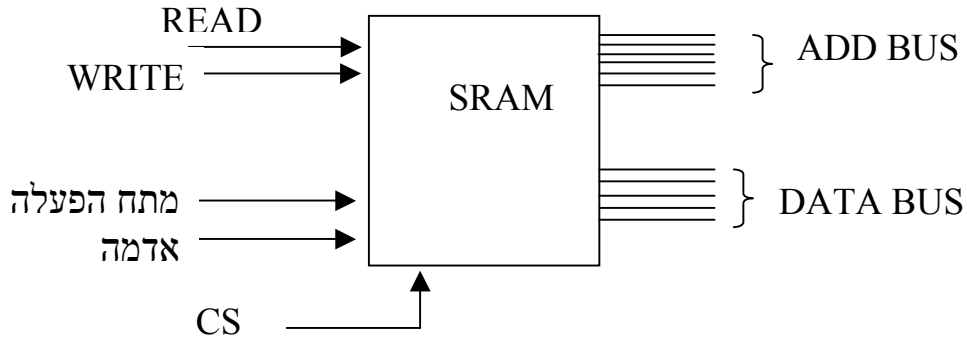
מקרא:
DATE BUS – 32Bit
ADD BUS – 22Bit

TOTAL MEMORY:
רכיב 1: 0-4
רכיב 2: 4-8
רכיב 3: 8-12

חלוקה לרכיבים מבחינת משאבי זיכרוז

אם צריך 10M הוא מאחד את המשאבים משלושת הרכיבים ומוציא מרכיב שלישי. אם צריך מעבר ל – 12M אז המעבד לא מתייחס לזה ומסתכל על זה בתור זבל.
המעבד "מחזיק מפה" באמצעות ה - CS אפשר לפנות לכל רכיב בנפרד. אי אפשר ששני רכיבים "ידברו" סימולטנית.
רכיב אקטיבי ADD BUS שלו הוא דו סטרי.
של ה – ROM רק בכיוון הקוד הבינארי.
ולכן הרכיבים בתרשים הם רכיבים פסיבים אין "שיחה" בינם לבין עצמם. רכיבי זיכרון הם פסיבים המעבד מדבר איתם – הוא היוזם

SRAM




נפח – 512K
זמן גישה – 520nsec
אין רענון – אין נדיפות כתלות בזמן.
עדיין יש לו תלות של מתח – מגרעה.
אין הבדילים ב – ADD/DATA BUS.
חוסר הרענון משפר את הרכיב כי לא צריך לדאוג לרענון:

1. רכיב חיצוני שמבצע את הפעולה.
2. חוסך זמן – לא צריך לכתוב שוב.

חסרון – יותר יקר. SRAM מיוצר מחומר יותר טוב ולכן יותר יקר אי אפשר להרשות שיהיה הרבה מזה במחשב ולכן זה מיועד למטרה מאד ספציפית.
במשפחת הזיכרונות הזמניים היה ה – DRAM זול יותרת צריך רענון אפשר להשתמש בו בכמויות נדיבות.
במחשב שלנו הוא משמש כזיכרון זמני (ראשי). גודל הזיכרון של 128-32M עולה בסביבות 200 ש"ח.

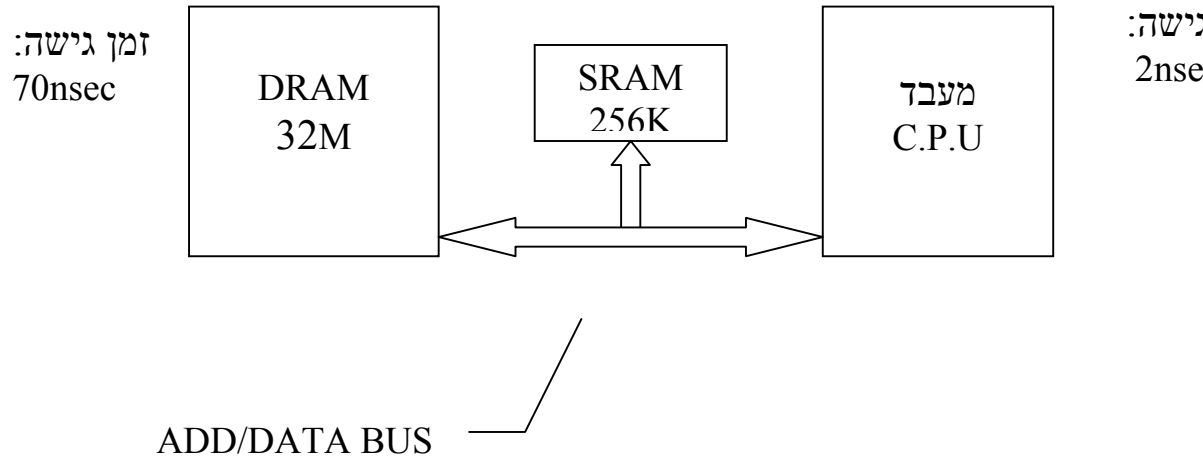
SRAM – כל 64K עולה 40 ש"ח כלומר 600 ש"ח ל – 1M.
זמן גישה מטרף – פי 3-4 יותר מהיר.
ה – SRAM משמש כזיכרון מטמון.

זיכרון מטמון – CASHE MEMORY

יש שני LEVEL של זיכרון מטמון.
LEVEL 1 - יושב בתוך המעבד .
LEVEL 2 - יושב חיצוני כלומר על לוח האם .

L1- פנימה.
L2 – החוצה.

L2 חיצוני:

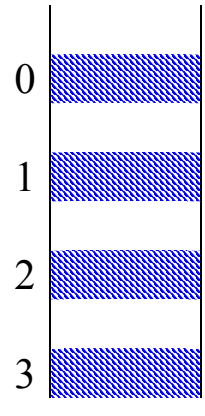


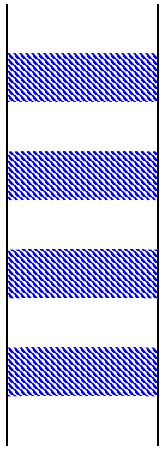

DRAM – הוא צוואר הבקבוק במערכת הזאת כי הוא יותר איטי. המעבד הרכיב הכי מהיר במחשב. ברגע שמשפרים את המהירות של אחד מהחלקים במחשב משפרים את סה"כ המהירות.
מהירות החוטים: 66MHZ / 100MHZ.
איך מקטינים את זמן הגישה ומשפרים את המהירות הכוללת?
מתקינים באמצע SRAM בגודל של 256K. בגלל שהוא קטן הוא בעצם סוג של מטמון. כי מכניסים לתוכו מידע שנחשב למידע איכותי – נתונים שנמצאים בכתובות ומשתמשים בהם בתדירות גבוהה.
<u>לדוגמא</u>:
נניח שקראנו אותה כתובת 10 פעמים בשניה האחרונה.
נתון איכותי = קריאה * 10 פעמים במשך 1 שניה.
צריך לנהל מערכת שתספור את כמות הפעמים שקראנו לנתון. המערכת נמצאת בתוך ה – BIOS המערכת בוחנת האם הנתון שנקרא מ – DRAM יותר מעשר פעמים בשניה. אם הכתובת נקראה במהלך השניה שחלפה יותר מעשר פעמים מעתיקים אותה (לא מעבירים) ל – SRAM.

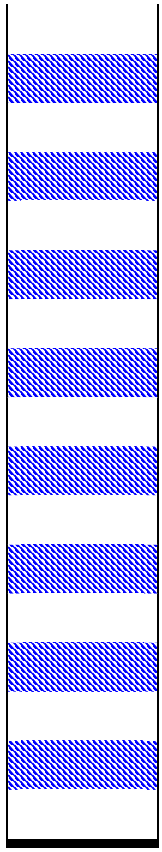
אחרי שמבצעים העתקה קוראים אותה ישירות מ – SRAM. וכך חוסכים זמן – בקשה לקריאה של נתון.
אם הנתון נמצא ב – SRAM הבקשה הלאה ל – DRAM מתבטלת. אם מגיעה תשובה שלילית מה – SRAM הבקשה ממשיכה ל – DRAM וממנו מוציאה את הנתון המבוקש.
כתובת שנמצאה איכותית – הועתקה מ – DRAM ל – SRAM נמצאת שם.
בכל מקרה יש את הנתון ולכן מעתיקים ולא מעבירים.
בפועל מתבצעת דחיסה – ה – SRAM מתמלא ואז הוא מתחיל לדחוס את הנתונים בסדר יורד.
פגיעה ב – CASHE : שנגשים ל – SRAM ומוצאים את הנתון המבוקש בעצם ביצענו פגיעה ב – CASHE. כשלא מצאנו את הנתון המבוקש אין פגיעה ב – CASHE. ככל שיש יותר פגיעות ב – CASHE זה אומר שהמערכת טובה יותר.
אחוזי פגיעה טובים: 30-40% טוב מאד. 15% - טוב.
CASHE – לא רכיב אלא תפיסה, רעיון. רעיון לשיפור זמני גישה בין המעבד לזיכרון הראשי במערכת ה – L2.

מימוש מטמון באמצעות מחסנית:
קיימים שני סוגי מחסניות:

1. מחסנית FIFO.
2. מחסנית LIFO.





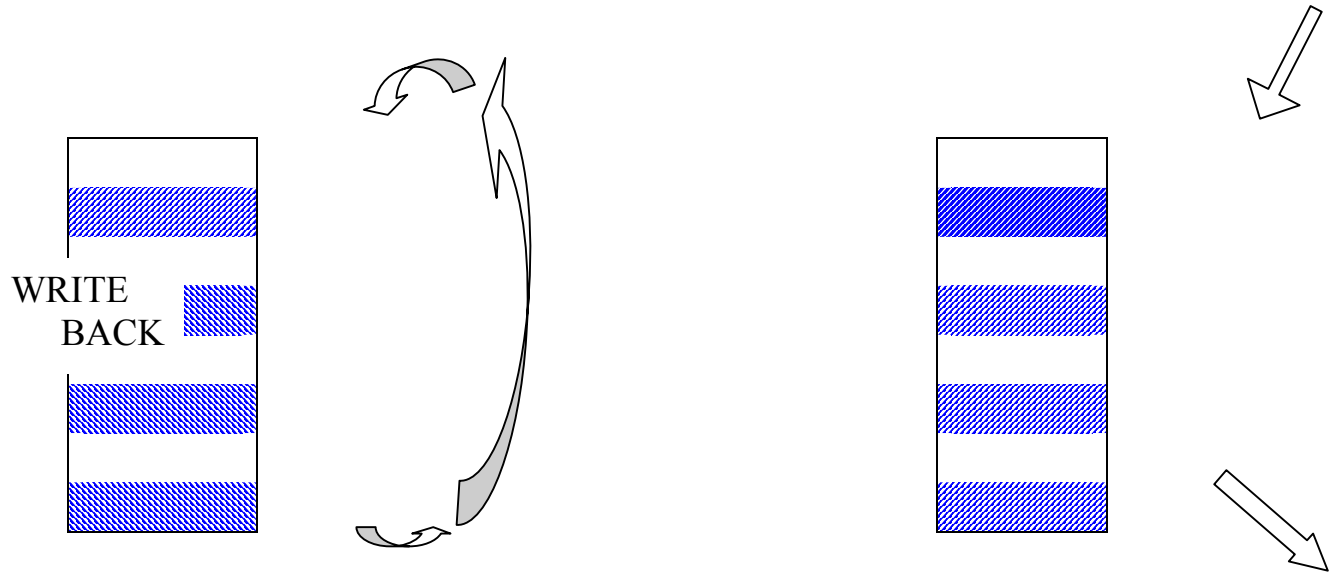




העליון
צריך

מחסנית FIFO – היא מחסנית שבה יש פתח הכניסה ופתח יציאה. הנתונים נכנסים זה אחר זה וממלאים את המחסנית שהנתון הראשון שנכנס הוא הנתון הראשון שיוצא כתוצאת מכניסת נתון חדש לגמרי.
מחסנית LIFO – היא מחסנית שפתח הכניסה הוא גם פתח היציאה. כאשר המחסנית מתמלאת נתון חדש יגיע יתיישב המקום הראשון שנכנס ודחיפת השלבים תעשה כלפי מעלה. בכיוון הפוך לכניסת הנתונים. הנתון האחרון שהגיע הוא הראשון שיפלט החוצה.
כמות שלבי המחסנית תלוי בנפח הזיכרון.

המחסניות מנוהלות ע"י תוכנה. התוכנה מצויה ב – BIOS ונקראת CASHE.
ניתן למלא מחסנית לאו דווקא בכתובות רציפות. האידיאלי הוא בכתובות רציפות כי אז יש חסכון בזמן חיפוש.

מימוש מטמון באמצעות מחסנית FIFO:




WRITE BACK - מימוש בעזרת FIFO.
SRAM משמש כמחסנית. כל עוד הנתון במחסנית הוא חשוף מבחינת פגיעה ב – CASHE.
ניהול המחסנית עובד בשיטה הבאה: יש מקום אחד ריק מילאנו את המחסנית ואז שבא נתון חדש לא נפלט הנתון האחרון אלא נפלט מחדש. ואז כולם זזים לתא הריק, שכולם זזו קדימה התא הריק מופיע למעלה הנתון הולך לתא הריק ואז מתפנה עוד תא – תגובת שרשרת.
המחסנית עובדת בשיטה שתא אחד ריק מזיזים אותו וביחד כל המחסנית זזה כלפי מטה.
אורך חיים של נתון: נתון עובר X מסוים של כתובות חזרה להתחלה ובפעם ה – X1+ הוא נפלט החוצה. ככל שנתון נמצא יותר פעמים ב – SRAM יש סיכוי יותר טוב לפגוע בו. יש אינטרס להשאיר אותו כה שיותר זמן ב – SRAM.
מחסנית משמרת נתונים X פעמים בתוך המחסנית. כל נתון חיי במחסנית X1+ פעמים ואח"כ נפלט. ואז מתפנה מקום שנתון חדש נכנס אליו (רענון) השימוש בצורה זו מתאים למערכות איטיות – שהמעבד שלו נמוך. בעיקר במעבדי 486 ומטה – כה מערכת איטית צריכה שישמרו עבורה את הנתונים יותר זמן פגיעה – המעבד מצא נתון במחסנית.
תופעת הריקון – ברגע הפגיעה בנתון כל הנתונים הנמצאים מתחתיו נופלים. הראשון נכתב חזרה וכל השאר נופלים ואז נכנסים החדשים למעלה שמרוקנים הכל זה בעצם אומר שמאבדים נתונים.

הפגיעה היא הרגע החשוב. כל הנתונים שנופלים באותו רגע בעצם לא חשובים באותו רגע ולכן אפשר לוותר עליהם. אם צריך אותם אז מביאים אותם שוב מה – DRAM.
לגודל המחסנית יש חשיבות – ברור שזה צריך להיות משהו באמצע לא גדול מידי ולא קטן מידי. אם המחסנית גדולה מידי כל פגיעה מורידה יותר נתונים מהמערכת זה שיש יותר שלבים לא אומר שיש יותר פגיעות. עדיף מחסנית יותר קטנה כי אז החלוקה משפרת את יחס הפגיעה. קניה של עוד SRAM לאוו דווקא משפרת את המהירות. גודל המחסנית לא משפר או לא קובע את יחס הפגיעות.

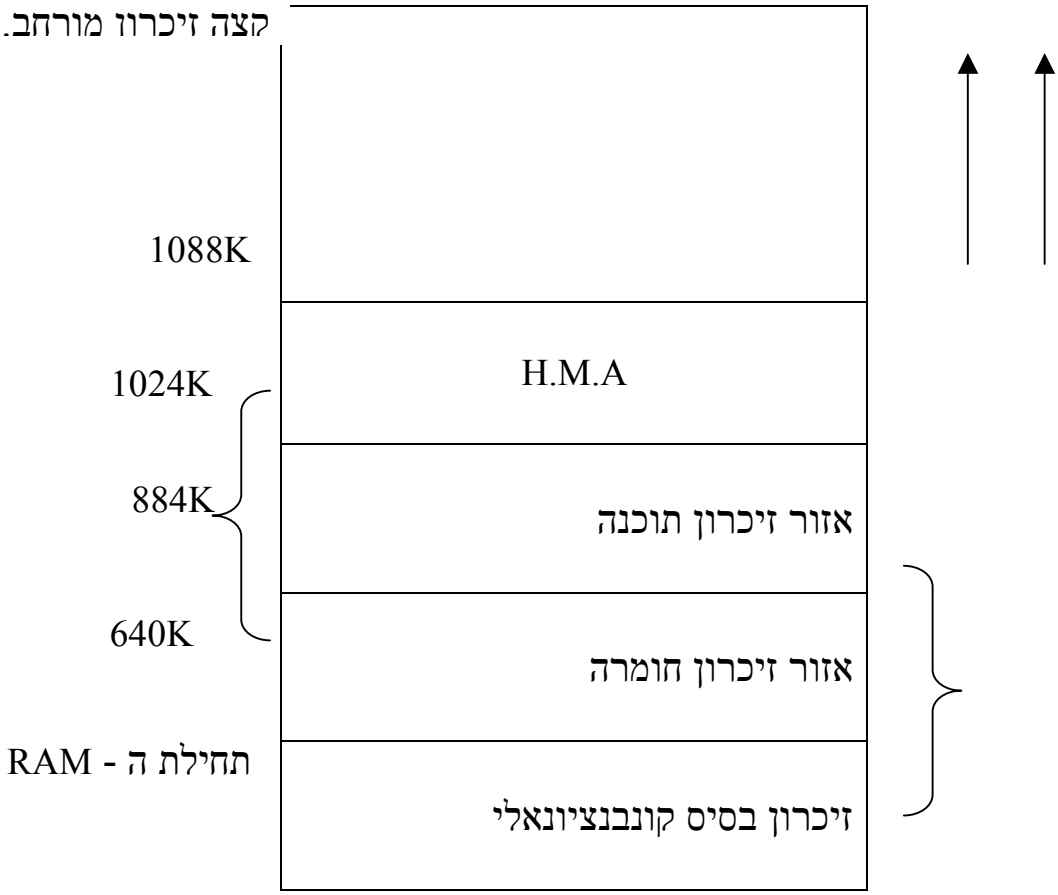
מימוש PIPELINE –
אינה התומכת בכתובת חזרה. כלומר שימור הנתון בתוך המחסנית הוא מחזור חיים בודד ואחד. השיטה היא שיטה שלא תומכת בשמירת נתונים אלא ברענון נתוני המחסנית. ככל שמרעננים יותר את המחסנית משפרים יותר את אחוזי הפגיעה.
מתאים למערכות מהירות – כשהמערכת מהירה יא מנסה לפגוע בנתונים חדשים. מחשבי פנטיום ומעלה.

SDRAM, EDO / CACHE
SDRAM – רכיב DRAM רגיל עם יתרון בולט – מהירות הגישה אליו גבוה יותר בערך ב – 40-50nsec.
EDO – EXTENDED DATA OUT
תכונות של DRAM עם שיפור של שני דברים:

1. אפשר לקרוא ולכתוב פעם אחת בבלוקים. במחזור קריאה אחד מוציא בלוק נתונים – דבר המשפר את המהירות.
   מקטין זמני גישה – מעביר יותר נתונים.
2. READ WHILE WRITE – רכיב שמסוגל לקרוא נתון תוך כדי ביצוע כתיבה לנתון אחד.

מערכת מחשב שמכילה CACHE MEMORY במקום DRAM מוסיפים SDRAM EDO לא משפרים את מהירות הגישה לזיכרון.
וכל זאת משום שתפקיד ה – CASHE להגביר את המהירות של ה – DRAM. ה – CASHE לא עובד בבלוקים ולכן הוא לא יעיל מול ה – EDO
אם הוא עובד מול ה – CASHE הוא עובד בבודדים ולא בבלוקים. מערכת שיש בה את שני הרכיבים פוסלת אחד את השני כלומר האחד משפר והשני לא עושה כלום ולכן זה לא גורע ולא מוסיף.

מיפוי זיכרון במחשבי DOS / WIN95.



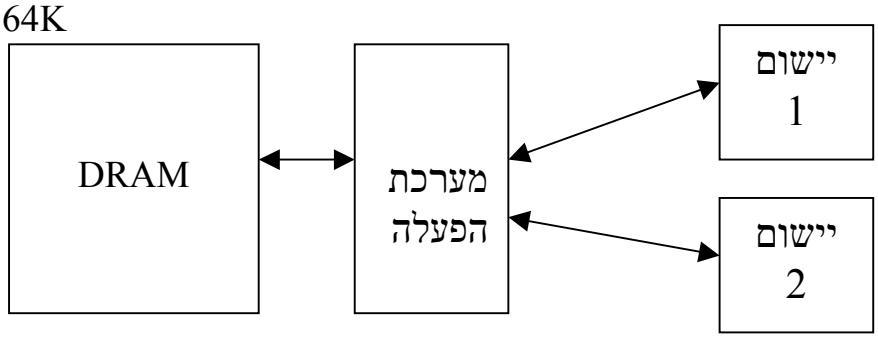

<u>מערכת הפעלה</u>:

הפעלה בסיסית + מתקדמת:
מערכת הפעלה זו תוכנה שמנהלת עצמאית את רכיבי החומרה.
BIOS - מספר פקודות שאחריות על הפעלה ראשונית של רכיבי המחשב.
יש פה מין סתירה – קודם כל ה – BIOS – הדלקנו את המחשב ואז ה – BIOS עולה. מערכת ההפעלה צרובה על הדיסק הקשיח ואליו צריך להגיע כדי לקרוא אותה. תפקיד מערכת ההפעלה – לעשות אותם דברים כמו ה – BIOS אבל ברמה יותר גבוהה.
ה – BIOS מתייחס לכל ה – RAM, DRAM.

1. רוטינת כתיבה קריאה ה – BIOS יודע לעשות זאת לבד.
2. זיהוי תקינות וכמות – בדיקת וספירת הזיכרון הפנימי. אומד את כמות הזיכרון ובודק אותו לתקינות.

בדיקת הזיכרון – מכניס נתון וקורא אותה ביציאה אם לא השתנה כל מה שנשאר זה להכניס את כל הכתובות במחשב.

מערכת הפעלה-
לא בודקת תקינות אבל יש לה את רוטינת הקריאה הכתיבה.




מערכת ההפעלה מקצה ליישומים את כתובות הזיכרון.
התוכנה שולחת בקשה למערכת ההפעלה וזו פונה לזיכרון, הזיכרון מקצה מספר כתובות כמבוקש ומערכת ההפעלה מעבירה את הכתובות ליישומים.
אסור ששני יישומים ישתפו את אותו שטח בזיכרון.
מערכת ההפעלה מנהלת את הזיכרון.
מערכת ההפעלה מסתכלת על החומרה הפיסית.

<u>הסבר המיפוי</u>:

קצה הזיכרון המורחב – לפי כמות הזיכרון הרצוי או המצוי. כמה שיותר זיכרון יכולת העיבוד עולה.
זיכרון בסיסי – קונבנציונאלי – רגיל סטנדרטי.
אזור הזיכרון הבסיסי משמש להרצת קבצים הקרואים קבצי הפעלה.
קבצים אלו הם בעלי הסיומת הבאות: EXE, BAT, COM.
כל קובץ EXE הוא למעשה קוד מקור של תוכנה מסוימת שעבר קומפילציה.
נפח האחסון – המקום שתופס הזיכרון בדיסק הקשיח (אחרי קומפילציה).
נפח עבודה (ריצה) – נפח הזיכרון שתופס היישום ב – DRAM כשהוא במצב הרצה.
נפח הרצה של יישום תמיד יותר קטן מנפח האחסון שלו.
ה – EXE יותר גדול. הוא מכיל אלמנטים שגורמים לקובץ לרוץ ולכן הקובץ הסופי הוא יותר גדול ב – 30-40%.
נפח הריצה – מקום בזיכרון DRAM שהקובץ תופס שהוא פעיל. אם מאחסנים 10 נתונים בתוך תאי הזיכרון. נפח התאים – 10 תאים או 10 בתים.

אף תוכנית שגודלה מבחינת נפח הריצה גדול מ – 640K בתיאוריה לא תרוץ. יש תוכנית שנפח הריצה הדרוש הוא 641K כלומר היא לא תרוץ אלא אם נפצל אותה אבל בפיצול יש בעיה הרבה זמן מתבזבז בקפיצות אבל לפעמים אין בריבה. בונים כמה קבצים שהסיומות שלהם EXE ומחלקים לפי פונקציות עבודה. כך עושים בתוכניות "כבדות".

למה לא מגדלים את נפח הזיכרון הבסיס מיותר מ – 640K?

מחשבי ה – XT הראשונים שווקו עם 128K זיכרון שהספיק לכל מה שהיה צריך אז, כי תוכנית עם 100 שורות קוד נחשבה אז לגדולה.

במשך השנים התחילו להוסיף עוד 128 ואז עוד 128 באיזה שהוא שלב קבעו מגבלה של 640K זיכרון מתוך הנחה ש – 640K יספיק למה שהיה אז בשוק ולמה שיהיה בעתיד.

אך זה לא היה נכון לגבי העתיד.

ה – 640K הספיקו למשך שנתיים שבהם נוצר הסטנדרט לגודל הזיכרון.

כל החברות שכתבו תוכניות כיוונו את הזיכרונות של התוכניות שלהם מ – 640-0K. שנתיים כתבו בצורה כזו ואם לא היה מספיק זיכרון היו מפצלים את התוכנית לשני EXE אבל כל הקבצים עבדו במרחב הכתובות 640-0K.

BATCH – קובץ שכתוב בשיטת DOS שמריץ פקודות DOS בצורה עצמאית קובץ רימה אוטומטית הזיכרון הבסיסי מוגבל כתוצאה משגיאה של איש תוכנה מהעבר. ולמעשה הוא קלקל כי הייתה תאימות כלפי מטה ואז לקחו גם את השגיאות ואת מחסום ה – 640K.

384K – לא טובים לשימושי EXE ו – COM אז כולם כתבו ב – 640-0K.

באזור הזיכרון הבסיסי הדבר הכי חשוב הוא שצריך לקחת בחשבון שהמקום מוגבל.

בתוך משתמש מתקדם רצוי שהמקום יהי מינימום תפוס פנוי ככל האפשר הוא לא יכול להיות ריק.

יושבים עליו: גרעין מערכת ההפעלה – KERNEL.

קבצי הגרעין של DOS: IO.SYS

MSDOS.SYS

COMMAND.COM

הגרעין הטהור מורכב משניים העליונים גם ב – DOS וגם ב – WIN.

תפקידן נותנים את הפונקציות הבסיסיות:

IO – INPUT OUT: קובץ שמכיל רוטינת של קלט/פלט הם יותר טובות ומתקדמות מה – BIOS.

MSDOS: לעשות את כל השאר: מחזיקות שעון, דואגות להקצאות זיכרון.

COMMAND.COM: קובץ תרגום דו סטרי.

פקודה חיצונית שכדי להריץ אותה צריך להריץ קובץ. דוגמא לפקודה חיצונית: FORMAT.
COMMAND.COM – תפקידו לקשר בין המשתמש לבין המחשב – מפענח פקודות.
שמפעילים את המחשב ומאחסנים את KERNEL הוא תופס בזיכרון הבסיסי בערך 65K שזה 11% מהזיכרון הבסיסי הכולל. זה תמיד ישב שם כל עוד זה פעיל. פחות מקום להרצת קבצי ה – EXE .

שטח שהנפח שלו הוא 384K:
הוא משתף בתוכו שני אזורים – אזור זיכרון החומרה נקרא SHADO (אזור הצללים) ואזור זיכרון תוכנה UPPER MEMORY AREA - U.M.A.
אזור זיכרון תוכנה – במקורו מיועד לאחסן DRIVERS. זו רוטינה תוכנה שתפקידה לקשר בין מערכת ההפעלה לבין חומרה ספציפית דוגמאות – כרטיס מסך, כרטיס קול, מקלדת, מודם, עכבר.
DREIVER – למעשה קובץ ספציפי לחומרה ספציפית.
כל כרטיס והדריבר שלו אין דריבר אוניברסלי. ל – WIN יש דריברים שיהיו עדכניים לרגע הוצאת התוכנה אם יש חומרה מעודכנת צריך דרייבר חדש ומעודכן.
כאשר נטען דרייבר עבור חומרה ייעודית הוא תופס מקום בזיכרון הבסיסי. ולכן בנו מקום שמיועד לדרייברים.
אזור זיכרון תוכנה – נפחו ~80K180-K.
זה משתנה בכמה התניות:

1. ארכיטקטורת בנית החומרה – כל יצרן בונה קצת אחרת.
2. סוג מערכת ההפעלה – כל מערכת הפעלה בנויה לפי הצרכים שלה.

ייחוד ה – U.M.A.:
אם לא נפתח את האזור לא נוכל להשתמש בו. כלומר אזור חבוי. צריך להגדיר אותו ב – CONFIG.SYS.

מטרות (התשובות למטרות יופיעו שנלמד על ה - CONFIG.SYS):

1. לפנות חלק מהמקום שתופס הגרעין בתוך אזור הזיכרון הבסיסי.
2. כיצד להעביר דרייברים מאזור הזיכרון הבסיסי לאזור התוכנה.
3. פתיחת האפשרות להשתמש ב – U.M.A.

גרעין מערכת ההפעלה נטען לאזור הבסיסי כברירת מחדל.

דרייבר – רוטינת תוכנה שמקשרת בין התקנים החיצוניים (חומרה) לבין מערכת ההפעלה.

אין דרייברים כוללים לכל התקנים במערכת ההפעלה היות והם מתחדשים כל הזמן.

פתרונות:

1. משווקים עם התוכנה / חומרה דיסקט דרייבר עם הוראות הפעלה.
2. דרייבר ישן יעבוד עם כרטיס חדש מאותו הסוג אבל הוא לא יכיר את התכונות החדשות שנוספו לו. ולכן צריך ליצור עידכונים למערכת ההפעלה (באגים). כל רבעון החברה שהפיקה את מערכת ההפעלה יוצרת גרסת עדכון למערכת ההפעלה (מעדכנת את הדרייברים, כרטיסי מסך, C.D. וכו') מעדכנים גם את הבאגים. באג לא תמיד עוצר את התוכנית ובגרסה הבאה מתקנים את הבאגים.

תוכנה צריכה להיות יעילה שזה אומר קצרה ככל האפשר.

אזור התוכנה:

אזור זיכרון תוכנה – נפחו ~80K180-K.

זה משתנה בכמה התניות:

1. ארכיטקטורת בנית החומרה – כל יצרן בונה קצת אחרת.
2. סוג מערכת ההפעלה – כל מערכת הפעלה בנויה לפי הצרכים שלה.

אזור החומרה:

אזור הצללים ~384-0K.

U.M.B. – UPPER MAMORY BLOUCKES.

שטח שבעם מחלוק לבלוקים שטח ה - U.M.B מחולק ל – 6 בלוקים הממופים בהתאם לחלוקת האזורים (אזור תוכנה / אזור חומרה).

יכול להיות מצב שאזור החומרה בלע את ההגדרה של אזור התוכנה. קרי – אין אזור תוכנה. הבעיה היא שהזיכרון הבסיסי מלא בדרייברים ואין אפשרות להריץ אותם באזור התוכנה.

אזור התוכנה – אזור מענין מאחר ואם לא נעשה פעילות קונפיגורציה האזור סגור. צריך לתת פקודות מיוחדות בקובץ CONFIG.SYS כדי לקבל את האפשרות להשתמש באזור התוכנה אן לא אזור החומרה משתלט על כל השטח והוא מקבל את הטולראנס העליון.

H.M.A. – HIGH MEMORY AREA:

נפחו ~1088K -1024K.

אזור הזיכרון הגבוה – להבדיל מהעליון.

הגדרה – מיועד לאחסן חלק מגרעין מערכת ההפעלה שמגיע מהזיכרון הבסיסי.

OFFRET – אנחנו רוצים להעביר נתון מכתובת 100H (נניח) אבל רוצים להעביר ראותו דרך כתובת 0H. ולכן שותלים פקודה שמקפיצה אותו בכמות מסוימת של כתובות. הנתון חייב להגיע לכתובת 0 כברירת מחדל אבל הוא צריך להיות בכתובת 100H ולכן אומרים שה – OFFSRT הוא 100.

OFFSET – מספר פקודות. מסיט בערך מסוים קדימה. ההקפצה יא בכתובות קבועות.
נותנים פקודה ב - CONFIG.SYSשמתורגמת לפקודת מכונה שיוצרת את כל הסיפור של ה – OFFSET.
האם ה – OFFSET קבוע? לא! כי…

1. אזור התוכנה גמיש – מחשב לפי הגודל הדרוש.
2. כמות הדרייברים שונה ומשתנה.
3. גודל הדרייברים משתנה.

אזור ה – SHADOW משמש ל:

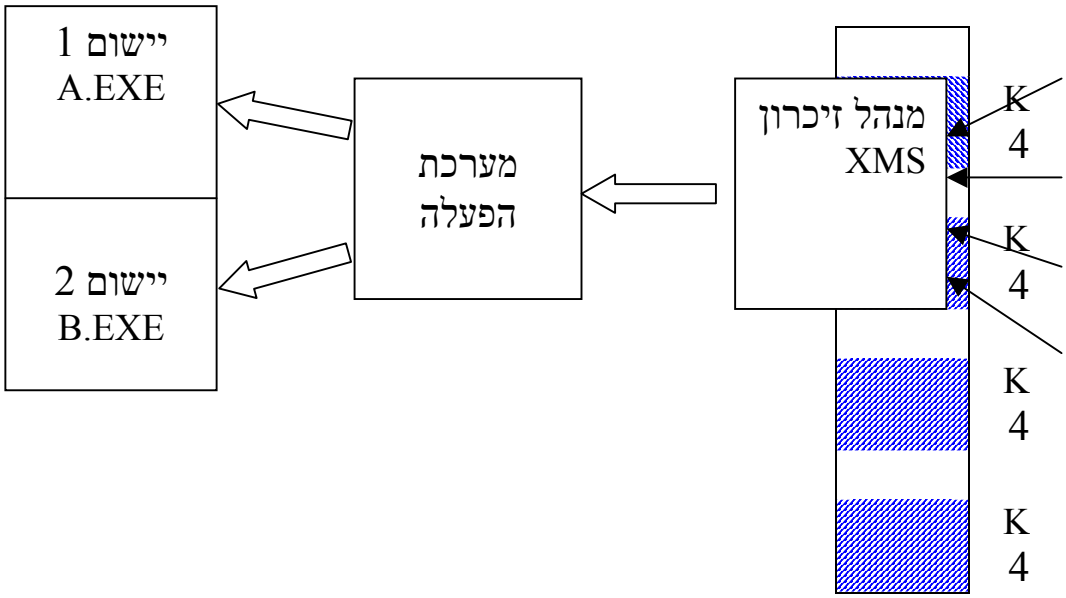
1. רכיבי החומרה שנמצאים על לוח האם זקוקים מעת לעת לזיכרון זמני זמין. ה – DRAM באזור החומרה, הם זקוקים לעד 64K בכללם.
2. אם ה – BIOS יושב על רכיב אז הפעלתם של רוטינות ה – BIOS יותר מהירה ולכן מעתיקים את ה – BIOS לאזור החומרה. לא מעבירים אלא מעתיקים כי באתחול הבא נצטרך את ה – BIOS שוב.

חלוקת השטח:
גודל ה – BIOS הסטנדרטי הוא ~128K -100K. כאשר אזור התוכנה סגור כל שטח פנוי שנשאר מבוזבז.
ההצללה של ה – BIOS היא לא היחידה ניתן להצליל BIOS שיושב בתוך המערכת. כרטיס מסך – מערכת שלמה שמכילה מעבד, BIOS וזיכרון זמני.
זיכרון מורחב:
אחסון של קבצים של נתונים. בקיצור הכל חוץ מ – COM,BAT,EXE.
אפשר לאחסן את הקבצים הללו באזור ההרחבה אך אי אפשר להריץ אותם שם.
שמושכים אותם לאזור הזיכרון הבסיסי הם מקבלים תוכניות הרצה וזה מאפשר זמן גישה מהיר יותר.
זמן גישה לקבצים שטוענים בתוך האזור המורחב קטן בהרבה.
לכל תוכנה – יש שאיפה לטעון כמה שיותר לאזור המורחב כדי שהאפשרות להפעיל אותם תהיה מהירה יותר. אם נגמר המקום עוברים לדיסק הקשיח או לאזור העליון. יוצרים קשרים בין חלקי קובץ על מנת לא לאבד את רצף העבודה.

איזה גורם – תוכנה או חומרה אחראי לקביעת ה – OFFSET המקפיץ את הדרייברים?
אי אפשר לקבוע את זה ע"י החומרה. חומרה זה אלקטרוניקה, חשמל, 1/0 לא מסוגל לבצע את ה – OFFSET. לעומתו תוכנה כן מסוגלת ע"י מערכת הפעלה שמנהלת את אזור הזיכרון הבסיסי שלשם נטענים הדרייברים כברירת מחדל.
ולכן מי שמנהלת הוא זה שאחראי על ה – OFFSET.

מדוע מנהל גרעין מערכת ההפעלה זיכרון בסיסי עד 640K?
בנו את מערכת ההפעלה ככה מפני שלא יודעים כמה זיכרון יש לכל מחשב ומהי תצורת הזיכרון. אבל בוודאות יודעים שיש 640K. ולכן מנהלים את האזור שבטוח שיש בו זיכרון ובהתאם לתצורת הזיכרון אפשר לנהל איתם גורמים חיצוניים.
640K אחראי להרצת קבצים EXE ו – COM שהם קבצי הריצה של כל תוכנה, זה אומנם עובד לאט אבל זה רץ. במידה ואפשר להרחיב , תרחיב אם לא אז במרחב הבסיסי הוא יכול להריץ את התוכניות.

ניהול זיכרון לפי שיטת XMS:




היישומים מתוקשרים למערכת ההפעלה שהיא אחראית לזיכרון הבסיסי ששם היישומים רצים.
התקשורת של היישומים היא אך ורק עם מערכת ההפעלה וכל הדרישות נענות ע"י מערכת ההפעלה.
במידה והיישום צריך מעבר ל – 640K זיכרון. מערכת ההפעלה מקבלת דרישה לטעון יותר מ – 640K ואז היא מעבירה אותו לאזור הזיכרון המורחב.
מנהל זיכרון XMS –
XMS – שיטה לניהול זיכרון שלוקחת את כל מרחב הזיכרון מקצה ה – 1M הראשון ועד קצה גבול היכולת של המחשב, היא מחלקת את כל המרחב לבלוקים של 4K ואותם היא מקצה למערכת ההפעלה.

היישום פונה למערכת ההפעלה ומבקש 1.5M.מערכת ההפעלה מפנה את הדרישה למנהל הזיכרון XMS. מנהל הזיכרון בודק אם היכן יש לו מקום פנוי והוא אוסף אותם עד שהוא מגיע ל - 1.5M בבלוקים של 4K. (לא חייב להיות רציף – למרות שזה יותר מהיר ונוח שזה רציף). עם הכתובת שהיא מוצאת בתור פנוי היא מפנה חזרה למערכת ההפעלה ומערכת ההפעלה יודעת להקצות ולהעביר חזרה ליישום המבוקש.

למה 4K? הליכה בין יעיל לנוח. ככל שהבלוק קטן יותר יותר יעיל לנהל אותו- למרות שהוא יותר איטי. אין הקצאות אזורים לחינם. השיטה יותר איטית כי יש יותר בלוקים לנהל. אם נניח היה 1K זה היה מאד איטי ואם זה היה 8K היינו מבזבזים יותר מקום אם היינו מבקשים 3K ולכן כל שאר הבלוק היה הולך לאיבוד.

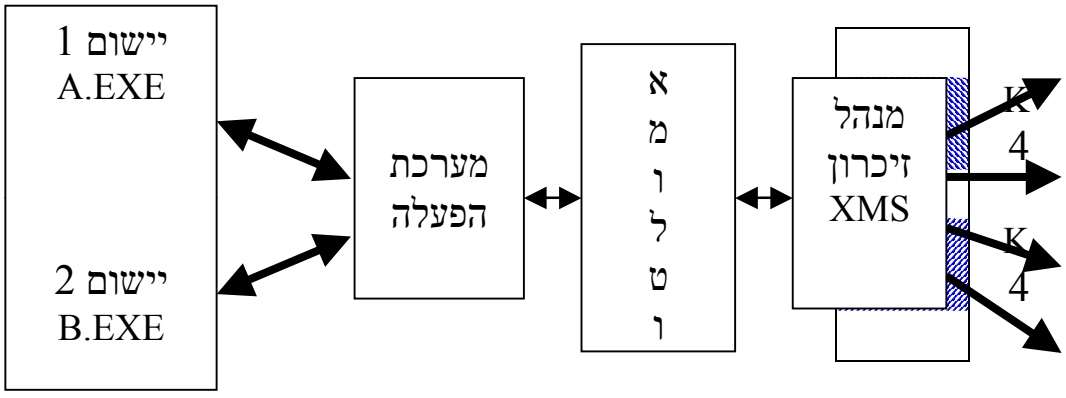
ניהול זיכרון XMS מבוצע ע"י קובץ HIMEM.SYS. זה מנהל הזיכרון העליון של מיקרוסופט. עדיף להשתמש במנהל הזיכרון של מיקרוסופט כי היא מבטיחה תאימות מלאה. לא יהיו, למשל, שני ישומיים שירוצו על אותה כתובת.
חוץ מזה שמיקרוסופט נותנת את מנהלת הזיכרון כשאתה קונה את מערכת ההפעלה. מנהל הזיכרון והקובץ HIMEM.SYS תופסים 4-2K בזיכרון והם נמצאים באזור הזיכרון הבסיסי ומשם הם רצים.
שטוענים את HIMEM.SYS אין אזור תוכנה אולם יש אזור H.M.A. שמשמש להכיל חלק מגרעין מערכת ההפעלה. הוא פנוי, אך ורק לפקודה שאומרת להטעין חלק מגרעין מערכת ההפעלה באזור הזה. ולכן שמבקשים אינדיקציה על מצב הזיכרון הוא מדווח כמקום תפוס. קיים אבל לא בשבילך.
קיים מנהל זיכרון נוסף EMS – השיטה הישנה:
מנהל הזיכרון הראשון הוא ניהל את הזיכרון לפי בלוקים של 64K. ולכן הקצאות הזיכרון הן בזבזניות יותר.
היו לה מגבלה עיקרית: אפשר היה להרחיב זיכרון מ – 640K עד 16M וזה הכל. שיטה שהייתה דומיננטית עד אמצע שנות השמונים.

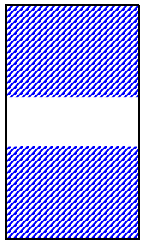
ניהול הזיכרון:
איך עורכים את הקובץ CONFIG.SYS – ע"י כל עורך.
השורה שטוענת את HIMEM.SYS לתוך הזיכרון:
DEVICE = C:\ HIMEM.SYS.
DEVICE – התקן / הטען.
C:\ - לרוב ב – C בספריית הקובץ הנתיב של קובץ.
טוענת לזיכרון הבסיסי את מנהל הזיכרון HIMEM.SYS.
כל שינוי או עדכון בקובץ CONFIG.SYS מחייב ביצוע REAST למחשב – כדי שמערכת ההפעלה תיקח עוד פעם את CONFIG.SYS ותפעל עפ"י מה שרשום שם.
אין במנהל תוכנה ל- EMS איתי כי אין זיכרון בשיטה זו. ז"א אין שורה ב – CONFIG.SYS שטוענת רק מנהל זיכרון EMS לא טוענת XMS וזה עובד.
כלומר אי אפשר לנהל את ה – DRAM רק ב – XMS לא עובד כי אין כרטיסי הרחבה כאלו.
קובץ EXE בעל יותר מ – 640K מחולק. מעלים אותו לזיכרון המורחב ומורידים נתונים – שהופכים אח"כ לפקודות – לאזור הבסיסי ויוצרים קשרים בין חלקי הנתונים השונים.

תצורת המערכת – CONFIG.SYS:
קובץ בר תכנות. כתוב בשורות שדואג לעשות את מיפוי הזיכרון. חלקו מפקח על החומרה – הדרייברים. גם המערכות החדשות WIN95/98 מחזיקות קובץ כזה גם אם לא תמיד בפורמט קבוע אלא למטרות מסוימות. DOS ורשתות משתמשות בו הרבה.
חמשת החלקים שמגדירים את הקובץ:

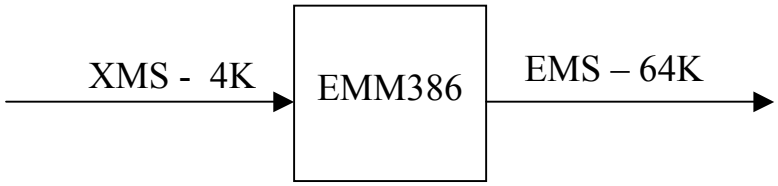
1. ניהול הזיכרון.
2. תצורת טעינת DOS.
3. הגדרות המערכת.
4. ניהול רכיבי חומרה.
5. קביעת סביבת העבודה.

<u>ניהול הזיכרון</u>:
טעינת מנהלי הזיכרון. טוענים את קבצי הגרעין (פעולת האתחול). שלושת הקבצים נטענים למערכת ותופסים 65K. זיכרון מוכר למערכת 640K מתוכו תפוס 11%. שקונים מחשב עם הרבה זיכרון איך אפשר להשתמש בשאר הזיכרון? אי אפשר לנצל את יתרת הזיכרון שנמצאת מעל האזור הזה ולכן יש שיטות שונות לטיפול בבעיה.
שיטת XMS – שיטה לניהול זיכרון עליון.
לומר חייב לטעון לפי שיטה זו.
יכול להיות מצב של יישומים שיודעים לקרוא זיכרון לפי EMS והם מקבלים הקצאות זיכרון דרך XMS הם לא יודעים שהם מקבלים זיכרון. תוכנה שמקבלת לפי הקצאת זיכרון 4K במקום 64K תיתן הודעה שהיא לא קיבלה זיכרון. דוגמא לתחנה שעדיין עובדת לפי שיטת EMS – חשבשבת.
כדי שזה יעבוד מהשיטה הישנה לשיטה החדשה נותנים: 64K=___*4K
כלומר צריך לתת 64K בכל דרך אפשרית. ולכן יש מנהל זיכרון EMS הוא רק אמולטור - מדמה. והוא נקרא EMS386.EXE.




K
4
K
4

נניח שהיישום זקוק ל – 64K, הבקשה מופנת למערכת ההפעלה שפומה לאמו לטור ומבקשת EMS 64K, לאמולטור אין גישה ישירה לזיכרון. הוא נעזר בשירותים של מנהל הזיכרון XMS ולכן הוא פונה בבקשה ל – 64K, מנהל הזיכרון אוסף בלוקים של 4K בסה"כ 46 בלוקים של 4K שיהוו 64K, רצוי שזה יהיה רציף כי אז הוא נותן כתובת התחלה וכתובת סיום בהתאמה, אחרי שהוא אוסף את הכתובות הוא מחזיר דרך כל הדרך עש ליישום המבוקש. אם צריך K 128 הוא נותן שני בלוקים של 64K כל אחד. מדמה בלוק של 64K.
למערכת ההפעלה יש תפקיד מכריע והיא חייבת לדעת כיצד לטפל במגבלות אלו.
היישום רואה רק את מערכת ההפעלה ומבחינתו לא איכפת לו איך מערכת ההפעלה תשיג את הקצאת הזיכרון המבוקשת על ידו.
לכן מערת ההפעלה טוענת EMS386 לאזור הזיכרון הבסיסי ומשם הוא מבצע את הניהול ואת האמולציה.

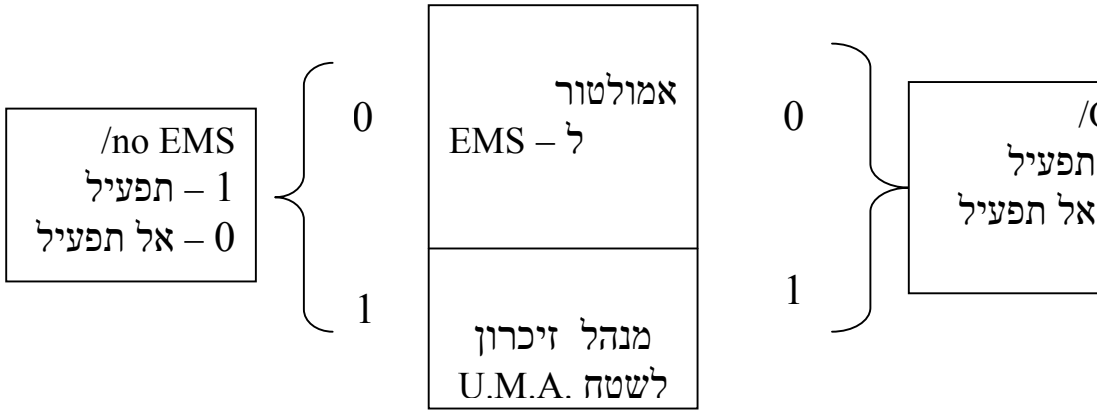



חייב לתת את שתי השורות כי השורה השניה היא רק מדמה ובפועל השורה הראשונה היא שנותנת את הזיכרון.
DEVICE=C:\HIMEM.SYS
DEVICE=C:\EMM386.EXE / _____

לאחר ה - / צריך לתת לו אחת מארבעת הוריאציות הבאות:
/no EMS
/AUTO
/OFF
/ON

/ON - הפעל, טען. לא משתמשים בזה כי זה ברירת המחדל שלו.
/OFF – תטען אותו אבל אל תפעיל אותו, למעשה זה מפסק בקרה, מתי שאני ארצה אני אפעיל אותו.
/no EMS – תעלה ותטען את עצמך אבל אל תוציא EMS דומה ל - /OFF. כל הזיכרון מנוהל ע"י XMS. תעלה רק את מנהל הזיכרון לשטח U.M.A. ואל תעשה אמולציה.

EMM386.EXE הוא מודם של תוכנה שמכיל שני חלקים.




לנהל זה אומר – לפתוח אותו, לדרוש אותו ונהל אותו – לדעת להקצות ממנו שטחים.
בלי שמפעילים no EMS/ אי אפשר להשתמש באזור התוכנה. אם רוצים לסלק דריבריים מאזור הזיכרון הבסיסי צריך לטעון no EMS/ לזיכרון.
ששמים NO/ יישומים XMS מדווחים על זיכרון כי הם מקבלים בלוקים של K 64 ואז הם לא יכולים לרוץ גם ב - NO/ אין U.M.A.
AUTO/ – אני אקצה EMS רק עפ"י בקשה ורק עפ"י הצורך – כמה שצריך. לפי בקשות ספציפיות של יישומים
נותן אפשרות להפעיל גם EMS וגם XMS (עבור יישומים שונים). משתמשים ב - AUTO/ אבל אין U.M.A. מתוך הנחה שישומי EMS הם ללא דרייברים. שזה פועל ב - AUTO/ האמולטור תמיד עובד.
אזור החומרה משתלט על השטח של ה – U.M.A.
השורות של הפקודות הראשונות בקובץ של ה – CONFIG.SYS:

```
DEVICE=C:\HIMEM.SYS
DEVICE=C:\EMM386.EXE /_____
```

מטפלות בניהול הזיכרון של המחשב.

תצורת טעינה ל – DOS מורכב משני שורות:

```
DOS = HIGH
DOS = UMB
```

מהוות את החלק השני של CONFIG.SYS ש"מספרות" ל – DOS איך להיטען.
DOS = HIGH – קח חלק מגרעין המערכת בערך 45K ותעביר אותו ב – OFFSET ל – H.M.A. במטרה לפנות את אזור הזיכרון.

DOS = UMB – במידה ואזור התוכנה פתוח עשינו no EMS/ תשתמש בו. הוא נותן ל – DOS הוראה להשתמש באזור התוכנה. בלי השורה הזאת גם אם פתחנו את אזור התוכנה באמצעות no EMS/ אי אפשר להשתמש בה.

שיעור ראשון - 9.5.99
מעבדה

ספק כוח - הפרמטר הכי חשוב מתחי הכניסה שלו נעים בין 220-115V ומאפשרים שימוש בכל המדינות. גם 220V זה לא מתח יציב. ולכן נותנים לספק אפשרות לעבוד בטולראנס מסויים. כך שהקפיצה בין הבדלי הוולט לא ישרפו. יש מספק שבורר בין 220-115V.
שקע זכר - כניסת 220V שקע מאורך, כשמחברים לחשמל רצוי שהאדמה תופיע בשקע.
שקע נקבה - מגיע למוניטור יציאה ממותגת מקבלת מתח אך ורק בתלות של חשמל יציאת 220V.
יתרונות - נוחיות, חוסך תקע בקיר.
חסרונות - אם נכנס מתח גבוה 300V אז הספק מתחמם מאד - הספק שהוא פזור על הרכיבים הוא מפזר חום.
המאוורר הוא מאוורר של 12V בהספק של 3-2W ישנם מארזים ישנים שאין מאוורר פנימי של המחשב. ע"י חרירים מהצד השני של הספק.
תחזוקה מונעת - אחת לחצי שנה - ספרי שימון - WD-40.
ספק שהמאוורר שלו מסתובב לאט בגלל לכלוך ואבק ואז הספק מתחמם ואם יש חריגה מתחמים ואז יש הרבה בעיות.
כשנכנס מתח גבוה הוא שורף את השנאי אבל לפני השנאי יש פיוז - לפעמים כניסות של מתח גבוה שורפות את הפיוז פותחים את המארז מקפצים את הפיוז עם הפיוז עם טסטר ומחליפים את הפיוז. לפני שפותחים 24 שעות לא מחובר לחשמל (מנותק לא בחשמל).

יציאות המתח של הספק מחולקים לשלוש:

1. P8 -P9:
בצבע לבן ויותר גדול ויש להם עוגנים (וו שמחזיק אותם).
חוטים אלו מספקים מתחים ללוח האם - 12V ±, 5V ±, אדמה וקו שנקרא POWER GOOD.
חוטים:

| | |
|---|---|
| אדומים: 12V + | כחול: 5V - |
| לבנים: 12V + | צהוב: 5V + |
| שחורים: אדמה | כתום: POWER GOOD |

POWER GOOD - יש לספק מנגנון בדיקה שאומר האם המתחים תקינים (תקין אומר - הטולראנס של החוט). פולס שמוציא הספק ללוח האם האם המתח תקין או לא. הטולראנס של 12 V הוא:

12.02-11.98V אם יש פלוס חיובי שאומר ללוח האם שהוא יכול לעבוד רק אז הוא עובד.
זמן ההתייצבות - הזמן שלוקח לספק לייצב מתחים ביציאה כי בשניה הראשונה הוא לא עושה זאת.
המתח מגיע ללוח האם באמצעות 2 הפלגים הללו.

2. צבעי החוטים נשארים אותו דבר.
הם משמשים לדיסקים קשיחים, צורבים, CD וכו'.
קבוצת הפלגים - שקופים ופחות יציאות והם זהים. כיום הקבוצה מכילה 5 -6 פלגים ואם צריך עוד אז יש מפצלים אפשר לפצל עד 15-20 יציאות.

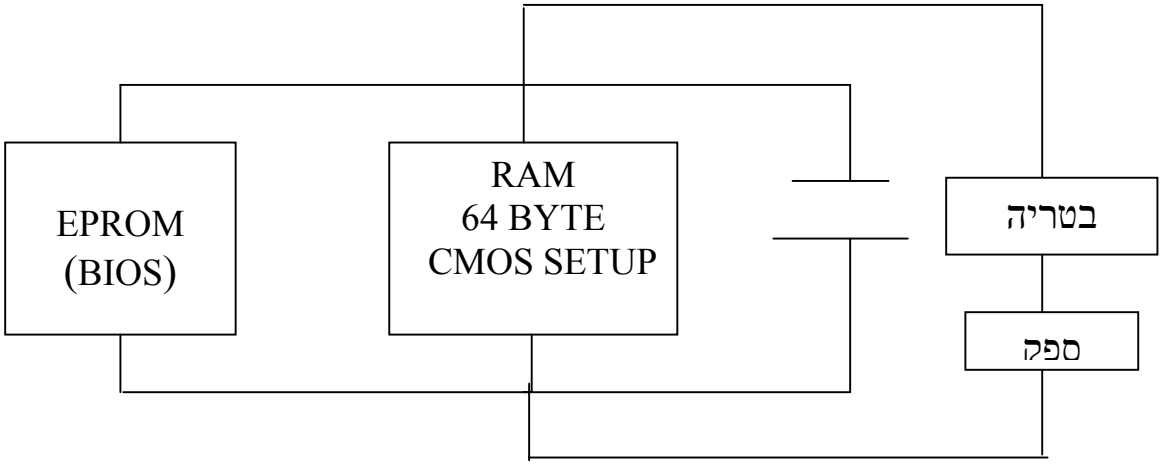
3. קבוצת יחיד - למרות שהיום יש יותר. משתמשים לכונני ה-FLOOPY (1.44). אותם צבעים אותם מתחים גם אם יוצא מאותו מקום.
גם לזה יש מפצלים 10-15 פיצולים וזה עובד.

הספק - הכוח של הספק.

| סוג המעבד | הספק ספק נדרש |
|---|---|
| 286 | 100W |
| 386 | 150-100W |
| 486 | 150W |
| P | 150W |
| P-PRO | 200W מומלץ 20-150 W |
| P-2 | 250W לא פחות |

יותר התקנים דורש יותר הספק.
עקרונית - לא צריך לספק את הספק המאוורר מקרר אותו רצוי להשאיר את המחשב דולק כל הזמן (לא את המסך).
לוח אם -
חיבור P8 -P9 ללוח - החיבור נעשה ששני השחורים מכל פלג קרובים אחד שלני.
אם זה מחובר ההפך מתחים מגעים למקומות שלא נכונים.
מכניסים באלכסון מיישרים ומורידים.
ישנה אופציה חדשה שתומכת בשיטת ATX.
ATX - P8 -P9 נראים קצת אחרת כי התושבת של לוח האם נראה אחרת.
ATX - ספק שמבצע כיבוי בעצמו.

בטריה RAM - WPROM (שמכיל את ה - BIOS).

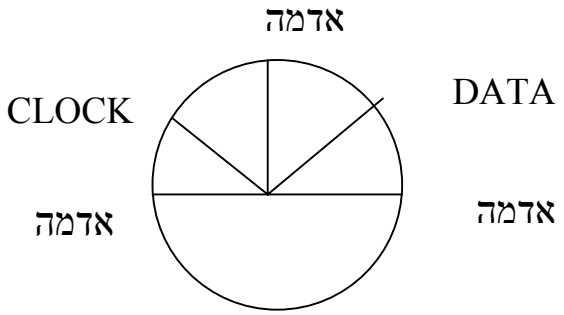



המשולש הזה אומר - יש בטריה שמחוברת לרכיב זיכרון RAM שמכיל את הגדרות ה - SETUP של ה - BIOS. רכיב RAM של 64 בית.
הגדרות SETUP - לא נוכל לכל מחשב יש את אותן הגדרות חומרה ולכן הוא יודע לזהות את ההגדרות שהמשתמש מכניס לו. איזה דיסק איזה מסך מה לעלות קודם וכו.
BIOS - מכיל את הפונקציות שהוא נעזר ב - SETUP.

בטריה - שהספק ב - OFF הבטריה מספקת לו את המתח. ה - RAM הוא נדיף וכדי שהגדרות ה - SETUP לא ילכו לאיבוד מגיבים את ה - RAM ע"י הבטריה.
שהספק ב - ON הבטריה לא אמורה לעבוד. הבטריה מסוג ניקל קדמיום כלומר נטענת. שהספק ב - ON הבטריה יכולה להיטען. כשהספק כבוי היא אמורה לתת מתח ל - RAM.
כשהספק סגור הבטריה מתחילה להתפרק דרך הספק וגם מספקת ל– RAM ולכן הפסקו להשתמש בבטריות האלו אורך החיים של הבטריה היה כמה חודשים.
כיום משתמשים בבטריות מסוג כפתור (כמו שעון).
כשמחליפים את הבטריה המחשב כבוי והמידע נמחק.
דגם הבטריה מסוג כפתור CR2032עובדת בערך שלוש שנים.
סימנים מזהים לבטריות שעומדות להיגמר:
השעון מתחיל לזייף.
איך מחליפים את הבטריה הישנה: קטר בשני הרגלים ומורידים את הדבר האפוקסי.
פתרון נוסף היה בעזרת ארבעה פינים שיושבים משמאל לבטריה.

בטריה חיצונית – היה מתקן חיצוני שהתחבר ל – 4 פינים שנכנסו בו 4 בטריות אצבע ששימשו בתור בטריות גיבוי. ואז שנגמרו הבטריות החליפו אותם וכך לא צריך להשתמש בדבק ובהלחמות.
הבטריות החדשות יכולות להיות בכל מקום על הלוח.
EPEOM –חלון קטן לאור אולטרא סגול. הרכיב יושב על תושבת כך שאפר להחליף אותו למטרת שידרוג ואז להביא למעבדה למטרת מחיקה ותיקון.

מקלדת: הכניסה למקלדת היא ליד החיבור של P8 -P9 המחבר נקרא DIN5 כלומר 5 פינים בריבוד




כיום יש 104 מקשים למקלדת. המקלדת עובדת בשיטה של מטריצה.
המקדד מזהה במטריצה את המיקום ומוציא אותו החוצה לפי קוד ASCII בפולסים. המקדד יושב על קו ה – DATA על ה - CLOCK ויש פולס זיכרון בין המקלדת ללוח.
צריך מפענח כדי לדעת איך לפענח את הקוד ASCII המפענח יושב ליד ה – RAM.

תושבות של ה – DRAM - 8 תושבות שיושבים ליד ה - P8 -P9 בצבע לבן.
באים בשלושה סוגים של הגדלים 30 פין או 70 פין והחדשים 168 פין (יותר ארוך).
SOCKET – תושבת שבה יושבים ה – SIM.
חלוקה לבנקים: