# Lecture 4

סט פקודות של 8051

טסליצקי מרק markalex012@gmail.com www.elecstudy.org

# סט פקודות של 8051

- מבוא
- ארכיטקטורה וארגון זיכרון של CIP-51.
- אפשרויות גישה.
  - Register addressing גישה דרך אוגרים.
  - Direct addressing גישה ישירה.
  - Indirect addressing גישה לא ישירה.
  - Immediate constant addressing גישה בעזרת קבועים מיידיים.
  - Relative addressing גישה לפי כתובת יחסית.
  - Absolute addressing גישה לפי כתובת מוחלטת.
  - Long addressing  גישה לפי כתובת ארוכה.
  - Indexed addressing גישה "ממוספרת".
- סוגי פקודות.
  - פקודות אריתמטיות.
  - פקודות לוגיות.
  - פקודות להעברת נתונים.
  - פקודות משתנים בוליאניים.
  - פקודות קפיצה.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

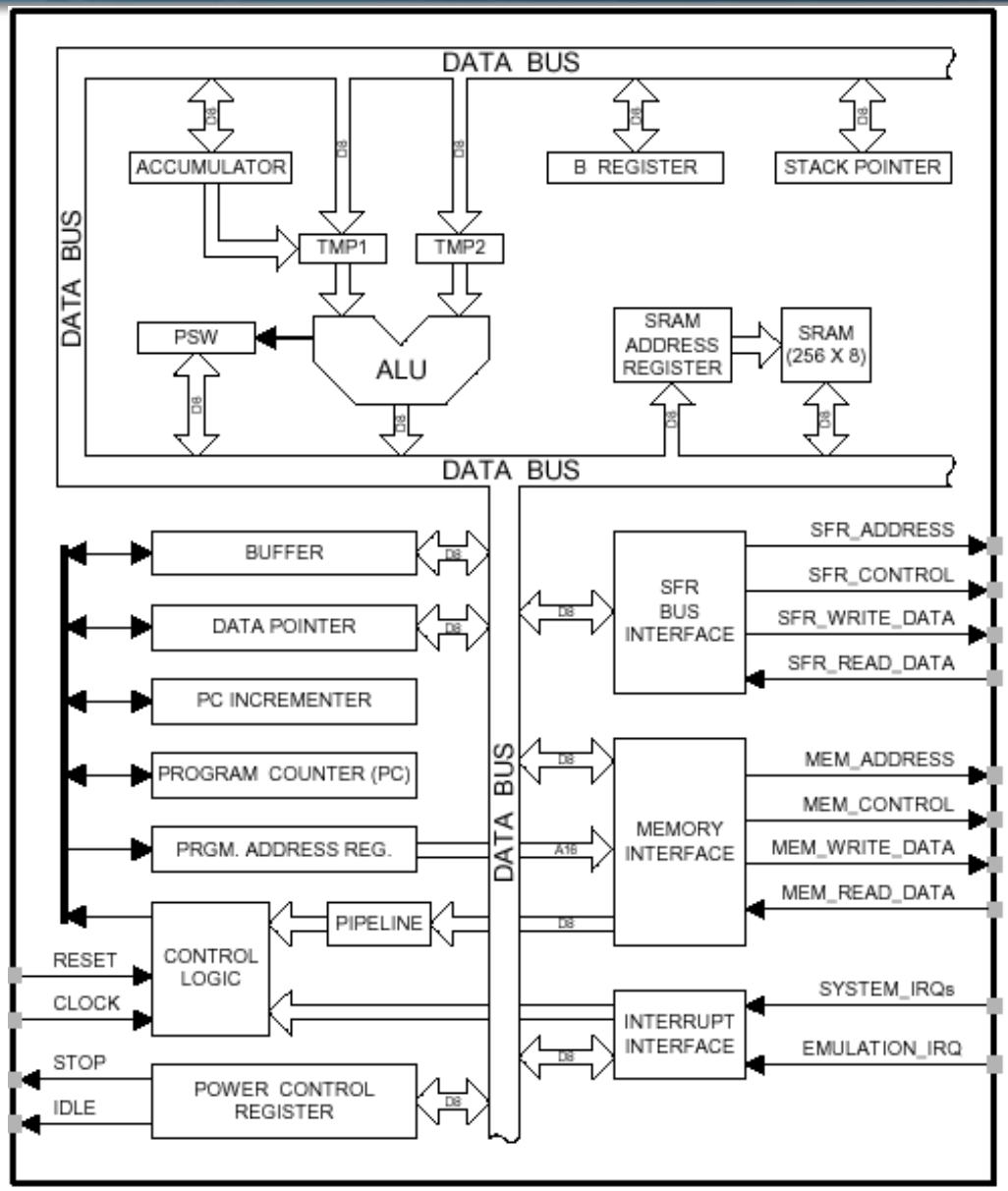
# מבוא

- פקודת מחשב בנויה מקוד הפעלה (op-code), ואחריו באים (או לא באים) אופרנדים לפי סוג הפקודה.
- לפי op-code ניתן לדעת מהו סוג הפקודה, כמה אופרנדים באים ולפי אופרנדים יודעים מה הוא מקור ומה הוא היעד.
- אופרנדים יכולים להיות:
  - ערכים עצמם.
  - אוגרים.
  - זיכרון.
  - פורטים.
- במידה ופקודה באה עם יותר מאופרנד בודד פורמט שלה תמיד יהיה בצורה הבאה:

*Instruction Destination, Source*


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

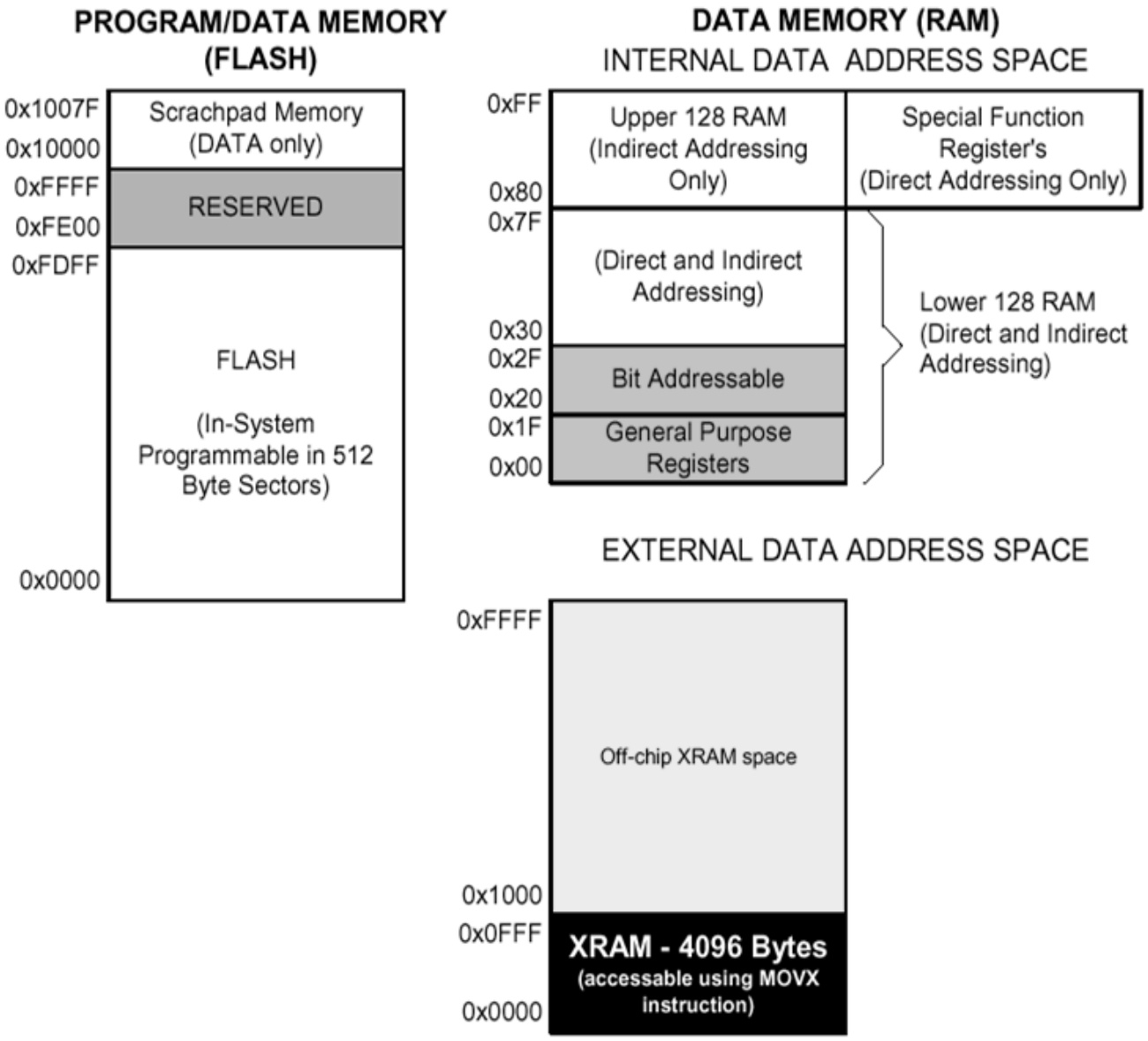
# ארכיטקטורה של CIP-51






www.elecstudy.org markalex012@gmail.com טסליצקי מרק

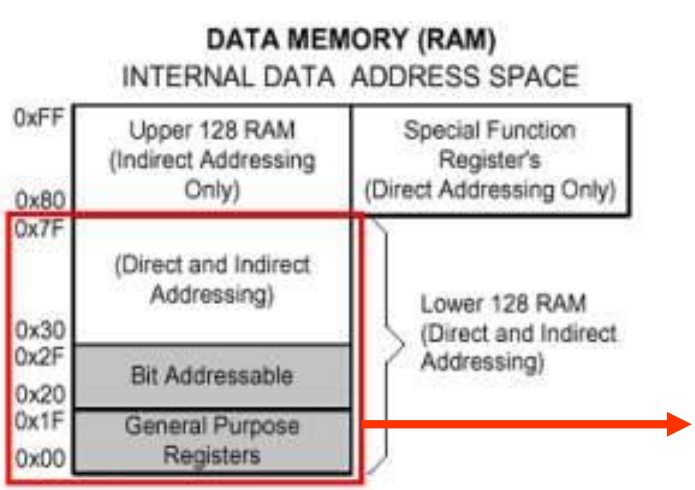
# ארגון הזיכרון




- ארגון הזיכרון של C8051F020 דומה לארגון הזיכרון של 8051 סטנדרטי.
- קוד ונתונים נמצאים באותו מרחב זיכרון, אך צורת הגישה היא שונה.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

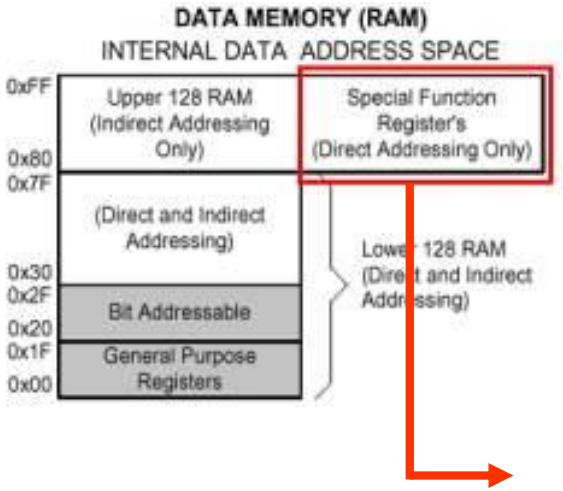
# זיכרון פנימי.




| | Byte Address | Bit Address | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 7F | General Purpose RAM | | | | | | | |
| | 30 | | | | | | | | |
| B | 2F | 7F | 7E | 7D | 7C | 7B | 7A | 79 | 78 |
| i | 2E | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 |
| t | 2D | 6F | 6E | 6D | 6C | 6B | 6A | 69 | 68 |
| | 2C | 67 | 66 | 65 | 64 | 63 | 62 | 61 | 60 |
| A | 2B | 5F | 5E | 5D | 5C | 5B | 5A | 59 | 58 |
| d | 2A | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 |
| d | 29 | 4F | 4E | 4D | 4C | 4B | 4A | 49 | 48 |
| r | 28 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| e | 27 | 3F | 3E | 3D | 3C | 3B | 3A | 39 | 38 |
| s | 26 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 |
| s | 25 | 2F | 2E | 2D | 2C | 2B | 2A | 29 | 28 |
| a | 24 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| b | 23 | 1F | 1E | 1D | 1C | 1B | 1A | 19 | 18 |
| l | 22 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| e | 21 | 0F | 0E | 0D | 0C | 0B | 0A | 09 | 08 |
| | 20 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| | 1F | Bank 3 | | | | | | | |
| | 18 | | | | | | | | |
| | 17 | Bank 2 | | | | | | | |
| | 10 | | | | | | | | |
| | 0F | Bank 1 | | | | | | | |
| | 08 | | | | | | | | |
| | 07 | Default Register Bank for R0 – R7 | | | | | | | |
| | 00 | | | | | | | | |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# אוגרים מיוחדים SFR.




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| F8 | SPI0CN | PCA0H | PCA0CPH0 | PCA0CPH1 | PCA0CPH2 | PCA0CPH3 | PCA0CPH4 | WDTCN |
| F0 | B | SCON1 | SBUF1 | SADDR1 | TL4 | TH4 | EIP1 | EIP2 |
| E8 | ADC0CN | PCA0L | PCA0CPL0 | PCA0CPL1 | PCA0CPL2 | PCA0CPL3 | PCA0CPL4 | RSTSRC |
| E0 | ACC | XBR0 | XBR1 | XBR2 | RCAP4L | RCAP4H | EIE1 | EIE2 |
| D8 | PCA0CN | PCA0MD | PCA0M0 | PCA0CPM1 | PCA0CPM2 | PCA0CPM 3 | PCA0CPM 4 | |
| D0 | PSW | REF0CN | DAC0L | DAC0H | DAC0CN | DAC1L | DAC1H | DAC1CN |
| C8 | T2CON | T4CON | RCAP2L | RCAP2H | TL2 | TH2 | | SMB0CR |
| C0 | SMB0CN | SMB0ST A | SMB0DAT | SMB0ADR | ADC0GTL | ADC0GTH | ADC0LTL | ADC0LTH |
| B8 | IP | SADEN0 | AMX0CF | AMX0SL | ADC0CF | P1MDIN | ADC0L | ADC0H |
| B0 | P3 | OSCXCN | OSCICN | | | P74OUT | FLSCL | FLACL |
| A8 | IE | SADDR0 | ADC1CN | ADC1CF | AMX1SL | P3IF | SADEN1 | EMI0CN |
| A0 | P2 | EMI0TC | | EMI0CF | P0MDOUT | P1MDOUT | P2MDOUT | P3MDOUT |
| 98 | SCON0 | SBUF0 | SPI0CFG | SPIODAT | ADC1 | SPI0CKR | CPT0CN | CPT1CN |
| 90 | P1 | TMR3CN | TMR3RLL | TMR3RLH | TMR3L | TMR3H | P7 | |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | CKCON | PSCTL |
| 80 | P0 | SP | DPL | DPH | P4 | P5 | P6 | PCON |
| | 0(8) Bit addressable | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# אפשרויות גישה.

- במיקרו בקר C8051F020 ישנם 8 אפשרויות גישה.

| אפשרויות גישה. | דוגמא |
|---|---|
| **Register** | **MOV A, B** |
| **Direct** | **MOV 30H,A** |
| **Indirect** | **ADD A,@R0** |
| **Immediate Constant** | **ADD A,#80H** |
| **Relative*** | **SJMP AHEAD** |
| **Absolute*** | **AJMP BACK** |
| **Long*** | **LJMP FAR_AHEAD** |
| **Indexed** | **MOVC A,@A+PC** |

* Related to program branching instructions


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# גישה דרך אוגרים Register Addressing

- גישה דרך אוגרים מאפשרת העברת מידע בין אוגר לאוגר.
- *לדוגמא*

```
MOV     R0, A
```

- הפקודה מעבירה תוכן המצבר לתוך אוגר R0 . בחירת בנק (Bank 0, 1, 2 or 3) נעשה לפני פקודה הזאת.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Direct Addressing גישה ישירה

- צורה הזאת מאפשרת עברת נתונים ע"י כתובת של זיכרון או שם "כינוי" של פורטים. "כינויים" רשומים בתוך קובץ "C8051F020.inc".

- *דוגמא*:

```
MOV    A, P3
```

**מעבירים תוכן של פורט 3 למצבר**

```
MOV    A, 020H
```

**מעבירים תוכן של תא מספר 20 הקסה של זיכרון RAM לתוך מצבר**


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Indirect Addressing גישה לא ישירה (עקיפה)

- גישה הזאת משתמשת במצביע כדי לשמור כתובת אפקטיבית של אופרנד.
- רק אוגרים R0, R1 ואוגר DPTR מיועדים לצורת העבודה הזאת.
- אוגרים R0, R1 יכולים להצביע על כתובת 8 סיביות ואוגר DPTR יכול להצביע על כתובת 16 סיביות.
- ***דוגמא:***

```
MOV     @R0,A
```

**התוכן של המצבר מועבר לתוכן שאוגר R0 מצביע עליו (לדוגמא 60H )**

```
MOVX    A,@DPTR
```

**תוכן של תא שאוגר DPTR מצביע עליו (לדוגמא 1234H) מועבר למצבר.**


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Immediate Constant Addressing גישה בעזרת קבועים מיידיים

- ניתן להשתמש בקבועים באורך 8 או 16 סיביות, וזה תלוי בסוג אופרנדים.
- אוגר היעד צריך להיות באותו גודל כמו הקבוע שמעתיקים עליו.

- *דוגמא:*

```
ADD A,#030H
```

**הוספת ערך 30H (8 סיביות) לאוגר (שהוא גם 8 סיביות).**

```
MOV DPTR,#0FE00H
```

**העברת ערך FE00H (16 סיביות) למצביע DPTR (שהוא גם 16 סיביות).**


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# גישה לפי כתובת יחסית Relative Addressing

- צורה הזאת שימושית בפקודות מסוג קפיצות במרחקים קצרים. JNZ ,SJMP (short jump)
- בעזרת פקודות האלה ניתן להגיע מצד אחד של התוכנה לצד אחר.
- כתובת ביעד צריכה להיות במרחק לא יותר מ- 127 בית קדימה או לא יותר מ- 128 אחרוה מהפקודה הנוכחית, עקב כך שאסת של הפקודה הוא 2 256 = $2^8$.
- במידת הצורך אפשר להשתמש בקפיצות כפולות כדי להגיע לנקודה רחוקה יותר.
- *דוגמא:*

```
GoBack:    DEC     A          ;Decrement A
           JNZ     GoBack ;If A is not zero, loop back
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Absolute Addressing גישה ע"י כתובת מוחלטת.

- ישנם 2 פקודות שעובדות לפי צורה הזאת: ACALL ו- AJMP.
- פקודות האלה תופסות 2 בית כל אחד ומצביעות על כתובת מוחלטת באורך 11 סיביות בתור אופרנד.
- 5 סיביות עליונות מתוך 16 סיביות לא ניתנות לגישה או לשינוי – הן תמיד אפסים, עקב כך שזיכרון מוגבל ל2048 תאים. (2K) .
- *דוגמא*:

```
        ACALL   PORT_INIT
```

**PORT_INIT צריכה להיות במרחק של 2K**

```
PORT_INIT: MOV     P0, #0FH
```

**קפיצה לתת שגרה PORT_INIT**


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Long Addressing גישה לכתובת ארוכה

- צורה הזאת ממומשת ע"י פקודות LCALL ו- LJMP.
- פקודות האלה הן באורך 3 בית ו-2 בית אחרונים מכילים כתובת היעד לקפיצה של 16 סיביות.
- זה מאפשר עבודה עם אזור זיכרון של 64 K.
- תוכנה תמיד תקפוץ למיקום שצוין באופן מוחלט ולא משנה באיזה מקום היא הייתה.
- *דוגמא*:

```
        LCALL TIMER_INIT
```

**פקודה שבשפת C נראת כך:** `\\Timer_Init();`

**תת שגרה** `TIMER_INIT: ORL TMOD,#01H  ;TIMER_INIT`


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Indexed Addressing גישה "ממוספרת"

- משתמשים בגישה הזאת, כאשר מדובר בנתונים סדרתיים שנמצאים ב"טבלה" (רצף).
- אוגר 16 סיביות מצביע על כתובת הבסיסי של הטבלה ומצבר מכיל כתובת יחסית של כל נתון.
- סכום של שני נתונים האלה נותן כתובת אפקטיבית לפקודות JMP ו- MOVC.
- *דוגמא:*

```
MOV     A,#08H          ;כתובת של נתון מתחילת הטבלה
MOV     DPTR,#01F00H    ;כתובת של הטבלה
MOVC    A,@A+DPTR       ;מצבר מקבל נתון ששני אוגרים מצביעים עליו
```

- לאחר הפעלת הקוד שכתוב כאן נתון שנמצא בכתובת (1F00H+08H) 1F08H יועבר למצבר.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# סוגי פקודות:

- במיקרו בקר C8051F020 ישנם 5 סוגי פקודות:
  - פעולות אריתמטיות Arithmetic operations
  - פעולות לוגיות Logical operations
  - פעולות עברת נתונים Data transfer operations
  - פעולות עם משתנים בוליאניים Boolean variable operations
  - פעולות קפיצה בתוכנה Program branching operations


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פעולות אריתמטיות:

| Mnemonic | Description |
|---|---|
| ADD A, Rn | A = A + [Rn] |
| ADD A, direct | A = A + [direct memory] |
| ADD A,@Ri | A = A + [memory pointed to by Ri] |
| ADD A,#data | A = A + immediate data |
| ADDC A,Rn | A = A + [Rn] + CY |
| ADDC A, direct | A = A + [direct memory] + CY |
| ADDC A,@Ri | A = A + [memory pointed to by Ri] + CY |
| ADDC A,#data | A = A + immediate data + CY |
| SUBB A,Rn | A = A - [Rn] - CY |
| SUBB A, direct | A = A - [direct memory] - CY |
| SUBB A,@Ri | A = A - [@Ri] - CY |
| SUBB A,#data | A = A - immediate data - CY |
| INC A | A = A + 1 |
| INC Rn | [Rn] = [Rn] + 1 |
| INC direct | [direct] = [direct] + 1 |
| INC @Ri | [@Ri] = [@Ri] + 1 |
| DEC A | A = A - 1 |
| DEC Rn | [Rn] = [Rn] - 1 |
| DEC direct | [direct] = [direct] - 1 |
| DEC @Ri | [@Ri] = [@Ri] - 1 |
| MUL AB | Multiply A & B |
| DIV AB | Divide A by B |
| DA A | Decimal adjust A |

- בהתאם לתוצאת הפקודה דגלים מסוימים בתוך אוגר דגלים PSW יכולים להשתנות: לדוגמא דגל נשא, דגל אפס וכו'.
- כאשר כתוב @Ri – מדובר על מצביע על אזור הזיכרון שמבצעים בעזרת אוגרים R0 ו- R1.
- כאשר כתוב Rn מדובר על האוגרים R0-R7 שנמצאים בתוך בנק האוגרים הנבחר מראש.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פעולות לוגיות.

| Mnemonic | Description |
|---|---|
| ANL A, Rn | A = A & [Rn] |
| ANL A, direct | A = A & [direct memory] |
| ANL A,@Ri | A = A & [memory pointed to by Ri] |
| ANL A,#data | A= A & immediate data |
| ANL direct,A | [direct] = [direct] & A |
| ANL direct,#data | [direct] = [direct] & immediate data |
| ORL A, Rn | A = A OR [Rn] |
| ORL A, direct | A = A OR [direct] |
| ORL A,@Ri | A = A OR [@RI] |
| ORL A,#data | A = A OR immediate data |
| ORL direct,A | [direct] = [direct] OR A |
| ORL direct,#data | [direct] = [direct] OR immediate data |
| XRL A, Rn | A = A XOR [Rn] |
| XRL A, direct | A = A XOR [direct memory] |
| XRL A,@Ri | A = A XOR [@Ri] |
| XRL A,#data | A = A XOR immediate data |
| XRL direct,A | [direct] = [direct] XOR A |
| XRL direct,#data | [direct] = [direct] XOR immediate data |
| CLR A | Clear A |
| CPL A | Complement A |
| RL A | Rotate A left |
| RLC A | Rotate A left (through C) |
| RR A | Rotate A right |
| RRC A | Rotate A right (through C) |
| SWAP A | Swap nibbles |

- פעולות מבצעות פעולות בוליאניות סיבית על סיבית. (AND, OR, XOR, and NOT)
- דוגמאות:

```
ANL  A, #02H
```

**משירים סיבית 1 במצבר כל היתר מאפסים**

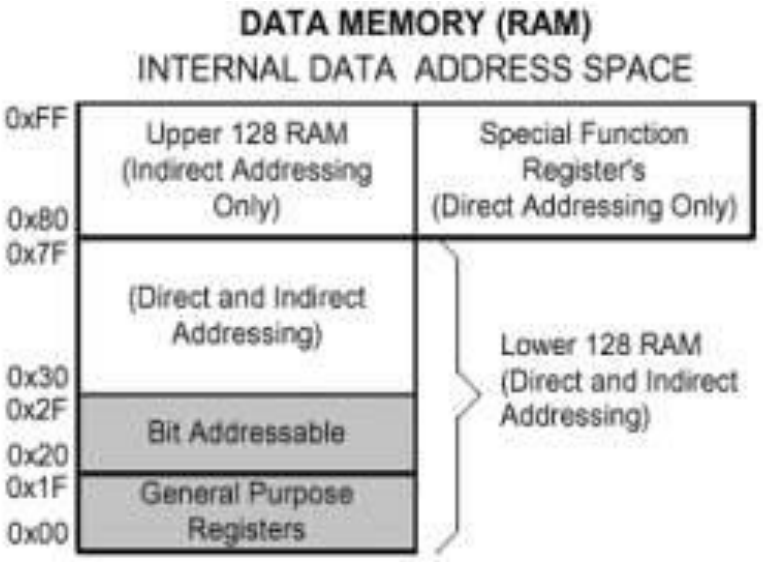
```
ORL  TCON, A
```

**פעולות OR לוגי בין TCON ומצבר עם הכנסת תוצאה לתוך TCON.**


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Data Transfer Instructions פקודות להעברת נתונים

- העברת נתונים מתבצעת כדי להעביר נתונים מRAM פנימי לאוגרים מיוחדים ללא שימוש במצבר.
- בנוסף יש אפשרות להעביר נתונים בין RAM פנימי ל- RAM חיצוני בעזרת גישה עקיפה.
- 128 בית עליונים של RAM ניתנים לגישה אך ורק בגישה אקיפה וגישה לאוגרים מיוחדים SFR שנמצאים באזור הזה ניתנים לגישה רק דרך גישה ישירה.




| Mnemonic | Description |
|---|---|
| MOV @Ri, direct | [@Ri] = [direct] |
| MOV @Ri, #data | [@Ri] = immediate data |
| MOV DPTR, #data 16 | [DPTR] = immediate data |
| MOVC A,@A+DPTR | A = Code byte from [@A+DPTR] |
| MOVC A,@A+PC | A = Code byte from [@A+PC] |
| MOVX A,@Ri | A = Data byte from external ram [@Ri] |
| MOVX A,@DPTR | A = Data byte from external ram [@DPTR] |
| MOVX @Ri, A | External[@Ri] = A |
| MOVX @DPTR,A | External[@DPTR] = A |
| PUSH direct | Push into stack |
| POP direct | Pop from stack |
| XCH A,Rn | A = [Rn], [Rn] = A |
| XCH A, direct | A = [direct], [direct] = A |
| XCH A, @Ri | A = [@Rn], [@Rn] = A |
| XCHD A,@Ri | Exchange low order digits |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Boolean Variable Instructions פקודות בוליאניות

- מיקרו בקר C8051F020 יכול לבצע פעולות עם סיביות בודדות.
- פקודות האלה הן: לאפס, לאתחל, או, וגם, הפיכה לוגית.
- בנוסף ישנה אפשרות להעברת סיביות וקפיצות שונות בהתאם למצב סיביות.
- כל פעולות עם סיביות דורשות גישה ישירה.
- **דוגמאות:**

```
SETB   TR0      ;לאתחל טיימר 0.
POLL:  JNB      TR0, POLL
```

**;להיות בלולאה כל עוד טיימר 0 שווה ל-0**

| Mnemonic | Description |
|---|---|
| CLR C | Clear C |
| CLR bit | Clear direct bit |
| SETB C | Set C |
| SETB bit | Set direct bit |
| CPL C | Complement c |
| CPL bit | Complement direct bit |
| ANL C,bit | AND bit with C |
| ANL C,/bit | AND NOT bit with C |
| ORL C,bit | OR bit with C |
| ORL C,/bit | OR NOT bit with C |
| MOV C,bit | MOV bit to C |
| MOV bit,C | MOV C to bit |
| JC rel | Jump if C set |
| JNC rel | Jump if C not set |
| JB bit,rel | Jump if specified bit set |
| JNB bit,rel | Jump if specified bit not set |
| JBC bit,rel | if specified bit set then clear it and jump |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Program Branching Instructions פקודות קפיצה

- פקודות קפיצה מבצעים העברה למקום מסוים של קוד עם או ללא תנאי מסוים.
- ישנם פקודות להפעלת שגרות, חזרות משגרות או פסיקות, קפיצות לפי כתובת יחסית או לפי כתובת מוחלטת, קפיצות בהתאם לתנאי בדיקה.

| Mnemonic | Description |
|---|---|
| ACALL addr11 | Absolute subroutine call |
| LCALL addr16 | Long subroutine call |
| RET | Return from subroutine |
| RETI | Return from interrupt |
| AJMP addr11 | Absolute jump |
| LJMP addr16 | Long jump |
| SJMP rel | Short jump |
| JMP @A+DPTR | Jump indirect |
| JZ rel | Jump if A=0 |
| JNZ rel | Jump if A NOT=0 |
| CJNE A,direct,rel | Compare and Jump if Not Equal |
| CJNE A,#data,rel | |
| CJNE Rn,#data,rel | |
| CJNE @Ri,#data,rel | |
| DJNZ Rn,rel | Decrement and Jump if Not Zero |
| DJNZ direct,rel | |
| NOP | No Operation |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פירות פקודות.

8051 Instruction

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Arithmetic Operations פקודות אריתמטיות

- במידה וכתוב [@Ri] מדובר על תוכן שאוגרים R0 או R1 מצביעים עליו.
- במידה וכתוב Rn מדובר על אוגרים R0-R7 מהבנק שנבחר מראש.

| Mnemonic | Description |
|---|---|
| ADD A, Rn | A = A + [Rn] |
| ADD A, direct | A = A + [direct memory] |
| ADD A,@Ri | A = A + [memory pointed to by Ri] |
| ADD A,#data | A = A + immediate data |
| ADDC A,Rn | A = A + [Rn] + CY |
| ADDC A, direct | A = A + [direct memory] + CY |
| ADDC A,@Ri | A = A + [memory pointed to by Ri] + CY |
| ADDC A,#data | A = A + immediate data + CY |
| SUBB A,Rn | A = A - [Rn] - CY |
| SUBB A, direct | A = A - [direct memory] - CY |
| SUBB A,@Ri | A = A - [@Ri] - CY |
| SUBB A,#data | A = A - immediate data - CY |
| INC A | A = A + 1 |
| INC Rn | [Rn] = [Rn] + 1 |
| INC direct | [direct] = [direct] + 1 |
| INC @Ri | [@Ri] = [@Ri] + 1 |
| DEC A | A = A - 1 |
| DEC Rn | [Rn] = [Rn] - 1 |
| DEC direct | [direct] = [direct] - 1 |
| DEC @Ri | [@Ri] = [@Ri] - 1 |
| MUL AB | Multiply A & B |
| DIV AB | Divide A by B |
| DA A | Decimal adjust A |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ADD A,<בית מקור> ADDC A,<בית מקור>

- פקודה ADD מוסיפה תוכן של בית מקור למצבר ומשנה תוכן של המצבר. בית מקור נשאר ללא שינוי.
- פקודה ADDC מוסיפה תוכן של בית מקור לערך של סיבית נשא C וכל זה למצבר ומשנה תוכן של המצבר. בית מקור נשאר ללא שינוי.
- תוצאת הפעולות האלה יכולות לשנות אוגרים הבאים: דגל נשא (CY) , דגל נשא נוסף (AC) ודגל מילוי יתר (OV). דגלים האלה נמצאים באוגר PSW. (Program Status Word).
  - CY=1 אם הייה מילוי יתר מסיבית 7 דגל נשא עולה ל-1 אחרת מתאפס.
  - AC =1 אם הייה מילוי יתר מ4 סיביות נמוכות של בית של מצבר (זאת אומרת מסיבית 3 ) דגל נשא עזר עולה ל-1, אחרת מתאפס.
  - OV=1 אם תוצאה מסומנת לא יכולה להיכנס לאופרנד היעד דגל מילוי יתר עולה ל-1 אחרת הוא מתאפס.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# SUBB A,< בית מקור >

- פקודה SUBB בדומה לפקודה ADDC מחסירה ערך של בית מקור וערך של נשא מהתוכן של מצבר ומכניסה ערך חדש לתוך המצבר ומשנה דגלים הרלוונטיים.

CY=1 במידה ויש צורך בנשא לסיבית מספר 7 דגל זה עולה ל-1 אחרת מתאפס.

AC =1 במידה ויש צורך בנשא לסיבית מספר 3 דגל זה עולה ל-1 אחרת מתאפס.

OV=1 במידה ויש צורך בנשא לסיבית מספר 7 אבל לא לסיבית מספר 6, או לסיבית מספר 6 ולא לסיבית מספר 7, דגל זה עולה ל-1 אחרת מתאפס.

- *דוגמא* :

מצבר מכיל ערך 0C1H, אוגר R1 מכיל ערך 40H ודגל נשא שווה ל-1 (CY=1).
לאחר פקודה:

```
SUBB   A, R1
```

**ערך של R1 לא ישתנה, ערך של מצבר ישתנה ל-** 70H ומצב הדגלים יהיה:

CY=0 AC=0 OV=1


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# INC <יעד>

- מעלה ב-1 תוכן של תא, אוגר או אוגר שמצביע על תא של זיכרון.
- *דוגמא*:

```
INC    6FH
```

במידה ותא בכתובת 6FH מכיל ערך 30H הערך של תא לאחר ביצוע פקודה ישתנה לערך 31H.

- *Example*:

```
MOV    R1, #5E
INC    R1
INC    @R1
```

- התוכן של R1 יהיה 5E , לאחר פקודה שנייה התוכן של אוגר יעלה ב-1 לערך 5F. לאחר פקודה שלישית, תוכן של תא ב-RAM הפנימי שאוגר R1 מצביע עליו יעלה ב-1.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# DEC <יעד>

- פעולה מורידה 1 מהערך שיש ביעד.
- מורידה 1 מהתוכן של תא, אוגר או אוגר שמצביע על תא של זיכרון.
- במידה ומורידים 1 מהתא שמכיל 0 הערך החדש יהיה 0FFH.
- דגלים לא משתנים.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# INC DPTR

- פקודה מעלה ב-1 מצביע באורך 16 סיביות.
- DPTR זה האוגר היחיד שמכיל 16 סיביות.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MUL AB

- פקודה מכפילה תוכן של מצבר A בתוכן של אוגר B ותוצאה נכנסת בחזרה לשני אוגרים האלה: חלק התחתון נכנס למצבר A וחלק העליון נכנס לאוגר B.
- מספרים A ו- B הם מספרים לא מסומנים!!!!

- במידה ותוצאה גדולה מ-255 (FFH) דגל מילוי יתר (OV) עולה ל-1, אחרת הוא מתאפס. דגל נשא תמיד מתאפס ללא תלות בתוצאת הכפל.
- לדוגמא אם מצבר מכיל ערך ACC=85 (55H) ואוגר B מכיל B=23 (17H), אז לאחר ביצוע פקודה נקבל תוצאה: 1955 (07A3H), ואז מצבר מכיל ערך A3H ואוגר B מכיל 07H. דגל OV יעלה ל-1 ודגל C מתאפס.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# DIV AB פעולת חילוק.

- מחלק תוכן של אוגר A בתוכן של אוגר B.
- חלק שלם של תוצאה נכנס לתוך אוגר A ושארית נכנס לתוך אוגר B.

- לדוגמא מצבר מכיל ערך 91 (5BH) ACC=91 ואוגר B מכיל ערך 5 B=05(05H), לאחר פעולת חילוק 91/5 = 18, מצבר יכיל ערך 18 (12H) ושארית 1 יעובר לאוגר B.

- לאחר סיום הפקודה גם דגל נשא וגם דגל מילוי יתר יתאפסו.

- *במידה ואוגר B יכיל 0 לפני פעולת חילוק, לאחר פעולת חילוק תוכן של A ותוכן של B יהיו לא מוגדרים, דגל נשא יתאפס, ודגל מילוי יתר יעלה ל-1.*


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פעולה להמרה מהקסה דצימלי לקוד BCD. DA A

- פעולה מיועדת להכנת מספר לפורמט BCD.
- פעולה הזאת עובדת רק על המצבר ורק לאחר פעולת ADD או ADC ובונה מספר שמורכב מ2 ספרות בפורמט BCD (מ-0 עד 99).
- פקודה מוסיפה ערכים 00H, 06H, 60H or 66H בהתאם למצב של מצבר ו- PSW.
- במידה וערך בסיביות A3-0 גדול יותר מ9 (xxxx1010-xxxx1111) , או דגל AC שווה ל-1 אז פקודה מוסיפה למצבר ערך 6 כדי ליצור ערך נכון ברביעה תחתונה של המצבר.
- אם דגל CY=1, עקב כך שמספר ברביעה עליונה גדול מ-9 (1010xxxx-1111xxxx) אז מוסיפים 6 לחלק העליון כדי ליצור מספר BCD נכון, אבל דגל CY לא מתאפס.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Logical Operations פעולות לוגיות.

| Mnemonic | Description |
|---|---|
| ANL A, Rn | A = A & [Rn] |
| ANL A, direct | A = A & [direct memory] |
| ANL A,@Ri | A = A & [memory pointed to by Ri] |
| ANL A,#data | A= A & immediate data |
| ANL direct,A | [direct] = [direct] & A |
| ANL direct,#data | [direct] = [direct] & immediate data |
| ORL A, Rn | A = A OR [Rn] |
| ORL A, direct | A = A OR [direct] |
| ORL A,@Ri | A = A OR [@RI] |
| ORL A,#data | A = A OR immediate data |
| ORL direct,A | [direct] = [direct] OR A |
| ORL direct,#data | [direct] = [direct] OR immediate data |
| XRL A, Rn | A = A XOR [Rn] |
| XRL A, direct | A = A XOR [direct memory] |
| XRL A,@Ri | A = A XOR [@Ri] |
| XRL A,#data | A = A XOR immediate data |
| XRL direct,A | [direct] = [direct] XOR A |
| XRL direct,#data | [direct] = [direct] XOR immediate data |
| CLR A | Clear A |
| CPL A | Complement A |
| RL A | Rotate A left |
| RLC A | Rotate A left (through C) |
| RR A | Rotate A right |
| RRC A | Rotate A right (through C) |
| SWAP A | Swap nibbles |

- פעולות לוגיות: AND, OR, XOR, NOT מבצעים בדרך כלל עם אוגר A ופעולות עובדות סיבית מול סיבית.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ANL <dest-byte>,<source-byte> פעולת וגם

- פעולה מבצעת פעולת AND לוגי סיבית מול סיבית בין מקור ליעד ומעדכנת תוכן של אוגר היעד.
- אף דגל לא משתנה
- *דוגמא:*

```
ANL    A,R2
```

במידה וACC מכיל ערך ( 11010011) D3H ואוגר R2 מכיל 75H (01110101), אז תוצאת הפעולה היא ( 01010001) ACC=51H

- פעולת AND מממשת פעולת בדיקת סיביות מסוימות או כיבוי מנורות. לדוגמא:

```
ANL    P1,#10111001B
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ORL <dest-byte>,<source-byte> פעולת או לוגי

- פעולה מבצעת פעולת OR לוגי סיבית מול סיבית בין מקור ליעד ומעדכנת תוכן של אוגר היעד.
- אף דגל לא משתנה

- *דוגמא:*

```
ORL    A,R2
```

אם אוגר ACC=D3H (11010011) ואוגר R2=75H (01110101) אז לאחר פעולת OR (11110111) ACC=F7H.

- *עוד דוגמא:*

```
ORL    P1,#11000010B
```

הוראה זאת מעלה סיביות 1,6,7 של P1, ללא שינוי של רגליים אחרות.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פעולת לוגי XOR <dest-byte>,<source-byte> XRL

- פעולה מבצעת פעולת XOR לוגי סיבית מול סיבית בין מקור ליעד ומעדכנת תוכן של אוגר היעד.
- אף דגל לא משתנה

- *דוגמא*:

```
XRL    A,R0
```

אם אוגר ACC=C3H (11000011) ואוגר R0=AAH (10101010) אז תוצאת פעולת XOR שנכנסת לאוגר ACC=69H (01101001)

- *דוגמא*:

```
XRL    P1,#00110001
```

פקודה הזאת הופכת סיביות 0,4,5 של P1 (0 ← 1 ,1 ← 0).

(0 ← 1 ,1 ← 0).


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# CLR A and CPL A

## CLR A

- פקודה הזאת מנקה את התוכן של אוגר A.
- דגלים לא משתנים.

## CPL A

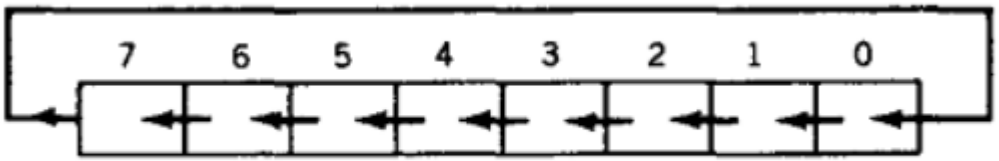
- פקודה מבצעת היפוך לוגי של אוגר A (כל אחד מ8 סיביות משנה ערך שלו (0 ← 1, 1 ← 0). (one's complement)
- דגלים לא משתנים .
- אם תוכן אוגר (ACC=C3H (11000011, אז לאחר פעולת CPL ערך החדש יהיה (ACC=3CH (00111100.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RL A

- 8 סיביות של המצבר מוזזים שמאלה כל אחד וסיבית 7 מוזז למקום 0.
- דגלים לא משתנים.

- לדוגמא אם מצבר (ACC=C3H (11000011 , לאחר פעולת הזזה ערך שלו (ACC=87H (10000111 ודגל CY נשאר ללא שינוי.

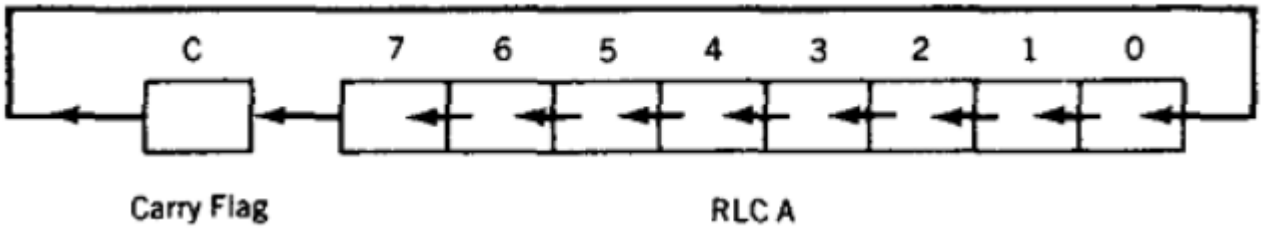


RL A


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RLC A

- מצבר מוזז שמאלה דרך דגל נשא. זאת אומרת סיביות מ-0 עד 6 זזות שמאלה סיבית 7 נכנס לתוך דגל נשא ודגל נשא נכנס לתוך ביט 0.
- כל היתר דגלים לא משתנים.
- לדוגמא נתון ACC=C3H (11000011) ודגל נשא שווה ל-1. התוצאה תהיה ACC=87H (10000111) ודגל נשא יהיה גם 1.

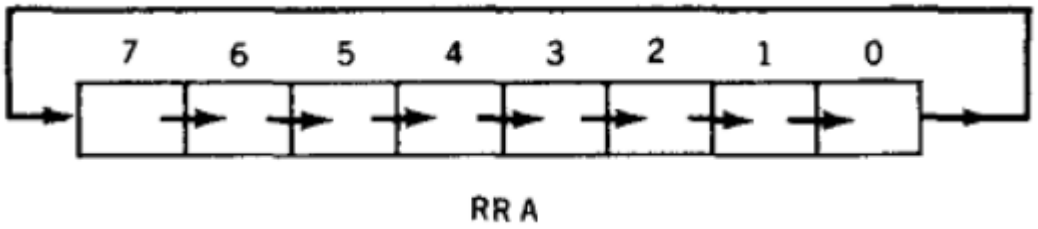




www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RR A

- 8 סיביות של המצבר מוזזים ימינה כל אחד וסיבית 7 מוזז למקום 0.
- דגלים לא משתנים.
- לדוגמא אם מצבר (ACC=C3H (11000011 , לאחר פעולת הזזה ערך שלו (ACC=E1H (11100001 ודגל CY נשאר ללא שינוי.

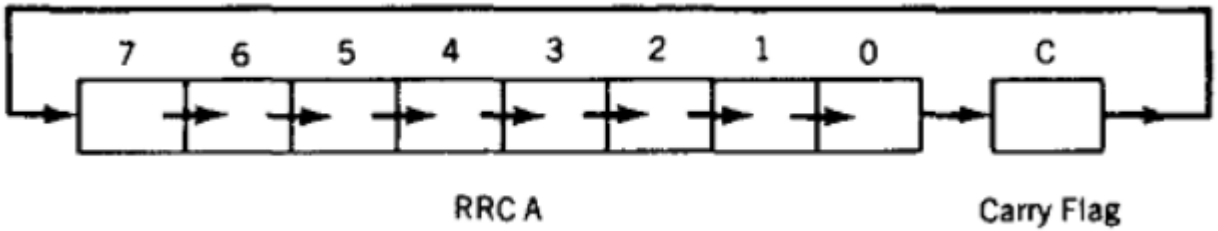


RR A


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RRC A

- מצבר מוזז ימינה דרך דגל נשא. זאת אומרת סיביות מ-1 עד 7 זזות ימינה סיבית 0 נכנס לתוך דגל נשא ודגל נשא נכנס לתוך ביט 7.
- כל היתר דגלים לא משתנים.
- לדוגמא נתון ACC=C3H (11000011) ודגל נשא שווה ל-0. התוצאה תהיה ACC= 61H (01100001) ודגל נשא יהיה 1.

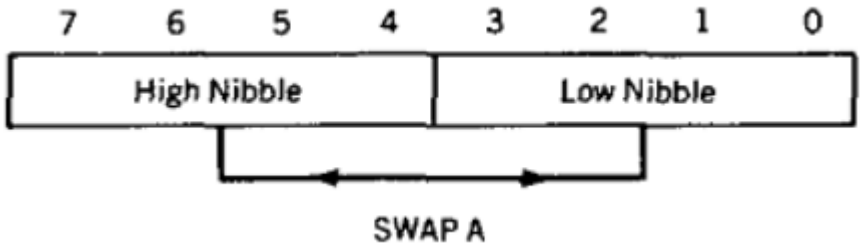




www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# SWAP A

- פקודה הזאת מחליפה 4 סיביות נמוכות של מצבר עם 4 סיביות עליונות.
- פעולה הזאת יכולה להתבצע ע"י פעול RR או 4 RL פעמים.
- דגלים לא משתנים.
- במידה ותוכן מצבר (11000011) ACC=C3H, אז אחרי פעולת SWAP תוכן של מצבר יהיה (00111100) ACC=3CH.





www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Data Transfer Instructions פקודות להעברת נתונים

- פקודות להעברת נתונים משתמשים כדי להעביר נתונים בין אוגרים מיוחדים לבין RAM ללא שימוש במצבר.
- ניתן להשתמש בהעברת נתונים בין RAM פנימי לבין RAM חיצוני בעזרת "העברה לא ישירה".
- 128 בית עליונים של RAM ניתנים לגישה בצורה "לא ישירה" ואוגרי SFR בגישה "ישירה".


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Data Transfer Instructions פעולות העברת נתונים

| פקודה | הסבר |
|---|---|
| MOV @Ri, direct | [@Ri] = [direct] |
| MOV @Ri, #data | [@Ri] = immediate data |
| MOV DPTR, #data 16 | [DPTR] = immediate data |
| MOVC A,@A+DPTR | A = Code byte from [@A+DPTR] |
| MOVC A,@A+PC | A = Code byte from [@A+PC] |
| MOVX A,@Ri | A = Data byte from external ram [@Ri] |
| MOVX A,@DPTR | A = Data byte from external ram [@DPTR] |
| MOVX @Ri, A | External[@Ri] = A |
| MOVX @DPTR,A | External[@DPTR] = A |
| PUSH direct | Push into stack |
| POP direct | Pop from stack |
| XCH A,Rn | A = [Rn], [Rn] = A |
| XCH A, direct | A = [direct], [direct] = A |
| XCH A, @Ri | A = [@Rn], [@Rn] = A |
| XCHD A,@Ri | Exchange low order digits |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOV <dest-byte>,<source-byte>

- פקודה מעבירה (מעתיקה) תוכן של מקור לתוך היעד.
- תוכן המקור לא משתנה אף דגל לא משתנה.

- *דוגמאות:*

```
MOV   R1,#60      ;R1=60H
MOV   A,@R1       ;A=[60H]
MOV   R2,#61      ;R2=61H
ADD   A,@R2       ;A=A+[61H]
MOV   R7,A        ;R7=A
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOV DPTR, #data 16

- פקודה מעתיקה ערך של קבוע 16 סיביות למצביע לנתונים.
- *דוגמא*:

```
MOV    DPTR,#1032
```

פקודה מעתיקה DPH=10H וערך DPL=32H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOVC A,@A + <base-reg>

- פקודה הזאת מעבירה את בית של קוד לתוך ACC.
- כתובת אפקטיבית מורכבת מכתובת 8סיביות ועוד כתובת של אוגר הבסיס שזה יכול להיות או DPTR או PC. (סופר פקודות). כך בונים 16 סיביות כתובת. אף דגל לא משתנה.
- פקודה הזאת שימושית כדי לקרוא טבלאות בתוך אזור זיכרון קוד.
- אם PC (סופר פקודות) נמצא בשימוש, הוא מתקדם ב-1 לכתובת לפני שהוא מועבר למצבר.
- *דוגמא*:

```
                CLR     A
LOC1:           INC     A
                MOVC    A,@A + PC
                RET
Look_up         DB      10H
                DB      20H
                DB      30H
                DB      40H
```

- שגרה הזאת לוקחת ערך שנמצא במצבר שנמצא בין תאים 1 עד 4 שהוגדרו בעזרת פקודה DB.
- לאחר סיום השגרה בתוך המצבר יהיה ערך ACC=20H


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOVX <dest-byte>,<source-byte>

- פקודה מעבירה מידע בין מצבר לבין זיכרון חיצוני.
- ישנם 2 צורות לפקודה הזאת – גודל אזור הזיכרון החיצוני : 8 סיביות או 16 סיביות.
- כאשר משתמשים בצורה 8 סיביות משתמשים באוגר מיוחד EMI0CN כדי לשמור חלק עליון של הכתובת ואוגר R0 או R1 כדי לציין חלק תחתון של הכתובת הגישה.
- *לדוגמא*:

```
MOV     EMI0CN,#10H     \\ הכנסת בית עליון  של הכתובת לתוך אוגר עזר

MOV     R0,#34H         \\ הכנסת בית תחתון של הכתובת לתוך אוגר עזר
MOVX    A,@R0           \\ תוכן של תא זיכרון 1034 חיצוני נכנס לתוך מצבר
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOVX <dest-byte>,<source-byte>

- אפשרות שנייה של פעולה הזאת זה שימוש מצביע לזיכרון באורך 16 סיביות.
- *דוגמא:*

```
MOV   DPTR,#1034H  ;Load DPTR with 16 bit
                   ;address to read (1034H).
MOVX  A,@DPTR      ;Load contents of 1034H
                   ;into ACC.
```

- דוגמא שנמצאת בעמוד הזה משתמשת בכתובת באורך 16 סיביות שנמצאת באוגר DPTR והוא אוגר היחיד שיכול להצביע ישירות לאזור הזיכרון החיצוני.

- ישנה אפשרות לגשת לכל אחד מהאוגרים שמרכיבים את DPTR (DPL ו- DPH ) ולהשתמש בכל אחד מהם בנפרד.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# PUSH Direct

- פקודה הזאת מכניסה למחסנית נתון ומעלה מצביע לראש המחסנית ב-1.
- תוכן של התא של זיכרון הפנימי מעתק לראש המחסנית שנמצא באזור RAM מיועד למחסנית.
- דגלים לא משתנים ולא מושפעים.

- *דוגמא*:

```
PUSH   22H
PUSH   23H
```

- נניח שערך התחלתי של SP שווה ל- 4FH ותוכן של תאים 22H, 23H מכילים ערכים 11H ו- 12H.
לאחר ביצוע תוכנה ערך של מצביע SP=51H וערך של תאים 50H ו- 51H מכילים 11H ו- 12H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# POP Direct

- פקודה קולטת נתון מאזור של מחסנית ומעבירה אותו לאוגר היעד. בנוסף היא מורידה מצביע של מחסנית ב-1. דגלים לא מושפעים ולא משתנים.
- *דוגמא*:

```
POP   DPH
POP   DPL
```

- אם במצב התחלתי SP=51H ותוכן תאי זיכרון 4FH, 50H ו-51H יהיה 30H, 11H ו-12H , אז לאחר סיום הדוגמא ערך של DPTR=1211H וערך של SP=4FH.

```
POP   SP
```

- בהמשך לתוכנית ערך של SP קודם ירד עד 4EH ואז ערך שלו יהיה SP=30H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# XCH A,<byte>

- פקודה הזאת מחליפה ערך שנמצא בתוך מצבר עם ערך שנמצא באזור נתונים. אף דגל לא משתנה ולא מושפע.

- *דוגמא*:

```
XCH    A,@R0
```

- נניח שאוגר R0=2EH, מצבר (11110011) ACC=F3H וערך תא (01110110) 2EH=76H. לאחר ביצוע פקודה ACC=76H וערך תא 2EH=F3H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# XCHD A,@Ri

- פקודה לוקחת חלק תחתון של מצבר (סיביות 0-3) ומחליפה עם אותם סיביות של תא זיכרון שאוגר Ri מצביע עליו.
- סיביות של חלק גדול (4-7) לא משנות את מיקומם.
- אף דגל לא משתנה ולא מושפע.
- דוגמא:

```
XCHD    A,@R0
```

**לפני ביצוע:**

RAM 2EH=F3H (11110011) ,R0=2EH, ACC=76H (01110110)

**אחרי ביצוע:**

RAM 2EH=F6H (11110110) ,ACC=73H (01110011)


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# פעולות עם סיביות ודגלים

- מיקרו בקר C8051F020 מסוגל לבצע פעולות עם סיביות.
- בסה"כ ניתן לבצע פעולות הבאות: *set, clear*, *and, or* , not
- ניתן לבצע גם פעולות קפיצה בהתאם מצב דגלים וסיביות.
- כל הפעולות עובדות לפי גישה ישירה.

| Mnemonic | Description |
|---|---|
| CLR C | Clear C |
| CLR bit | Clear direct bit |
| SETB C | Set C |
| SETB bit | Set direct bit |
| CPL C | Complement c |
| CPL bit | Complement direct bit |
| ANL C,bit | AND bit with C |
| ANL C,/bit | AND NOT bit with C |
| ORL C,bit | OR bit with C |
| ORL C,/bit | OR NOT bit with C |
| MOV C,bit | MOV bit to C |
| MOV bit,C | MOV C to bit |
| JC rel | Jump if C set |
| JNC rel | Jump if C not set |
| JB bit,rel | Jump if specified bit set |
| JNB bit,rel | Jump if specified bit not set |
| JBC bit,rel | if specified bit set then clear it and jump |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# CLR <bit>

- פקודה מנקה (מאפסת) סיבית ספציפית (במידה ויש פגישה עליו.
- אף דגל לא משתנה ולא מושפע. (במידה ולא מדובר על איפוס של דגל עצמו).

- *דוגמא*:

```
CLR    P2.7
```

מצב התחלתי Port 2 ,(11011100) DCH

מצב סופי (01011100) 5CH


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# SETB <bit>

- פקודה הזאת מאתחלת סיבית המבוקשת לערך "1".
- פעולה הזאת עובדת עם כל דגל או סיבית שנתון לגישה לפי הגדרה.
- אף דגל לא מושפע, אלא אם כן פקודה משנה את הדגל באופן מכוון.

- *דוגמא*:

```
SETB   C
SETB   P2.0
```

**נניח שלפני פעולות ערך של פורט 2** 24H (00100100)

- לאחר פקודה ראשונה דגל נשא מאותחל (C שווה ל-1)
- לאחר פקודה שנייה (00100101) Port 2 25H


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# CPL <bit>

- פקודה הזאת הופכת סיבית המבוקשת.
- פעולה הזאת עובדת עם כל דגל או סיבית שנתון לגישה לפי הגדרה.
- אף דגל לא מושפע, אלא אם כן פקודה משנה את הדגל באופן מכוון.

- *דוגמא*:

```
CPL   P2.1
CPL   P2.2
```

- נניח שמצב התחלתי של פורט 2 יהיה 53H (01010011)
- לאחר ביצוע פקודות מצב של פורט 2 יהיה 55H (01010101)


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ANL C, <source-bit>

- עושים AND לוגי בין דגל CARRY עם סיבית כלשהי עם הכנסת תוצאה לתוך דגל CARRY.
- אם סיבית היא 0 אז דגל CARRY מתאפס, אחרת דגל לא משתנה.
- במידה וכותבים סימן "/" ליד סיבית, אז מתכוונים אל ערך הפוך של סיבית, אבל לא משנים אותו.
- אף דגל אחר לא משתנה
- *דוגמאות*:

```
MOV    C,P2.0          ;Load C with input pin
                       ;state of P2.0.
ANL    C,P2.7          ;AND carry flag with
                       ;bit 7 of P2.
MOV    P2.1,C          ;Move C to bit 1 of Port 2.
ANL    C,/OV           ;AND with inverse of OV flag.
```

- במידה ו- P2.0=1, P2.7=0 ודגל OV=0 אז לאחר רצף פעולות נקבל:

P2.1=0, CY=0, OV=0

www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ORL C, <source-bit>

- עושים OR לוגי בין דגל CARRY עם סיבית כלשהי עם הכנסת תוצאה לתוך דגל CARRY.
- אם סיבית היא 0 אז דגל CARRY לא משתנה, אחרת דגל עולה ל-1.
- במידה וכותבים סימן "/" ליד סיבית, אז מתכוונים אל ערך הפוך של סיבית, אבל לא משנים אותו.
- אף דגל אחר לא משתנה

- *דוגמא*:

```
MOV     C,P2.0          ;Load C with input pin
                        ;state of P2.0.
ORL     C,P2.7          ;OR carry flag with
                        ;bit 7 of P2.
MOV     P2.1,C          ;Move C to bit 1 of
                        ;port 2.
ORL     C,/OV           ;OR with inverse of OV
                        ;flag.
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# MOV <dest-bit>,<source-bit>

- פקודה מעתיקה את הסיבית המקור לסיבית היעד.
- אחד הסיביות חייבת להיות דגל CARRY. (לא משנה מקור או יעד).
- אף דגל אחר לא מושפע.

- *דוגמא*:

```
MOV    P2.3,C
MOV    C,P3.3
MOV    P2.0,C
```

- אם נניח ש- P3.3=0 ,P2=C5H (11000101) ודגל CARRY CY=1, אז לאחר רצף פעולות האלה (11001100) P2=CCH , CY=0 .


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# תווית JC

- פקודה הזאת היא פקודת קפיצה שמבצעת אותה במידה ודגל CARRY נמצא במצב "1". אחרת תוכנה ממשיכה בדרכה.
- אף דגל לא משתנה.

- *דוגמא*:

```
    CLR   C
    SUBB  A,R0
    JC    ARRAY1
    MOV   A,#20H
ARRAY1:
```

- דגל carry מתאפס לאחר פקודה ראשונה, לאחר פקודת חיסור אם A יותר קטן מערך שנמצא בתוך אוגר R0 דגל carry עולה ל-1 ופקודה JC מעבירה את PC לתווית ARRAY1 , אחרת אנו ממשיכים לבצע פקודה MOV, מעתיקים ערך **20H** לתוך המצבר ואז ממשיכים את התוכנית.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# תווית JNC

- פקודה הזאת היא פקודת קפיצה שמבצעת אותה במידה ודגל CARRY לא נמצא במצב "1". אחרת תוכנה ממשיכה בדרכה.
- אף דגל לא משתנה.

- *דוגמא*:

```
    CLR   C
    SUBB  A,R0
    JNC   ARRAY2
    MOV   A,#20H
ARRAY2:
```

- דגל carry מתאפס לאחר פקודה ראשונה, לאחר פקודת חיסור אם A יותר גדול או שווה לערך שנמצא בתוך אוגר R0 דגל carry עולה ל-1 ופקודה JC מעבירה את PC לתווית ARRAY2 , אחרת אנו ממשיכים לבצע פקודה MOV, מעתיקים ערך **20H** לתוך המצבר ואז ממשיכים את התוכנית.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JB <סיבית>,תווית

- פקודה הזאת היא פקודת קפיצה שמבצעת אותה במידה וסיבית הנבדקת נמצאת במצב "1". אחרת תוכנה ממשיכה בדרכה.
- אף דגל לא משתנה.

- *דוגמא*:

```
JB      ACC.7,ARRAY1
JB      P1.2,ARRAY2
```

- נניח שמצבר מכיל 01001010 ופורט 1 מכיל Port 1=57H (0101011) . פקודה ראשונה לא מבצעת קפיצה ופקודה שנייה כן מבצעת ומעבירה אותנו לפי תווית ARRAY2.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JNB <bit>,rel

- פקודה הזאת היא פקודת קפיצה שמבצעת אותה במידה וסיבית הנבדקת נמצאת במצב "0". אחרת תוכנה ממשיכה בדרכה.
- אף דגל לא משתנה.

- *דוגמא*:

```
JNB     ACC.6,ARRAY1
JNB     P1.3,ARRAY2
```

- נניח שמצבר מכיל 01001010 ופורט 1 מכיל Port 1=57H (01010111) . פקודה ראשונה לא מבצעת קפיצה ופקודה שנייה כן מבצעת ומעבירה אותנו לפי תווית ARRAY2.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JBC <סיבית>,תווית

- פקודה הזאת היא פקודת קפיצה שמבצעת אותה במידה וסיבית הנבדקת נמצאת במצב "1" , **<u>היא גם כן מאפסת את הסיבית לאחר בדיקה.</u>** אחרת תוכנה ממשיכה בדרכה ללא שינוי סיבית.
- אף דגל לא משתנה.

- *דוגמא*:

```
JBC     P1.3,ARRAY1
JBC     P1.2,ARRAY2
```

- נניח שמצב של פורט P1=56H (01010110). פקודה ראשונה לא משנה כלום וממשיכה לפקודה שנייה. לאחר פקודה שנייה תוכנה קופצת לתווית ARRAY2 וערך של פורט P1 ישתנה לערך 52H (01010010).


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# Program Branching Instructions פקודות קפיצה

- בפקודות האלה אנו משתמשים כדי לבצע סדר פעולות רצוי וקפיצות בין קטע קוד לקטע קוד אחר, לקריאת שגרות וחזרה מפסיקות.
- ישנם פקודות שמבצעות קפיצות תמיד או בהתאם לתנאים מסוימים.

| Mnemonic | Description |
|---|---|
| ACALL addr11 | קריאה לתת שגרה (עד 11 סיביות) Absolute subroutine call |
| LCALL addr16 | קריאה לתת שגרה (עד 16 סיביות) Long subroutine call |
| RET | חזרה משגרה Return from subroutine |
| RETI | חזרה מפסיקה Return from interrupt |
| AJMP addr11 | קפיצה עד 11 סיביות Absolute jump |
| LJMP addr16 | קפיצה עד 16 סיביות Long jump |
| SJMP rel | קפיצה קצרה Short jump |
| JMP @A+DPTR | קפיצה של גישה עוקבת Jump indirect |
| JZ rel | קפיצה במידה ומצבר ריק Jump if A=0 |
| JNZ rel | קפיצה במידה ומצבר לא ריק Jump if A NOT=0 |
| CJNE A,direct,rel | Compare and Jump if Not Equal<br>צורות שונות של תשווה ותקפוץ במידה אם לא שווה |
| CJNE A,#data,rel | |
| CJNE Rn,#data,rel | |
| CJNE @Ri,#data,rel | |
| DJNZ Rn,rel | Decrement and Jump if Not Zero<br>תחסיר 1 ותקפוץ אם לא שווה ל-0. |
| DJNZ direct,rel | |
| NOP | פעולה שלט מבצעת כלום. No Operation |


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# ACALL addr11

- פקודה הזאת מבצעת קריאה ללא תנאי של תת שגרה לפי כתובת שלה.
- פקוד מעלה את PC ב-2, דוחפת ערך של PC (16 סיביות) לתוך מחסנית (קודם בית תחתון ואז בית עליון) ומעלה SP ב-2.
- הערך החדש נכנס לPC ותוכנה ממשיכה להתבצע מהנקודה הזאת.
- תת שגרה חייבת להיות באותו אזור של זיכרון (2 kB block).
- אף דגל לא משתנה.

- *דוגמא*:

```
ACALL LOC_SUB
```

- מצב נוכחי: SP=07H, LOC_SUB נמצא במקום 0567H, וערך של PC שווה ל-0230H.
- מצב לאחר ביצוע פקודה: מחסנית מקומות 08H ו- 09H מכילה 32H ו- 02H, SP=09H וערך חדש של PC=0567H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# LCALL addr16

- פקודה הזאת מבצעת קריאה ללא תנאי של תת שגרה לפי כתובת שלה.
- פקוד מעלה את PC ב-3, דוחפת ערך של PC (16 סיביות) לתוך מחסנית (קודם בית תחתון ואז בית עליון) ומעלה SP ב-2.
- הערך החדש נכנס לPC ותוכנה ממשיכה להתבצע מהנקודה הזאת.
- תת שגרה יכולה להיות בכל אזור של זיכרון (64 kB block).
- אף דגל לא משתנה.

- *דוגמא*:

```
LCALL LOC_SUB
```

- מצב נוכחי: SP=07H, LOC_SUB נמצא במקום 2034H, וערך של PC שווה ל-0230H.
- מצב לאחר ביצוע פקודה: מחסנית מקומות 08H ו- 09H מכילה 33H ו- 02H, SP=09H וערך חדש של PC=2034H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RET

- פקודה מחזירה את התוכנה מתת שגרה לתוכנה ראשית.

- פקודה מעבירה בית עליון ובית תחתון מהמחסנית לאוגר PC ומורידה 2 מאוגר SP. (2 ביתים האלה הם בעצם כתובת חזרה משגרה לתוכנה ראשית ומוחקת את הנתונים האלה מהמחסנית).
- פקודה הזאת מסיימת את הפעלת השגרה שמפעילים אותה בעזרת פקודה "call" .
- אף דגל לא משתנה.

- נניח שאוגר SP=0BH וכתובות בראש המחסנית (חלק של RAM) 0AH ו-0BH מכילות ערכים 30H ו-02H .
לאחר ביצוע פקודה SP=09H ותוכנה ממשיכה לרוץ מכתובת 0230H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# RETI

- פקודה מחזירה את התוכנה מפסיקה לתוכנה ראשית.
- פקודה מעבירה בית עליון ובית תחתון מהמחסנית לאוגר PC ומורידה 2 מאוגר SP. (2 ביתים האלה הם בעצם כתובת חזרה משגרה לתוכנה ראשית ומוחקת את הנתונים האלה מהמחסנית).
- פקודה הזאת מסיימת את הפעלת הפסיקה ומחזירה לוגיקה של פסיקות למצב התחלתי כדי שיהיה אפשרות לקלוט פסיקות נוספות.
- נניח שאוגר SP=0BH וכתובות בראש המחסנית (חלק של RAM) 0AH ו-0BH מכילות ערכים 30H ו-02H .
  לאחר ביצוע פקודה SP=09H ותוכנה ממשיכה לרוץ מכתובת 0230H.
- אוגר PSW לא מתחדש באופן אוטומטי.
- לאחר ביצוע פקודה תוכנה ממשיכה לרוץ מאותו מקום איפה שעצרנו אותה עקב פסיקה.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# AJMP addr11

- פקודה AJMP מאפשרת קפיצה באותו בלוק זיכרון – לפי כתובת של 11 סיביות.
- הכתובת היעד חייבת להיות באותו בלוק זיכרון במרחק של לא יותר מ-2kB.
- אף דגל לא משתנה.

- *דוגמא*:

```
AJMP   NEAR
```

- נניח שכרגע PC מכיל ערך של 0120H ותווית NEAR נמצא בכתובת 0234H. לאחר ביצוע פקודה ערך חדש של PC יהיה 0234H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# LJMP addr16

- פקודה LJMP מאפשרת קפיצה באותו בלוק זיכרון – לפי כתובת של 16 סיביות.
- הכתובת היעד יכולה להיות בכל מרחב זיכרון במרחק של לא יותר מ-64kB.
- אף דגל לא משתנה.

- *דוגמא*:

```
LJMP   FAR
```

- נניח שכרגע PC מכיל ערך של 0120H ותווית FAR נמצא בכתובת 3234H. לאחר ביצוע פקודה ערך חדש של PC יהיה 3234H.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# SJMP rel

- פקודה לקפיצה קצרה. היא מוסיפה ל-PC 2 ואחר כך מוסיפה לו ערך '*rel*' שיכול להיות עד 8 סיביות (signed 8-bit).
- זאת תהיה כתובת חדשה שנמצאת ב-PC ולשם תוכנה תתקדם בשלב הבא.
- בעזרת פקודה הזאת ניתן להגיע לכתובות במרחק עד 127 בית קדימה או 128 בית אחורה.

- *דוגמא*:

```
        SJMP   RELSRT
        ...
        ...
        ...
RELSRT:
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JMP @A + DPTR

- פקודה הזאת מוסיפה ערך 8 סיביות שנמצא בתו אוגר ACC (מצבר) לאוגר 16 סיביות ותוצאה נכנסת לתוך אוגר PC (זה אומר שתוכנה קופצת למיקום חדש).
- אף אחד מהאוגרים ( DPTR,ACC) לא משתנה
- אף דגל לא משתנה.

- *דוגמא*:

```
        MOV    DPTR, #LOOK_TBL
        JMP    @A + DPTR
LOOK_TBL:
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JZ rel

- פקודה הזאת מבצעת קפיצה לתווית rel בתנאי שתוכן האוגר ACC שווה ל-0, אחרת תוכנה ממשיכה לפקודה הבאה ברצף.
- אוגר ACC לא משתנה ואף דגל לא משתנה.

- *דוגמא*:

```
SUBB A,#20H
JZ   LABEL1
DEC  A
```

- במידה ואוגר ACC מכיל ערך 20H ודגל CARRY שווה ל-0 CY=0, לאחר פקודת חיסור תוכנה תקפוץ לתווית LABEL1, אחרת תוכנה תמשיך לפקודה DEC A .


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# JNZ rel

- פקודה הזאת מבצעת קפיצה לתווית rel בתנאי שתוכן האוגר ACC שונה מ-0, אחרת תוכנה ממשיכה לפקודה הבאה ברצף.
- אוגר ACC לא משתנה ואף דגל לא משתנה.

- *דוגמא*:

```
DEC    A
JNZ    LABEL2
MOV    RO, A
```

- במידה ואוגר ACC מכיל ערך שונה מ-1, לאחר פקודת חיסור 1 תוכנה תקפוץ לתווית LABEL2, אחרת תוכנה תמשיך לפקודה MOV.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# CJNE <dest-byte>,<source-byte>,rel

- פקודה משווה ערך שנמצא ב-*dest-byte* עם ערך שנמצא ב-*source-byte*. *אם הם שונים היא מבצעת קפיצה לתווית* rel.

- אם *dest-byte קטן מ-source-byte* *דגל* carry עולה ל-"1" אחרת דגל מתאפס. (שני ערכים הם לא מסומנים).

- אף אופרנד לא משתנה.

- *דוגמא*:

```
CJNE   R3,#50H,NEQU
       … …
NEQU:  JC     LOC1
LOC1:  … …
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# DJNZ <byte>,<rel-addr>

- פקודה הזאת "decrement jump not zero" אפשר לתרגם תחסיר אחד ותקפוץ אם תוצאה שונה מ-0.
- היא מחסירה 1 מהתוכן של זיכרון ואם התוצאה שונה מ-0 תוכנית קופצת לפי תווית.
- במידה ותוכן התא שווה ל-0 ערך חדש שיהיה בתוך התא יהיה FFH.
- אף דגל לא מושפע.

- *דוגמא*:

```
DJNZ 20H,LOC1
DJNZ 30H,LOC2
DJNZ 40H,LOC3
```

- נניח שתאי זיכרון20H, 30H , 40H, מכילים ערכים 1,2,3 . לאחר הורדת 1 מתא 20H נקבל שם ערך 0 ותוכנית תמשיך לפקודה הבאה. אם נוריד 1 מתא 30H נקבל תוכן 1 (שהוא שונה מ-0) ואז תוכנית תקפוץ לפי תווית LOC2.


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

# NOP

- פקודה שלא מבצעת שום פעולה.
- פקודה מתבצעת בפולס שעון בודד.
- משתמשים בפקודה כדי ליצור השהייה מדויקת ללא שימוש בטיימרים.
- נניח שתדר הגביש 12 MHz, זה אומר שתדר של פולס פנימי הוא 12 MHz/12 ושווה ל- 1 MHz. זמן פעולה הבסיסית ביותר הוא 1 מיקרו שנייה. בדוגמא הזאת אנו בונים גל ריבועי ברגל P1.2 – "0" 4 מיקרו שניות, "1" 4 מיקרו שניות. 125 kHz.

- *דוגמא*:

```
home:  CLR    P1.2
       NOP
       NOP
       NOP
       SETB   P1.2
       NOP
       sjmp home
```


www.elecstudy.org markalex012@gmail.com טסליצקי מרק

**www.silabs.com/MCU**