Lecture 6

Assembler Directives הוראות של אסמבלר

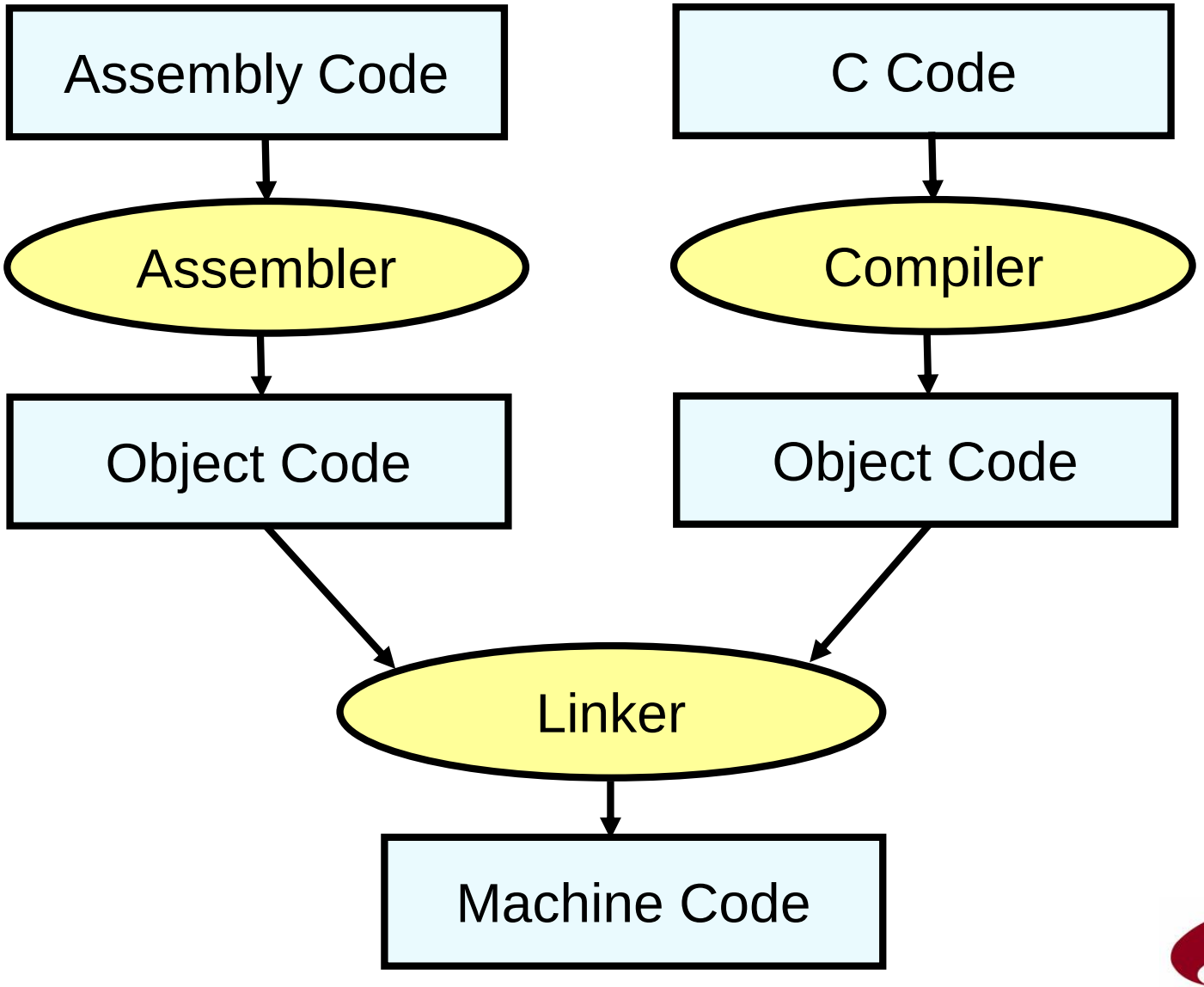
www.elecstudy.org markalex012@gmail.com טסליצקי מרק

Assembler Directives הוראות של אסמבלר

- כיצד בונים קוד הרצה.
- הוראות של אסמבלר – מבוא.
- בקרת סגמנטים:
  - סגמנטים כלליים (SEGMENT, RSEG)
  - סגמנטים מוחלטים (CSEG, DSEG and XSEG)
- בקרת כתובות.
  - ORG, USING, END
- הגדרת סימנים:
  - EQU, SET, CODE, DATA, IDATA, XDATA
- ניהול והגדרת זיכרון:
  - DB, DW, DD, DS
- דוגמא.

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

Code Generation Flow כיצד בונים קוד הרצה.



הוראות של אסמבלר – מבוא.

- הוראות של אסמבלר הן לא פקודות של שפת "סמבלי" הן הוראות שלא מתורגמות לשפת מכונה, אלא הוראות לניהול זיכרון.
  - הוראות האלה נמצאות בתוך תוכנה שכתובה בשפת "אסמבלי", אבל הן לא נמצאות בתוך רשימת הפונקציות כמו ADD, SBB וכו'.
  - ניתן להשתמש בהם כדי לדיר ערך של משתנים, לשמור מיקום בזיכרון, לשים תונה במקום מיוחד וכו'.

- לכל תוכנת הרצה יש הוראות משלו. אנו נשתמש בהוראות שתוכנת **Keil A51 Assembler** מאפשרת.
- ניתן לחלק את כל ההוראות ל-4 קבוצות הבאות:
  - Segment control בקרת סגמנט
  - Address control בקרת כתובות
  - Symbol definition הגדרת סימנים
  - Memory initialization/reservation אתחול ושמירת מקום לזיכרון

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

Segment Control בקרת סגמנטים

- בבקרים x51 בלוק של קוד או של נתונים, בדרך כלל, מוגדר בתור סגמנט.
  - לדוגמא:
    - הגדרת פונקציה מוגדרת בתוך אזור פקודות (code memory)
    - מערך מוגדר בתוך אזור נתונים (data memory)

- מבחינת מיקום בזיכרון ניתן לחלק כל הסגמנטים לשני סוגים:
  - *Generic* or *relocatable* – *סגמנטים שניתנים להזזה.*
  - *Absolute* – *סגמנטים שלא ניתנים להזזה.*

- ישנם 5 סוגי סגמנטים: **five memory classes**
  - CODE, DATA, IDATA, XDATA, BDATA/BIT

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

סגמנטים שניתנים להזזה

- כדי ליצור סגמנט כזה משתמשים בהוראה "SEGMENT ".
- הכתובת של סגמנט מסוג זה מוגדר באופן אוטומטי ע"י "מקשר" (linker) ולא ע"י משתמש.
- צורת ההגדרה היא:

<Symbol> SEGMENT *<segment_memory_class>*

- **לדוגמא:**

MYDATA SEGMENT DATA

- הוראה הבאה מגדירה סגמנט שניתן להזזה בשם MYDATA שנמצא בתוך אזור הזיכרון DATA.
  - ברגע שהגדנו את הסגמנט, צעד הבא זה לשייך את הסגמנט בעזרת הוראה RSEG לפי דוגמא הבא:

RSEG MYDATA

- ברגע שפקודה הזאת מופיעה סגמנט MYDATA מתחיל להיות פעיל עד שRSEG מופיע פעם נוספת וסגמנט הבא מתחיל להיות פעיל.

Absolute Segment סגמנט מוחלט

- סגמנטים מוחלטים – זה אומר שהם תופסים מקום קבוע בזיכרון. מדובר כאן על סגמנטים CSEG, DSEG ו-XSEG . (סגמנטים של קוד, נתונים ונתונם חיצוניים).
- מיקום סופי של הסגמנטים ידוע בזמן ביצוע קומפילציה.
- ניתן להגדיר הוראות בצורה הבאה:

```
CSEG   AT <address>  ; defines an absolute code segment
DSEG   AT <address>  ; defines an absolute data segment
XSEG   AT <address>  ; defines an absolute external data segment
```

- *דוגמא*:

```
CSEG   AT   0300H     ;select code segment and set
                      ;the starting address at 0300H
DSEG   AT   0400H     ;select data segment and set
                      ;the starting address at 0400H
```

Address Control—ORG בקרת כתובות

- צורת ההוראה היא:

```
ORG     <ביטוי\מיקום>
```

- משתמשים בהוראת ORG כדי להגדיר מונה פקודות בסגמנט הנוכחי לכתובת שמוגדרת ע"י **<ביטוי\מיקום>**
- כמובן זה לא משנה את הכתובת של סגמנט – היא נשארת כמו שהגדרנו בתחילת תוכנית.

- **דוגמא:**

```
ORG     80H     ;Set location counter to 80H
```

- ניתן להשתמש בהוראת ORG לא רק להגדרת קוד, אלא גם להגדרת מיקום של נתונים.
- לדוגמא: כדי להגדיר מיקום של שניות ודקות באזור זיכרון ניתן לכתוב רצף הבא:

```
DSEG            ;data segment
ORG 30H
SECONDS:        DS      1
MINUTES:        DS      1
```

Address Control—END הוראת סיום.

- הוראה הזאת מסיימת תוכנית.

  END

- הוראה אומרת לאסמבלר עד איזה קטע לבדוק את התוכנית ולתרגם אותה לשפת מכונה.
  - כל מה שכתוב מתחת להוראה הזאת לא נבדק ע"י אסמבלר.

- הוראה הזאת היא הוראת חובה בסוף של כל קובץ שכתוב בשפת אסמבלי.
  - במידה והוראה הזאת לא מופיע אסמבלר תעצור את הקומפילציה ותרשום הודעת שגיאה.

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

Symbol Definition הגדרת סימנים

- ניתן לתת שם משלנו לאוגר או לביטוי.
  - ביטוי יכול להיות מספר קבוע, מצביע לזיכרון או שם כלשהו.
- לפעמים יותר קל לתת שמות לוגיות לאוגרים או ביטויים כדי שתוכנה תהיה "קריאה יותר.

- יתרון נוסף שבהגדרה כזאת אין צורך לשנות ערכים במשך תוכנית, אלא רק בתחילת תוכנית. נגיד חיברנו חיישן לפורט P0.3 ונתנו לו שם TTT, לאחר זמן רב החלטנו לחבר אותו לפורט אחר, מספיק לשנות שיוך בתחילת תוכנית ואין צורך לרוץ בקוד ולחפש כל הנקודות איפה עוד לשנות.

Symbol Definition—EQU, SET הוראות

- מגדירים אותם בצורה הבאה:

Symbol	EQU	<*ביטוי*>
Symbol	EQU	<*אוגר*>
Symbol	SET	<*ביטוי*>
Symbol	SET	<*אוגר*>

- הוראה הזאת דומה למקרו "#define" של שפת C.
- *ביטוי יכול להכיל אופרטורים מתמטיים פשוטים כמו:* '+', '-', ' * ', '/', MOD.

- *אוגרים מכילים:* A, R0, R1, R2, R3, R4, R5, R6 and R7

Symbol Definition—EQU, SET הוראות

- *דוגמאות*:

```
COUNT       EQU   R3      ; COUNT במקום אוגר R3
TOTAL       EQU   200     ;TOTAL שווה ל-200
AVERG       SET   TOTAL/5
TABLE       EQU   10
VALUE       SET   TABLE*TABLE
```

אזורי זיכרון—CODE, DATA, IDATA, XDATA

- הגדרות הבאות משייכות סגמנטים למיקום שלהם במרחב כתובות.

Symbol	BIT	<*bit_address*>
Symbol	CODE	<*code_address*>
Symbol	DATA	<*data_address*>
Symbol	IDATA	<*idata_address*>
Symbol	XDATA	<*xdata_address*>

bit_address	The bit address which is available from bit-addressable location 00H through 7FH as an offset from byte location 20H
code_address	The code address ranging from 0000H to 0FFFFH
data_address	The address is from 00H to 7FH (internal data memory) and the special function register address from 80H to 0FFH
idata_address	The address is ranging from 00H to 0FFH
xdata_address	The external data space ranging from 0000H to 0FFFFH

Symbol Definition—CODE, DATA, IDATA, XDATA

- *דוגמאות*:

```
Act_bit BIT  2EH   ;Use bit location 2EH
                   ;as Act_bit
Port2   DATA A0H   ;A special function
                   ;register, P2
```

ניהול זיכרון Memory Initialization/Reservation

- ניתן לשייך, להגדיר משתנים בגודל בית (DB), מילה (DW) או מילה כפולה (DD).
- הוראות האלה יעשו התחול וישמרו מקום למשתנים על חשבון מיקום חופשי באזור קוד ולא באזור נתונים.

- ישנה אפשרות לתפיסת מקום בזיכרון ללא התחול בעזרת פקודה DS.

- הוראה הזאת תשמור מיקום בזיכרון באותו סגמנט איפה שהיא מופיעה.

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

הוראה DB (Define Byte)

- הוראה DB מאתחלת בית בתוך אזור קוד. (**code memory**)
- הוראה נראת כך:

<*label***>**: DB <*expression*>, <*expression*>, …

label – שם של משתנה איפה שנתונים יהיו.

expression - ערך בגודל בית – תו או מספר עד 255 (8 סיביות).

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

DB (Define Byte)

- *דוגמא*:

```
        CSEG AT   200H
MSG:    DB 'Please enter your password', 0
ARRAY:  DB 10H,20H,30H,40H,50H
```

- בקוד סגמנט החל מכתובת 200H נגדיר MSG שיכיל רצף תווים בצורה הבא: [200H]=50H, [201H]=6CH וכו'.

- הוראה DB יכולה להיות מוגדרת רק באזור קוד. (code segment)
  - במידה והיא מוגדרת באזור אחר אסמבלר תעצור את הקומפילציה ותרשום ודעת שגיאה.

DW (Define Word)

- הוראה DW מאתחלת מילה (2 בית) בתוך אזור קוד. (**code memory**)
- הוראה נראת כך :

<*label***>**: DW <*expression*>, <*expression*>, …

- *דוגמא*:

```
;2 words allocated
CNTVAL:  DW    1025H, 2340H
;10 values of 1234H starting from location XLOC
XLOC:    DW    10 DUP (1234H)
```

- אופרטור DUP מכפיל תפיסת זיכרון לפי מספר שרשום לפני. בדוגמא שנייה יש 10 תאים זיכרון בגודל 2 בית וכל אחד יכיל ערך **1234H**.
- הוראה DW יכולה להיות מוגדרת רק באזור קוד. (code segment)
  - במידה והיא מוגדרת באזור אחר אסמבלר תעצור את הקומפילציה ותרשום ודעת שגיאה.

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

DD (Define Double Word)

- הוראה DD מאתחלת מילה כפולה (4 בית) בתוך אזור קוד. (**code memory)**
- הוראה נראת כך :

<*label***>**: DD <*expression*>, <*expression*>, …

- *דוגמא*:

```
ADDR:   DD   820056EFH, 10203040H
EMPT:   DD   3 DUP ( 0 )
```

- כמו הוראות DB ו- DW, הוראה DD יכולה להיות מוגדרת רק באזור קוד. (code segment)
  - במידה והיא מוגדרת באזור אחר אסמבלר תעצור את הקומפילציה ותרשום ודעת שגיאה.

DS (Define Storage) הוראה לתפיסת מקום ללא התחול

- הוראה DS תופסת מספר בית בסגמנט נוכחי ללא ערך התחלתי.
- ניתן להגדיר מקום בתוך סגמנטים הבאים CSEG, ISEG, DSEG או XSEG.
- הוראה נראת כך :

<label>: DS *<expression>*

- ביטוי יכול להכיל רק מספר

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

DS (Define Storage) הוראה לתפיסת מקום ללא התחול

- *דוגמא*:

```
XSEG  AT 1000H      ;select memory block from
                    ;external memory, starting
                    ;address from 1000H
Input:      DS   16     ; reserve 16 bytes
Wavetyp:    DS   1      ; reserve 1 byte
```

- בדוגמא הזאת מפתח צריך לא יותר מ16 ערכים בתוך זיכרון על מצביע INPUT כדי לא לדרוס את היתר מקומות.

- חשוב להבין שמקום נתפס ללא ערכים התחלתיים.

דוגמא לתוכנית הכוללת

```
;-----------------------------------------------------------
$include (c8051f020.inc)  ;Include register definition file
;-----------------------------------------------------------
; EQUATES
;-----------------------------------------------------------
CR EQU   0DH            ;Set CR (carriage return) to 0DH
;-----------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-----------------------------------------------------------
                        ; Reset Vector
   CSEG AT 0            ; Jump to the start of code at
   LJMP Main            ; the reset vector
                        ; Timer 4 Overflow Vector
   ORG  83h             ; Jump to the start of code at
   LJMP TIMER4INT                ; the Timer4 Interrupt vector
;-----------------------------------------------------------
; DATA SEGMENT
;-----------------------------------------------------------
MYDATA   SEGMENT  DATA
         RSEG     MYDATA  ; Switch to this data segment.
         ORG      30h
Input:   DS       16
temp:    DS       1
```

דוגמא לתוכנית הכוללת המשך.

```
;------------------------------------------------------------
; CODE SEGMENT
;------------------------------------------------------------
MYCODE   SEGMENT              CODE
         RSEG       MYCODE              ; Switch to this code segment
         USING      0          ; Specify register bank
                               ; for main code.
Main:               ; Insert Main Routine of program here
                    ; … …
                    ; … …
;------------------------------------------------------------
; Timer 4 Interrupt Service Routine
;------------------------------------------------------------
TIMER4INT:          ; Insert Timer 4 ISR here
                    ; … …
                    ; … …
                    RETI
;------------------------------------------------------------
; Global Constant
;------------------------------------------------------------
Rdm_Num_Table:
DB   05eh, 0f0h, 051h, 0c9h, 0aeh, 020h, 087h, 080h
DB   092h, 01ch, 079h, 075h, 025h, 07ch, 02bh, 047h
;------------------------------------------------------------
; End of file.
END
```

www.silabs.com/MCU