

תוכנית דוגמא hellola.asm

תוכנית הדוגמא הראשונה שלנו תמחיש חלק משמעותי מהמבנה של תוכניות האסמבלי לפי השיטה שנעבוד בהם בשלב זה - תוכניות אסמבלי טהורות תוך שימוש בהנחיות הסגמנטים המקוצרות. בשיטה הזו האסמבלר פורש בשבילכם באופן אוטומטי הגדרות שונות ופותר אתכם מזה, וזה נוח בשלב המוקדם הזה.

- תזכורת: כל תוכנית הדוגמא כתובות בצורה כזו שכל מילה על טהרת האותיות הגדולות הם מילים שמורות בשפת האסמבלי. כל מילה שמכילה גם אותיות קטנות הם בחזקת מזהים שיכולים להיות מוחלפים בכל מילה אחרת.

- הנחיות לאסמבלר: מדובר בשורות בתוכנית שמגדירות לאסמבלר איך לפרש שורות פקודה שמתחתיין אך אינם משתקפים ישירות בקובץ הבינארי.

- בכל שורה מה שנכתב לאחר ה-" ;" הינו הערה. האסמבלר פשוט יתעלם מכל תוכן שיש שם עד לשורה הבאה.

- ההנחיה

.MODEL SMALL

אומרת לאסמבלר לבחור ערכים מסוימים מבין אפשרויות שונות שיש. לא ניכנס לכך בשלב זה.

- ההנחיה

.STACK 100h

מנחה את האסמבלר להגדיר מחסנית בגודל 100 הקסדצימל (256 עשירוני) בתים. למה זה נחוץ נלמד בהמשך הקורס. הגדרה שקולה לחלוטין תהיה:

.STACK 256

ככל הקשור לקבועים, אם התו האחרון בקבוע הוא H או D או B הקבוע

מפורש כהקסהדצימלי, עשרוני או בינארי בהתאמה. אם הוא על טהרת המספרים הוא יפורש כעשרוני. מספר הקסה שמתחיל באות (כמו 0B800h) חייב להכתב עם אפס מוביל (0B800h כדוגמא) כי אחרת האסמבלר ינסה לפרש את המספר כשם של משתנה.

- ההנחיה

.DATA

אומר שמה שמחואר להלן יהיה תאור של שטח מידע - משתנים במושגים של שפת תכנות נוסח C. גם את התמונה המלאה לזה נראה מאוחר יותר.

- שורת הפקודה

```
DisplayString DB 'Hello World!',13,10,'$'
```

הינה הגדרת משתנה בשם DisplayString שהוא למעשה מערך של תוים (בית בודד כל אחד).

הסיבה שמדובר במערך של בתים נקבעת על ידי ההנחיה DB. האפשרויות הם:

DB - בית אחד

DW - 2 בתים

DD - 4 בתים

DF, DP - 6 בתים

DQ - 8 בתים

DT - 10 בתים

באופן עקרוני איברים במערך מוגדרים בין פסיקים (...3,5,4,6...) אבל במחרזות ניתן להגדיר בין פסיקים ('Hello') שקול ל-'s','l','l','e','H'. האסמבלר מקצה מקום לקבועים הללו ודואג שיאותחלו במה שהמתכנת מצין.

הערכים 13,10 הם "עבור לשורה הבאה". ה-\$' הוא לצורכי סיום הדפסה - עוד נדבר על כך.

- ההנחיה

.CODE

מנחה את האסמבלי לכך שמה שמתואר להלן הוא החלק הביצועי של התוכנית.

- השורה

Begin:

הוא סמן (label) הנותן שם לפקודה. במקרה הזה אנחנו עושים זאת בכדי לסמן לאסמבלר מי הפקודה הראשונה לביצוע של התוכנית הזאת. הנקודה שבו מוכרות הפקודה הזו כראשונה לביצוע היא בהנחיה

END Begin

ה-END מנחה את האסמבלר שזהו סוף התוכנית ובמקרה הזה גם מציין ש-Begin הוא הפקודה הראשונה לביצוע של התוכנית.

- הפקודות

MOV AX,@DATA

MOV DS,AX

גורמים לאוגר המיוחד DS להצביע על שטח המשתנים של התוכנית (מה משמעות הדבר ולמה זה נחוץ נעמוד בהמשך).

- הפקודה INT 21h וקלט פלט

בתוכנית אסמבלי ביצוע קלט/פלט הוא לכאורה עניין טכני מאד הכרוך

בידעה מדויקת של מנגנון הקלט/פלט של המחשב אולם יש דרך להתחמק מכך וזה להסתמך על רוטינות קיימות במחשב. במקרה הזה אנחנו מסתמכים על מערכת ההכעלה DOS. INT 21h היא פניה לספריית רוטינות של DOS, הרוטינה המדויקת נקבעת לפי הערך של AH כרגע הקריאה. אנחנו מסתמכים על הרוטינות:

INT 21h, AH = 9

"הדפס למסך תוים מהנקודה DS:DX עד שתתקל בתו '\$".

INT 21h, AH = 4Ch

"סיים ריצה והחזור שליטה ל-DOS".

הפקודה INT היא אחת מפקודות ההסתעפות (שמשנות את הפקודה הבאה לביצוע). היא פקודת הסתעפות די מיוחדת במספר מובנים, בין השאר שהיא מציגת איכשוא איך לחזור לתוכנית. INT 21h היא הסתעפות לשטח זכרון שמור ל-DOS שבו הוא מאכטן רוטינות קלט/פלט שלו. הרוטינות הללו משמשים את שורת הפקודה אך עומדות גם לרשות תוכנית אפליקציה.

בתוכנית אסמבלי עצירת התוכנית היא פקודה שהתוכנית חייבת לבצע אותה (שום דבר בתוכנית לא תבצע את זה אוטומטית). אם לא בצע את "פקודה החזרה" הזו, התוכנית תמשיך לקרוא זכרון ולנסות לפרש את התוכן כתאור של פקודות מכוונה ולנסות לבצע אותם, דבר שבמקרה הטוב יתקע את התוכנית.

- הפקודה

MOV DX,OFFSET DisplayString

מציב ל-DX חלק מהכתובת של המשתנה DisplayString (ל-DS כבר דאגנו בשתי הפקודות הראשונות של התוכנית). זו איננה הדרך היחידה (או אפילו העיקרית) לחישוב כתובות, אנחנו נלמד על הפקודה LEA כשלב יותר מאוחר.

אם נניח ש- Var1 הוא שם של משתנה (2 בתים נניח) אז צריך להבדיל

~~MOV AX,OFFSET Var1~~

שמציבה ל-AX את הכתובת של Var1 לבין

MOV AX,Var1

המציבה ל-AX את התוכן של Var1. זה בערך כמו ההכרז לבין $x = &y$; x = y; בשפת C.

- הידור התוכנית

בשלב הזה, אחרי שנקליד תוכנית נוסח hellola.asm כאמצעות תוכנת עריכה (editor) נהרגם אותם לקובץ exe בשני שלבים, תוך יצירת קובץ ביניים עם סיומת obj:

tasm hellola.asm

-

tlink hellola.obj

ההרצה עצמה תהיה הרצת הקובץ hellola.exe.

במידה ויש שגיאות ה-tasm יודיע על כך ולא ייווצר קובץ ה-obj.

אם אנחנו רוצים להשתמש ב-Turbo Debugger בכדי לאתר שגיאות נריץ:

tasm /zi hellola.asm

tlink /v hellola.obj

td hellola.exe

```

;
; hellola.asm - send message 'Hello World!' to the screen.
;
.MODEL SMALL
.STACK 100h
.DATA
DisplayString DB 'Hello World!',13,10,'$'
;
.CODE
Begin:
MOV AX,@DATA      ; DS can be written to only through a register
MOV DS,AX         ; Set DS to point to data segment
MOV AH,9          ; Set print option for int 21h
MOV DX,OFFSET DisplayString ; Set DS:DX to point to DisplayString
INT 21h           ; Print DisplayString
;
MOV AH,4Ch        ; Set terminate option for int 21h
INT 21h           ; Return to DOS (terminate program)
END Begin

```

```

E:\>tasm hellola.asm
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

```

```

Assembling file:  hellola.asm
Error messages:   None
Warning messages: None
Passes:          1
Remaining memory: 389k

```

```

E:\>tlink hellola.obj
Turbo Link Version 5.0 Copyright (c) 1992 Borland International

```

```

E:\>hellola.exe
Hello World!

```

```

E:\>

```

תוכנית דוגמא hello4.asm

זוהי תוכנית דוגמה עקרונית ל-hello4.asm אך שואלת את המשתמש אם השעה היא אחרי 12 בצהריים ולפי תגובת המשתמש מדפיסה או את "Good afternoon world!" או "Good morning world!". אם התגובת המשתמש היא 'y' או 'Y' יודפס "Good afternoon world!" ובמקרה של כל תגובה אחרת יודפס "Good morning world!". על מנת שהתוכנית לא תבדיל בין 'y' ל-'Y' היא חייבת לבצע את שתי ההשוואות.

בנוסף לרוטינות INT 21h עליהם הסתמכנו ב-hello4.asm אנחנו מסתמכים על הרוטינה:

```
INT 21h, AH = 1
```

שמשמעותה "המתן ללחיצת מקש (עם קוד Ascii) מהמקלדת והחזר את קוד ה-Ascii ב-AL".

סימו לב למבנה של קטעי התוכנית

```
MOV AH,1
INT 21h
CMP AL,'y'
JE IsAfternoon
```

.....

```
MOV DX,OFFSET GoodMorningMessage
JMP DisplayGreeting
```

IsAfternoon:

```
MOV DX,OFFSET GoodAfternoonMessage
```

DisplayGreeting:

```
MOV AH,9
INT 21h
```

כאן אנחנו למעשה רואים פחות או יותר איך בפועל ממומשים תוכנית

בשפת מכונה ובאסמבלי: כל הפקודות שעשויות להתבצע נמצאות בגוף התוכנית. פקודות ההסתעפות של התוכנית דואגות לכך שה-CPU יעקוף את הפקודות שלפי הנסיבות אמורות שלא להתבצע ולהגיע אל אלה שכן. הדבר נעשה ע"י שילוב של פקודת השוואה (CMP) ופקודות הסתעפות (JE, JMP). כמושגים של שפת C זה כאילו יש בשפה רק "if" ו-"goto" אבל אין מכנים כמו "{" ו-"}" שלא לדבר על while וכו'. אפקט ה-"if" מתקבל ע"י פקודות הסתעפות מותנות שבו שינוי הפקודה הבאה מתבצעת רק אם תנאי מסוים מתקיים, ואחרת הפקודה הבאה לביצוע היא הפקודה העוקבת בזכרון. בחוכנית הזו משתמשים בפקודה "JE" שמשמעותה "הסתעף במקרה של שיוון". המבנה שמדובר כאן הוא שהפקודה

'CMP AL,'y

משווה את תוכן AL עם 'y' ושומר את התוצאה היכן שהוא. הפקודה

JE IsAfternoon

מבצעת את הסתעפות לנקודה האמורה (שינוי הפקודה הבאה לביצוע) רק אם הם אכן שוים. במידה ולא הפקודה הבאה לביצוע תישאר הפקודה העוקבת בזכרון, ואז מתבצעים הפקודות העוקבות תוך עקיפת הפקודות של המקרה השני על ידי פקודת ההסתעפות הבלתי מותנית JMP.


```

;
; hello4.asm - Conditional response.
;
.MODEL SMALL
.STACK 100h
.DATA
TimePrompt DB 'Is it after 12 noon (y/n)?',13,10,':$'
GoodAfternoonMessage DB 13,10
                DB 'Good afternoon, world!',13,10,'$'
GoodMorningMessage DB 13,10
                DB 'Good morning, world!',13,10,'$'
                ;
                ;
.CODE
ProgStart:
MOV AX,@DATA           ; DS can be written to only through a register
MOV DS,AX              ; Set DS to point to data segment
MOV AH,9               ; Set print option for int 21h
MOV DX,OFFSET TimePrompt ; Set DS:DX to point to TimePrompt
INT 21h                ; Print TimePrompt
MOV AH,1               ; DOS get character function #
INT 21h                ; Get a single character from keyboard
CMP AL,'y'             ; AL has input. Compare with 'y'
JE IsAfternoon         ; If AL = 'y' then go to IsAfternoon
CMP AL,'Y'             ; Compare with 'Y'
JE IsAfternoon         ; If AL = 'Y' then go to IsAfternoon

IsMorning:
MOV DX,OFFSET GoodMorningMessage ; Point display message to morning
JMP DisplayGreeting           ; Avoid following code

IsAfternoon:
MOV DX,OFFSET GoodAfternoonMessage ; Point display message to afternoon

DisplayGreeting:
MOV AH,9           ; Set print option for int 21h
INT 21h           ; Print chosen message
MOV AH,4Ch        ; Set terminate option for int 21h
INT 21h           ; Return to DOS (terminate program)
END ProgStart

```

```

E:\>tasm hello4.asm
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

```

```

Assembling file:  hello4.asm
Error messages:   None
Warning messages: None
Passes:          1
Remaining memory: 376k

```

```

E:\>tlink hello4.obj
Turbo Link Version 5.0 Copyright (c) 1992 Borland International

```

```

E:\>hello4.exe

```

```

Is it after 12 noon (y/n)?
:y
Good afternoon, world!

```

```

E:\>

```

22