תקציר מס 9

הנחיות סגמנטים סטנדרטיות

התוכניות שראינו עד כה, נכתבו תוך שימוש בהנחיות הסגמנטים המקוצרות. כלומר, על מנת להגדיר סגמנטים השתמשנו בהנחיות .DATA, .CODE וכו'. ההנחיות המקוצרות הומצאו כאשר מיקרוסופט כתבה את MASM - Macro Assembler גירסה 5.1, והן נוחות כאשר צריך ללמד את השפה. אבל בדר"כ תוכניות אסמבלי נכתבות באסמבלי הסטנדרטי, שהיה קיים מן ההתחלה.

בכדי להבין את ההנחיות הסטנדרטיות, צריך לזכור את עובדת היסוד הבאה:

תוכנית אסמבלי (סטנדרטית) הוא תאור של אוסף של סגמנטים, שאחד מהם הוא הסגמנט הביצועי הראשון, ומכיל את הפקודה הראשונה לביצוע איפוא שהוא בתוכו. סגמנט מיוחד נוסף הוא סגמנט המחסנית הראשון.

הנחיות הסגמנטים הסטנדריות נועדו לאפשר שליטה מלאה של התוכניתן על הקוד הנפרש.

הנחית ה-SEGMENT

ההנחיה הזו מגדירה התחלה של סגמנט.

מבנה

```
name   SEGMENT   align   combine   use   'class'
```

היא תהיה תקפה עד לשורה

```
name ENDS
```

כל השדות הנוספים מלבד name הינם אופציונליים.

שדה ה-name הוא פשוט שם (מזהה) של הסגמנט. האסמבלר מזהה אותו מכך שהוא השדה היחיד שמופיע לפני מילת המפתח SEGMENT בשורת ההנחיה.

לשדות align, combine ו-use יש סידרה קבועה של ערכים אפשריים ואין חפיפה בין הערכים האפשריים של השדות אלו. לפיכך האסמבלר יכול לזהות לפי

הערך המופיע בתוכנית באיזה שדה מדובר. סדר הערכים של השדות אינו משנה.
class הוא שדה היכול לקבל כל ערך משום שכמו השדה name הוא בעצם מזהה.
מסיבה זו הערך שלו מופיע בין גרשיים.

שדה ה-combine

השדה הזה קובע כיצד הסגמנט הזה מתואם או מתחבר עם הגדרות סגמנטים בעלי שדה name זהה.

הערכים האפשריים לשדה ה-combine הם

AT	PRIVATE
COMMON	PUBLIC
MEMORY	STACK

AT - יוצר מצב שבו הסגמנט משמש תמונת זכרון template בכתובת אבסולוטית.

למשל שימוש ב- AT 0B000h או AT 0B800h יגרמו לתאור הסגמנט לחפוף את שטח הוידאו של מסך המונוכרום או מסך ה-VGA בהתאמה.

COMMON - יוצר מצב שההגדרות הסגמנטים בעלי אותם שמות (שדה name) הם על שיטחי זכרון חופפים.

משהו כמו variant records של פסקל או unions של שפת C.

הזכרון שנתפס ע"י הסגנמט הוא כגודלו של תאור הסגמנט הגדול ביותר.

המאפינים של התאורים השונים של הסגמנט חייבות להיות זהים.

PUBLIC, MEMORY - יוצר מצב שההגדרות הסגמנטים בעלי אותם שמות (שדה name) מצטברים יחד לסגמנט אחד גדול.

הגודל המצטבר אינו יכול להיות גדול יותר מ-64K.

זהו מצב שבו מודולים שונים משתמשים באותו סגמנט, אבל לכל אחד יש משתנים - שטח - משלו.

משתנים גלובליים נעשים נגישים בדרך הזו.

בין השאר זה מאפשר חסכון בעדכון באוגרי הסגמנט במעבר בין מודולים, בתנאי שיחד אינם עוברים את המגבלה של 64K.

STACK - יוצר מצב שכל הסגמנט המתואר להלן מצטבר (בסגנון PUBLIC) לסגמנט אחד עם כל הסגמנטים האחרים ששדה ה-align שלהם הוא STACK ועם הרצת התוכנית, המערכת תדאג לכך שהזוג SS:SP יצביע מעבר לסוף השטח. עבור תאורי הסגמנט של המחסנית יש צורך גם להגדיר שדה class עם הערך 'STACK'.

STACK הוא combine type המשמש להגדרת מחסנית ולא שום דבר אחר.

PRIVATE - יוצר מצב שבו הגדרת הסגמנט הזה אין שום קשר לסגמנטים בעלי אותו שם (שדה ה-name). הסגמנט המתואר יקבל מספר סגמנט פיזי יחודי לו, בלי קשר לשאלה האם יש תאורי סגמנטים אחרים בעלי שדה name זהה ואיזה שדות יש להם.

זהו מצב שבו מונעים יחסי גומלין בין סגמנטים שבמקרה קיבלו אותו שם.

שדה ההתיצבות align

גורם לסגמנט להתחיל בגבול מסוים, כולל הזה אם יש צורך.

הערכים האפשריים לשדה ה-align הם

BYTE,	PARA (16 byte),
WORD,	PAGE (256 byte).
DWORD,	

כלומר אם ערך שדה ה-align הוא DWORD, אז השדה הראשון המתואר בסגמנט יוצב בכתובת המתחלקת ב-4.

סגמנט תמיד מתחיל בכתובת המתחלקת ב-16. לשדה ה-align יש משמעות רק כאשר סגמנט אחד מתואר במספר מקומות בתוכנית (בקבצים שונים למשל) באופן מצטבר. במקרה כזה כל תאור סגמנט מתחיל בכתובת שמתאימה לשדה ה-align, במידה והוא לא הראשון.

השיקול הוא יעילות -

מחשבי 286 עובדים מהר יותר בגבולות ה-WORD

מחשבי 386 עובדים מהר יותר בגבולות ה-DWORD.

התיצבות על BYTE נותן את התוכנית הקצרה ביותר.

שדה ה-use

ערכים אפשריים לשדה ה-use הם USE16 ו-USE32. USE16 מגדיר סמגנט עם היסט 16 ביט - עד 64K. USE32 מגדיר סמגנט עם אופסט 32 ביט - עד 4G.

סגמנטים 32 ביט הם מעשיים רק עבור המחשבים 386 ואילך. ברירת המחדל היא כמובן USE16.

שדה ה-class

השדה הזה המופיע בין גרשיים נותן שם לקבוצה של סגמנטים, כך שהם ישובצו ברצף בלי קשר לשאלה איך הם מופיעים בקבצי המקור.

גודל class של סגמנטים מוגבל רק בכמות הזכרון הפנוי במחשב.

הנחית ה-ASSUME

ASSUME segreg:segname

או

ASSUME NOTHING

הנחית ה-ASSUME מורה לאסמבלר לאיזה סגמנט אתם מניחים שאוגר הסגמנט הזה מצביע. הנחית ה-ASSUME אינה פורשת קוד ביצועי. המתכנת צריך לדאוג בנפרד שיהיה קוד הגורם לכך, שהנחית ה-ASSUME היא נכונה. למשל עבור DS, דאגנו לכך ע"י הפקודות

```
MOV AX,@DATA
MOV DS,AX
```

לגבי CS ו-SS, הדבר נעשה ע"י DOS.

הנחית ה-ASSUME היא בעצם האמצעי המאפשר לנו שלא לציין במפורש את שם אוגר הסגמנט כאשר אנחנו משתמשים ב-label-ים לצורך שימוש בזכרון למידע, למשל בפקודה

```
MOV Var1,AX
```

כאשר האסמבלר רואה פקודה כזו, הוא בודק באיזה סגמנט ה-label (Var1 בדוגמא) מוגדר. ברגע שהוא מוצא אותו, הוא בודק אם יש אוגר סגמנט שיש לו הגדרת ASSUME לסגמנט הזה, ואם יש הוא בוחר אותו / אחד מהם. במידה <u>ואין</u> הנחיה כזו לאף אחד מאוגרי הסגמנטים, מתקבלת הודעת שגיעה.

הנחית ה-ASSUME גם מאפשרת למתכנת לגרום לאסמבלר לבדוק אם הפקודות המשתמשות ב-label-ים הן הגיוניות, מעין אמצעי למנוע גישה שגויה ל-label-ים שנמצעים בסגמנטים שאף אוגר סגמנט לא מצביע עליהם.

ההנחיה ASSUME NOTHING גרומת לכך שאין הנחית ASSUME פעילה עבור אף אוגר סגמנט.
תוכנית אסמבלי המשתמשת בהנחיות הסטנדרטיות, נקודת ההתחלה היא ASSUME NOTHING.

<u>DOSSEG</u>

הנחיה האומרת לאסמבלר לסדר את הסגמנטים לפי המוסכמות של MASM של מיקרוסופט.

באופן עקרוני ראשית סגמנטיה- CODE אחר כך סגמנטי זכרון ולבסוף STACK.

בדרך כלל אין חשיבות איך הסגמנטים מאורגנים, אבל לפעמים חשוב לדעת איך.

<u>הנחית ה-GROUP</u>

ראינו קודם איך ניתן לדאוג לכך שהגדרות סגמנטים בעלי שם <u>זהה</u> יצטברו יחד לסגמנט פיזי אחד או, להבדיל, לסגמנטים פיזיים שונים. ישנו גם אמצעי שהוא מעין אמצעי הפוך: הגרום לסגמנטים בעלי שם בפירוש <u>שונה</u> להצטבר לסגמנט פיזי

אחד. האמצעי הזה נקרא הנחית GROUP. למשל

```
DGROUP  GROUP _DATA,STACK
```

גורם לסגמנטים בשם _DATA ו-STACK להצטבר לסגמנט פיזי אחד, למרות שהשמות שונים. יש לזה שימושים בעיקר בישום מודלים של שפות עיליות (למשל התלת מגבלה על סגמנטי מידע או הסרתה הוא עינין של הכנסת שורה נוספת לקוד או מחיקתו) אך מתכנתי אסמבלי רשאים כמובן להשתמש בו במידה ויש להם עינין בכך.

הנחיות סגמנטים מקוצרות

האסמבלר מתרגם הנחיות הסגמנטים המקוצרות להנחיות סגמנטים סטנדרטיות עם מעין ערכים של ברירות מחדל לשדות שלעיל. הערכים האלו תלויים במודל (מושג שקיים רק בהנחיות המקוצרות).

לדוגמא, תחת מודל SMALL נפרש תחת CODE. ההנחיה:

```
_TEXT SEGMENT WORD PUBLIC 'CODE'
ASSUME CS:_TEXT
```

פורש: .DATA

```
_DATA SEGMENT WORD PUBLIC 'DATA'
ASSUME DS:_DATA
```

פורש: .STACK 100h

```
STACK SEGMENT PARA STACK 'STACK'
     DB 100h DUP (?)
STACK_ENDS
ASSUME SS:STACK
```

וכן

```
DGROUP GROUP _DATA,STACK
```

```
.
.
mov  ax,DataSeg
mov  ds,ax
mov  es,ax
mov  [MemVar],1
.
.
.
```

Consequently, you should exercise care in making sure that your **ASSUME** directives correspond to the actual settings of the segment registers at all times.

The Simplified Segment Directives

We discussed the simplified segment directives in some detail in Chapter 4. However, the main aspect of simplified segment directives that we haven't covered yet is exactly what segments the various simplified segment directives create. That's not something you'll normally need to know, but if you're mixing simplified and standard segment directives, you might need that information.

The segments and segment groups created by the **.CODE**, **.DATA**, **.DATA?**, **.STACK**, **.CONST**, **.FARDATA**, and **.FARDATA?** directives depend on the memory model selected by the **.MODEL** directive. (Recall that we discussed memory models in Chapter 4.) The following tables show the correspondence of memory models and the segments created by the simplified segment directives:

Table 0.1: Default Segments and Types for Tiny Memory Model

Directive	Name	Align	Combine	Class	Group
.CODE	_TEXT	WORD	PUBLIC	'CODE'	DGROUP
.FARDATA	FAR_DATA	PARA	private	'FAR_DATA'	
.FARDATA?	FAR_BSS	PARA	private	'FAR_BSS'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.CONST	CONST	WORD	PUBLIC	'CONST'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

Table 10.2: Default Segments and Types for Small Memory Model

Directive	Name	Align	Combine	Class	Group
.CODE	_TEXT	WORD	PUBLIC	'CODE'	
.FARDATA	FAR_DATA	PARA	private	'FAR_DATA'	
.FARDATA?	FAR_BSS	PARA	private	'FAR_BSS'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.CONST	CONST	WORD	PUBLIC	'CONST'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

Table 10.3: Default Segments and Types for Medium Memory Model

Directive	Name	Align	Combine	Class	Group
.CODE	name_TEXT	WORD	PUBLIC	'CODE'	
.FARDATA	FAR_DATA	PARA	private	'FAR_DATA'	
.FARDATA?	FAR_BSS	PARA	private	'FAR_BSS'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.CONST	CONST	WORD	PUBLIC	'CONST'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

Table 10.4: Default Segments and Types for Compact Memory Model

Directive	Name	Align	Combine	Class	Group
.CODE	_TEXT	WORD	PUBLIC	'CODE'	
.FARDATA	FAR_DATA	PARA	private	'FAR_DATA'	
.FARDATA?	FAR_BSS	PARA	private	'FAR_BSS'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.CONST	CONST	WORD	PUBLIC	'CONST'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

Table 10.5: Default Segments and Types for Large or Huge Memory Model

Directive	Name	Align	Combine	Class	Group
.CODE	_TEXT	WORD	PUBLIC	'CODE'	
.FARDATA	FAR_DATA	PARA	private	'FAR_DATA'	
.FARDATA?	FAR_BSS	PARA	private	'FAR_BSS'	
.DATA	_DATA	WORD	PUBLIC	'DATA'	DGROUP
.CONST	CONST	WORD	PUBLIC	'CONST'	DGROUP
.DATA?	_BSS	WORD	PUBLIC	'BSS'	DGROUP
.STACK	STACK	PARA	STACK	'STACK'	DGROUP

תוכנית דוגמא sdhello5.asm

sdhello5.asm הוא ישום של התוכנית hellola.asm באסמבלי הסטנדרטי. המזהים שנבחרו בתוכנית הזו לאו דווקא מומלצים. התוכנית sdhello5.asm נכתבה בצורה כזו שבכל מקרה שאפשר היה לבחור מזהה (שם) כלשהוא הדבר נעשה והשם הנבחר מכיל אותיות קטנות. המילים בתוכנית הזו שהם על טהרת האותיות הגדולות הם בדיוק אותם מילים שמורות של השפה שחייבות היו להכתב כמו שנכתבו.

כמעט כל המאפינים של הנחיות הסגמנטים לא היו חייבים להיות הערכים שנתנו בפועל. למשל סגמנט הקוד Code1 הוגדר כ-PRIVATE וסגמנט המידע Data1 הוגדר כ-PUBLIC, אלו לא היו חייבים להיות מוגדרים כך. המקרה היחיד שבו המאפינים היו הכרחיים היה במקרה של סגמנט המחסנית Stack1 שחייב להיות מוגדר עם הנחיות combine ו-class שיהיו 'STACK' .STACK. אחרת התוכנית לא תרוץ בצורה תקינה.

שם לב שאין יותר הנחיות .DATA, .CODE, .STACK, MODEL וגם לא @DATA. במקום זאת הסגמנטים על המאפינים שלהם הוגדרו במפורש, וכן הנחיות ה-ASSUME. אם יש צורך בהתיחסות למספר סגמנט, מצינים את השם שלו במפורש. בתוכנית הזאת זה בא לידי ביטוי בפקודות

```
MOV AX,Data1
MOV DS,AX
```

```
; sdhello5.asm - Implement "Hello World!" using
;                           standard segment directives.
;
Stack1 SEGMENT PARA STACK 'STACK'
       DB  100h DUP (?)
Stack1 ENDS
       ASSUME SS:Stack1
                                ;
Data1    SEGMENT WORD 'Data1' PUBLIC
DisplayString DB 'Hello World!',13,10,'$'
Data1    ENDS
                                ;
Code1  SEGMENT WORD 'Code1' PRIVATE
       ASSUME CS:Code1, DS:Data1
Begin:
       MOV AX,Data1    ; 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)
Code1  ENDS
       END Begin      ; Set first instruction
```

```
E:\>sdhello5.exe
Hello World!

E:\>
```