# فصل ششم

# تشکیلات و ساختار برنامه اسمبلی

## مقدمه

در فصل قبل چگونگی نوشتن و اجرای دستورات برنامه اسمبلی را در محیط نرم‌افزاری Debug بررسی کردیم. ولی این عملیات در محیط نرم‌افزاری Debug محدودیت زیادی دارد و فقط برای برنامه‌های کوچک عملی است و بسیار کند می‌باشد و اصولاً این نرم‌افزار جهت اشکال‌یابی برنامه‌ها طراحی شده است. راه عملی نوشتن برنامه‌ی اسمبلی، تایپ برنامه در یک محیط ویرایشگر (مانند EDIT در سیستم عامل، یا در محیط NC و ...) و اسمبل کردن آن توسط برنامه‌ی مترجم MASM یا TASM و سپس پیوند دادن آن توسط پیوند دهنده LINK یا TLINK است، که در این مرحله برنامه‌ی نهایی به صورت EXE یا COM تولید می‌شود و قابلیت اجرا توسط سیستم عامل را پیدا می‌کند. ولی برای اجرای عملیات مذکور، یک سری مقرراتی باید رعایت شود تا اینکه برنامه مترجم اسمبلر بتواند برنامه اسمبلی را ترجمه نماید و سپس برنامه پیوند دهنده بتواند آن‌ها را پیوند دهد که قابل اجرا گردد.

## قواعد بر چسب دستور و نام متغیرها

اصولاً برچسب، آدرس دستور در حافظه است و نام متغیر نیز، آدرس اطلاعات در حافظه می‌باشد. به عنوان مثال در دستور:

BACK1: MOV AH, OEH

BACK1: که برچسب دستور است، در حقیقت نام سمبولیک آدرس حافظه‌ای است که دستور MOV AHوOEH در آن قرار دارد و البته آدرس حافظه یک عدد است، که ما برای درک بهتر برنامه، به آن یک نام سمبولیک اختیاری مانند BACK1 داده‌ایم.

**نکته**1: برچسب دستور حتماً باید دو نقطه داشته باشد مانند :BACK

**نکته** 2: نام متغیرها دو نقطه ندارند ولی برچسب دستورها دو نقطه دارند.

برنامه اسمبلی چهار نوع برچسب و یا نام دارد که عبارتند از:

1- برچسب دستور: نام سمبولیک به آن دستور می‌دهد که با استفاده از آن نام می‌توان به آن دستور رجوع نمود.
2- برچسب یا نام روال: نام سمبولیک به یک روال یا یک سابروتین می‌دهد که این نام باید در شبه دستور PROC و ENDP یکسان باشد.
3- نام سگمنت: نامی به یک سگمنت نسبت می‌دهند و این نام در دستورات، برای مراجعه به این متغیرها استفاده می‌شوند.
4- نام متغیرها: نامی است که به داده‌ها نسبت می‌دهند و این نام در دستورات، برای مراجعه به این متغیرها مورد استفاده قرار می‌گیرد. البته شبه دستورات DB، DW، DD، DQ و DT برای تعریف این متغیرها استفاده می‌شوند.

## راهنما یا شبه دستور

برنامه‌ی مترجم، یا اسمبلر دارای فرمان‌هایی است که ما را در کنترل اسمبل کردن یا ترجمه نمودن و تهیه لیست برنامه یاری می‌نماید. این فرمان‌ها به شبه‌دستور، یا راهنمای اسمبلر معروفند و کد زبان ماشین ندارند و فقط در زمان اسمبل یا ترجمه کردن برنامه، عمل می‌نمایند. با استفاده از این شبه‌دستورات یا راهنماهای برنامه اسمبلر، ما می‌توانیم به برنامه یک نام بدهیم و شکل صفحات برنامه را مشخص کنیم و برنامه را به سگمنت‌های کد، داده، پشته و ... تقسیم نمائیم و ابتدا و انتهای آنها را

مشخص کنیم. شروع و پایان برنامه را مشخص نمائیم و .... که معمول‌ترین شبه‌دستورات یا راهنماهای اسمبلر به شرح ذیل می‌باشند.

### شبه‌دستورات PAGE و TITLE

برنامه‌ی مترجم اسمبلر تعدادی شبه‌دستور، برای کنترل لیست برنامه‌ی اسمبلی دارد از جمله شبه‌دستورهایی که به این منظور می‌توانیم در هر برنامه استفاده کنیم PAGE و TITLE می‌باشند. با شبه‌دستور PAGE در شروع یک برنامه می‌توانیم، تعداد خطوط روی یک صفحه و تعداد ماکزیمم حرف روی خط را برای چاپگر تعیین نمائیم.

شکل کلی این شبه‌دستور به صورت: PAGE [Length],[Width] که Length تعداد خطوط در صفحه و Widthتعداد حروف در خط را معین می‌کند. مثلاً در شبه‌دستور: PAGE 40,80 تعداد ماکزیمم خطوط یک صفحه، 40 و تعداد حروف هر خط 80 تعیین شده است. برای این که یک عنوان در بالای هر صفحه، در سطر 2 از برنامه چاپ شود، از شبه دستور TITLE که شکل کلی آن بصورت زیر است استفاده می‌کنیم: TITLE Text [Comment] یک روش این است که به جای Text از نام برنامه‌ای که بر روی دیسک ذخیره کرده‌ایم استفاده کنیم.

### شبه‌دستور END

این شبه‌دستور نقطه‌ی انتهای یک برنامه را به مترجم اسمبلر اعلام می‌دارد و شکل کلی آن به صورت: END [Entry-Point Label] و یا [ برچسب نقطه شروع دستورات برنامه ] END می‌باشد که جلوی END باید برچسب یا LABEL نقطه شروع دستورات برنامه اسمبلی باشد.

### شبه‌دستورات تعریف قطعه، یا سگمنت SEGMENT و ENDS

همان‌طور که قبلاً ذکر شد، یک برنامه اسمبلی در کامپیوترهای شخصی از یک یا چند سگمنت تشکیل می‌شود که سگمنت پشته، محل حافظه پشته است و سگمنت داده و سگمنت کد به ترتیب محل‌های ذخیره داده‌ها و دستورات برنامه می‌باشد. شبه‌دستوری که ابتدای سگمنت را مشخص می کندSEGMENT و شبه‌دستوری که انتهای سگمنت را مشخص می‌نماید ENDS، نام دارد. شکل کلی شبه‌دستورتعریف سگمنت به صورت:

Name SEGMENT [Align] [Combine] [Class] شروع سگمنت

اختیاری

دستورات

یا

داده ها

Name ENDS پایان سگمنت

با این شبه‌دستور می‌توانیم یک سگمنت را تعریف و ابتدا و انتهای آن را مشخص نمائیم. لازم به ذکر است که حداکثر ظرفیت هر سگمنت 64k بایت است. هر سگمنت باید یک نام داشته باشد که قبل از شبه‌دستور SEGMENT و ENDS قرار می-گیرد. البته نام سگمنت اختیاری است. همان‌طوری که ملاحظه می‌شود نام اختیاری سگمنت قبل از شبه-دستورات SEGMENT و ENDS، که به ترتیب ابتدا و انتهای یک سگمنت داده را مشخص می‌کنند، بصورت یکسان آورده شده است. این شبه‌دستور می‌تواند سه عملوند داشته باشد که ذیلاً به شرح آن‌ها می‌پردازیم.

### الف) هم‌ترازی Align

عملوند مذکور اختیاری است و جهت انطباق شروع یک سگمنت با آدرس حافظه، در مواقعی که چندین برنامه با هم پیوند داده می‌شوند به کار می‌رود. این عملوند می‌تواند یکی از کلمات BYTE، WORD، PARA و PAGE باشد که هر کدام معنای خاص خود را در ارتباط با شروع سگمنت دارد.

BYTE: سگمنت از هر آدرس می‌تواند شروع کند.

WORD: سگمنت از هر آدرس زوجی می‌تواند شروع کند.

PARA: سگمنت از هر آدرس که قابلیت تقسیم بر16 را داشته باشد می‌تواند شروع شود (این شبیه‌دستور پیش‌فرض برنامه اسمبلی می‌باشد و نیازی به تکرار آن نیست). در این صورت اگر سگمنتی به آدرس مثلاً 00024H پایان یابد، سگمنت بعدی از آدرس00030H شروع می‌شود که قابل تقسیم بر 10Hاست به عبارت دیگر کوچکترین رقم آدرس سگمنت در این نوع ترازبندی صفر می‌باشد.

PAGE: سگمنت از هر آدرس که قابل تقسیم بر 256(100H) را داشته باشد می‌تواند شروع شود. در برنامه‌های معمولی می-توان کلاً این قسمت از شبه‌دستور SEGMENT را حذف نمود.

### ب) ترکیب Combine

این عملوند چگونگی ترکیب سگمنت در حال تعریف با سگمنت‌های همنام در سایر برنامه‌ها را مشخص می‌کند به عبارت دیگر مشخص می‌نماید که در موقع پیوند برنامه، آیا این سگمنت با سایر سگمنت‌ها باید پیوند داده شوند یا نه. عملوند Combine اکثراً یکی از کلمات public یا STACK می‌تواند باشد که در موقع پیوند، در حافظه به طور متوالی قرار می‌دهد و تشکیل یک سگمنت بزرگتری را می‌نماید. لذا در برنامه‌های معمولی که فقط یک سگمنت پشته و داده و کد دارند، نیازی به کلمه PUBLIC نیست.

کلمه‌ی STACK فقط در سگمنت پشته استفاده می‌شود و گذاشتن آن در برنامه اجباری است. به عنوان مثال:

Name SEGMENT STACK

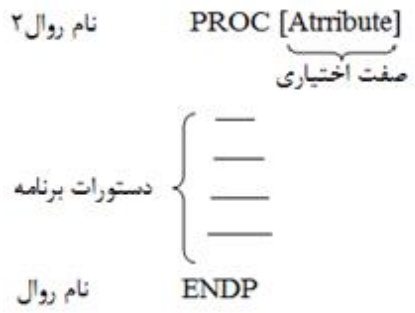
مشخص می‌نماید در موقع پیوند برنامه، سگمنت پشته برنامه کاربر با سگمنت پشته سیستم‌عامل ترکیب شود و یک سگمنت پشته واحد برای اجرای برنامه درست کند.

### ج) کلاس Class

کلمه‌ی کلاس در موقع پیوند برنامه‌ها، برای ترکیب سگمنت‌های از یک نوع به کار می‌رود. به عنوان مثال سگمنت داده از یک برنامه، با سگمنت داده از برنامه دیگر و سگمنت کد از یک برنامه با سگمنت کد از برنامه دیگر باهم ترکیب می‌شوند. برای این منظور کلمات 'CODE و 'DATA' و 'STACK' در کوتیشن به ترتیب برای سگمنت کد، داده و پشته استفاده می‌شوند.

### شبه‌دستورات روال شامل PROC، ENDP، FAR، NEAR

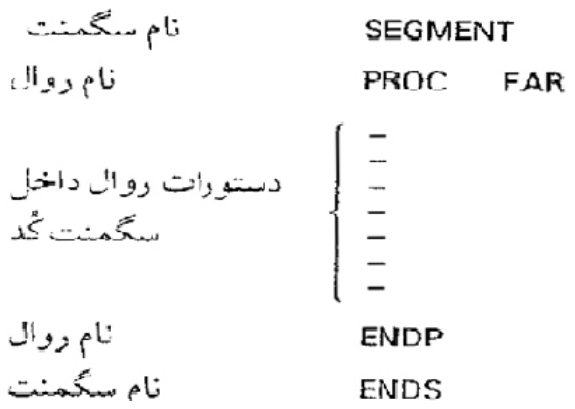
سگمنت کد که حاوی دستورات برنامه است حداقل یک یا چند روال دارد که با شبه‌دستور PROC تعریف می‌شود شکل کلی شبه‌دستور PROC به صورت زیر می‌باشد.




که صفت Attribute می‌تواند FAR و یا NEAR باشد.

شبه‌دستور PROC و ENDP ابتدا و انتهای یک روال یا سابروتین را مشخص می‌نماید و دارای نام برچسب یکسانی هستند. سابروتین یا روال به مجموعه‌ای از دستورات اطلاق می‌شود که کار معینی را انجام می‌دهد و می‌تواند از نقاط مختلف برنامه فراخوانی و اجرا شود و برنامه‌نویس را از تکرار دستورات رهایی بخشد.

اگر سگمنت کد فقط دارای یک روال باشد طبق زیر:



تعریف می‌شود. نام روال باید مشخص و منحصر به فرد باشد و از مقررات نامگذاری برنامه‌ی اسمبلی پیروی نماید. برای صفت در اینجا کلمه FAR استفاده شده است، که یک صفت برای روال می‌باشد و شبه دستور PROC FAR در یک برنامه EXE، برای سیستم‌عامل مشخص می‌کند که این نقطه ابتدای برنامه اصلی کاربر است و دستورات از اینجا به بعد باید اجرا شوند. شبه‌دستور ENDP انتهای روال را مشخص می‌کند و باید دارای همان نامی باشد که در شبه‌دستور PROC به کار برده شده است و شبه دستور ENDS پایان سگمنت را اعلام می‌کند و می‌بایستی دارای همان نامی باشد که در جلوی شبه‌دستور SEGMENT است.

سابروتین یا روال‌ها به دو دسته داخلی و خارجی تقسیم می‌شوند. اگر روال فقط از داخل سگمنتی که این روال تعریف شده قابل فراخوانی باشد، روال یا سابروتین را داخلی نامند و در شبه‌دستور PROC باید از صفت NEAR استفاده شود. درصورتی که روال تعریف شده بتواند از سگمنت‌های دیگر نیز فراخوانی شود، مثلاً از قطعه‌ای که سیستم‌عامل در آن قرار دارد، آن‌گاه روال را خارجی نامند و در شبه‌دستور PROC اگر صفت قید نشود، اسمبلر صفت NEAR را به طور پیش‌فرض در نظر می-گیرد.

### شبه‌دستور ASSUME

شبه‌دستور ASSUME ارتباط بین نام هر سگمنت و ثبات‌های سگمنت را برقرار می‌کند. پروسسور، ثبات سگمنتSS را برای آدرس شروع سگمنت پشته، ثبات سگمنت DS را جهت آدرس شروع سگمنت داده و ثبات سگمنت CS را برای آدرس شروع سگمنت کد استفاده می‌کند. شبه‌دستور ASSUME در سگمنت کد بلافاصله قبل از شبه‌دستور PROC قرار می‌گیرد که ثبات‌های سگمنت را برابر نام سگمنتی که در برنامه بکار رفته قرار دهد.

برنامه‌ی اسمبلر می‌تواند دارای یکی یا چند سگمنت داده باشد. ولی چون فقط یک ثبات، برای هر سگمنت وجود دارد، لذا در هر لحظه فقط یکی از سگمنت‌ها توسط CPU می‌تواند دستیابی شود. پس ASSUME در هر لحظه به اسمبلر می‌گوید، در حال حاضر کدام سگمنت (که توسط شبه‌دستور SEGMENT تعریف شده‌اند) به کار برده می‌شود. همچنین اسمبلر را راهنمایی می‌کند که افست آدرس متغیرها از ابتدای سگمنت داده و افست آدرس دستورات از ابتدای سگمنت کد چقدر است. این شبه دستور به صورت زیر می‌باشد:

ASSUME SS:نام سگمنت پشته، DS:نام سگمنت داده، CS: نام سگمنت کد

### ساختار اصولی یا ساختار استاندارد یک برنامه‌ی اسمبلی

همان‌طوری که قبلاً بحث شد، هر برنامه اسمبلی برای کامپیوترهای شخصی می‌تواند دارای سگمنت پشته، سگمنت داده و سگمنت کد باشد که دستورات کامپیوتر در سگمنت کد قرار می‌گیرد. علاوه بر این برنامه اسمبلی دارای یک عنوان یا نام است در ضمن سگمنت پشته، داده و کد، باید دارای نام، با شبه‌دستورات SEGMENT و ENDS باشد. شکل زیر ساختار اصولی یک برنامه‌ی اسمبلی با سگمنت پشته به نام STACKSG را نشان می‌دهد. سگمنت داده به نام DATASG و سگمنت کد نیز با نام CODESG تعریف شده‌اند.

```
        PAGE 110,100
TITLE   'nemoune1.asm' an EXE program
;-----------------------------------------------
;               1- Define stack segment
;               -----------------------
STACKSG SEGMENT STACK 'STACK'
        DW 32H DUP(0)     ; 32H word for stack
STACKSG ENDS              ; End of segment
;-----------------------------------------------
;               2- Define data segment
DATASG SEGMENT  'DATA'
;
;         متغیرها یا داده‌های برنامه
;         ----------------
;
DATASG  ENDS              ;End of segment
;-----------------------------------------------
;               3- Define code segment
;               ----------------------
;
CODESG  SEGMENT  'CODE'
        ASSUME SS:STACKSG,DS:DATASG,CS:CODESG
MAIN    PROC FAR
        MOV AX,DATASG     ;Initialize DS
        MOV DS,AX         ;  register
;
;               دستورات برنامه
;         ----------------------
;
        MOV AX,4C00H      ; End of
        INT 21H           ; processing
MAIN    ENDP              ; End of procedure
CODESG  ENDS              ; End of segment
        END MAIN          ; End of program
```

## تعریف متغیر و تخصیص مقدار اولیه به آن در سگمنت داده‌ی برنامه‌ی اسمبلی

برای تعریف متغیرها و تخصیص مقدار اولیه به آن‌ها، از شبه دستور DB، DW، DD، DQ، DT و DUP استفاده می‌شود که در سگمنت داده قرار می‌گیرند. فرم کلی تعریف متغیرها به صورت زیر است:

[نام] Dn  عملوند

که نام داخل کروشه یک نام سمبولیک و اختیاری است. ولی اگر دستورات برنامه به آن رجوع نمایند نام مذکور لازم می‌باشد. در ذیل هر یک از شبه‌دستورات فوق و کاربرد آن‌ها تشریح می‌گردد.

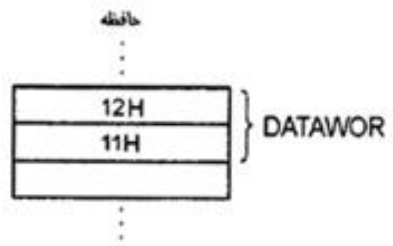
### الف) شبه‌دستور DB

این شبه‌دستور در سگمنت داده تعریف می‌شود و یک بایت حافظه برای متغیرها تخصیص می‌دهد. به عنوان مثال شبه‌دستور: COUNT DB 13H باعث می‌شود که برنامه‌ی مترجم اسمبلر، خانه حافظه‌ای به نام متغیر COUNT، یا آدرس COUNT در حافظه رزرو نماید و مقدار آن را 13H قرار دهد.

**ب) شبه‌دستور DW**

این شبه‌دستور در سگمت داده تعریف می‌شود و دو بایت حافظه برای متغیرها تخصیص می‌دهد.

به عنوان مثال: DATAWORD DW 1112H




برنامه‌ی مترجم اسمبلر دو بایت حافظه از آدرس DATAWOR را برای متغیر مذکور رزرو نماید که در بایت اول مقدار کوچکتر عدد یعنی 12Hو در بایت بعدی مقدار بزرگتر عدد یعنی 11Hرا قرار می‌دهد.

**ج) شبه‌دستورات DD، DQ وDT برای تعریف متغیرها**

این شبه‌دستورات در سگمنت داده برنامه‌ی اسمبلی برای تعریف متغیرها به کار می‌روند که شبه‌دستور DD تعداد دو کلمه (چهار بایت) و DQ تعداد تعداد چهار کلمه (هشت بایت) و DT تعداد 10بایت، برای متغیرها در نظر می‌گیرند. به عنوان مثال: DWORD5 DD 32445678H چهار بایت در حافظه رزرو می‌کند که در بایت اول78H، در بایت بعدی56 و در بایت‌های بعدی به ترتیب 44 و 32 قرار خواهند گرفت. به همین ترتیب در تعریف DQ به صورت DWORD5 DD 32445678H مقدار چهار کلمه یا 8 بایت آدرس در نظر می‌گیرد.

**د) شبه‌دستور DUP**

با این شبه‌دستور در سگمنت داده، می‌توان برای نوشتن یک سری اطلاعات مساوی در متغیرها استفاده نمود در غیر این صورت باید اطلاعات مذکور را چندین بار تکرار نمود. به عنوان مثال

DATA1 DB 20 DUP(80)

20 بار عدد 80 را برای متغیر DATA1 تکرار می‌کند.

اصولاً شبه‌دستور DUP به ما این امکان را می‌دهد که عبارت ساده‌تری برای متغیرها بنویسیم.

## بررسی اطلاعات سگمنت داده در حافظه

برای مشاهده‌ی نحوه قرار گرفتن اطلاعات متغیرها در حافظه، نرم افزار Debug اجرا کرده و سپس محتوای سگمنت داده توسط فرمان D DS:0 بر روی مانیتور نشان داده می‌شود.

## ساختار ساده شده برنامه‌ی اسمبلی

ساختار برنامه‌ی اسمبلی را می‌توان ساده‌تر نمود به شرطی که از رهنمای اسمبلی MODEL SMALL استفاده شود. راهنمای اسمبلی سگمنت‌های پشته، داده و کد را به صورت زیر تعریف می‌کند:

.STACK

.DATA

.CODE

که هر یک از راهنماهای اسمبلر فوق به طور خودکار، تولید راهنماهای اسمبلر SEGMENT و ENDS را می‌کنند و دیگر نیازی به نوشتن آن‌ها نیست. البته در این ساختار، دستوراتی که ثبات سگمنت داده DS را مقدار اولیه می‌دهند به صورت زیر می‌باشند:

MOV AX,@data ;Set the address of data

MOV DS, AX ;segment to DS

```
        PAGE 100,110
TITLE   'nemoune3.asm' a simplified program
;-----------------------------------------------
        .MODEL SMALL
        .STACK  64          ;Define stack
        .DATA               ;Define data segment
;
;               تعریف متغیرها یا داده‌های برنامه
;               ----------------
;
;-----------------------------------------------
        .CODE               ;Define code segment
MAIN    PROC FAR
        MOV AX,@data        ;1- Set data segment
        MOV DS,AX           ;2- address
;
;
;                   دستورات برنامه
;               ----------------------
;
;
        MOV AX,4C00H        ; End of
        INT 21H             ; processing
MAIN    ENDP                ; End of procedure
        END MAIN            ; End of program
```