برنامه نویسی اسمبلی

در این صفحه ساختار کلی یک برنامه به زبان اسمبلی توضیح داده می شود. به نحوه تعریف متغیرها و ثابت ها، استفاده از راهنماهای اسمبلر و اسمبل کردن و اجرای برنامه نیز اشاره شده است. .

ساختار یک برنامه اسمبلی

هربرنامه اسمبلی از تعدادی خط تشکیل شده است (line oriented). هر خط اگر توضیح نباشد شامل چهار فیلد می تواند باشد: برچسب، نماد سمبلیک، عملوندها و توضیحات. قالب کلی یک عبارت به صورت زیر است:

```
[label] mnemonic [operand][;comment]
```

مثال.

```
MovInstruction:  mov   AX,BX   ;this is a MOV instruction
```

برچسب

برچسب ها در برنامه اسمبلی مانند زبان های سطح بالا بکار گرفته می شوند. برچسب (label) یک فیلد اختیاری است که نامی را به عبارت اختصاص می دهد وامکان مراجعه و پرش به این دستور را از دستورات دیگر فراهم می کند.

برچسب های درون سگمنت کد به علامت کولن (:) ختم می شوند و اغلب به عنوان مقصد در دستورات پرش استفاده می شوند.

مثال.

```
                mov     CX, 25
Loop1:          mov     AX, CX
                call    PrintInteger
                loop    Loop1
```

یک برچسب می تواند حداکثر ۳۱ کاراکتر باشد. کاراکترهای معتبر برچسب عبارتند از: حروف، ارقام و کاراکترهای $ _ @ و . هستند. برچسب نباید از کلمات کلیدی باشد و با رقم شروع شود و کاراکتر نقطه تنها می تواند به عنوان اولین کاراکتر آن استفاده شود.

نکته. برای خوانائی برنامه بهتر است برچسب از ستون اول آغاز شود.
نکته. بعضی از نمادهای سمبلیک نیازمند یک برچسب هستند.
نکته. اسمبلر حساس به متن نیست. بین کاراکترهای کوچک و بزرگ تفاوتی قائل نمی شود.

نماد سمبلیک

فیلد نماد (mnemonic) شامل یک دستورالعمل است. دستورالعمل ها به دو دسته تقسیم می شوند: دستورالعمل های زبان اسمبلی، راهنماهای اسمبلر (assembler directives).

منظور از دستورالعمل های زبان اسمبلی دستورالعمل های 80x86 معرفی شده در بخش قبل هستند (مانند mov، add و ...). در زمان ترجمه برنامه اسمبلر معادل کد زبان ماشین آنها را پیدا کرده و جایگزین می کند.

راهنماهای اسمبلر دستورات خاصی هستند که از دستورالعمل های معتبر 80x86 نیستند و در فایل ترجمه شده ظاهر نمی شوند بلکه هنگام ترجمه برنامه به اسمبلر مطالبی را می فهمانند (model، .stack، .data، .code، dw و end).

عملوند

فیلد عملوند (operand) شامل پارامترهای موردنیاز دستورالعمل مشخص شده در فیلد نماد هستند. هر دستورالعمل با توجه به عملکردش ممکن است شامل هیچ، یک یا دو عملوند باشد. عملوند مکان داده مورد نیاز پردازنده جهت انجام عمل خواسته شده را مشخص می کند. فیلد عملوند در راهنماهای اسمبلر معمولا شامل اطلاعات بیشتری است.

مثال.

```
nop             ;no operands -- does nothing
inc AX          ;one operand -- adds 1 to the contents of AX
add Word1,2     ;two operands -- adds 2 to the contents
                ; of memory word WORD1
```

نکته. عملوندها هرگز تنها ظاهر نمی شوند. نوع و تعداد عملوندها بستگی به دستورالعمل دارد.
نکته. عملوندها با کاما از هم جدا می شود.
نکته. در دستورات دو عملوندی، عملوند اول مقصد (destination) و عملوند دوم مبدا (source) نامیده می شود.

توضیح

فیلد توضیح (comment) اجازه می دهد برای هر خط برنامه یادداشتی را بنویسید. فیلد توضیح همیشه با (;) شروع می شود. وقتی اسمبلر یک خط را پردازش می کند هر چیزی که با سمیکولن شروع شود را ندیده می گیرد. درک برنامه اسمبلی بدون توضیحات تقربا غرر ممکن است. برنامه های خوب برای هر دستور توضیحی می نویسند .

مثال.

```
; Initialize registers
    mov AX,0
    mov BX,0
    mov CX,0    ;CX counts terms, initially 0
```

نکته. هر عبارت زبان اسمبلی در یک خط ظاهر می شود. نمی توانید چند عبارت زبان اسمبلی را در یک خط داشته باشید.
نکته. می توانید بین خط های برنامه خط خالی قرار دهید. خط های خالی برای جدا کردن بخش های برنامه مفید هستند.
نکته. فیلدهای مختلف می توانند در هر ستونی قرار بگیرند. هر تعداد فضای خالی می تواند فیلدها را از هم جدا کند. استفاده درست از فضای خالی در برنامه باعث خوانایی بیشتر برنامه می شود.
نکته. بین فیلدهای یک عبارت حداقل یک فاصله وجود دارد.
نکته. تنها فیلدی که باید در هر دستور زبان اسمبلی وجود داشته باشد نماد سمبلیک است.

شناسه ها

شناسه (identifier)، سمبل ، یا برچسب اسمی است که به مقدار مشخصی نسبت داده می شود. این مقدار می تواند آفستی در یک سگمنت، یک مقدار ثابت، یک آدرس سگمنت، آفستی درون یک رکورد یا حتی عملوند یک دستورالعمل باشد. در هر صورت شناسه امکان ارائه چیزی را با نام آشنا و قابل فهمی را می دهد.

نام شناسه از حروف، ارقام و کاراکترهای خاص تشکیل شده است. با محدودیت های زیر:

- یک شناسه نمی تواند با یک رقم عددی شروع شود.
- یک اسم می تواند هر ترکیبی از حروف بزرگ و کوچک باشد. اسمبلر با حساس به متن نیست.
- یک شناسه ممکن است هر تعداد کاراکتری باشد ولی تنها ۳۱ کاراکتر اول آن استفاده می شود. اسمبلر کاراکترهای بعدی را ندیده می گیرد.
- کاراکترهای _، $، ? و @ ممکن است درون یک سمبل ظاهر شوند. البته کارکترهای **$** و **?** خاص هستند و نمی توانید شناسه را منحصرا با این کاراکترها بسازید.
- یک شناسه نمی تواند از اسامی رزرو شده باشد. دستورالعمل های 80x86 و نام ثبات ها رزرو شده هستند.

مثال. چند شناسه مجاز.

```
Item1          Bletch         RightHere      Right_Here
__Special      $1234          @Home          Dollar$
WhereAmI?      @1234          .TEST          SUM_OF_DIGITS
```

مثال. بعضی شناسه های غیرمجاز.

```
1TooMany       Hello.There    $
LABEL          Right Here     Hi,There
```

مدل حافظه

مدل حافظه برنامه توسط راهنمای model. مشخص می شود. عملوند مقابل آن می تواند یکی از انتخاب های زیر باشد:

مدل حافظه	تعداد سگمنت داده	تعداد سگمنت کد
Tiny	-	۱
Small	۱	۱
Medium	۱	بیشتر از یکی
Campact	بیشتر از یکی	۱
Large	بیشتر از یکی	بیشتر از یکی
Huge	آرایه های بزرگتر از 64K	
Flat	بدون سگمنت، تنها در مد محافظت شده	

راهنماهای سگمنت

هر برنامه شامل یک یا چند سگمنت است. هنگامی که برنامه دارد اجرا می شود ثبات های سگمنت به سگمنت های جاری اشاره می کنند. چهار سگمنت را در آن واحد می توان داشت؛ کد، داده، پشته و اضافی. در مد حقیقی هر سگمنت حداکثر 64KB است. البته معمولا برنامه ها کمتر از 64KB را استفاده می کنند. اسمبلر اندازه سگمنت را بر اساس تعداد بایت های مورد استفاده سگمنت تنظیم می کند. بنابراین اگر برنامه ای برای نمونه تنها10KB برای ذخیره داده نیاز دارد سگمنت داده 10KB می شود نه 64KB.

برنامه های exe. سه سگمنت اول را باید داشته باشند.

- سگمنت داده برای ذخیره متغیرهاست. آدرس متغیرها به صورت آفستی از شروع این سگمنت محاسبه می شوند.
- سگمنت کد شامل دستورالعمل های اجرائی برنامه است.
- سگمنت پشته برای نگهداری داده های موقتی و آدرس های برگشتی از برنامه پشته رزرو می شود. آدرس های پشته به صورت آفستی از ابتدای این سگمنت محاسبه می شوند.

وقتی اجرای برنامه آغاز می شود سیستم عامل دو ثبات سگمنت CS و SS را برای اشاره به کد برنامه و سگمنت پشته مقداردهی می کند. برای دسترسی به سگمنت داده ثبات ds باید حاوی آدرس سگمنت داده باشد. قبل از دسترسی به هر داده ای برنامه باید آدرس سگمنت را در ثبات DS ذخیره کند.

سگمنت ها در برنامه اسمبلی توسط راهنماهای segment و ends مشخص می شوند. یک سگمنت به فرم کلی زیر مشخص می شود:

```
segmentname     segment    {READONLY} {align} {combine} {use} {'class'}

                <statements>
segmentname     ends
```

segmentname شناسه ای است که نام سگمنت معین می کند. نام سگمنت برای بدست آوردن آدرس آنها توسط اسمبلر استفاده می شود. نام سگمنت باید در راهنمای ends هم مشخص شود.

align می تواند یکی از کلمات byte، word، dword، para یا page باشد. این پارامتر مشخص می کنند سگمنت در محدوده بایت، کلمه، کلمه مضاعف، پاراگراف یا صفحه بار شود. اگر بایت باشد سگمنت از اولین بایت آزاد بعد از آخرین سگمنت ذخیره می شود. این فیلد می تواند حذف شود. پیش فرض پاراگراف است. پاراگراف مضربی از ۱۶ بایت است.

فیلد combine ترتیبی را که سگمنت های هم نام در فایل مقصد توسط اسمبلر نوشته می شوند را کنترل می کند و می تواند یکی از کلمات public، stack، common یا memory باشد. نوع stack برای سگمنت های پشته و public برای بقیه سگمنت ها استفاده می شود.

مثال.

```
DSEG            segment
Item1           byte            0
Item2           word            0
DSEG            ends

CSEG            segment
                mov             AX, 10
                add             AX, Item1
                ret
CSEG            ends
```

هرزمان نام سگمنت به عنوان عملوند دستوری بکار برود اسمبلر بلافاصله آدرس سگمنت را جایگزین می کند.

مثال. دستور زیر آدرس سگمنت داده را در ثبات DS قرار می دهد.

```
mov     AX, dseg        ;Loads AX with segment address of dseg.
mov     DS, AX          ;Point ds at dseg.
```

راهنماهای stack.، data. و code. راهنماهای ساده شده سگمنت هستند که محل شروع سگمنت های پشته، داده و کد را مشخص می کنند. راهنمای stack. فضائی را برای پشته برنامه رزرو می کند. اندازه پشته در مقابل آن ذکر می شود. پیش فرض مقدار پشته 512 بایت درنظر گرفته می شود.

سگمنت ها به ترتیبی که در برنامه تعریف شده اند در حافظه بار می شوند.

ساختار کلی یک برنامه

```
SSeg      SEGMENT PARA
    DW 32 dup(0)
SSeg      ENDS

DSeg      SEGMENT PARA
    ;declarations
DSeg      ENDS

CSeg      SEGMENT PARA
    Main      PROC FAR
        ASSUME SS:SSeg, DS:DSeg, CS:CSeg
        mov AX,DSeg
        mov DS,AX
          ...
        mov AX,4c00h
        int 21h
    Main      ENDP
CSeg      ENDS
    END Main
```

ساختار کلی برنامه با راهنماهای ساده شده سگمنت

```
        .MODEL small
        .STACK [size]
        .DATA
    ;declarations
        .CODE
Main:
    mov AX,@Data
    mov DS,AX
     ...
    mov AX,4c00h
    int 21h       ;return to DOS
END Main
```

نکته. ابتدای هر برنامه اسمبلی باید آدرس سگمنت داده در ثبات DS قرار گیرد.

مثال. برنامه first.asm برای نمایش پیغام روی صفحه.

```
; First.asm
;
        .MODEL small
        .STACK
        .DATA
message   db "Hello world, I'm learning Assembly !!!", "$"

        .CODE
main    PROC
    mov   AX,seg message
    mov   DS,AX

    mov   AH,09
    lea   DX,message
    int   21h

    mov   AX,4c00h
    int   21h
main   ENDP
END main
```

تعریف ثابت ها و متغیرها

تعریف داده ها در سگمنت داده صورت می گیرد که با راهنمای data. شروع می شود.

ثابت ها

یک ثابت واقعی ثابتی است که مقدارش صریحا ذکر شده است. ثابت های واقعی نمایش آنچه هستند که معمولا برای مقدار دنیای واقعی انتظار داریم. ماکرو اسمبلر دارای انواع مختلفی از ثابت های صحیح، حقیقی، رشته و غیره است.

مثال.

```
123
3.14159
"Literal String Constant"
0FABCh
'A'
```

یک ثابت عددی مقداری است که می تواند در مبنای ۲، ۱۰ یا ۱۶ نوشته شود. برای مشخص کردن مبنای عدد از پسوندهای جدول زیر استفاده می شود. اگر مبنا صریحا ذکر نشود پیش فرض مبنای ۱۰ است.

پسوند	مبنا
B یا b	Binary
D یا d یا T یا t	decimal
H یا h	hexadecimal

ثابت های رشته ای درون گیومه (") یا تک گیومه (') قرار می گیرند.

مثال. ثابت های عددی.

```
0F000h          12345d          0110010100b
```

مثال. ثابت های رشته ای.

```
"This is a string"
'So is this'
'Doesn''t this look weird?'
"Doesn't this look weird?"
"Microsoft claims ""Our software is very fast.""  Do you believe them?"
'Microsoft claims "Our software is very fast." Do you believe them?'
```

ثابت نامدار (named constant) نام سمبلیکی است که نشانگر مقدار ثابتی طی فرآیند اسمبلی است. ثابت ها به صورت کلی زیر تعریف می شوند:

```
ConstantName   EQU   Value
ConstantName   =     Value
```

ConstantName نام ثابت است و Value مقداری است که به ثابت اختصاص داده می شود.

مثال.

```
One             equ     1
Minus1          equ     -1
TryAgain        equ     'Y'
String          equ     "Hello there"
Num             =       16
Size = Count * Element
```

نکته. علامت مساوی تنها برای مقدارهای عددی بکار می رود.

متغیرها

متغیرها را در هر سگمنتی می توان تعریف کرد اما اکثر برنامه نویسان همه آنها را در سگمنت داده تعریف می کنند. هر متغیر به فرم کلی زیر تعریف می شود:

```
VariableName    Type    InitialValue|?
```

Type نوع متغیر را مشخص می کند که می تواند یکی از نوع های جدول زیر باشد. نوع هائی که اغلب مورد استفاده قرار می گیرند DB و DW هستند. InitialValue مقداراولیه متغیر است. اگر نخواهیم مقدار اولیه بدهیم علامت سوال (?) می گذاریم.

نوع	تعداد بایت
byte/sbyte/db	1
word/sword/dw	2
dword/sdword/dd	4
qword/dq	8
tbyte/dt	10

مثال.

```
num     db    25h
sum     dd    ?
ANum    db    -4
```

مثال. محل های پشت سر هم که دارای یک نوع هستند آرایه نامیده می شود. رشته ها توسط راهنمای db اعلان می شوند.

```
X   dw   040Ch,10b,-13,0
Y   db   'This is an array'
Z   dd   10, 13, 'A','B','C'
```

مثال. برای تعریف یک متغیر آرایه از راهنمای dup استفاده می شود.

```
Memory     db    30 dup('$')
BigAry     dw   100 dup(?)
```

برنامه اصلی

دستورالعمل های برنامه در سگمنت کد قرار می گیرند. کد معمولا در زیر برنامه نوشته می شود. برای پایان اجرای برنامه و بازگشت به محیط سیستم عامل تابع 4c از وقفه 21h در انتهای هر برنامه باید فراخوانی شود. اگر کد صفر در ثبات AL ذخیره شود به معنی اینستکه برنامه با موفقیت به پایان رسیده است. در انتهای هر برنامه دستورات زیر باید اضافه شود.

```
Mov AX,4c00h
Int 21h
```

راهنمای end انتهای سگمنت کد و پایان برنامه را برای اسمبلر مشخص می کند. عملوندی که در مقابل آن ذکر می شود نقطه آغاز اجرا یا نام تابع اصلی برنامه را به سیستم عامل می گوید و باعث می شود انتقال کنترل اجرا هنگام شروع اجرای برنامه می شود. در واقع ثبات های CS و IP را تنظیم می کند. نقطه شروع الزاما بلافاصله بعد از راهنمای code. نیست. اگر این عملوند نباشد سیستم عامل از اولین بایت سگمنت کد شروع به اجرا می کند.

اسمبل و اجرای برنامه

برای ایجاد برنامه به سه ابزار نیاز است: یک ادیتور متن، یک اسمبلر برای تبدیل برنامه به فایل مقصد و یک لینکر برای تولید فایل اجرائی.

برنامه اسمبلی را در یک ادیتور متن نوشته و با پسوند asm. ذخیره کنید. فراموش نکنید که حتما از یک ادیتور اسکی استفاده کنید. توسط اسمبلر (masm.exe یا tasm.exe) از فایل مبدا asm. فایل مقصد obj. را ایجاد کنید. اسمبلر برنامه زبان اسمبلی را به کد ماشین تبدیل می کند. اگر خطائی در برنامه وجود داشته باشد اسمبلر خطا را گزارش می دهد. لینکر (link.exe یا tlink.exe) از یک یا ترکیب چند فایل obj. یک برنامه قابل اجرا از نوع exe. یا com. را می سازد.

مثال. برنامه first.asm را در ادیتور متن ذخیره کنید. سپس در خط فرمان سیستم عامل، در محلی که اسبلر نصب شده است، دستورات زیر را به ترتیب وارد کنید تا فایل اجرائی first.exe ایجاد شود.

```
C:\masm>masm first
C:\masm>link first
C:\masm>first
```

مثال: در برنامه زیر دو عدد با هم جمع می شود.

```
        .MODEL small
        .STACK 100h
        .DATA
number1 DW 0800h                ;=128
number2 DW ffebh                ;=-493
sum     DW ?                    ;store result

        .CODE
begin:
        mov AX,@Data
        mov DS,AX           ;initialize data segment register
        mov AX,number1      ;get first number in AX
        add AX,number2      ;add AX with second number
        mov sum,AX          ;store result in Sum
        mov AX,4c00h        ;Call Function 4ch, Int 21h with return code 0
        int 21h
End begin
```

مثال: تکه برنامه زیر اعداد ۱ تا ۱۰ را در آرایه ای از نوع word ذخیره می نماید.

```
        .MODEL small
        .STACK 100h
        .DATA
Array     DW   10 dup(?)
        .CODE
begin:
        mov  AX,@Data
        mov  DS,AX
        mov  CX,1
        mov  SI, offset Array
Forl:   mov  [SI], CX
        inc  SI
        inc  SI
        cmp  CX,10
        je   endf
        inc  CX
        jmp  forl
Endf:
```

برای دسترسی به عناصر آرایه معمولا از عملوند غیرمستقیم ثباتی استفاده می شود. برای تخصیص آدرس آفست یک آرایه می توان از دستور LEA هم استفاده کرد. دقت کنید چون عناصر آرایه دو بایتی هستند هربار دو واحد به SI اضافه می شود. در حالتی که عناصر آرایه از نوع بایت تعریف می شوند به اشاره گر آرایه یک واحد اضافه می شود.

مثال: کاراکتر * را نمایش می دهد.

```
        .MODEL small
        .STACK 100h
        .DATA
        .CODE
main    PROC
        mov  AH,2h
        mov  DL,2ah
       int   21h

       mov   AX,4c00h
      int   21h
main   ENDP
END main
```