

زیربرنامه

این صفحه نگاهی به زیربرنامه ها در زبان اسمبلی دارد. چگونگی تعریف زیربرنامه، دستورات فراخوانی و برگشت از زیربرنامه و نحوه ارسال و دریافت پارامترها شرح داده خواهد شد.

تعریف زیربرنامه

زیربرنامه های near و far

دستورات فراخوانی و بازگشت زیربرنامه

ارسال و دریافت پارامترها

زیر برنامه (procedure) مجموعه ای از دستورات است که یکبار تعریف و به دفعات استفاده می شود. با بکارگیری زیربرنامه خوانانی برنامه بالاتر رفته و از تکرار دستورات مشابه جلوگیری می شود. علاوه براین اشکال زدائی و تغییر برنامه آسان تر انجام گیرد.

وقتی یک زیربرنامه فراخوانی می شود کنترل اجرای برنامه به زیربرنامه هدایت می شود. آدرس دستورالعمل بعدی در پشته ذخیره می شود بنابراین هنگامی که زیربرنامه اجرا شد کنترل اجرا قادر خواهد بود به خط بعد از فراخوانی زیربرنامه بر می گردد.

تعریف زیربرنامه

تعریف زیربرنامه باید در سگمنت کد انجام بگیرد. از دو راهنمای proc و endp برای تعیین بلاک زیربرنامه استفاده می شود.

```
ProcedureName PROC [NEAR|FAR]
    ...
    RET
ProcedureName ENDP
```

Procedurename نام زیربرنامه است که قبل از راهنماهای proc و endp قرار می گیرد و باید یکسان باشد. عملوند near یا far اختیاری است. کلمه near به اسمبلر می گوید که زیربرنامه از نوع داخلی است. برای تعریف یک زیربرنامه خارجی از کلمه far به جای near استفاده می شود.

دستور ret باعث خروج از زیربرنامه و برگشت به فراخواننده می شود.

نکته: اگر عملوندی مقابل تعریف زیربرنامه قرار نگیرد از نوع near در نظر گرفته می شود. نکته: اگرچه تعریف یک زیربرنامه که عملاً داخلی است به صورت خارجی اشکالی ایجاد نمی کند ولی بهتر است این کار را نکنید.

مثال: زیربرنامه جمع دو عدد در AH و AL و نگهداری مجموع در ثبات BX.

```
Adding PROC near
    mov BX, AL
    add BX, AH
    Ret
Adding ENDP
```

مثال: زیربرنامه Putc برای نمایش کاراکتری که در ثبات al قرار دارد.

```
Putc PROC
    mov DL,AL
    mov AH,02
    int 21h
    ret
Putc ENDP
```

زیربرنامه های far و near

دو نوع زیربرنامه وجود دارد داخلی (intrasegment) و خارجی (intersegment).

- زیربرنامه های داخلی در همان سگمنتی که تعریف شده اند قابل فراخوانی هستند و در تعریف آنها از صفت near استفاده می شود.
- زیربرنامه های خارجی روال هائی که در سگمنت دیگری قرار دارند و از سایر سگمنت ها قابل فراخوانی می باشند و در تعریف آنها از صفت far استفاده می شود. زیربرنامه های خارجی در فایل جداگانه ای قرار دارد و هنگام لینک کردن باید به برنامه پیوند داده شوند. نتیجه کار بعد از لینک مانند زیربرنامه داخلی است.

فراخوانی از نوع near کنترل را درون همان سگمنت کد جابجا می کند و تنها مقدار IP در پشته ذخیره می شود. فراخوانی far کنترل را بین سگمنت های مختلف عبور می دهد. هر دو مقادیر CS و IP در پشته قرار می گیرند.

نکته. دستورات call و ret نوع فراخوانی را مشخص نمی کنند بلکه عملوند near|far راهنمای proc به اسمبلر می گوید فراخوانی از کدام نوع است.

دستورات فراخوانی و بازگشت زیربرنامه

دوستورالعمل که پشته را استفاده می کنند و فراخوانی و برگشت زیربرنامه را انجام می دهند call و ret هستند. برای هدایت کنترل اجرا به زیربرنامه باید آنرا فراخوانی کرد. زیربرنامه ها در هر کجای برنامه که به آن نیاز داریم با دستور call فراخوانی می شوند. دستور call به صورت زیر است:

```
call ProcedureName
```

دستورالعمل call باعث یک پرش غیر شرطی به زیربرنامه می شود و آدرس دستورالعمل بعدی را در پشته ذخیره می کند. CPU در برخورد با دستور call به آدرس شروع زیربرنامه رجوع می کند و دستورات آنرا اجرا می نماید. با برخورد به دستور ret به برنامه فراخوان بر می گردد و دستورات بعد از call را اجرا می نماید.

مثال. فراخوانی زیربرنامه Putc.

```
call Putc
```

CPU در برخورد با دستور Call عملیات زیر را انجام می دهد:

فراخوانی از نوع داخلی

۱. مقدار ثبات IP (که حاوی آدرس دستور بعد از call است) را در پشته ذخیره می کند.
۲. آدرس ذکر شده مقابل دستور call را در ثبات IP قرار می دهد.

فراخوانی از نوع خارجی

۱. مقدار ثبات CS را در پشته ذخیره می کند.
۲. بخش سگمنت آدرس ذکر شده مقابل دستور call را در ثبات CS قرار می دهد.
۳. مقدار ثبات IP را در پشته ذخیره می کند.
۴. بخش آفست آدرس ذکر شده در جلوی دستور call را در ثبات IP قرار می دهد.

دستورالعمل ret آدرس ذخیره شده IP را از پشته بر می دارد و به برنامه اصلی بر می گردد. CPU در برخورد با دستور Ret عملیات زیر را انجام می دهد:

بازگشت از زیربرنامه داخلی

۱. مقدار ذخیره شده در پشته را در داخل ثبات IP قرار می دهد.

بازگشت از زیربرنامه خارجی

۱. مقدار ذخیره شده در پشته را در داخل ثبات IP قرار می دهد.
۲. مقدار ذخیره شده در پشته را در داخل ثبات CS قرار می دهد.

نکته. اگر دستور `ret` در انتهای زیربرنامه حذف شود کنترل اجرای برنامه به زیربرنامه بعدی می رود نه دستورالعمل بعدی در برنامه اصلی.

نکته. معمولاً در ابتدای هر زیربرنامه بهتر است مقادیر ثبات هائی که تغییر می کنند را در پشته ذخیره نماییم و در انتهای زیربرنامه و قبل از دستور `ret` مقادیر آنها را از پشته بازیابی کنیم. باید توجه کنیم که دستورات `pop` متناظر با دستورات `push` باشند و کلیه داده هائی که در زیربرنامه در پشته `push` شده اند باید `pop` شوند وگرنه به با دستور `ret` به آدرس درست پرش نمی کند.

مثال. زیربرنامه برای نمایش ۴۰ کاراکتر `space`. توجه کنید زیربرنامه `Putc` درون زیربرنامه `PrintSpaces` فراخوانی شده است.

```
PrintSpaces PROC    near
    push    AX
    push    CX
    mov     AL, ' '
    mov     cx, 40
PSLoop:    call    putc
    loop   PSLoop
    pop     CX
    pop     AX
    ret
PrintSpaces ENDP
```

در ابتدای زیربرنامه ثبات های `AX` و `CX` در پشته قرار می گیرند و در انتها به ترتیب عکس بازیابی می شوند. زیربرنامه فوق به صورت زیر فراخوانی می شود.

```
call PrintSpaces
```

ارسال و دریافت پارامترها

پارامترها مقادیری هستند که می توانید به زیربرنامه بدهید یا بگیرید. برای ارسال یا دریافت پارامترها معمولاً از ثبات، متغیرهای سراسری یا پشته استفاده می شود.

ارسال پارامتر از طریق ثبات

مثال. زیربرنامه زیر طول یک رشته را محاسبه و در ثبات `CX` برمیگرداند. آدرس شروع رشته در ثبات `SI` قرار دارد.

```
StrLen PROC
    push SI
    mov  CX, 0
Whl:   cmp  Byte Ptr[SI], '$'
    jc   EndW
    inc  CX
    inc  SI
    jmp Whl
EndW:  pop  SI
    ret
StrLen ENDP
```

ارسال پارامتر از طریق پشته

پارامترهائی که به زیربرنامه داده می شوند را می توان قبل از فراخوانی زیربرنامه در پشته اضافه کرد. پارامترها در زیربرنامه `pop` نمی شوند بلکه مستقیماً از پشته دسترسی می شوند زیرا قبل از دستور `call` در پشته اضافه شده اند و آدرس برگشتی بعد از آن اضافه می شود. علاوه بر این چون ممکن است در چندین جای زیربرنامه استفاده شوند معمولاً درون ثبات نگهداری نمی شوند و بهتر است در حافظه پشته باقی بمانند.

یک برنامه خارجی که یک پارامتر از طریق پشته را ارسال می کند در نظر بگیرید. وقتی زیربرنامه درخواست می شود پارامتر می تواند با آدرس دهی غیرمستقیم `[SP+4]` دسترسی شود. اگر پشته هم در زیربرنامه برای ذخیره داده استفاده شود عدد بیشتری باید به `SP` اضافه شود. ثبات `BP` را برای ارجاع به داده های درون پشته می توان به کار برد. ثبات `SP` با هر `push` و `pop` تغییر

می کند اما BP ابتدا برابر با SP می شود و سپس ثابت می ماند در انتهای زیربرنامه مقدار اولیه BP باید برگردانده شود. بعد از اینکه زیربرنامه تمام شد پارامترهایی که در پشته اضافه شده اند باید حذف شوند.

مثال. تابع زیر طول رشته را محاسبه و آدرس شروع رشته از طریق پشته به زیربرنامه ارسال می شود.

```
StrLen PROC
    push BP
    mov BP, SP
    mov SI, [BP+4]
    sub CX, 0
Wh1:   cmp byte ptr [SI], '$'
        jc Endw
        inc CX
        inc SI
        jmp Wh1
EndW:  pop BP
        ret
StrLen ENDP
```

مثال. محاسبه مجموع سه عدد که از طریق پشته به زیربرنامه ارسال شده اند.

```
Add3 PROC
    push BP
    mov BP, SP
    mov AX, [BP+4]
    add AX, [BP+6]
    add AX, [BP+8]
    pop BP
    ret
Add3 ENDP
```

