قرارداد

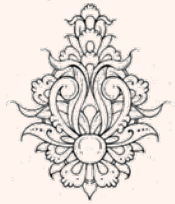
فراخوانی تابع

زبان ماشین و اسمبلی
(۱۳-۱۱-۰۰۵)

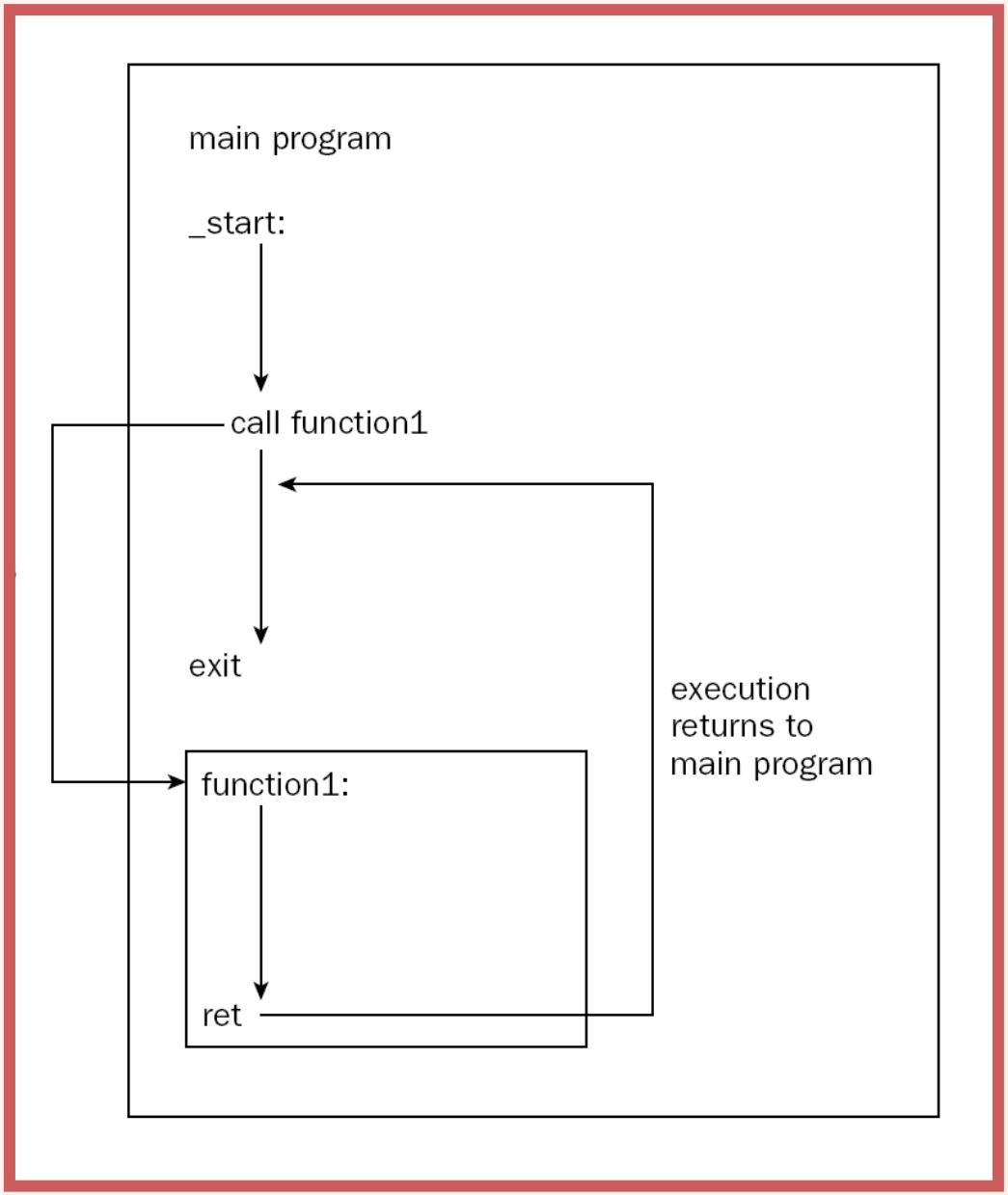
دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
بهار ۱۳۹۴
احمد محمودی ازناوه

فهرست مطالب

- قراردادهای فراخوانی تابع
  - قرارد فراخوانی تابع در زبان C (cdecl)
  - نحوه‌ی ارسال پارامتر
  - تعریف متغیرهای محلی
- ارسال پارامتر به برنامه از طریق خط فرمان

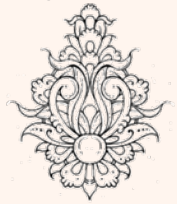


توابع


main program
_start:
call function1
exit
execution
returns to
main program
function1:
ret

توابع

- برای نوشتن تابع مراحل زیر مورد نیاز است:
  - مشخص کردن ورودی‌ها
  - نوشتن بدنه‌ی تابع
  - مشخص کردن نحوه‌ی ارسال خروجی‌ها
- ارسال آرگومان به تابع و ارسال خروجی به فراخواننده(caller) به شیوه‌های زیر امکان‌پذیر است:
  - از طریق ثبات
  - از طریق پشته
  - از طریق متغیرهای سراسری



استفاده از ثبات‌ها و متغیرهای سراسری

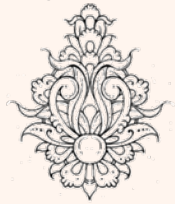
- در صورت ارسال متغیر از طریق ثبات،
  - در صورتی که تعداد آرگومان‌ها افزایش یابد، دیگر استفاده از ثبات‌ها جوابگو نخواهد بود.
  - از آن ثبات دیگر در تابع نمی‌توان استفاده کرد.
- در صورتی که مقدار خروجی از طریق ثبات بازگردد، پیش از هر اقدامی باید مقدار خروجی را به مکانی مطمئن انتقال داد.
- استفاده از تابع بازگشتی، مستلزم حفظ مقدار ثبات پیش از فراخوانی مجدد و پس از فراخوانی مجدد می‌باشد.
- استفاده از متغیرهای سراسری باعث می‌شود، استفاده‌کننده از یک تابع ملزم به تعریف یک سری متغیر سراسری شود.

شیوه‌های فراخوانی تابع در x86

Calling Convention

شیوه‌ی پیاده‌سازی سطح پایین زیربرنامه (subroutine) که از caller پارامترهایی دریافت کرده و مقدار خروجی برمی‌گرداند.
در جدول زیر سه نمونه دیده می‌شود:

Keyword	Stack Cleanup	Parameter Passing
__cdecl C convention	caller	Pushes parameters on the stack, in reverse order (*right to left*)
__stdcall Fortran convention	callee	Pushes parameters on the stack, in reverse order (*right to left*)
__fastcall	callee	Stored in registers, then pushed on stack




```
int DubNum(int) __attribute__((fastcall));
```

gcc

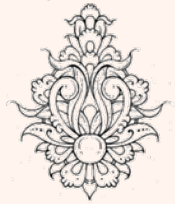
```
int __stdcall DubNum(int num){
```

visual studio

شیوه‌ی ارسال آرگومان به تابع در C

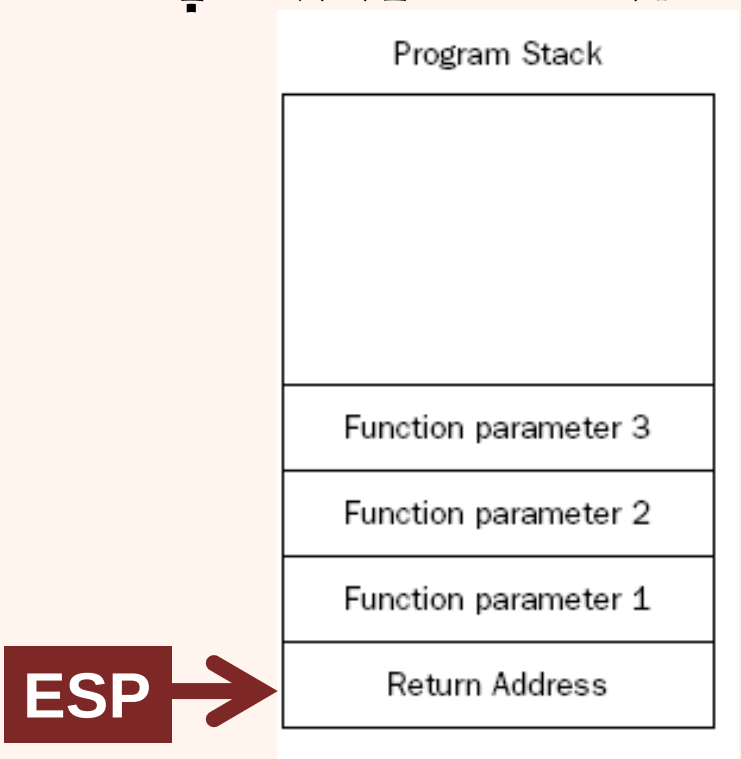
- با توجه به این که همه‌ی توابع و برنامه‌ی اصلی به پشته دسترسی دارند، بی‌دردسرترین راه برای ارسال آرگومان استفاده از پشته است.
- مقدار خروجی هم روی یک ثبات خواهد بود.

داده‌های سی‌ودو بیتی	EAX
داده‌های شصت و چهار بیتی	EDX:EAX
داده‌های ممیز شناور	st



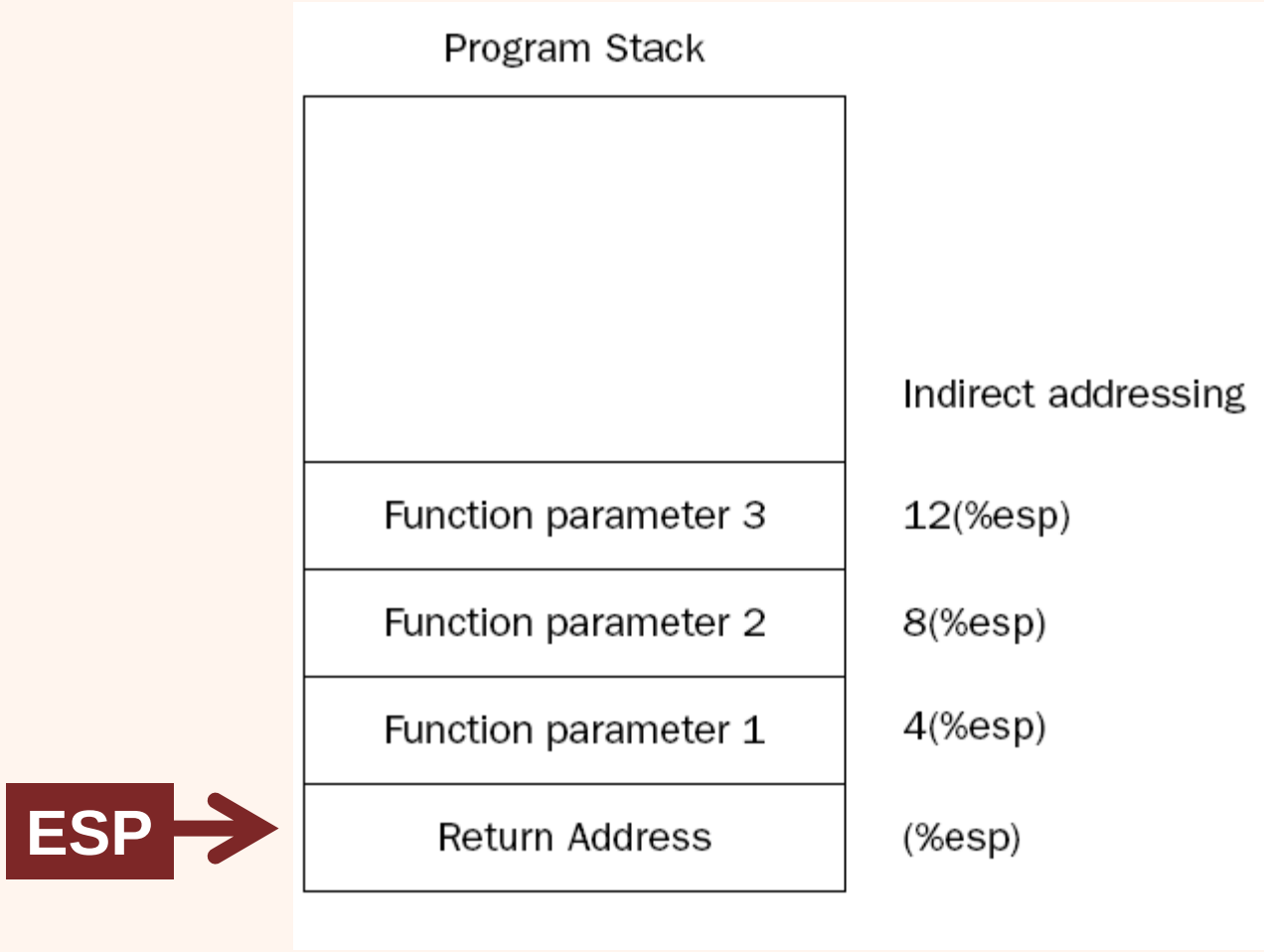
شیوه‌ی ارسال آرگومان به تابع (ادامه...)

- آرگومان‌ها به عکس ترتیبی که در پیش‌نمونه آمده است، روی پشته قرار می‌گیرند.
- با اجرای دستور call آدرس برگشت نیز روی پشته قرار خواهد گرفت.



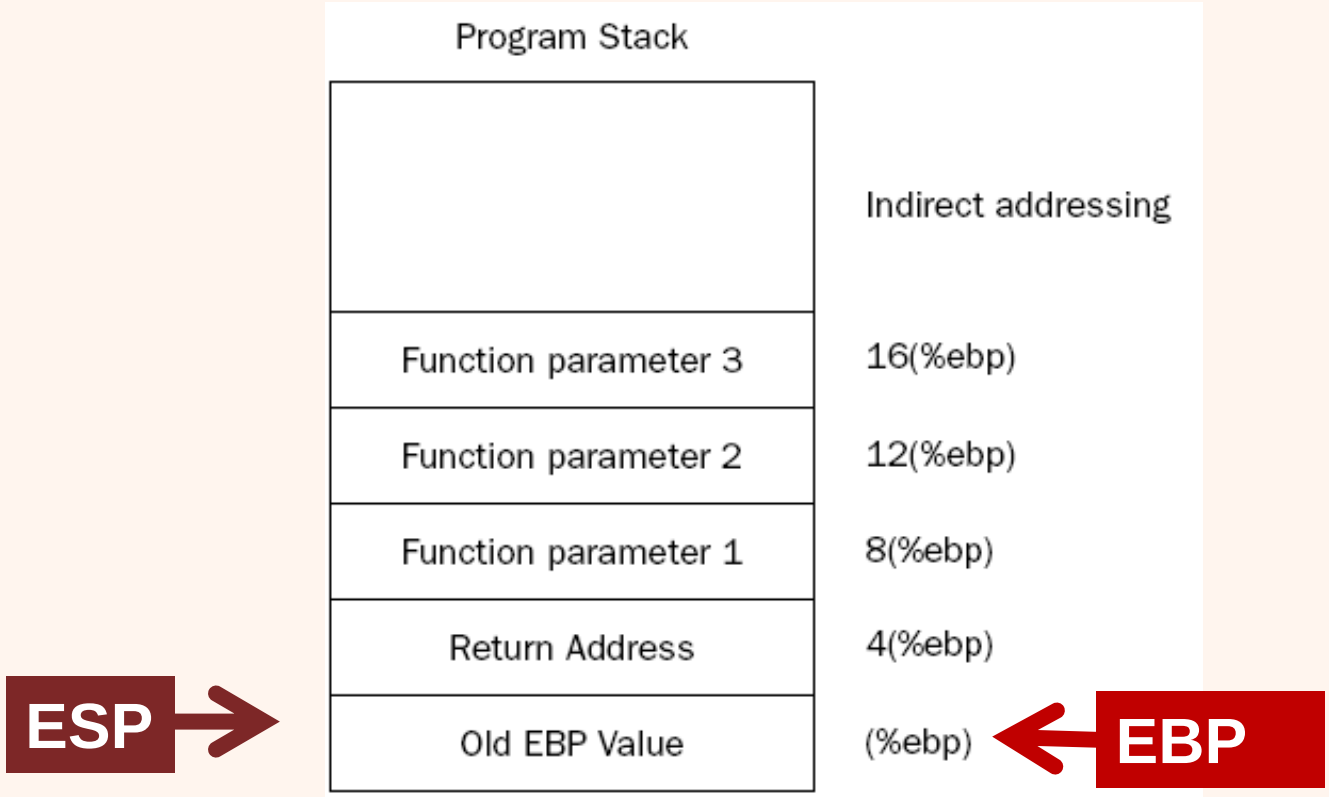
نحوه‌ی دسترسی به پارامتر، از طریق آدرس‌دهی غیر مستقیم است

شیوه‌ی ارسال آرگومان به تابع (ادامه...)



در صورتی که مقدار esp تغییر کند، محل نسبی داده‌ها تغییر می‌کند.

شیوه‌ی ارسال آرگومان به تابع (ادامه...)



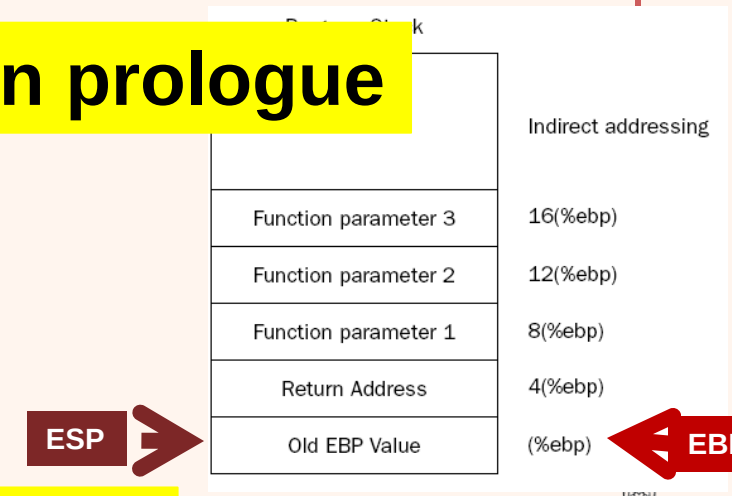
برای حل این مشکل از ebp کمک گرفته می‌شود، البته باید به گونه‌ای عمل شود که مقدار این ثبات در صورت فراخوانی تابع دیگری تغییر نکند، برای این منظور کافیست روشی اتخاذ شود که مقدار این ثبات هنگام ورود و خروج تابع یکسان باشد

آغاز و پایان یک تابع

```
function:
pushl %ebp
movl %esp, %ebp
.
.
movl %ebp, %esp
popl %ebp
ret
```

function prologue

function epilogue



LEAVE

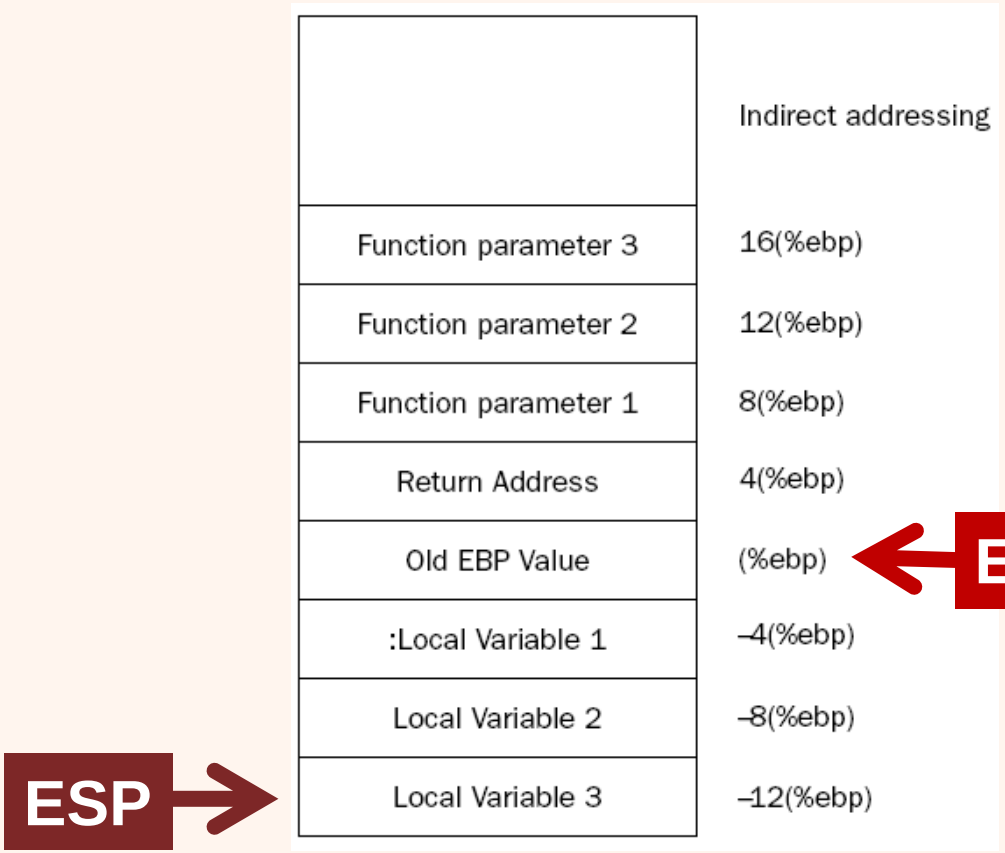
Restore Stack for Procedure Exit

+80188

متغیرهای محلی تابع

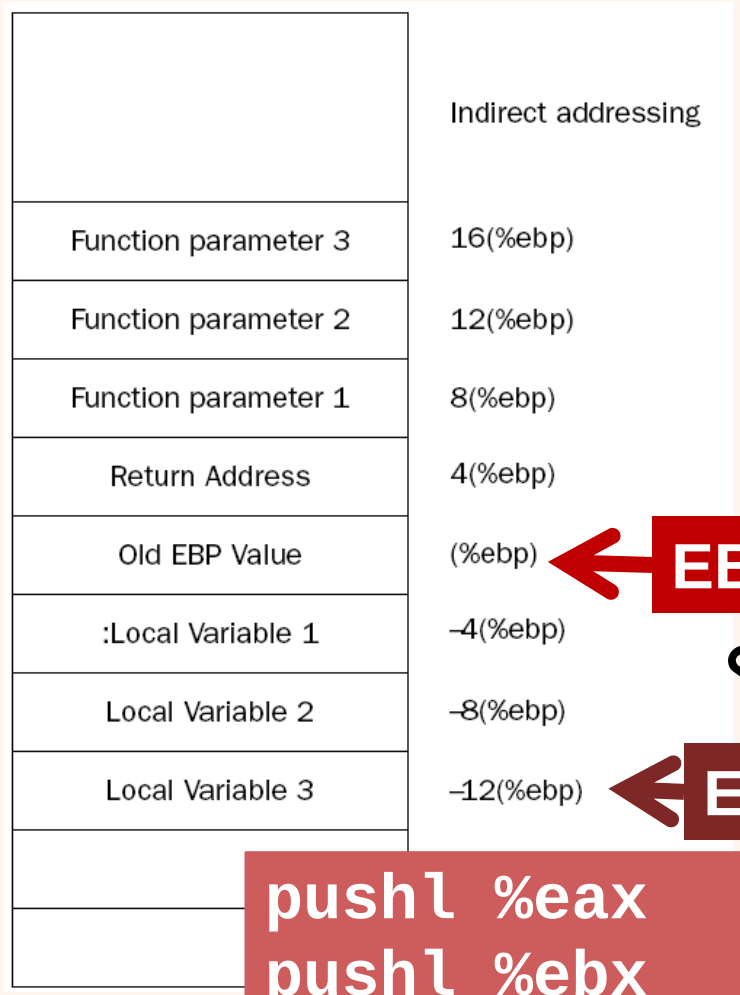
- برای این منظور نیز می‌توان از متغیرهای سراسری و ثبات‌ها استفاده کرد، ولی باز هم مشکلات قبلی پیش می‌آید.

- متغیرهای محلی نیز روی پشته تعریف می‌شوند.
- در ابتدای تابع باید برای آن‌ها فضا گرفت.



متغیرهای محلی تابع(ادامه...)

به بخش سرآغاز باید یک دستور دیگر اضافه کرد.




```
function:
pushl %ebp
movl %esp, %ebp
subl $12, %esp
.
.
```

function prologue

آیا بخش پایانی نیاز به تغییرات دارد؟

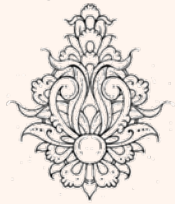
در caller چه اقدامات دیگری باید صورت پذیرد؟

```
pushl %eax
pushl %ebx
call compute
addl $12, %esp
```

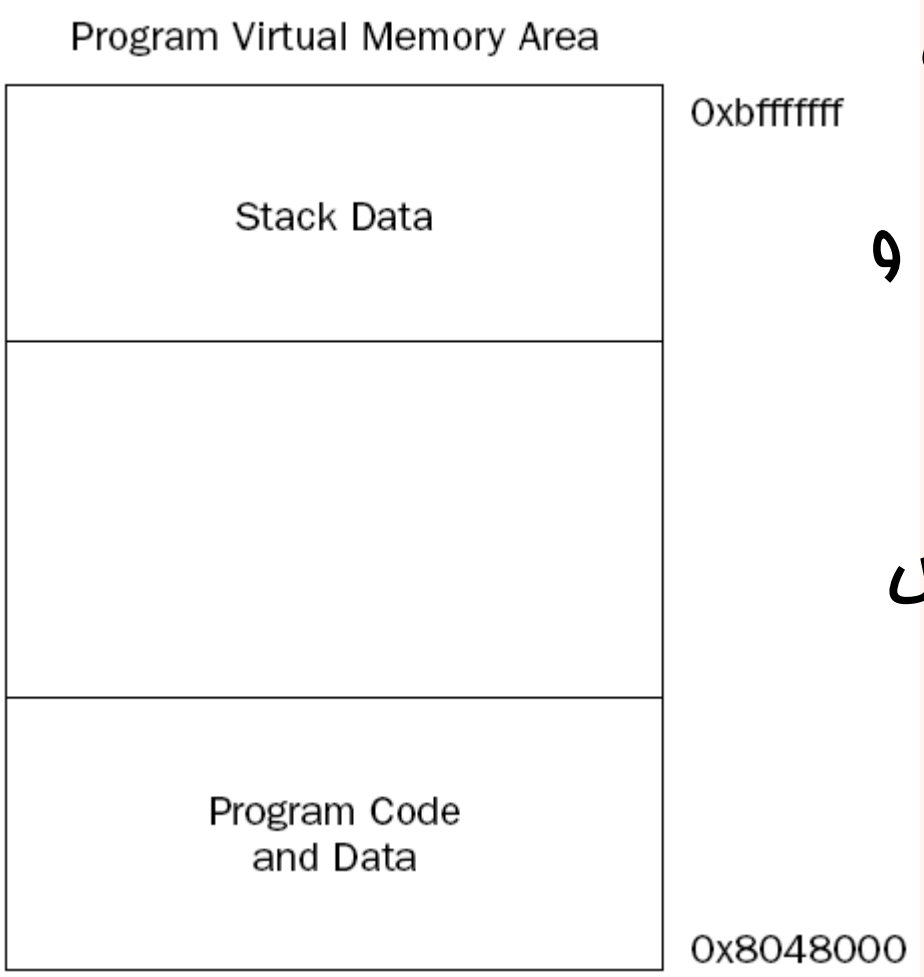
ارسال پارامتر به تابع
از طریق خط فرمان

ارسال پارامتر به تابع اصلی

- در ادامه به مبحث ارسال پارامتر به برنامه از طریق خط فرمان می‌پردازیم.
- هنگامی که در سیستم‌عامل لینوکس یک برنامه فراخوانی شود، این برنامه در هر جایی از حافظه‌ی فیزیکی می‌تواند قرار گیرد.
- برای سادگی به هر برنامه یک حافظه‌ی مجازی اختصاص داده می‌شود.



ارسال پارامتر به تابع اصلی (ادامه...)



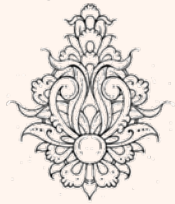
- این فضا برای همه‌ی برنامه‌ها از آدرس 0x8048000 شروع و تا آدرس 0xbfffffff ادامه دارد.
- وظیفه‌ی تبدیل آدرس بر عهده‌ی سیستم‌عامل است.

ارسال پارامتر به تابع اصلی (ادامه...)

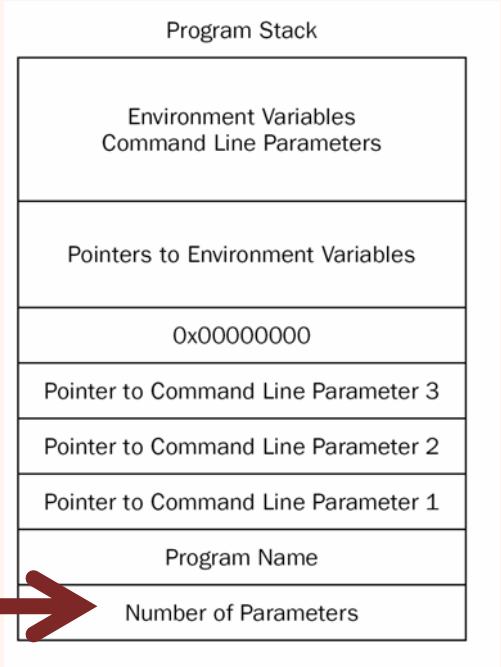
```
(gdb) b 1
Breakpoint 1 at 0x8048100: file functest4.s, line 1.
(gdb) r
Starting program: /home/ahmad/Courses/Assembly/chapter7/functest4

Breakpoint 1, _start () at functest4.s:10
10          nop
(gdb) print $esp
$1 = (void *) 0xbffff490
(gdb)
```

- مقدار esp با مقدار گفته شده تفاوت دارد، در واقع پیش از اجرای برنامه، سیستم‌عامل یک سری اطلاعات را روی پشته قرار می‌دهد:
  - تعداد پارامترهای خط فرمان
  - اسم برنامه
  - پارامترهای خط فرمان
  - متغیرهای محیطی مربوط به سیستم‌عامل



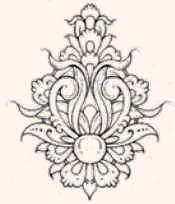
ارسال پارامتر به تابع اصلی (ادامه...)



- روی پشته ابتدا تعداد پارامترهای فرستاده شده (شامل نام برنامه)، اشاره‌گر به نام برنامه و سپس اشاره‌گرهایی به پارامترهای ارسالی وجود دارد.
- پس از آن صفر قرار دارد.

```
(gdb) print $esp
$2 = (void *) 0xbffff490
```

```
(gdb) x /20x 0xbffff490
0xbffff490:     0x00000001      0xbffff60a      0x00000000      0xbffff63a
0xbffff4a0:     0xbffff65b      0xbffff66e      0xbffff67e      0xbffff689
0xbffff4b0:     0xbffff6d9      0xbffff6eb      0xbffff715      0xbffff735
0xbffff4c0:     0xbffff740      0xbffffbe1      0xbffffc07      0xbffffc16
0xbffff4d0:     0xbffffc68      0xbffffc9a      0xbffffca6      0xbffffcd2
(gdb) x /s 0xbffff60a
0xbffff60a:      "/home/ahmad/Courses/Assembly/chapter7/functest4"
```

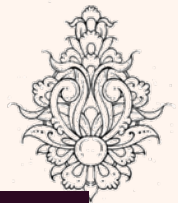


مثال

```
.section .data
output1:
.asciz "There are %d parameters:\n"
output2:
.asciz "%s\n"
.section .text
.globl _start
_start:
movl (%esp), %ecx
pushl %ecx
pushl $output1
call printf
addl $4, %esp
popl %ecx
movl %esp, %ebp
addl $4, %ebp
```

```
loop1:
pushl %ecx
pushl (%ebp)
pushl $output2
call printf
addl $8, %esp
popl %ecx
addl $4, %ebp
loop loop1
pushl $0
call exit
```

```
ahmad@ubuntu:~/Courses/Assembly/chapter7$ ./paramtest1 1 2 3 4 5
There are 6 parameters:
./paramtest1
1
2
3
4
5
```



مثال

```
.section .data
output:
.asciz "The area is: %f\n"
.section .bss
.lcomm result, 4
.section .text
.globl _start
_start:
nop
finit
pushl 8(%esp)
call atoi
addl $4, %esp
movl %eax, result
fldpi
filds result
fmul %st(0), %st(0)
fmul %st(1), %st(0)
fstpl (%esp)
pushl $output
call printf
addl $12, %esp
```

```
ahmad@ubuntu:~/Courses/Assembly/chapter7$ ./paramtest3 2
The area is: 12.566371
ahmad@ubuntu:~/Courses/Assembly/chapter7$ ./paramtest3 1
The area is: 3.141593
ahmad@ubuntu:~/Courses/Assembly/chapter7$
```

این تابع یک عدد صحیح به عنوان شعاع از طریق خط فرمان دریافت و مساحت را چاپ می‌کند