

دوره مهندسی معکوس نرم افزار

- Site: OnHexGroup.ir
- Youtube: @onhexgroup
- Telegram: onhex_ir
- Twitter: @onhexgroup
- Github: onhexgroup

نسخه ی ۳۲ و ۶۴ بیتی

پلتفرم: ویندوز

ارائه دهنده : **OnhexGroup**

■ به کمک توابع میتوانیم برنامه های بزرگ رو به قسمت های کوچک تقسیم کنیم:

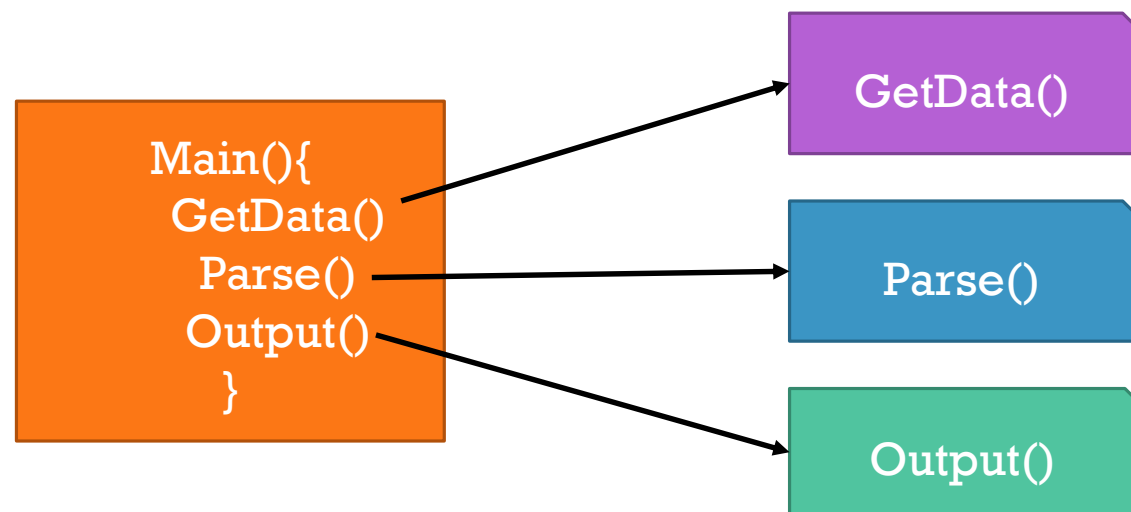
■ توسعه برنامه ساده تر میشه

■ نگهداری و عیب یابی برنامه ساده تر میشه.

■ تابع، روال ، رویه، متد، زیربرنامه، **Subroutine** ،
Procedure

Onhexgroup.ir

توابع



■ برای کار با توابع:

■ نحوه ی تعریف یک تابع

■ نحوه ی فراخوانی یک تابع

Twitter:Onhexgroup

کار با توابع

Main()

Caller

فراخوانی

GetData()

Callee

■ یک تابع میتواند از موارد زیر تشکیل بشه:

1. نام

2. پارامترهای ورودی

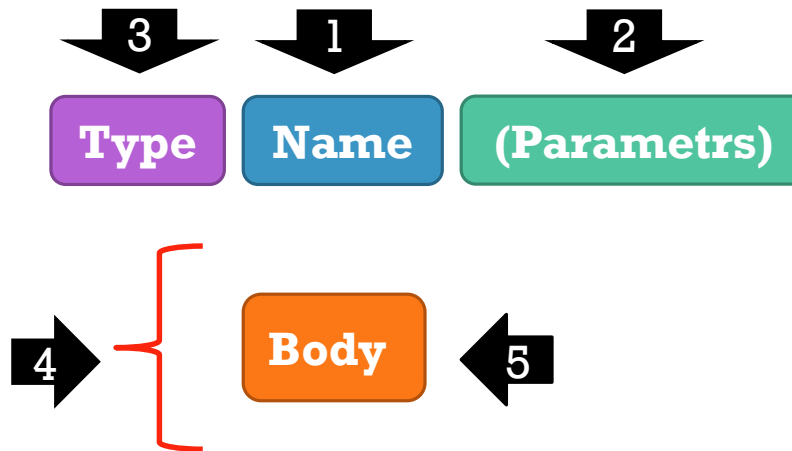
3. نوع

4. محدوده

5. بدنه

Twitter:Onhexgroup

موارد تشکیل
دهنده ی تابع



■ در اسمبلی برای تعریف توابع از ساختار زیر استفاده میکنیم:

Github: Onhexgroup

تعریف توابع

Function_name

Proc

Near/far

Function Body

Function_name

ENDP

■ در **C/C++** برای تعریف توابع از ساختار زیر استفاده میکنیم:
(**Inline assembly 32 bit**)

Github: Onhexgroup

تعریف توابع

Return_Type

Function_name

(Parameters)

{

Function Body

}

Onhexgroup.ir

فراخوانی توابع

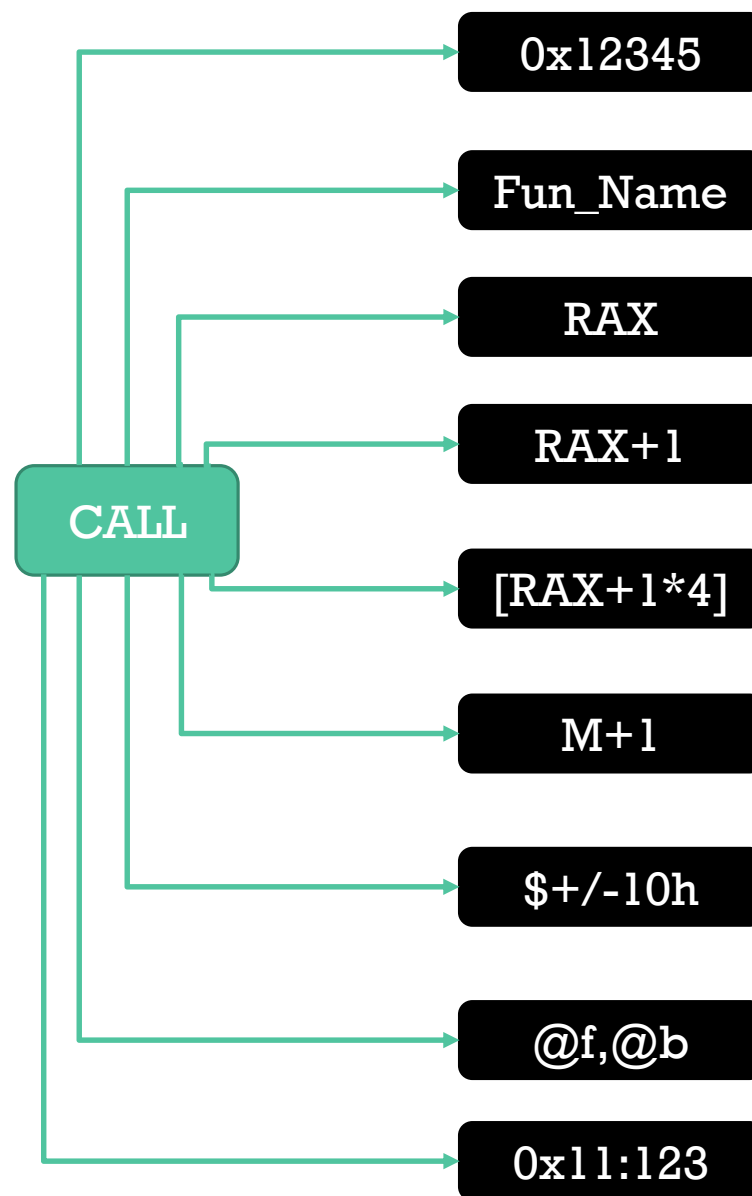
- برای فراخوانی توابع از دستور **Call** استفاده میکنیم.
- عملکرد این دستور:
- آدرس بازگشت را در پشته قرار میدهد
- اجرای برنامه رو به تابع فراخوانی شده، میبره. (سر تابع)

CALL R/M/IMM

- انواع دستور **CALL**:
- **NEAR** یا نزدیک : در همان سگمنت کد
- **FAR** یا دور: در یک سگمنت دیگه
- **Inter-privilege-level far** : همان نوع **FAR** فقط سطح دسترسی فرق داره.
- **Task switch**: در یک تسک دیگه (در ۶۴ بیتی پشتیبانی نمیشه)

Onhexgroup.ir

فراخوانی توابع



@@:
Mov al,bl

Call @b
Call @f

@@:
MOV ax, bx

Onhexgroup.ir

بازگشت از توابع

- برای بازگشت از توابع از دستور **RET** استفاده میکنیم.

- عملکرد این دستور:

- آدرس بازگشت رو از پشته در **EIP/RIP** قرار میده

- اجرای برنامه رو به آدرس مورد نظر، میبره. (آدرس بازگشت)

RET IMM16

- انواع دستور **RET**:

- **NEAR** یا نزدیک : در همان سگمنت کد

- **FAR** یا دور: در یک سگمنت دیگه

- **Inter-privilege-level far** : همان نوع **FAR** فقط

سطح دسترسی فرق داره.

■ قرارداد فراخوانی یا **Calling Convention** یک قرار داد بین **Caller** و **Callee** هستش که موارد زیر رو مشخص میکنه:

■ نحوه ی ارسال آرگومانها

■ نحوه ی بازگردانی نتایج

■ مشخص کردن مسئول پاکسازی و مدیریت پشته

Onhexgroup.ir

قرارداد فراخوانی



Onhexgroup.ir

قراردادهای فراخوانی در ویندوز

■ قراردادهای فراخوانی در نسخه ی ۳۲ بیتی:

Cdecl ■

Stdcall ■

Fastcall ■

Thiscall ■

■ قراردادهای فراخوانی در نسخه ی ۶۴ بیتی:

Windows x64 Calling Convention ■

قرارداد فراخوانی در ۳۲ بیتی

■ نوع **Cdecl (C Declaration)**:

■ ترتیب ارسال پارامترها: از راست به چپ

■ ارسال پارامترها: از طریق پشته.

■ سائز: براساس دستور **PUSH** - معمولاً **Dword**

■ بازگرداندن نتیجه: از طریق **EAX**

■ سائز بسته به نوع دارد. بایت **AL** و ...

■ پاکسازی پشته: بر عهده **Caller**

■ فراخوانی پیش فرض در زبان های **C** و **C++**

■ در توابع **Varargs (Variadic)** استفاده میشه:

```
Add(int a , int b);
```

```
Printf("onhexgroup");
```

```
Printf("Result of %d+%d= %d",a,b,c)
```

قرارداد فراخوانی در ۳۲ بیتی

■ نوع Stdcall (Standard Call):

- ترتیب ارسال پارامترها: از راست به چپ
- ارسال پارامترها: از طریق پشته.
- سائز: براساس دستور **PUSH** - معمولاً **Dword**
- بازگرداندن نتیجه: از طریق **EAX**
- سائز بسته به نوع دارد. بایت **AL=** و ...
- پاکسازی پشته: بر عهده **Callee**
- این توافق به طور گسترده در **Windows API** استفاده میشه و به همین دلیل معمولاً به عنوان **WinAPI calling convention** هم شناخته میشه.
- چون تابع خودش مسئول پاک کردن پشته هستش، بنابراین فراخوانی توابع ساده هستش.

قرارداد فراخوانی در ۳۲ بیتی

■ نوع **Fastcall**:

- ترتیب ارسال آرگومانها : از راست به چپ
- ارسال پارامترها: دو پارامتر اول در **ECX** و **EDX**، و بقیه از طریق پشته.
- سائز: بسته به نوع داده + دستور **PUSH**
- بازگرداندن نتیجه: : از طریق **EAX**
- سائز بسته به نوع دارد. بایت **AL** و ...
- پاکسازی پشته: بر عهده **Callee**
- برای بهبود عملکرد و زمانیکه نیاز به سرعت بالاست استفاده میشه.

Onhexgroup.ir

قرارداد فراخوانی در ۳۲ بیتی

■ نوع **thiscall**:

■ برای متدهای کلاسها استفاده میشه.

■ اشاره گر **this** به طور ضمنی در **ecx** قرار میگیره. (سایز
Dword)

■ نوع **Windows x64 Calling Convention**

■ **X64 fastcall**

■ ترتیب ارسال پارامترها: از راست به چپ

■ ارسال پارامترها: (سایز بسته به مقدار آرسالی دارد)

■ پارامتر اول: **RCX** پارامتر دوم: **RDX**

■ پارامتر سوم: **R8** پارامتر چهارم: **R9**

■ بقیه: پشته

■ بازگرداندن نتیجه: از طریق **RAX**

■ **Byte=ax – word,dword=eax,qword=rax**

■ پاکسازی پشته: **Caller**

■ با توجه به اینکه از رجیسترها استفاده میکند، باعث بهبود عملکرد میشود.

Onhexgroup.ir

قرارداد فراخوانی در
۶۴ بیتی

متغیرهای محلی

- متغیرهای محلی، متغیرهایی هستند که در داخل بدنه ی یک تابع تعریف میشن و تا زمان اجرای تابع، عمر دارن.
- در کدهای زیر متغیر **r**، یک متغیر محلی است.

```
int ADD(int a, int b){  
  
    int r;  
  
    r= a+b;  
  
    return r;  
  
}
```

- متغیرهای محلی داخل پشته ذخیره میشن.

■ در ۳۲ بیتی برای دسترسی به متغیرهای محلی از **EBP** استفاده میکنیم.

```
Main(){  
    int r=add(1,2);  
}  
  
Int add(int a,int b){  
    int r=0;  
    r=a+b;  
    return r;  
}
```

Onhexgroup.ir

متغیرهای محلی در
نسخه ی ۳۲ بیتی

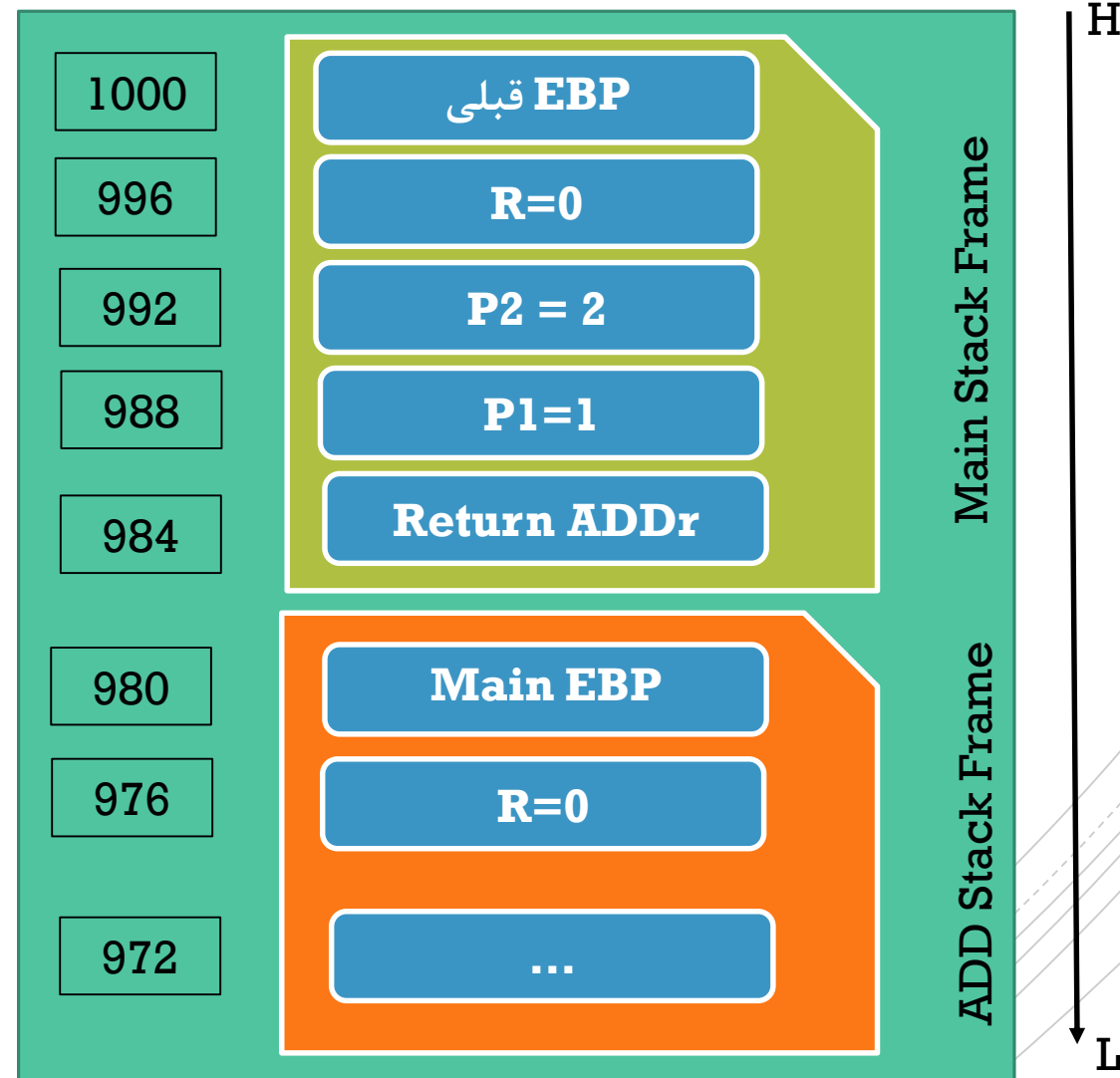
Mov dword ptr [ebp-4],0

Function Frame

EBP

EBP-4

ESP



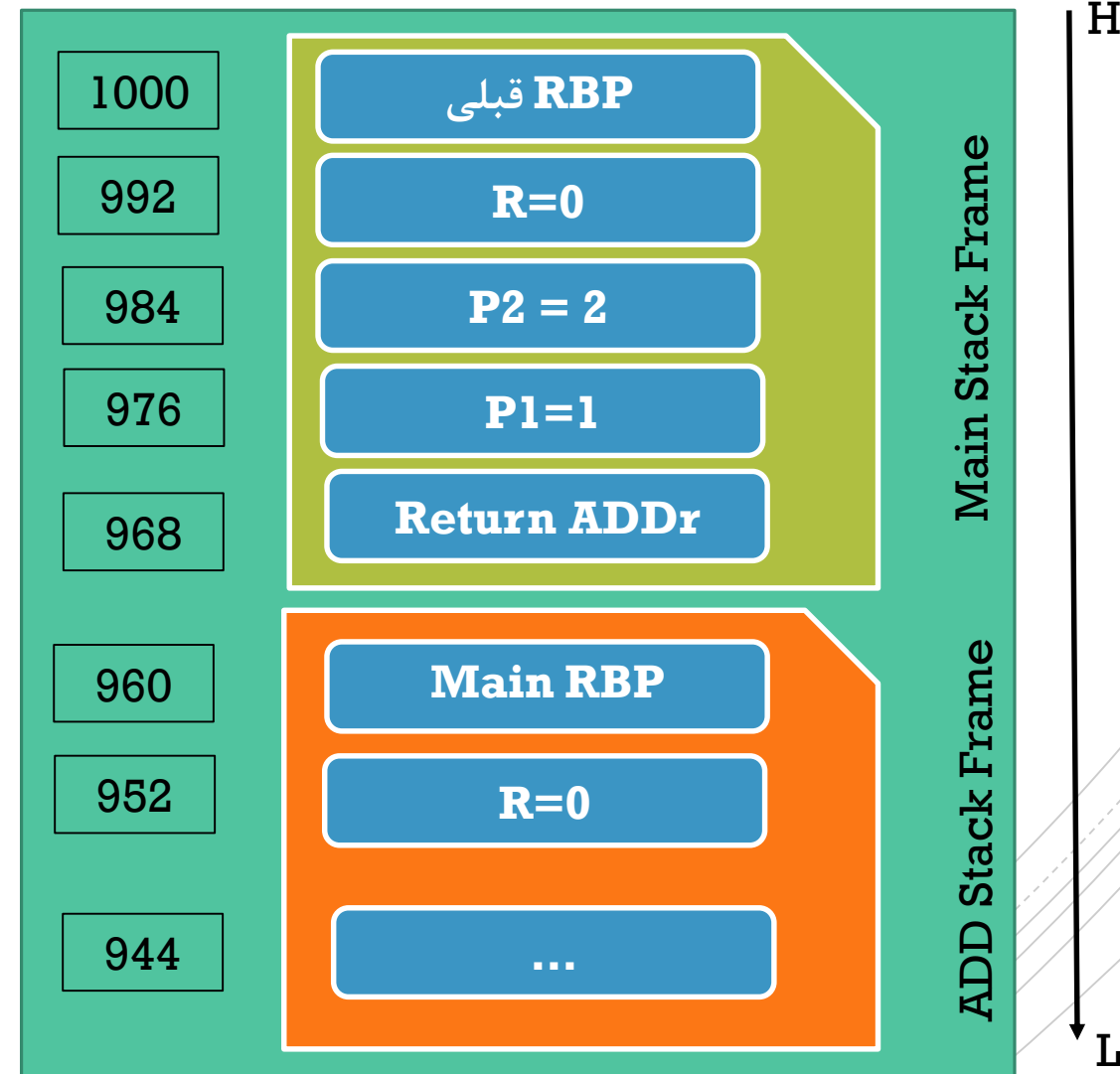
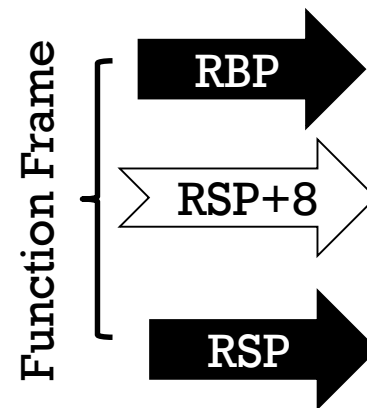
■ در 64 بیتی برای دسترسی به متغیرهای محلی از **RSP** استفاده میکنیم.

```
Main(){  
    int r=add(1,2);  
}  
  
Int add(int a,int b){  
    int r=0;  
    r=a+b;  
    return r;  
}
```

Onhexgroup.ir

متغیرهای محلی در
نسخه ی 64 بیتی

Mov dword ptr [RSP+8],0



Onhexgroup.ir

Epilog و Prolog

00401000	55	push ebp
00401001	8BEC	mov ebp,esp
00401003	83EC 0C	sub esp,C
00401006	C745 FC 00000000	mov dword ptr ss:[ebp-4],0
0040100D	C745 F8 01000000	mov dword ptr ss:[ebp-8],1
00401014	C745 F4 02000000	mov dword ptr ss:[ebp-C],2
00401018	8B45 08	mov eax,dword ptr ss:[ebp+8]
0040101E	0345 0C	add eax,dword ptr ss:[ebp+C]
00401021	0345 10	add eax,dword ptr ss:[ebp+10]
00401024	0345 14	add eax,dword ptr ss:[ebp+14]
00401027	0345 18	add eax,dword ptr ss:[ebp+18]
0040102A	0345 F8	add eax,dword ptr ss:[ebp-8]
0040102D	0345 F4	add eax,dword ptr ss:[ebp-C]
00401030	8B45 FC	mov dword ptr ss:[ebp-4],eax
00401033	8B45 FC	mov eax,dword ptr ss:[ebp-4]
00401036	8BE5	mov esp,ebp
00401038	5D	pop ebp
00401039	C3	ret

PROLOG

BODY

EPILOG

Prolog در ۳۲ بیتی

■ **Prolog** شامل دستوراتی هستش که رجیسترها و فضای پشته رو برای اجرای تابع آماده میکنه.

■ در ۳۲ بیتی معمولا ۳ کار انجام میده:

■ ۱- مقدار **EBP** رو داخل پشته قرار میده.

■ ۲- مقدار فعلی **ESP** رو به **EBP** انتقال میده.

■ ۳- مقدار **ESP** رو برای متغیرهای محلی و پارامترها بروز میکنه

```
push ebp
mov ebp, esp
sub esp, C
```

■ قدم اول: مقدار **EBP** رو داخل پشته قرار میده. (**EBP** قبلی)

■ از **EBP** برای دسترسی به متغیرهای محلی و پارامترها استفاده میکنیم.

■ میتونیم آدرس بازگشت رو پیدا کنیم.

■ میتونیم قاب پشته رو مشخص کنیم.

■ میتونیم ازش در **Call Stack** استفاده کنیم.

Onhexgroup.ir

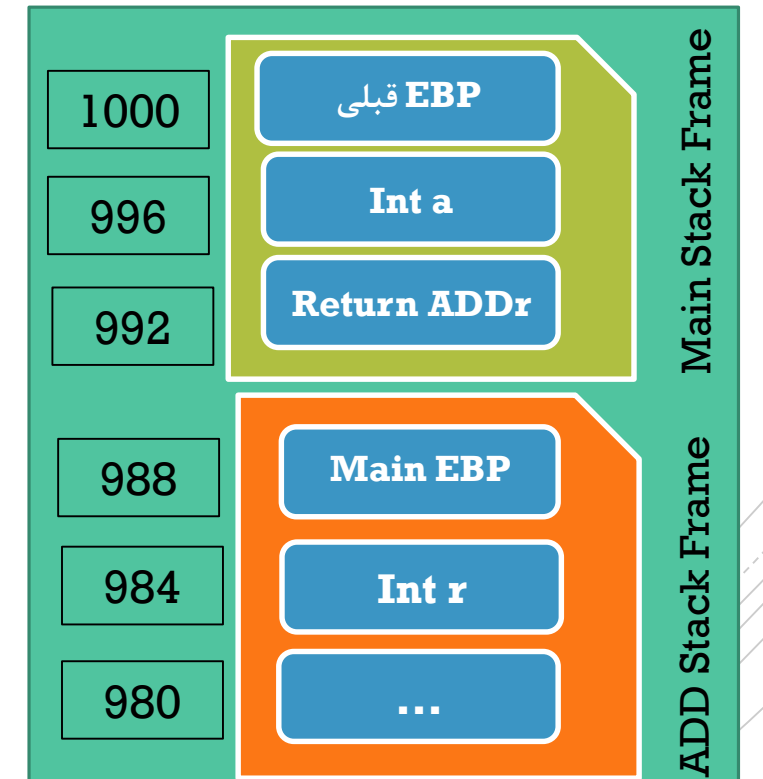
Prolog در ۳۲ بیتی

```
main(){  
    int a=0;  
    a=add(1,2);  
}
```

```
Int add(int a,int b){  
    int r= a+b;  
    return r;  
}
```

EBP →

ESP →

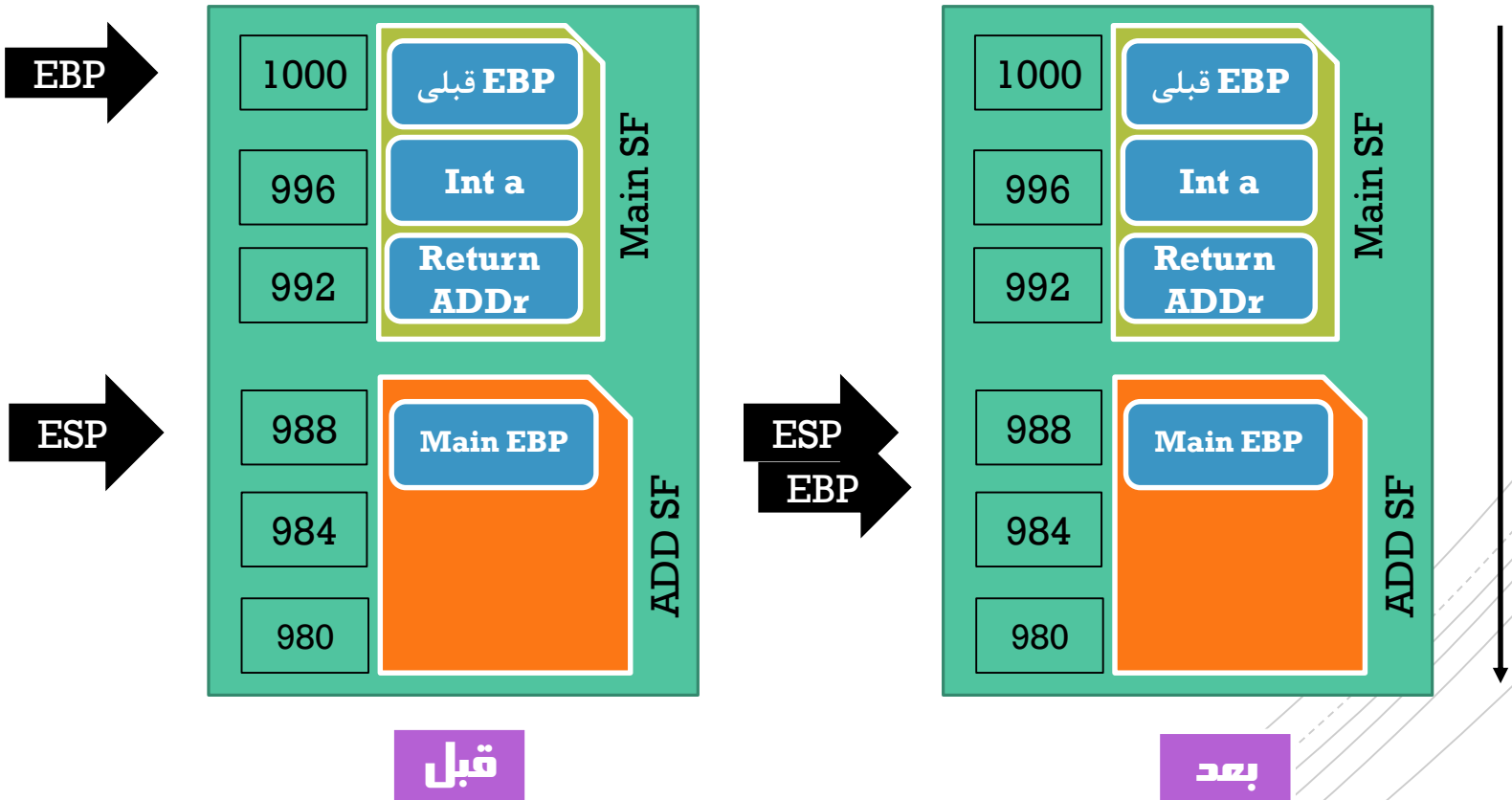


- قدم دوم: مقدار فعلی **ESP** رو به **EBP** انتقال میده.
- (**EBP** بروز میشه)

■ باز اهمیت **EBP**.

Onhexgroup.ir

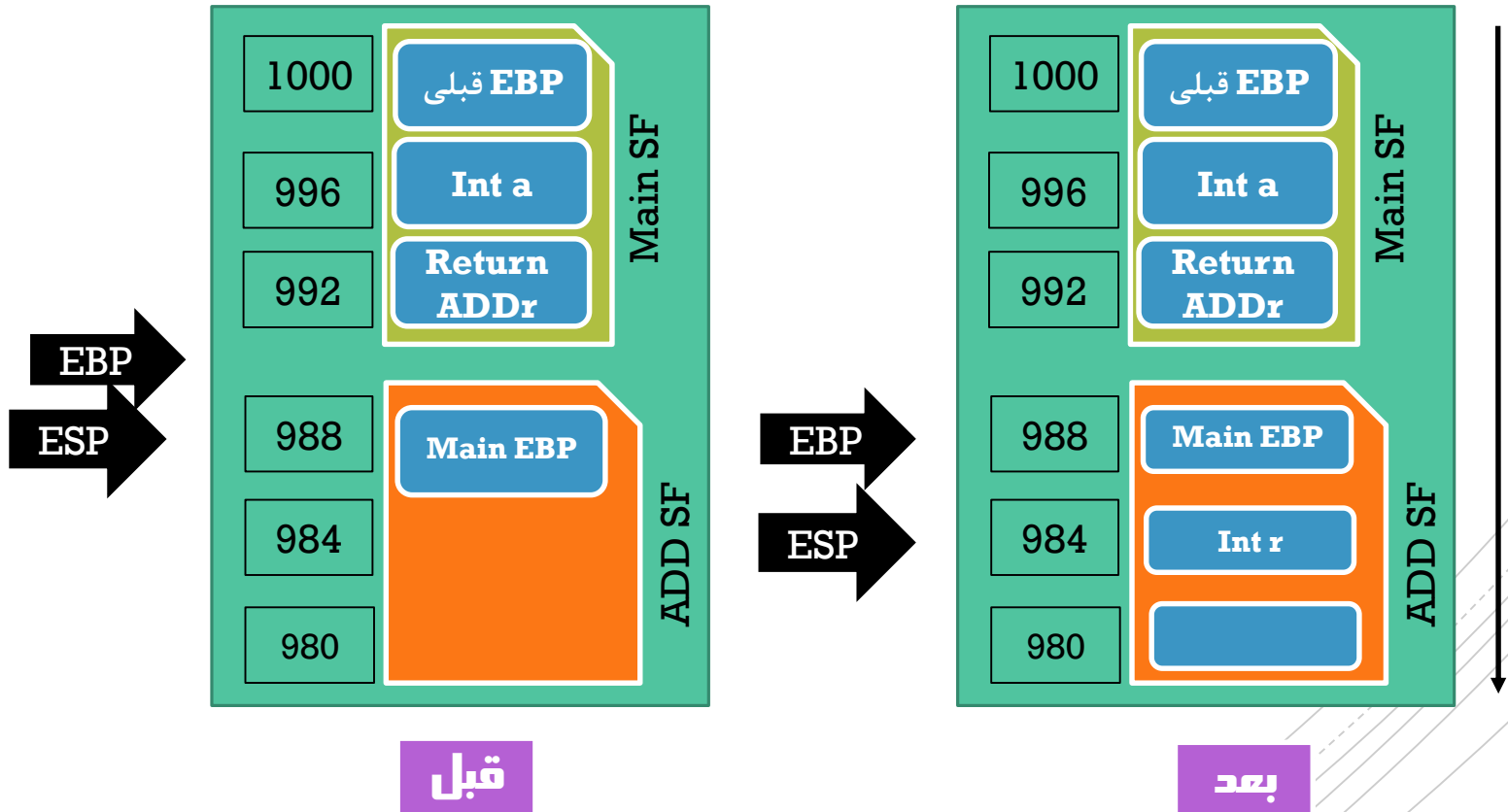
Prolog در ۳۲ بیتی



- قدم سوم: مقدار **ESP** رو برای متغیرهای محلی و پارامترهای ارسال شده از طریق رجیستر بروز میکنه.
- همه چیز برای اجرای تابع آماده باشه.

Onhexgroup.ir

Prolog در ۳۲ بیتی



Onhexgroup.ir

Epilog در ۳۲ بیتی

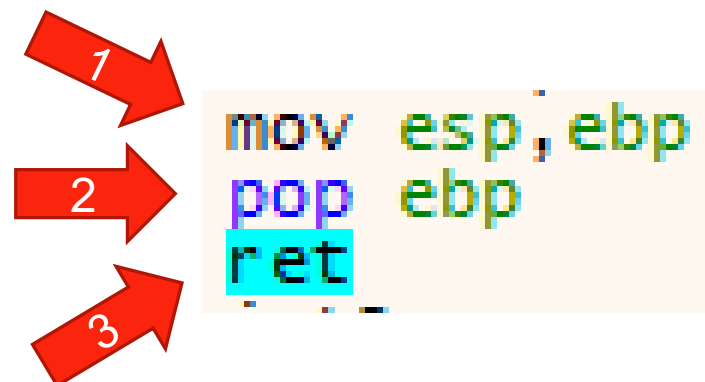
■ **Epilog** شامل دستوراتی هستش که رجیسترها و فضای پشته رو به حالت قبل برمیگردونه.

■ در ۳۲ بیتی معمولا ۳ کار انجام میشه: (معکوس **Prolog**)

■ ۱- مقدار **EBP** رو به **ESP** انتقال میشه

■ ۲- مقدار **EBP** قبلی رو به **EBP** انتقال میشه.

■ ۳- از تابع برمیگرده

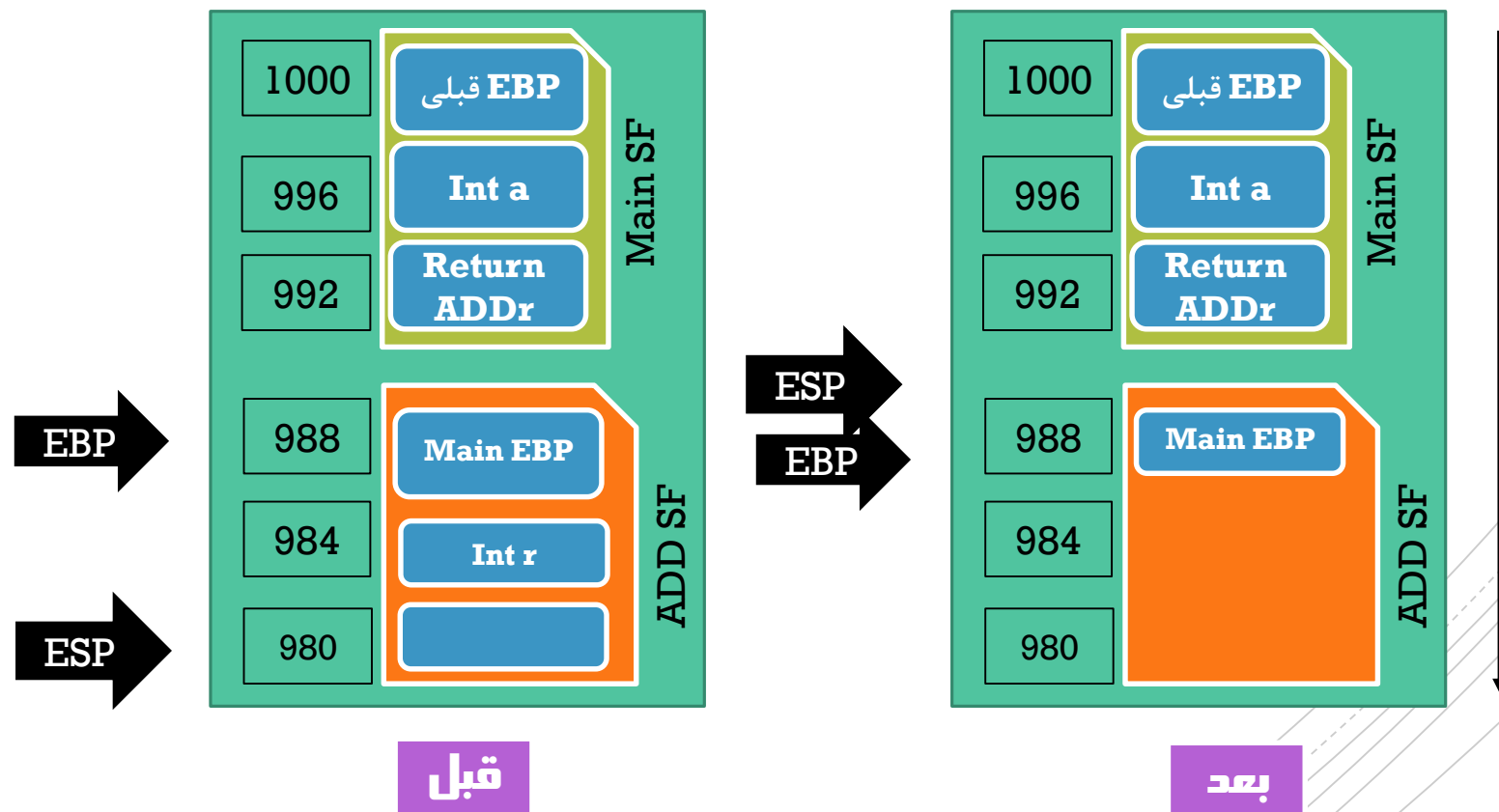


```
mov esp, ebp  
pop ebp  
ret
```

- قدم اول: مقدار **EBP** رو به **ESP** انتقال میده
- پارامترها و متغیرهای محلی رو حذف میکنه.

Onhexgroup.ir

Epilog در ۳۲ بیتی



Onhexgroup.ir

Epilog در ۳۲ بیتی



■ مقدار **EBP** قبلی رو به **EBP** انتقال میده.

■ امکان کار با تابع **Caller** رو داشته باشیم.



قبل

بعد

■ از تابع برمیگردد

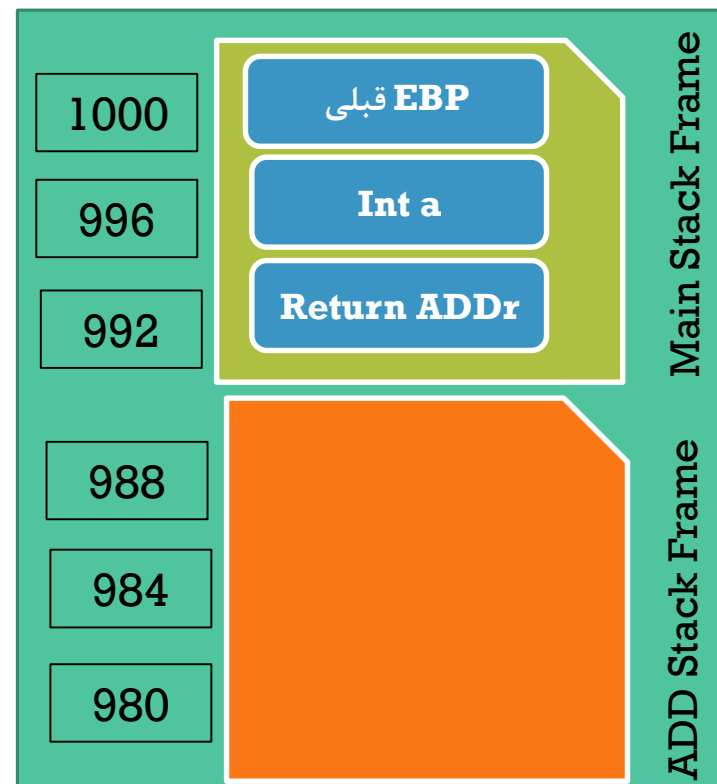
■ برمیگردیم به آدرس بازگشت

Onhexgroup.ir

Epilog در ۳۲ بیتی

EBP →

ESP →



EIP ←

Prolog در ۶۴ بیتی

■ معمولاً ۳ کار انجام میدهد:

■ ۱- ذخیره پارامترهای ارسالی از طریق رجیسترها به پشت.

■ ۲- ذخیره ی رجیسترهای **non-volatile** که قراره تغییر پیدا کنن.

■ ۳- ایجاد فضا برای متغیرهای محلی و پارامترها

1

3

```
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov eax,dword ptr ss:[rsp+28]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

■ قدم اول: ذخیره پارامترهای ارسالی از طریق رجیسترها به پشته.

■ برای اینکه از پارامترها استفاده کنیم باید داخل پشته قرار بگیره.

■ در x64 از **x64 Fastcall** استفاده میکنیم:

■ **RCX,RDX,R8,R9**

■ اصطلاحاً به این قسمت **Parameter Homing Area**

Onhexgroup.ir

Prolog در ۶۴ بیتی

```
sub rsp,38
mov dword ptr ss:[rsp+20],0
mov edx,2
mov ecx,1
call <prolog_epilog_64bit.int __cdecl add(int, int)>
mov dword ptr ss:[rsp+20],eax
xor eax,eax
add rsp,38
ret
```

```
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov eax,dword ptr ss:[rsp+28]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

Prolog در ۶۴ بیتی

■ قدم دوم: ذخیره ی رجیسترهای **Non-volatile** که قراره تغییر پیدا کنن.

■ رجیسترهای **Volatile** (فرار): رجیسترهایی که توابع

میتونن بدون ذخیره مقادیرشون اونارو تغییر بدن. از

جمله: **RAX, RCX, RDX, R8, R9, R10, R11**

■ رجیسترهای **Non-volatile** (غیر فرار): رجیسترهایی

که توابع برای استفاده از اونا باید مقادیرشون رو ذخیره

کنن و بعد از پایان تابع، باید مقادیرشون بازیابی بشن. از

جمله: **RBX, RDI, RSI, RBP, RSP, R12,**

R13, R14 , R15

```
→ push rbx
sub rsp,20
xor eax,eax
mov rbx,rcx
test rcx,rcx
je ucrtbased.7FFAD265682D
xor r8d,r8d
call <ucrtbased._mbsnbcnt_1>
mov eax,eax
add rax,rbx
add rsp,20
→ pop rbx
ret
```

Onhexgroup.ir

Prolog در ۶۴ بیتی

■ قدم سوم: ایجاد فضا برای متغیرهای محلی و پارامترها

■ این فضا مجموعه ای از مقادیر زیر است:

■ ۸ بایت فاصله ی بین آدرس بازگشت و متغیرهای محلی

■ سائز متغیرهای محلی

■ ۳۲ (۴*۸) بایت پارامترهای ارسالی (در صورت وجود)

■ رعایت قانون تراز ۱۶ بایتی



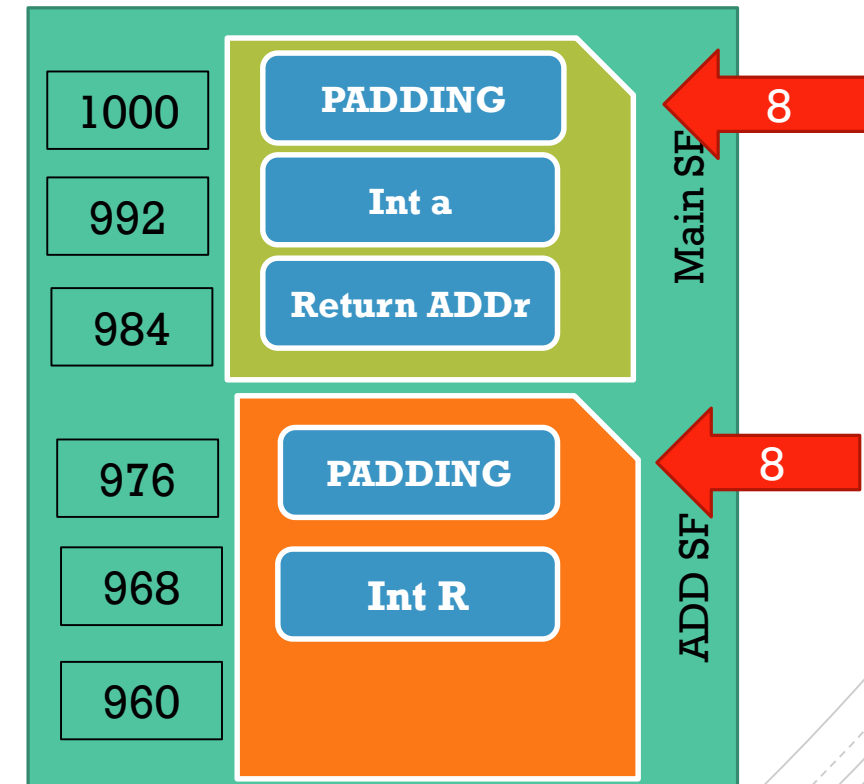
```
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov eax,dword ptr ss:[rsp+28]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```


- یک فاصله (PADDING) ۸ بایتی بین آدرس برگشت و متغیرها وجود دارد.

Onhexgroup.ir

Prolog در ۶۴ بیتی

```
main(){  
    int a=0;  
    a=add(1,2);  
}  
  
Int add(int a,int b){  
    int r= a+b;  
    return r;  
}
```



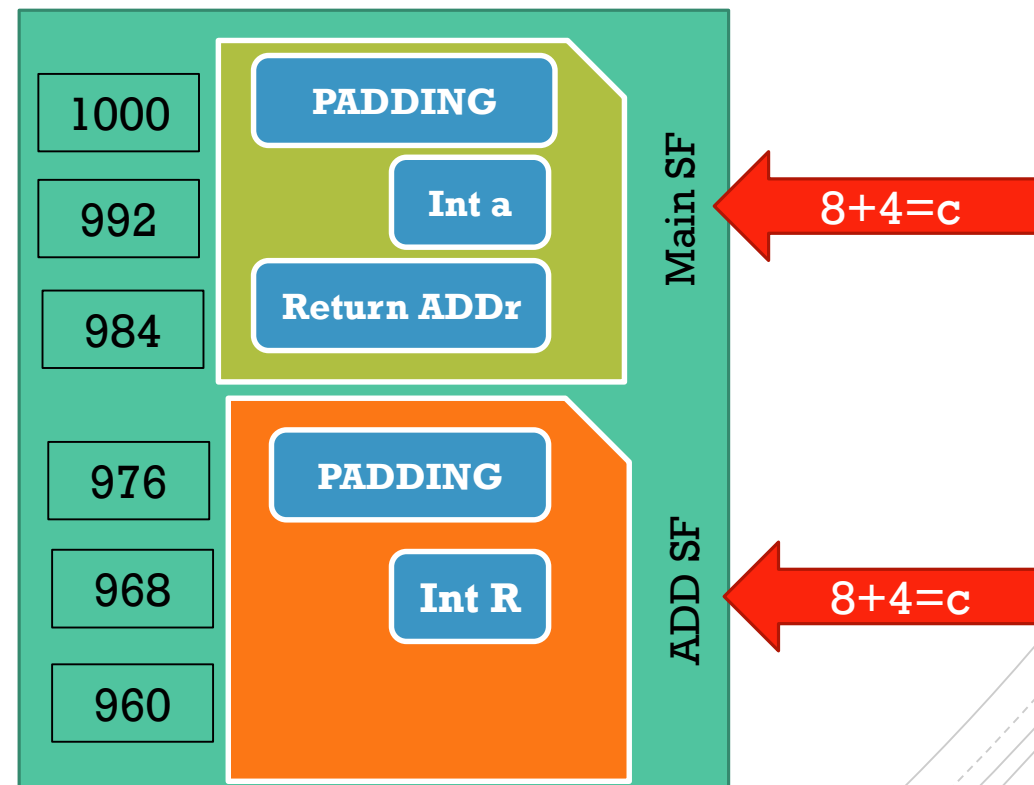
■ سائز متغیرهای محلی.

Onhexgroup.ir

Prolog در ۶۴ بیتی

```
main(){  
    int a=0;  
    a=add(1,2);  
}
```

```
Int add(int a,int b){  
    int r= a+b;  
    return r;  
}
```



۳۲ (۸*۴) بایت پارامترهای ارسالی از طریق رجیسترها (در صورت وجود)

توابع **Leaf**: توابعی که، تابع دیگه ای رو فراخوانی نمیکنن

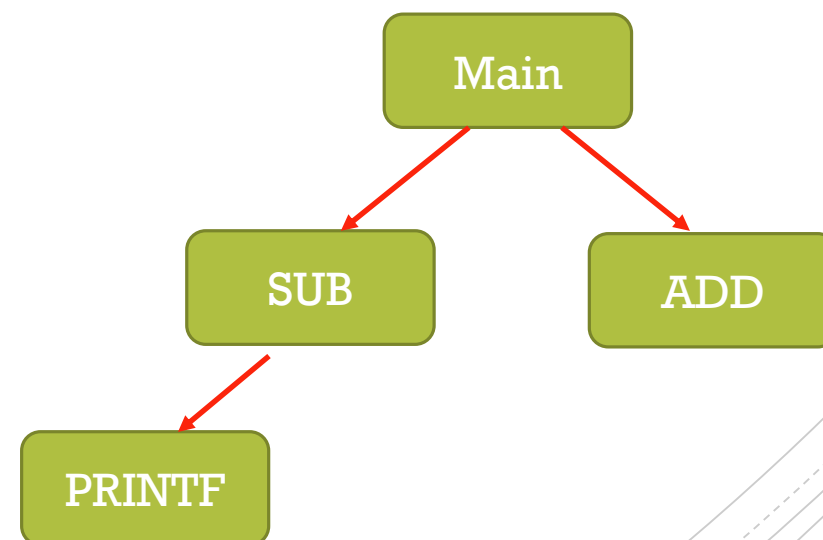
توابع **Non-Leaf**: توابعی که، توابع دیگه رو فراخوانی میکنند.

اصطلاحاً به این فضای رزرو شده **Shadow Space** میگویند

Prolog در ۶۴ بیتی

Onhexgroup.ir

```
Main(){
    add(1,2);
    sub(1,2);
}
Add(int a,int b){
    a+b;
}
Sub(int a,int b){
    printf(b-a);
}
```



■ رعایت قانون تراز ۱۶ تایی (16-byte aligned)

■ مقادیر داخل پشته (پارامترها و متغیرهای محلی) باید تراز ۱۶ تایی رو حفظ کنن. یعنی در بسته های ۱۶ بایتی باشن.

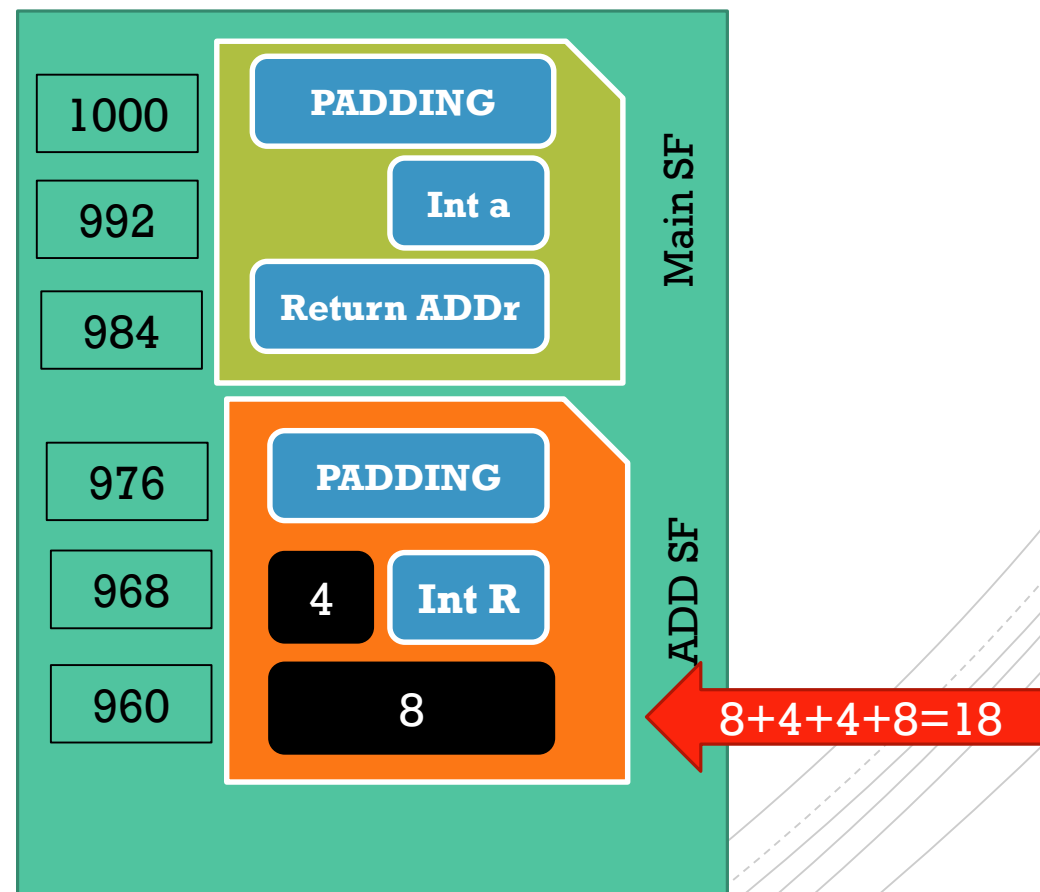
■ برای دستورات SIMD

Onhexgroup.ir

Prolog در ۶۴ بیتی

```
main(){  
    int a=0;  
    a=add(1,2);  
}
```

```
Int add(int a,int b){  
    int r= a+b;  
    return r;  
}
```



Epilog در ۶۴ بیتی

■ معمولا ۳ کار انجام میدهد:

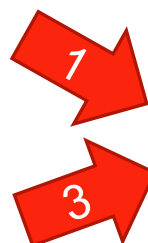
■ ۱- پاکسازی فضای پشته اختصاص داده شده به

متغیرهای محلی و پارامترها

■ ۲- بازیابی رجیسترهای **non-volatile**

■ ۳- بازگشت به آدرس برگشت

```
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov eax,dword ptr ss:[rsp+28]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```



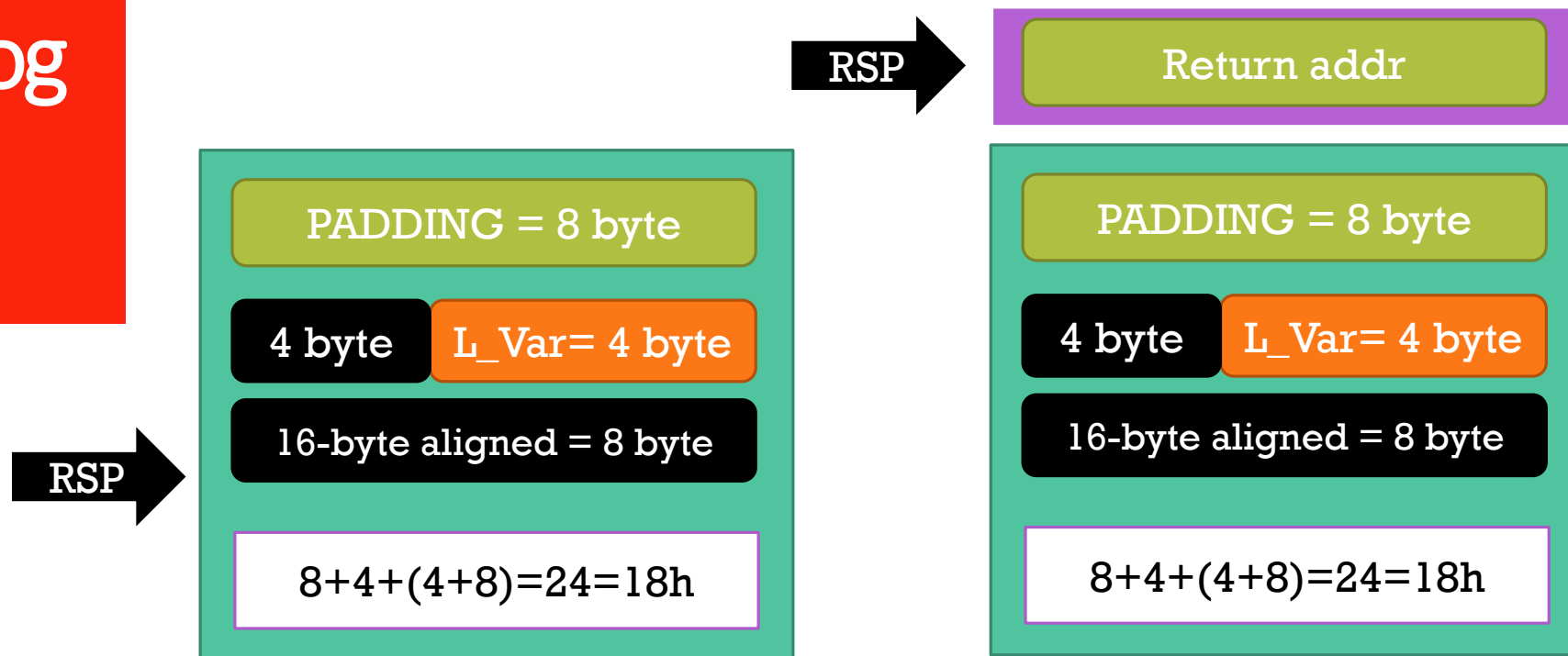
قدم اول: پاکسازی فضای پشته اختصاص داده شده به متغیرهای محلی و پارامترها

Onhexgroup.ir

Epilog در ۶۴ بیتی

```
Int add(int a,int b){  
    int r= a+b;  
    return r;  
}
```

```
mov dword ptr ss:[rsp+10],edx  
mov dword ptr ss:[rsp+8],ecx  
sub rsp,18  
mov eax,dword ptr ss:[rsp+28]  
mov ecx,dword ptr ss:[rsp+20]  
add ecx,eax  
mov eax,ecx  
mov dword ptr ss:[rsp],eax  
mov eax,dword ptr ss:[rsp]  
add rsp,18  
ret
```



■ قدم دوم: بازیابی رجیسترهای Non-volatile

Onhexgroup.ir

Prolog در ۶۴ بیتی



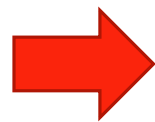
```
push rbx
sub rsp,20
xor eax,eax
mov rbx,rcx
test rcx,rcx
je ucrtbased.7FFAD265682D
xor r8d,r8d
call <ucrtbased._mbsnbcnt_1>
mov eax,eax
add rax,rbx
add rsp,20
pop rbx
ret
```



■ بازگشت به آدرس برگشت

Onhexgroup.ir

Epilog در ۶۴ بیتی



```
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov eax,dword ptr ss:[rsp+28]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```


Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

mov amd64.reg.15, 0x401513
push 2
call <prolog_epilog.int __cdecl add2(int)>
add esp, 4

0019FEE0	00000001
0019FEE4	0019FF04
0019FEE8	00401513
0019FEEC	00000000

ESP

پارامترهای تابع (از آخر)

push 2
call <prolog_epilog.int __cdecl add2(int)>
add esp, 4

0019FEDC	00000002	
0019FEE0	00000001	
0019FEE4	0019FF04	
0019FEE8	00401513	ret

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push 2  
call <prolog_epilog.int __cdecl add2(int)>  
add esp,4
```

0019FEDC	00000002	
0019FEE0	00000001	
0019FEE4	0019FF04	
0019FEE8	00401513	ret

```
push ebp  
mov ebp,esp  
sub esp,8  
mov dword ptr ss:[ebp-4],0  
mov dword ptr ss:[ebp-8],1  
mov eax,dword ptr ss:[ebp+8]  
add eax,dword ptr ss:[ebp-4]  
add eax,dword ptr ss:[ebp-8]  
mov dword ptr ss:[ebp-4],eax  
mov eax,dword ptr ss:[ebp-4]  
mov esp,ebp  
pop ebp  
ret
```

0019FED8	0040113C	return to prolog
0019FEDC	00000002	
0019FEE0	00000001	

پارامترهای تابع [از آخر]

Return ADDR

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FED8	0040113C	return to prolo
0019FEDC	00000002	
0019FEE0	00000001	



```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FED4	0019FEE4	
0019FED8	0040113C	re
0019FEDC	00000002	
0019FEE0	00000001	

ESP

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FED4	0019FEE4	
0019FED8	0040113C	re
0019FEDC	00000002	
0019FEE0	00000001	

```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FED4	0019FEE4	
0019FED8	0040113C	re
0019FEDC	00000002	
0019FEE0	00000001	

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

ESP

EBP

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FED4	0019FEE4	
0019FED8	0040113C	re
0019FEDC	00000002	
0019FEE0	00000001	

```
push ebp
mov ebp, esp
sub esp, 8
mov dword ptr ss:[ebp-4], 0
mov dword ptr ss:[ebp-8], 1
mov eax, dword ptr ss:[ebp+8]
add eax, dword ptr ss:[ebp-4]
add eax, dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4], eax
mov eax, dword ptr ss:[ebp-4]
mov esp, ebp
pop ebp
ret
```

0019FECC	0019FEE0	
0019FED0	00000001	
0019FED4	0019FEE4	
0019FED8	0040113C	r
0019FEDC	00000002	
0019FEE0	00000001	

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

EBP

ESP

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov ebp,esp
sub esp,8
mov dword ptr ss:[ebp-4],0
mov dword ptr ss:[ebp-8],1
mov eax,dword ptr ss:[ebp+8]
add eax,dword ptr ss:[ebp-4]
add eax,dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4],eax
mov eax,dword ptr ss:[ebp-4]
mov esp,ebp
pop ebp
ret
```

0019FECC	0019FEE0	
0019FED0	00000001	
0019FED4	0019FEE4	
0019FED8	0040113C	r
0019FEDC	00000002	
0019FEE0	00000001	



```
push ebp
mov ebp,esp
sub esp,8
mov dword ptr ss:[ebp-4],0
mov dword ptr ss:[ebp-8],1
mov eax,dword ptr ss:[ebp+8]
add eax,dword ptr ss:[ebp-4]
add eax,dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4],eax
mov eax,dword ptr ss:[ebp-4]
mov esp,ebp
pop ebp
ret
```

0019FECC	0019FEE0	
0019FED0	00000000	
0019FED4	0019FEE4	
0019FED8	0040113C	r
0019FEDC	00000002	
0019FEE0	00000001	

EBP

ESP

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

LOCAL VAR 1

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov ebp,esp
sub esp,8
mov dword ptr ss:[ebp-4],0
mov dword ptr ss:[ebp-8],1
mov eax,dword ptr ss:[ebp+8]
add eax,dword ptr ss:[ebp-4]
add eax,dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4],eax
mov eax,dword ptr ss:[ebp-4]
mov esp,ebp
pop ebp
ret
```

0019FECC	0019FEE0	
0019FED0	00000001	
0019FED4	0019FEE4	
0019FED8	0040113C	r
0019FEDC	00000002	
0019FEE0	00000001	

```
push ebp
mov ebp,esp
sub esp,8
mov dword ptr ss:[ebp-4],0
mov dword ptr ss:[ebp-8],1
mov eax,dword ptr ss:[ebp+8]
add eax,dword ptr ss:[ebp-4]
add eax,dword ptr ss:[ebp-8]
mov dword ptr ss:[ebp-4],eax
mov eax,dword ptr ss:[ebp-4]
mov esp,ebp
pop ebp
ret
```

0019FECC	00000001	
0019FED0	00000000	
0019FED4	0019FEE4	
0019FED8	0040113C	
0019FEDC	00000002	
0019FEE0	00000001	

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

LOCAL VAR 1

LOCAL VAR 2

Onhexgroup.ir

ترسیم قاب پشته
در ۳۲ بیتی

```
push ebp
mov  ebp,esp
sub  esp,8
mov  dword ptr ss:[ebp-8],ecx
mov  dword ptr ss:[ebp-4],0
mov  eax,dword ptr ss:[ebp-8]
add  eax,dword ptr ss:[ebp-4]
```

EBP

▪ در حالت Fastcall

پارامترهای تابع (از آخر)

Return ADDR

SAVE MAIN EBP

LOCAL VAR 1

Register Param

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

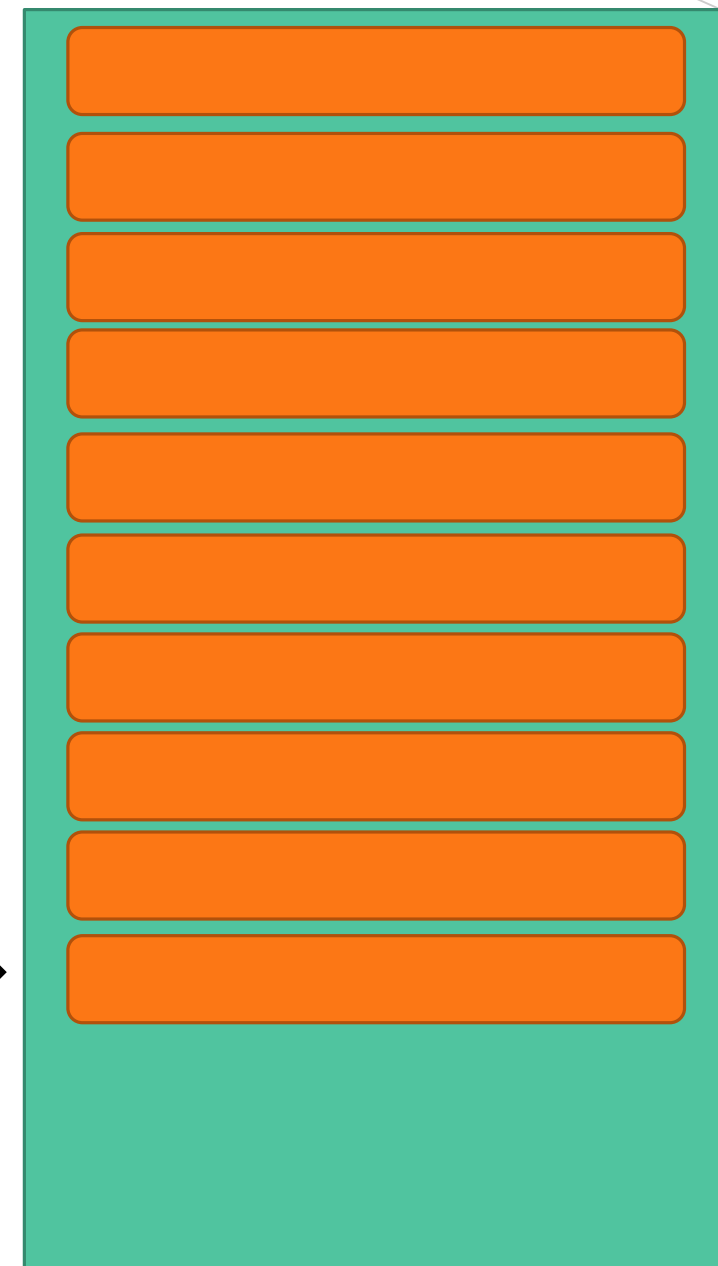
```
sub rsp,48
mov dword ptr ss:[rsp+30],0
mov dword ptr ss:[rsp+28],6
mov dword ptr ss:[rsp+20],5
mov r9d,4
mov r8d,3
mov edx,2
mov ecx,1
call <stack64.int __cdecl add(int, int, int, int, int, int)>
```

0000000000014FE08	000000001400013C9
0000000000014FE10	00003518000000001
0000000000014FE18	00007FFFE0E94219
0000000000014FE20	00000000000000000
0000000000014FE28	00000000140001A1D
0000000000014FE30	00000000000000001
0000000000014FE38	0000000000005D9C80
0000000000014FE40	0000000000005D60F0
0000000000014FE48	00000000140001180
0000000000014FE50	00000000000000000
0000000000014FE58	000000001400012EE
0000000000014FE60	00000000140001100



0000000000014FDC0	00000000000000000
0000000000014FDC8	00000000000000000
0000000000014FDD0	00000000100000001
0000000000014FDD8	00000000140001179
0000000000014FDE0	00000000100000000
0000000000014FDE8	00000000140001195
0000000000014FDF0	000027AB000000000
0000000000014FDF8	BC000400000000000
0000000000014FE00	FFFA32031F88FBFF
0000000000014FE08	000000001400013C9

RSP



Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

```
sub rsp,48
mov dword ptr ss:[rsp+30],0
mov dword ptr ss:[rsp+28],6
mov dword ptr ss:[rsp+20],5
mov r9d,4
mov r8d,3
mov edx,2
mov ecx,1
call <stack64.int __cdecl add(int, int, int, int, int, int)>
```

000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000000
000000000014FDE8	0000000140001195
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	00000001400013C9



000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000000
000000000014FDE8	0000000140001195
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	00000001400013C9

RSP

Local var 1

```

sub    rsp,48
mov    dword ptr ss:[rsp+30],0
mov    dword ptr ss:[rsp+28],6
mov    dword ptr ss:[rsp+20],5
mov    r9d,4
mov    r8d,3
mov    edx,2
mov    ecx,1
call   <stack64.int __cdecl add(int, int, int, int, int, int)>

```

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000000
000000000014FDE8	0000000140001195
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000000
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

RSP

Local var 1

Param 6

```

sub    rsp,48
mov    dword ptr ss:[rsp+30],0
mov    dword ptr ss:[rsp+28],6
mov    dword ptr ss:[rsp+20],5
mov    r9d,4
mov    r8d,3
mov    edx,2
mov    ecx,1
call   <stack64.int __cdecl add(int, int, int, int, int, int)>

```

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	00000000100000001
000000000014FDD8	00000000140001179
000000000014FDE0	00000000100000000
000000000014FDE8	00000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	000000001400013C9



000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	00000000100000001
000000000014FDD8	00000000140001179
000000000014FDE0	00000000100000005
000000000014FDE8	00000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	000000001400013C9

RSP

Local var 1

Param 6

Param 5

```

sub rsp,48
mov dword ptr ss:[rsp+30],0
mov dword ptr ss:[rsp+28],6
mov dword ptr ss:[rsp+20],5
mov r9d,4
mov r8d,3
mov edx,2
mov ecx,1
call <stack64.int __cdecl add(int, int, int, int, int, int)>

```

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9



000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

RSP

Local var 1

Param 6

Param 5

Return ADder

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

```
mov dword ptr ss:[rsp+20],r9d
mov dword ptr ss:[rsp+18],r8d
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov dword ptr ss:[rsp],0
mov eax,dword ptr ss:[rsp+48]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000140001179
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

RSP

Local var 1

Param 6

Param 5

Param 4 (reg)

Return ADder

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

```
mov dword ptr ss:[rsp+20],r9d
mov dword ptr ss:[rsp+18],r8d
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov dword ptr ss:[rsp],0
mov eax,dword ptr ss:[rsp+48]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000001
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	00000001400013C9

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000003
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8BF8FF
000000000014FE08	00000001400013C9

RSP

Local var 1

Param 6

Param 5

Param 4 (reg)

Param 3 (reg)

Return ADder

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

```
mov dword ptr ss:[rsp+20],r9d
mov dword ptr ss:[rsp+18],r8d
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov dword ptr ss:[rsp],0
mov eax,dword ptr ss:[rsp+48]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000000
000000000014FDD0	0000000100000003
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000002
000000000014FDD0	0000000100000003
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

RSP

Local var 1

Param 6

Param 5

Param 4 (reg)

Param 3 (reg)

Param 2 (reg)

Return ADder

Onhexgroup.ir

ترسیم قاب پشته
در ۶۴ بیتی

```
mov dword ptr ss:[rsp+20],r9d
mov dword ptr ss:[rsp+18],r8d
mov dword ptr ss:[rsp+10],edx
mov dword ptr ss:[rsp+8],ecx
sub rsp,18
mov dword ptr ss:[rsp],0
mov eax,dword ptr ss:[rsp+48]
mov ecx,dword ptr ss:[rsp+20]
add ecx,eax
mov eax,ecx
mov dword ptr ss:[rsp],eax
mov eax,dword ptr ss:[rsp]
add rsp,18
ret
```

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000000
000000000014FDC8	0000000000000002
000000000014FDD0	0000000100000003
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

000000000014FDB8	0000000140001077
000000000014FDC0	0000000000000001
000000000014FDC8	0000000000000002
000000000014FDD0	0000000100000003
000000000014FDD8	0000000100000004
000000000014FDE0	0000000100000005
000000000014FDE8	0000000100000006
000000000014FDF0	000027AB00000000
000000000014FDF8	BC00040000000000
000000000014FE00	FFFA32031F8FBFF
000000000014FE08	00000001400013C9

RSP

Local var 1

Stack Param 6

Stack Param 5

Param 4 (reg)

Param 3 (reg)

Param 2 (reg)

Param 1 (reg)

Return ADder

Telegram: onhex_ir

درصد

ONHEXGROUP

97%

NOP 10%

PUSH 15%

CALL 8%

LEA 5%

MOV 27%

INT3 5%

ADD 3%

JNZ 2%

POP 3%

JMP 2%

XOR 2%

XADD 1%

CMP 3%

JG 1%

DEC 1%

JZ 2%

TEST 3%

RET 2%

SUB 2%

OTHRES
5%