.text:00000000 _sub:	push ebp
.text:00000001	mov ebp, esp
.text:0000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp ⊠
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp
.text:00000039	pop ebp

retn

.text:0000003A

eax	0xcafe ₩
есх	0xbabe ₩
edx	0xfeed ₩
ebp	0x0012FF50 ₩
esp	0x0012FF24 ₩

0x0012FF30	0x12FFB0 (char ** argv)策
0x0012FF2C	0x2 (int argc) 業
0x0012FF28	Addr after "call _main" 光
0x0012FF24	0x0012FF50(saved ebp)™
0x0012FF20	undef
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

Key: executed instruction ⊠, modified value \mathbb{M}, arbitrary example start value \mathbb{H}

.text:00000000 _sub:	push ebp
.text:00000001	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp ⊠
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp
.text:00000039	pop ebp
.text:0000003A	retn

eax	0xcafe
ecx	0xbabe
edx	0xfeed
ebp	0x0012FF24 M)
esp	0x0012FF24

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	undef
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

push

mov

retn

ebp

ebp, esp

.text:00000000 _sub:
.text:0000001
.text:00000003
.text:00000006
.text:00000009
.text:0000000C
.text:000000D
.text:00000010 _main:
.text:00000011
.text:00000013

Caller-save, or space for local var? This time it turns out to be space for local var since there is no corresponding pop, and the address is used later to refer to the value we know is stored in a.

```
eax, [ebp+8]
mov
       ecx, [ebp+0Ch]
mov
     eax, [ecx+eax*2]
lea
      ebp
pop
retn
      ebp
push
      ebp, esp
mov
       ecx 🗵
push
       eax, [ebp+0Ch]
mov
       ecx, [eax+4]
mov
bush
      ecx
     dword ptr ds: imp atoi
call
add
      esp, 4
      [ebp-4], eax
mov
       edx, [ebp-4]
mov
bush
       edx
       eax, [ebp+8]
mov
bush
       eax
     _sub
call
add
      esp, 8
      esp, ebp
mov
      ebp
pop
```

eax	0xcafe
ecx	0xbabe
edx	0xfeed
ebp	0x0012FF24
esp	0x0012FF20 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	Oxbabe (int a) M
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

retn

.text:00000000 _sub:
.text:0000001
.text:00000003
.text:00000006
.text:00000009
.text:0000000C
.text:000000D
.text:00000010 _main:
.text:00000011
.text:00000013
text:00000014

Getting the base of the argv char * array (aka argv[0])

.text:0000002F .text:00000034 .text:00000037 .text:00000039 .text:0000003A

push	ebp
mov	ebp, esp
mov	eax, [ebp+8]
mov	ecx, [ebp+0Ch]
lea	eax, [ecx+eax*2]
pop	ebp
retn	
push	ebp
	ebp, esp
push	
	eax, [ebp+0Ch] ⊠
mov	ecx, [eax+4]
push	
	dword ptr ds:impatoi
	esp, 4
	[ebp-4], eax
	edx, [ebp-4]
ı ·	edx
/	eax, [ebp+8]
•	eax
call	
	esp, 8
	esp, ebp
pop	ebp

eax	0x12FFB0 11)
ecx	0xbabe
edx	0xfeed
ebp	0x0012FF24
esp	0x0012FF20

0x0012FF30	0x12FFB0 (char ** argv)		
0x0012FF2C	0x2 (int argc)		
0x0012FF28	Addr after "call _main"		
0x0012FF24	0x0012FF50 (saved ebp)		
0x0012FF20	0xbabe (int a)		
0x0012FF1C	undef		
0x0012FF18	undef		
0x0012FF14	undef		
0x0012FF10	undef		
0x0012FF0C	undef		

.text:00000000 _sub:
.text:0000001
.text:0000003
.text:00000006
.text:00000009
.text:0000000C
.text:000000D
.text:00000010 _main:
.text:00000011
.text:00000013
.text:00000014
.text:00000017

Getting the char * at argv[1]
(I chose 0x12FFD4 arbitrarily since it's out of the stack scope we're currently looking at)

```
push
       ebp
       ebp, esp
mov
      eax, [ebp+8]
mov
      ecx, [ebp+0Ch]
mov
     eax, [ecx+eax*2]
lea
      ebp
pop
retn
push
      ebp
      ebp, esp
mov
push
      ecx
      eax, [ebp+0Ch]
mov
       ecx, [eax+4] ⊠
mov
push
      ecx
call
     dword ptr ds:__imp__atoi
add
      esp, 4
      [ebp-4], eax
mov
      edx, [ebp-4]
mov
      edx
push
       eax, [ebp+8]
mov
push
       eax
call
     sub
add
      esp, 8
      esp, ebp
mov
pop
      ebp
```

eax	0x12FFB0
ecx	0x12FFD4帧(arbitrary器)
edx	Oxfeed
ebp	0x0012FF24
esp	0x0012FF20

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0xbabe (int a)
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

.text:00000000 _sub: .text:00000001 .text:00000003 .text:00000006

Saving some slides... This will push the address of the string at argy[1] (0x12FFD4). atoi() will read the string and turn in into an int, put that int in eax, and return. Then the adding 4 to esp will negate the having pushed the input parameter and make 0x12FF1C undefined again (this is indicative of cdecl)

```
push
       ebp
      ebp, esp
mov
mov
      eax, [ebp+8]
      ecx, [ebp+0Ch]
mov
     eax, [ecx+eax*2]
lea
      ebp
pop
retn
      ebp
push
mov
      ebp, esp
push
      ecx
      eax, [ebp+0Ch]
mov
      ecx, [eax+4]
mov
       ecx 🗵
bush
tall
     dword ptr ds:__imp__atoi ⊠
       esp, 4 🗵
add
      [ebp-4], eax
mov
      edx, [ebp-4]
mov
      edx
push
       eax, [ebp+8]
mov
push
       eax
     sub
call
add
      esp, 8
      esp, ebp
mov
      ebp
pop
retn
```

eax	Ox100帧 (arbitrary器)
ecx	0x12FFD4
edx	0xfeed
ebp	0x0012FF24
esp	0x0012FF20

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0xbabe (int a)
0x0012FF1C	undef M)
0x0012FF18	undef M)
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

push

ebp

.text:00000000 _sub:
.text:00000001
.text:00000003
.text:00000006
.text:00000000
.text:0000000C
.text:0000000D
.text:00000010 _main:
.text:00000011
.text:00000013

First setting "a" equal to the return value. Then pushing "a" as the second parameter in sub(). We can see an obvious optimization would have been to replace the last two instructions with "push eax".

.text.uuuuuusA

mov ebp, esp mov eax, [ebp+8] ecx, [ebp+0Ch] mov eax, [ecx+eax*2] lea ebp pop retn push ebp mov ebp, esp push ecx eax, [ebp+0Ch] mov ecx, [eax+4]mov push ecx dword ptr ds:__imp__atoi call add esp, 4 [ebp-4], eax **⊠** mov √mov edx, [ebp-4] ⊠ push edx 🗵 eax, [ebp+8] mov push eax call sub add esp, 8 esp, ebp mov ebp pop retn

eax	0x100
ecx	0x12FFD4
edx	0x100 M)
ebp	0x0012FF24
esp	0x0012FF1C 110

0x0012FF30	0x12FFB0 (char ** argv)	
0x0012FF2C	0x2 (int argc)	
0x0012FF28	Addr after "call _main"	
0x0012FF24	0x0012FF50 (saved ebp)	
0x0012FF20	0x100 (int a) M)	
0x0012FF1C	0x100 (int y) M	
0x0012FF18	undef	
0x0012FF14	undef	
0x0012FF10	undef	
0x0012FF0C	undef	

.text:00000000 _sub:	push ebp
.text:0000001	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:0000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
	mov eax, [ebp+8] ⊠
Pushing argc	/push eax ⊠
as the first	call _sub
parameter (int	add esp, 8
paramotor (int	mov esp ebp

mov

pop retn

x) to sub()

.text:0000003A

esp, ebp

ebp

eax	0x2 M)
ecx	0x12FFD4
edx	0x100
ebp	0x0012FF24
esp	0x0012FF18 M)

0x0012FF30	0x12FFB0 (char ** argv)	
0x0012FF2C	0x2 (int argc)	
0x0012FF28	Addr after "call _main"	
0x0012FF24	0x0012FF50 (saved ebp)	
0x0012FF20	0x100 (int a)	
0x0012FF1C	0x100 (int y)	
0x0012FF18	0x2 (int x) M	
0x0012FF14	undef	
0x0012FF10	undef	
0x0012FF0C	undef	
!		

toxt:00000000 oub.	nuah	ahn
.text:00000000 _sub:	push	•
.text:0000001	mov	I.) I.
.text:00000003	mov	eax, [ebp+8]
.text:00000006	mov	ecx, [ebp+0Ch]
.text:00000009	lea	eax, [ecx+eax*2]
.text:000000C	pop	ebp
.text:000000D	retn	
.text:00000010 _main:	push	ebp
.text:00000011	mov	ebp, esp
.text:00000013	push	ecx
.text:00000014	mov	eax, [ebp+0Ch]
.text:00000017	mov	ecx, [eax+4]
.text:0000001A	push	ecx
.text:0000001B	call	dword ptr ds:impatoi
.text:00000021	add	esp, 4
.text:00000024	mov	[ebp-4], eax
.text:00000027	mov	edx, [ebp-4]
.text:0000002A	push	edx
.text:0000002B	mov	eax, [ebp+8]
.text:0000002E	push	eax
.text:0000002F	call	_sub ⊠
.text:00000034	add	esp, 8
.text:00000037	mov	esp, ebp
.text:00000039	pop	ebp
.text:0000003A	retn	

eax	0x2
ecx	0x12FFD4
edx	0x100
ebp	0x0012FF24
esp	0x0012FF14 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	0x00000034 M)
0x0012FF10	undef
0x0012FF0C	undef

.text:00000000 _sub:

push ebp ⊠

	•
.text:0000001	mov ebp, esp ⊠
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:0000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp
.text:00000039	pop ebp
.text:0000003A	retn

eax	0x2
ecx	0x12FFD4
edx	0x100
ebp	0x0012FF10 M)
esp	0x0012FF10 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	0x00000034
0x0012FF10	0x0012FF24(saved ebp)™
0x0012FF0C	undef

```
.text:00000000 sub:
                        push
                               ebp
.text:00000001
                               ebp, esp
                        mov
                               eax, [ebp+8] ⊠
                        mov
                        mov
                               ecx, [ebp+0Ch] ⊠
  Move "x" into eax,
                             eax, [ecx+eax*2]
                        lea
   and "y" into ecx.
                              ebp
                        pop
                        retn
.text:00000010 main:
                       push
                              ebp
.text:00000011
                       mov
                              ebp, esp
.text:00000013
                        push
                              ecx
                               eax, [ebp+0Ch]
.text:00000014
                       mov
.text:00000017
                               ecx, [eax+4]
                       mov
.text:0000001A
                        push
                              ecx
.text:0000001B
                        call
                             dword ptr ds: imp atoi
.text:00000021
                        add
                              esp, 4
.text:00000024
                              [ebp-4], eax
                       mov
.text:00000027
                               edx, [ebp-4]
                       mov
.text:0000002A
                              edx
                        push
.text:0000002B
                               eax, [ebp+8]
                        mov
.text:0000002E
                        push
                               eax
.text:0000002F
                        call
                             _sub
.text:0000034
                        add
                              esp, 8
.text:00000037
                              esp, ebp
                       mov
.text:00000039
                              ebp
                        pop
.text:0000003A
                        retn
```

eax	0x2 1 (no value change)
есх	0x100 m
edx	0x100
ebp	0x0012FF10
esp	0x0012FF10

i	-
0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	0x00000034
0x0012FF10	0x0012FF24 (saved ebp)
0x0012FF0C	undef

retn

.text:00000000 _sub: .text:00000001 .text:00000003

Set the return value
(eax) to 2*x + y.
Note: neither
pointer arith, nor an
"address" which
was loaded. Just an
afficient way to do a
calculation.

.text:0000001B .text:00000021 .text:00000024 .text:00000027 .text:0000002A .text:0000002B .text:0000002E .text:0000002F .text:00000034 .text:00000037

.text:00000039

.text:0000003A

push ebp ebp, esp mov mov eax, [ebp+8] ecx, [ebp+0Ch] mov eax, [ecx+eax*2] 🗵 ⊿lea ebp pop retn push ebp mov ebp, esp push ecx eax, [ebp+0Ch] mov ecx, [eax+4]mov push ecx dword ptr ds:__imp__atoi call add esp, 4 [ebp-4], eax mov edx, [ebp-4] mov push edx eax, [ebp+8] mov push eax call sub add esp, 8 esp, ebp mov ebp pop

eax	0x104 M)
ecx	0x100
edx	0x100
ebp	0x0012FF10
esp	0x0012FF10

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	0x00000034
0x0012FF10	0x0012FF24 (saved ebp)
0x0012FF0C	undef

.text:00000000 _sub:	push	ebp
.text:00000001	mov	ebp, esp
.text:00000003	mov	eax, [ebp+8]
.text:00000006	mov	ecx, [ebp+0Ch]
.text:00000009	lea	eax, [ecx+eax*2]
.text:000000C	pop	ebp ⊠
.text:000000D	retn	
.text:00000010 _main:	push	ebp
.text:00000011	mov	ebp, esp
.text:00000013	push	ecx
.text:00000014	mov	eax, [ebp+0Ch]
.text:00000017	mov	ecx, [eax+4]
.text:0000001A	push	ecx
.text:0000001B	call	dword ptr ds:impatoi
.text:00000021	add	esp, 4
.text:00000024	mov	[ebp-4], eax
.text:00000027	mov	edx, [ebp-4]
.text:0000002A	push	edx
.text:0000002B	mov	eax, [ebp+8]
.text:0000002E	push	eax
.text:0000002F	call	_sub
.text:00000034	add	esp, 8
.text:00000037	mov	esp, ebp
.text:00000039	pop	ebp
.text:0000003A	retn	

eax	0x104
ecx	0x100
edx	0x100
ebp	0x0012FF24 M)
esp	0x0012FF14 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	0x00000034
0x0012FF10	undef M)
0x0012FF0C	undef

.text:00000000 _sub:	push ebp
.text:00000001	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn ⊠
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp
.text:00000039	pop ebp
.text:0000003A	retn

eax	0x104
ecx	0x100
edx	0x100
ebp	0x0012FF24
esp	0x0012FF18 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	0x100 (int y)
0x0012FF18	0x2 (int x)
0x0012FF14	undef M)
0x0012FF10	undef
0x0012FF0C	undef

.text:00000000 _sub:	push ebp
.text:00000000 _sdb.	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8 ⊠
.text:00000037	mov esp, ebp
.text:00000039	pop ebp
.text:0000003A	retn

eax	0x104
ecx	0x100
edx	0x100
ebp	0x0012FF24
esp	0x0012FF20 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	0x100 (int a)
0x0012FF1C	undef M)
0x0012FF18	undef My
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

.text:00000000 _sub:	push ebp
.text:0000001	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp ⊠
.text:00000039	pop ebp

retn

.text:000003A

eax	0x104
ecx	0x100
edx	0x100
ebp	0x0012FF24
esp	0x0012FF24 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	0x0012FF50 (saved ebp)
0x0012FF20	undef M)
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef

.text:00000000 _sub:	push ebp
.text:00000001	mov ebp, esp
.text:00000003	mov eax, [ebp+8]
.text:00000006	mov ecx, [ebp+0Ch]
.text:00000009	lea eax, [ecx+eax*2]
.text:000000C	pop ebp
.text:000000D	retn
.text:00000010 _main:	push ebp
.text:00000011	mov ebp, esp
.text:00000013	push ecx
.text:00000014	mov eax, [ebp+0Ch]
.text:00000017	mov ecx, [eax+4]
.text:0000001A	push ecx
.text:0000001B	call dword ptr ds:impatoi
.text:00000021	add esp, 4
.text:00000024	mov [ebp-4], eax
.text:00000027	mov edx, [ebp-4]
.text:0000002A	push edx
.text:0000002B	mov eax, [ebp+8]
.text:0000002E	push eax
.text:0000002F	call _sub
.text:00000034	add esp, 8
.text:00000037	mov esp, ebp
.text:00000039	pop ebp ⊠
.text:0000003A	retn

eax	0x104
ecx	0x100
edx	0x100
ebp	0x0012FF50 M)
esp	0x0012FF28 M)

0x0012FF30	0x12FFB0 (char ** argv)
0x0012FF2C	0x2 (int argc)
0x0012FF28	Addr after "call _main"
0x0012FF24	undef M)
0x0012FF20	undef
0x0012FF1C	undef
0x0012FF18	undef
0x0012FF14	undef
0x0012FF10	undef
0x0012FF0C	undef