



Reverzne inzinierstvo




Bezpecnost informacnych systemov z pohladu praxe

Peter Svec

>shellcode vysledky

Rank		Team	Score
#1		Exploit Sigmas	11
#2		mSUS	11
#3		impostor	11
#4		blbi_a_blbsy	11
#5		team	11

celkovy stav ->

			
1. Exploit Sigmas	1	0	0
2. mSUS	0	1	0
3. impostor	0	0	1

>motivacia

- >schopnost pochopit program aj bez zdrojoveho kodu
- >vyvoj exploitov
- >analyza malveru
- >cracking, patching



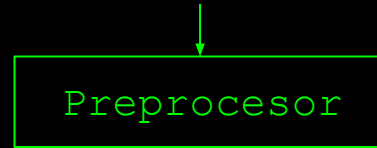
>standardny proces

```
#include <stdio.h>
int main()
{
    printf("");
    return 0;
}
```



```
00 0f ff 48 22 01 55
42 12 69 12 00 48 12
13 22 f5 a5 ...
```

zdrojovy kod



spustitelny subor

← zdrojovy kod (makra,
komentare,
hlavickove subory)

← zdrojovy kod (jazyk
sym. instrukcii)

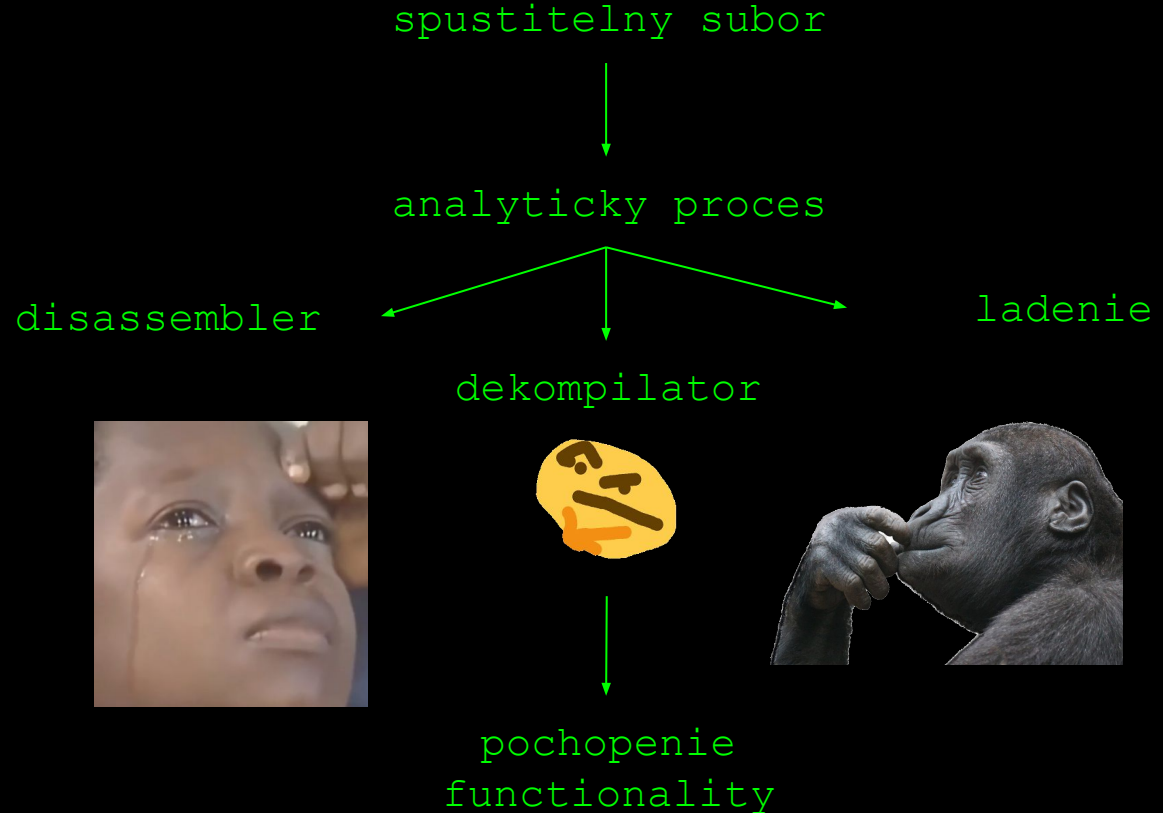
← objektovy kod

>zostavovací process

- >Pocas zostavovania programu stracame mnozstvo informacii:
 - >nazvy premennych
 - >nazvy funkcii, tried,...
 - >komentare
 - >struktury
 - >optimalizacie prekladaca (inlining, loop unrolling,...)

>reverzne inzinierstvo

>opacny proces



>zasobnik

>LIFO datova struktura pouzivana pri volani funkcii (call stack)

>Na zasobnik sa ukladaju:

>lokalne premenne

>navratova adresa



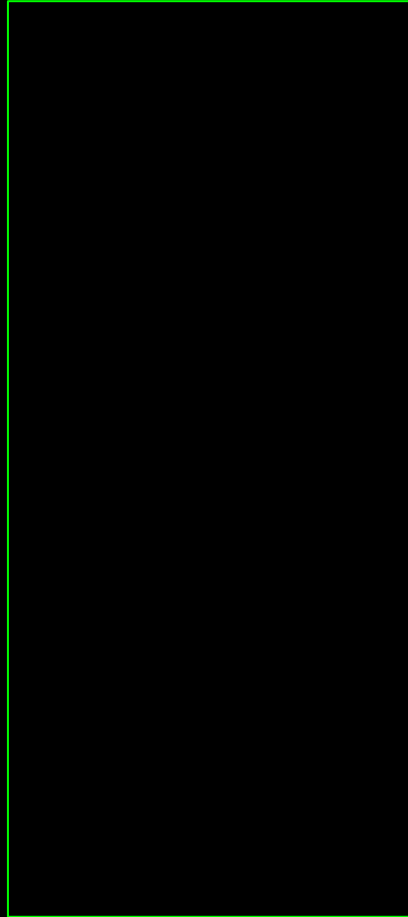
>zasobnikovy ramec z predchadzajuceho volania

>pri vysokom mnozstve argumentov aj argumenty

**ZASOBNIK RASTIE OPACNYM SMEROM
OD VYSOKYCH ADRIES (0xFFFF...) PO NIZSIE (0x000...)**

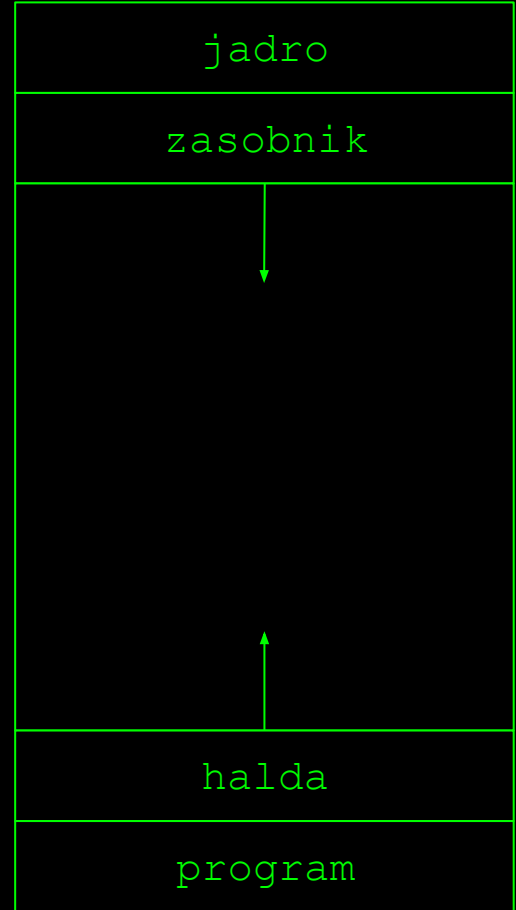
>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```



0xffff

0x0000



0x08

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

call foo

0xffff

RBP →

RSP →

jadro

main

halda

0x0000

program

0x09

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

call foo

0xffff

RBP →

RSP →

0x0000

jadro

main

navratova addr

halda

program

0x0a

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
push rbp
```

0xffff

RBP →

RSP →

0x0000

jadro

main

navratova addr

halda

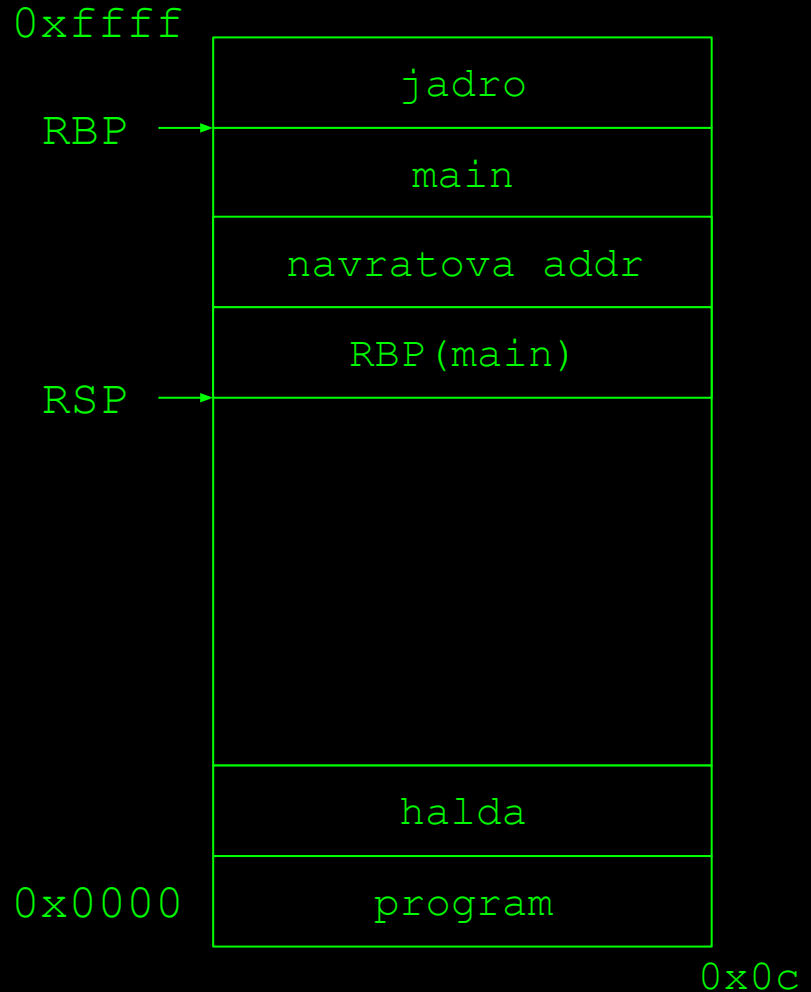
program

0x0b

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

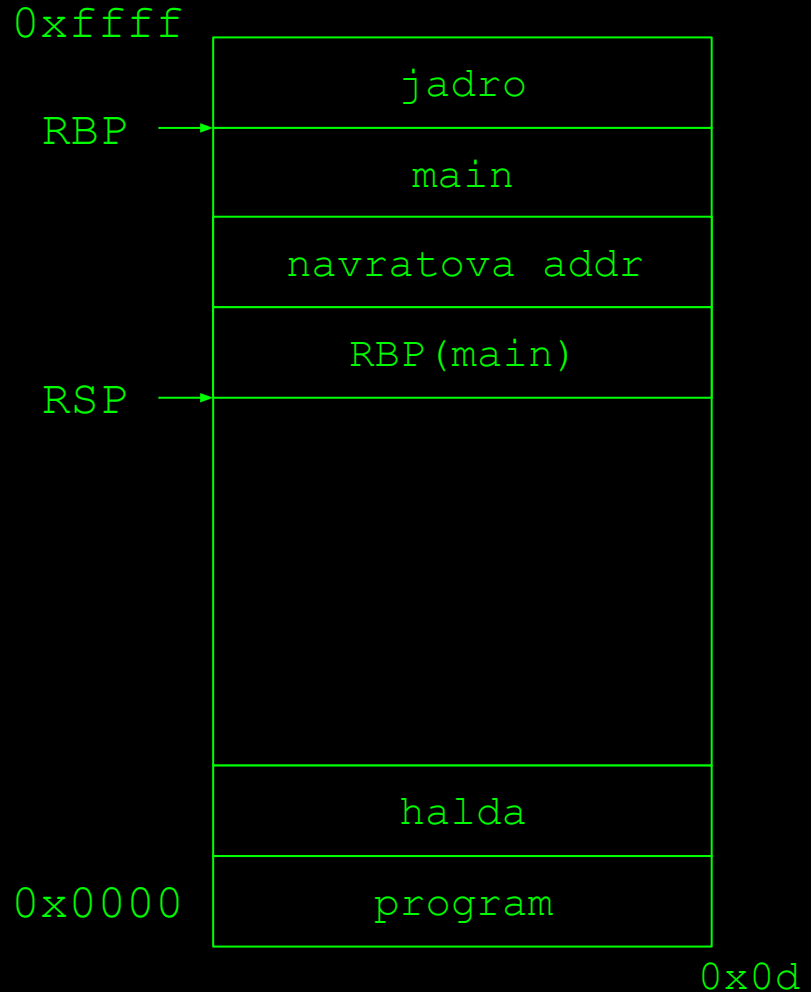
```
call foo  
push rbp
```



>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp
```



>zasobnik

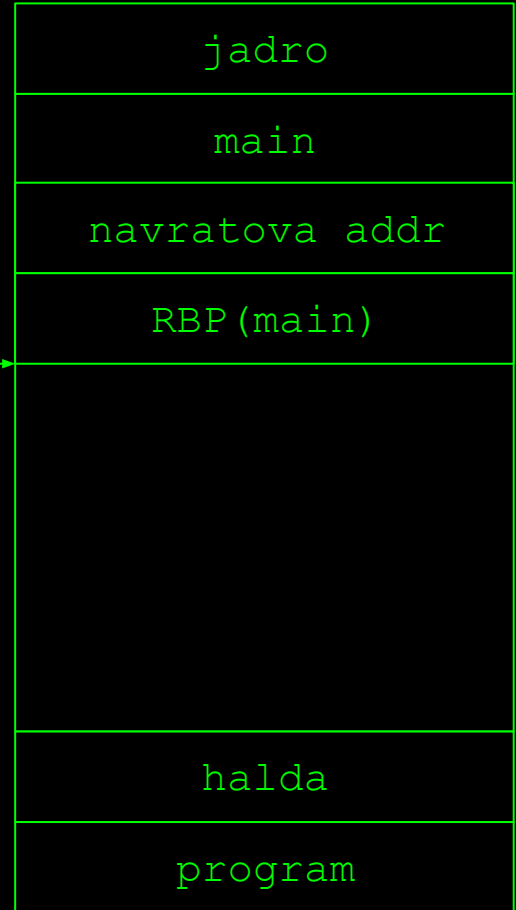
```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp
```

0xffff

RBP
RSP →

0x0000



0x0e

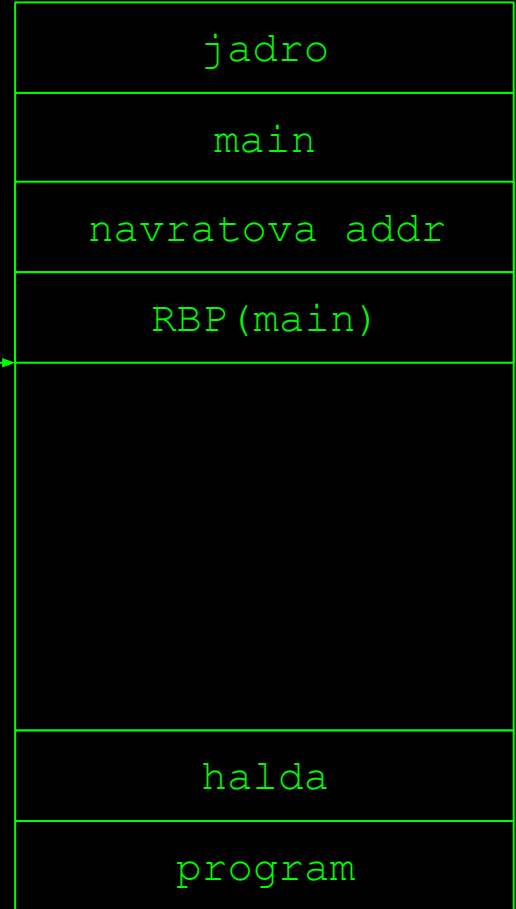
>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10
```

0xffff

RBP
RSP →



0x0000

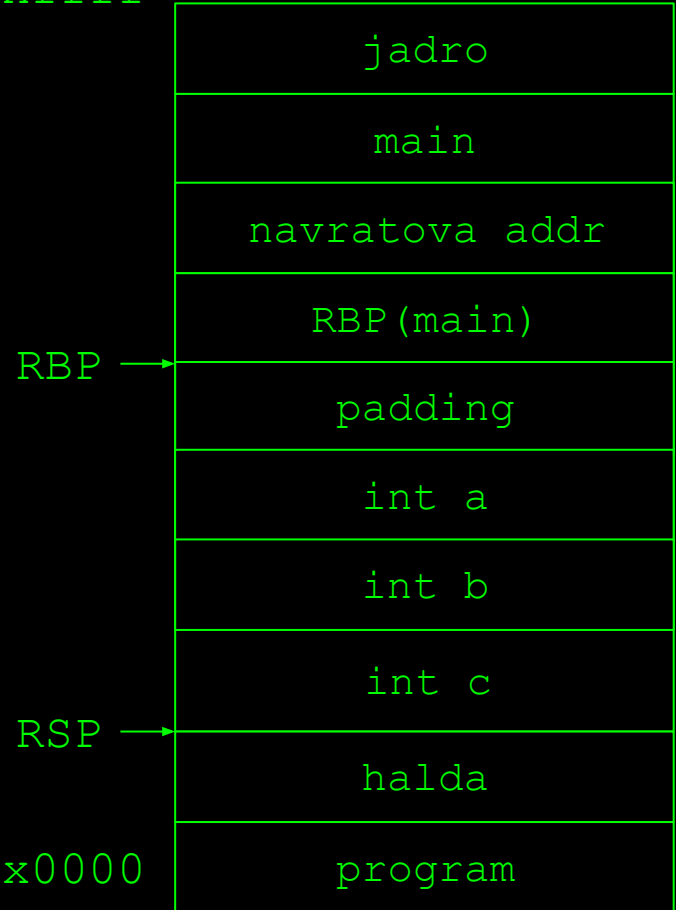
0x0f

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10
```

0xffff



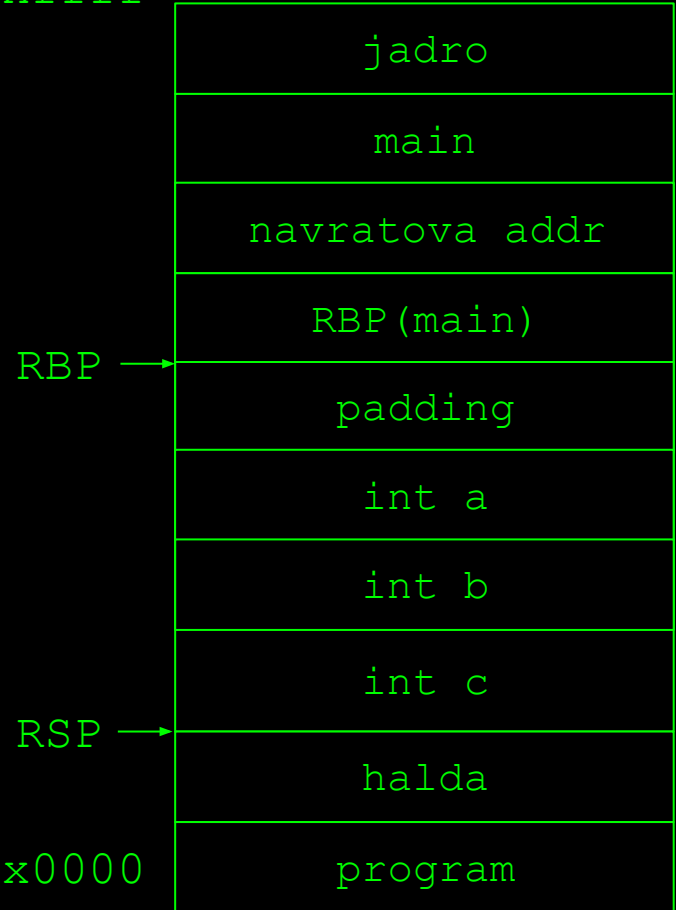
0x10

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2
```

0xffff



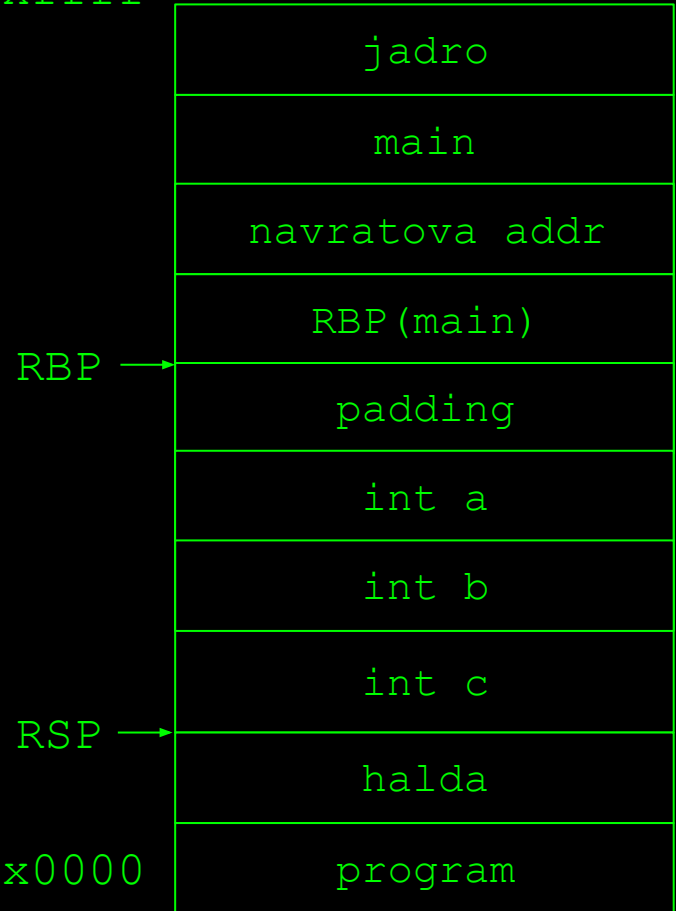
0x11

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...
```

0xffff



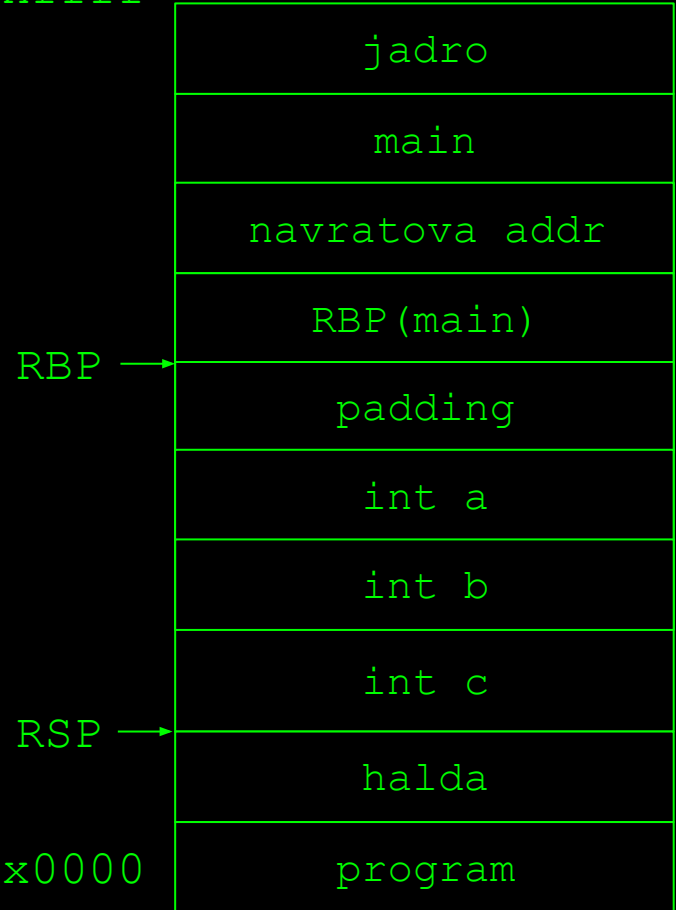
0x12

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...  
  
mov rax,[rbp-4]
```

0xffff



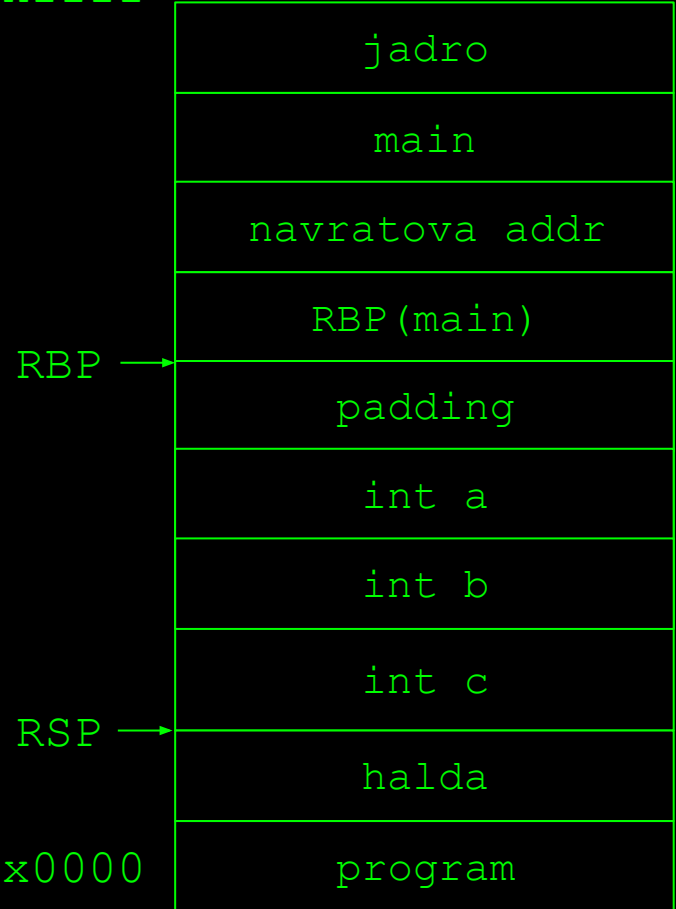
0x13

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...  
  
mov rax,[rbp-4]  
mov rsp, rbp
```

0xffff



0x14

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

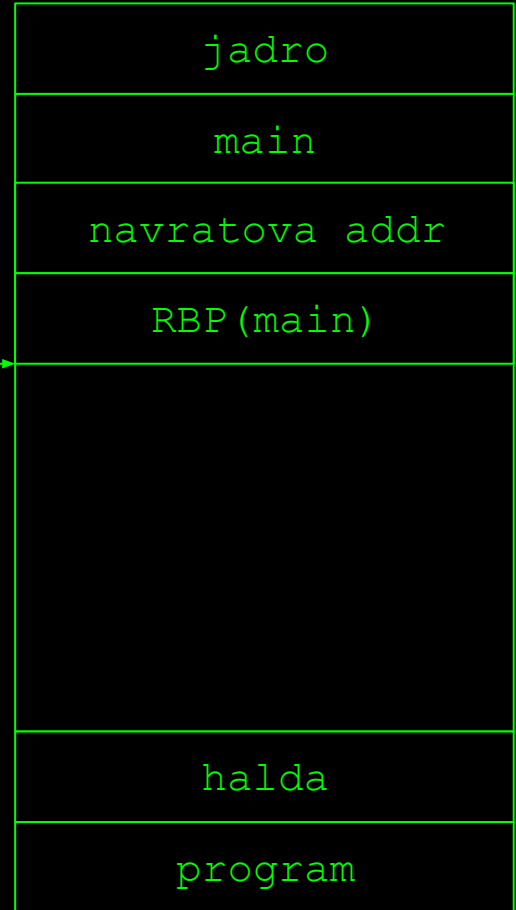
```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...  
  
mov rax,[rbp-4]  
mov rsp, rbp
```

0xffff

RBP

RSP →

0x0000



>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

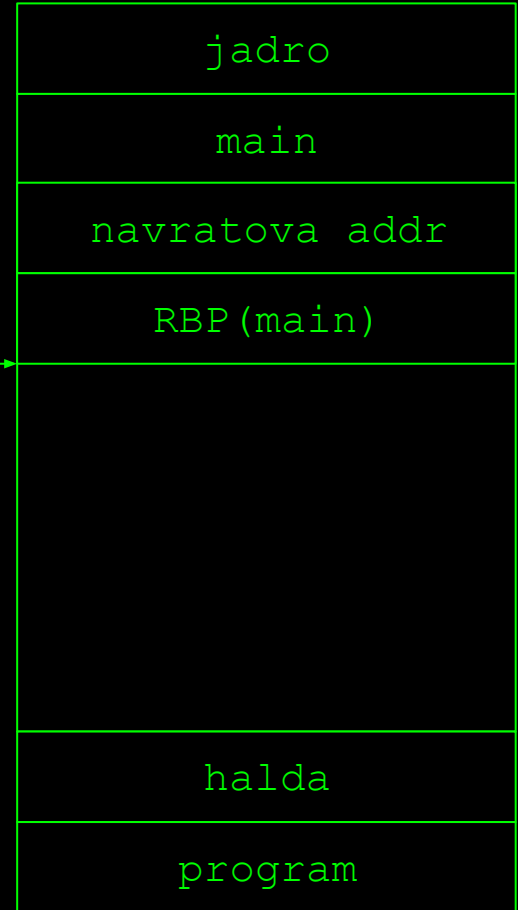
```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...  
  
mov rax,[rbp-4]  
mov rsp, rbp  
pop rbp
```

0xffff

RBP

RSP →

0x0000

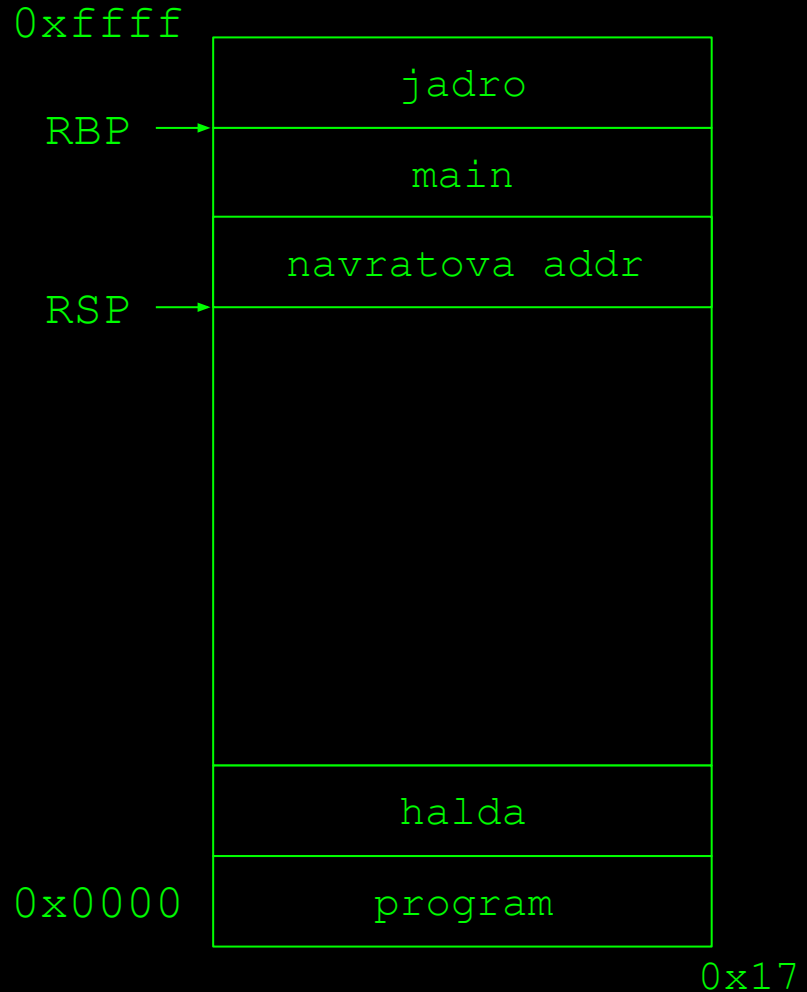


0x16

>zasobnik

```
int foo()  
{  
    int a,b,c;  
    b=1;c=2;  
    a=b+c;  
    return a;  
}  
  
int main()  
{  
    foo();  
    return 0;  
}
```

```
call foo  
  
push rbp  
mov rbp, rsp  
sub rsp,0x10  
mov[rbp-8],1  
mov[rbp-12],2  
  
vypocty...  
  
mov rax,[rbp-4]  
mov rsp, rbp  
pop rbp
```



>zasobnik

```
int foo()
{
    int a,b,c;
    b=1;c=2;
    a=b+c;
    return a;
}

int main()
{
    foo();
    return 0;
}
```

```
call foo

push rbp
mov rbp, rsp
sub rsp,0x10
mov[rbp-8],1
mov[rbp-12],2

vypocty...

mov rax,[rbp-4]
mov rsp, rbp
pop rbp
ret
```



ret = nacita do RIP vrch zasobnika

0x18

>zasobnik

```
int foo()
{
    int a,b,c;
    b=1;c=2;
    a=b+c;
    return a;
}

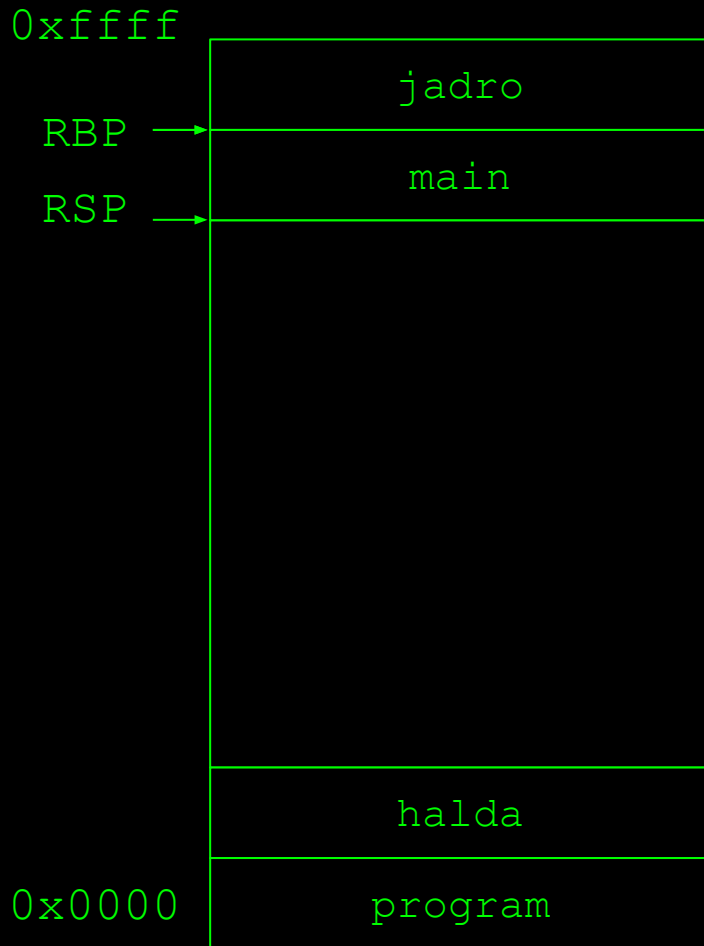
int main()
{
    foo();
    return 0;
}
```

```
call foo

push rbp
mov rbp, rsp
sub rsp,0x10
mov[rbp-8],1
mov[rbp-12],2

vypocty...

mov rax,[rbp-4]
mov rsp, rbp
pop rbp
ret
```



ret = nacita do RIP vrch zasobnika

0x19

>nastroje

>staticka analyza (bez spustenia)

>Binary Ninja¹ (cloud verzia)

>IDA Free 7.6

>Ghidra

>Radare2

>dynamicka analyza (so spustenim)

>strace

>GDB



¹<https://cloud.binary.ninja/>

>ulohy

- >cielom uloh bude zreverzovat binarky a najst licencny kluc
 - >najst komunikacny kanal
 - >zreverzovat trasnformacie a format
- >analyticke ulohy
 - >ziadne programovanie, t.j. odovzdat **kratku** dokumentaciu

```
scp -i ./key -O hacker@feictf.xyz:/challenge/level1.0 .
```

>vela stastia

