# Reverzne inzinierstvo

Bezpecnost informacnych systemov z pohladu praxe

Peter Svec

0x01

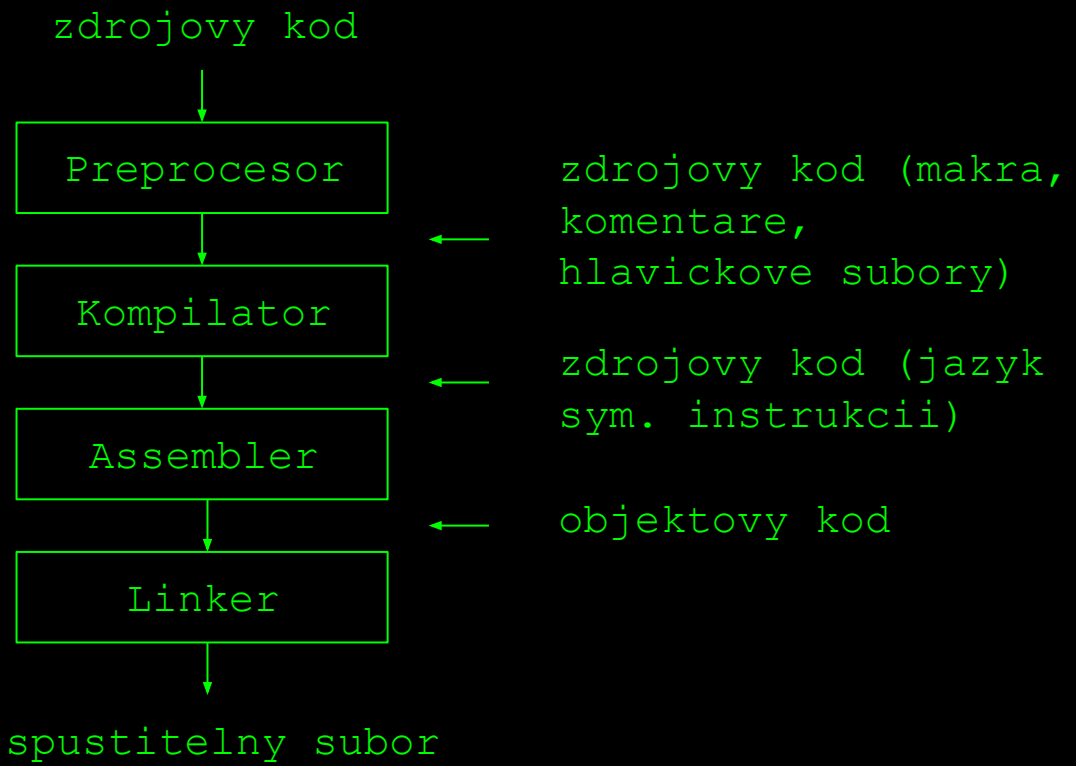
# >motivacia

>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 …
```



# >zostavovaci process

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

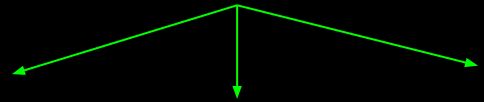
# >reverzne inzinierstvo

>opacny proces

spustitelny subor



analyticky proces



disassembler

dekompilator

ladenie



pochopenie
functionality

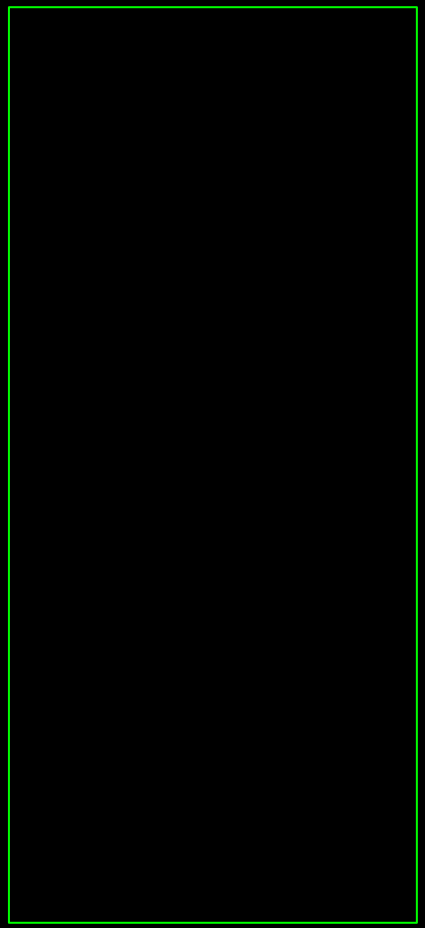
# >zasobnik

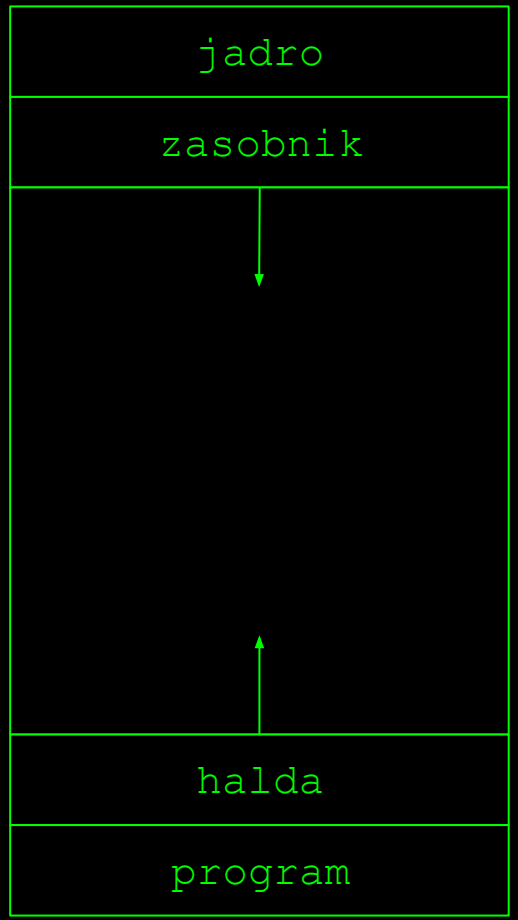
>LIFO datova struktura pouzivana pri volani funkcii (call stack)
>Na zasobnik sa ukladaju:
    >lokalne premenne
    >navratova adresa 
    >zasobnikovy ramec z predchadzajuceho volania
    >pri vysokom mnozstve argumentov aj argumenty

**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

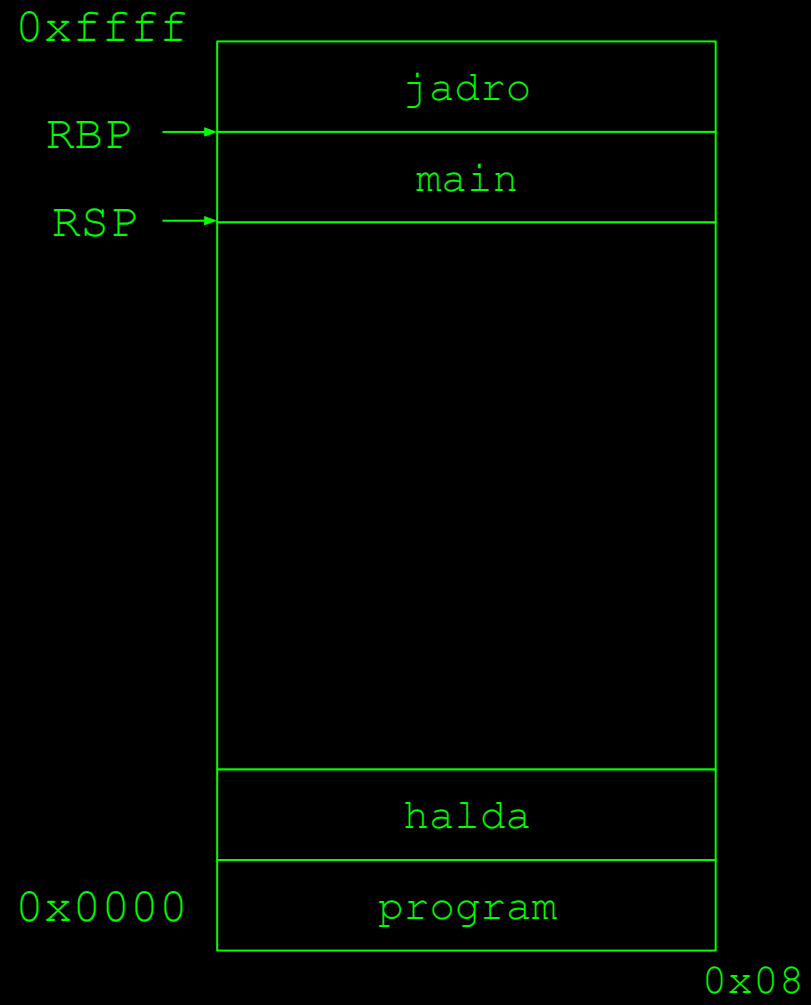
0x0000

0x07

# >zasobnik

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

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

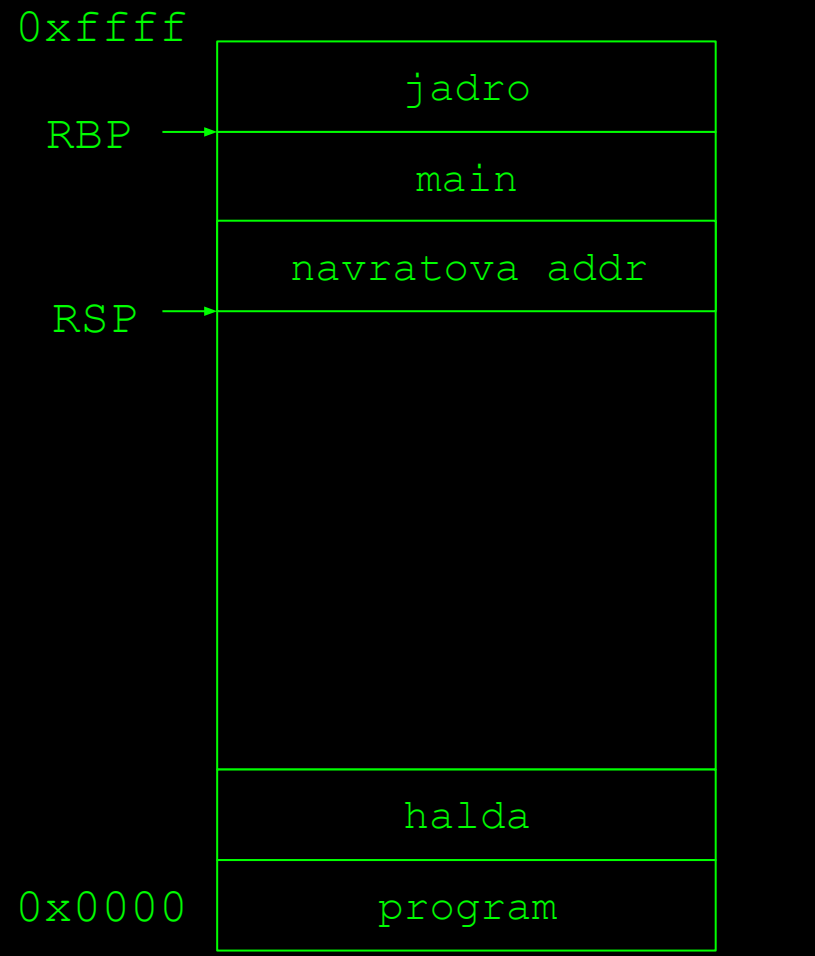
```
call foo
```



# >zasobnik

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

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

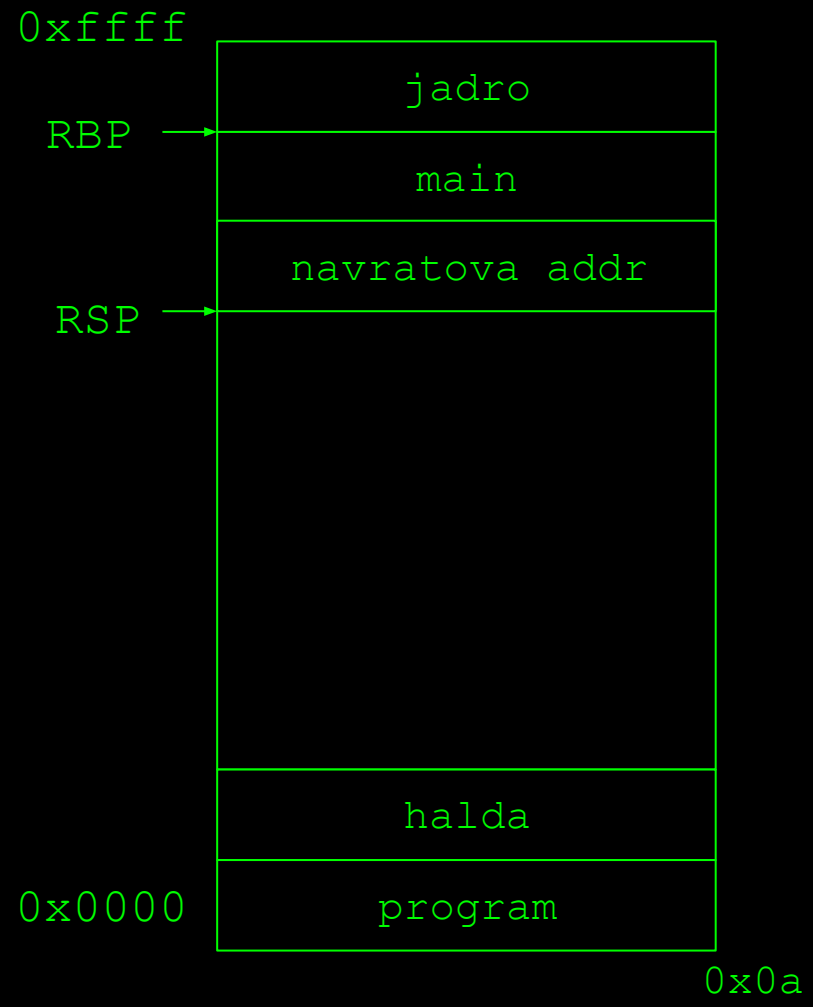
```
call foo
```



# >zasobnik

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

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

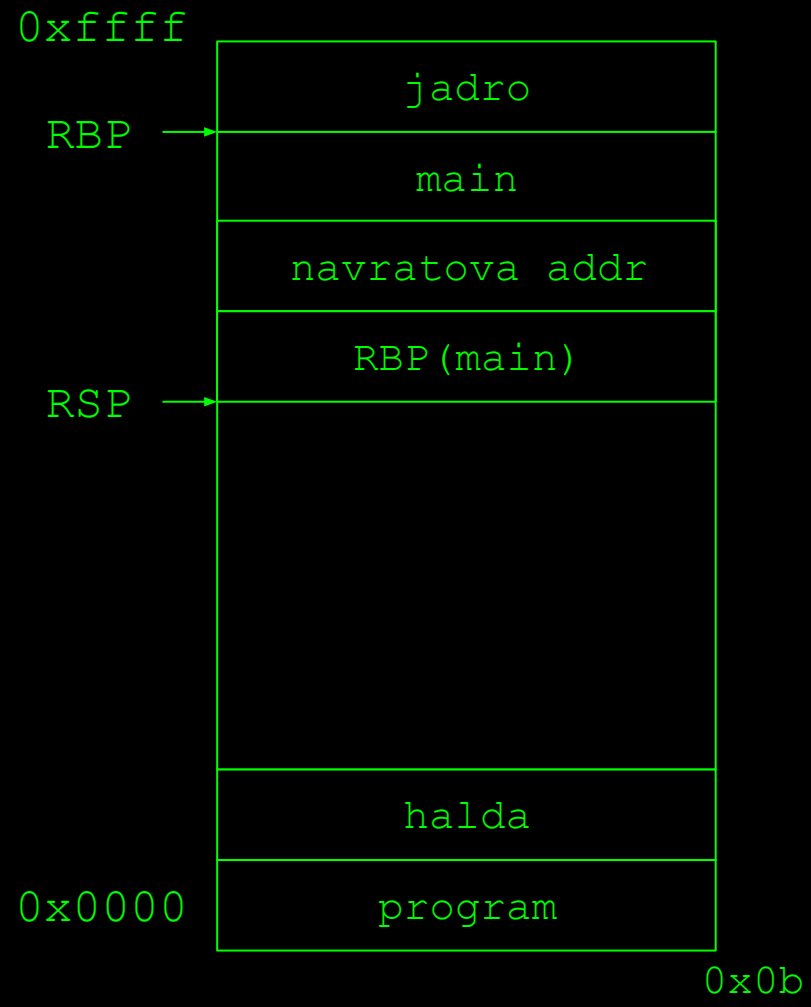
```
call foo

push rbp
```



# >zasobnik

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

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

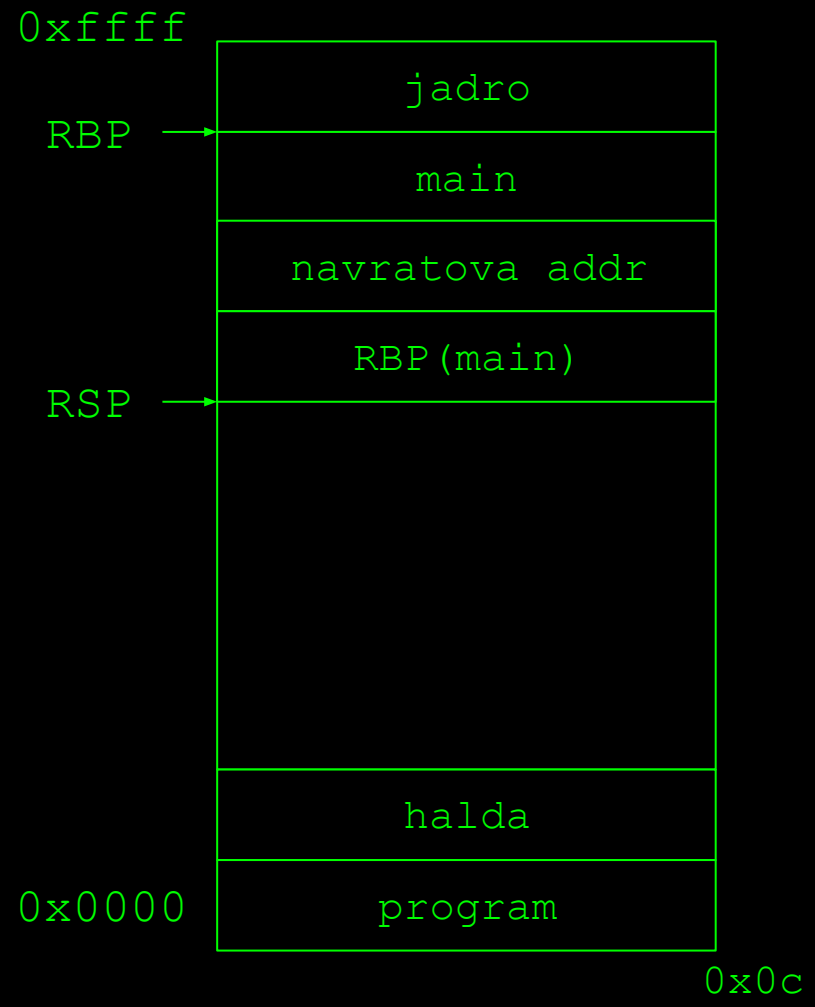
```
call foo

push rbp
```



# >zasobnik

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

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

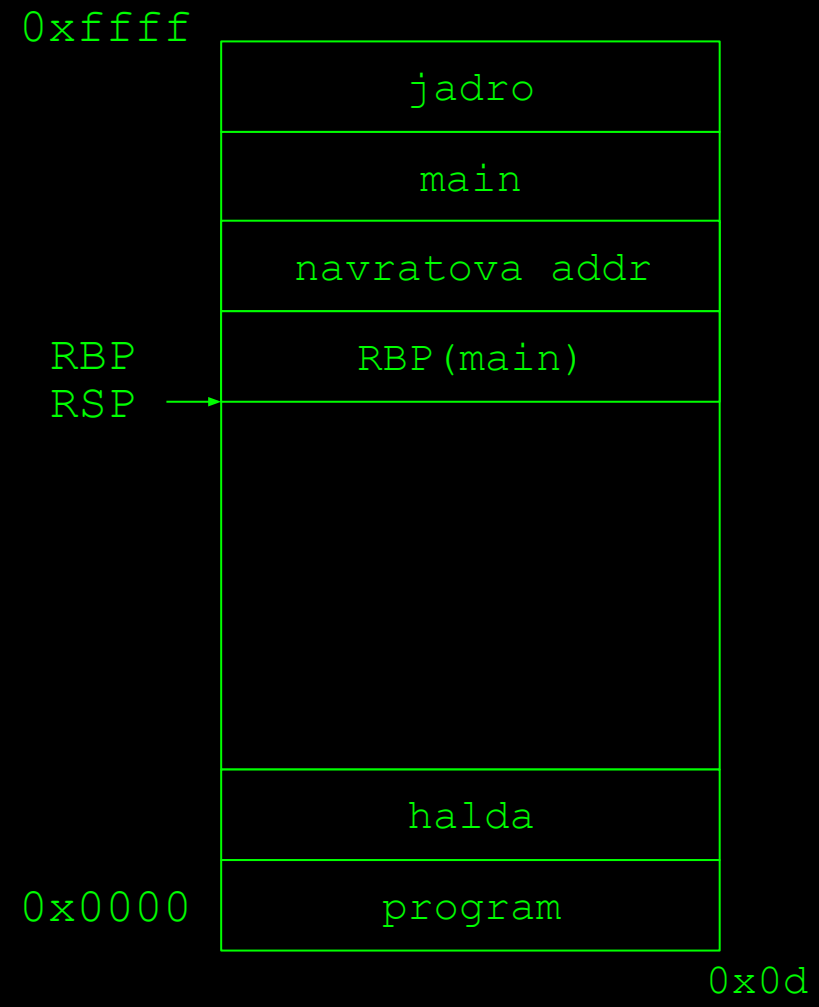
```
call foo

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;
}
```

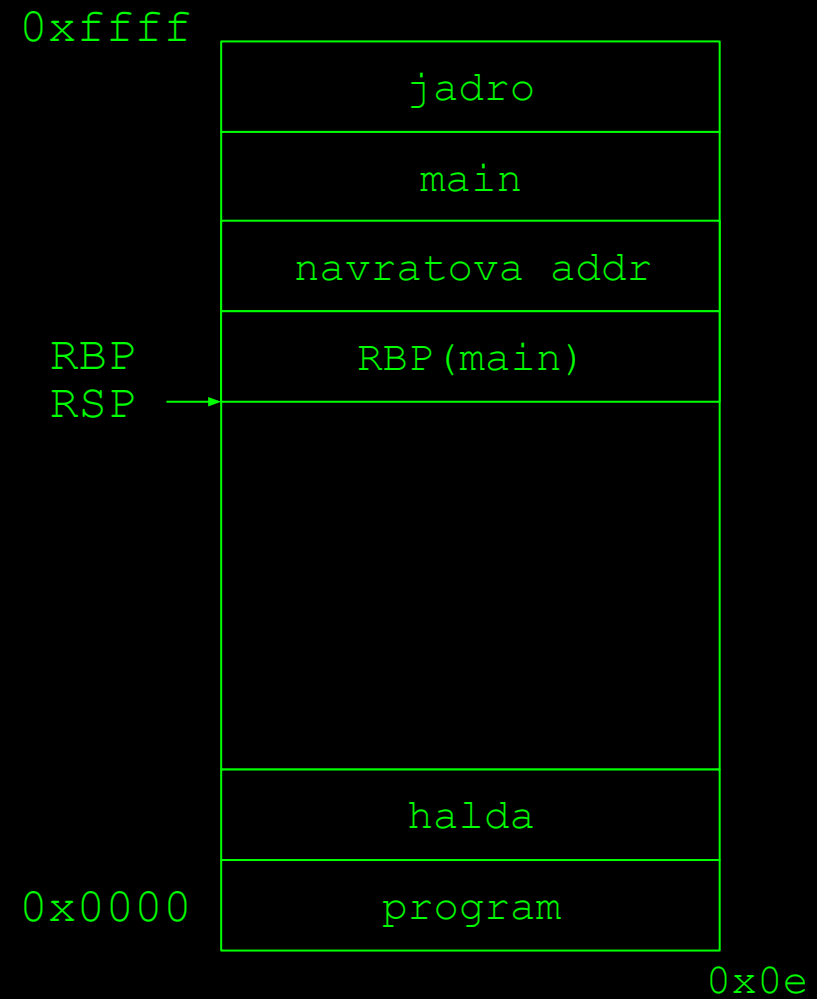
```
call foo

push rbp
mov rbp, rsp
```



# >zasobnik

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

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

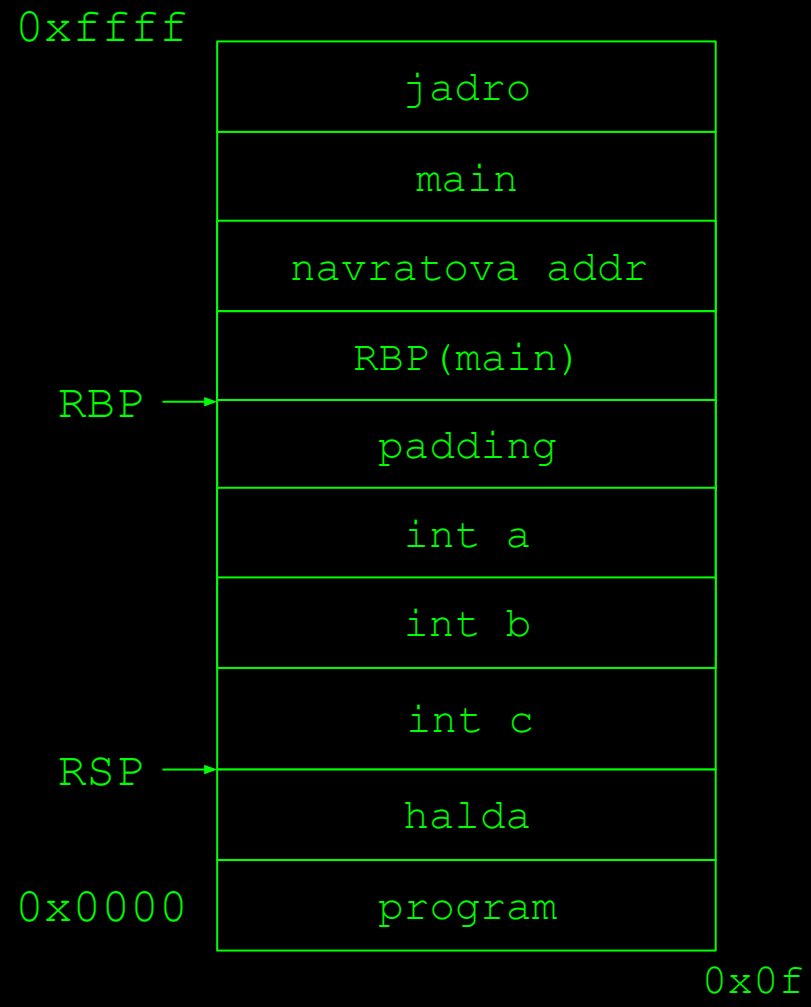
```
call foo

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



# >zasobnik

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

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

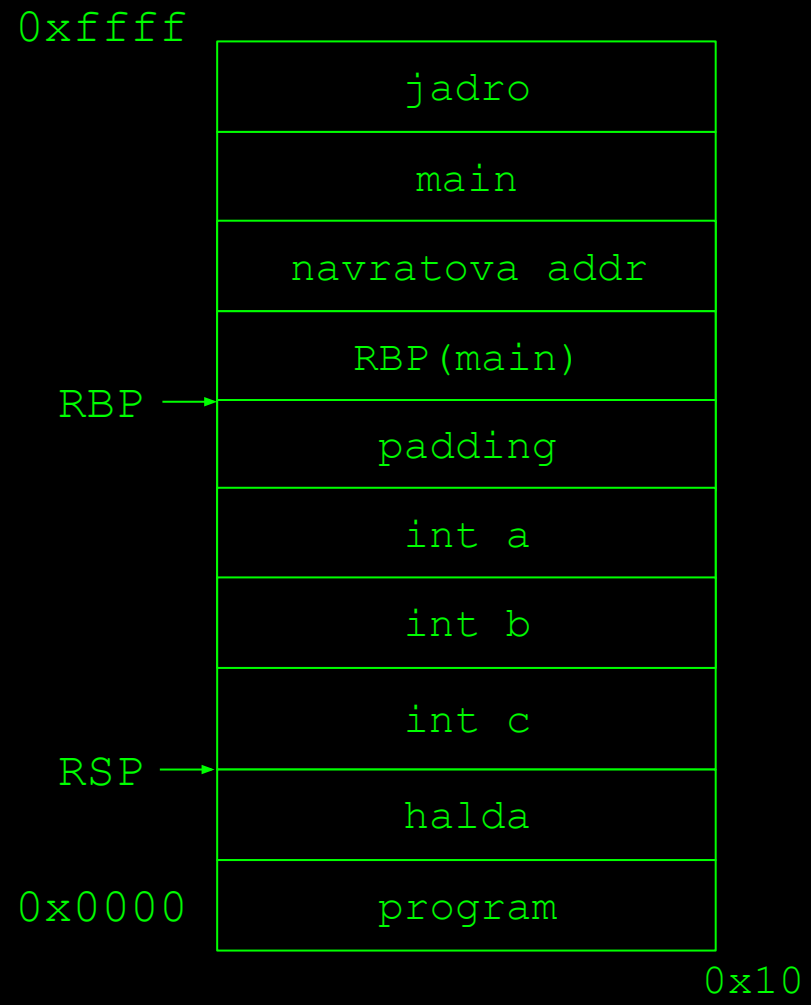
```
call foo

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



# >zasobnik

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

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

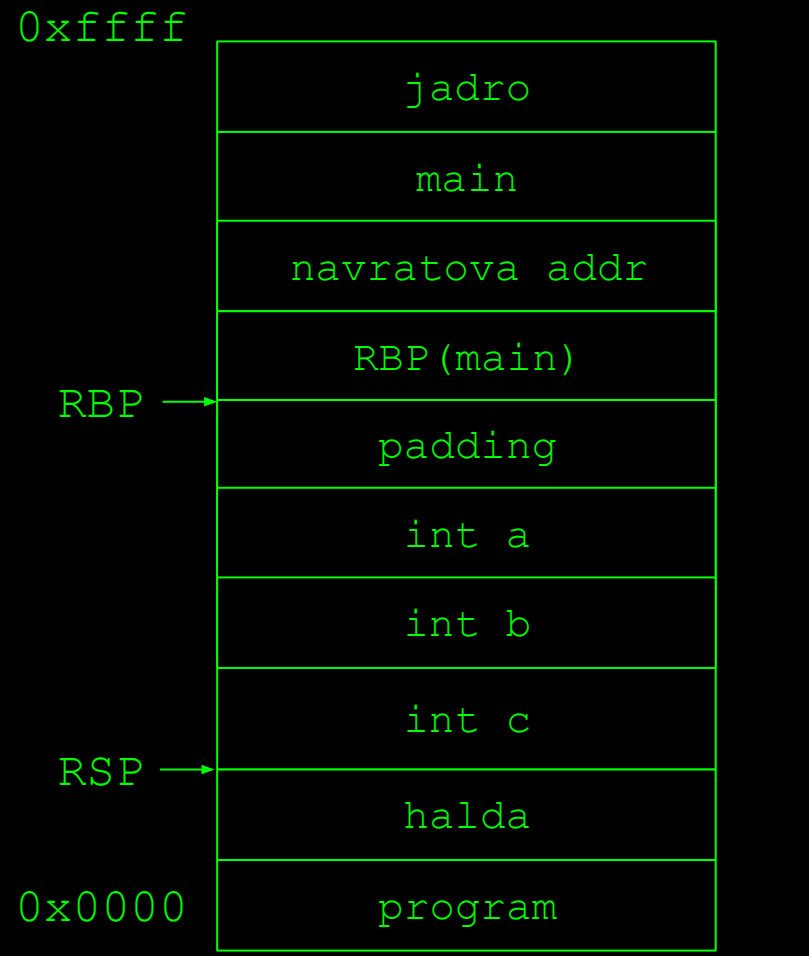
```
call foo

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



# >zasobnik

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

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

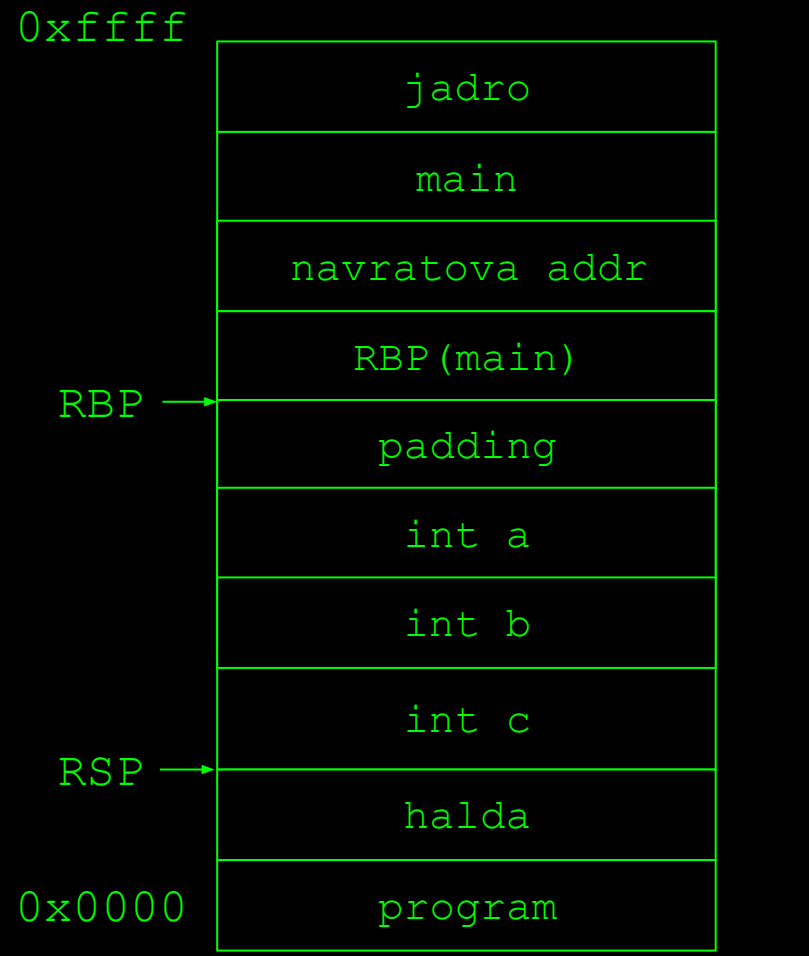
```
call foo

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

vypocty…
```



# >zasobnik

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

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

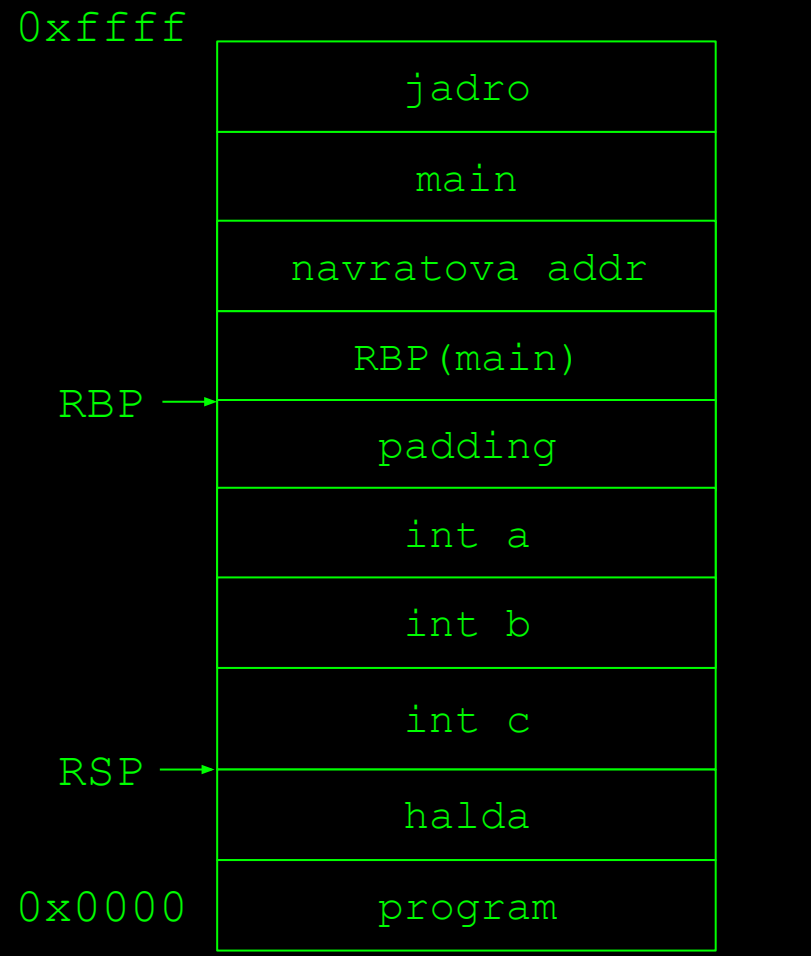
```
call foo

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

vypocty…

mov rax,[rbp-4]
```



# >zasobnik

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

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

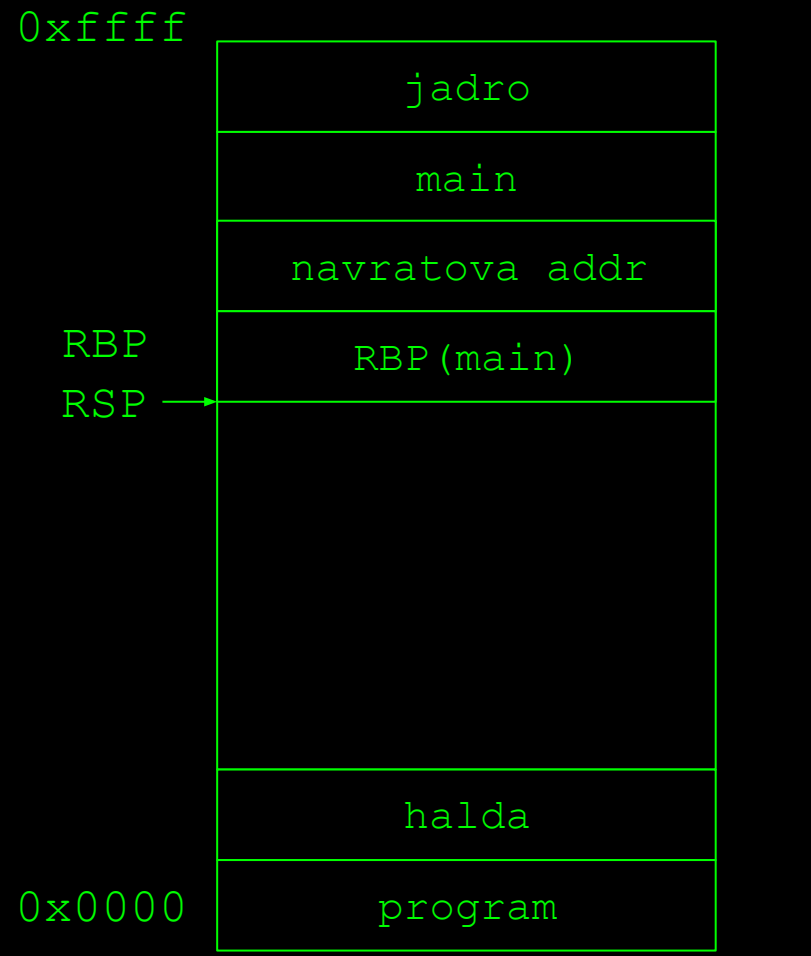
```
call foo

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

vypocty…

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



# >zasobnik

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

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

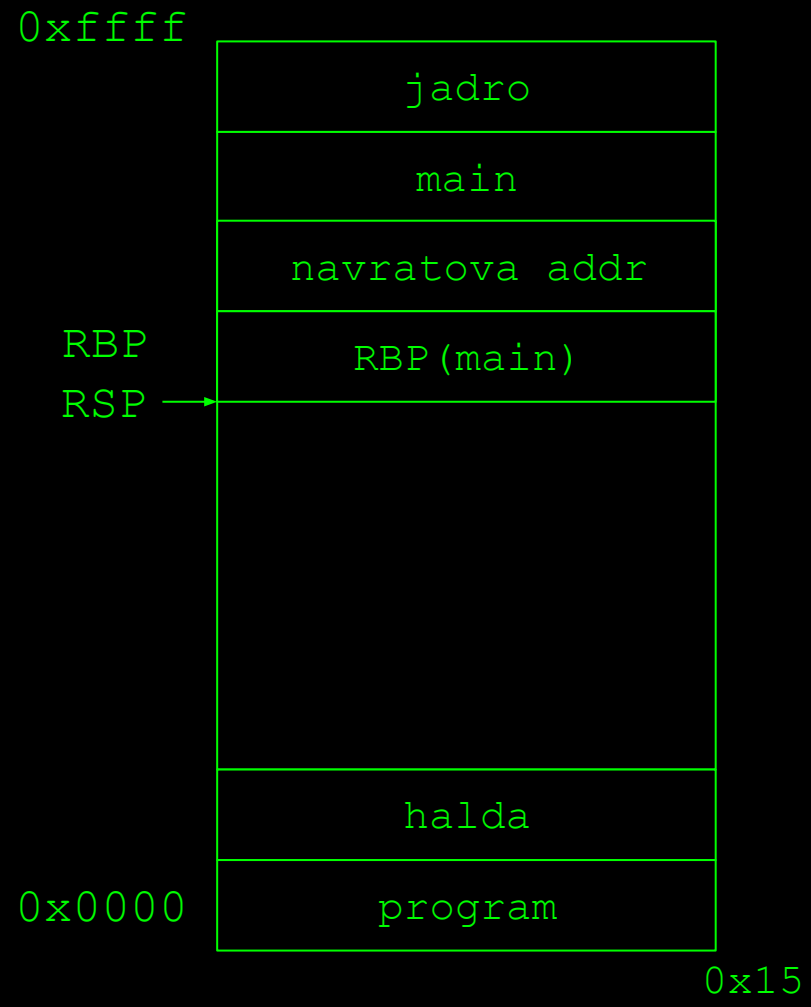
```
call foo

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

vypocty…

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



# >zasobnik

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

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

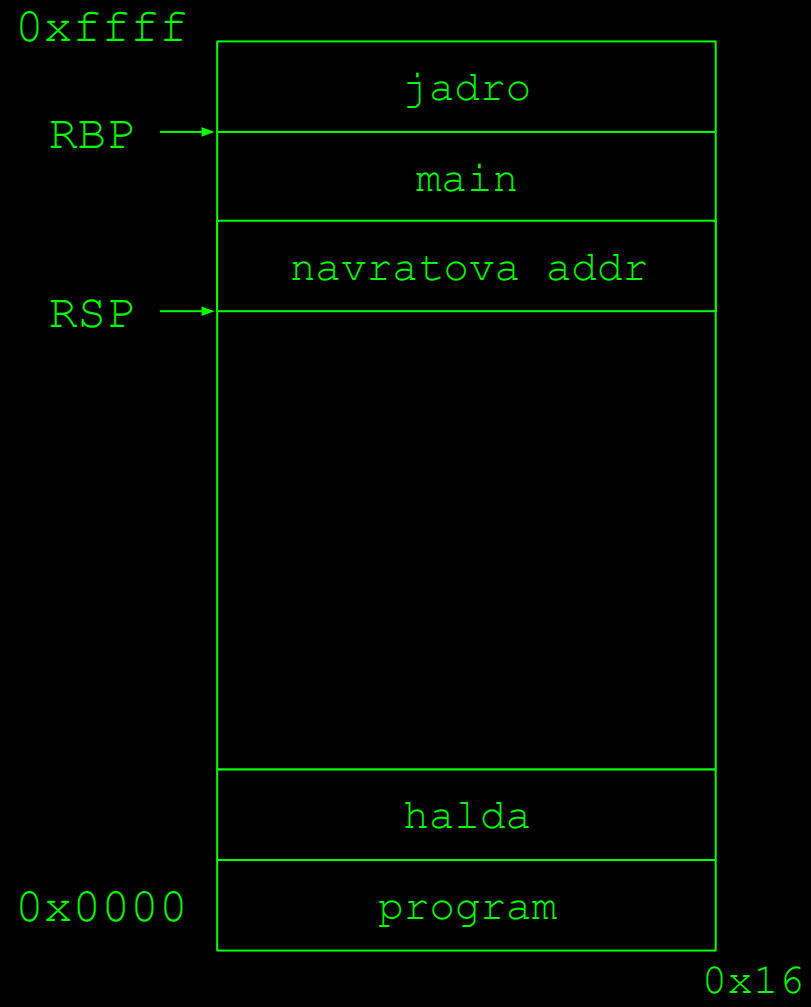
```
call foo

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;
}
```

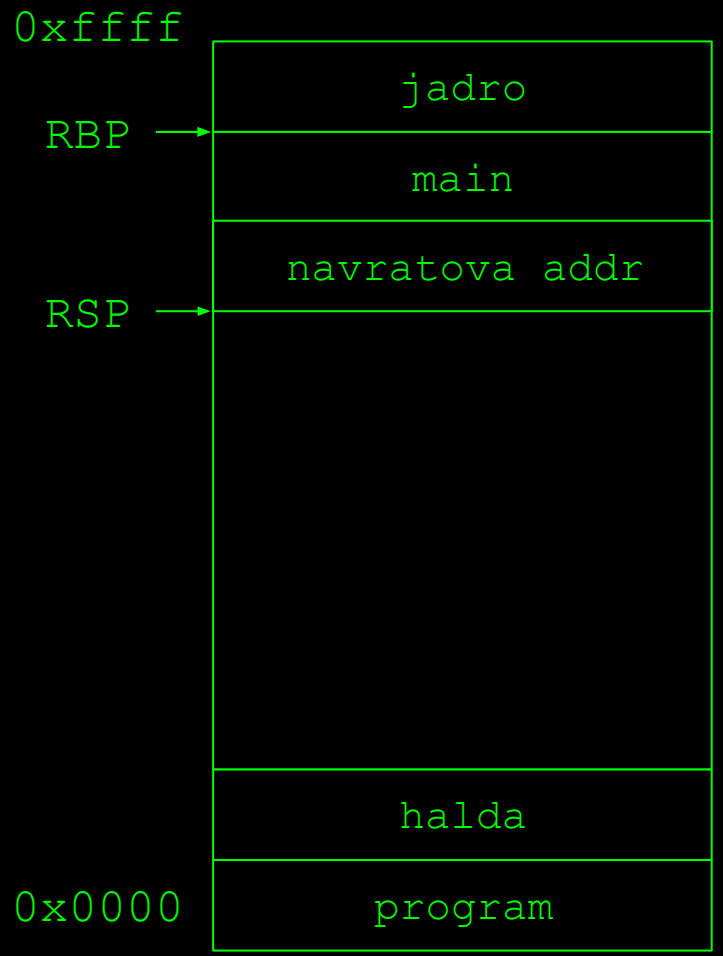
```
call foo

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
```



# >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
```

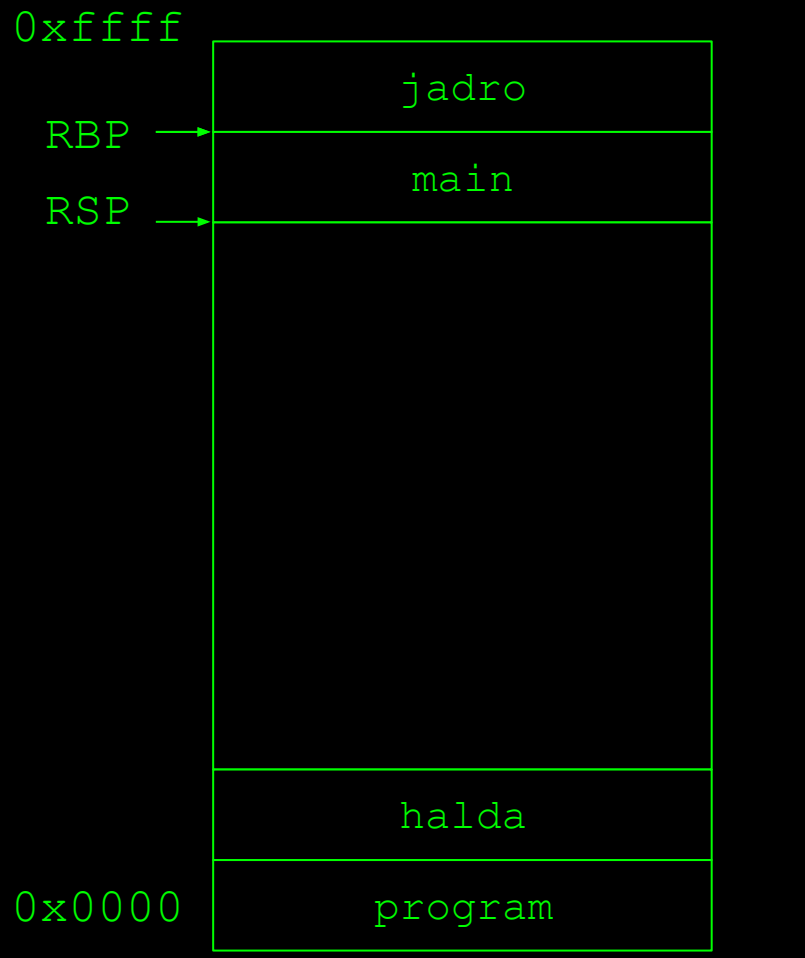



ret = nacita do RIP vrch zasobnika

# >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

# >nastroje

>staticka analyza (bez spustenia)
- >Binary Ninja[1] (cloud verzia)
- >IDA Free 7.6
- >Ghidra
- >Radare2

>dynamicka analyza (so spustenim)
- >strace
- >GDB

[1]https://cloud.binary.ninja/

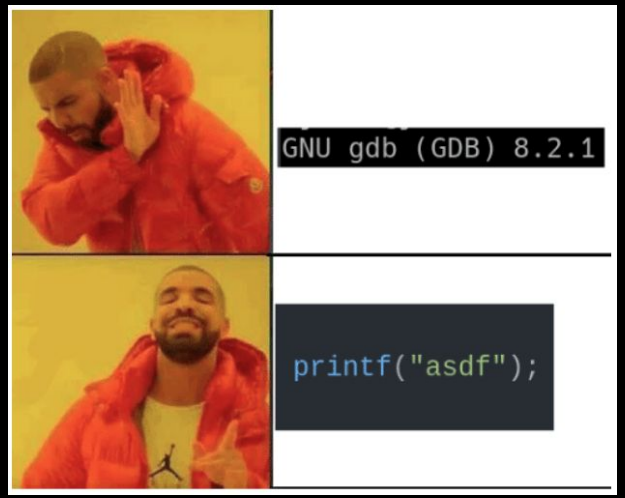
# >dynamicka analyza

>kontrola systemovych volani

```
strace ./level_1.0
```

>GDB (pwndbg plugin)

```
echo source /opt/pwndbg/gdbinit.py >> ~/.gdbinit
```



# >GDB

```
>gdb program
>si (Step Instruction) -> dalsia instrukcia (vnorenie do CALL)
>ni (Next Instruction) -> dalsia instrukcia (preskocenie CALL)
>prehliadanie registrov:
    x/gx $rsp
    x/8b $rax
    x/20i $rip
>breakpointy:
    >manualne instrukcia int3 (0xcc) v kode
    >navestia/mena funkcii v kode (break navestie)
    >break *adresa (break *0x1337000)
>dalsie prikazy[1]
```

[1] https://users.ece.utexas.edu/~adnan/gdb-refcard.pdf

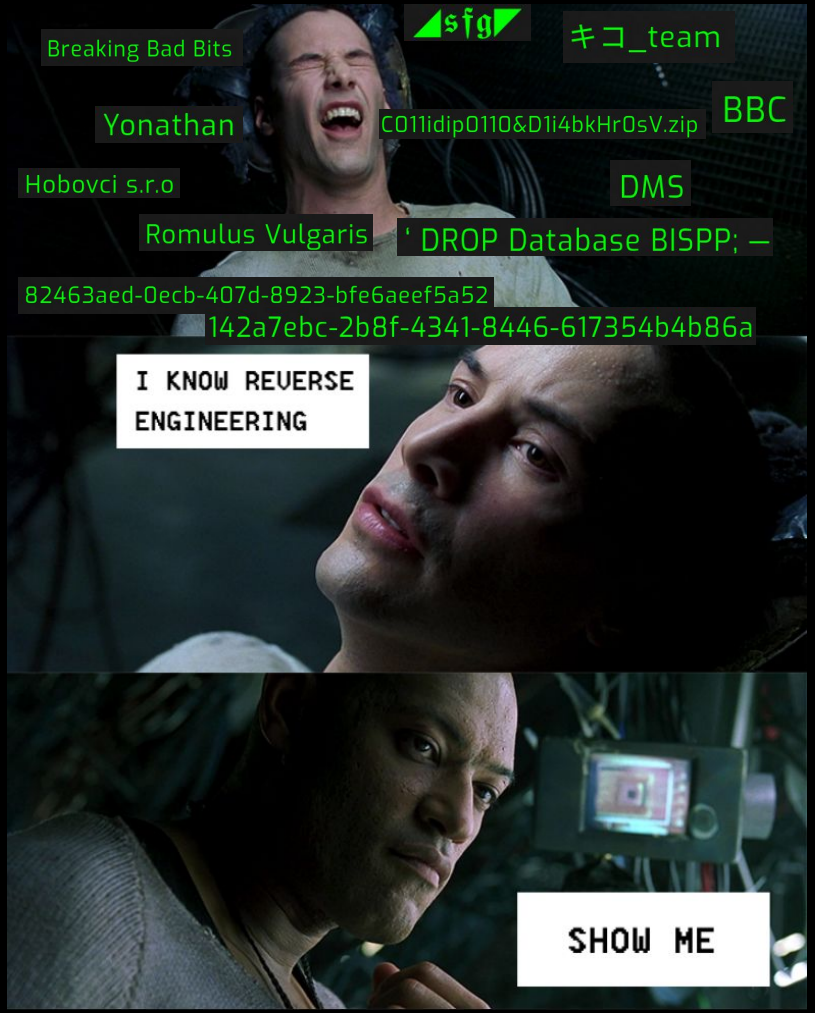
# >ulohy

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

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

# >vela stastia

deadline: 5.3.2024 13:37




0x1d