

# Od hamburgeru ke krávě aneb jak z binárky získat zdroják

Petr Zemek

Fakulta informačních technologií VUT v Brně  
Božetěchova 2, 612 00 Brno, ČR  
<http://www.fit.vutbr.cz/~izemek>



## Petr Zemek

- Ph.D. student @ VUT FIT (od 2010)
  - teorie formálních jazyků
- vývojař v AVG Technologies a člen projektu Lissom @ VUT FIT (od 2011)
  - vývoj a výzkum v oblasti dekompilace



## Odborná orientace

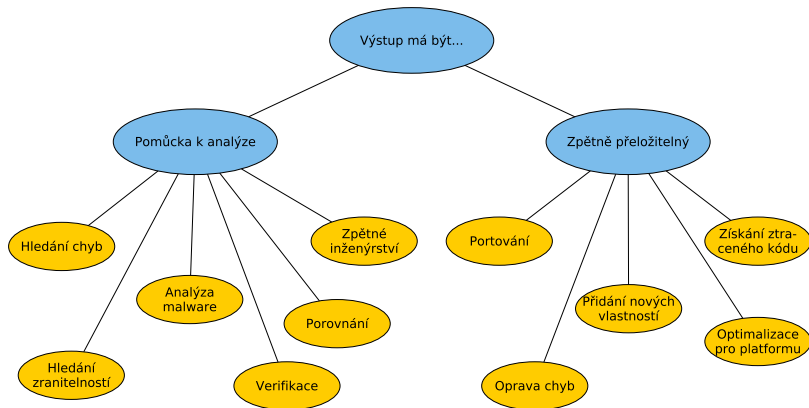
- teoretická informatika, diskrétní matematika
- programování (C, C++, Python, Haskell a další)
- zpětné inženýrství, dekompilace
- operační systémy (GNU/Linux)

- Co je to ta dekompilace?
- K čemu je to dobré?
- Od překladu k dekompilaci
- Seznámení s dekompilátorem projektu Lissom
- Prohlídka online dekompilační služby

Překlad (kompilace)



Zpětný překlad (dekompilace)



Výstup disassembleru je

- nudný
- nepřenositelný
- opakující se
- náchylný k chybám
- vyžaduje speciální znalosti
- už jsem říkal opakující se?

```
mov     dword ptr [ebp-0x17b5c],eax
mov     edx,dword ptr [ebp-0x17b5c]
mov     dword ptr [ebp-0x14fbc],edx
mov     eax,dword ptr [ebp-0x14fbc]
push    eax
call    dword ptr ds:0x405030
mov     dword ptr [ebp-0x17b60],eax
mov     ecx,dword ptr [ebp-0x17b60]
mov     dword ptr [ebp-0x150bc],ecx
push    0x4063ec
lea     ecx,[ebp-0x14fb0]
call    0x401000
mov     dword ptr [ebp-0x17b64],eax
mov     edx,dword ptr [ebp-0x17b64]
mov     dword ptr [ebp-0x14fbc],edx
mov     eax,dword ptr [ebp-0x14fb4]
push    eax
call    dword ptr ds:0x405004
mov     ecx,dword ptr [ebp-0x14fbc]
push    ecx
mov     edx,dword ptr [ebp-0x150bc]
push    edx
call    dword ptr ds:0x40503c
mov     dword ptr [ebp-0x17b68],eax
mov     eax,dword ptr [ebp-0x17b68]
mov     dword ptr [ebp-0x14c80],eax
mov     ecx,dword ptr [ebp-0x14c94]
mov     dword ptr [ebp-0x150c4],ecx
```

Výstup disassembleru je

- nudný
- nepřenositelný
- opakuje se
- náchylný k chybám
- vyžaduje speciální znalosti
- už jsem říkal opakuje se?

... I přesto ho někteří milují :).



```
mov     dword ptr [ebp-0x17b5c], eax
mov     edx, dword ptr [ebp-0x17b5c]
mov     dword ptr [ebp-0x14fbc], edx
mov     eax, dword ptr [ebp-0x14fbc]
push    eax
call    dword ptr ds:0x405030
mov     dword ptr [ebp-0x17b60], eax
mov     ecx, dword ptr [ebp-0x17b60]
mov     dword ptr [ebp-0x150bc], ecx
push    0x4063ec
lea     ecx, [ebp-0x14fb0]
call    0x401000
mov     dword ptr [ebp-0x17b64], eax
mov     edx, dword ptr [ebp-0x17b64]
mov     dword ptr [ebp-0x14fbc], edx
mov     eax, dword ptr [ebp-0x14fb4]
push    eax
call    dword ptr ds:0x405004
mov     ecx, dword ptr [ebp-0x14fbc]
push    ecx
mov     edx, dword ptr [ebp-0x150bc]
push    edx
call    dword ptr ds:0x40503c
mov     dword ptr [ebp-0x17b68], eax
mov     eax, dword ptr [ebp-0x17b68]
mov     dword ptr [ebp-0x14c80], eax
mov     ecx, dword ptr [ebp-0x14c94]
mov     dword ptr [ebp-0x150c4], ecx
```

module.c



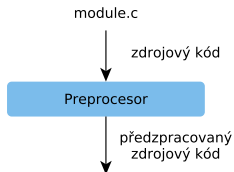
zdrojový kód

```
// Contents of module.c:
```

```
// Returns the answer.
```

```
int func() {  
    return 40 + 2;  
}
```



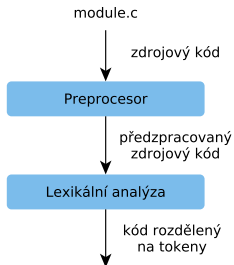


// Contents of module.c:

```
// Returns the answer.  
int func() {  
    return 40 + 2;  
}
```



```
int func() {  
    return 40 + 2;  
}
```

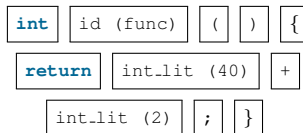


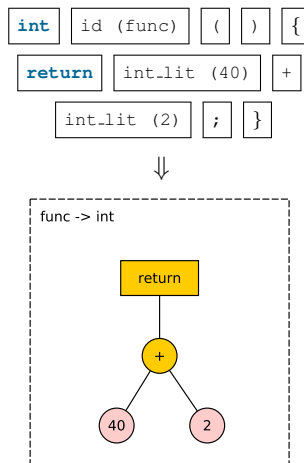
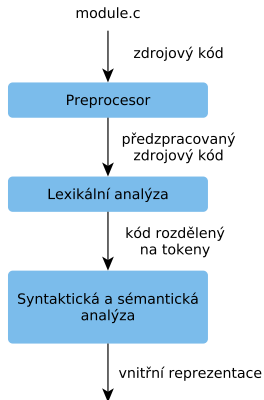
// Contents of module.c:

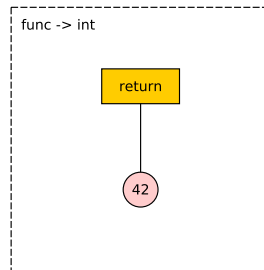
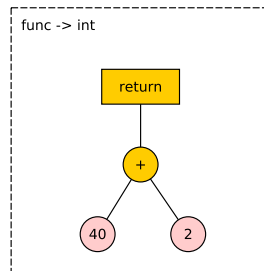
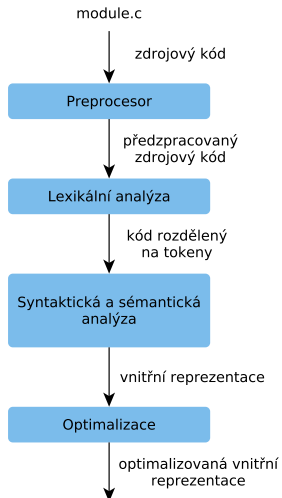
```
// Returns the answer.  
int func() {  
    return 40 + 2;  
}
```

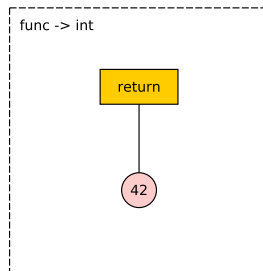
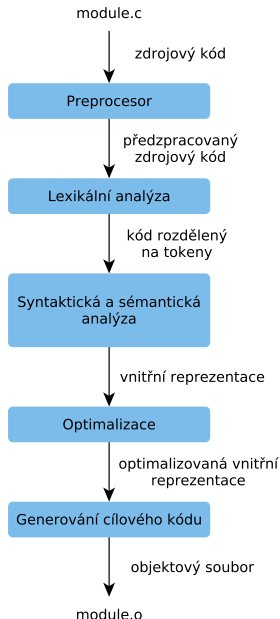


```
int func() {  
    return 40 + 2;  
}
```

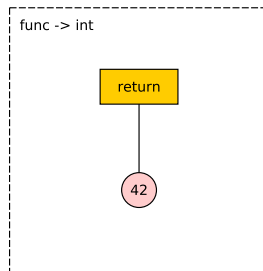
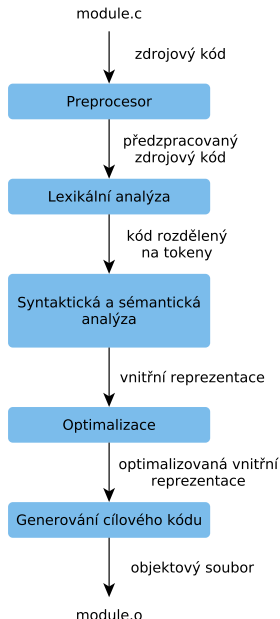








7f 45 4c 46 01 01 01 00 00 00 00 00  
01 00 03 00 01 00 00 00 00 00 00 00  
...



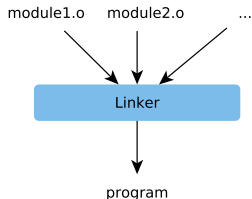
```
7f 45 4c 46 01 01 01 00 00 00 00 00  
01 00 03 00 01 00 00 00 00 00 00 00  
...
```

program.o: file format elf32-i386

Disassembly of section .text:

00000000 <func>:

0: 55	<b>push</b> %ebp
1: 89 e5	<b>mov</b> %esp, %ebp
3: b8 2a 00 00 00	<b>mov</b> \$0x2a, %eax
8: 5d	<b>pop</b> %ebp
9: c3	<b>ret</b>



```
... ; skipped 126 lines
80483cd: 66 90          xchg    %ax,%ax
80483cf: 90             nop
80483d0: 55             push    %ebp
80483d1: 89 e5          mov     %esp,%ebp
80483d3: b8 2a 00 00 00 mov     $0x2a,%eax
80483d8: 5d             pop     %ebp
80483d9: c3             ret
80483da: 66 90          xchg    %ax,%ax
80483dc: 55             push    %ebp
80483dd: 89 e5          mov     %esp,%ebp
80483df: 83 e4 f0       and     $0xfffffffff0,%esp
80483e2: e8 e9 ff ff ff call    80483d0
80483e7: b8 00 00 00 00 mov     $0x0,%eax
80483ec: c9             leave
80483ed: c3             ret
80483ee: 66 90          xchg    %ax,%ax
80483f0: 55             push    %ebp
80483f1: 57             push    %edi
80483f2: 31 ff          xor     %edi,%edi
80483f4: 56             push    %esi
80483f5: 53             push    %ebx
... ; skipped 50 more lines
```



- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant



- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant
- nerozhodnutelnost mnoha problémů
  - separace kódu od dat  $\Leftrightarrow$  problém zastavení TS

- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant
- nerozhodnutelnost mnoha problémů
  - separace kódu od dat  $\Leftrightarrow$  problém zastavení TS
- nízkoúrovňové operace
  - příznaky, subregistry (`rax`, `eax`, `ax`, `ah`, `al`)
  - nepřímé skoky

- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant
- nerozhodnutelnost mnoha problémů
  - separace kódu od dat  $\Leftrightarrow$  problém zastavení TS
- nízkoúrovňové operace
  - příznaky, subregistry (`rax`, `eax`, `ax`, `ah`, `al`)
  - nepřímé skoky
- obfuskace
  - zdrojového kódu
  - binárky

- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant
- nerozhodnutelnost mnoha problémů
  - separace kódu od dat  $\Leftrightarrow$  problém zastavení TS
- nízkoúrovňové operace
  - příznaky, subregistry (`rax`, `eax`, `ax`, `ah`, `al`)
  - nepřímé skoky
- obfuskace
  - zdrojového kódu
  - binárky
- packing, anti-debugging ochrany

- ztráta důležitých informací
  - komentáře, makra, direktivy, ...
  - typy, znaménkovost
  - vysokoúrovňové konstrukce
  - jména proměnných, funkcí, symbolických konstant
- nerozhodnutelnost mnoha problémů
  - separace kódu od dat  $\Leftrightarrow$  problém zastavení TS
- nízkoúrovňové operace
  - příznaky, subregistry (`rax`, `eax`, `ax`, `ah`, `al`)
  - nepřímé skoky
- obfuskace
  - zdrojového kódu
  - binárky
- packing, anti-debugging ochrany
- škodlivý software (malware)



- mnoho architektur
  - x86, x86-64, ARM, MIPS, PowerPC, SPARC, ...
  - CISC, RISC
  - bitová šířka, instrukční sada, endianita, FP
  - verze, rozšíření

- mnoho architektur
  - x86, x86-64, ARM, MIPS, PowerPC, SPARC, ...
  - CISC, RISC
  - bitová šířka, instrukční sada, endianita, FP
  - verze, rozšíření
- různá binární rozhraní (ABI)
  - bitové šířky, layout a zarovnání datových typů
  - volací konvence
  - mnoho způsobů, jak udělat totéž

```
mov    eax, 0
and    eax, 0
mul    eax, 0
sub    eax, eax
xor    eax, eax
lea    eax, [0]
```

- mnoho architektur
  - x86, x86-64, ARM, MIPS, PowerPC, SPARC, ...
  - CISC, RISC
  - bitová šířka, instrukční sada, endianita, FP
  - verze, rozšíření
- různá binární rozhraní (ABI)
  - bitové šířky, layout a zarovnání datových typů
  - volací konvence
  - mnoho způsobů, jak udělat totéž

```
mov    eax, 0
and    eax, 0
mul    eax, 0
sub    eax, eax
xor    eax, eax
lea    eax, [0]
```

- různé souborové formáty
  - ELF, COFF, PE, Mach-O, ...



- mnoho programovacích jazyků a překladačů
  - C: GCC, Clang, MSVC, ICC, ...



- mnoho programovacích jazyků a překladačů
  - C: GCC, Clang, MSVC, ICC, ...
- různé typy optimalizací
  - GCC: `-O0`, `-O1`, `-O2`, `-O3`, `-Os`, `-Og`, `-Ofast`



- mnoho programovacích jazyků a překladačů
  - C: GCC, Clang, MSVC, ICC, ...
- různé typy optimalizací
  - GCC: `-O0`, `-O1`, `-O2`, `-O3`, `-Os`, `-Og`, `-Ofast`
- idiomy

`4 * x`       $\Rightarrow$       `x << 2`

`x % 2`       $\Rightarrow$       `((x >> 31) + x) & 1 - (x >> 31)`

- mnoho programovacích jazyků a překladačů
  - C: GCC, Clang, MSVC, ICC, ...
- různé typy optimalizací
  - GCC: -O0, -O1, -O2, -O3, -Os, -Og, -Ofast
- idiomy

`4 * x`       $\Rightarrow$       `x << 2`

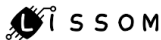
`x % 2`       $\Rightarrow$       `((x >> 31) + x) & 1 - (x >> 31)`

A to zdaleka není vše... 





A nebo ne?



- vyvíjený od roku 2011
- Lissom @ VUT FIT ve spolupráci s AVG Technologies
- cíl: generická dekompilace binárního kódu
- vstup:
  - platformně závislý binární program
    - architektury: x86 (i386), ARM, ARM-Thumb (1+2), MIPS
    - souborové formáty: ELF, PE
    - aplikace napsané v C/assembleru
  - popis platformy (CPU, ABI, knihovny, ...)
- výstup:
  - uniformní výstup v jazyku vyšší úrovně (C, Python')
  - disassemblovaný kód
  - grafické reprezentace (graf toku řízení, graf volání)
  - statistiky

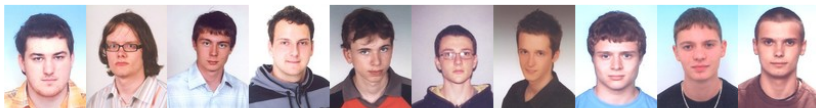
## Vedení



## Jádro



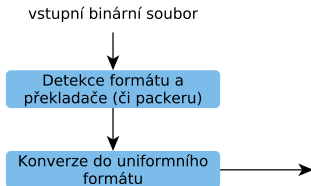
## Diplomanti (VUT FIT)



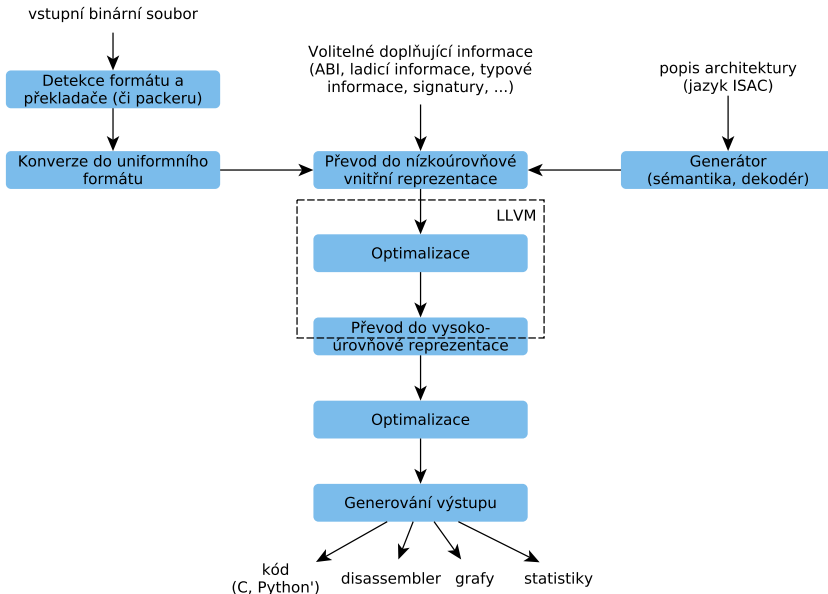
a další...

<http://www.fit.vutbr.cz/research/groups/lissom/>









## Původní kód:

```
#include <stdio.h>
#include <stdlib.h>

int my_sum(int i) {
    int b = rand();
    int c = rand();
    int d = b - c;
    return (b + c) * (i + d);
}

int main(int argc, char **argv) {
    int a = rand();
    int b = rand() + 2;
    int c = 0;
    if (a > b) {
        printf("TRUE");
        c = my_sum(a);
    } else {
        printf("FALSE");
        c = my_sum(b);
    }
    return a - b - c;
}
```

## Dekompilovaný kód:

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int32_t function_8218(int32_t a) {
    int32_t x = rand();
    int32_t y = rand();
    return (x + a - y) * (x + y);
}

int main(int argc, char **argv) {
    int32_t apple = rand();
    int32_t banana = rand() + 2;
    if (apple > banana) {
        printf("TRUE");
        return apple - banana -
            function_8218(apple);
    }
    printf("FALSE");
    return apple - banana -
        function_8218(banana);
}
```

## Původní kód:

```

1  int factorial(int n) {
2      if (n == 0)
3          return 1;
4      return n * factorial(n - 1);
5  }
6
7  int calculate(int a, int b) {
8      return a * factorial(b);
9  }
10
11 int main(int argc, char **argv) {
12     // ...
13     // ...
14     // ...
15     // ...
16     // ...
17 }

```

## Dekompilovaný kód:

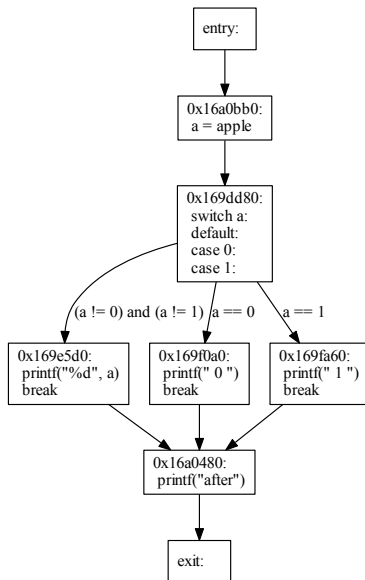
```

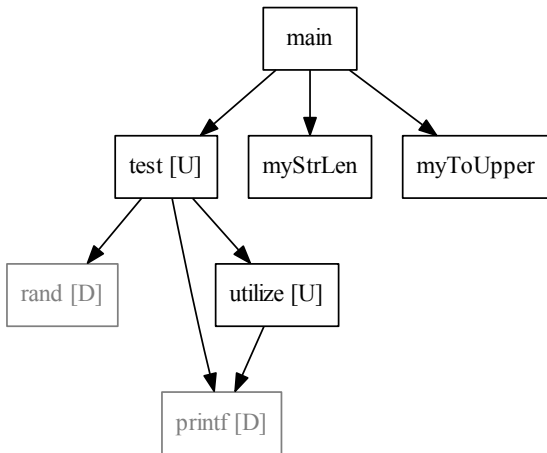
// From module: test.c
// Address range: 0x401560 - 0x401585
// Line range: 1 - 5
int32_t factorial(int32_t n) {
    int32_t result;
    if (n != 0) {
        result = factorial(n - 1) * n;
    } else {
        result = 1;
    }
    return result;
}

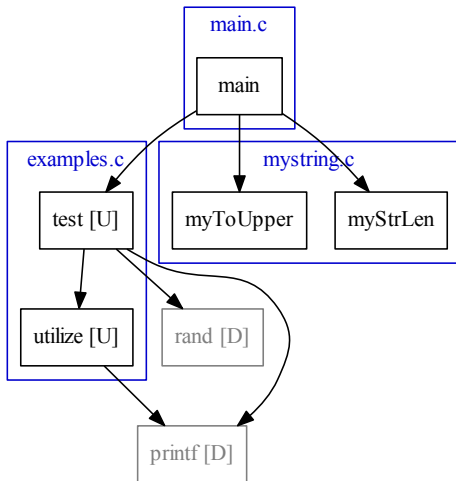
// From module: test.c
// Address range: 0x401587 - 0x40159c
// Line range: 7 - 9
int32_t calculate(int32_t a, int32_t b) {
    return factorial(b) * a;
}

// From module: test.c
// Address range: 0x40159e - 0x401606
// Line range: 11 - 17
int main(int argc, char **argv) {
    // ...
}

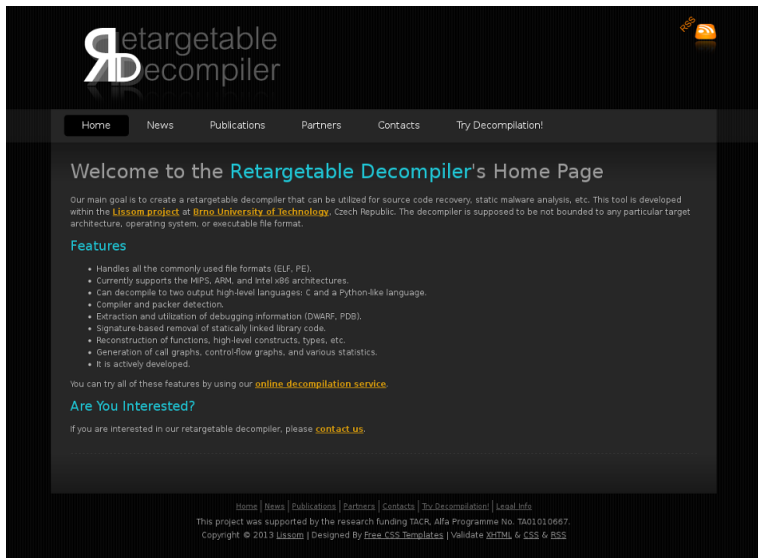
```











The screenshot shows the homepage of the Retargetable Decompiler. At the top left is the logo, which consists of a stylized 'RD' followed by the text 'retargetable decompiler'. At the top right is a small orange icon with the text 'PSS' above it. Below the logo is a navigation bar with links: Home, News, Publications, Partners, Contacts, and Try Decompile!. The main content area has a heading 'Welcome to the Retargetable Decompiler's Home Page'. Below this is a paragraph describing the tool's purpose and its development at the Lissom project at Brno University of Technology. A 'Features' section lists several capabilities, such as handling various file formats, supporting different architectures, and performing static analysis. It also mentions an 'online decompilation service'. A section titled 'Are You Interested?' encourages users to contact the developers. At the bottom, there is a footer with navigation links, a statement of support from TACR and Alfa Programme No. TA01010667, and copyright information for 2013.

**retargetable decompiler**

PSS

Home News Publications Partners Contacts Try Decompile!

## Welcome to the Retargetable Decompiler's Home Page

Our main goal is to create a retargetable decompiler that can be utilized for source code recovery, static malware analysis, etc. This tool is developed within the [Lissom project](#) at [Brno University of Technology](#), Czech Republic. The decompiler is supposed to be not bounded to any particular target architecture, operating system, or executable file format.

### Features

- Handles all the commonly used file formats (ELF, PE).
- Currently supports the MIPS, ARM, and Intel x86 architectures.
- Can decompile to two output high-level languages: C and a Python-like language.
- Compiler and packer detection.
- Extraction and utilization of debugging information (DWARF, PDB).
- Signature-based removal of statically linked library code.
- Reconstruction of functions, high-level constructs, types, etc.
- Generation of call graphs, control-flow graphs, and various statistics.
- It is actively developed.

You can try all of these features by using our [online decompilation service](#).

### Are You Interested?

If you are interested in our retargetable decompiler, please [contact us](#).

[Home](#) | [News](#) | [Publications](#) | [Partners](#) | [Contacts](#) | [Try Decompile!](#) | [Legal Info](#)

This project was supported by the research funding TACR, Alfa Programme No. TA01010667.  
Copyright © 2013 [Lissom](#) | Designed By [Free CSS Templates](#) | Validate [XHTML](#) & [CSS](#) & [RSS](#)

<http://decompiler.fit.vutbr.cz/>



*That's All  
Folks!*

