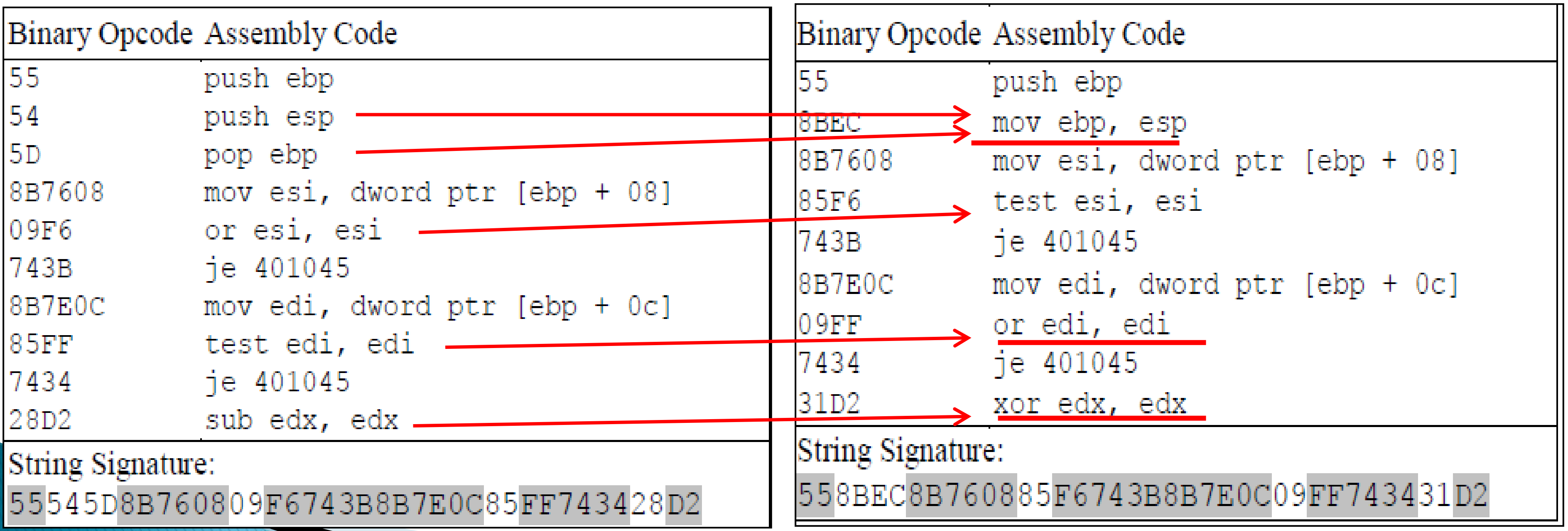
Andrea Di Sorbo, PhD student
Sicurezza delle Reti e dei Sistemi Software
CdLM in Ingegneria Informatica
Università degli Studi del Sannio
(disorbo@unisannio.it)

Metamorphic Malware

Implementation of a Metamorphic Engine

Obfuscation techniques

3. **Instruction Replacement:** This method actually substitutes some instructions with their equivalent instructions in newer copies. This method is like using different synonyms in human language. Win95.Bistro used this technique.



Binary Opcode	Assembly Code
55	push ebp
54	push esp
5D	pop ebp
8B7608	mov esi, dword ptr [ebp + 08]
09F6	or esi, esi
743B	je 401045
8B7E0C	mov edi, dword ptr [ebp + 0c]
85FF	test edi, edi
7434	je 401045
28D2	sub edx, edx
String Signature:	
55545D8B760809F6743B8B7E0C85FF743428D2	

Binary Opcode	Assembly Code
55	push ebp
8BEC	mov ebp, esp
8B7608	mov esi, dword ptr [ebp + 08]
85F6	test esi, esi
743B	je 401045
8B7E0C	mov edi, dword ptr [ebp + 0c]
09FF	or edi, edi
7434	je 401045
31D2	xor edx, edx
String Signature:	
558BEC8B760885F6743B8B7E0C09FF743431D2	

Examples of instruction replacements

- Some examples of readily realizable replacements:
  - Replace register moves with push/pop sequences

```
movl %eax, %ebx  →  pushl %eax
                    popl %ebx
```

  - xor/sub replacement

```
xorl %edx, %edx  →  subl %edx,%edx
```

  - or/test replacement

```
testl %eax, %eax  →  orl %eax,%eax
```

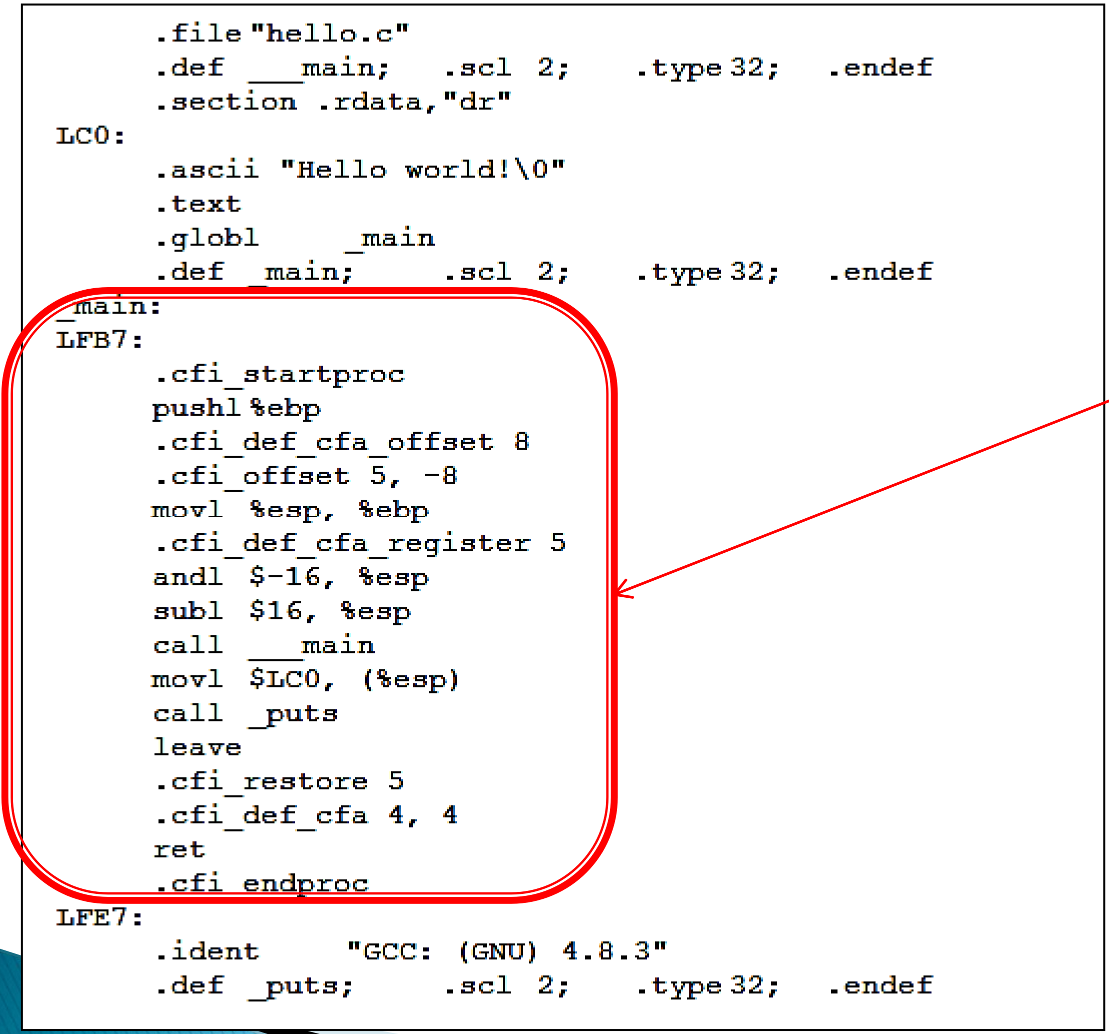
  - add/sub (with complement operand) replacement

```
addl $2, %eax   →  subl $-2, %eax
```

Exercise 3

- In the metamorphic engine implemented in the previous exercise, add a new method to perform Instruction Replacement operations on the target code.
- The new method
  1. Takes in input a file in assembly code (hello_mutation2.s)
  2. Returns in output a new variant (hello_mutation3.s) of the input file obtained through operations of Instruction Replacement (Each execution may produce a different variant of the original file)
- Recompile the resulting file and verify that the two executions (hello_mutation2.s and hello_mutation3.s) are equivalent.

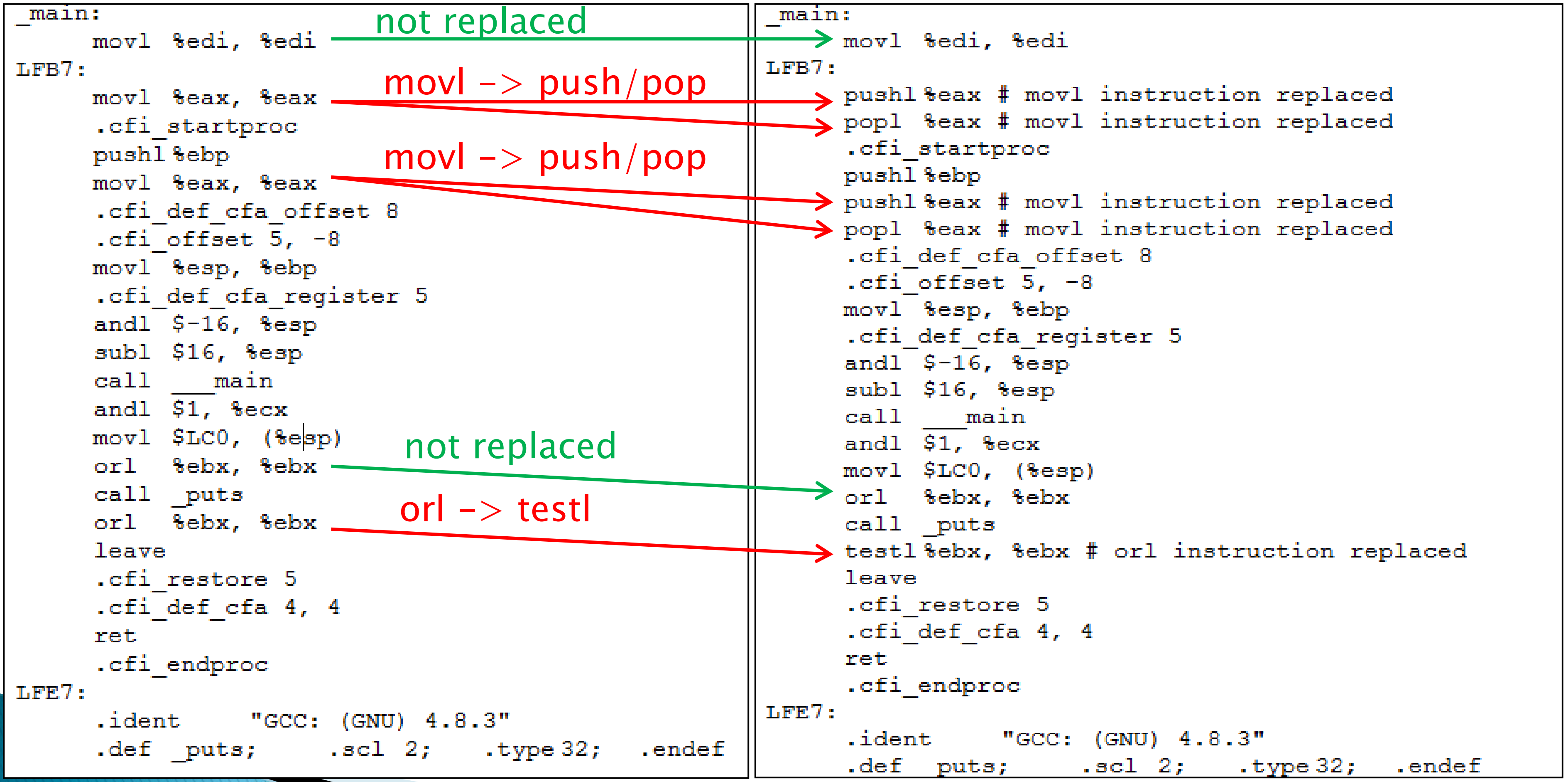
hello.s



```
        .file "hello.c"
        .def  ___main;   .scl  2;    .type 32;   .endef
        .section .rdata,"dr"
LC0:
        .ascii "Hello world!\0"
        .text
        .globl      _main
        .def  _main;      .scl  2;    .type 32;   .endef
_main:
LFB7:
        .cfi_startproc
        pushl %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        movl  $LC0, (%esp)
        call  _puts
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident      "GCC: (GNU) 4.8.3"
        .def  _puts;      .scl  2;    .type 32;   .endef
```

Target Code. In this code block we can apply obfuscation techniques

hello_mutation3.s



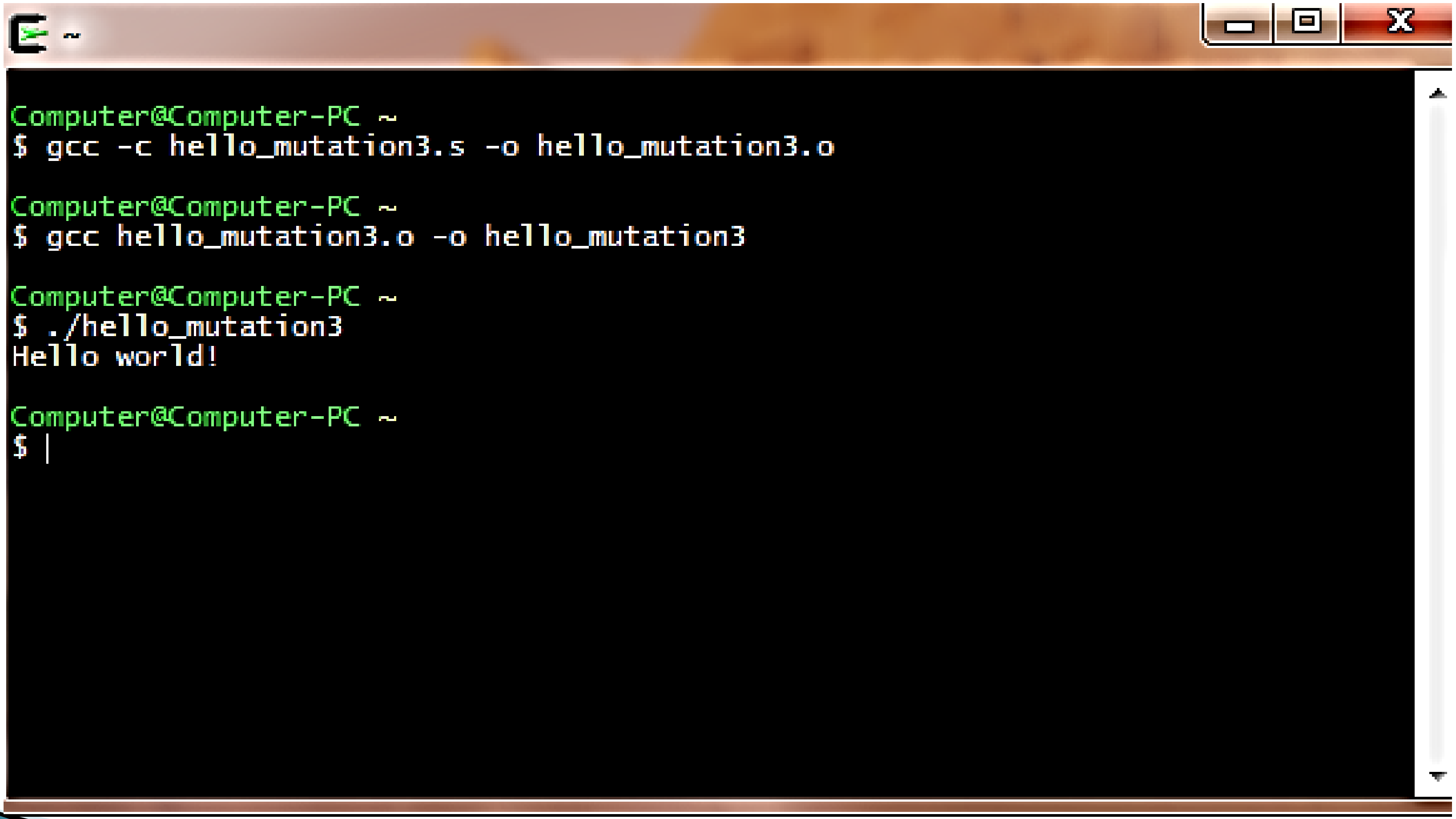
hello_mutation2.s

```
_main:
        movl  %edi, %edi
LFB7:
        movl  %eax, %eax
        .cfi_startproc
        pushl %ebp
        movl  %eax, %eax
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        andl  $1, %ecx
        movl  $LC0, (%esp)
        orl   %ebx, %ebx
        call  _puts
        orl   %ebx, %ebx
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident     "GCC: (GNU) 4.8.3"
        .def  _puts;    .scl  2;    .type 32;   .endef
```

hello_mutation3.s

```
_main:
        movl  %edi, %edi
LFB7:
        pushl %eax # movl instruction replaced
        popl  %eax # movl instruction replaced
        .cfi_startproc
        pushl %ebp
        pushl %eax # movl instruction replaced
        popl  %eax # movl instruction replaced
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl  %esp, %ebp
        .cfi_def_cfa_register 5
        andl  $-16, %esp
        subl  $16, %esp
        call  ___main
        andl  $1, %ecx
        movl  $LC0, (%esp)
        orl   %ebx, %ebx
        call  _puts
        testl %ebx, %ebx # orl instruction replaced
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
LFE7:
        .ident     "GCC: (GNU) 4.8.3"
        .def  _puts;    .scl  2;    .type 32;   .endef
```

Recompiling and executing hello_mutation3.s



```
Computer@Computer-PC ~
$ gcc -c hello_mutation3.s -o hello_mutation3.o

Computer@Computer-PC ~
$ gcc hello_mutation3.o -o hello_mutation3

Computer@Computer-PC ~
$ ./hello_mutation3
Hello world!

Computer@Computer-PC ~
$ |
```