



Thai text placeholder

Thai text placeholder

Thai text placeholder
<dennis(a)yurichev.com>



©2013-2015, Thai text placeholder.

<http://creativecommons.org/licenses/by-nc-nd/3.0/>.

(2 เมษายน พ.ศ. 2559).

[beginners.re.](#)

@yurichev^{1, 2}.

[facebook.](#)

¹twitter.com/yurichev

²yurichev.com

beginners.re.

สารบัญ

I	รูปแบบของโค้ด	1
1		3
2		4
2.1	x86	4
3		5
3.1	x86	5
3.1.1	MSVC	5
3.2	x86-64	6
3.2.1	MSVC---x86-64	6
3.3		7
4		8
4.1		8
5		9
5.1		9
5.2		9
5.2.1		9
5.2.2		10
5.2.3		11
5.2.4	x86:	11
5.2.5	(Windows) SEH	12
5.2.6		12
5.2.7		12
5.3		12
6	printf()	13
6.1	x86	13
6.1.1	x86:	13
6.1.2	x64:	14
6.2		14
6.3		15
7	scanf()	16
7.1		16
7.1.1		16
7.1.2	x86	16
7.1.3	x64	17

สารบัญ	สารบัญ
7.2	18
7.2.1	MSVC: x86 18
7.2.2	MSVC: x64 19
7.3	20
7.3.1	MSVC: x86 20
7.3.2	MSVC: x86 + Hiew 22
7.3.3	MSVC: x64 23
7.4	24
8	25
8.1	x86 25
8.1.1	MSVC 25
8.2	x64 26
8.2.1	MSVC 26
9	28
9.1	28
9.2	29
10	30
10.1	31
11	32
11.1	32
11.1.1	x86 32
11.2	36
11.2.1	MSVC 36
11.3	37
11.3.1	x86 37
11.3.2	38
11.4	39
11.4.1	32-bit 39
11.5	40
11.5.1	x86 40
11.5.2	40
12	switch()/case/default 41
12.1	41
12.1.1	x86 41
12.1.2	43
12.2	43
12.2.1	x86 43
12.2.2	46
12.3	46
12.3.1	MSVC 47
12.4	Fall-through 48
12.4.1	MSVC x86 49
13	51
13.1	51
13.1.1	x86 51
13.1.2	52
13.2	52

13.2.1	53
13.3	53
14	55
14.1	55
14.1.1	55
15	57
15.1	57
15.1.1	57
15.1.2	57
15.1.3	58
15.2	60
15.2.1	60
16	61
16.1	61
16.1.1	61
16.2	62
16.2.1	62
16.2.2	63
16.3	66
16.4	66
16.4.1	67
16.5	68
16.5.1	69
16.5.2	70
16.5.3	71
16.6	73
17	74
17.1	74
17.1.1	74
17.2	75
17.2.1	75
17.3	76
17.4	76
17.4.1	77
17.4.2	78
17.5	80
17.5.1	80
17.5.2	80
17.5.3	81
17.5.4	81
17.5.5	81
17.5.6	81
18	82
18.1	82
18.2	83
19	85
19.1	85

สารบัญ	สารบัญ
19.1.1	86
19.2	87
19.3	89
19.3.1 x86	89
19.3.2	91
19.4	92
19.5	93
19.5.1	93
20	96
20.1	96
20.1.1 x86	96
20.2	96
20.2.1 x86	97
20.3	97
20.3.1 x86	98
20.4	99
20.4.1 x86	99
20.5	99
20.5.1 x86	99
21	101
21.1 x86-64	101
II	102
22	104
23	106
III	107
24 (win32)	109
24.1 Windows API	109
24.2 tracer:	110
25	111
25.1	111
25.1.1	111
25.1.2 Borland Delphi	111
25.1.3 Unicode	112
25.1.4 Base64	114
25.2	114
25.3	115
26	116
27	117
27.1 Magic numbers	117
27.1.1 DHCP	117
27.2	118

28		119
29		121
29.1		121
29.2		121
30		123
31		124
31.1		124
31.2		124
31.3		124
31.3.1		125
31.3.2		125
IV		126
32		127
32.1	IDA	127
33		128
33.1	tracer	128
34		129
35		130
V		131
36		132
36.1	Windows	132
36.2		132
36.3	x86 / x86-64	132
36.4	ARM	132
36.5		132
37		133
37.1	Windows	133
38		134
		136
39		136
		139
Thai text placeholder		140

สารบัญ	สารบัญ
ดรรชนี	141
Thai text placeholder	143

... <http://challenges.re>.



dennis(a)yurichev.com, dennis.yurichev.

Thai text placeholder

- ?It's very well done .. and for free .. amazing?³ Daniel Bilar, Siege Technologies, LLC.
- ?... excellent and free?⁴ Pete Finnigan, Oracle RDBMS.
- ?... book is interesting, great job!? Michael Sikorski, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*.
- ?... my compliments for the very nice tutorial!? Herbert Bos, Vrije Universiteit Amsterdam, *Modern Operating Systems (4th Edition)*.
- ?... It is amazing and unbelievable.? Luis Rocha, CISSP / ISSAP, Technical Manager, Network & Information Security at Verizon Business.
- ?Thanks for the great work and your book.? Joris van de Vis, SAP Netweaver & Security .
- ?... reasonable intro to some of the techniques?⁵ Mike Stay, Federal Law Enforcement Training Center, Georgia, US.
- ?I love this book! I have several students reading it at the moment, plan to use it in graduate course?⁶ , Research Assistant Professor Dartmouth College
- ?Dennis @Yurichev has published an impressive (and free!) book on reverse engineering?⁷ Tanel Poder, .
- ?This book is some kind of Wikipedia to beginners...? Archer, Chinese Translator, IT Security Researcher.

?herm1t? ,

, Shell Rocket, Zhu Ruijin, Changmin Heo.

Arnaud Patard (rtp #debian-arm IRC), .

³twitter.com/daniel_bilar/status/436578617221742593

⁴twitter.com/petefinnigan/status/400551705797869568

⁵[reddit](https://www.reddit.com)

⁶twitter.com/sergeybratus/status/505590326560833536

⁷twitter.com/TanelPoder/status/524668104065159169

Antiy Labs (antiy.cn) Archer.

: Byungho Min.

: , , , ``Logxen" , Yuan Jochen Kang, Mal Malakov, Lewis Porter, Jarle Thorsen.

2 * Oleg Vygovsky (50+100 UAH), Daniel Bilar (\$50), James Truscott (\$4.5), Luis Rocha (\$63), Joris van de Vis (\$127), Richard S Shultz (\$20), Jang Minchang (\$20), Shade Atlas (5 AUD), Yao Xiao (\$10), Pawel Szczur (40 CHF), Justin Simms (\$20), Shawn the R0ck (\$27), Ki Chan Ahn (\$50), Triop AB (100 SEK), Ange Albertini (€10+50), Sergey Lukianov (300 RUR), Ludvig Gislason (200 SEK), Gérard Labadie (€40), Sergey Volchkov (10 AUD), Vankayala Vigneswararao (\$50), Philippe Teuwen (\$4), Martin Haerberli (\$10), Victor Cazacov (€5), Tobias Sturzenegger (10 CHF), Sonny Thai (\$15), Bayna ALZaabi (\$75), Redfive B.V. (€25), Joonas Oskari Heikkilä (€5), Marshall Bishop (\$50), Nicolas Werner (€12), Jeremy Brown (\$100), Alexandre Borges (\$25), Vladimir Dikovski (€50), Jiarui Hong (100.00 SEK), Jim Di (500 RUR), Tan Vincent (\$30), Sri Harsha Kandrakota (10 AUD), Pillay Harish (10 SGD), Timur Valiev (230 RUR), Carlos Garcia Prado (€10), Salikov Alexander (500 RUR), Oliver Whitehouse (30 GBP), Katy Moe (\$14), Maxim Dyakonov (\$3), Sebastian Aguilera (€20), Hans-Martin Münch (€15), Jarle Thorsen (100 NOK), Vitaly Osipov (\$100), Yuri Romanov (1000 RUR), Aliaksandr Autayeu (€10), Tudor Azoitei (\$40), Z0vsky (€10), Yu Dai (\$10).

mini-Thai text placeholder

Q: Thai text placeholder

A: ⁸.

Thai text placeholder

Q:

A:

Q:

A:

Q:

A:

Thai text placeholder

Q:

A: (2013 Q3, 2014).

2014 Q2.

Q:

A: (dennis(a)yurichev.com).

.

Byungho Min ([twitter/tais9](https://twitter.com/tais9)).

⁸Thai text placeholder: [Fog13]

: facebook/andydinka.

ภาค I

รูปแบบของโค้ด

เมื่อครั้งที่ผู้แต่งหนังสือเริ่มต้นเรียนรู้ภาษา C และในเวลาถัดมา C++ เขาเคยเขียนโค้ดสั้น ๆ และทำการคอมไพล์ แล้วสังเกตดูผลลัพธ์ที่ได้จากการคอมไพล์ที่เป็นภาษาแอสเซมบลี ซึ่งช่วยให้เข้าใจได้ง่ายขึ้นว่าโค้ดที่เขียนนั้นมีการทำงานอย่างไร เขาทำแบบนี้บ่อยครั้งจนความสัมพันธ์ระหว่างโค้ดภาษา C/C++ กับผลลัพธ์ที่ได้จากการคอมไพล์ประทับอยู่ในจิตใจ มันเป็นการง่ายมากที่จะนึกถึงถึงรูปร่างและฟังก์ชันของโค้ดภาษา C ขึ้นมาทันที บางครั้งเทคนิคนี้อาจจะมีประโยชน์กับคนอื่นก็ได้

บทที่ 1

CPU¹.

:

: CPU . CPU (ISA²).

: CPU. .

:

: CPU (GPR³). ≈ 8 x86, ≈ 16 x86-64, ≈ 16 ARM. . . !

¹Central processing unit

²Instruction Set Architecture

³General Purpose Registers

บทที่ 2

:

Listing 2.1:

```
int f()
{
    return 123;
};
```

2.1 x86

:

Listing 2.2: GCC/MSVC ()

```
f:
    mov     eax, 123
    ret
```

บทที่ 3

“The C programming Language”[Ker88]:

```
#include <stdio.h>

int main()
{
    printf("hello, world\n");
    return 0;
}
```

3.1 x86

3.1.1 MSVC

MSVC 2010:

```
cl 1.cpp /Fa1.asm
```

Listing 3.1: MSVC 2010

```
CONST    SEGMENT
$SG3830 DB      'hello, world', 0AH, 00H
CONST    ENDS
PUBLIC   _main
EXTRN    _printf:PROC
; Function compile flags: /OdtP
_TEXT    SEGMENT
_main    PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG3830
    call    _printf
    add     esp, 4
    xor     eax, eax
    pop     ebp
    ret     0
_main    ENDP
_TEXT    ENDS
```

:

```
#include <stdio.h>

const char $SG3830[]="hello, world\n";
```

```
int main()
{
    printf($SG3830);
    return 0;
}
```

. : 25.1.1 on page 111.

: main().¹.

: CALL _printf.

ADD ESP, 4 ADD ESP, 4

:

Listing 3.2: Oracle RDBMS 10.2 Linux (app.o)

```
.text:0800029A            push    ebx
.text:0800029B            call    qksfroChild
.text:080002A0            pop     ecx
```

XOR EAX, EAX.² MOV EAX, 0---

SUB EAX, EAX,

3.2 x86-64

3.2.1 MSVC---x86-64

:

Listing 3.3: MSVC 2012 x64

```
$SG2989 DB        'hello, world', 0AH, 00H

main    PROC
        sub      rsp, 40
        lea      rcx, OFFSET FLAT:$SG2989
        call     printf
        xor      eax, eax
        add      rsp, 40
        ret      0
main    ENDP
```

. (fastcall). ---. RCX, RDX, R8, R9. .

..

RAX/EAX/AX/AL x86-64:

Thai text placeholder	
RAX ^{x64}	
	EAX
	AX
AH	AL

. ?shadow space?, : 8.2.1 on page 27.

¹ (4 on page 8).

²wikipedia

บทที่ 4

```
push    ebp
mov     ebp, esp
sub     esp, X
```

```
mov     esp, ebp
pop     ebp
ret     0
```

4.1

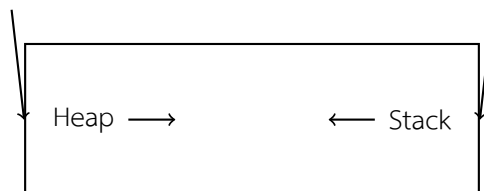
: ?? on page ??.

บทที่ 5

1.

5.1

.
...



[RT74] :

The user-core part of an image is divided into three logical segments. The program text segment begins at location 0 in the virtual address space. During execution, this segment is write-protected and a single copy of it is shared among all processes executing the same program. At the first 8K byte boundary above the program text segment in the virtual address space begins a nonshared, writable data segment, the size of which may be extended by a system call. Starting at the highest address in the virtual address space is a stack segment, which automatically grows downward as the hardware's stack pointer fluctuates.

5.2

5.2.1

x86

```
void f()
{
    f();
};
```

```
c:\tmp6>cl ss.cpp /Fass.asm
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation. All rights reserved.
```

¹wikipedia.org/wiki/Call_stack

```
ss.cpp
c:\tmp6\ss.cpp(4) : warning C4717: 'f' : recursive on all control paths, function will cause ↗
↳ runtime stack overflow
```

...:

```
?f@@YAXXZ PROC ; f
; File c:\tmp6\ss.cpp
; Line 2
    push    ebp
    mov     ebp, esp
; Line 3
    call    ?f@@YAXXZ ; f
; Line 4
    pop     ebp
    ret     0
?f@@YAXXZ ENDP ; f
```

...

```
?f@@YAXXZ PROC ; f
; File c:\tmp6\ss.cpp
; Line 2
$LL3@f:
; Line 3
    jmp     SHORT $LL3@f
?f@@YAXXZ ENDP ; f
```

5.2.2

?cdecl?:

```
push arg3
push arg2
push arg1
call f
add esp, 12 ; 4*3=12
```

ESP	
ESP+4	#1, arg_0
ESP+8	#2, arg_4
ESP+0xC	#3, arg_8
...	...

2.

```
printf("%d %d %d", 1234);
```

```
printf()
```

```
: main(),main(int argc, char *argv[]) main(int argc, char *argv[], char *envp[]).
```

```
push envp
push argv
push argc
```

```
call main
...
```

```
main(int argc, char *argv[]), main(int argc),.
```

5.2.3

5.2.4 x86:

3.

```
#ifdef __GNUC__
#include <alloca.h> // GCC
#else
#include <malloc.h> // MSVC
#endif
#include <stdio.h>

void f()
{
    char *buf=(char*)alloca (600);
#ifdef __GNUC__
    snprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // GCC
#else
    _snprintf (buf, 600, "hi! %d, %d, %d\n", 1, 2, 3); // MSVC
#endif

    puts (buf);
};
```

MSVC

(MSVC 2010):

Listing 5.1: MSVC 2010

```
...

mov     eax, 600          ; 00000258H
call    __alloca_probe_16
mov     esi, esp

push    3
push    2
push    1
push    OFFSET $SG2672
push    600              ; 00000258H
push    esi
call    __snprintf

push    esi
call    _puts
add     esp, 28          ; 0000001cH

...
```

³alloca16.asm chkstk.asm C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\src\intel

5.2.5 (Windows) SEH

.

5.2.6

([16.2 on page 62](#)).

5.2.7

5.3

...	...
ESP-0xC	#2, var_8
ESP-8	#1, var_4
ESP-4	EBP
ESP	
ESP+4	#1, arg_0
ESP+8	#2, arg_4
ESP+0xC	#3, arg_8
...	...

บทที่ 6

printf()

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d", 1, 2, 3);
    return 0;
};
```

6.1 x86

6.1.1 x86:

MSVC

```
$SG3830 DB      'a=%d; b=%d; c=%d', 00H

...

    push     3
    push     2
    push     1
    push     OFFSET $SG3830
    call     _printf
    add      esp, 16                      ; 00000010H
```

```
push a1
push a2
call ...
...
push a1
call ...
...
push a1
push a2
push a3
call ...
add esp, 24
```

Listing 6.1: x86

```
.text:100113E7      push     3
.text:100113E9      call     sub_100018B0 ; (3)
.text:100113EE      call     sub_100019D0 ;
.text:100113F3      call     sub_10006A90 ;
.text:100113F8      push     1
.text:100113FA      call     sub_100018B0 ; (1)
.text:100113FF      add      esp, 8      ;
```

6.1.2 x64:

:

```
#include <stdio.h>

int main()
{
    printf("a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d\n", 1, 2, 3, 4, 5, 6, 7, 8);
    return 0;
};
```

MSVC

RCX, RDX, R8, R9 . . .

Listing 6.2: MSVC 2012 x64

```
$SG2923 DB      'a=%d; b=%d; c=%d; d=%d; e=%d; f=%d; g=%d; h=%d', 0aH, 00H

main      PROC
          sub     rsp, 88

          mov     DWORD PTR [rsp+64], 8
          mov     DWORD PTR [rsp+56], 7
          mov     DWORD PTR [rsp+48], 6
          mov     DWORD PTR [rsp+40], 5
          mov     DWORD PTR [rsp+32], 4
          mov     r9d, 3
          mov     r8d, 2
          mov     edx, 1
          lea     rcx, OFFSET FLAT:$SG2923
          call    printf

          ; 0
          xor     eax, eax

          add     rsp, 88
          ret     0
main      ENDP
_TEXT     ENDS
END
```

6.2

:

Listing 6.3: x86

```
...  
PUSH  
PUSH  
PUSH  
CALL  
;  

```

Listing 6.4: x64 (MSVC)

```
MOV RCX,  
MOV RDX,  
MOV R8,  
MOV R9,  
...  
PUSH  
CALL  
;  

```

6.3

CPU .

บทที่ 7

scanf()

7.1

```
#include <stdio.h>

int main()
{
    int x;
    printf ("Enter X:\n");

    scanf ("%d", &x);

    printf ("You entered %d...\n", x);

    return 0;
};
```

7.1.1

; memcpy(),, void*,
(). scanf().

7.1.2 x86

MSVC

```
CONST    SEGMENT
$SG3831  DB      'Enter X:', 0aH, 00H
$SG3832  DB      '%d', 00H
$SG3833  DB      'You entered %d...', 0aH, 00H
CONST    ENDS
PUBLIC   _main
EXTRN    _scanf:PROC
EXTRN    _printf:PROC
; Function compile flags: /Odtp
_TEXT    SEGMENT
_x$ = -4                                ; size = 4
_main    PROC
    push  ebp
    mov   ebp, esp
```

```

push    ecx
push    OFFSET $SG3831 ; 'Enter X:'
call    _printf
add     esp, 4
lea     eax, DWORD PTR _x$[ebp]
push    eax
push    OFFSET $SG3832 ; '%d'
call    _scanf
add     esp, 8
mov     ecx, DWORD PTR _x$[ebp]
push    ecx
push    OFFSET $SG3833 ; 'You entered %d...'
call    _printf
add     esp, 8

; 0
xor     eax, eax
mov     esp, ebp
pop     ebp
ret     0
_main   ENDP
_TEXT   ENDS

```

:

...	...
EBP-8	#2, var_8
EBP-4	#1, var_4
EBP	EBP
EBP+4	
EBP+8	#1, arg_0
EBP+0xC	#2, arg_4
EBP+0x10	#3, arg_8
...	...

lea eax, [ebp-4].

You entered %d...\n.

7.1.3 x64

.

MSVC

Listing 7.1: MSVC 2012 x64

```

_DATA   SEGMENT
$SG1289 DB      'Enter X:', 0aH, 00H
$SG1291 DB      '%d', 00H
$SG1292 DB      'You entered %d...', 0aH, 00H
_DATA   ENDS

_TEXT   SEGMENT
x$ = 32
main    PROC

```

```

$LN3:
    sub     rsp, 56
    lea     rcx, OFFSET FLAT:$SG1289 ; 'Enter X:'
    call    printf
    lea     rdx, QWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG1291 ; '%d'
    call    scanf
    mov     edx, DWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG1292 ; 'You entered %d...'
    call    printf

    ; 0
    xor     eax, eax
    add     rsp, 56
    ret     0
main      ENDP
_TEXT     ENDS

```

7.2

```

#include <stdio.h>

//
int x;

int main()
{
    printf ("Enter X:\n");

    scanf ("%d", &x);

    printf ("You entered %d...\n", x);

    return 0;
};

```

7.2.1 MSVC: x86

```

_DATA      SEGMENT
COMM       _x:DWORD
$SG2456    DB      'Enter X:', 0aH, 00H
$SG2457    DB      '%d', 00H
$SG2458    DB      'You entered %d...', 0aH, 00H
_DATA      ENDS
PUBLIC     _main
EXTRN     _scanf:PROC
EXTRN     _printf:PROC
; Function compile flags: /Odtp
_TEXT      SEGMENT
_main      PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG2456
    call    _printf

```

```

add     esp, 4
push    OFFSET _x
push    OFFSET $SG2457
call    _scanf
add     esp, 8
mov     eax, DWORD PTR _x
push    eax
push    OFFSET $SG2458
call    _printf
add     esp, 8
xor     eax, eax
pop     ebp
ret     0
_main   ENDP
_TEXT   ENDS

```

```
int x=10; //
```

```

_DATA   SEGMENT
_x      DD      0aH

...

```

```

.data:0040FA80 _x          dd ?           ; DATA XREF: _main+10
.data:0040FA80           ; _main+22
.data:0040FA84 dword_40FA84 dd ?           ; DATA XREF: _memset+1E
.data:0040FA84           ; unknown_libname_1+28
.data:0040FA88 dword_40FA88 dd ?           ; DATA XREF: ___sbh_find_block+5
.data:0040FA88           ; ___sbh_free_block+2BC
.data:0040FA8C ; LPVOID lpMem
.data:0040FA8C lpMem      dd ?           ; DATA XREF: ___sbh_find_block+B
.data:0040FA8C           ; ___sbh_free_block+2CA
.data:0040FA90 dword_40FA90 dd ?           ; DATA XREF: _V6_HeapAlloc+13
.data:0040FA90           ; __calloc_impl+72
.data:0040FA94 dword_40FA94 dd ?           ; DATA XREF: ___sbh_free_block+2FE

```

7.2.2 MSVC: x64

Listing 7.2: MSVC 2012 x64

```

_DATA   SEGMENT
COMM    x:DWORD
$SG2924 DB      'Enter X:', 0aH, 00H
$SG2925 DB      '%d', 00H
$SG2926 DB      'You entered %d...', 0aH, 00H
_DATA   ENDS

_TEXT   SEGMENT
main    PROC
$LN3:
    sub     rsp, 40

    lea     rcx, OFFSET FLAT:$SG2924 ; 'Enter X:'
    call    printf
    lea     rdx, OFFSET FLAT:x
    lea     rcx, OFFSET FLAT:$SG2925 ; '%d'
    call    scanf

```



```

    mov     edx, DWORD PTR x
    lea     rcx, OFFSET FLAT:$SG2926 ; 'You entered %d...'
    call    printf

    ; 0
    xor     eax, eax

    add     rsp, 40
    ret     0
main      ENDP
_TEXT     ENDS

```

x86. . DWORD PTR.

7.3

```

#include <stdio.h>

int main()
{
    int x;
    printf ("Enter X:\n");

    if (scanf ("%d", &x)==1)
        printf ("You entered %d...\n", x);
    else
        printf ("What you entered? Huh?\n");

    return 0;
};

```

scanf()¹

:

```

C:\...\>ex3.exe
Enter X:
123
You entered 123...

C:\...\>ex3.exe
Enter X:
ouch
What you entered? Huh?

```

7.3.1 MSVC: x86

(MSVC 2010):

```

    lea     eax, DWORD PTR _x$[ebp]
    push    eax
    push    OFFSET $SG3833 ; '%d', 00H
    call    _scanf
    add     esp, 8
    cmp     eax, 1

```

¹scanf, wscanf: [MSDN](#)

```
    jne     SHORT $LN2@main
    mov     ecx, DWORD PTR _x$[ebp]
    push    ecx
    push    OFFSET $SG3834 ; 'You entered %d...', 0aH, 00H
    call    _printf
    add     esp, 8
    jmp     SHORT $LN1@main
$LN2@main:
    push    OFFSET $SG3836 ; 'What you entered? Huh?', 0aH, 00H
    call    _printf
    add     esp, 4
$LN1@main:
    xor     eax, eax
```

7.3.2 MSVC: x86 + Hiew

..

MSVCR*.DLL (/MD)², main() .text. .text (Enter, F8, F6, Enter, Enter).

:

```

Hiew: ex3.exe
C:\Polygon\ollydbg\ex3.exe  FRO ----- a32 PE .00401000 Hiew 8.02 (c)SEN
.00401000: 55          push     ebp
.00401001: 8BEC       mov     ebp,esp
.00401003: 51          push     ecx
.00401004: 6800304000 push     000403000 ;'Enter X:' --1
.00401009: FF1594204000 call    printf
.0040100F: 83C404     add     esp,4
.00401012: 8D45FC     lea     eax,[ebp][-4]
.00401015: 50          push     eax
.00401016: 680C304000 push     00040300C --2
.0040101B: FF158C204000 call    scanf
.00401021: 83C408     add     esp,8
.00401024: 83F801     cmp     eax,1
.00401027: 7514       jnz     .00040103D --3
.00401029: 8B4DFC     mov     ecx,[ebp][-4]
.0040102C: 51          push     ecx
.0040102D: 6810304000 push     000403010 ;'You entered %d...' --4
.00401032: FF1594204000 call    printf
.00401038: 83C408     add     esp,8
.0040103B: EB0E       jmps    .00040104B --5
.0040103D: 6824304000 3push   000403024 ;'What you entered? Huh?' --6
.00401042: FF1594204000 call    printf
.00401048: 83C404     add     esp,4
.0040104B: 33C0       xor     eax,eax
.0040104D: 8BE5       mov     esp,ebp
.0040104F: 5D          pop     ebp
.00401050: C3         retn    ; ^.^.^.^.^.^.^.^.^.^.^.^.^.^.^.^
.00401051: B84D5A0000 mov     eax,00005A4D ;' ZM'
1Global 2FileBlk 3CryBlk 4ReLoad 5OrdLdr 6String 7Direct 8Table 9byte 10Leave 11Naked 12AddName

```

รูปที่ 7.1: Hiew: main()

Hiew ASCIIZ³.²dynamic linking?³ASCII Zero ()

.00401027 (JNZ), F3, ?9090?(NOP⁴):

C:\Polygon\ollydbg\ex3.exe FWO EDITMODE a32 PE 00000429 Hiew 8.02 (c)SEN

00000400:	55	push	ebp
00000401:	8BEC	mov	ebp, esp
00000403:	51	push	ecx
00000404:	6800304000	push	000403000 ; ' @0 ' .
00000409:	FF1594204000	call	d, [000402094]
0000040F:	83C404	add	esp, 4
00000412:	8D45FC	lea	eax, [ebp] [-4]
00000415:	50	push	eax
00000416:	680C304000	push	00040300C ; ' @00 ' .
0000041B:	FF158C204000	call	d, [00040208C]
00000421:	83C408	add	esp, 8
00000424:	83F801	cmp	eax, 1
00000427:	90	nop	
00000428:	90	nop	
00000429:	8B4DFC	mov	ecx, [ebp] [-4]
0000042C:	51	push	ecx
0000042D:	6810304000	push	000403010 ; ' @00 ' .
00000432:	FF1594204000	call	d, [000402094]
00000438:	83C408	add	esp, 8
0000043B:	EB0E	jmps	00000044B
0000043D:	6824304000	push	000403024 ; ' @0\$ ' .
00000442:	FF1594204000	call	d, [000402094]
00000448:	83C404	add	esp, 4
0000044B:	33C0	xor	eax, eax
0000044D:	8BE5	mov	esp, ebp
0000044F:	5D	pop	ebp
00000450:	C3	retn	; _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

1 2NOPs 3 4 5 6 7 8Table 9 10 11 12

รูปที่ 7.2: Hiew: JNZ NOP

F9 (update).

NOP . JNZ .

7.3.3 MSVC: x64

Listing 7.3: MSVC 2012 x64

```

_DATA    SEGMENT
$SG2924 DB      'Enter X:', 0aH, 00H
$SG2926 DB      '%d', 00H
$SG2927 DB      'You entered %d...', 0aH, 00H
$SG2929 DB      'What you entered? Huh?', 0aH, 00H
_DATA    ENDS

_TEXT    SEGMENT
x$ = 32
main     PROC
$LN5:
        sub     rsp, 56
        lea     rcx, OFFSET FLAT:$SG2924 ; 'Enter X:'

```

⁴No OPeration

```
    call    printf
    lea     rdx, QWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG2926 ; '%d'
    call    scanf
    cmp     eax, 1
    jne     SHORT $LN2@main
    mov     edx, DWORD PTR x$[rsp]
    lea     rcx, OFFSET FLAT:$SG2927 ; 'You entered %d...'
    call    printf
    jmp     SHORT $LN1@main
$LN2@main:
    lea     rcx, OFFSET FLAT:$SG2929 ; 'What you entered? Huh?'
    call    printf
$LN1@main:
    ; 0
    xor     eax, eax
    add     rsp, 56
    ret     0
main      ENDP
_TEXT    ENDS
END
```

7.4

- <http://challenges.re/53>

บทที่ 8

Listing 8.1:

```
#include <stdio.h>

int f (int a, int b, int c)
{
    return a*b+c;
};

int main()
{
    printf ("%d\n", f(1, 2, 3));
    return 0;
};
```

8.1 x86

8.1.1 MSVC

(MSVC 2010 Express):

Listing 8.2: MSVC 2010 Express

```
_TEXT SEGMENT
_a$ = 8 ; size = 4
_b$ = 12 ; size = 4
_c$ = 16 ; size = 4
_f PROC
    push ebp
    mov ebp, esp
    mov eax, DWORD PTR _a$[ebp]
    imul eax, DWORD PTR _b$[ebp]
    add eax, DWORD PTR _c$[ebp]
    pop ebp
    ret 0
_f ENDP

_main PROC
    push ebp
    mov ebp, esp
    push 3 ;
    push 2 ;
    push 1 ;
    call _f
    add esp, 12
```

```

    push    eax
    push    OFFSET $SG2463 ; '%d', 0aH, 00H
    call    _printf
    add     esp, 8
    ; 0
    xor     eax, eax
    pop     ebp
    ret     0
_main     ENDP

```

8.2 x64

8.2.1 MSVC

MSVC:

Listing 8.3: MSVC 2012 x64

```

$SG2997 DB    '%d', 0aH, 00H

main     PROC
    sub     rsp, 40
    mov     edx, 2
    lea     r8d, QWORD PTR [rdx+1] ; R8D=3
    lea     ecx, QWORD PTR [rdx-1] ; ECX=1
    call    f
    lea     rcx, OFFSET FLAT:$SG2997 ; '%d'
    mov     edx, eax
    call    printf
    xor     eax, eax
    add     rsp, 40
    ret     0
main     ENDP

f        PROC
    ; ECX -
    ; EDX -
    ; R8D -
    imul    ecx, edx
    lea     eax, DWORD PTR [r8+rcx]
    ret     0
f        ENDP

```

:

Listing 8.4: MSVC 2012 x64

```

f        proc near

; shadow space:
arg_0    = dword ptr  8
arg_8    = dword ptr 10h
arg_10   = dword ptr 18h

    ; ECX -
    ; EDX -
    ; R8D -
    mov     [rsp+arg_10], r8d

```

```

    mov     [rsp+arg_8], edx
    mov     [rsp+arg_0], ecx
    mov     eax, [rsp+arg_0]
    imul    eax, [rsp+arg_8]
    add     eax, [rsp+arg_10]
    retn
f
endp

main      proc near
    sub     rsp, 28h
    mov     r8d, 3 ;
    mov     edx, 2 ;
    mov     ecx, 1 ;
    call    f
    mov     edx, eax
    lea     rcx, $SG2931 ; "%d\n"
    call    printf

    ; 0
    xor     eax, eax
    add     rsp, 28h
    retn
main      endp

```

?shadow space? ¹: : 1) 2) ².

¹MSDN

²MSDN

บทที่ 9

1

9.1

```
push envp
push argv
push argc
call main
push eax
call exit
```

```
exit(main(argc,argv,envp));
```

Thai text placeholder

. main() void:

```
#include <stdio.h>

void main()
{
    printf ("Hello, world!\n");
};
```

Linux.

GCC 4.8.1 printf() puts(), puts() printf(). EAX main().

Listing 9.1: GCC 4.8.1

```
.LC0:
    .string "Hello, world!"
main:
    push    ebp
    mov     ebp, esp
    and     esp, -16
    sub     esp, 16
    mov     DWORD PTR [esp], OFFSET FLAT:.LC0
    call    puts
    leave
    ret
```

:

¹Thai text placeholder: MSDN: Return Values (C++): [MSDN](#)

Listing 9.2: tst.sh

```
#!/bin/sh
./hello_world
echo $?
```

:

```
$ tst.sh
Hello, world!
14
```

14 .

9.2

```
int f()
{
    // skip first 3 random values
    rand();
    rand();
    rand();
    // and use 4th
    return rand();
};
```

บทที่ 10

[Dij68], [Knu74], [Yur13, p. 1.3.2].

:

```
#include <stdio.h>

int main()
{
    printf ("begin\n");
    goto exit;
    printf ("skip me!\n");
exit:
    printf ("end\n");
};
```

MSVC 2012:

Listing 10.1: MSVC 2012

```
$SG2934 DB      'begin', 0aH, 00H
$SG2936 DB      'skip me!', 0aH, 00H
$SG2937 DB      'end', 0aH, 00H

_main  PROC
    push    ebp
    mov     ebp, esp
    push    OFFSET $SG2934 ; 'begin'
    call    _printf
    add     esp, 4
    jmp     SHORT $exit$3
    push    OFFSET $SG2936 ; 'skip me!'
    call    _printf
    add     esp, 4
$exit$3:
    push    OFFSET $SG2937 ; 'end'
    call    _printf
    add     esp, 4
    xor     eax, eax
    pop     ebp
    ret     0
_main  ENDP
```

Thai text placeholder

10.1

Listing 10.2: MSVC 2012

```
$SG2981 DB      'begin', 0aH, 00H
$SG2983 DB      'skip me!', 0aH, 00H
$SG2984 DB      'end', 0aH, 00H

_main  PROC
        push     OFFSET $SG2981 ; 'begin'
        call     _printf
        push     OFFSET $SG2984 ; 'end'
$exit$4:
        call     _printf
        add      esp, 8
        xor      eax, eax
        ret      0
_main  ENDP
```

?skip me!? .

บทที่ 11

11.1

```
#include <stdio.h>

void f_signed (int a, int b)
{
    if (a>b)
        printf ("a>b\n");
    if (a==b)
        printf ("a==b\n");
    if (a<b)
        printf ("a<b\n");
};

void f_unsigned (unsigned int a, unsigned int b)
{
    if (a>b)
        printf ("a>b\n");
    if (a==b)
        printf ("a==b\n");
    if (a<b)
        printf ("a<b\n");
};

int main()
{
    f_signed(1, 2);
    f_unsigned(1, 2);
    return 0;
};
```

11.1.1 x86

x86 + MSVC

Listing 11.1: MSVC 2010

```
_a$ = 8
_b$ = 12
_f_signed PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    cmp     eax, DWORD PTR _b$[ebp]
```

```

jle     SHORT $LN3@f_signed
push    OFFSET $SG737      ; 'a>b'
call    _printf
add     esp, 4
$LN3@f_signed:
mov     ecx, DWORD PTR _a$[ebp]
cmp     ecx, DWORD PTR _b$[ebp]
jne     SHORT $LN2@f_signed
push    OFFSET $SG739      ; 'a==b'
call    _printf
add     esp, 4
$LN2@f_signed:
mov     edx, DWORD PTR _a$[ebp]
cmp     edx, DWORD PTR _b$[ebp]
jge     SHORT $LN4@f_signed
push    OFFSET $SG741      ; 'a<b'
call    _printf
add     esp, 4
$LN4@f_signed:
pop     ebp
ret     0
_f_signed ENDP

```

Jump if Less or Equal. JNE: Jump if Not Equal. .

JGE: Jump if Greater or Equal---

f_unsigned()

Listing 11.2: GCC

```

_a$ = 8   ; size = 4
_b$ = 12  ; size = 4
_f_unsigned PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    cmp     eax, DWORD PTR _b$[ebp]
    jbe     SHORT $LN3@f_unsigned
    push    OFFSET $SG2761    ; 'a>b'
    call    _printf
    add     esp, 4
$LN3@f_unsigned:
    mov     ecx, DWORD PTR _a$[ebp]
    cmp     ecx, DWORD PTR _b$[ebp]
    jne     SHORT $LN2@f_unsigned
    push    OFFSET $SG2763    ; 'a==b'
    call    _printf
    add     esp, 4
$LN2@f_unsigned:
    mov     edx, DWORD PTR _a$[ebp]
    cmp     edx, DWORD PTR _b$[ebp]
    jae     SHORT $LN4@f_unsigned
    push    OFFSET $SG2765    ; 'a<b'
    call    _printf
    add     esp, 4
$LN4@f_unsigned:
    pop     ebp
    ret     0
_f_unsigned ENDP

```

JBE---Jump if Below or Equal JAE---Jump if Above or Equal. (JA/JAE/JB/JBE) JG/JGE/JL/JLE

([22 on page 104](#)).

Listing 11.3: main()

```
_main  PROC
        push    ebp
        mov     ebp, esp
        push    2
        push    1
        call    _f_signed
        add     esp, 8
        push    2
        push    1
        call    _f_unsigned
        add     esp, 8
        xor     eax, eax
        pop     ebp
        ret     0
_main  ENDP
```

x86 + MSVC + Hiew

f_unsigned() ?a==b?, . Hiew:

```

Hiew: 7_1.exe
C:\Polygon\ollydbg\7_1.exe  FRO -----  a32 PE .00401000 Hiew 8.02 (c)SEN
.00401000: 55          push     ebp
.00401001: 8BEC       mov     ebp,esp
.00401003: 8B4508     mov     eax,[ebp+8]
.00401006: 3B450C     cmp     eax,[ebp+00C]
.00401009: 7E0D      jle     .00401018 --1
.0040100B: 6800B04000 push    00040B000 --2
.00401010: E8AA000000 call     .004010BF --3
.00401015: 83C404     add     esp,4
.00401018: 8B4D08     1mov    ecx,[ebp+8]
.0040101B: 3B4D0C     cmp     ecx,[ebp+00C]
.0040101E: 750D      jnz     .0040102D --4
.00401020: 6800B04000 push    00040B008 ; 'a==b' --5
.00401025: E895000000 call     .004010BF --3
.0040102A: 83C404     add     esp,4
.0040102D: 8B5508     4mov    edx,[ebp+8]
.00401030: 3B550C     cmp     edx,[ebp+00C]
.00401033: 7D0D      jge     .00401042 --6
.00401035: 6810B04000 push    00040B010 --7
.0040103A: E880000000 call     .004010BF --3
.0040103F: 83C404     add     esp,4
.00401042: 5D        6pop    ebp
.00401043: C3        retn    ; ^.^.^.^.^.^.^.^.^.^.^.^.^.^.^.^
.00401044: CC        int     3
.00401045: CC        int     3
.00401046: CC        int     3
.00401047: CC        int     3
.00401048: CC        int     3
1Global 2FilBlk 3CryBlk 4Reload 5OrdLdr 6String 7Direct 8Table 9byte 10Leave 11Naked 12AddNam

```

รูปที่ 11.1: Hiew: f_unsigned()

:

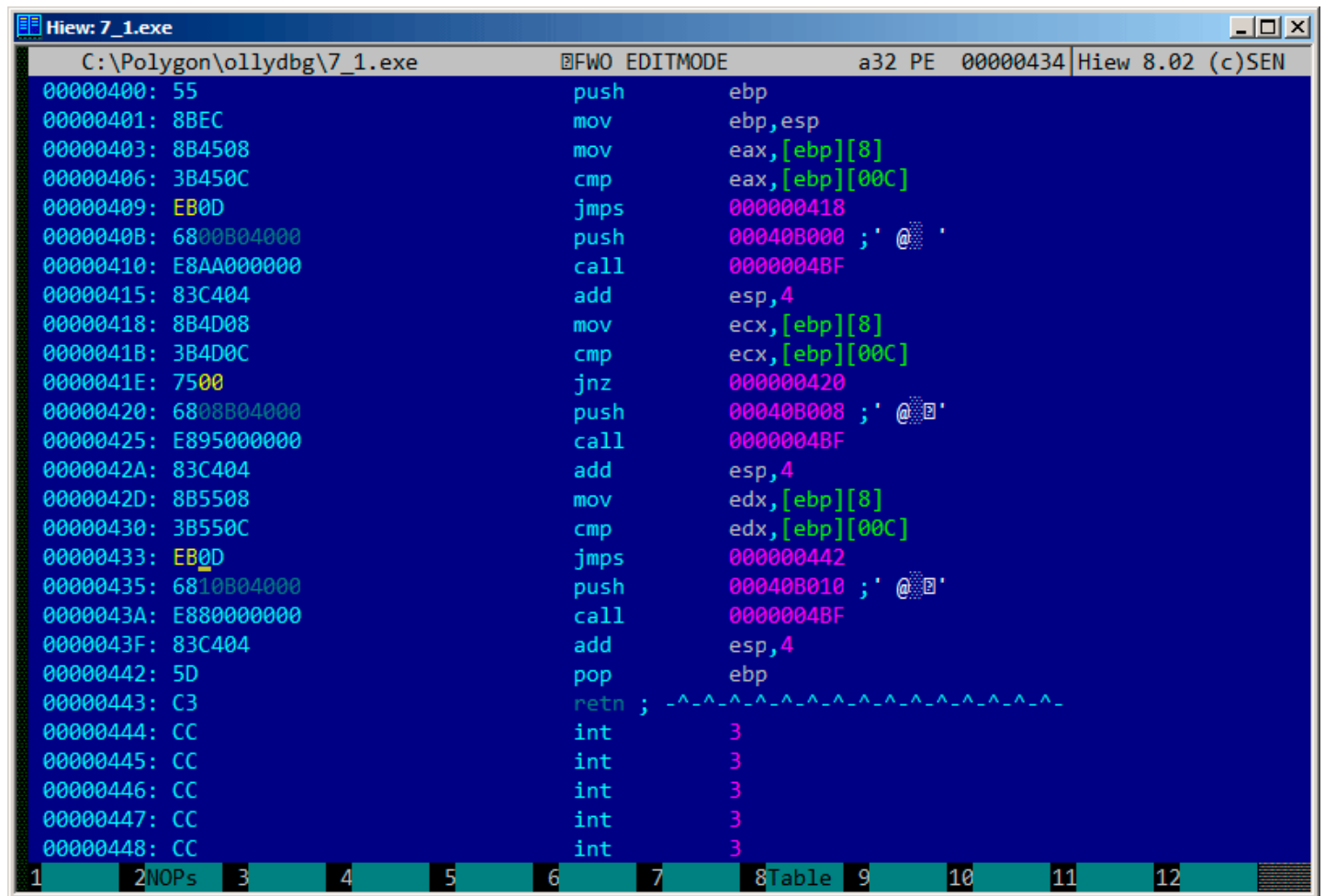
- ;
- ;
- .

printf(), ?a==b?.

:

- JMP, .
-
-

:



รูปที่ 11.2: Hiew: f_unsigned()

11.2

:

```
int my_abs (int i)
{
    if (i<0)
        return -i;
    else
        return i;
};
```

11.2.1 MSVC

Listing 11.4: MSVC 2012 x64

```
i$ = 8
my_abs PROC
; ECX = input
    test    ecx, ecx
;
;
```

```

        jns     SHORT $LN2@my_abs
;
        neg     ecx
$LN2@my_abs:
; EAX:
        mov     eax, ecx
        ret     0
my_abs   ENDP

```

11.3

:

```
expression ? expression : expression
```

:

```

const char* f (int a)
{
    return a==10 ? "it is ten" : "it is not ten";
};

```

11.3.1 x86

Listing 11.5: MSVC 2008

```

$SG746 DB      'it is ten', 00H
$SG747 DB      'it is not ten', 00H

tv65 = -4 ;
_a$ = 8
_f      PROC
        push    ebp
        mov     ebp, esp
        push    ecx
; 10
        cmp     DWORD PTR _a$[ebp], 10
; $LN3@f
        jne     SHORT $LN3@f
; :
        mov     DWORD PTR tv65[ebp], OFFSET $SG746 ; 'it is ten'
;
        jmp     SHORT $LN4@f
$LN3@f:
; :
        mov     DWORD PTR tv65[ebp], OFFSET $SG747 ; 'it is not ten'
$LN4@f:
;
        mov     eax, DWORD PTR tv65[ebp]
        mov     esp, ebp
        pop     ebp
        ret     0
_f      ENDP

```

Listing 11.6: MSVC 2008

```

$SG792 DB      'it is ten', 00H
$SG793 DB      'it is not ten', 00H

_a$ = 8 ; size = 4
_f      PROC
; 10
    cmp     DWORD PTR _a$[esp-4], 10
    mov     eax, OFFSET $SG792 ; 'it is ten'
; $LN4@f
    je      SHORT $LN4@f
    mov     eax, OFFSET $SG793 ; 'it is not ten'
$LN4@f:
    ret     0
_f      ENDP

```

:

Listing 11.7: MSVC 2012 x64

```

$SG1355 DB      'it is ten', 00H
$SG1356 DB      'it is not ten', 00H

a$ = 8
f      PROC
;
    lea     rdx, OFFSET FLAT:$SG1355 ; 'it is ten'
    lea     rax, OFFSET FLAT:$SG1356 ; 'it is not ten'
; 10
    cmp     ecx, 10
; ("it is ten")
;
    cmov     rax, rdx
    ret     0
f      ENDP

```

GCC 4.8 x86 CMOVcc.

11.3.2

```

const char* f (int a)
{
    if (a==10)
        return "it is ten";
    else
        return "it is not ten";
};

```

Listing 11.8: GCC 4.8

```

.LC0:
    .string "it is ten"
.LC1:
    .string "it is not ten"
f:
.LFB0:
; 10
    cmp     DWORD PTR [esp+4], 10

```

```

    mov     edx, OFFSET FLAT:.LC1 ; "it is not ten"
    mov     eax, OFFSET FLAT:.LC0 ; "it is ten"
;
;
    cmovne  eax, edx
    ret

```

MSVC 2012 .

11.4

11.4.1 32-bit

```

int my_max(int a, int b)
{
    if (a>b)
        return a;
    else
        return b;
};

int my_min(int a, int b)
{
    if (a<b)
        return a;
    else
        return b;
};

```

Listing 11.9: MSVC 2013

```

_a$ = 8
_b$ = 12
_my_min PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
; :
    cmp     eax, DWORD PTR _b$[ebp]
; :
    jge     SHORT $LN2@my_min
;
    mov     eax, DWORD PTR _a$[ebp]
    jmp     SHORT $LN3@my_min
    jmp     SHORT $LN3@my_min ; JMP
$LN2@my_min:
; B
    mov     eax, DWORD PTR _b$[ebp]
$LN3@my_min:
    pop     ebp
    ret     0
_my_min ENDP

_a$ = 8
_b$ = 12
_my_max PROC
    push    ebp

```

```

    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
; :
    cmp     eax, DWORD PTR _b$[ebp]
; :
    jle     SHORT $LN2@my_max
;
    mov     eax, DWORD PTR _a$[ebp]
    jmp     SHORT $LN3@my_max
    jmp     SHORT $LN3@my_max ; JMP
$LN2@my_max:
; B
    mov     eax, DWORD PTR _b$[ebp]
$LN3@my_max:
    pop     ebp
    ret     0
_my_max ENDP

```

11.5

11.5.1 x86

:

Listing 11.10: x86

```

CMP register, register/value
Jcc true ; cc=
false:
... ..
JMP exit
true:
... ..
exit:

```

11.5.2

บทที่ 12

switch()/case/default

12.1

```
#include <stdio.h>

void f (int a)
{
    switch (a)
    {
        case 0: printf ("zero\n"); break;
        case 1: printf ("one\n"); break;
        case 2: printf ("two\n"); break;
        default: printf ("something unknown\n"); break;
    };
};

int main()
{
    f (2); // test
};
```

12.1.1 x86

MSVC

(MSVC 2010):

Listing 12.1: MSVC 2010

```
tv64 = -4 ; size = 4
_a$ = 8 ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]
    mov     DWORD PTR tv64[ebp], eax
    cmp     DWORD PTR tv64[ebp], 0
    je      SHORT $LN4@f
    cmp     DWORD PTR tv64[ebp], 1
    je      SHORT $LN3@f
    cmp     DWORD PTR tv64[ebp], 2
```

```

je      SHORT $LN2@f
jmp     SHORT $LN1@f
$LN4@f:
push    OFFSET $SG739 ; 'zero', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN7@f
$LN3@f:
push    OFFSET $SG741 ; 'one', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN7@f
$LN2@f:
push    OFFSET $SG743 ; 'two', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN7@f
$LN1@f:
push    OFFSET $SG745 ; 'something unknown', 0aH, 00H
call    _printf
add     esp, 4
$LN7@f:
mov     esp, ebp
pop     ebp
ret     0
_f      ENDP

```

```

void f (int a)
{
    if (a==0)
        printf ("zero\n");
    else if (a==1)
        printf ("one\n");
    else if (a==2)
        printf ("two\n");
    else
        printf ("something unknown\n");
};

```

MSVC

MSVC (/Ox): cl 1.c /Fa1.asm /Ox

Listing 12.2: MSVC

```

_a$ = 8 ; size = 4
_f    PROC
    mov     eax, DWORD PTR _a$[esp-4]
    sub     eax, 0
    je      SHORT $LN4@f
    sub     eax, 1
    je      SHORT $LN3@f
    sub     eax, 1
    je      SHORT $LN2@f
    mov     DWORD PTR _a$[esp-4], OFFSET $SG791 ; 'something unknown', 0aH, 00H
    jmp     _printf
$LN2@f:
    mov     DWORD PTR _a$[esp-4], OFFSET $SG789 ; 'two', 0aH, 00H

```

```

    jmp     _printf
$LN3@f:
    mov     DWORD PTR _a$[esp-4], OFFSET $SG787 ; 'one', 0aH, 00H
    jmp     _printf
$LN4@f:
    mov     DWORD PTR _a$[esp-4], OFFSET $SG785 ; 'zero', 0aH, 00H
    jmp     _printf
_f        ENDP

```

- ESP--- [RA¹](#)
- ESP+4---

12.1.2

Thai text placeholder[12.1.1](#).

12.2

```

#include <stdio.h>

void f (int a)
{
    switch (a)
    {
        case 0: printf ("zero\n"); break;
        case 1: printf ("one\n"); break;
        case 2: printf ("two\n"); break;
        case 3: printf ("three\n"); break;
        case 4: printf ("four\n"); break;
        default: printf ("something unknown\n"); break;
    };
};

int main()
{
    f (2); // test
};

```

12.2.1 x86

MSVC

(MSVC 2010):

Listing 12.3: MSVC 2010

```

tv64 = -4 ; size = 4
_a$ = 8 ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]

```

¹Thai text placeholder


```

mov     DWORD PTR tv64[ebp], eax
cmp     DWORD PTR tv64[ebp], 4
ja      SHORT $LN1@f
mov     ecx, DWORD PTR tv64[ebp]
jmp     DWORD PTR $LN11@f[ecx*4]
$LN6@f:
push    OFFSET $SG739 ; 'zero', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN9@f
$LN5@f:
push    OFFSET $SG741 ; 'one', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN9@f
$LN4@f:
push    OFFSET $SG743 ; 'two', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN9@f
$LN3@f:
push    OFFSET $SG745 ; 'three', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN9@f
$LN2@f:
push    OFFSET $SG747 ; 'four', 0aH, 00H
call    _printf
add     esp, 4
jmp     SHORT $LN9@f
$LN1@f:
push    OFFSET $SG749 ; 'something unknown', 0aH, 00H
call    _printf
add     esp, 4
$LN9@f:
mov     esp, ebp
pop     ebp
ret     0
npad    2 ;
$LN11@f:
DD      $LN6@f ; 0
DD      $LN5@f ; 1
DD      $LN4@f ; 2
DD      $LN3@f ; 3
DD      $LN2@f ; 4
_f      ENDP

```

*jumptable branch table*².

GCC

:

Listing 12.4: GCC 4.4.1

```

public f
f      proc near ; CODE XREF: main+10

```

²computed GOTO : [wikipedia](https://en.cppreference.com/w/cpp/string/basic/basic_string_view). !

```

var_18 = dword ptr -18h
arg_0  = dword ptr  8

        push    ebp
        mov     ebp, esp
        sub     esp, 18h
        cmp     [ebp+arg_0], 4
        ja      short loc_8048444
        mov     eax, [ebp+arg_0]
        shl     eax, 2
        mov     eax, ds:off_804855C[eax]
        jmp     eax

loc_80483FE: ; DATA XREF: .rodata:off_804855C
        mov     [esp+18h+var_18], offset aZero ; "zero"
        call    _puts
        jmp     short locret_8048450

loc_804840C: ; DATA XREF: .rodata:08048560
        mov     [esp+18h+var_18], offset aOne ; "one"
        call    _puts
        jmp     short locret_8048450

loc_804841A: ; DATA XREF: .rodata:08048564
        mov     [esp+18h+var_18], offset aTwo ; "two"
        call    _puts
        jmp     short locret_8048450

loc_8048428: ; DATA XREF: .rodata:08048568
        mov     [esp+18h+var_18], offset aThree ; "three"
        call    _puts
        jmp     short locret_8048450

loc_8048436: ; DATA XREF: .rodata:0804856C
        mov     [esp+18h+var_18], offset aFour ; "four"
        call    _puts
        jmp     short locret_8048450

loc_8048444: ; CODE XREF: f+A
        mov     [esp+18h+var_18], offset aSomethingUnkno ; "something unknown"
        call    _puts

locret_8048450: ; CODE XREF: f+26
                ; f+34...
        leave
        retn
f            endp

off_804855C dd offset loc_80483FE ; DATA XREF: f+12
            dd offset loc_804840C
            dd offset loc_804841A
            dd offset loc_8048428
            dd offset loc_8048436

```

12.2.2*switch()*:

Listing 12.5: x86

```

MOV REG, input
CMP REG, 4 ;
JA default
SHL REG, 2 ;
MOV REG, jump_table[REG]
JMP REG

case1:
;
JMP exit
case2:
;
JMP exit
case3:
;
JMP exit
case4:
;
JMP exit
case5:
;
JMP exit

default:
...

exit:
....

jump_table dd case1
           dd case2
           dd case3
           dd case4
           dd case5

```

JMP jump_table[REG*4]. JMP jump_table[REG*8] x64.

[16.4 on page 66.](#)

12.3

```

#include <stdio.h>

void f(int a)
{
    switch (a)
    {
        case 1:
        case 2:
        case 7:

```

```

    case 10:
        printf ("1, 2, 7, 10\n");
        break;
    case 3:
    case 4:
    case 5:
    case 6:
        printf ("3, 4, 5\n");
        break;
    case 8:
    case 9:
    case 20:
    case 21:
        printf ("8, 9, 21\n");
        break;
    case 22:
        printf ("22\n");
        break;
    default:
        printf ("default\n");
        break;
};

int main()
{
    f(4);
};

```

12.3.1 MSVC

Listing 12.6: MSVC 2010

```

1  $SG2798 DB      '1, 2, 7, 10', 0aH, 00H
2  $SG2800 DB      '3, 4, 5', 0aH, 00H
3  $SG2802 DB      '8, 9, 21', 0aH, 00H
4  $SG2804 DB      '22', 0aH, 00H
5  $SG2806 DB      'default', 0aH, 00H
6
7  _a$ = 8
8  _f      PROC
9          mov     eax, DWORD PTR _a$[esp-4]
10         dec     eax
11         cmp     eax, 21
12         ja      SHORT $LN1@f
13         movzx   eax, BYTE PTR $LN10@f[eax]
14         jmp     DWORD PTR $LN11@f[eax*4]
15 $LN5@f:
16         mov     DWORD PTR _a$[esp-4], OFFSET $SG2798 ; '1, 2, 7, 10'
17         jmp     DWORD PTR __imp__printf
18 $LN4@f:
19         mov     DWORD PTR _a$[esp-4], OFFSET $SG2800 ; '3, 4, 5'
20         jmp     DWORD PTR __imp__printf
21 $LN3@f:
22         mov     DWORD PTR _a$[esp-4], OFFSET $SG2802 ; '8, 9, 21'
23         jmp     DWORD PTR __imp__printf
24 $LN2@f:

```

```

25      mov     DWORD PTR _a$[esp-4], OFFSET $SG2804 ; '22'
26      jmp     DWORD PTR __imp__printf
27 $LN1@f:
28      mov     DWORD PTR _a$[esp-4], OFFSET $SG2806 ; 'default'
29      jmp     DWORD PTR __imp__printf
30      npad    2 ; $LN11@f
31 $LN11@f:
32      DD      $LN5@f ; '1, 2, 7, 10'
33      DD      $LN4@f ; '3, 4, 5'
34      DD      $LN3@f ; '8, 9, 21'
35      DD      $LN2@f ; '22'
36      DD      $LN1@f ; 'default'
37 $LN10@f:
38      DB      0 ; a=1
39      DB      0 ; a=2
40      DB      1 ; a=3
41      DB      1 ; a=4
42      DB      1 ; a=5
43      DB      1 ; a=6
44      DB      0 ; a=7
45      DB      2 ; a=8
46      DB      2 ; a=9
47      DB      0 ; a=10
48      DB      4 ; a=11
49      DB      4 ; a=12
50      DB      4 ; a=13
51      DB      4 ; a=14
52      DB      4 ; a=15
53      DB      4 ; a=16
54      DB      4 ; a=17
55      DB      4 ; a=18
56      DB      4 ; a=19
57      DB      2 ; a=20
58      DB      2 ; a=21
59      DB      3 ; a=22
60 _f      ENDP

```

: (\$LN10@f), (\$LN11@f) .

(13).

: 0 (1, 2, 7, 10), 1 (3, 4, 5), 2 (8, 9, 21), 3 (22), 4 .

(14).

.

? ([12.2.1 on page 44](#)), ? .

12.4 Fall-through

switch() .:

```

1 #define R 1
2 #define W 2
3 #define RW 3
4
5 void f(int type)
6 {
7     int read=0, write=0;

```

```

8
9     switch (type)
10    {
11        case RW:
12            read=1;
13        case W:
14            write=1;
15            break;
16        case R:
17            read=1;
18            break;
19        default:
20            break;
21    };
22    printf ("read=%d, write=%d\n", read, write);
23 };

```

type = 1 (R), *read* 1, *type* = 2 (W), *write* 2. *type* = 3 (RW), *read* *write* 1.

type = RW *type* = W. .

12.4.1 MSVC x86

Listing 12.7: MSVC 2012

```

$SG1305 DB      'read=%d, write=%d', 0aH, 00H

_write$ = -12   ; size = 4
_read$ = -8     ; size = 4
tv64 = -4       ; size = 4
_type$ = 8      ; size = 4
_f        PROC
    push     ebp
    mov     ebp, esp
    sub     esp, 12
    mov     DWORD PTR _read$[ebp], 0
    mov     DWORD PTR _write$[ebp], 0
    mov     eax, DWORD PTR _type$[ebp]
    mov     DWORD PTR tv64[ebp], eax
    cmp     DWORD PTR tv64[ebp], 1 ; R
    je      SHORT $LN2@f
    cmp     DWORD PTR tv64[ebp], 2 ; W
    je      SHORT $LN3@f
    cmp     DWORD PTR tv64[ebp], 3 ; RW
    je      SHORT $LN4@f
    jmp     SHORT $LN5@f
$LN4@f: ; case RW:
    mov     DWORD PTR _read$[ebp], 1
$LN3@f: ; case W:
    mov     DWORD PTR _write$[ebp], 1
    jmp     SHORT $LN5@f
$LN2@f: ; case R:
    mov     DWORD PTR _read$[ebp], 1
$LN5@f: ; default
    mov     ecx, DWORD PTR _write$[ebp]
    push     ecx
    mov     edx, DWORD PTR _read$[ebp]
    push     edx

```

```
    push    OFFSET $SG1305 ; 'read=%d, write=%d'
    call    _printf
    add     esp, 12
    mov     esp, ebp
    pop     ebp
    ret     0
_f        ENDP
```

\$LN4@f \$LN3@f: \$LN4@f, *read write.* *type = W*, \$LN3@f, .

บทที่ 13

13.1

13.1.1 x86

Thai text placeholder

for().

```
{
    ;
}

:
#include <stdio.h>

void printing_function(int i)
{
    printf ("f(%d)\n", i);
};

int main()
{
    int i;

    for (i=2; i<10; i++)
        printing_function(i);

    return 0;
};
```

(MSVC 2010):

Listing 13.1: MSVC 2010

```
_i$ = -4
_main PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     DWORD PTR _i$[ebp], 2    ;
    jmp     SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp] ; :
```



```

    add    eax, 1          ;
    mov    DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp    DWORD PTR _i$[ebp], 10 ;
    jge    SHORT $LN1@main ;
    mov    ecx, DWORD PTR _i$[ebp] ; printing_function(i)
    push   ecx
    call   _printing_function
    add    esp, 4
    jmp    SHORT $LN2@main ;
$LN1@main:
    xor    eax, eax
    mov    esp, ebp
    pop    ebp
    ret    0
_main    ENDP

```

Listing 13.2: MSVC

```

_main    PROC
    push   esi
    mov    esi, 2
$LL3@main:
    push   esi
    call   _printing_function
    inc    esi
    add    esp, 4
    cmp    esi, 10 ; 0000000aH
    jl     SHORT $LL3@main
    xor    eax, eax
    pop    esi
    ret    0
_main    ENDP

```

13.1.2

: *i*

```

for (i=0; i<total_entries_to_process; i++)
    ;

```

total_entries_to_process 0, .

13.2

```

#include <stdio.h>

void my_memcpy (unsigned char* dst, unsigned char* src, size_t cnt)
{
    size_t i;
    for (i=0; i<cnt; i++)
        dst[i]=src[i];
};

```

13.2.1

Listing 13.3: GCC 4.9 x64 (-Os)

```

my_memcpy:
; RDI =
; RSI =
; RDX =

;
    xor     eax, eax
.L2:
;
    cmp     rax, rdx
    je      .L5
; RSI+i:
    mov     cl, BYTE PTR [rsi+rax]
; RDI+i:
    mov     BYTE PTR [rdi+rax], cl
    inc     rax ; i++
    jmp     .L2
.L5:
    ret

```

13.3

:

Listing 13.4: x86

```

    mov [counter], 2 ;
    jmp check
body:
;
;
;
    add [counter], 1 ;
check:
    cmp [counter], 9
    jle body

```

Listing 13.5: x86

```

    MOV [counter], 2 ;
    JMP check
body:
;
;
;
    MOV REG, [counter] ;
    INC REG
    MOV [counter], REG
check:
    CMP [counter], 9
    JLE body

```

Listing 13.6: x86

```

    MOV EBX, 2 ;
    JMP check
body:
    ;
    ;
    ;
    INC EBX ;
check:
    CMP EBX, 9
    JLE body

```

Listing 13.7: x86

```

    MOV [counter], 2 ;
    JMP label_check
label_increment:
    ADD [counter], 1 ;
label_check:
    CMP [counter], 10
    JGE exit
    ;
    ;
    ;
    JMP label_increment
exit:

```

Listing 13.8: x86

```

    MOV REG, 2 ;
body:
    ;
    ;
    ;
    ;
    INC REG ;
    CMP REG, 10
    JL body

```

Listing 13.9: x86

```

    ;
    MOV ECX, 10
body:
    ;
    ;
    ;
    ;
    LOOP body

```

บทที่ 14

14.1 strlen()

```
int my_strlen (const char * str)
{
    const char *eos = str;

    while( *eos++ ) ;

    return( eos - str - 1 );
}

int main()
{
    // test
    return my_strlen("hello!");
};
```

14.1.1 x86

MSVC

```
_eos$ = -4 ; size = 4
_str$ = 8 ; size = 4
_strlen PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _str$[ebp] ; "str"
    mov     DWORD PTR _eos$[ebp], eax ; "eos"
$LN2@strlen_:
    mov     ecx, DWORD PTR _eos$[ebp] ; ECX=eos

    ;

    movsx   edx, BYTE PTR [ecx]
    mov     eax, DWORD PTR _eos$[ebp] ; EAX=eos
    add     eax, 1 ; EAX
    mov     DWORD PTR _eos$[ebp], eax ; "eos"
    test    edx, edx ; EDX ?
    je      SHORT $LN1@strlen_ ;
    jmp     SHORT $LN2@strlen_ ;
$LN1@strlen_:
```

```

;

mov     eax, DWORD PTR _eos$[ebp]
sub     eax, DWORD PTR _str$[ebp]
sub     eax, 1                ;
mov     esp, ebp
pop     ebp
ret     0
_strlen_ ENDP

```

?? (22 on page 104).

MSVC

:

Listing 14.1: MSVC 2012 /Ob0

```

_str$ = 8                ; size = 4
_strlen PROC
    mov     edx, DWORD PTR _str$[esp-4] ; EDX ->
    mov     eax, edx                ; EAX
$LL2@strlen:
    mov     cl, BYTE PTR [eax]      ; CL = *EAX
    inc     eax                    ; EAX++
    test    cl, cl                  ; CL==0?
    jne     SHORT $LL2@strlen      ;
    sub     eax, edx                ;
    dec     eax                    ; EAX
    ret     0
_strlen ENDP

```

INC/DEC---

บทที่ 15

, ADD SUB: 18 Thai text placeholder??.

15.1

15.1.1

:

Listing 15.1: MSVC 2010

```
unsigned int f(unsigned int a)
{
    return a*8;
};
```

```
_TEXT SEGMENT
_a$ = 8 ; size = 4
_f PROC
; File c:\polygon\c\2.c
    mov     eax, DWORD PTR _a$[esp-4]
    add     eax, eax
    add     eax, eax
    add     eax, eax
    ret     0
_f ENDP
_TEXT ENDS
END
```

15.1.2

```
unsigned int f(unsigned int a)
{
    return a*4;
};
```

Listing 15.2: MSVC 2010

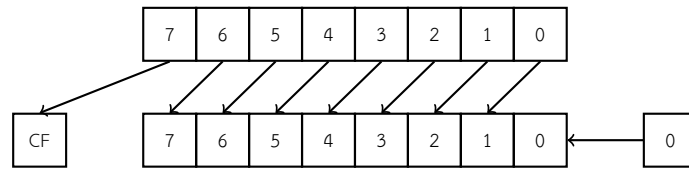
```
_a$ = 8 ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _a$[ebp]
    shl     eax, 2
    pop     ebp
```

```

ret    0
_f     ENDP

```

:



15.1.3

32-

```

#include <stdint.h>

int f1(int a)
{
    return a*7;
};

int f2(int a)
{
    return a*28;
};

int f3(int a)
{
    return a*17;
};

```

x86

Listing 15.3: MSVC 2012

```

; a*7
_a$ = 8
_f1   PROC
    mov     ecx, DWORD PTR _a$[esp-4]
; ECX=a
    lea     eax, DWORD PTR [ecx*8]
; EAX=ECX*8
    sub     eax, ecx
; EAX=EAX-ECX=ECX*8-ECX=ECX*7=a*7
    ret     0
_f1   ENDP

; a*28
_a$ = 8
_f2   PROC
    mov     ecx, DWORD PTR _a$[esp-4]
; ECX=a

```

```

        lea     eax, DWORD PTR [ecx*8]
; EAX=ECX*8
        sub     eax, ecx
; EAX=EAX-ECX=ECX*8-ECX=ECX*7=a*7
        shl     eax, 2
; EAX=EAX<<2=(a*7)*4=a*28
        ret     0
_f2     ENDP

; a*17
_a$ = 8
_f3     PROC
        mov     eax, DWORD PTR _a$[esp-4]
; EAX=a
        shl     eax, 4
; EAX=EAX<<4=EAX*16=a*16
        add     eax, DWORD PTR _a$[esp-4]
; EAX=EAX+a=a*16+a=a*17
        ret     0
_f3     ENDP

```

64-

```

#include <stdint.h>

int64_t f1(int64_t a)
{
    return a*7;
};

int64_t f2(int64_t a)
{
    return a*28;
};

int64_t f3(int64_t a)
{
    return a*17;
};

```

x64

Listing 15.4: MSVC 2012

```

; a*7
f1:
        lea     rax, [0+rdi*8]
; RAX=RDI*8=a*8
        sub     rax, rdi
; RAX=RAX-RDI=a*8-a=a*7
        ret

; a*28
f2:
        lea     rax, [0+rdi*4]
; RAX=RDI*4=a*4

```



```

        sal     rdi, 5
; RDI=RDI<<5=RDI*32=a*32
        sub     rdi, rax
; RDI=RDI-RAX=a*32-a*4=a*28
        mov     rax, rdi
        ret

; a*17
f3:
        mov     rax, rdi
        sal     rax, 4
; RAX=RAX<<4=a*16
        add     rax, rdi
; RAX=a*16+a=a*17
        ret

```

15.2

15.2.1

:

```

unsigned int f(unsigned int a)
{
    return a/4;
};

```

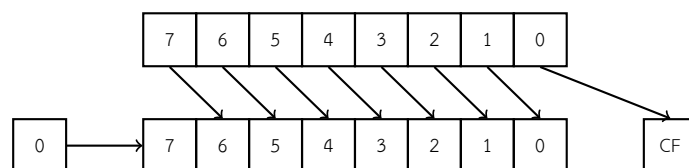
(MSVC 2010):

Listing 15.5: MSVC 2010

```

_a$ = 8                                     ; size = 4
_f     PROC
    mov     eax, DWORD PTR _a$[esp-4]
    shr     eax, 2
    ret     0
_f     ENDP

```



Thai text placeholder

บทที่ 16

16.1

```
#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<20; i++)
        a[i]=i*2;

    for (i=0; i<20; i++)
        printf ("a[%d]=%d\n", i, a[i]);

    return 0;
};
```

16.1.1 x86

MSVC

:

Listing 16.1: MSVC 2008

```
_TEXT    SEGMENT
_i$ = -84                ; size = 4
_a$ = -80                ; size = 80
_main    PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 84        ; 00000054H
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN6@main
$LN5@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN6@main:
    cmp     DWORD PTR _i$[ebp], 20    ; 00000014H
    jge     SHORT $LN4@main
    mov     ecx, DWORD PTR _i$[ebp]
    shl     ecx, 1
```

```

    mov     edx, DWORD PTR _i$[ebp]
    mov     DWORD PTR _a$[ebp+edx*4], ecx
    jmp     SHORT $LN5@main
$LN4@main:
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN3@main
$LN2@main:
    mov     eax, DWORD PTR _i$[ebp]
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN3@main:
    cmp     DWORD PTR _i$[ebp], 20      ; 00000014H
    jge     SHORT $LN1@main
    mov     ecx, DWORD PTR _i$[ebp]
    mov     edx, DWORD PTR _a$[ebp+ecx*4]
    push    edx
    mov     eax, DWORD PTR _i$[ebp]
    push    eax
    push    OFFSET $SG2463
    call    _printf
    add     esp, 12          ; 0000000cH
    jmp     SHORT $LN2@main
$LN1@main:
    xor     eax, eax
    mov     esp, ebp
    pop     ebp
    ret     0
_main     ENDP

```

16.2

16.2.1

```

#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<20; i++)
        a[i]=i*2;

    printf ("a[20]=%d\n", a[20]);

    return 0;
};

```

(MSVC 2008):

Listing 16.2: MSVC 2008

```

$SG2474 DB      'a[20]=%d', 0aH, 00H

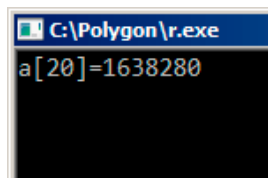
_i$ = -84 ; size = 4
_a$ = -80 ; size = 80
_main  PROC

```

```

push    ebp
mov     ebp, esp
sub     esp, 84
mov     DWORD PTR _i$[ebp], 0
jmp     SHORT $LN3@main
$LN2@main:
mov     eax, DWORD PTR _i$[ebp]
add     eax, 1
mov     DWORD PTR _i$[ebp], eax
$LN3@main:
cmp     DWORD PTR _i$[ebp], 20
jge     SHORT $LN1@main
mov     ecx, DWORD PTR _i$[ebp]
shl     ecx, 1
mov     edx, DWORD PTR _i$[ebp]
mov     DWORD PTR _a$[ebp+edx*4], ecx
jmp     SHORT $LN2@main
$LN1@main:
mov     eax, DWORD PTR _a$[ebp+80]
push    eax
push    OFFSET $SG2474 ; 'a[20]=%d'
call    DWORD PTR __imp__printf
add     esp, 8
xor     eax, eax
mov     esp, ebp
pop     ebp
ret     0
_main   ENDP
_TEXT   ENDS
END

```



รูปที่ 16.1: OllyDbg:

16.2.2

```

#include <stdio.h>

int main()
{
    int a[20];
    int i;

    for (i=0; i<30; i++)
        a[i]=i;

    return 0;
};

```

Listing 16.3: MSVC 2008

```
_TEXT    SEGMENT
_i$ = -84 ; size = 4
_a$ = -80 ; size = 80
_main    PROC
push     ebp
mov      ebp, esp
sub      esp, 84
mov      DWORD PTR _i$[ebp], 0
jmp      SHORT $LN3@main
$LN2@main:
mov      eax, DWORD PTR _i$[ebp]
add      eax, 1
mov      DWORD PTR _i$[ebp], eax
$LN3@main:
cmp      DWORD PTR _i$[ebp], 30 ; 0000001eH
jge      SHORT $LN1@main
mov      ecx, DWORD PTR _i$[ebp]
mov      edx, DWORD PTR _i$[ebp] ;
mov      DWORD PTR _a$[ebp+ecx*4], edx ;
jmp      SHORT $LN2@main
$LN1@main:
xor      eax, eax
mov      esp, ebp
pop      ebp
ret      0
_main    ENDP
```

CPU - main thread, module w

Address	Hex dump	ASCII (ANSI)
00401000	55	
00401001	8BEC	
00401003	83EC 54	
00401006	C745 AC 0000	
0040100D	EB 09	
0040100F	8B45 AC	
00401012	83C0 01	
00401015	8945 AC	
00401018	837D AC 1E	
0040101C	7D 0C	
0040101E	8B4D AC	
00401021	8B55 AC	
00401024	89548D B0	
00401028	EB E5	
0040102A	33C0	
0040102C	8BE5	
0040102E	5D	
0040102F	C3	
00401030	68 14144000	
00401035	E8 90030000	
0040103A	A1 40304000	
0040103F	C70424 2C3040	
00401046	FF35 3C304000	

Top of stack [0018FF48]=00000015

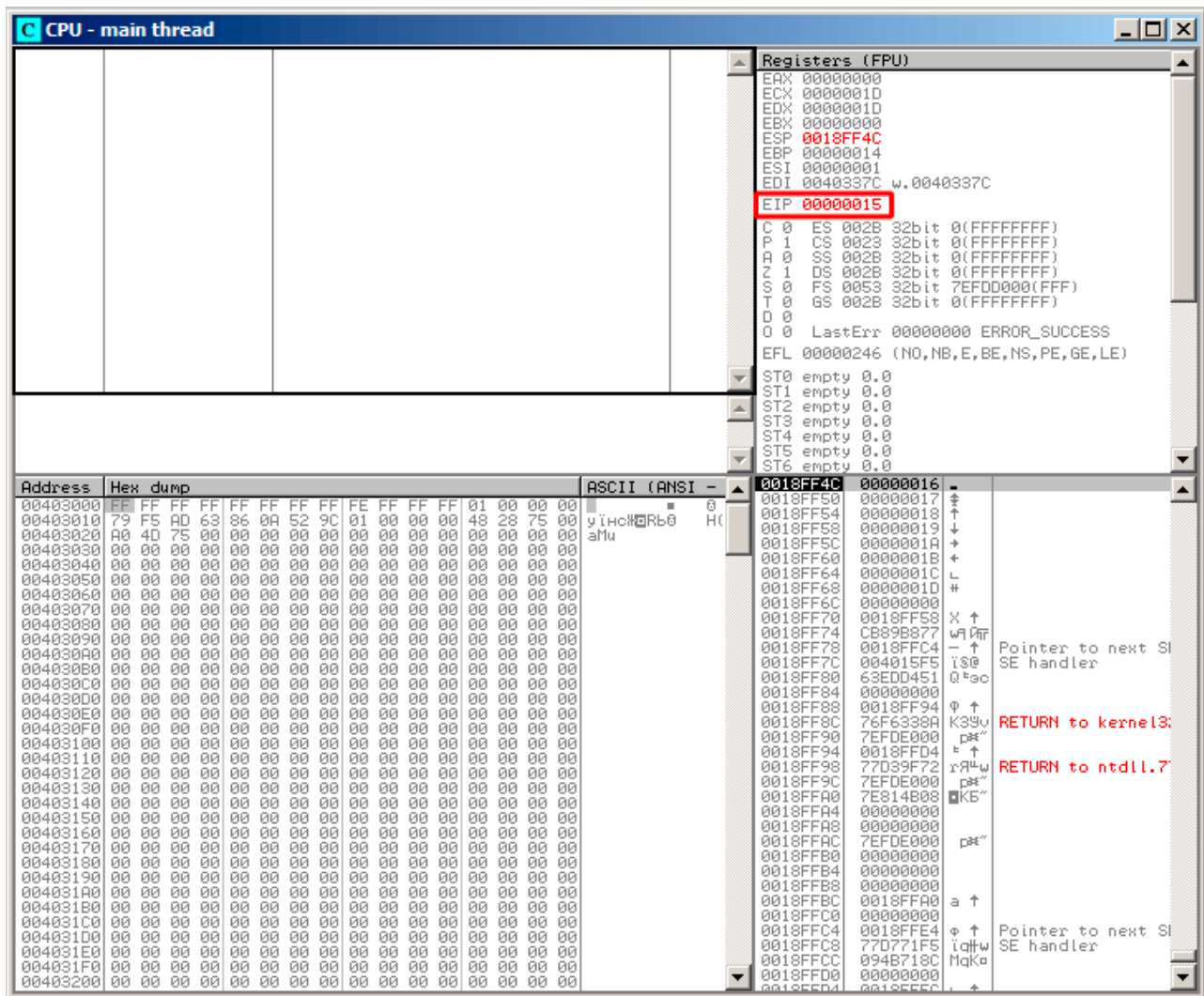
Registers (FPU)

Register	Value
EAX	00000000
ECX	0000001D
EDX	0000001D
EBX	00000000
ESP	0018FF48
EBP	00000014
ESI	00000001
EDI	0040337C w.0040337C
EIP	0040102F w.0040102F
C 0	ES 002B 32bit 0(FFFFFFFF)
P 1	CS 0023 32bit 0(FFFFFFFF)
A 0	SS 002B 32bit 0(FFFFFFFF)
Z 1	DS 002B 32bit 0(FFFFFFFF)
S 0	FS 0053 32bit 7EFD0000(FFF)
T 0	GS 002B 32bit 0(FFFFFFFF)
D 0	
O 0	LastErr 00000000 ERROR_SUCCESS
EFL	00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty 0.0
ST1	empty 0.0
ST2	empty 0.0
ST3	empty 0.0
ST4	empty 0.0
ST5	empty 0.0
ST6	empty 0.0

Memory Dump

Address	Hex dump	ASCII (ANSI)
00403000	FF FF FF FF FF FF FF FE FF FF FF 01 00 00 00	
00403010	79 F5 AD 63 86 0A 52 9C 01 00 00 00 48 28 75 00	yincRb0
00403020	A0 4D 75 00 00 00 00 00 00 00 00 00 00 00 00 00	afu
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004030F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403120	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403140	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403170	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
004031F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

รูปที่ 16.2: OllyDbg:

$$\vdots$$


รูปที่ 16.3: OllyDbg:

```
main():
```

ESP	
ESP+4	
ESP+84	
ESP+88	

buffer overflow¹.

16.3

```
void f(int size)
{
    int a[size];
    ...
};
```

16.4

¹wikipedia

Listing 16.4:

```

#include <stdio.h>

const char* month1[]=
{
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
};

//
const char* get_month1 (int month)
{
    return month1[month];
};

```

16.4.1 x64

Listing 16.5: MSVC 2013 x64

_DATA	SEGMENT	
month1	DQ	FLAT:\$SG3122
	DQ	FLAT:\$SG3123
	DQ	FLAT:\$SG3124
	DQ	FLAT:\$SG3125
	DQ	FLAT:\$SG3126
	DQ	FLAT:\$SG3127
	DQ	FLAT:\$SG3128
	DQ	FLAT:\$SG3129
	DQ	FLAT:\$SG3130
	DQ	FLAT:\$SG3131
	DQ	FLAT:\$SG3132
	DQ	FLAT:\$SG3133
\$SG3122	DB	'January', 00H
\$SG3123	DB	'February', 00H
\$SG3124	DB	'March', 00H
\$SG3125	DB	'April', 00H
\$SG3126	DB	'May', 00H
\$SG3127	DB	'June', 00H
\$SG3128	DB	'July', 00H
\$SG3129	DB	'August', 00H
\$SG3130	DB	'September', 00H
\$SG3156	DB	'%s', 0aH, 00H
\$SG3131	DB	'October', 00H
\$SG3132	DB	'November', 00H
\$SG3133	DB	'December', 00H
_DATA	ENDS	


```

month$ = 8
get_month1 PROC
    movsxd    rax, ecx
    lea       rcx, OFFSET FLAT:month1
    mov       rax, QWORD PTR [rcx+rax*8]
    ret       0
get_month1 ENDP

```

:
 • 2.
 •
 • .

GCC 4.9 ³:

Listing 16.6: GCC 4.9 x64

```

movsx    rdi, edi
mov      rax, QWORD PTR month1[0+rdi*8]
ret

```

32-bit MSVC

Listing 16.7: MSVC 2013 x86

```

_get_month$ = 8
_get_month1 PROC
    mov     eax, DWORD PTR _month$[esp-4]
    mov     eax, DWORD PTR _month1[eax*4]
    ret     0
_get_month1 ENDP

```

16.5

0	[0][0]
1	[0][1]
2	[0][2]
3	[0][3]
4	[1][0]
5	[1][1]
6	[1][2]
7	[1][3]
8	[2][0]
9	[2][1]
10	[2][2]
11	[2][3]

ตารางที่ 16.1:

0	1	2	3
4	5	6	7
8	9	10	11

ตารางที่ 16.2:

row-major order, Python. row-major order
column-major order () FORTRAN, MATLAB R. column-major order
?

16.5.1

0..3:

Listing 16.8:

```
#include <stdio.h>

char a[3][4];

int main()
{
    int x, y;

    //
    for (x=0; x<3; x++)
        for (y=0; y<4; y++)
            a[x][y]=0;

    // 0..3:
    for (y=0; y<4; y++)
        a[1][y]=y;
};
```

. 0, 1, 2 3:

Address	Hex dump
00C33370	00 00 00 00 00 01 02 03 00 00 00 00 00 00 00 00
00C33380	02 00 00 00 C3 66 47 4E C3 66 47 4E 00 00 00 00
00C33390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C333A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C333B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

รูปที่ 16.4: OllyDbg:

0..2:

Listing 16.9:

```
#include <stdio.h>

char a[3][4];
```

```

int main()
{
    int x, y;

    //
    for (x=0; x<3; x++)
        for (y=0; y<4; y++)
            a[x][y]=0;

    // 0..2:
    for (x=0; x<3; x++)
        a[x][2]=x;
};

```

. 0, 1 2.

Address	Hex dump
01033380	00 00 00 00 00 00 01 00 00 00 02 00 02 00 00 00
01033390	00 00 00 00 00 1E HH EF 31 1E HH EF 31 00 00 00 00
010333A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010333B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

รูปที่ 16.5: OllyDbg:

16.5.2

```

#include <stdio.h>

char a[3][4];

char get_by_coordinates1 (char array[3][4], int a, int b)
{
    return array[a][b];
};

char get_by_coordinates2 (char *array, int a, int b)
{
    //
    // 4
    return array[a*4+b];
};

char get_by_coordinates3 (char *array, int a, int b)
{
    // ,
    //
    // 4
    return *(array+a*4+b);
};

int main()
{
    a[2][3]=123;
    printf ("%d\n", get_by_coordinates1(a, 2, 3));
    printf ("%d\n", get_by_coordinates2(a, 2, 3));
    printf ("%d\n", get_by_coordinates3(a, 2, 3));
};

```

Listing 16.10: MSVC 2013 x64

```

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates3 PROC
; RCX=
; RDX=a
; R8=b
    movsxd    rax, r8d
; EAX=b
    movsxd    r9, edx
; R9=a
    add       rax, rcx
; RAX=b+
    movzx     eax, BYTE PTR [rax+r9*4]
; AL= RAX+R9*4=b++a*4==a*4+b
    ret       0
get_by_coordinates3 ENDP

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates2 PROC
    movsxd    rax, r8d
    movsxd    r9, edx
    add       rax, rcx
    movzx     eax, BYTE PTR [rax+r9*4]
    ret       0
get_by_coordinates2 ENDP

array$ = 8
a$ = 16
b$ = 24
get_by_coordinates1 PROC
    movsxd    rax, r8d
    movsxd    r9, edx
    add       rax, rcx
    movzx     eax, BYTE PTR [rax+r9*4]
    ret       0
get_by_coordinates1 ENDP

```

16.5.3

:

Listing 16.11:

```

#include <stdio.h>

int a[10][20][30];

void insert(int x, int y, int z, int value)
{
    a[x][y][z]=value;
};

```

x86

(MSVC 2010):

Listing 16.12: MSVC 2010

```

_DATA    SEGMENT
COMM     _a:DWORD:01770H
_DATA    ENDS
PUBLIC   _insert
_TEXT    SEGMENT
_x$ = 8           ; size = 4
_y$ = 12          ; size = 4
_z$ = 16          ; size = 4
_value$ = 20      ; size = 4
_insert   PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _x$[ebp]
    imul    eax, 2400           ; eax=600*4*x
    mov     ecx, DWORD PTR _y$[ebp]
    imul    ecx, 120           ; ecx=30*4*y
    lea     edx, DWORD PTR _a[eax+ecx] ; edx=a + 600*4*x + 30*4*y
    mov     eax, DWORD PTR _z$[ebp]
    mov     ecx, DWORD PTR _value$[ebp]
    mov     DWORD PTR [edx+eax*4], ecx ; *(edx+z*4)=
    pop     ebp
    ret     0
_insert   ENDP
_TEXT    ENDS

```

Listing 16.13: GCC 4.4.1

```

insert      public insert
            proc near

x           = dword ptr 8
y           = dword ptr 0Ch
z           = dword ptr 10h
value       = dword ptr 14h

            push    ebp
            mov     ebp, esp
            push    ebx
            mov     ebx, [ebp+x]
            mov     eax, [ebp+y]
            mov     ecx, [ebp+z]
            lea     edx, [eax+eax]           ; edx=y*2
            mov     eax, edx                ; eax=y*2
            shl     eax, 4                  ; eax=(y*2)<<4 = y*2*16 = y*32
            sub     eax, edx                ; eax=y*32 - y*2=y*30
            imul    edx, ebx, 600           ; edx=x*600
            add     eax, edx                ; eax=eax+edx=y*30 + x*600
            lea     edx, [eax+ecx]          ; edx=y*30 + x*600 + z
            mov     eax, [ebp+value]
            mov     dword ptr ds:a[edx*4], eax ; *(a+edx*4)=
            pop     ebx
            pop     ebp
            retn

insert      endp

```

$$. : (y + y) \ll 4 - (y + y) = (2y) \ll 4 - 2y = 2 \cdot 16 \cdot y - 2y = 32y - 2y = 30y. .$$

16.6

บทที่ 17

17.1

17.1.1 x86

```
HANDLE fh;

fh=CreateFile ("file", GENERIC_WRITE | GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_ALWAYS↵
↵ , FILE_ATTRIBUTE_NORMAL, NULL);
```

(MSVC 2010):

Listing 17.1: MSVC 2010

```
push    0
push    128                ; 00000080H
push    4
push    0
push    1
push    -1073741824        ; c0000000H
push    OFFSET $SG78813
call    DWORD PTR __imp__CreateFileA@28
mov     DWORD PTR _fh$[ebp], eax
```

WinNT.h:

Listing 17.2: WinNT.h

```
#define GENERIC_READ          (0x80000000L)
#define GENERIC_WRITE         (0x40000000L)
#define GENERIC_EXECUTE       (0x20000000L)
#define GENERIC_ALL           (0x10000000L)
```

, GENERIC_READ | GENERIC_WRITE = 0x80000000 | 0x40000000 = 0xC0000000, CreateFile()¹.

Listing 17.3: KERNEL32.DLL (Windows XP SP3 x86)

```
.text:7C83D429      test    byte ptr [ebp+dwDesiredAccess+3], 40h
.text:7C83D42D      mov     [ebp+var_8], 1
.text:7C83D434      jz      short loc_7C83D417
.text:7C83D436      jmp     loc_7C810817
```

(7.3.1 on page 21)).

```
if ((dwDesiredAccess&0x40000000) == 0) goto loc_7C83D417
```

¹[msdn.microsoft.com/en-us/library/aa363858\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/aa363858(VS.85).aspx)

17.2

:

```

#include <stdio.h>

#define IS_SET(flag, bit)      ((flag) & (bit))
#define SET_BIT(var, bit)     ((var) |= (bit))
#define REMOVE_BIT(var, bit)  ((var) &= ~(bit))

int f(int a)
{
    int rt=a;

    SET_BIT (rt, 0x4000);
    REMOVE_BIT (rt, 0x200);

    return rt;
};

int main()
{
    f(0x12340678);
};

```

17.2.1 x86

MSVC

(MSVC 2010):

Listing 17.4: MSVC 2010

```

_rt$ = -4          ; size = 4
_a$ = 8           ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    push    ecx
    mov     eax, DWORD PTR _a$[ebp]
    mov     DWORD PTR _rt$[ebp], eax
    mov     ecx, DWORD PTR _rt$[ebp]
    or      ecx, 16384          ; 00004000H
    mov     DWORD PTR _rt$[ebp], ecx
    mov     edx, DWORD PTR _rt$[ebp]
    and     edx, -513          ; ffffffffH
    mov     DWORD PTR _rt$[ebp], edx
    mov     eax, DWORD PTR _rt$[ebp]
    mov     esp, ebp
    pop     ebp
    ret     0
_f ENDP

```

MSVC

Listing 17.5: MSVC


```

_a$ = 8 ; size = 4
_f PROC
    mov     eax, DWORD PTR _a$[esp-4]
    and     eax, -513 ; fffffdfffH
    or      eax, 16384 ; 00004000H
    ret     0
_f ENDP

```

17.3

SHL (SHift Left) SHR (SHift Right) .

[15.1.2 on page 57](#), [15.2.1 on page 60](#).

17.4

?population count?².

```

#include <stdio.h>

#define IS_SET(flag, bit)      ((flag) & (bit))

int f(unsigned int a)
{
    int i;
    int rt=0;

    for (i=0; i<32; i++)
        if (IS_SET (a, 1<<i))
            rt++;

    return rt;
};

int main()
{
    f(0x12345678); // test
};

```

$1 \ll i \quad i = 0 \dots 31$:

2

1 << 0	1	1	1
1 << 1	2 ¹	2	2
1 << 2	2 ²	4	4
1 << 3	2 ³	8	8
1 << 4	2 ⁴	16	0x10
1 << 5	2 ⁵	32	0x20
1 << 6	2 ⁶	64	0x40
1 << 7	2 ⁷	128	0x80
1 << 8	2 ⁸	256	0x100
1 << 9	2 ⁹	512	0x200
1 << 10	2 ¹⁰	1024	0x400
1 << 11	2 ¹¹	2048	0x800
1 << 12	2 ¹²	4096	0x1000
1 << 13	2 ¹³	8192	0x2000
1 << 14	2 ¹⁴	16384	0x4000
1 << 15	2 ¹⁵	32768	0x8000
1 << 16	2 ¹⁶	65536	0x10000
1 << 17	2 ¹⁷	131072	0x20000
1 << 18	2 ¹⁸	262144	0x40000
1 << 19	2 ¹⁹	524288	0x80000
1 << 20	2 ²⁰	1048576	0x100000
1 << 21	2 ²¹	2097152	0x200000
1 << 22	2 ²²	4194304	0x400000
1 << 23	2 ²³	8388608	0x800000
1 << 24	2 ²⁴	16777216	0x1000000
1 << 25	2 ²⁵	33554432	0x2000000
1 << 26	2 ²⁶	67108864	0x4000000
1 << 27	2 ²⁷	134217728	0x8000000
1 << 28	2 ²⁸	268435456	0x10000000
1 << 29	2 ²⁹	536870912	0x20000000
1 << 30	2 ³⁰	1073741824	0x40000000
1 << 31	2 ³¹	2147483648	0x80000000

ssl_private.h Apache 2.4.6:

```
/**
 * Define the SSL options
 */
#define SSL_OPT_NONE (0)
#define SSL_OPT_RELSET (1<<0)
#define SSL_OPT_STDENVVARS (1<<1)
#define SSL_OPT_EXPORTCERTDATA (1<<3)
#define SSL_OPT_FAKEBASICAUTH (1<<4)
#define SSL_OPT_STRICTREQUIRE (1<<5)
#define SSL_OPT_OPTRENEGOTIATE (1<<6)
#define SSL_OPT_LEGACYDNFORMAT (1<<7)
```

17.4.1 x86

MSVC

(MSVC 2010):

Listing 17.6: MSVC 2010

```

_rt$ = -8          ; size = 4
_i$ = -4          ; size = 4
_a$ = 8           ; size = 4
_f PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 8
    mov     DWORD PTR _rt$[ebp], 0
    mov     DWORD PTR _i$[ebp], 0
    jmp     SHORT $LN4@f
$LN3@f:
    mov     eax, DWORD PTR _i$[ebp]    ; i
    add     eax, 1
    mov     DWORD PTR _i$[ebp], eax
$LN4@f:
    cmp     DWORD PTR _i$[ebp], 32    ; 00000020H
    jge     SHORT $LN2@f              ; ?
    mov     edx, 1
    mov     ecx, DWORD PTR _i$[ebp]
    shl     edx, cl                   ; EDX=EDX<<CL
    and     edx, DWORD PTR _a$[ebp]
    je      SHORT $LN1@f              ; ?
    ;
    mov     eax, DWORD PTR _rt$[ebp]
    add     eax, 1                    ; rt
    mov     DWORD PTR _rt$[ebp], eax
$LN1@f:
    jmp     SHORT $LN3@f
$LN2@f:
    mov     eax, DWORD PTR _rt$[ebp]
    mov     esp, ebp
    pop     ebp
    ret     0
_f        ENDP

```

17.4.2 x64

:

```

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

#define IS_SET(flag, bit)      ((flag) & (bit))

int f(uint64_t a)
{
    uint64_t i;
    int rt=0;

    for (i=0; i<64; i++)
        if (IS_SET (a, 1ULL<<i))
            rt++;

    return rt;
};

```

MSVC 2010

Listing 17.7: MSVC 2010

```

a$ = 8
f      PROC
; RCX =
        xor     eax, eax
        mov     edx, 1
        lea     r8d, QWORD PTR [rax+64]
; R8D=64
        npad    5
$LL4@f:
        test    rdx, rcx
; ?
; .
        je      SHORT $LN3@f
        inc     eax      ; rt++
$LN3@f:
        rol     rdx, 1    ; RDX=RDX<<1
        dec     r8        ; R8--
        jne     SHORT $LL4@f
        fatret    0
f      ENDP

```

ROL SHL, ?rotate left? ?shift left?, SHL.

R8 . .

:

RDX	R8
0x0000000000000001	64
0x0000000000000002	63
0x0000000000000004	62
0x0000000000000008	61
...	...
0x4000000000000000	2
0x8000000000000000	1

MSVC 2012

Listing 17.8: MSVC 2012

```

a$ = 8
f      PROC
; RCX =
        xor     eax, eax
        mov     edx, 1
        lea     r8d, QWORD PTR [rax+32]
; EDX = 1, R8D = 32
        npad    5
$LL4@f:
;
        test    rdx, rcx
        je      SHORT $LN3@f
        inc     eax      ; rt++
$LN3@f:
        rol     rdx, 1    ; RDX=RDX<<1
; -----

```

```

;
    test    rdx, rcx
    je      SHORT $LN11@f
    inc     eax      ; rt++
$LN11@f:
    rol     rdx, 1   ; RDX=RDX<<1
; -----
    dec     r8       ; R8--
    jne     SHORT $LL4@f
    fatret   0
f      ENDP

```

MSVC 2012 MSVC 2010, .

17.5

17.5.1

Listing 17.9:

```

if (input&0x40)
    ...

```

Listing 17.10: x86

```

TEST REG, 40h
JNZ is_set
;

```

Listing 17.11: x86

```

TEST REG, 40h
JZ is_cleared
;

```

17.5.2

Listing 17.12:

```

if ((value>>n)&1)
    ....

```

Listing 17.13: x86

```

; REG=input_value
; CL=n
SHR REG, CL
AND REG, 1

```

Listing 17.14:

```

if (value & (1<<n))
    ....

```

Listing 17.15: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
AND input_value, REG
```

17.5.3

Listing 17.16:

```
value=value|0x40;
```

Listing 17.17: x86

```
OR REG, 40h
```

17.5.4

Listing 17.18:

```
value=value|(1<<n);
```

Listing 17.19: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
OR input_value, REG
```

17.5.5

Listing 17.20:

```
value=value&(~0x40);
```

Listing 17.21: x86

```
AND REG, 0FFFFFFBFh
```

Listing 17.22: x64

```
AND REG, 0FFFFFFFFFFFFFFBFh
```

17.5.6

Listing 17.23:

```
value=value&(~(1<<n));
```

Listing 17.24: x86

```
; CL=n
MOV REG, 1
SHL REG, CL
NOT REG
AND input_value, REG
```

บทที่ 18

```
#include <stdint.h>

//
#define RNG_a 1664525
#define RNG_c 1013904223

static uint32_t rand_state;

void my_srand (uint32_t init)
{
    rand_state=init;
}

int my_rand ()
{
    rand_state=rand_state*RNG_a;
    rand_state=rand_state+RNG_c;
    return rand_state & 0x7fff;
}
```

. [\[Pre+07\]](#).

18.1 x86

Listing 18.1: MSVC 2013

```
_BSS    SEGMENT
_rand_state DD  01H DUP (?)
_BSS    ENDS

_init$ = 8
_srand  PROC
    mov     eax, DWORD PTR _init$[esp-4]
    mov     DWORD PTR _rand_state, eax
    ret     0
_srand  ENDP

_TEXT   SEGMENT
_rand   PROC
    imul    eax, DWORD PTR _rand_state, 1664525
    add     eax, 1013904223          ; 3c6ef35fH
    mov     DWORD PTR _rand_state, eax
    and     eax, 32767              ; 00007fffH
    ret     0
_rand   ENDP
```

```
_TEXT    ENDS
```

:

Listing 18.2: MSVC 2013

```
_BSS     SEGMENT
_rand_state DD    01H DUP (?)
_BSS     ENDS

_init$ = 8
_srand PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _init$[ebp]
    mov     DWORD PTR _rand_state, eax
    pop     ebp
    ret     0
_srand ENDP

_TEXT    SEGMENT
_rand PROC
    push    ebp
    mov     ebp, esp
    imul    eax, DWORD PTR _rand_state, 1664525
    mov     DWORD PTR _rand_state, eax
    mov     ecx, DWORD PTR _rand_state
    add     ecx, 1013904223          ; 3c6ef35fH
    mov     DWORD PTR _rand_state, ecx
    mov     eax, DWORD PTR _rand_state
    and     eax, 32767              ; 00007fffH
    pop     ebp
    ret     0
_rand ENDP

_TEXT    ENDS
```

18.2 x64

Listing 18.3: MSVC 2013 x64

```
_BSS     SEGMENT
rand_state DD    01H DUP (?)
_BSS     ENDS

init$ = 8
my_srand PROC
; ECX =
    mov     DWORD PTR rand_state, ecx
    ret     0
my_srand ENDP

_TEXT    SEGMENT
my_rand PROC
    imul    eax, DWORD PTR rand_state, 1664525      ; 0019660dH
    add     eax, 1013904223                          ; 3c6ef35fH
```



```
    mov     DWORD PTR rand_state, eax
    and     eax, 32767                ; 00007fffH
    ret     0
my_rand ENDP

_TEXT     ENDS
```

บทที่ 19

19.1 MSVC:

Listing 19.1: WinBase.h

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME;
```

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

void main()
{
    SYSTEMTIME t;
    GetSystemTime (&t);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
        t.wYear, t.wMonth, t.wDay,
        t.wHour, t.wMinute, t.wSecond);

    return;
};
```

(MSVC 2010):

Listing 19.2: MSVC 2010 /GS-

```
_t$ = -16 ; size = 16
_main      PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 16
    lea     eax, DWORD PTR _t$[ebp]
    push    eax
    call    DWORD PTR __imp__GetSystemTime@4
    movzx   ecx, WORD PTR _t$[ebp+12] ; wSecond
    push    ecx
    movzx   edx, WORD PTR _t$[ebp+10] ; wMinute
    push    edx
```

```

movzx  eax, WORD PTR _t$[ebp+8] ; wHour
push   eax
movzx  ecx, WORD PTR _t$[ebp+6] ; wDay
push   ecx
movzx  edx, WORD PTR _t$[ebp+2] ; wMonth
push   edx
movzx  eax, WORD PTR _t$[ebp] ; wYear
push   eax
push   OFFSET $SG78811 ; '%04d-%02d-%02d %02d:%02d:%02d', 0aH, 00H
call   _printf
add    esp, 28
xor    eax, eax
mov    esp, ebp
pop    ebp
ret    0
_main  ENDP

```

19.1.1

```

#include <windows.h>
#include <stdio.h>

void main()
{
    WORD array[8];
    GetSystemTime (array);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
        array[0] /* wYear */, array[1] /* wMonth */, array[3] /* wDay */,
        array[4] /* wHour */, array[5] /* wMinute */, array[6] /* wSecond */);

    return;
};

```

```

systemtime2.c(7) : warning C4133: 'function' : incompatible types - from 'WORD [8]' to '↵
↳ LPSYSTEMTIME'

```

:

Listing 19.3: MSVC 2010

```

$SG78573 DB      '%04d-%02d-%02d %02d:%02d:%02d', 0aH, 00H

_array$ = -16    ; size = 16
_main  PROC
    push    ebp
    mov     ebp, esp
    sub     esp, 16
    lea     eax, DWORD PTR _array$[ebp]
    push    eax
    call    DWORD PTR __imp__GetSystemTime@4
    movzx   ecx, WORD PTR _array$[ebp+12] ; wSecond
    push    ecx
    movzx   edx, WORD PTR _array$[ebp+10] ; wMinute
    push    edx
    movzx   eax, WORD PTR _array$[ebp+8] ; wHour
    push    eax

```

```

    movzx    ecx, WORD PTR _array$[ebp+6] ; wDay
    push     ecx
    movzx    edx, WORD PTR _array$[ebp+2] ; wMonth
    push     edx
    movzx    eax, WORD PTR _array$[ebp] ; wYear
    push     eax
    push     OFFSET $SG78573
    call     _printf
    add      esp, 28
    xor      eax, eax
    mov      esp, ebp
    pop      ebp
    ret      0
_main      ENDP

```

!

.

.

19.2

```

#include <windows.h>
#include <stdio.h>

void main()
{
    SYSTEMTIME *t;

    t=(SYSTEMTIME *)malloc (sizeof (SYSTEMTIME));

    GetSystemTime (t);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
        t->wYear, t->wMonth, t->wDay,
        t->wHour, t->wMinute, t->wSecond);

    free (t);

    return;
};

```

Listing 19.4: MSVC

```

_main      PROC
    push     esi
    push     16
    call     _malloc
    add      esp, 4
    mov      esi, eax
    push     esi
    call     DWORD PTR __imp__GetSystemTime@4
    movzx    eax, WORD PTR [esi+12] ; wSecond
    movzx    ecx, WORD PTR [esi+10] ; wMinute
    movzx    edx, WORD PTR [esi+8] ; wHour
    push     eax
    movzx    eax, WORD PTR [esi+6] ; wDay

```

```

push    ecx
movzx   ecx, WORD PTR [esi+2] ; wMonth
push    edx
movzx   edx, WORD PTR [esi] ; wYear
push    eax
push    ecx
push    edx
push    OFFSET $SG78833
call    _printf
push    esi
call    _free
add     esp, 32
xor     eax, eax
pop     esi
ret     0
_main   ENDP

```

```

#include <windows.h>
#include <stdio.h>

void main()
{
    WORD *t;

    t=(WORD *)malloc (16);

    GetSystemTime (t);

    printf ("%04d-%02d-%02d %02d:%02d:%02d\n",
        t[0] /* wYear */, t[1] /* wMonth */, t[3] /* wDay */,
        t[4] /* wHour */, t[5] /* wMinute */, t[6] /* wSecond */);

    free (t);

    return;
};

```

:

Listing 19.5: MSVC

```

$SG78594 DB      '%04d-%02d-%02d %02d:%02d:%02d', 0aH, 00H

_main   PROC
        push    esi
        push    16
        call    _malloc
        add     esp, 4
        mov     esi, eax
        push    esi
        call    DWORD PTR __imp__GetSystemTime@4
        movzx   eax, WORD PTR [esi+12]
        movzx   ecx, WORD PTR [esi+10]
        movzx   edx, WORD PTR [esi+8]
        push    eax
        movzx   eax, WORD PTR [esi+6]
        push    ecx
        movzx   ecx, WORD PTR [esi+2]
        push    edx

```

```

    movzx    edx, WORD PTR [esi]
    push     eax
    push     ecx
    push     edx
    push     OFFSET $SG78594
    call     _printf
    push     esi
    call     _free
    add      esp, 32
    xor      eax, eax
    pop      esi
    ret      0
_main      ENDP

```

19.3

```

#include <stdio.h>

struct s
{
    char a;
    int b;
    char c;
    int d;
};

void f(struct s s)
{
    printf ("a=%d; b=%d; c=%d; d=%d\n", s.a, s.b, s.c, s.d);
};

int main()
{
    struct s tmp;
    tmp.a=1;
    tmp.b=2;
    tmp.c=3;
    tmp.d=4;
    f(tmp);
};

```

19.3.1 x86

Listing 19.6: MSVC 2012 /GS- /Ob0

```

1  _tmp$ = -16
2  _main      PROC
3      push    ebp
4      mov     ebp, esp
5      sub     esp, 16
6      mov     BYTE PTR _tmp$[ebp], 1      ;  a
7      mov     DWORD PTR _tmp$[ebp+4], 2   ;  b
8      mov     BYTE PTR _tmp$[ebp+8], 3    ;  c
9      mov     DWORD PTR _tmp$[ebp+12], 4  ;  d
10     sub     esp, 16

```

```

11  mov     eax, esp
12  mov     ecx, DWORD PTR _tmp$[ebp] ;
13  mov     DWORD PTR [eax], ecx
14  mov     edx, DWORD PTR _tmp$[ebp+4]
15  mov     DWORD PTR [eax+4], edx
16  mov     ecx, DWORD PTR _tmp$[ebp+8]
17  mov     DWORD PTR [eax+8], ecx
18  mov     edx, DWORD PTR _tmp$[ebp+12]
19  mov     DWORD PTR [eax+12], edx
20  call    _f
21  add     esp, 16
22  xor     eax, eax
23  mov     esp, ebp
24  pop     ebp
25  ret     0
26 _main   ENDP
27
28 _s$ = 8 ; size = 16
29 ?f@@YAXUs@@@Z PROC ; f
30     push  ebp
31     mov   ebp, esp
32     mov   eax, DWORD PTR _s$[ebp+12]
33     push  eax
34     movsx ecx, BYTE PTR _s$[ebp+8]
35     push  ecx
36     mov   edx, DWORD PTR _s$[ebp+4]
37     push  edx
38     movsx eax, BYTE PTR _s$[ebp]
39     push  eax
40     push  OFFSET $SG3842
41     call  _printf
42     add   esp, 20
43     pop   ebp
44     ret   0
45 ?f@@YAXUs@@@Z ENDP ; f
46 _TEXT    ENDS

```

(/Zp1) (/Zp[n] pack structures on n-byte boundary).

Listing 19.7: MSVC 2012 /GS- /Zp1

```

1  _main   PROC
2      push  ebp
3      mov   ebp, esp
4      sub   esp, 12
5      mov   BYTE PTR _tmp$[ebp], 1      ; a
6      mov   DWORD PTR _tmp$[ebp+1], 2   ; b
7      mov   BYTE PTR _tmp$[ebp+5], 3    ; c
8      mov   DWORD PTR _tmp$[ebp+6], 4   ; d
9      sub   esp, 12                      ;
10     mov   eax, esp
11     mov   ecx, DWORD PTR _tmp$[ebp] ;
12     mov   DWORD PTR [eax], ecx
13     mov   edx, DWORD PTR _tmp$[ebp+4]
14     mov   DWORD PTR [eax+4], edx
15     mov   cx, WORD PTR _tmp$[ebp+8]
16     mov   WORD PTR [eax+8], cx
17     call  _f
18     add   esp, 12

```

```

19     xor     eax, eax
20     mov     esp, ebp
21     pop     ebp
22     ret     0
23 _main     ENDP
24
25 _TEXT     SEGMENT
26 _s$ = 8 ; size = 10
27 ?f@@YAXUs@@@Z PROC    ; f
28     push    ebp
29     mov     ebp, esp
30     mov     eax, DWORD PTR _s$[ebp+6]
31     push    eax
32     movsx   ecx, BYTE PTR _s$[ebp+5]
33     push    ecx
34     mov     edx, DWORD PTR _s$[ebp+1]
35     push    edx
36     movsx   eax, BYTE PTR _s$[ebp]
37     push    eax
38     push    OFFSET $SG3842
39     call    _printf
40     add     esp, 20
41     pop     ebp
42     ret     0
43 ?f@@YAXUs@@@Z ENDP    ; f

```

Listing 19.8: WinNT.h

```
#include "pshpack1.h"
```

Listing 19.9: WinNT.h

```
#include "pshpack4.h"           // 4 byte packing is the default
```

Listing 19.10: PshPack1.h

```

#if ! (defined(lint) || defined(RC_INVOKED))
#if ( _MSC_VER >= 800 && !defined(_M_I86)) || defined(_PUSHPOP_SUPPORTED)
#pragma warning(disable:4103)
#if !(defined( MIDL_PASS )) || defined( __midl )
#pragma pack(push,1)
#else
#pragma pack(1)
#endif
#else
#pragma pack(1)
#endif
#endif /* ! (defined(lint) || defined(RC_INVOKED)) */

```

19.3.2

```

void f(char a, int b, char c, int d)
{
    printf ("a=%d; b=%d; c=%d; d=%d\n", a, b, c, d);
};

```


19.4

```

#include <stdio.h>

struct inner_struct
{
    int a;
    int b;
};

struct outer_struct
{
    char a;
    int b;
    struct inner_struct c;
    char d;
    int e;
};

void f(struct outer_struct s)
{
    printf ("a=%d; b=%d; c.a=%d; c.b=%d; d=%d; e=%d\n",
           s.a, s.b, s.c.a, s.c.b, s.d, s.e);
};

int main()
{
    struct outer_struct s;
    s.a=1;
    s.b=2;
    s.c.a=100;
    s.c.b=101;
    s.d=3;
    s.e=4;
    f(s);
};

```

...

(MSVC 2010):

Listing 19.11: MSVC 2010 /Ob0

```

$SG2802 DB      'a=%d; b=%d; c.a=%d; c.b=%d; d=%d; e=%d', 0aH, 00H

_TEXT      SEGMENT
_s$ = 8
_f        PROC
    mov     eax, DWORD PTR _s$[esp+16]
    movsx   ecx, BYTE PTR _s$[esp+12]
    mov     edx, DWORD PTR _s$[esp+8]
    push    eax
    mov     eax, DWORD PTR _s$[esp+8]
    push    ecx
    mov     ecx, DWORD PTR _s$[esp+8]
    push    edx
    movsx   edx, BYTE PTR _s$[esp+8]
    push    eax
    push    ecx
    push    edx

```

```

push    OFFSET $SG2802 ; 'a=%d; b=%d; c.a=%d; c.b=%d; d=%d; e=%d'
call    _printf
add     esp, 28
ret     0
_f      ENDP

_s$ = -24
_main   PROC
    sub     esp, 24
    push    ebx
    push    esi
    push    edi
    mov     ecx, 2
    sub     esp, 24
    mov     eax, esp
    mov     BYTE PTR _s$[esp+60], 1
    mov     ebx, DWORD PTR _s$[esp+60]
    mov     DWORD PTR [eax], ebx
    mov     DWORD PTR [eax+4], ecx
    lea     edx, DWORD PTR [ecx+98]
    lea     esi, DWORD PTR [ecx+99]
    lea     edi, DWORD PTR [ecx+2]
    mov     DWORD PTR [eax+8], edx
    mov     BYTE PTR _s$[esp+76], 3
    mov     ecx, DWORD PTR _s$[esp+76]
    mov     DWORD PTR [eax+12], esi
    mov     DWORD PTR [eax+16], ecx
    mov     DWORD PTR [eax+20], edi
    call    _f
    add     esp, 24
    pop     edi
    pop     esi
    xor     eax, eax
    pop     ebx
    add     esp, 24
    ret     0
_main    ENDP

```

19.5

19.5.1

3:0 (4 Thai text placeholder)	Stepping
7:4 (4 Thai text placeholder)	Model
11:8 (4 Thai text placeholder)	Family
13:12 (2 Thai text placeholder)	Processor Type
19:16 (4 Thai text placeholder)	Extended Model
27:20 (8 Thai text placeholder)	Extended Family

```

#include <stdio.h>

#ifdef __GNUC__
static inline void cpuid(int code, int *a, int *b, int *c, int *d) {
    asm volatile("cpuid":"=a"(*a),"=b"(*b),"=c"(*c),"=d"(*d):"a"(code));
}
#endif

```

```

#ifdef _MSC_VER
#include <intrin.h>
#endif

struct CPUID_1_EAX
{
    unsigned int stepping:4;
    unsigned int model:4;
    unsigned int family_id:4;
    unsigned int processor_type:2;
    unsigned int reserved1:2;
    unsigned int extended_model_id:4;
    unsigned int extended_family_id:8;
    unsigned int reserved2:4;
};

int main()
{
    struct CPUID_1_EAX *tmp;
    int b[4];

#ifdef _MSC_VER
    __cpuid(b,1);
#endif

#ifdef __GNUC__
    cpuid (1, &b[0], &b[1], &b[2], &b[3]);
#endif

    tmp=(struct CPUID_1_EAX *)&b[0];

    printf ("stepping=%d\n", tmp->stepping);
    printf ("model=%d\n", tmp->model);
    printf ("family_id=%d\n", tmp->family_id);
    printf ("processor_type=%d\n", tmp->processor_type);
    printf ("extended_model_id=%d\n", tmp->extended_model_id);
    printf ("extended_family_id=%d\n", tmp->extended_family_id);

    return 0;
};

```

MSVC

:

Listing 19.12: MSVC 2008

```

_b$ = -16 ; size = 16
_main    PROC
    sub    esp, 16
    push   ebx

    xor    ecx, ecx
    mov    eax, 1
    cpuid
    push   esi
    lea    esi, DWORD PTR _b$[esp+24]

```

```

mov     DWORD PTR [esi], eax
mov     DWORD PTR [esi+4], ebx
mov     DWORD PTR [esi+8], ecx
mov     DWORD PTR [esi+12], edx

mov     esi, DWORD PTR _b$[esp+24]
mov     eax, esi
and     eax, 15
push    eax
push    OFFSET $SG15435 ; 'stepping=%d', 0aH, 00H
call    _printf

mov     ecx, esi
shr     ecx, 4
and     ecx, 15
push    ecx
push    OFFSET $SG15436 ; 'model=%d', 0aH, 00H
call    _printf

mov     edx, esi
shr     edx, 8
and     edx, 15
push    edx
push    OFFSET $SG15437 ; 'family_id=%d', 0aH, 00H
call    _printf

mov     eax, esi
shr     eax, 12
and     eax, 3
push    eax
push    OFFSET $SG15438 ; 'processor_type=%d', 0aH, 00H
call    _printf

mov     ecx, esi
shr     ecx, 16
and     ecx, 15
push    ecx
push    OFFSET $SG15439 ; 'extended_model_id=%d', 0aH, 00H
call    _printf

shr     esi, 20
and     esi, 255
push    esi
push    OFFSET $SG15440 ; 'extended_family_id=%d', 0aH, 00H
call    _printf
add     esp, 48
pop     esi

xor     eax, eax
pop     ebx

add     esp, 16
ret     0
_main   ENDP

```

บทที่ 20

20.1

```
#include <stdint.h>

uint64_t f ()
{
    return 0x1234567890ABCDEF;
};
```

20.1.1 x86

Listing 20.1: MSVC 2010

```
_f    PROC
      mov     eax, -1867788817      ; 90abcdefH
      mov     edx, 305419896       ; 12345678H
      ret     0
_f    ENDP
```

20.2

```
#include <stdint.h>

uint64_t f_add (uint64_t a, uint64_t b)
{
    return a+b;
};

void f_add_test ()
{
#ifdef __GNUC__
    printf ("%lld\n", f_add(12345678901234, 2345678901234));
#else
    printf ("%I64d\n", f_add(12345678901234, 2345678901234));
#endif
};

uint64_t f_sub (uint64_t a, uint64_t b)
{
    return a-b;
};
```

20.2.1 x86

Listing 20.2: MSVC 2012 /Ob1

```

_a$ = 8          ; size = 8
_b$ = 16         ; size = 8
_f_add PROC
    mov     eax, DWORD PTR _a$[esp-4]
    add     eax, DWORD PTR _b$[esp-4]
    mov     edx, DWORD PTR _a$[esp]
    adc     edx, DWORD PTR _b$[esp]
    ret     0
_f_add ENDP

_f_add_test PROC
    push    5461          ; 00001555H
    push    1972608889    ; 75939f79H
    push    2874          ; 00000b3aH
    push    1942892530    ; 73ce2ff_subH
    call    _f_add
    push    edx
    push    eax
    push    OFFSET $SG1436 ; '%I64d', 0aH, 00H
    call    _printf
    add     esp, 28
    ret     0
_f_add_test ENDP

_f_sub PROC
    mov     eax, DWORD PTR _a$[esp-4]
    sub     eax, DWORD PTR _b$[esp-4]
    mov     edx, DWORD PTR _a$[esp]
    sbb     edx, DWORD PTR _b$[esp]
    ret     0
_f_sub ENDP

```

f_add_test() .

.

...

. SUB: .

.

20.3

```

#include <stdint.h>

uint64_t f_mul (uint64_t a, uint64_t b)
{
    return a*b;
};

uint64_t f_div (uint64_t a, uint64_t b)

```

```

{
    return a/b;
};

uint64_t f_rem (uint64_t a, uint64_t b)
{
    return a % b;
};

```

20.3.1 x86

Listing 20.3: MSVC 2013 /Ob1

```

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_mul PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __allmul ; long long multiplication
    pop     ebp
    ret     0
_f_mul ENDP

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_div PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]
    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __aulldiv ; unsigned long long division
    pop     ebp
    ret     0
_f_div ENDP

_a$ = 8 ; size = 8
_b$ = 16 ; size = 8
_f_rem PROC
    push    ebp
    mov     ebp, esp
    mov     eax, DWORD PTR _b$[ebp+4]
    push    eax
    mov     ecx, DWORD PTR _b$[ebp]

```

```

    push    ecx
    mov     edx, DWORD PTR _a$[ebp+4]
    push    edx
    mov     eax, DWORD PTR _a$[ebp]
    push    eax
    call    __aullrem ; unsigned long long remainder
    pop     ebp
    ret     0
_f_rem    ENDP

```

20.4

```

#include <stdint.h>

uint64_t f (uint64_t a)
{
    return a>>7;
};

```

20.4.1 x86

Listing 20.4: MSVC 2012 /Ob1

```

_a$ = 8      ; size = 8
_f          PROC
    mov     eax, DWORD PTR _a$[esp-4]
    mov     edx, DWORD PTR _a$[esp]
    shrd    eax, edx, 7
    shr     edx, 7
    ret     0
_f          ENDP

```

20.5

```

#include <stdint.h>

int64_t f (int32_t a)
{
    return a;
};

```

20.5.1 x86

Listing 20.5: MSVC 2012

```

_a$ = 8
_f          PROC
    mov     eax, DWORD PTR _a$[esp-4]

```


	cdq	
	ret	0
_f	ENDP	

.

บทที่ 21

21.1 x86-64

- RAX, RBX, RCX, RDX, RBP, RSP, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15.

Thai text placeholder	
RAX ^{x64}	
	EAX
	AX
AH	AL

Thai text placeholder	
R8	
	R8D
	R8W
	R8L

XMM0-XMM15.

- System V AMD64 ABI (Linux, *BSD, Mac OS X)[[Mit13](#)] fastcall, RDI, RSI, RDX, RCX, R8, R9 . .
-
- .

ภาค II



บทที่ 22

			()
01111111	0x7f	127	127
01111110	0x7e	126	126
...			
00000110	0x6	6	6
00000101	0x5	5	5
00000100	0x4	4	4
00000011	0x3	3	3
00000010	0x2	2	2
00000001	0x1	1	1
00000000	0x0	0	0
11111111	0xff	255	-1
11111110	0xfe	254	-2
11111101	0xfd	253	-3
11111100	0xfc	252	-4
11111011	0xfb	251	-5
11111010	0xfa	250	-6
...			
10000010	0x82	130	-126
10000001	0x81	129	-127
10000000	0x80	128	-128

Thai text placeholder

-
- - `int64_t` (-9,223,372,036,854,775,808..9,223,372,036,854,775,807) (- 9.2.. 9.2)
`0x8000000000000000..0x7FFFFFFFFFFFFFFF`),
- `int` (-2,147,483,648..2,147,483,647 (- 2.15.. 2.15Gb) `0x80000000..0x7FFFFFFF`),
- `char` (-128..127 `0x80..0x7F`),
- `ssize_t`.
- `uint64_t` (0..18,446,744,073,709,551,615 (18) `0..0xFFFFFFFFFFFFFFFF`),
- `unsigned int` (0..4,294,967,295 (4.3Gb) `0..0xFFFFFFFF`),
- `unsigned char` (0..255 `0..0xFF`),
- `size_t`.
-
- [20.5 on page 99.](#)
-

-
- IDIV/IMUL DIV/MUL

บทที่ 23

-
-
- [19.2 on page 87.](#)

ภาค III

บทที่ 24

(win32)

.

: Process Monitor¹ SysInternals.Wireshark².

.

blog.yurichev.com.

24.1 Windows API

. . . CRT³.

- (advapi32.dll): RegEnumKeyEx^{4 5}, RegEnumValue^{6 5}, RegGetValue^{7 5}, RegOpenKeyEx^{8 5}, RegQueryValueEx^{9 5}.
- (kernel32.dll): GetPrivateProfileString^{10 5}.
- (user32.dll): MessageBox^{11 5}, MessageBoxEx^{12 5}, SetDlgItemText^{13 5}, GetDlgItemText^{14 5}.
- : (user32.dll): LoadMenu^{15 5}.
- (ws2_32.dll): WSARcv¹⁶, WSASend¹⁷.
- (kernel32.dll): CreateFile^{18 5}, ReadFile¹⁹, ReadFileEx²⁰, WriteFile²¹, WriteFileEx²².

¹<http://go.yurichev.com/17301>²<http://go.yurichev.com/17303>³C runtime library⁴MSDN⁵⁶MSDN⁷MSDN⁸MSDN⁹MSDN¹⁰MSDN¹¹MSDN¹²MSDN¹³MSDN¹⁴MSDN¹⁵MSDN¹⁶MSDN¹⁷MSDN¹⁸MSDN¹⁹MSDN²⁰MSDN²¹MSDN²²MSDN

- Internet (wininet.dll): WinHttpOpen ²³.
- (wintrust.dll): WinVerifyTrust ²⁴.
- (msvcr*.dll): assert, itoa, ltoa, open, printf, read, strcmp, atol, atoi, fopen, fread, fwrite, memcmp, rand, strlen, strstr, strchr.

24.2 tracer:

```
--one-time-INT3-bp:somedll.dll!.*
```

```
--one-time-INT3-bp:somedll.dll!xml.*
```

Thai text placeholder

```
tracer -l:uptime.exe --one-time-INT3-bp:cygwin1.dll!.*
```

:

```
One-time INT3 breakpoint: cygwin1.dll!__main (called from uptime.exe!OEP+0x6d (0x40106d))
One-time INT3 breakpoint: cygwin1.dll!_getuid32 (called from uptime.exe!OEP+0xba3 (0x401ba3))
One-time INT3 breakpoint: cygwin1.dll!_getuid32 (called from uptime.exe!OEP+0xbaa (0x401baa))
One-time INT3 breakpoint: cygwin1.dll!_getegid32 (called from uptime.exe!OEP+0xcb7 (0x401cb7))
One-time INT3 breakpoint: cygwin1.dll!_getgid32 (called from uptime.exe!OEP+0xcbe (0x401cbe))
One-time INT3 breakpoint: cygwin1.dll!sysconf (called from uptime.exe!OEP+0x735 (0x401735))
One-time INT3 breakpoint: cygwin1.dll!setlocale (called from uptime.exe!OEP+0x7b2 (0x4017b2))
One-time INT3 breakpoint: cygwin1.dll!_open64 (called from uptime.exe!OEP+0x994 (0x401994))
One-time INT3 breakpoint: cygwin1.dll!_lseek64 (called from uptime.exe!OEP+0x7ea (0x4017ea))
One-time INT3 breakpoint: cygwin1.dll!read (called from uptime.exe!OEP+0x809 (0x401809))
One-time INT3 breakpoint: cygwin1.dll!sscanf (called from uptime.exe!OEP+0x839 (0x401839))
One-time INT3 breakpoint: cygwin1.dll!uname (called from uptime.exe!OEP+0x139 (0x401139))
One-time INT3 breakpoint: cygwin1.dll!time (called from uptime.exe!OEP+0x22e (0x40122e))
One-time INT3 breakpoint: cygwin1.dll!localtime (called from uptime.exe!OEP+0x236 (0x401236))
One-time INT3 breakpoint: cygwin1.dll!sprintf (called from uptime.exe!OEP+0x25a (0x40125a))
One-time INT3 breakpoint: cygwin1.dll!setutent (called from uptime.exe!OEP+0x3b1 (0x4013b1))
One-time INT3 breakpoint: cygwin1.dll!getutent (called from uptime.exe!OEP+0x3c5 (0x4013c5))
One-time INT3 breakpoint: cygwin1.dll!endutent (called from uptime.exe!OEP+0x3e6 (0x4013e6))
One-time INT3 breakpoint: cygwin1.dll!puts (called from uptime.exe!OEP+0x4c3 (0x4014c3))
```

²³MSDN

²⁴MSDN

บทที่ 25

25.1

25.1.1

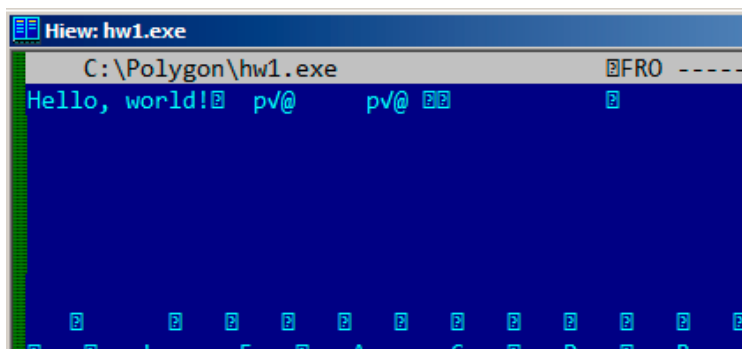
(ASCIIZ-).

. [Rit79] :

A minor difference was that the unit of I/O was the word, not the byte, because the PDP-7 was a word-addressed machine. In practice this meant merely that all programs dealing with character streams ignored null characters, because null was used to pad a file to an even number of characters.

:

```
int main()
{
    printf ("Hello, world!\n");
};
```



รูปที่ 25.1: Hiew

25.1.2 Borland Delphi

:

Listing 25.1: Delphi

```
CODE:00518AC8          dd 19h
CODE:00518ACC aLoading__Plea db 'Loading... , please wait.',0
...
CODE:00518AFC          dd 10h
```

CODE:00518B00 aPreparingRun__ db 'Preparing run...',0

25.1.3 Unicode

...

: UTF-8 () UTF-16LE (Windows).

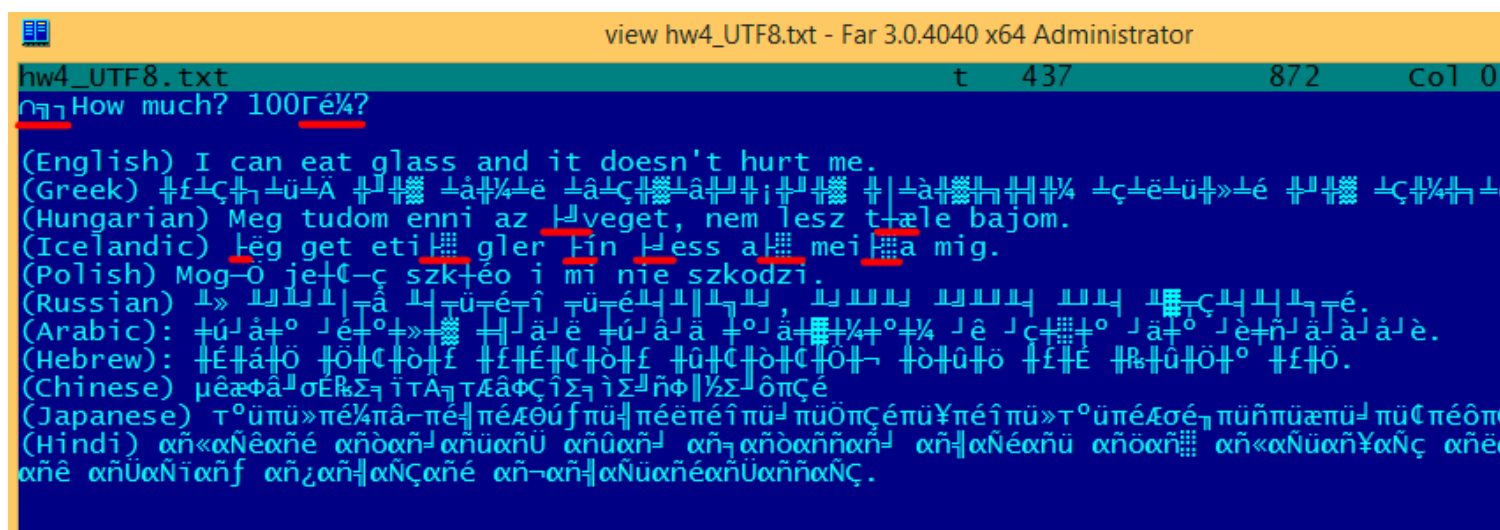
UTF-8

UTF-8 . . .

1:

How much? 100€?

(English) I can eat glass and it doesn't hurt me.
 (Greek) Μπορώ να φάω σπασμένα γυαλιά χωρίς να πάθω τίποτα.
 (Hungarian) Meg tudom enni az üveget, nem lesz tőle bajom.
 (Icelandic) Ég get etið gler án þess að meiða mig.
 (Polish) Mogę jeść szkło i mi nie szkodzi.
 (Russian) Я могу есть стекло, оно мне не вредит.
 (Arabic): أنا قادر على أكل الزجاج و هذا لا يؤلمني.
 (Hebrew): אני יכול לאכול זכוכית וזה לא מזיק לי.
 (Chinese) 我能吞下玻璃而不伤身体。
 (Japanese) 私はガラスを食べられます。それは私を傷つけません。
 (Hindi) मैं काँच खा सकता हूँ और मुझे उससे कोई चोट नहीं पहुंचती.



รูปที่ 25.2: FAR: UTF-8

.....

. BOM² .

UTF-16LE

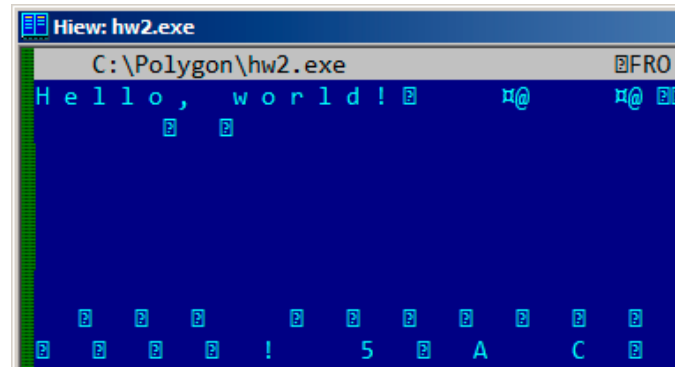
-A -W. (wide). .

:

¹: <http://go.yurichev.com/17304>

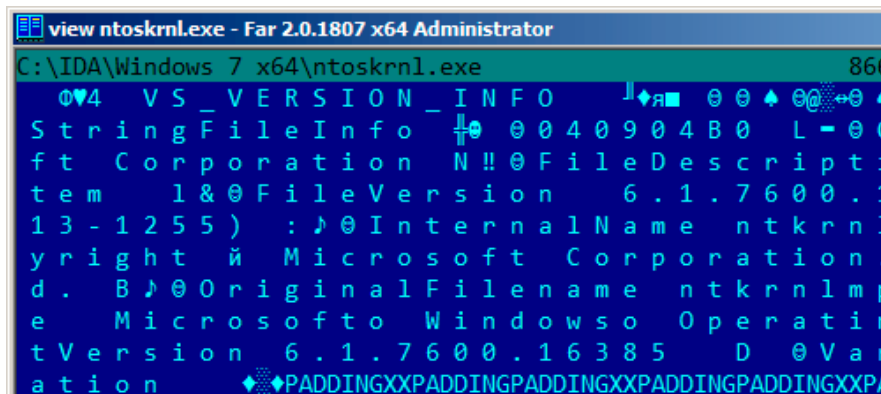
²Byte order mark

```
int wmain()
{
    wprintf (L"Hello, world!\n");
};
```



รูปที่ 25.3: Hiew

:

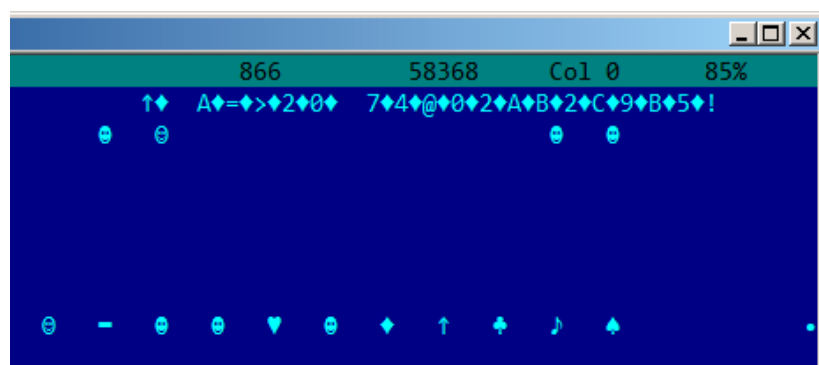


รูปที่ 25.4: Hiew

:

```
.data:0040E000 aHelloWorld:
.data:0040E000          unicode 0, <Hello, world!>
.data:0040E000          dw 0Ah, 0
```

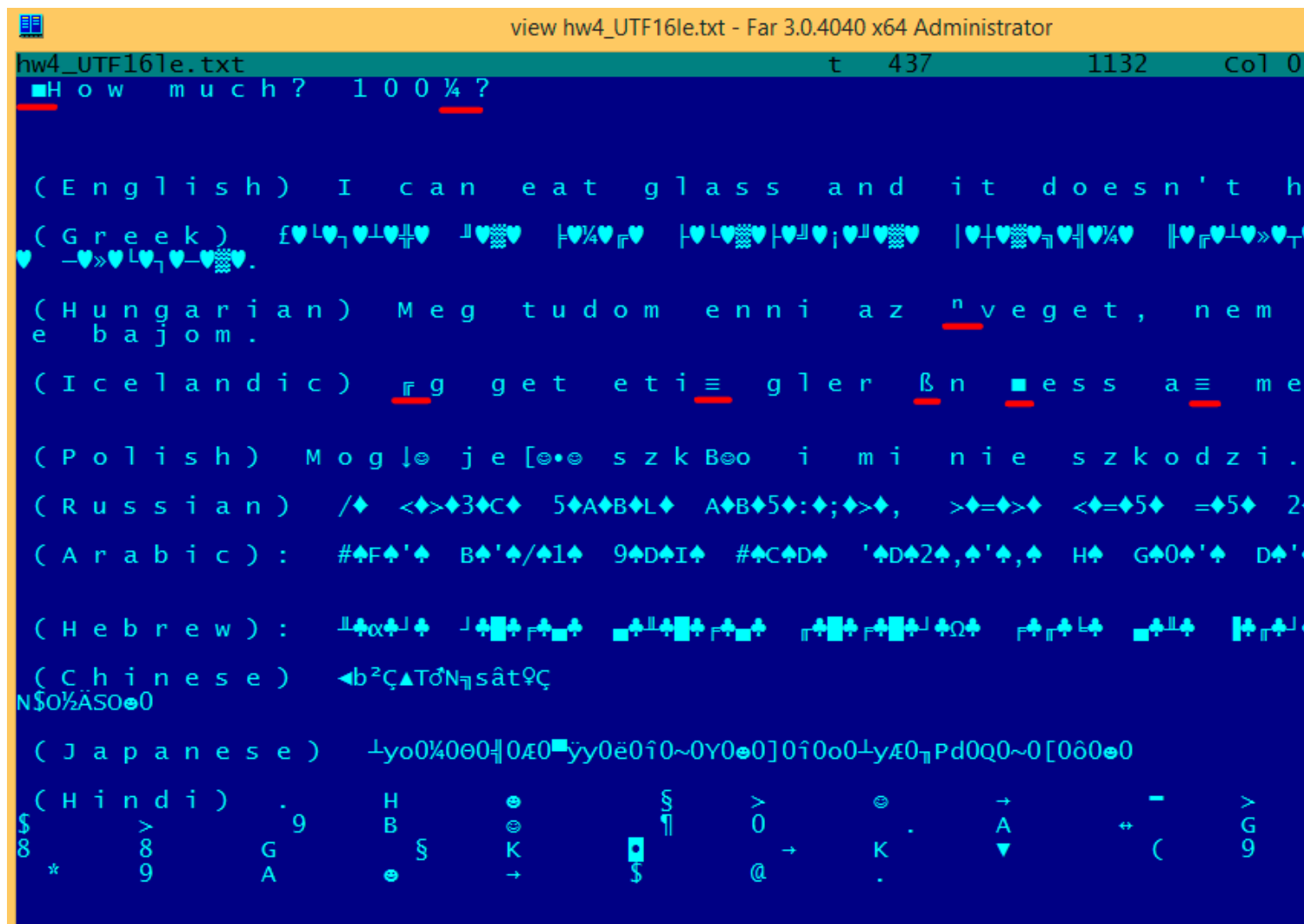
:



รูปที่ 25.5: Hiew: UTF-16LE

3. 0x400-0x4FF.

...



รูปที่ 25.6: FAR: UTF-16LE

...

25.1.4 Base64

.

AVjbbVSVfcUMu1xvjaMgjNtueRwBbxnyJw8dpGnLW8ZW8aKG3v4Y0icuQT+qEJAp9lA0uWs=

WVjbbVSVfcUMu1xvjaMgjNtueRwBbxnyJw8dpGnLW8ZW8aKG3v4Y0icuQT+qEJAp9lA0uQ==

25.2

4.

: blog.yurichev.com.

³wikipedia

⁴blog.yurichev.com

25.3

5.

http://192.168.0.1/userRpmNatDebugRpm26525557/start_art.html.

base64.

⁵<http://sekurak.pl/tp-link-httpftp-backdoor/>

บทที่ 26

Listing 26.1:

```
.text:107D4B29 mov dx, [ecx+42h]
.text:107D4B2D cmp edx, 1
.text:107D4B30 jz short loc_107D4B4A
.text:107D4B32 push 1Eh
.text:107D4B37 push offset aWrite_c ; "write.c"
.text:107D4B3C push offset aTdTd_planarcon ; "td->td_planarconfig == PLANARCONFIG_CON"
.text:107D4B41 call ds:_assert

...

.text:107D52CA mov edx, [ebp-4]
.text:107D52CD and edx, 3
.text:107D52D0 test edx, edx
.text:107D52D2 jz short loc_107D52E9
.text:107D52D4 push 58h
.text:107D52D6 push offset aDumpmode_c ; "dumpmode.c"
.text:107D52DB push offset aN30 ; "(n & 3) == 0"
.text:107D52E0 call ds:_assert

...

.text:107D6759 mov cx, [eax+6]
.text:107D675D cmp ecx, 0Ch
.text:107D6760 jle short loc_107D677A
.text:107D6762 push 2D8h
.text:107D6767 push offset aLzw_c ; "lzw.c"
.text:107D676C push offset aSpLzw_nbitsBit ; "sp->lzw_nbits <= BITS_MAX"
.text:107D6771 call ds:_assert
```

บทที่ 27

```
10, 100, 1000, .
: 10=0xA, 100=0x64, 1000=0x3E8, 10000=0x2710.
0xAAAAAAAA (10101010101010101010101010101010)
0x55555555 (01010101010101010101010101010101). 0x55AA, MBR1, ROM2.

var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476

Thai text placeholder

:
```

Listing 27.1: linux/lib/crc16.c

```
/** CRC table for the CRC-16. The poly is 0x8005 (x^16 + x^15 + x^2 + 1) */
u16 const crc16_table[256] = {
    0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
    0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
    0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
    ...
}
```

27.1 Magic numbers

?MZ?³.

```
cmp [buf], 0x6468544D ; "MThd"
jnz _error_not_a_MIDI_file
```

27.1.1 DHCP

DhcpExtractOptionsForValidation() DhcpExtractFullOptions():

Listing 27.2: dhcpcore.dll (Windows 7 x64)

```
.rdata:000007FF6483CBE8 dword_7FF6483CBE8 dd 63538263h ; DATA XREF: ↗
↳ DhcpExtractOptionsForValidation+79
.rdata:000007FF6483CBEC dword_7FF6483CBEC dd 63538263h ; DATA XREF: ↗
↳ DhcpExtractFullOptions+97
```

¹Master Boot Record

²Thai text placeholder

³[wikipedia](https://en.wikipedia.org/wiki/MZ_magic_number)

Listing 27.3: dhcpcore.dll (Windows 7 x64)

```
.text:000007FF6480875F  mov     eax, [rsi]
.text:000007FF64808761  cmp     eax, cs:dword_7FF6483CBE8
.text:000007FF64808767  jnz     loc_7FF64817179
```

Listing 27.4: dhcpcore.dll (Windows 7 x64)

```
.text:000007FF648082C7  mov     eax, [r12]
.text:000007FF648082CB  cmp     eax, cs:dword_7FF6483CBEC
.text:000007FF648082D1  jnz     loc_7FF648173AF
```

27.2

*binary grep*⁴.

⁴[GitHub](#)

บทที่ 28

Thai text placeholder

Thai text placeholder

```
cat EXCEL.lst | grep fdiv | grep -v dbl_ > EXCEL.fdiv
```

Thai text placeholder

Thai text placeholder

```
.text:3011E919 DC 33                                fdiv    qword ptr [ebx]
```

```
PID=13944|TID=28744|(0) 0x2f64e919 (Excel.exe!BASE+0x11e919)
EAX=0x02088006 EBX=0x02088018 ECX=0x00000001 EDX=0x00000001
ESI=0x02088000 EDI=0x00544804 EBP=0x0274FA3C ESP=0x0274F9F8
EIP=0x2F64E919
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=
FPU ST(0): 1.000000
```

[EBX].

```
.text:3011E91B DD 1E                                fstp    qword ptr [esi]
```

```
PID=32852|TID=36488|(0) 0x2f40e91b (Excel.exe!BASE+0x11e91b)
EAX=0x00598006 EBX=0x00598018 ECX=0x00000001 EDX=0x00000001
ESI=0x00598000 EDI=0x00294804 EBP=0x026CF93C ESP=0x026CF8F8
EIP=0x2F40E91B
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=C1 P
FPU ST(0): 0.333333
```

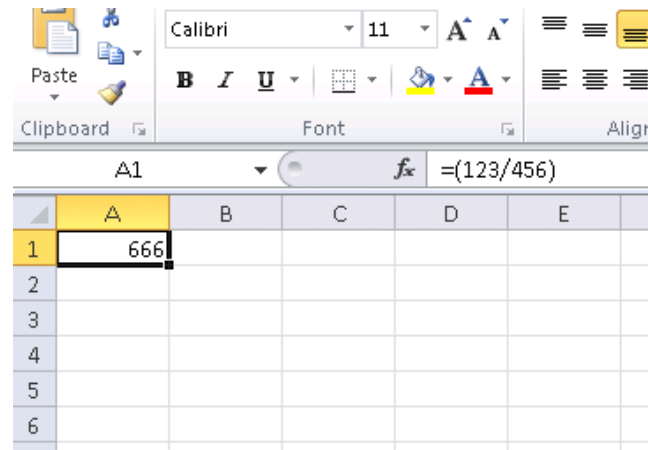
Thai text placeholder

```
tracer -l:excel.exe bpx=excel.exe!BASE+0x11E91B,set(st0,666)
```

```

PID=36540|TID=24056|(0) 0x2f40e91b (Excel.exe!BASE+0x11e91b)
EAX=0x00680006 EBX=0x00680018 ECX=0x00000001 EDX=0x00000001
ESI=0x00680000 EDI=0x00395404 EBP=0x0290FD9C ESP=0x0290FD58
EIP=0x2F40E91B
FLAGS=PF IF
FPU ControlWord=IC RC=NEAR PC=64bits PM UM OM ZM DM IM
FPU StatusWord=C1 P
FPU ST(0): 0.333333
Set ST0 register to 666.000000

```



รูปที่ 28.1:

```
tracer.exe -l:excel.exe bpx=excel.exe!BASE+0x1B7FCC,set(st0,666)
```

บทที่ 29

29.1

XOR op, op (, XOR EAX, EAX) ??. . Thai text placeholder.

:

```
gawk -e '$2=="xor" { tmp=substr($3, 0, length($3)-1); if (tmp!=$4) if($4!="esp") if ($4!="ebp")↵
↳ { print $1, $2, tmp, ",", $4 } }' filename.lst
```

29.2

Thai text placeholder

.

Windows 2003 (ntoskrnl.exe):

```
MultiplyTest proc near                ; CODE XREF: Get386Stepping
                xor     cx, cx
loc_620555:    ; CODE XREF: MultiplyTest+E
                push    cx
                call    Multiply
                pop     cx
                jb      short locret_620563
                loop    loc_620555
                clc
locret_620563:                ; CODE XREF: MultiplyTest+C
                retn
MultiplyTest endp

Multiply      proc near                ; CODE XREF: MultiplyTest+5
                mov     ecx, 81h
                mov     eax, 417A000h
                mul     ecx
                cmp     edx, 2
                stc
                jnz     short locret_62057F
                cmp     eax, 0FE7A000h
                stc
                jnz     short locret_62057F
                clc
```

```
locret_62057F:                ; CODE XREF: Multiply+10
                                ; Multiply+18
Multiply          retn
Multiply          endp
```

[WRK¹](#) v1.2 *WRK-v1.2\base\ntos\ke\i386\cpu.asm*.

¹Windows Research Kernel

บทที่ 30

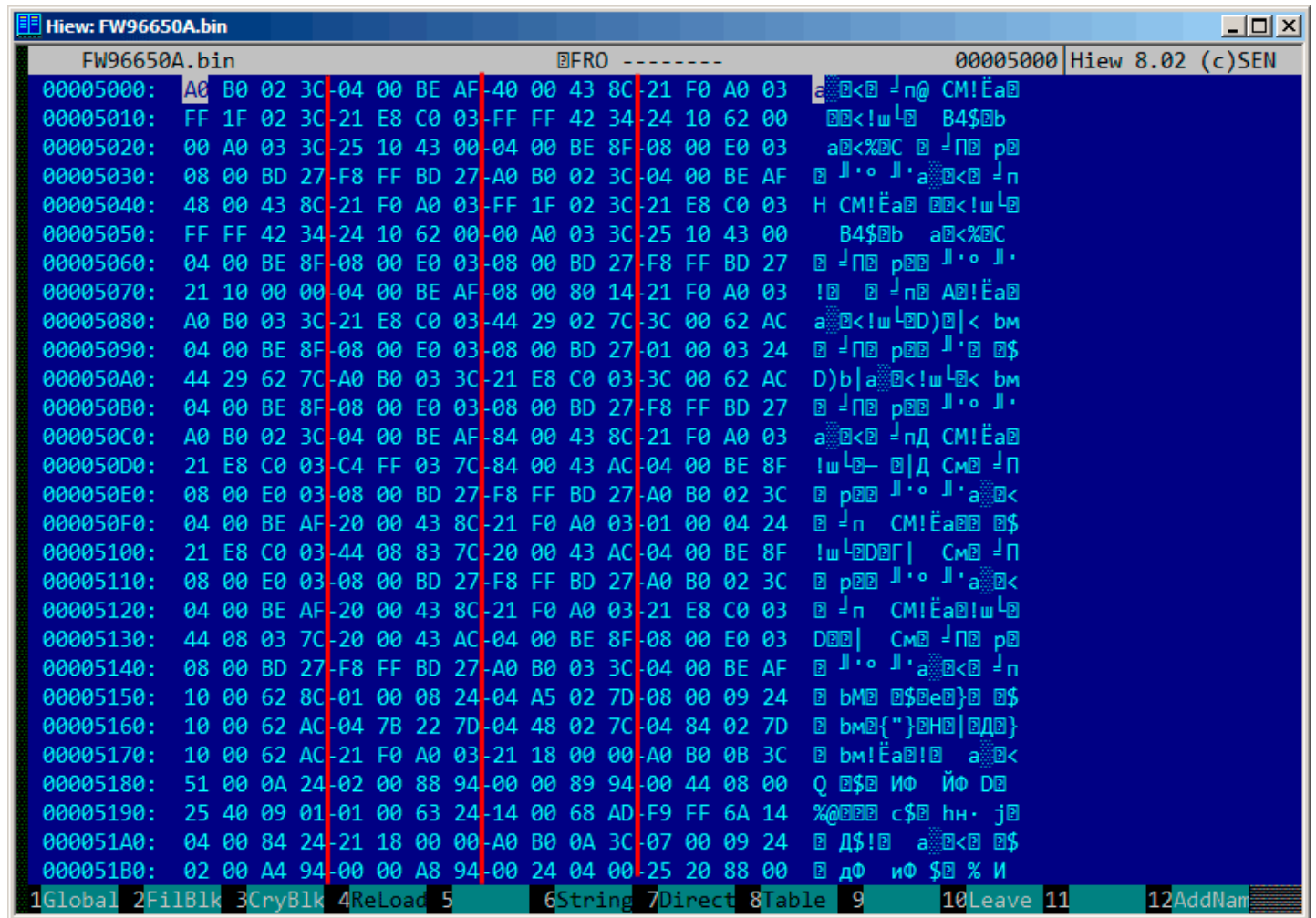
:

0x150bf66 (_kziaia+0x14), e=	1 [MOV EBX, [EBP+8]] [EBP+8]=0xf59c934
0x150bf69 (_kziaia+0x17), e=	1 [MOV EDX, [69AEB08h]] [69AEB08h]=0
0x150bf6f (_kziaia+0x1d), e=	1 [FS: MOV EAX, [2Ch]]
0x150bf75 (_kziaia+0x23), e=	1 [MOV ECX, [EAX+EDX*4]] [EAX+EDX*4]=0xf1ac360
0x150bf78 (_kziaia+0x26), e=	1 [MOV [EBP-4], ECX] ECX=0xf1ac360

บทที่ 31

31.1

31.2



รูปที่ 31.1: Hiew:

31.3

wikipedia.

31.3.1

.

31.3.2

ภาค IV

บทที่ 32

32.1 IDA

¹.

¹hex-rays.com/products/ida/support/download_freeware.shtml

บทที่ 33

33.1 tracer

tracer ¹

บทที่ 34

Thai text placeholder

hex-rays.com/products/decompiler/

บทที่ 35

- Microsoft Visual Studio Express¹:
- Hiew²:
- binary grep:

¹visualstudio.com/en-US/products/visual-studio-express-vs

²hiew.ru

ภาค V

บทที่ 36

36.1 Windows

[RA09].

36.2

[ISO13].

36.3 x86 / x86-64

[Int13], [AMD13]

36.4 ARM

<http://go.yurichev.com/17024>

36.5

[Sch94]

บทที่ 37

37.1 Windows

- [Microsoft: Raymond Chen](#)
- [nynaeve.net](#)

บทที่ 38

reddit.com/r/ReverseEngineering/ reddit.com/r/remath

Thai text placeholder

reverseengineering.stackexchange.com.

##re FreeNode¹.

¹freenode.net

บทที่ 39

<dennis(a)yurichev.com>

Thai text placeholder.

[GitHub](#).

[beginners.re.](#)

ROM Thai text placeholder	117
RA Thai text placeholder	43
CRT C runtime library	109
CPU Central processing unit	3
ISA Instruction Set Architecture	3
NOP No OPeration	23
ASCIIZ ASCII Zero ()	22
WRK Windows Research Kernel	122
GPR General Purpose Registers	3
BOM Byte order mark	112
MBR Master Boot Record	117

ดัชนี

- , 62
- , 16, 101
- C99
 - bool, 74
 - variable length arrays, 66
- const, 5
- for, 51
- if, 32, 42
- return, 6, 21
- switch, 41, 42
- while, 55
- alloca(), 11, 66, 106
- assert(), 116
- free(), 106
- longjmp(), 43
- malloc(), 88, 106
- memcmp(), 117
- memcpy(), 16
- rand(), 82, 109
- realloc(), 106
- scanf(), 16
- strlen(), 55
- , 40, 80
 - STL, 108
- , 108, 119, 123
- , 18
- , 66
- , 8, 9
- , 9, 25, 43
 - , 9
 - , 17
- , 42
- OllyDbg, 65, 69
- Oracle RDBMS, 6, 114
- ARM
 - ASR, 80
 - CSEL, 40
 - LSL, 80
 - LSR, 80
 - MOV, 4
 - MOVcc, 40
 - POP, 9
 - PUSH, 9
 - TEST, 56
- AWK, 121
- Base64, 114
- base64, 115
- bash, 28
- BASIC
 - POKE, 124
- binary grep, 118, 130
- Borland Delphi, 111
- cdecl, 13
- column-major order, 69
- Compiler intrinsic, 12
- Cygwin, 110
- DosBox, 123
- Error messages, 114
- fastcall, 6, 15
- FORTTRAN, 69
- Function epilogue, 8, 121
- Function prologue, 8, 121
- Hiew, 22, 35, 111
- IDA, 113
- Intel C++, 6
- jumptable, 44
- MD5, 117
- MIDI, 117
- MIPS, 124
- MS-DOS, 117, 123, 124
- Pascal, 111
- puts() printf(), 28
- Register allocation, 101
- row-major order, 69
- SAP, 108
- Security through obscurity, 115
- Shadow space, 27
- Signed numbers, 34, 104
- tracer, 110, 119, 123, 128

Unicode, [112](#)

UTF-16LE, [112](#)

UTF-8, [112](#)

Windows

KERNEL32.DLL, [74](#)

PDB, [108](#)

Structured Exception Handling, [12](#)

Win32, [74](#), [112](#)

x86

ADC, [97](#)

ADD, [6](#), [13](#), [26](#)

AND, [74](#), [75](#), [77](#), [80](#), [95](#)

CALL, [6](#), [9](#)

CBW, [104](#)

CDQ, [100](#), [104](#)

CDQE, [104](#)

CMOVcc, [38](#), [40](#)

CMP, [21](#)

CMPSB, [117](#)

CPUID, [93](#)

CWD, [104](#)

CWDE, [104](#)

DEC, [56](#)

DIV, [104](#)

DIVSD, [120](#)

FDIV, [119](#)

IDIV, [104](#)

IMUL, [26](#), [104](#)

INC, [56](#)

INT3, [110](#)

JA, [34](#), [104](#)

JAE, [34](#)

JB, [34](#), [104](#)

JBE, [34](#)

Jcc, [40](#)

JE, [43](#)

JG, [34](#), [104](#)

JGE, [33](#)

JL, [34](#), [104](#)

JLE, [33](#)

JMP, [9](#)

JNE, [21](#), [33](#)

JZ, [43](#)

LEA, [17](#), [26](#)

LOOP, [51](#), [54](#), [121](#)

MOV, [4](#), [6](#)

MOVSX, [56](#), [104](#)

MOVSXD, [68](#)

MOVZX, [88](#)

MUL, [104](#)

OR, [75](#)

POP, [6](#), [9](#)

PUSH, [6](#), [9](#), [17](#)

RCL, [121](#)

RET, [4](#), [6](#), [9](#)

ROL, [79](#)

SAR, [80](#), [104](#)

SBB, [97](#)

SHL, [58](#), [62](#), [80](#)

SHR, [60](#), [80](#), [95](#)

SHRD, [99](#)

SUB, [6](#), [21](#), [43](#)

TEST, [56](#), [74](#), [80](#)

XOR, [6](#), [21](#), [121](#)

, [21](#)

EAX, [21](#), [28](#)

EBP, [17](#), [26](#)

ESP, [13](#), [17](#)

JMP, [45](#)

ZF, [21](#), [74](#)

x86-64, [6](#), [14](#), [16](#), [17](#), [23](#), [26](#), [101](#)

บรรณานุกรม

- [AMD13] AMD. AMD64 Architecture Programmer's Manual. <http://go.yurichev.com/17284>. 2013.
- [Dij68] Edsger W. Dijkstra. ?Letters to the editor: go to statement considered harmful? In: Commun. ACM 11.3 (Mar. 1968), pp. 147–148. ISSN: 0001-0782. DOI: [10.1145/362929.362947](https://doi.org/10.1145/362929.362947). URL: <http://go.yurichev.com/17299>.
- [Fog13] Agner Fog. The microarchitecture of Intel, AMD and VIA CPUs / An optimization guide for assembly programmers and <http://go.yurichev.com/17278>. 2013.
- [Int13] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes:1, 2A, 2B, 2C, 3A, 3B, and 3C. <http://go.yurichev.com/17283>. 2013.
- [ISO13] ISO. ISO/IEC 14882:2011 (C++ 11 standard). <http://go.yurichev.com/17275>. 2013.
- [Ker88] Brian W. Kernighan. The C Programming Language. Ed. by Dennis M. Ritchie. 2nd. Prentice Hall Professional Technical Reference, 1988. ISBN: 0131103709.
- [Knu74] Donald E. Knuth. ?Structured Programming with go to Statements? In: ACM Comput. Surv. 6.4 (Dec. 1974). Also available as <http://go.yurichev.com/17271>, pp. 261–301. ISSN: 0360-0300. DOI: [10.1145/356635.356640](https://doi.org/10.1145/356635.356640). URL: <http://go.yurichev.com/17300>.
- [Mit13] Michael Matz / Jan Hubicka / Andreas Jaeger / Mark Mitchell. System V Application Binary Interface. AMD64 Architecture. <http://go.yurichev.com/17295>. 2013.
- [Pre+07] William H. Press et al. Numerical Recipes. 2007.
- [RA09] Mark E. Russinovich and David A. Solomon with Alex Ionescu. Windows® Internals: Including Windows Server 2008 and Windows Vista. 2009.
- [Rit79] Dennis M. Ritchie. ?The Evolution of the Unix Time-sharing System? In: (1979).
- [RT74] D. M. Ritchie and K. Thompson. ?The UNIX Time Sharing System? In: (1974). <http://go.yurichev.com/17270>.
- [Sch94] Bruce Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 1994.
- [Yur13] Dennis Yurichev. C/C++ programming language notes. <http://go.yurichev.com/17289>. 2013.