REVERSE ENGINEERING MACHINE CODE: PART 1

Function Conventions

- Standard Entry Sequence (cdecl)
  - Save the old base pointer
  - Set the new stack base pointer
  - Allocate space for variables

```
__function:
    push ebp      ; 55
    mov ebp, esp  ; 8BEC
    sub esp, x    ; Not always present
```

```
__function:
    enter       ; C8
    sub esp, x  ; Not always present
```

Function Conventions

- Standard Exit Sequence (cdecl)
  - Reload old stack pointer
  - Reload old stack base
  - Deallocate space for variables

```
...
    mov esp, ebp    ; 8BE5
    pop ebp         ; 5D
    ret             ; C3 near, CB far
```

```
...
    leave           ; C9
    ret             ; C3 near, CB far
```

Function Call Conventions

- cdecl
  - Used by GCC and GNU libraries
- stdcall
  - Used by Win32 API
  - Sometimes incorrectly called “pascal”
- fastcall
  - Many different implementations
  - Not standardized

Function Call Conventions

- cdecl
  - Parameters pushed right to left
  - EAX, ECX, EDX not preserved
  - Return values are returned in EAX
    - Floating point returns in ST0
  - Caller performs clean-up
- stdcall
  - Same as cdecl, except callee cleans-up
    - `RET imm` is a sign of this
- fastcall
  - One or more parameters passed in registers
  - MS VC++, GCC
    - First arg → ECX, second arg → EDX, remainder right → left

cdecl Function Call Convention

- Push Parameters on Stack
- Call the Function
- Save and Update EBP
- Save Registers that Will Be Overwritten
- Allocate Local Variables
- Execute Function
- Release Local Storage

cdecl Function Call Convention

- Restore Saved Registers
- Restore EBP
- Return
- Clean Up Parameters

stdcall Function Call Convention

- Push Parameters on Stack
- Call the Function
- Save and Update EBP
- Save Registers that Will Be Overwritten
- Allocate Local Variables
- Execute Function
- Release Local Storage

stdcall Function Call Convention

- Restore Saved Registers
- Restore EBP
- Clean Up Parameters
- Return

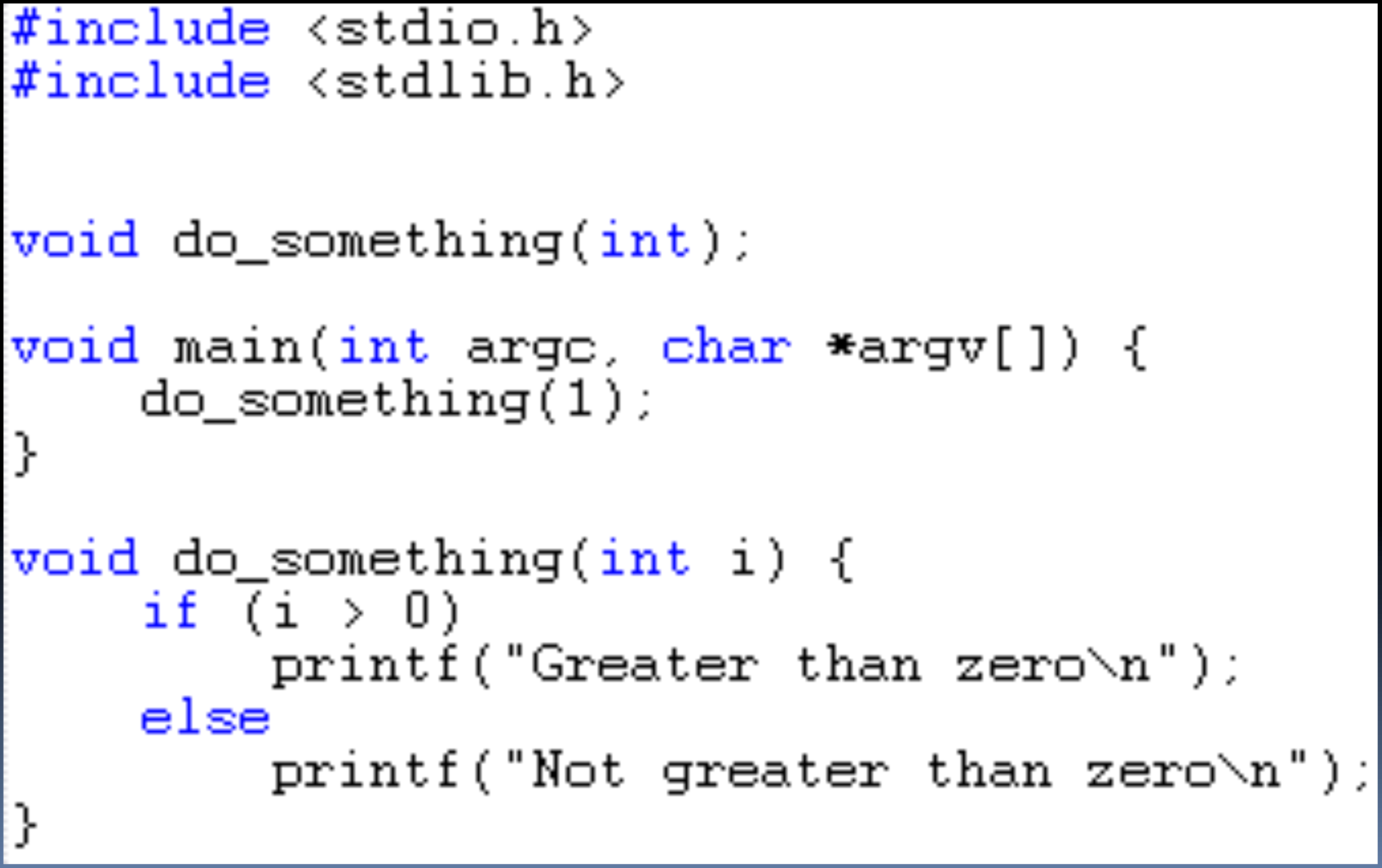
Function Call Conventions

- Others
  - pascal
    - Parameters pushed left to right
    - Windows 3.*
  - syscall
    - Parameter size passed in AL
  - safecall
    - Encapsulated COM error handling
  - thiscall
    - Either caller or callee clean-up
  - …

Control Statements

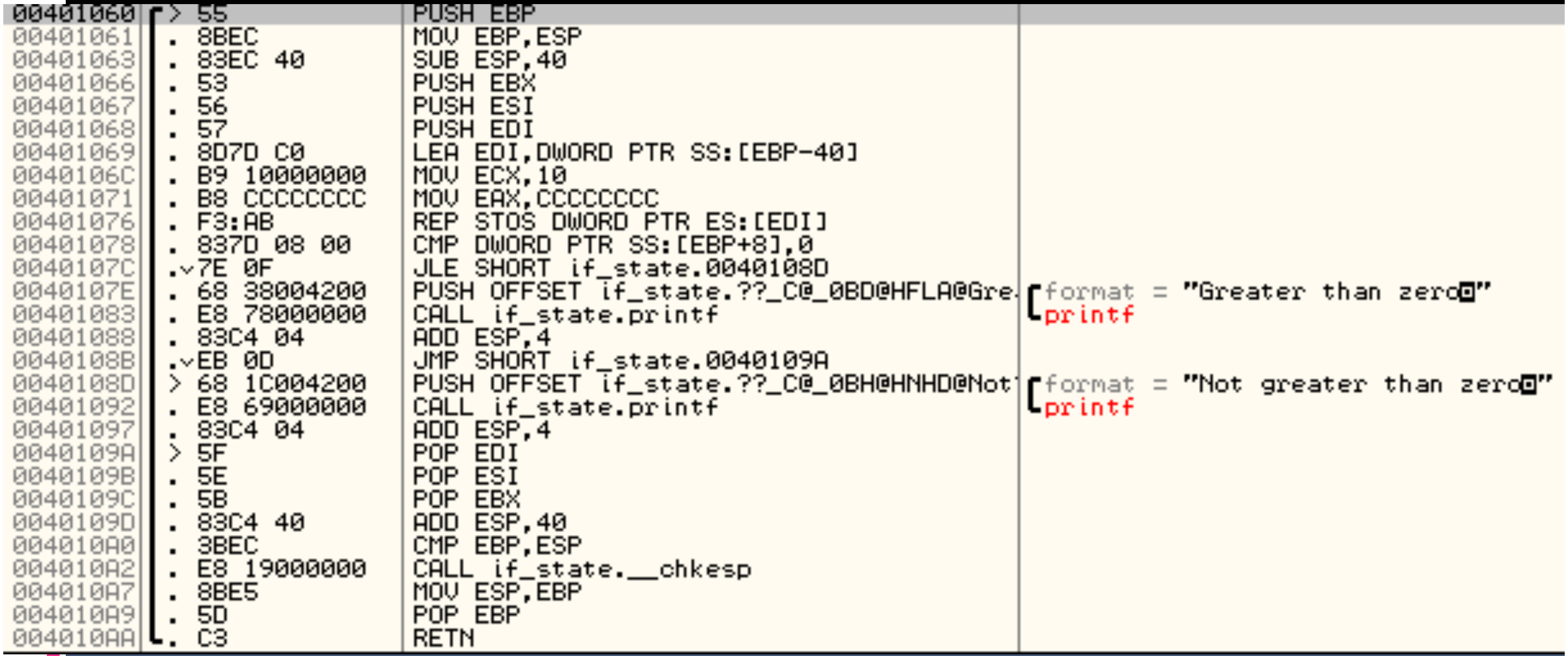
- If-Else
- Switch
- For
- While

If-Else Statement



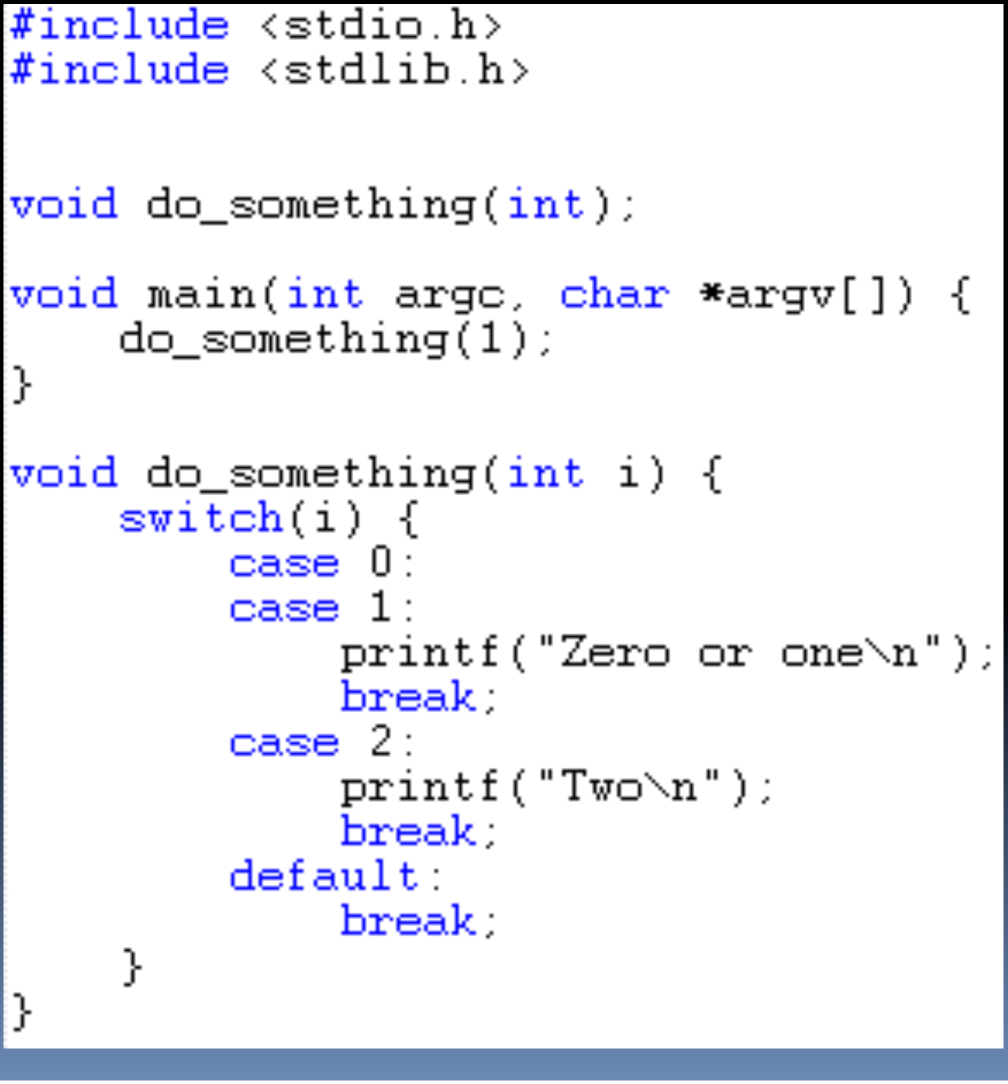
```
#include <stdio.h>
#include <stdlib.h>


void do_something(int);

void main(int argc, char *argv[]) {
    do_something(1);
}

void do_something(int i) {
    if (i > 0)
        printf("Greater than zero\n");
    else
        printf("Not greater than zero\n");
}
```

If-Else Statement



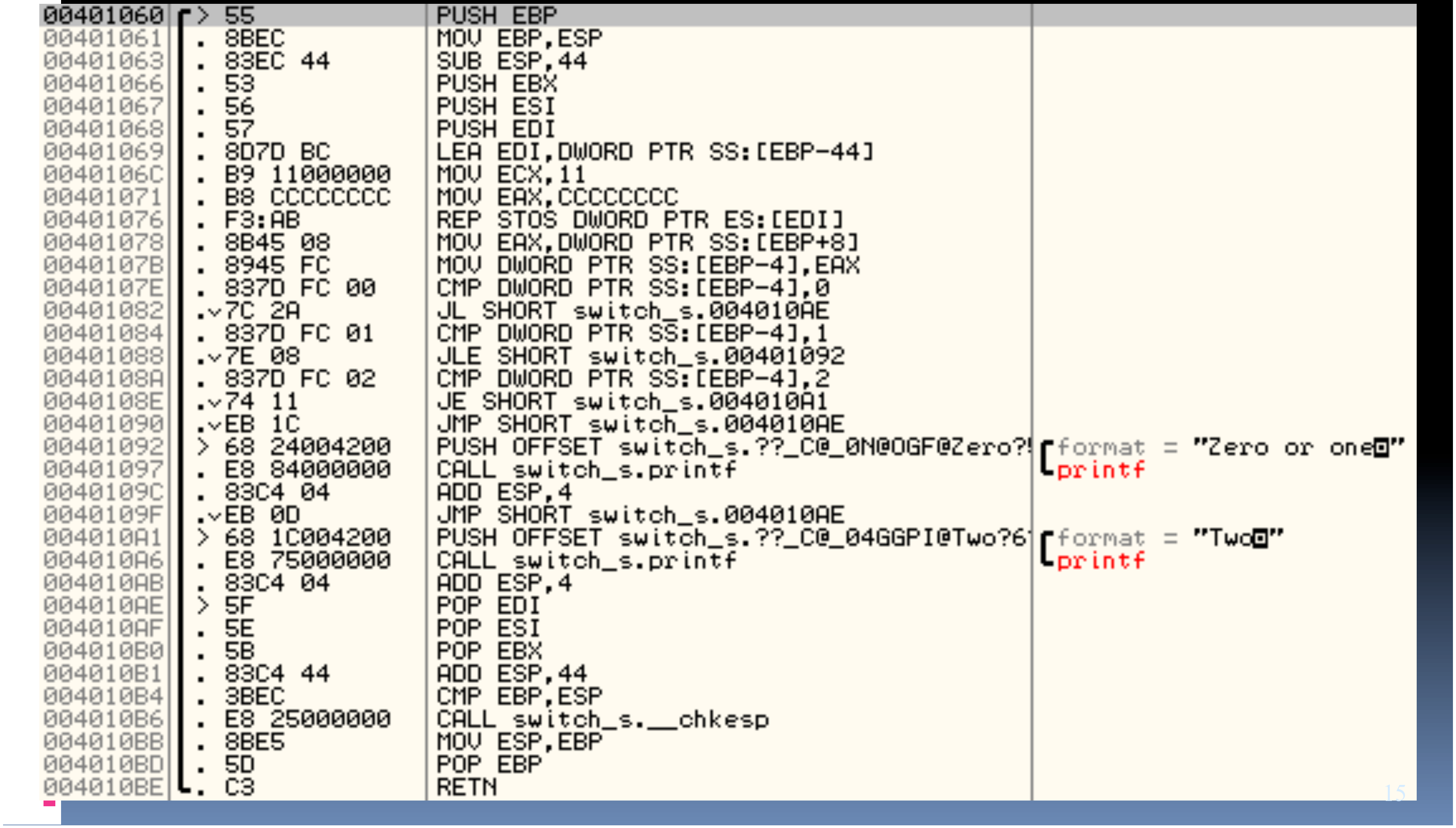
```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 40       SUB ESP,40
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D C0       LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000   MOV ECX,10
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00    CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F         JLE SHORT if_state.0040108D
0040107E │. 68 38004200   PUSH OFFSET if_state.??_C@_0BD@HFLA@Gre  ┌format = "Greater than zero◘"
00401083 │. E8 78000000   CALL if_state.printf                     └printf
00401088 │. 83C4 04       ADD ESP,4
0040108B │.vEB 0D         JMP SHORT if_state.0040109A
0040108D │> 68 1C004200   PUSH OFFSET if_state.??_C@_0BH@HNHD@Not  ┌format = "Not greater than zero◘"
00401092 │. E8 69000000   CALL if_state.printf                     └printf
00401097 │. 83C4 04       ADD ESP,4
0040109A │> 5F            POP EDI
0040109B │. 5E            POP ESI
0040109C │. 5B            POP EBX
0040109D │. 83C4 40       ADD ESP,40
004010A0 │. 3BEC          CMP EBP,ESP
004010A2 │. E8 19000000   CALL if_state.__chkesp
004010A7 │. 8BE5          MOV ESP,EBP
004010A9 │. 5D            POP EBP
004010AA └. C3            RETN
```

Switch Statement



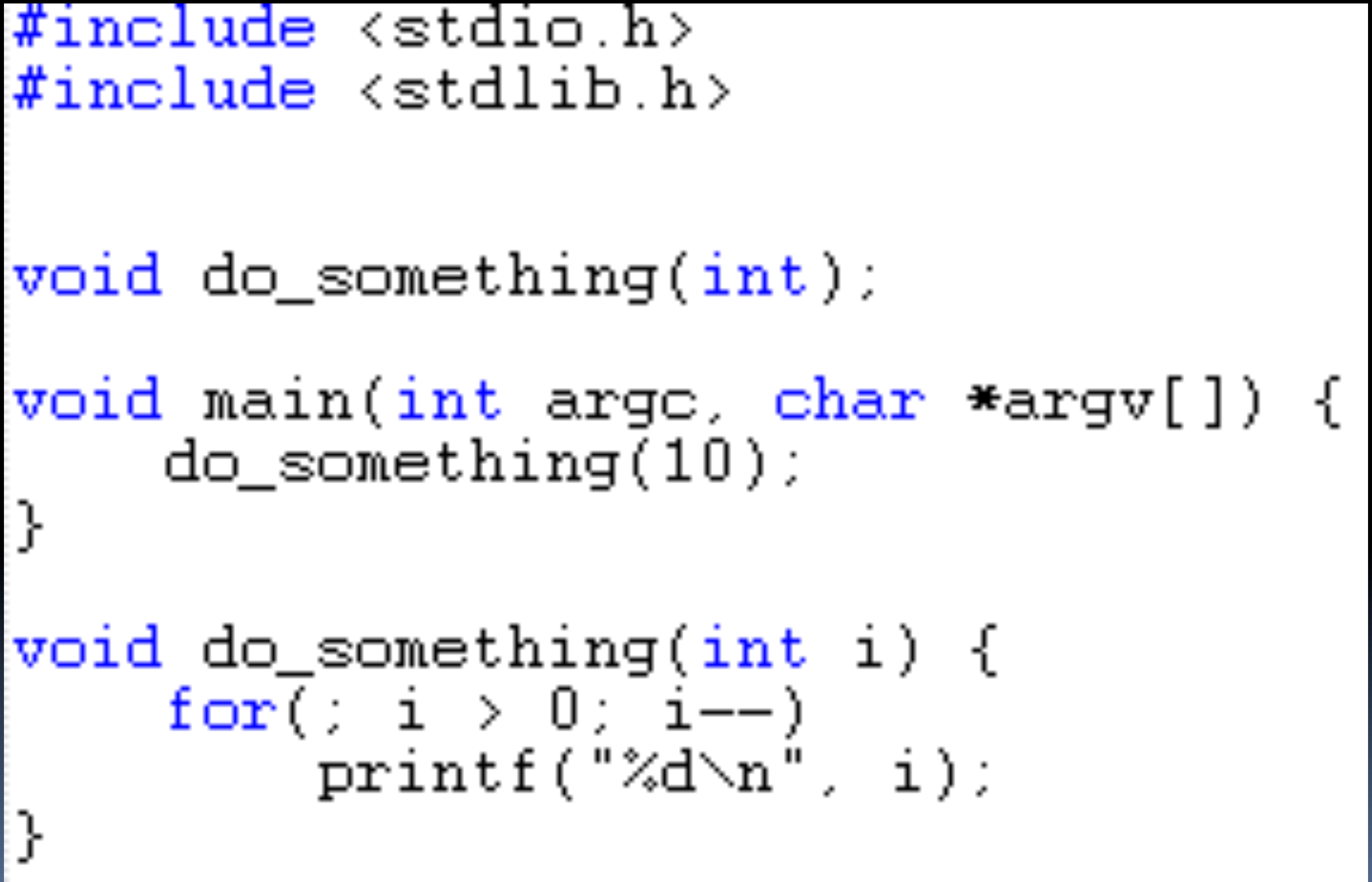
```
#include <stdio.h>
#include <stdlib.h>


void do_something(int);

void main(int argc, char *argv[]) {
    do_something(1);
}

void do_something(int i) {
    switch(i) {
        case 0:
        case 1:
            printf("Zero or one\n");
            break;
        case 2:
            printf("Two\n");
            break;
        default:
            break;
    }
}
```

Switch Statement



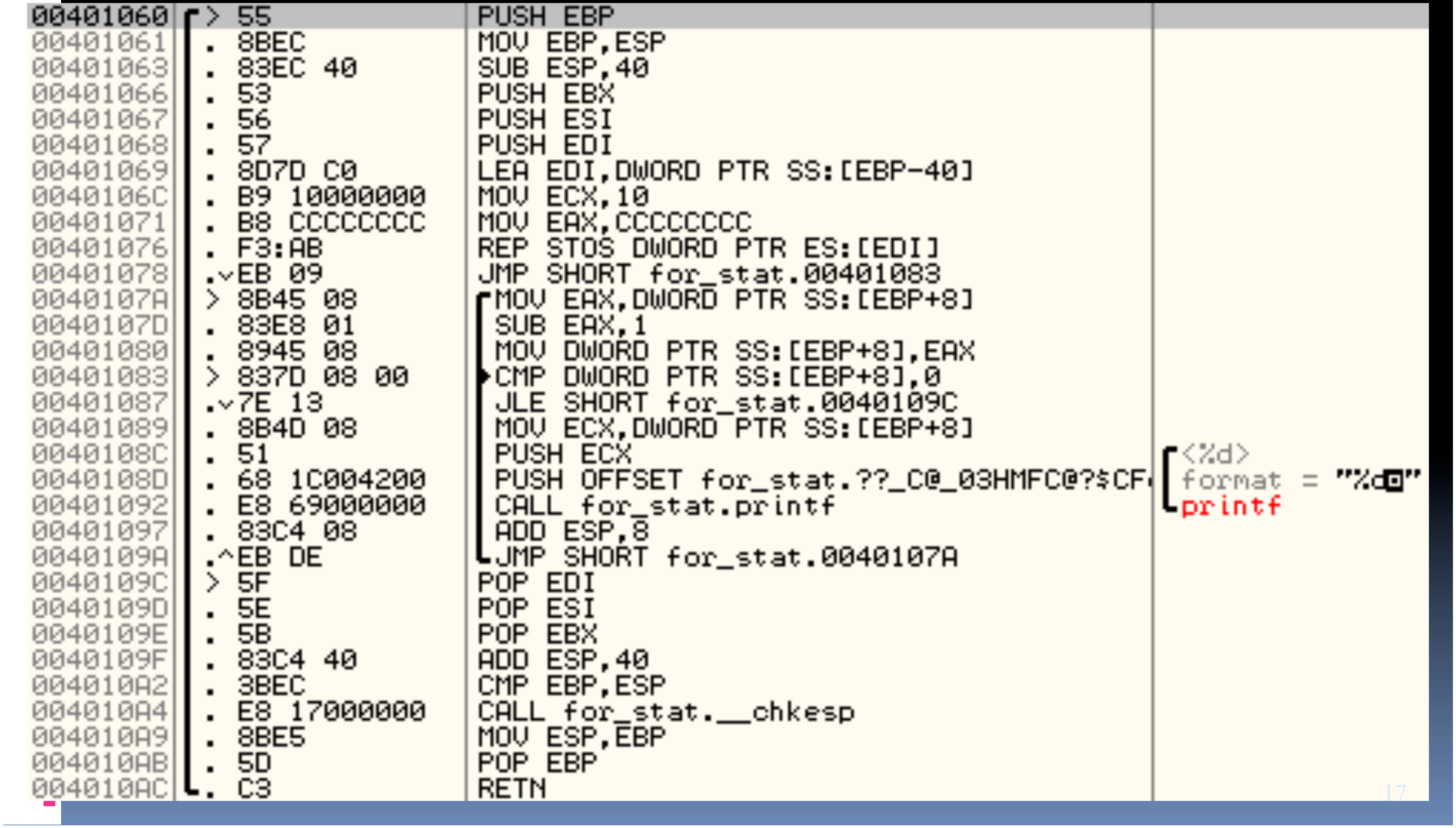
```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 44       SUB ESP,44
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D BC       LEA EDI,DWORD PTR SS:[EBP-44]
0040106C │. B9 11000000   MOV ECX,11
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │. 8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
0040107B │. 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
0040107E │. 837D FC 00    CMP DWORD PTR SS:[EBP-4],0
00401082 │.v7C 2A         JL SHORT switch_s.004010AE
00401084 │. 837D FC 01    CMP DWORD PTR SS:[EBP-4],1
00401088 │.v7E 08         JLE SHORT switch_s.00401092
0040108A │. 837D FC 02    CMP DWORD PTR SS:[EBP-4],2
0040108E │.v74 11         JE SHORT switch_s.004010A1
00401090 │.vEB 1C         JMP SHORT switch_s.004010AE
00401092 │> 68 24004200   PUSH OFFSET switch_s.??_C@_0N@OGF@Zero?! ┌format = "Zero or one□"
00401097 │. E8 84000000   CALL switch_s.printf                     └printf
0040109C │. 83C4 04       ADD ESP,4
0040109F │.vEB 0D         JMP SHORT switch_s.004010AE
004010A1 │> 68 1C004200   PUSH OFFSET switch_s.??_C@_04GGPI@Two?6  ┌format = "Two□"
004010A6 │. E8 75000000   CALL switch_s.printf                     └printf
004010AB │. 83C4 04       ADD ESP,4
004010AE │> 5F            POP EDI
004010AF │. 5E            POP ESI
004010B0 │. 5B            POP EBX
004010B1 │. 83C4 44       ADD ESP,44
004010B4 │. 3BEC          CMP EBP,ESP
004010B6 │. E8 25000000   CALL switch_s.__chkesp
004010BB │. 8BE5          MOV ESP,EBP
004010BD │. 5D            POP EBP
004010BE └. C3            RETN
```

For Statement



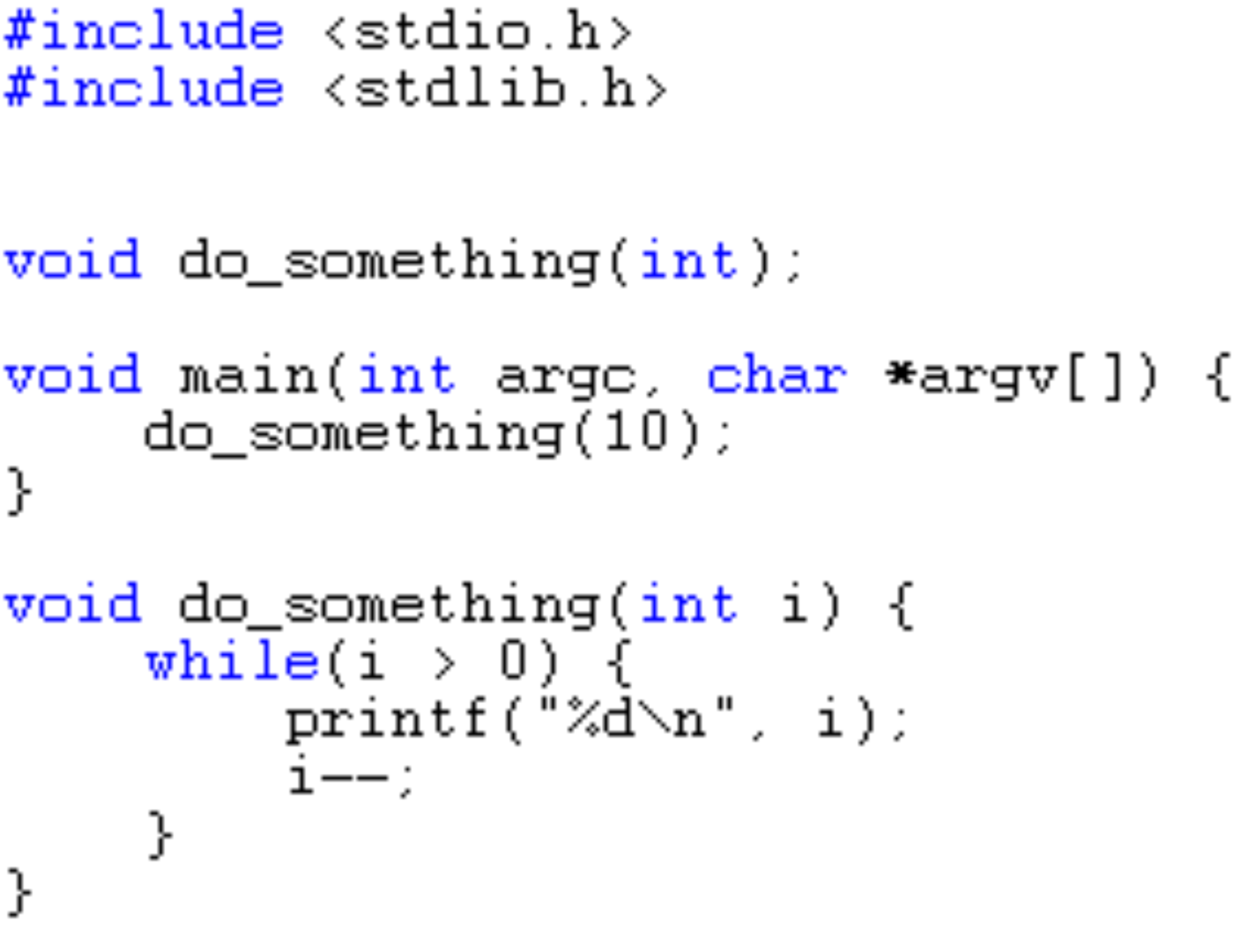
```
#include <stdio.h>
#include <stdlib.h>


void do_something(int);

void main(int argc, char *argv[]) {
    do_something(10);
}

void do_something(int i) {
    for(; i > 0; i--)
        printf("%d\n", i);
}
```

For Statement



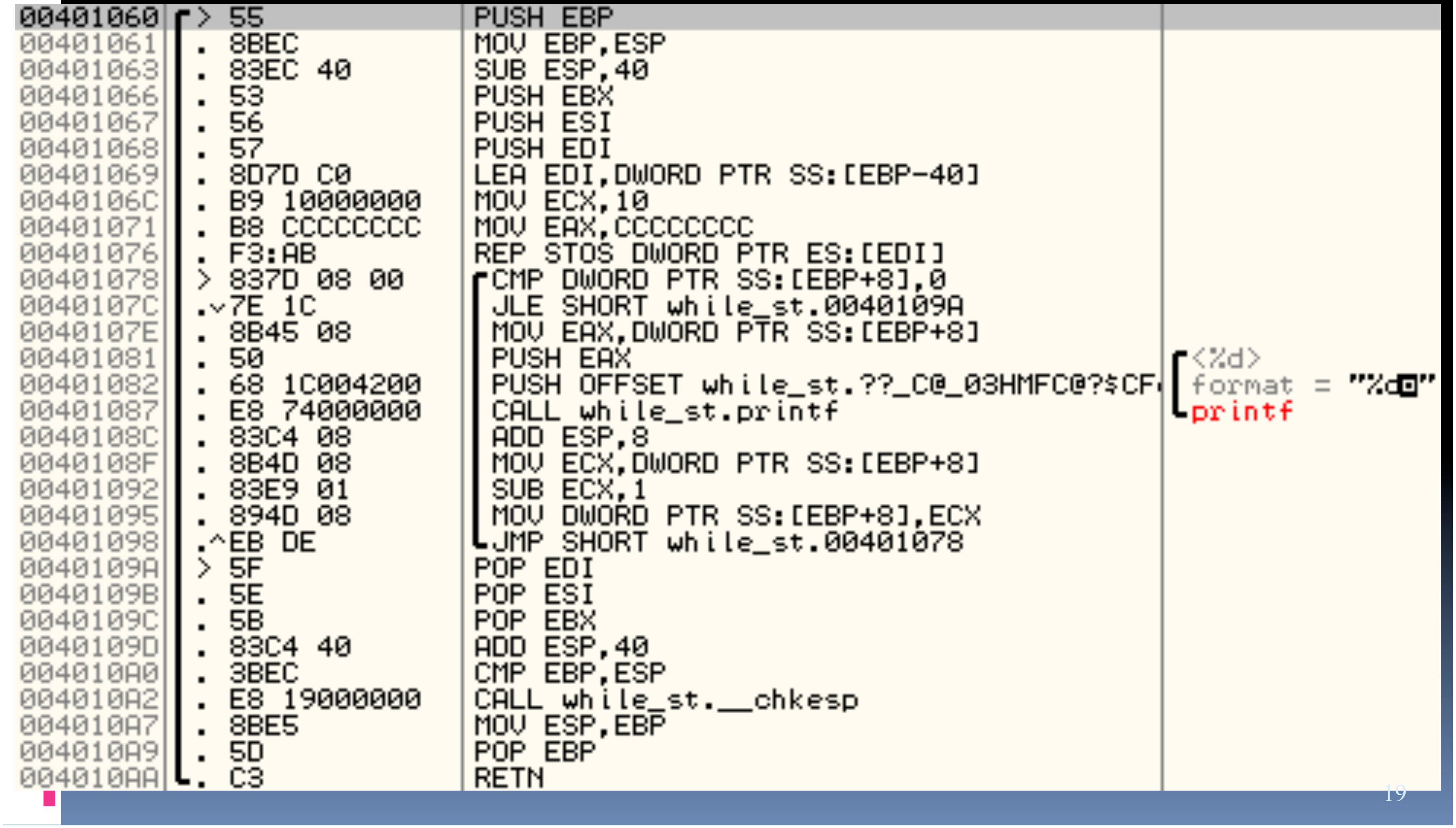
```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 40       SUB ESP,40
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D C0       LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000   MOV ECX,10
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │.vEB 09         JMP SHORT for_stat.00401083
0040107A │> 8B45 08       ┌MOV EAX,DWORD PTR SS:[EBP+8]
0040107D │. 83E8 01       │SUB EAX,1
00401080 │. 8945 08       │MOV DWORD PTR SS:[EBP+8],EAX
00401083 │> 837D 08 00    ▶CMP DWORD PTR SS:[EBP+8],0
00401087 │.v7E 13         │JLE SHORT for_stat.0040109C
00401089 │. 8B4D 08       │MOV ECX,DWORD PTR SS:[EBP+8]
0040108C │. 51            │PUSH ECX                                 ┌<%d>
0040108D │. 68 1C004200   │PUSH OFFSET for_stat.??_C@_03HMFC@?$CF  │format = "%d◘"
00401092 │. E8 69000000   │CALL for_stat.printf                     └printf
00401097 │. 83C4 08       │ADD ESP,8
0040109A │.^EB DE         └JMP SHORT for_stat.0040107A
0040109C │> 5F            POP EDI
0040109D │. 5E            POP ESI
0040109E │. 5B            POP EBX
0040109F │. 83C4 40       ADD ESP,40
004010A2 │. 3BEC          CMP EBP,ESP
004010A4 │. E8 17000000   CALL for_stat.__chkesp
004010A9 │. 8BE5          MOV ESP,EBP
004010AB │. 5D            POP EBP
004010AC └. C3            RETN
```

While Statement



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


void do_something(int);

void main(int argc, char *argv[]) {
    do_something(10);
}

void do_something(int i) {
    while(i > 0) {
        printf("%d\n", i);
        i--;
    }
}
```

While Statement



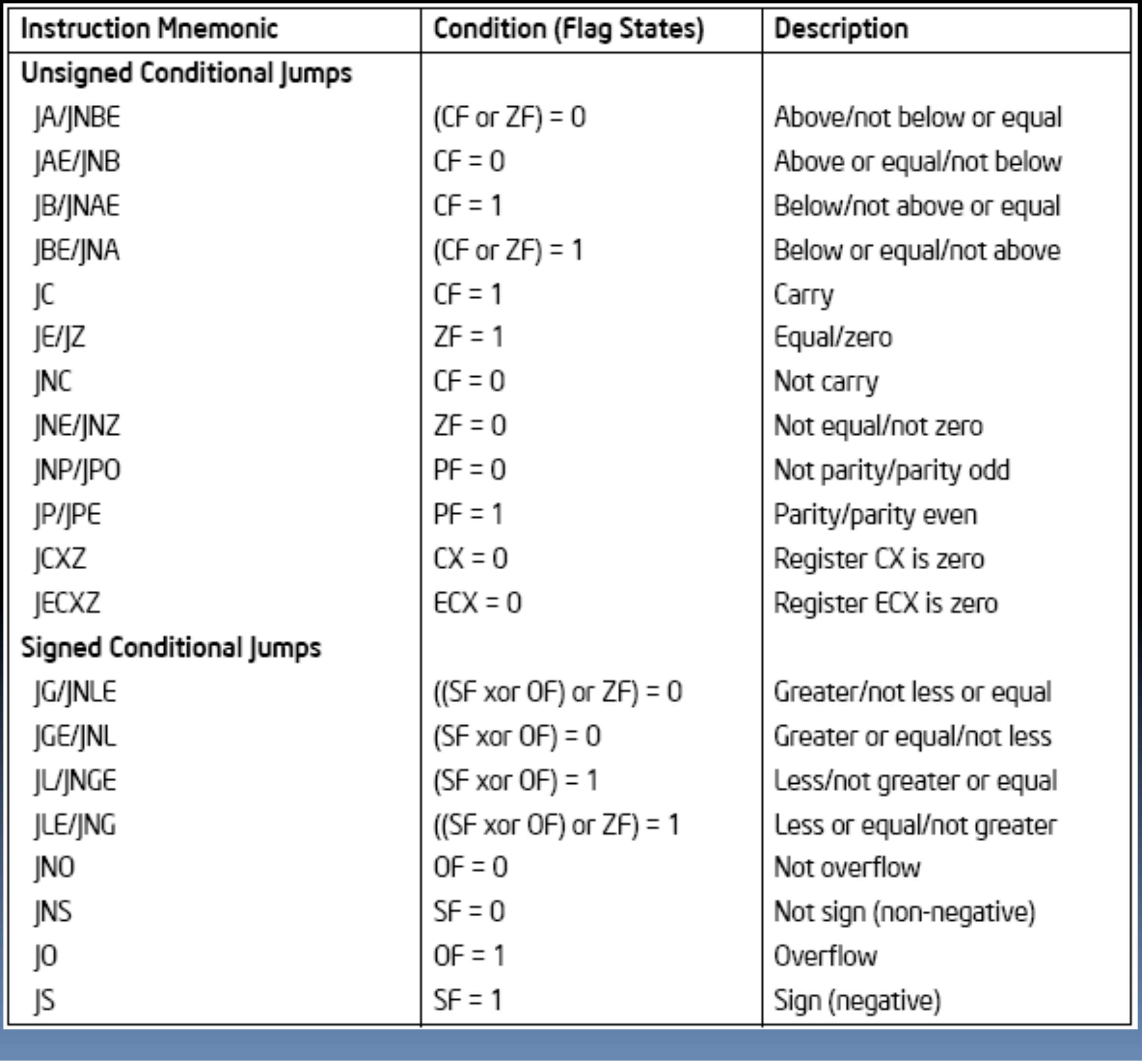
```
00401060  > 55              PUSH EBP
00401061  . 8BEC            MOV EBP,ESP
00401063  . 83EC 40         SUB ESP,40
00401066  . 53              PUSH EBX
00401067  . 56              PUSH ESI
00401068  . 57              PUSH EDI
00401069  . 8D7D C0         LEA EDI,DWORD PTR SS:[EBP-40]
0040106C  . B9 10000000     MOV ECX,10
00401071  . B8 CCCCCCCC     MOV EAX,CCCCCCCC
00401076  . F3:AB           REP STOS DWORD PTR ES:[EDI]
00401078  > 837D 08 00      CMP DWORD PTR SS:[EBP+8],0
0040107C  .v7E 1C           JLE SHORT while_st.0040109A
0040107E  . 8B45 08         MOV EAX,DWORD PTR SS:[EBP+8]
00401081  . 50              PUSH EAX                                    <%d>
00401082  . 68 1C004200     PUSH OFFSET while_st.??_C@_03HMFC@?$CF      format = "%d■"
00401087  . E8 74000000     CALL while_st.printf                        printf
0040108C  . 83C4 08         ADD ESP,8
0040108F  . 8B4D 08         MOV ECX,DWORD PTR SS:[EBP+8]
00401092  . 83E9 01         SUB ECX,1
00401095  . 894D 08         MOV DWORD PTR SS:[EBP+8],ECX
00401098  .^EB DE           JMP SHORT while_st.00401078
0040109A  > 5F              POP EDI
0040109B  . 5E              POP ESI
0040109C  . 5B              POP EBX
0040109D  . 83C4 40         ADD ESP,40
004010A0  . 3BEC            CMP EBP,ESP
004010A2  . E8 19000000     CALL while_st.__chkesp
004010A7  . 8BE5            MOV ESP,EBP
004010A9  . 5D              POP EBP
004010AA  . C3              RETN
```

]

Determining Signed-ness

- Signed and Unsigned Variables
  - Operations on signed/unsigned variables use different instructions
  - IMUL/MUL
  - IDIV/DIV
  - Jcc

Determining Signed-ness



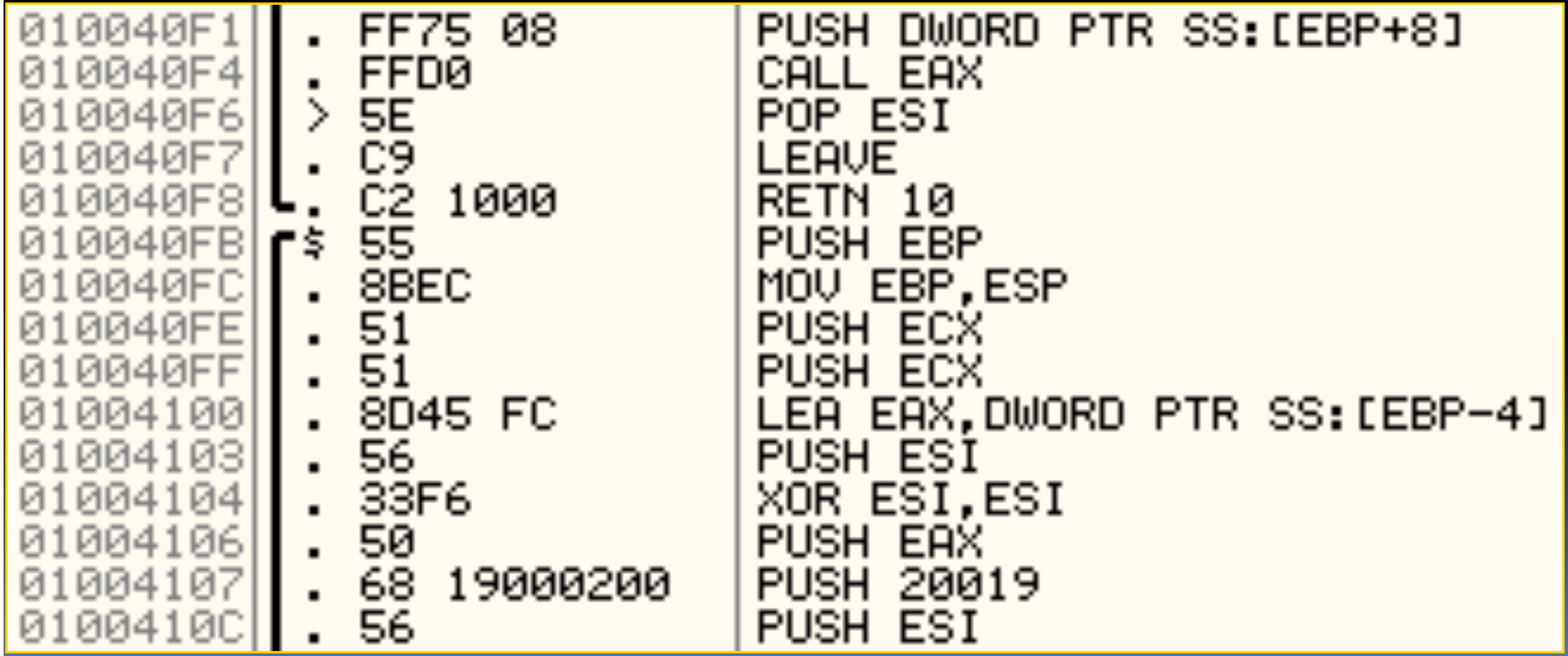
Instruction Mnemonic	Condition (Flag States)	Description
Unsigned Conditional Jumps		
JA/JNBE	(CF or ZF) = 0	Above/not below or equal
JAE/JNB	CF = 0	Above or equal/not below
JB/JNAE	CF = 1	Below/not above or equal
JBE/JNA	(CF or ZF) = 1	Below or equal/not above
JC	CF = 1	Carry
JE/JZ	ZF = 1	Equal/zero
JNC	CF = 0	Not carry
JNE/JNZ	ZF = 0	Not equal/not zero
JNP/JPO	PF = 0	Not parity/parity odd
JP/JPE	PF = 1	Parity/parity even
JCXZ	CX = 0	Register CX is zero
JECXZ	ECX = 0	Register ECX is zero
Signed Conditional Jumps		
JG/JNLE	((SF xor OF) or ZF) = 0	Greater/not less or equal
JGE/JNL	(SF xor OF) = 0	Greater or equal/not less
JL/JNGE	(SF xor OF) = 1	Less/not greater or equal
JLE/JNG	((SF xor OF) or ZF) = 1	Less or equal/not greater
JNO	OF = 0	Not overflow
JNS	SF = 0	Not sign (non-negative)
JO	OF = 1	Overflow
JS	SF = 1	Sign (negative)

Tools of the Trade

- Disassembler
  - Machine code to instructions
- Decompiler
  - Instructions to code (often to C code)
- Debugger
  - Real-time, step-thru-code debugging

Disassemblers

- Disassemblers
  - Converts machine code to instructions



```
010040F1  .  FF75 08        PUSH DWORD PTR SS:[EBP+8]
010040F4  .  FFD0           CALL EAX
010040F6  >  5E             POP ESI
010040F7  .  C9             LEAVE
010040F8  .  C2 1000        RETN 10
010040FB  $  55             PUSH EBP
010040FC  .  8BEC           MOV EBP,ESP
010040FE  .  51             PUSH ECX
010040FF  .  51             PUSH ECX
01004100  .  8D45 FC        LEA EAX,DWORD PTR SS:[EBP-4]
01004103  .  56             PUSH ESI
01004104  .  33F6           XOR ESI,ESI
01004106  .  50             PUSH EAX
01004107  .  68 19000200    PUSH 20019
0100410C  .  56             PUSH ESI
```

Decompilers

- Decompilers
  - Attempt to convert instructions or byte codes to higher-level languages
  - Good decompilers are implemented via p-code analysis
    - Allows decompiler code to be applied to various architectures as long as a p-code translation exists

Debuggers

- Debuggers
  - Modes
    - User-mode
    - Kernel-mode
  - Common features
    - Create/attach to a process
    - Set/clear breakpoint
    - Step into/over
    - Trace into/over

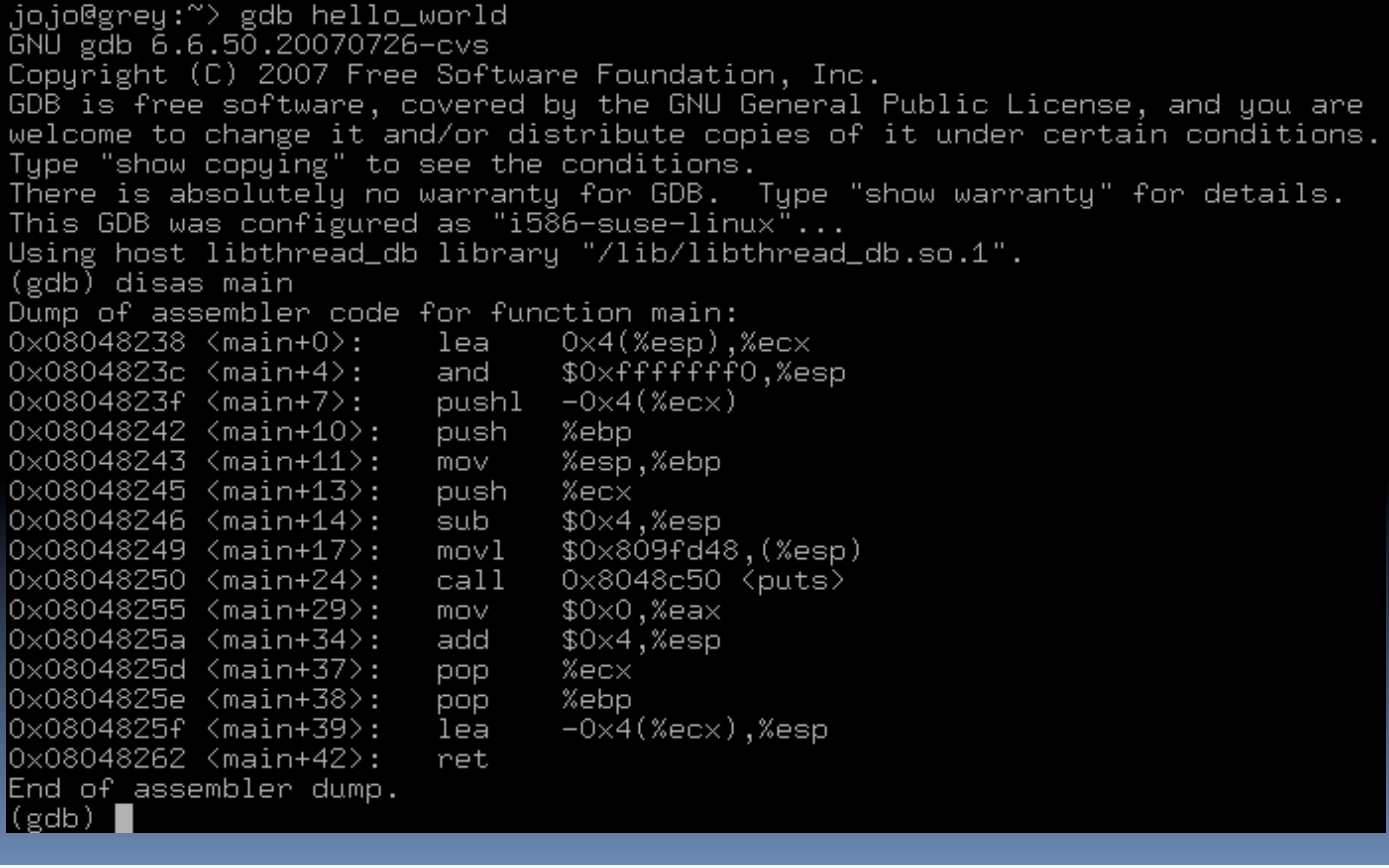
Debuggers

- Breakpoints
  - Software breakpoints
    - INT 3h (\xCC)
  - Memory breakpoints
  - Hardware breakpoints
    - Intel Dr0-Dr7 registers
- Traces
  - Records instructions and execution contexts
- Stepping
  - Step into/over

GNU Debugger (gdb)

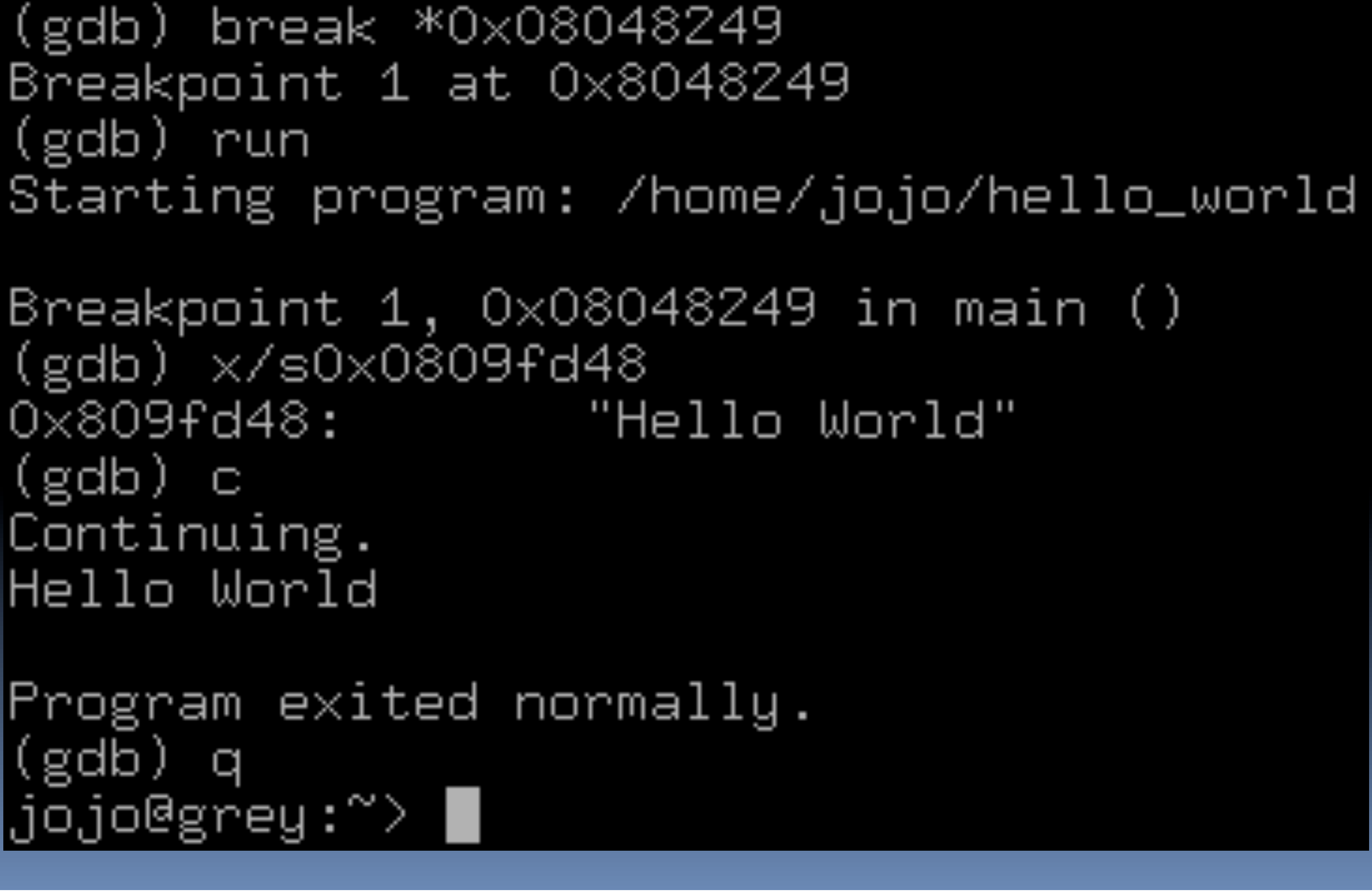
- Disassembler, Debugger
  - Command-line
    - Insight is a GUI wrapper for gdb
  - Not just for Linux
    - Native x86 Windows support
    - Special versions for various architectures

GNU Debugger (gdb) Breakpoint Tutorial



```
jojo@grey:~> gdb hello_world
GNU gdb 6.6.50.20070726-cvs
Copyright (C) 2007 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i586-suse-linux"...
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) disas main
Dump of assembler code for function main:
0x08048238 <main+0>:    lea    0x4(%esp),%ecx
0x0804823c <main+4>:    and    $0xfffffff0,%esp
0x0804823f <main+7>:    pushl  -0x4(%ecx)
0x08048242 <main+10>:   push   %ebp
0x08048243 <main+11>:   mov    %esp,%ebp
0x08048245 <main+13>:   push   %ecx
0x08048246 <main+14>:   sub    $0x4,%esp
0x08048249 <main+17>:   movl   $0x809fd48,(%esp)
0x08048250 <main+24>:   call   0x8048c50 <puts>
0x08048255 <main+29>:   mov    $0x0,%eax
0x0804825a <main+34>:   add    $0x4,%esp
0x0804825d <main+37>:   pop    %ecx
0x0804825e <main+38>:   pop    %ebp
0x0804825f <main+39>:   lea    -0x4(%ecx),%esp
0x08048262 <main+42>:   ret
End of assembler dump.
(gdb) 
```

GNU Debugger (gdb) Breakpoint Tutorial



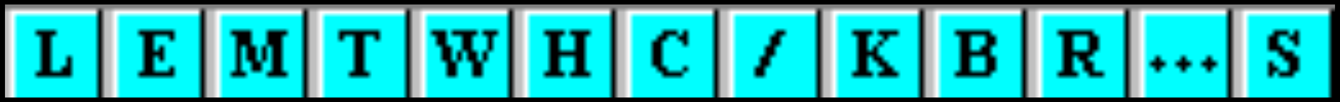
```
(gdb) break *0x08048249
Breakpoint 1 at 0x8048249
(gdb) run
Starting program: /home/jojo/hello_world

Breakpoint 1, 0x08048249 in main ()
(gdb) x/s0x0809fd48
0x809fd48:       "Hello World"
(gdb) c
Continuing.
Hello World

Program exited normally.
(gdb) q
jojo@grey:~> █
```

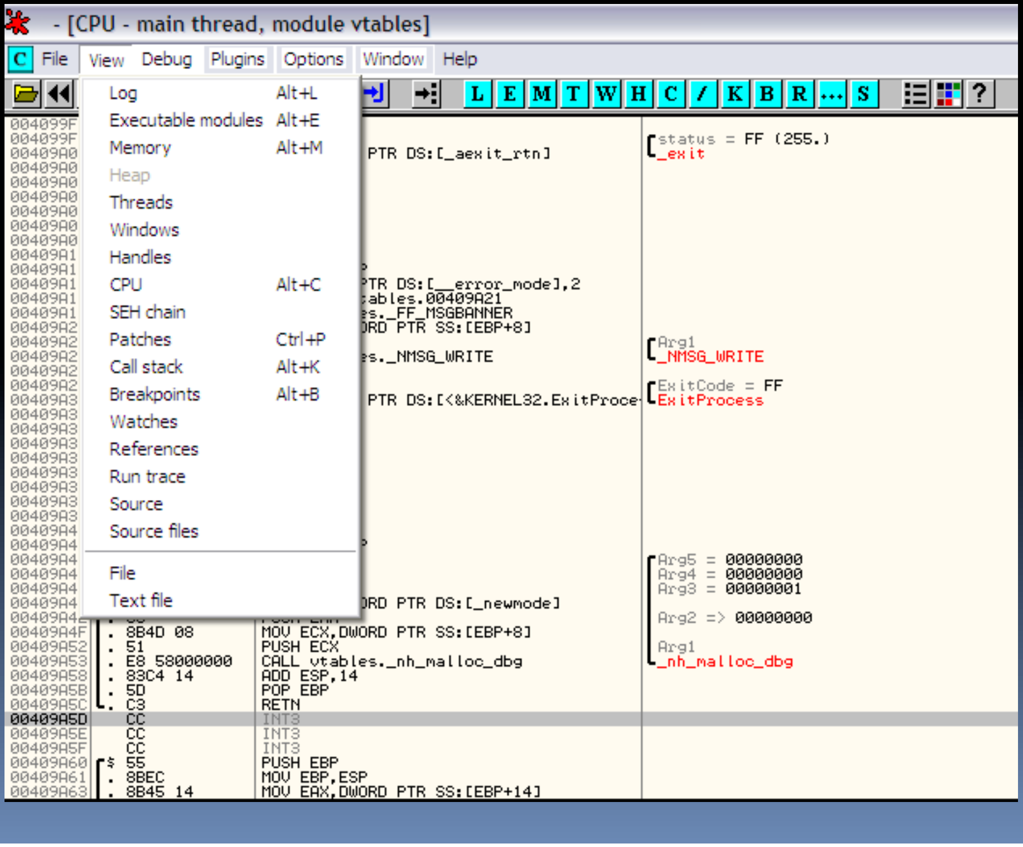
OllyDbg

- Disassembler
- Debugger
  - Open
    - Creates a process with debug privileges
  - Attach
    - Attach to a process already running
  - Detach (version 2.*)
    - Detaches the debugger and allows the process to continue
  - Terminate
    - Kills the debuggee

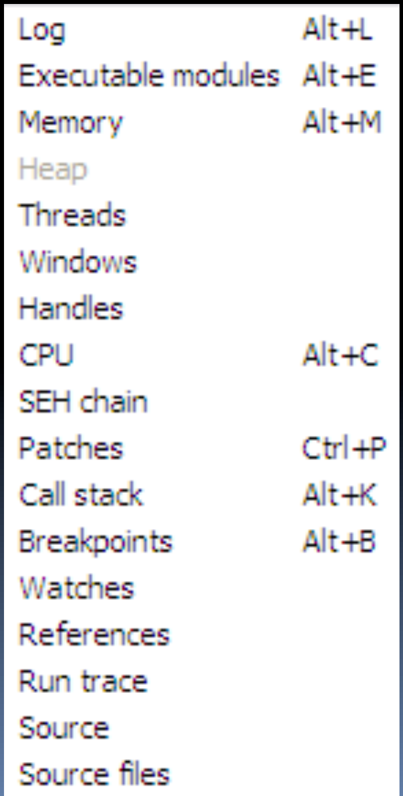
OllyDbg

- Views



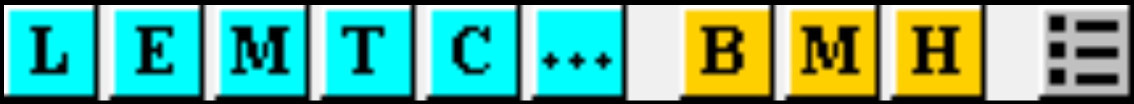


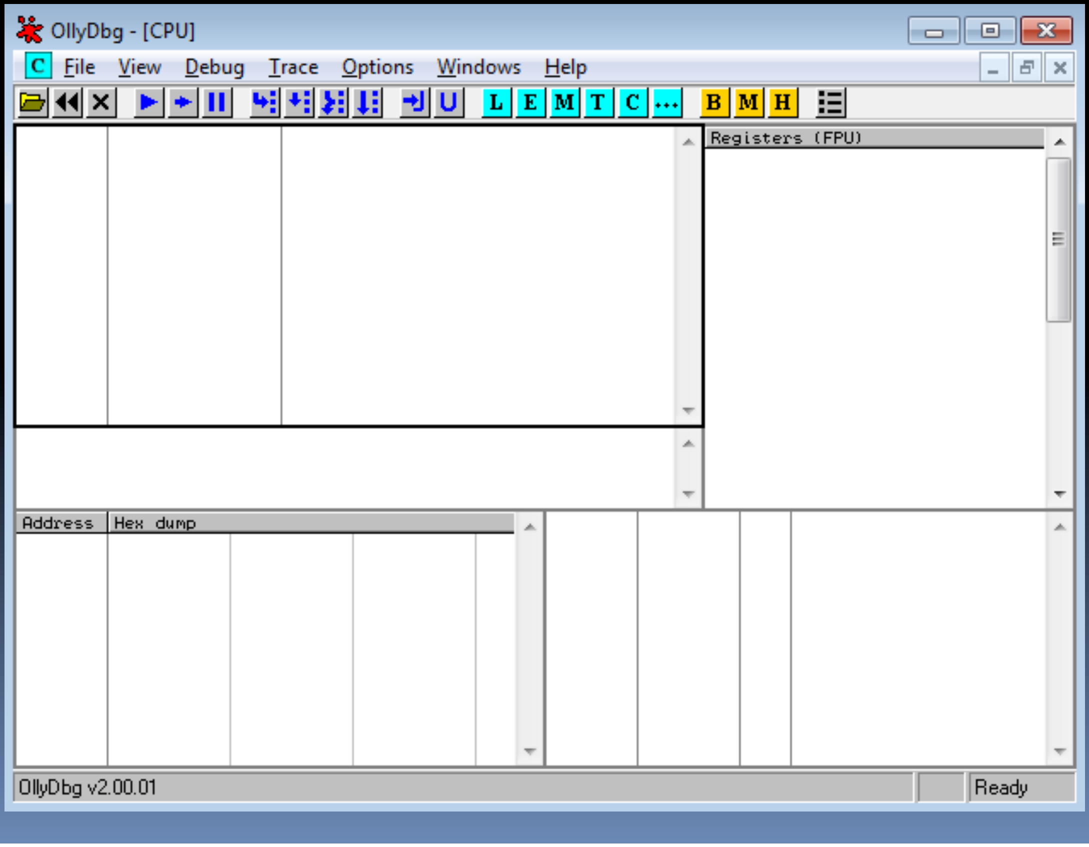
```
PTR DS:[_aexit_rtn]                    status = FF (255.)
                                       _exit
PTR DS:[__error_mode],2
ables.00409A21
es._FF_MSGBANNER
ORD PTR SS:[EBP+8]
                                       Arg1
es._NMSG_WRITE                         _NMSG_WRITE
                                       ExitCode = FF
PTR DS:[<&KERNEL32.ExitProce           ExitProcess

                                       Arg5 = 00000000
                                       Arg4 = 00000000
                                       Arg3 = 00000001
ORD PTR DS:[_newmode]
                                       Arg2 => 00000000
00409A4F  .  8B4D 08      MOV ECX,DWORD PTR SS:[EBP+8]
00409A52  .  51           PUSH ECX                  Arg1
00409A53  .  E8 58000000  CALL vtables._nh_malloc_dbg   _nh_malloc_dbg
00409A58  .  83C4 14      ADD ESP,14
00409A5B  .  5D           POP EBP
00409A5C  .  C3           RETN
00409A5D     CC           INT3
00409A5E     CC           INT3
00409A5F     CC           INT3
00409A60  $  55           PUSH EBP
00409A61  .  8BEC         MOV EBP,ESP
00409A63  .  8B45 14      MOV EAX,DWORD PTR SS:[EBP+14]
```

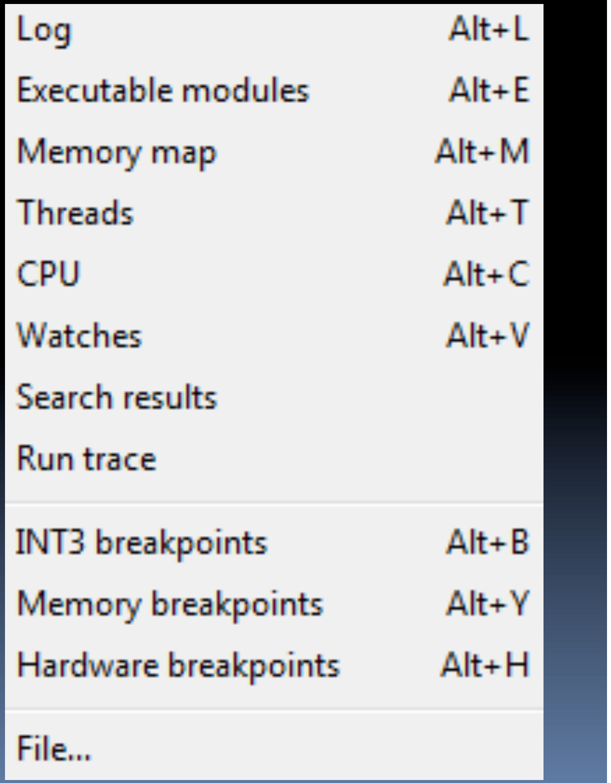


OllyDbg 2.0

- Views



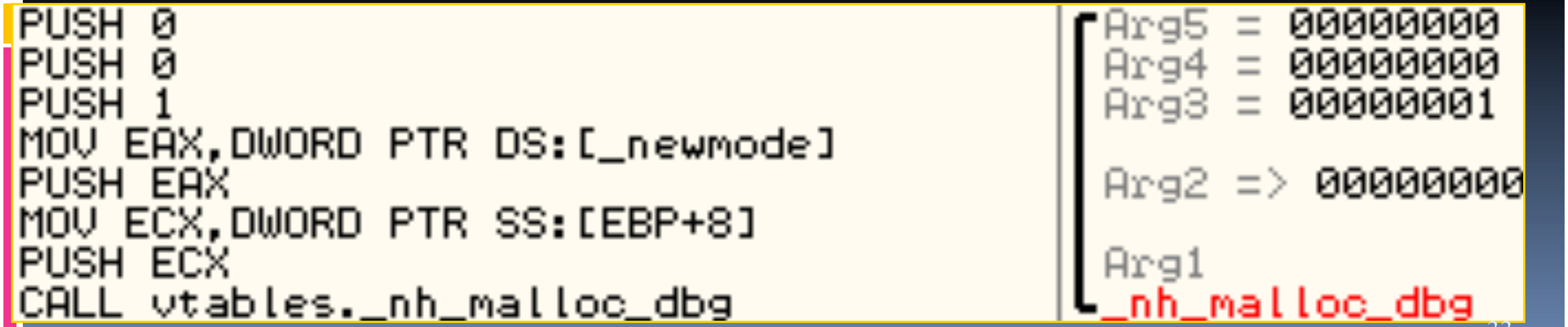




Log	Alt+L
Executable modules	Alt+E
Memory map	Alt+M
Threads	Alt+T
CPU	Alt+C
Watches	Alt+V
Search results	
Run trace	
INT3 breakpoints	Alt+B
Memory breakpoints	Alt+Y
Hardware breakpoints	Alt+H
File...	

OllyDbg

- Code Analysis
  - Right-click→Analysis→Analyse code (Ctrl + A)
  - Static code analysis
    - Argument labeling
    - Function address name resolution
    - Control logic labeling



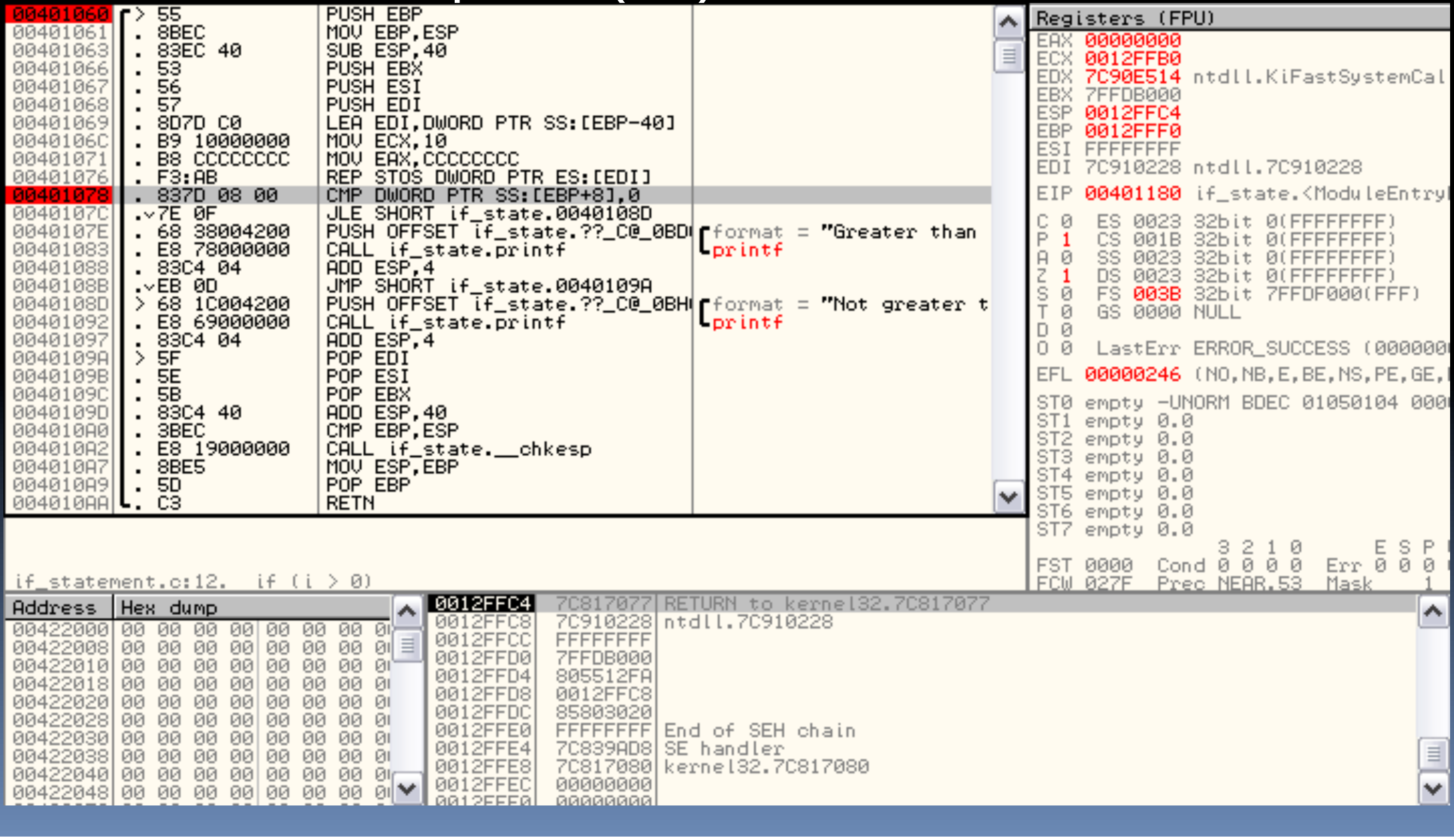
```
PUSH 0                                  ┌Arg5 = 00000000
PUSH 0                                  │Arg4 = 00000000
PUSH 1                                  │Arg3 = 00000001
MOV EAX,DWORD PTR DS:[_newmode]         │
PUSH EAX                                │Arg2 => 00000000
MOV ECX,DWORD PTR SS:[EBP+8]            │
PUSH ECX                                │Arg1
CALL vtables._nh_malloc_dbg             └_nh_malloc_dbg
```

OllyDbg

- Just-in-time Debugger
  - Options → Just-in-time debugging
  - Runs Olly when a fatal error occurs
- Plugins
  - Great feature
  - Well used by the reverse engineering community

OllyDbg Debugging Tutorial

- Breakpoints
  - Set a breakpoint (F2)



```
00401060 ┌> 55           PUSH EBP
00401061 │. 8BEC         MOV EBP,ESP
00401063 │. 83EC 40      SUB ESP,40
00401066 │. 53           PUSH EBX
00401067 │. 56           PUSH ESI
00401068 │. 57           PUSH EDI
00401069 │. 8D7D C0      LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000  MOV ECX,10
00401071 │. B8 CCCCCCCC  MOV EAX,CCCCCCCC
00401076 │. F3:AB        REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00   CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F        JLE SHORT if_state.0040108D
0040107E │. 68 38004200  PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000  CALL if_state.printf           └printf
00401088 │. 83C4 04      ADD ESP,4
0040108B │.vEB 0D        JMP SHORT if_state.0040109A
0040108D │> 68 1C004200  PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000  CALL if_state.printf           └printf
00401097 │. 83C4 04      ADD ESP,4
0040109A │> 5F           POP EDI
0040109B │. 5E           POP ESI
0040109C │. 5B           POP EBX
0040109D │. 83C4 40      ADD ESP,40
004010A0 │. 3BEC         CMP EBP,ESP
004010A2 │. E8 19000000  CALL if_state.__chkesp
004010A7 │. 8BE5         MOV ESP,EBP
004010A9 │. 5D           POP EBP
004010AA └. C3           RETN
```

```
if_statement.c:12.  if (i > 0)
```

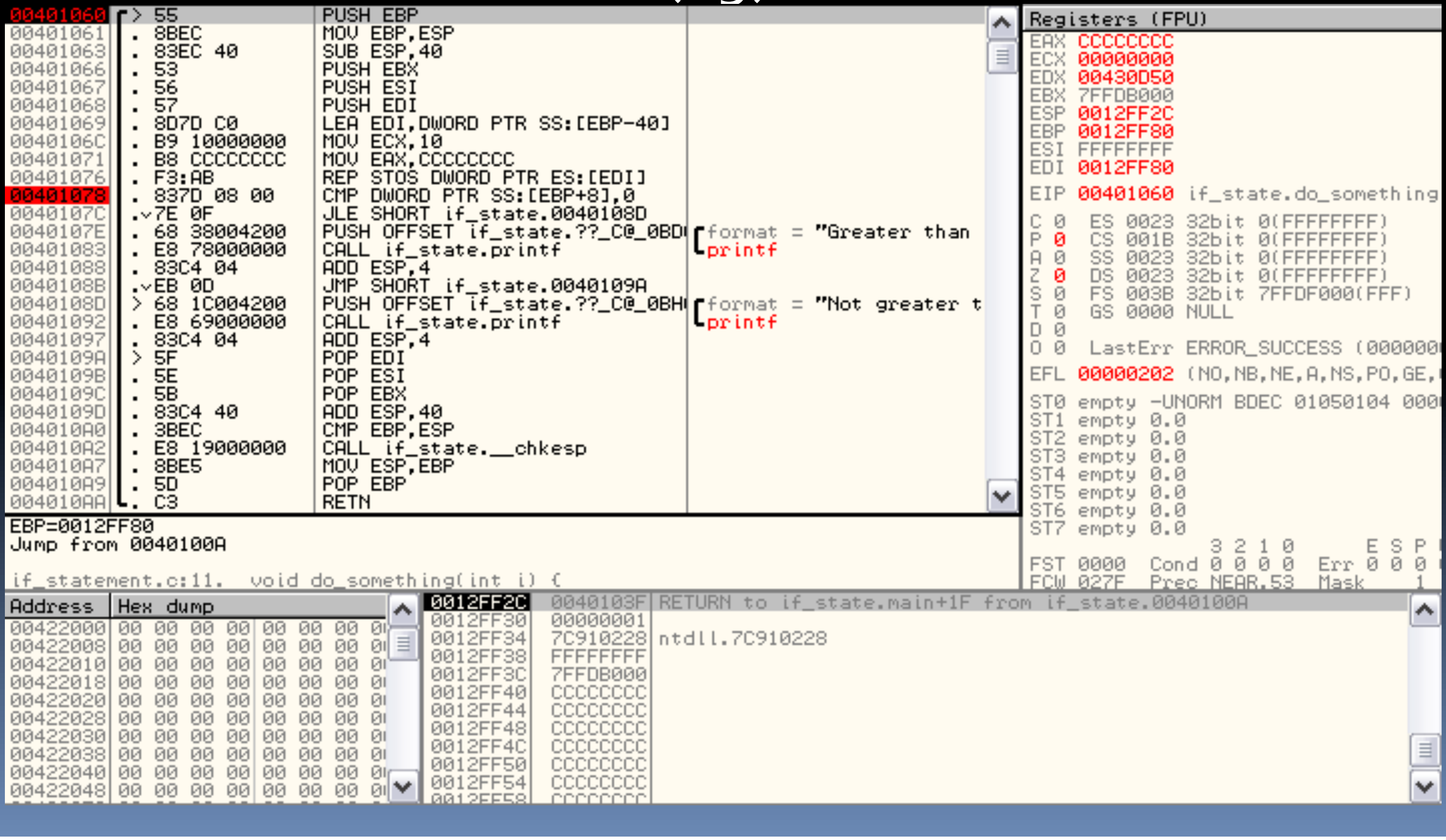
```
Registers (FPU)
EAX 00000000
ECX 0012FFB0
EDX 7C90E514 ntdll.KiFastSystemCal
EBX 7FFDB000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C910228 ntdll.7C910228

EIP 00401180 if_state.<ModuleEntry

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FFC4  7C817077 RETURN to kernel32.7C817077
0012FFC8  7C910228 ntdll.7C910228
0012FFCC  FFFFFFFF
0012FFD0  7FFDB000
0012FFD4  805512FA
0012FFD8  0012FFC8
0012FFDC  85803020
0012FFE0  FFFFFFFF End of SEH chain
0012FFE4  7C839AD8 SE handler
0012FFE8  7C817080 kernel32.7C817080
0012FFEC  00000000
0012FFF0  00000000
```

OllyDbg Debugging Tutorial

- Breakpoints
  - Resume execution (F9)



```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 40       SUB ESP,40
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D C0       LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000   MOV ECX,10
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00    CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F         JLE SHORT if_state.0040108D
0040107E │. 68 38004200   PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000   CALL if_state.printf           └printf
00401088 │. 83C4 04       ADD ESP,4
0040108B │.vEB 0D         JMP SHORT if_state.0040109A
0040108D │> 68 1C004200   PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000   CALL if_state.printf           └printf
00401097 │. 83C4 04       ADD ESP,4
0040109A │> 5F            POP EDI
0040109B │. 5E            POP ESI
0040109C │. 5B            POP EBX
0040109D │. 83C4 40       ADD ESP,40
004010A0 │. 3BEC          CMP EBP,ESP
004010A2 │. E8 19000000   CALL if_state.__chkesp
004010A7 │. 8BE5          MOV ESP,EBP
004010A9 │. 5D            POP EBP
004010AA └. C3            RETN
```

```
EBP=0012FF80
Jump from 0040100A

if_statement.c:11.  void do_something(int i) {
```

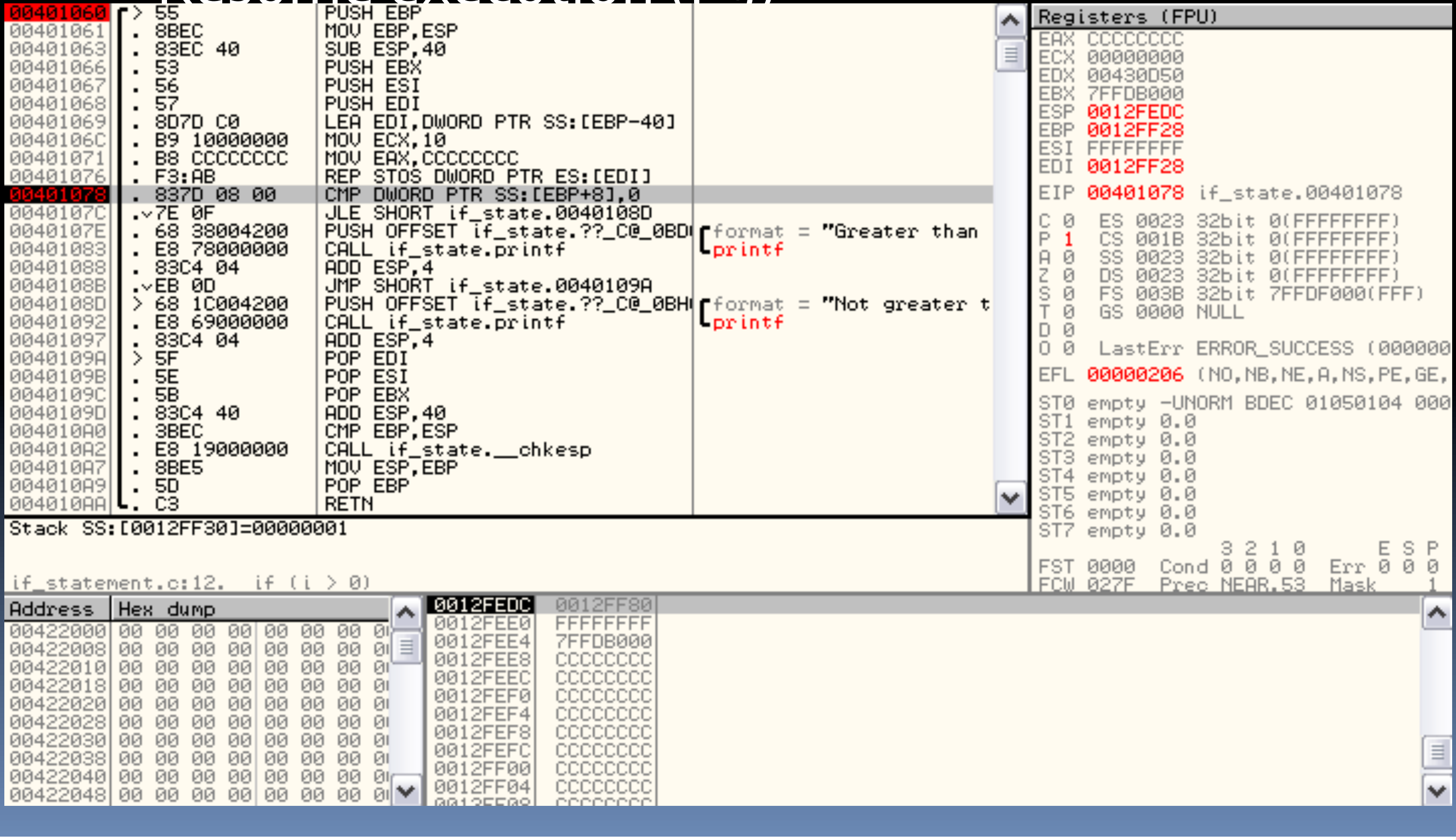
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FF2C
EBP 0012FF80
ESI FFFFFFFF
EDI 0012FF80

EIP 00401060 if_state.do_something

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FF2C  0040103F  RETURN to if_state.main+1F from if_state.0040100A
0012FF30  00000001
0012FF34  7C910228  ntdll.7C910228
0012FF38  FFFFFFFF
0012FF3C  7FFDB000
0012FF40  CCCCCCCC
0012FF44  CCCCCCCC
0012FF48  CCCCCCCC
0012FF4C  CCCCCCCC
0012FF50  CCCCCCCC
0012FF54  CCCCCCCC
0012FF58  CCCCCCCC
```

OllyDbg
Debugging Tutorial

- Breakpoints
  - Resume execution (F9)



```
00401060 ┌> 55          PUSH EBP
00401061 │. 8BEC        MOV EBP,ESP
00401063 │. 83EC 40     SUB ESP,40
00401066 │. 53          PUSH EBX
00401067 │. 56          PUSH ESI
00401068 │. 57          PUSH EDI
00401069 │. 8D7D C0     LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000 MOV ECX,10
00401071 │. B8 CCCCCCCC MOV EAX,CCCCCCCC
00401076 │. F3:AB       REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00  CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F       JLE SHORT if_state.0040108D
0040107E │. 68 38004200 PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000 CALL if_state.printf           └printf
00401088 │. 83C4 04     ADD ESP,4
0040108B │.vEB 0D       JMP SHORT if_state.0040109A
0040108D │> 68 1C004200 PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000 CALL if_state.printf           └printf
00401097 │. 83C4 04     ADD ESP,4
0040109A │> 5F          POP EDI
0040109B │. 5E          POP ESI
0040109C │. 5B          POP EBX
0040109D │. 83C4 40     ADD ESP,40
004010A0 │. 3BEC        CMP EBP,ESP
004010A2 │. E8 19000000 CALL if_state.__chkesp
004010A7 │. 8BE5        MOV ESP,EBP
004010A9 │. 5D          POP EBP
004010AA └. C3          RETN
```

```
Stack SS:[0012FF30]=00000001


if_statement.c:12.  if (i > 0)
```

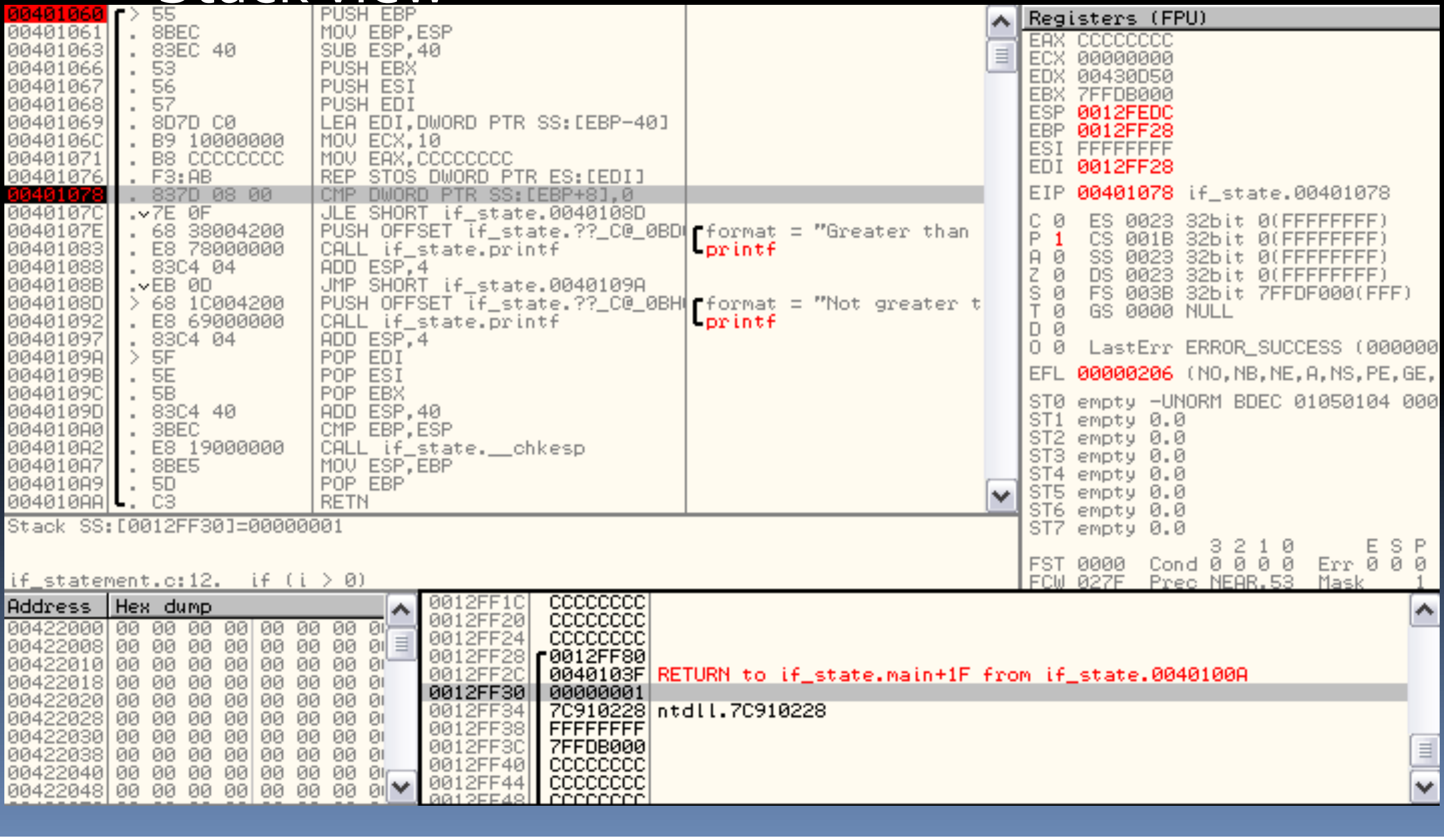
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FEDC
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 00401078 if_state.00401078

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FEDC   0012FF80
0012FEE0   FFFFFFFF
0012FEE4   7FFDB000
0012FEE8   CCCCCCCC
0012FEEC   CCCCCCCC
0012FEF0   CCCCCCCC
0012FEF4   CCCCCCCC
0012FEF8   CCCCCCCC
0012FEFC   CCCCCCCC
0012FF00   CCCCCCCC
0012FF04   CCCCCCCC
0012FF08   CCCCCCCC
```

OllyDbg
Debugging Tutorial

- Breakpoints
  - Stack view



```
00401060 ┌> 55             PUSH EBP
00401061 │. 8BEC           MOV EBP,ESP
00401063 │. 83EC 40        SUB ESP,40
00401066 │. 53             PUSH EBX
00401067 │. 56             PUSH ESI
00401068 │. 57             PUSH EDI
00401069 │. 8D7D C0        LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000    MOV ECX,10
00401071 │. B8 CCCCCCCC    MOV EAX,CCCCCCCC
00401076 │. F3:AB          REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00     CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F          JLE SHORT if_state.0040108D
0040107E │. 68 38004200    PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000    CALL if_state.printf           └printf
00401088 │. 83C4 04        ADD ESP,4
0040108B │.vEB 0D          JMP SHORT if_state.0040109A
0040108D │> 68 1C004200    PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000    CALL if_state.printf           └printf
00401097 │. 83C4 04        ADD ESP,4
0040109A │> 5F             POP EDI
0040109B │. 5E             POP ESI
0040109C │. 5B             POP EBX
0040109D │. 83C4 40        ADD ESP,40
004010A0 │. 3BEC           CMP EBP,ESP
004010A2 │. E8 19000000    CALL if_state.__chkesp
004010A7 │. 8BE5           MOV ESP,EBP
004010A9 │. 5D             POP EBP
004010AA └. C3             RETN
Stack SS:[0012FF30]=00000001

if_statement.c:12.  if (i > 0)
```

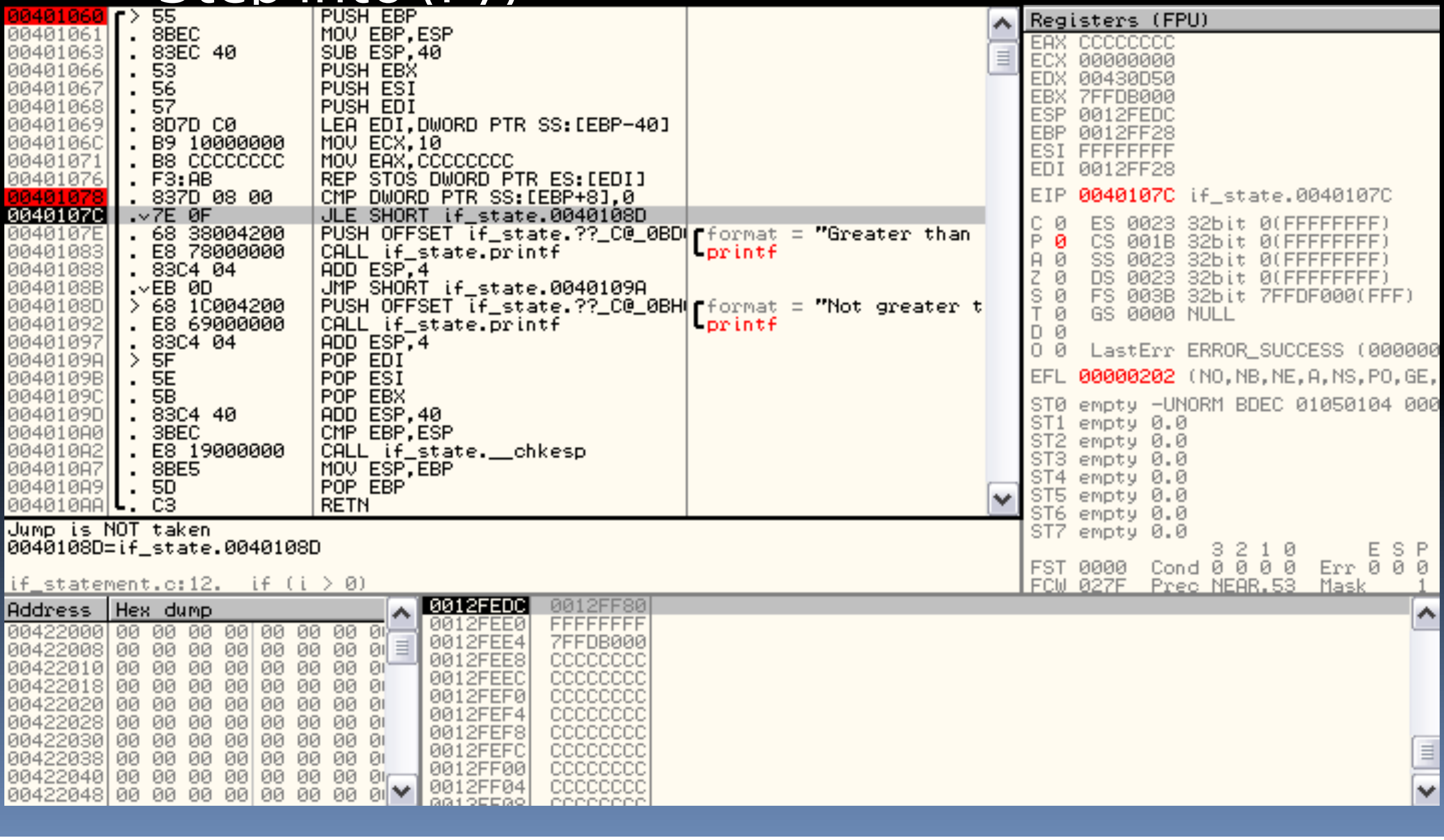
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FEDC
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 00401078 if_state.00401078

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0       E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask     1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FF1C  CCCCCCCC
0012FF20  CCCCCCCC
0012FF24  CCCCCCCC
0012FF28 ┌0012FF80
0012FF2C │0040103F RETURN to if_state.main+1F from if_state.0040100A
0012FF30 │00000001
0012FF34 │7C910228 ntdll.7C910228
0012FF38 │FFFFFFFF
0012FF3C │7FFDB000
0012FF40 │CCCCCCCC
0012FF44 │CCCCCCCC
0012FF48 │CCCCCCCC
```

OllyDbg Debugging Tutorial

- Stepping
  - Step into (F7)



```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 40       SUB ESP,40
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D C0       LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000   MOV ECX,10
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00    CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F         JLE SHORT if_state.0040108D
0040107E │. 68 38004200   PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000   CALL if_state.printf           └printf
00401088 │. 83C4 04       ADD ESP,4
0040108B │.vEB 0D         JMP SHORT if_state.0040109A
0040108D │> 68 1C004200   PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000   CALL if_state.printf           └printf
00401097 │. 83C4 04       ADD ESP,4
0040109A │> 5F            POP EDI
0040109B │. 5E            POP ESI
0040109C │. 5B            POP EBX
0040109D │. 83C4 40       ADD ESP,40
004010A0 │. 3BEC          CMP EBP,ESP
004010A2 │. E8 19000000   CALL if_state.__chkesp
004010A7 │. 8BE5          MOV ESP,EBP
004010A9 │. 5D            POP EBP
004010AA └. C3            RETN

Jump is NOT taken
0040108D=if_state.0040108D

if_statement.c:12.  if (i > 0)
```

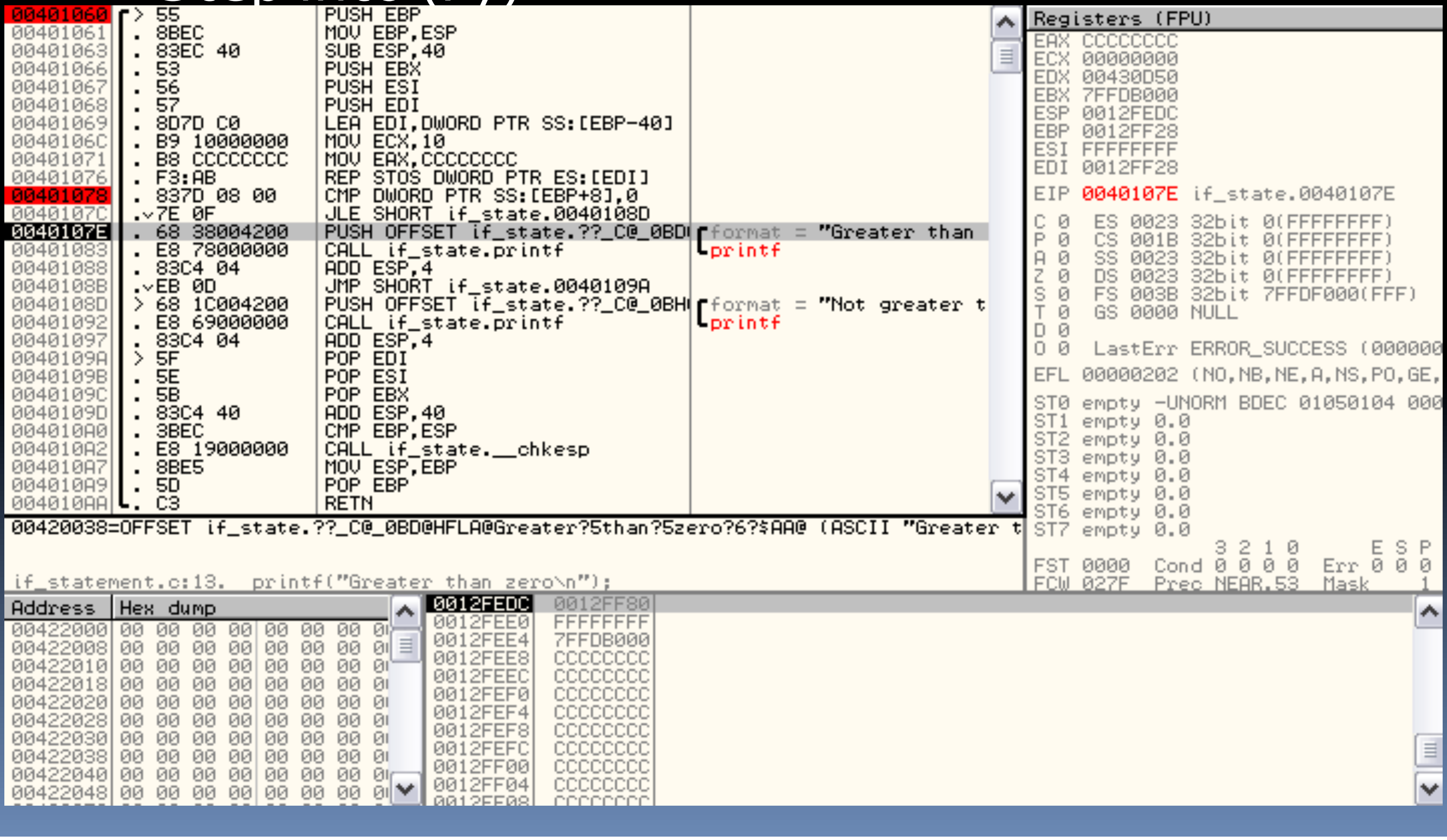
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FEDC
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 0040107C if_state.0040107C

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address   Hex dump
00422000  00 00 00 00 00 00 00 0
00422008  00 00 00 00 00 00 00 0
00422010  00 00 00 00 00 00 00 0
00422018  00 00 00 00 00 00 00 0
00422020  00 00 00 00 00 00 00 0
00422028  00 00 00 00 00 00 00 0
00422030  00 00 00 00 00 00 00 0
00422038  00 00 00 00 00 00 00 0
00422040  00 00 00 00 00 00 00 0
00422048  00 00 00 00 00 00 00 0
```

```
0012FEDC  0012FF80
0012FEE0  FFFFFFFF
0012FEE4  7FFDB000
0012FEE8  CCCCCCCC
0012FEEC  CCCCCCCC
0012FEF0  CCCCCCCC
0012FEF4  CCCCCCCC
0012FEF8  CCCCCCCC
0012FEFC  CCCCCCCC
0012FF00  CCCCCCCC
0012FF04  CCCCCCCC
0012FF08  CCCCCCCC
```

OllyDbg Debugging Tutorial

- Stepping
  - Step into (F7)



```
00401060 ┌> 55            PUSH EBP
00401061 │. 8BEC          MOV EBP,ESP
00401063 │. 83EC 40       SUB ESP,40
00401066 │. 53            PUSH EBX
00401067 │. 56            PUSH ESI
00401068 │. 57            PUSH EDI
00401069 │. 8D7D C0       LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000   MOV ECX,10
00401071 │. B8 CCCCCCCC   MOV EAX,CCCCCCCC
00401076 │. F3:AB         REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00    CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F         JLE SHORT if_state.0040108D
0040107E │. 68 38004200   PUSH OFFSET if_state.??_C@_0BD ┌format = "Greater than
00401083 │. E8 78000000   CALL if_state.printf           └printf
00401088 │. 83C4 04       ADD ESP,4
0040108B │.vEB 0D         JMP SHORT if_state.0040109A
0040108D │> 68 1C004200   PUSH OFFSET if_state.??_C@_0BH ┌format = "Not greater t
00401092 │. E8 69000000   CALL if_state.printf           └printf
00401097 │. 83C4 04       ADD ESP,4
0040109A │> 5F            POP EDI
0040109B │. 5E            POP ESI
0040109C │. 5B            POP EBX
0040109D │. 83C4 40       ADD ESP,40
004010A0 │. 3BEC          CMP EBP,ESP
004010A2 │. E8 19000000   CALL if_state.__chkesp
004010A7 │. 8BE5          MOV ESP,EBP
004010A9 │. 5D            POP EBP
004010AA └. C3            RETN
00420038=OFFSET if_state.??_C@_0BD@HFLA@Greater?5than?5zero?6?$AA@ (ASCII "Greater t

if_statement.c:13.  printf("Greater than zero\n");
```

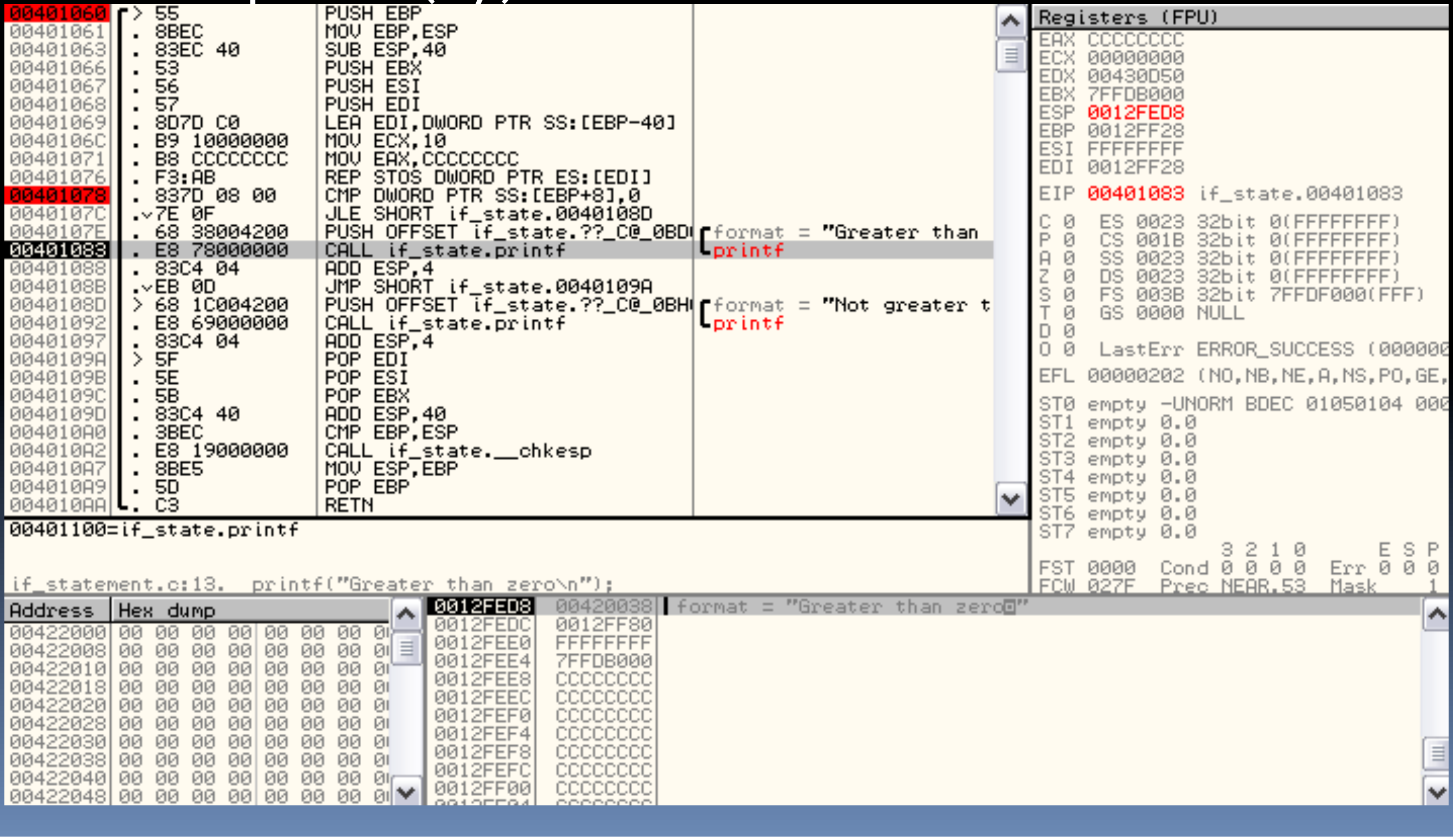
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FEDC
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 0040107E if_state.0040107E

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FEDC  0012FF80
0012FEE0  FFFFFFFF
0012FEE4  7FFDB000
0012FEE8  CCCCCCCC
0012FEEC  CCCCCCCC
0012FEF0  CCCCCCCC
0012FEF4  CCCCCCCC
0012FEF8  CCCCCCCC
0012FEFC  CCCCCCCC
0012FF00  CCCCCCCC
0012FF04  CCCCCCCC
```

OllyDbg Debugging Tutorial

- Stepping
  - Step into (F7)



```
00401060 ┌> 55              PUSH EBP
00401061 │. 8BEC            MOV EBP,ESP
00401063 │. 83EC 40         SUB ESP,40
00401066 │. 53              PUSH EBX
00401067 │. 56              PUSH ESI
00401068 │. 57              PUSH EDI
00401069 │. 8D7D C0         LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000     MOV ECX,10
00401071 │. B8 CCCCCCCC     MOV EAX,CCCCCCCC
00401076 │. F3:AB           REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00      CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F           JLE SHORT if_state.0040108D
0040107E │. 68 38004200     PUSH OFFSET if_state.??_C@_0BD  ┌format = "Greater than
00401083 │. E8 78000000     CALL if_state.printf            └printf
00401088 │. 83C4 04         ADD ESP,4
0040108B │.vEB 0D           JMP SHORT if_state.0040109A
0040108D │> 68 1C004200     PUSH OFFSET if_state.??_C@_0BH  ┌format = "Not greater t
00401092 │. E8 69000000     CALL if_state.printf            └printf
00401097 │. 83C4 04         ADD ESP,4
0040109A │> 5F              POP EDI
0040109B │. 5E              POP ESI
0040109C │. 5B              POP EBX
0040109D │. 83C4 40         ADD ESP,40
004010A0 │. 3BEC            CMP EBP,ESP
004010A2 │. E8 19000000     CALL if_state.__chkesp
004010A7 │. 8BE5            MOV ESP,EBP
004010A9 │. 5D              POP EBP
004010AA └. C3              RETN
00401100=if_state.printf

if_statement.c:13.  printf("Greater than zero\n");
```

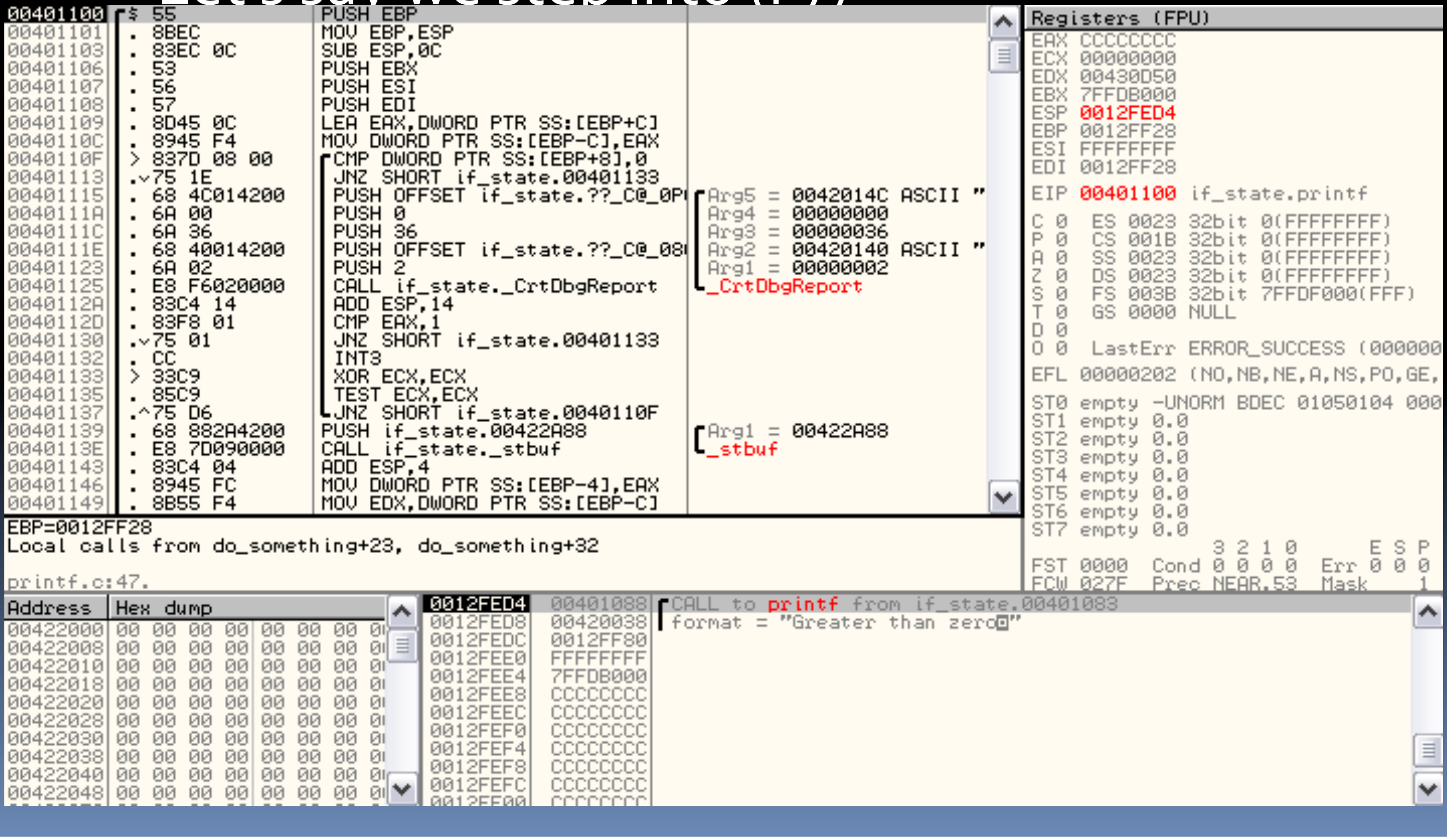
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FED8
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 00401083 if_state.00401083

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FED8  00420038  format = "Greater than zero◘"
0012FEDC  0012FF80
0012FEE0  FFFFFFFF
0012FEE4  7FFDB000
0012FEE8  CCCCCCCC
0012FEEC  CCCCCCCC
0012FEF0  CCCCCCCC
0012FEF4  CCCCCCCC
0012FEF8  CCCCCCCC
0012FEFC  CCCCCCCC
0012FF00  CCCCCCCC
```

OllyDbg Debugging Tutorial

- Stepping
  - Let’s say we step into (F7)



```
00401100 ┌$ 55            PUSH EBP
00401101 │. 8BEC          MOV EBP,ESP
00401103 │. 83EC 0C       SUB ESP,0C
00401106 │. 53            PUSH EBX
00401107 │. 56            PUSH ESI
00401108 │. 57            PUSH EDI
00401109 │. 8D45 0C       LEA EAX,DWORD PTR SS:[EBP+C]
0040110C │. 8945 F4       MOV DWORD PTR SS:[EBP-C],EAX
0040110F │> 837D 08 00    ┌CMP DWORD PTR SS:[EBP+8],0
00401113 │.v75 1E         │JNZ SHORT if_state.00401133
00401115 │. 68 4C014200   │PUSH OFFSET if_state.??_C@_0P ┌Arg5 = 0042014C ASCII "
0040111A │. 6A 00         │PUSH 0                        │Arg4 = 00000000
0040111C │. 6A 36         │PUSH 36                       │Arg3 = 00000036
0040111E │. 68 40014200   │PUSH OFFSET if_state.??_C@_08 │Arg2 = 00420140 ASCII "
00401123 │. 6A 02         │PUSH 2                        │Arg1 = 00000002
00401125 │. E8 F6020000   │CALL if_state._CrtDbgReport   └_CrtDbgReport
0040112A │. 83C4 14       │ADD ESP,14
0040112D │. 83F8 01       │CMP EAX,1
00401130 │.v75 01         │JNZ SHORT if_state.00401133
00401132 │. CC            │INT3
00401133 │> 33C9          │XOR ECX,ECX
00401135 │. 85C9          │TEST ECX,ECX
00401137 │.^75 D6         └JNZ SHORT if_state.0040110F
00401139 │. 68 882A4200   PUSH if_state.00422A88        ┌Arg1 = 00422A88
0040113E │. E8 7D090000   CALL if_state._stbuf          └_stbuf
00401143 │. 83C4 04       ADD ESP,4
00401146 │. 8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
00401149 │. 8B55 F4       MOV EDX,DWORD PTR SS:[EBP-C]
```

```
EBP=0012FF28
Local calls from do_something+23, do_something+32

printf.c:47.
```

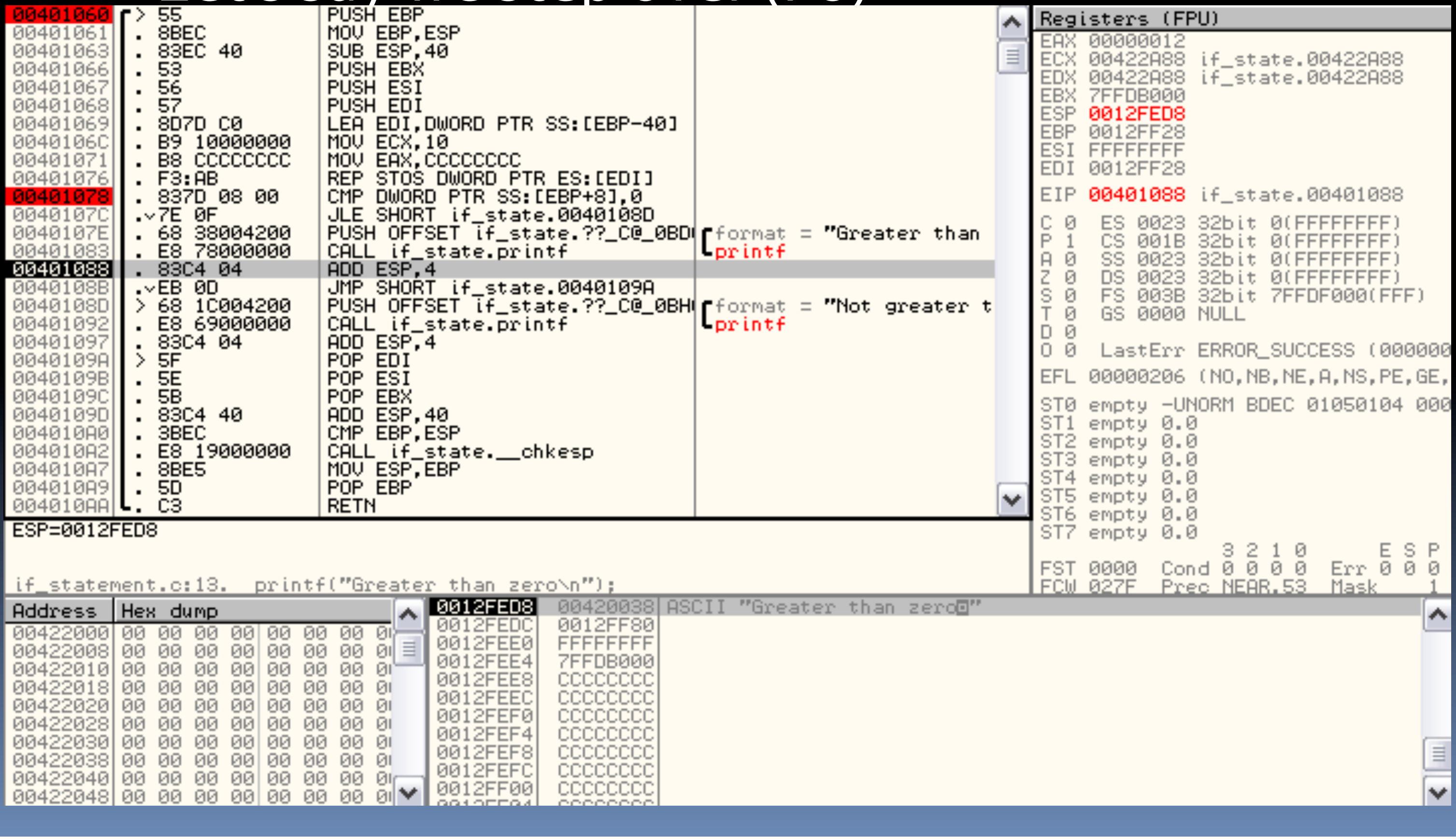
```
Registers (FPU)
EAX CCCCCCCC
ECX 00000000
EDX 00430D50
EBX 7FFDB000
ESP 0012FED4
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 00401100 if_state.printf

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FED4  00401088 ┌CALL to printf from if_state.00401083
0012FED8  00420038 │format = "Greater than zero◘"
0012FEDC  0012FF80
0012FEE0  FFFFFFFF
0012FEE4  7FFDB000
0012FEE8  CCCCCCCC
0012FEEC  CCCCCCCC
0012FEF0  CCCCCCCC
0012FEF4  CCCCCCCC
0012FEF8  CCCCCCCC
0012FEFC  CCCCCCCC
0012FF00  CCCCCCCC
```

OllyDbg Debugging Tutorial

- Stepping
  - Let’s say we step over (F8)



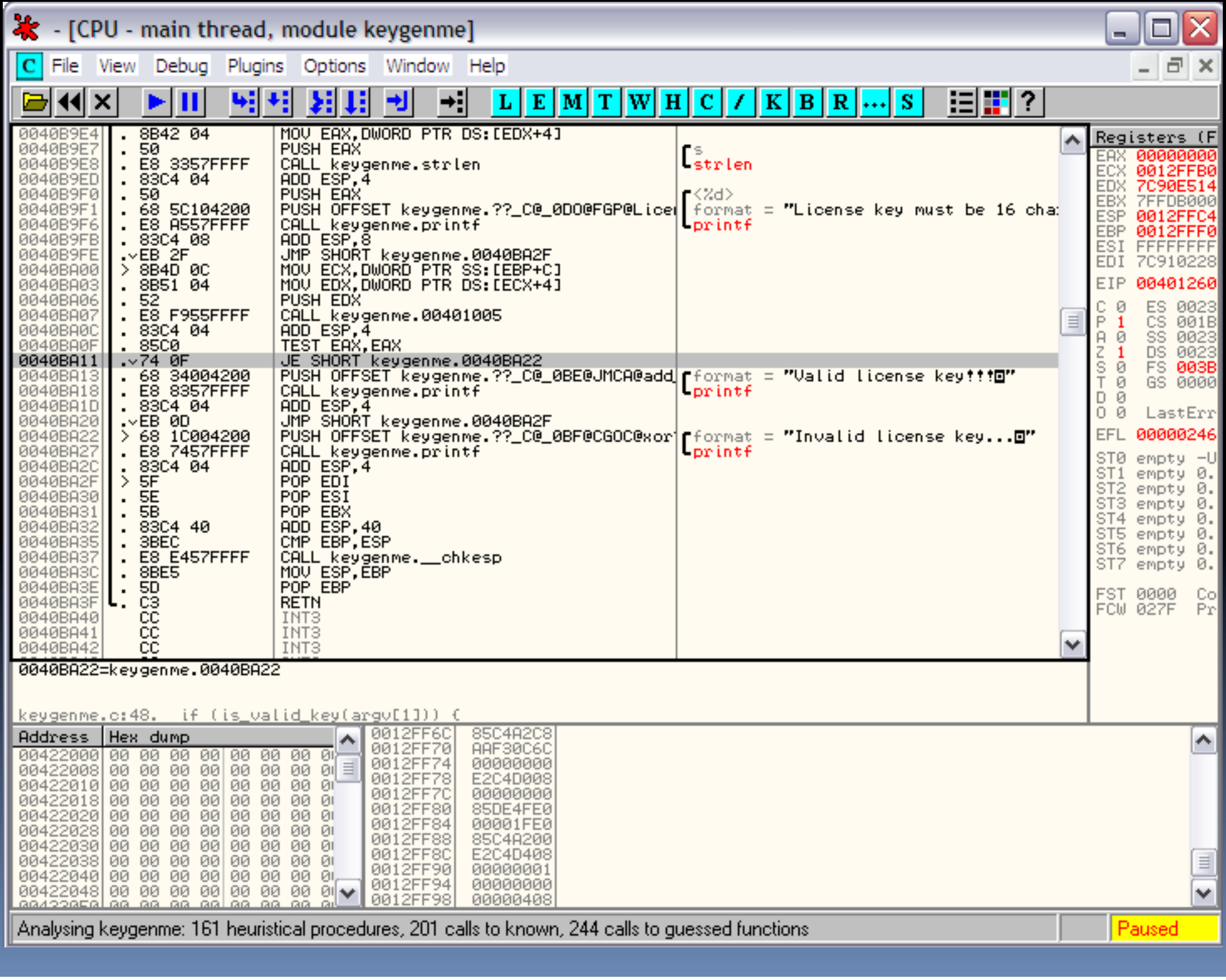
```
00401060 ┌> 55             PUSH EBP
00401061 │. 8BEC           MOV EBP,ESP
00401063 │. 83EC 40        SUB ESP,40
00401066 │. 53             PUSH EBX
00401067 │. 56             PUSH ESI
00401068 │. 57             PUSH EDI
00401069 │. 8D7D C0        LEA EDI,DWORD PTR SS:[EBP-40]
0040106C │. B9 10000000    MOV ECX,10
00401071 │. B8 CCCCCCCC    MOV EAX,CCCCCCCC
00401076 │. F3:AB          REP STOS DWORD PTR ES:[EDI]
00401078 │. 837D 08 00     CMP DWORD PTR SS:[EBP+8],0
0040107C │.v7E 0F          JLE SHORT if_state.0040108D
0040107E │. 68 38004200    PUSH OFFSET if_state.??_C@_0BD( ┌format = "Greater than
00401083 │. E8 78000000    CALL if_state.printf             └printf
00401088 │. 83C4 04        ADD ESP,4
0040108B │.vEB 0D          JMP SHORT if_state.0040109A
0040108D │> 68 1C004200    PUSH OFFSET if_state.??_C@_0BH( ┌format = "Not greater t
00401092 │. E8 69000000    CALL if_state.printf             └printf
00401097 │. 83C4 04        ADD ESP,4
0040109A │> 5F             POP EDI
0040109B │. 5E             POP ESI
0040109C │. 5B             POP EBX
0040109D │. 83C4 40        ADD ESP,40
004010A0 │. 3BEC           CMP EBP,ESP
004010A2 │. E8 19000000    CALL if_state.__chkesp
004010A7 │. 8BE5           MOV ESP,EBP
004010A9 │. 5D             POP EBP
004010AA └. C3             RETN
ESP=0012FED8

if_statement.c:13.  printf("Greater than zero\n");
```

```
Registers (FPU)
EAX 00000012
ECX 00422A88 if_state.00422A88
EDX 00422A88 if_state.00422A88
EBX 7FFDB000
ESP 0012FED8
EBP 0012FF28
ESI FFFFFFFF
EDI 0012FF28

EIP 00401088 if_state.00401088

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (000000

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,

ST0 empty -UNORM BDEC 01050104 000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P
FST 0000  Cond 0 0 0 0  Err 0 0 0
FCW 027F  Prec NEAR,53  Mask    1
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

```
0012FED8  00420038  ASCII "Greater than zero◘"
0012FEDC  0012FF80
0012FEE0  FFFFFFFF
0012FEE4  7FFDB000
0012FEE8  CCCCCCCC
0012FEEC  CCCCCCCC
0012FEF0  CCCCCCCC
0012FEF4  CCCCCCCC
0012FEF8  CCCCCCCC
0012FEFC  CCCCCCCC
0012FF00  CCCCCCCC
```

OllyDbg
Assembly Patching Tutorial



```
0040B9E4  . 8B42 04        MOV EAX,DWORD PTR DS:[EDX+4]
0040B9E7  . 50             PUSH EAX                                  [s
0040B9E8  . E8 3357FFFF    CALL keygenme.strlen                      [strlen
0040B9ED  . 83C4 04        ADD ESP,4
0040B9F0  . 50             PUSH EAX                                  [<%d>
0040B9F1  . 68 5C104200    PUSH OFFSET keygenme.??_C@_0DO@FGP@Lice|  |format = "License key must be 16 cha
0040B9F6  . E8 A557FFFF    CALL keygenme.printf                      [printf
0040B9FB  . 83C4 08        ADD ESP,8
0040B9FE  .vEB 2F          JMP SHORT keygenme.0040BA2F
0040BA00  > 8B4D 0C        MOV ECX,DWORD PTR SS:[EBP+C]
0040BA03  . 8B51 04        MOV EDX,DWORD PTR DS:[ECX+4]
0040BA06  . 52             PUSH EDX
0040BA07  . E8 F955FFFF    CALL keygenme.00401005
0040BA0C  . 83C4 04        ADD ESP,4
0040BA0F  . 85C0           TEST EAX,EAX
0040BA11  .v74 0F          JE SHORT keygenme.0040BA22
0040BA13  . 68 34004200    PUSH OFFSET keygenme.??_C@_0BE@JMCA@add_  [format = "Valid license key!!!◘"
0040BA18  . E8 8357FFFF    CALL keygenme.printf                      [printf
0040BA1D  . 83C4 04        ADD ESP,4
0040BA20  .vEB 0D          JMP SHORT keygenme.0040BA2F
0040BA22  > 68 1C004200    PUSH OFFSET keygenme.??_C@_0BF@CGOC@xor'  [format = "Invalid license key...◘"
0040BA27  . E8 7457FFFF    CALL keygenme.printf                      [printf
0040BA2C  . 83C4 04        ADD ESP,4
0040BA2F  > 5F             POP EDI
0040BA30  . 5E             POP ESI
0040BA31  . 5B             POP EBX
0040BA32  . 83C4 40        ADD ESP,40
0040BA35  . 3BEC           CMP EBP,ESP
0040BA37  . E8 E457FFFF    CALL keygenme.__chkesp
0040BA3C  . 8BE5           MOV ESP,EBP
0040BA3E  . 5D             POP EBP
0040BA3F  . C3             RETN
0040BA40    CC             INT3
0040BA41    CC             INT3
0040BA42    CC             INT3
```

```
0040BA22=keygenme.0040BA22

keygenme.c:48.  if (is_valid_key(argv[1])) {
```

```
Registers (F
EAX 00000000
ECX 0012FFB0
EDX 7C90E514
EBX 7FFDB000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C910228

EIP 00401260

C 0  ES 0023
P 1  CS 001B
A 0  SS 0023
Z 1  DS 0023
S 0  FS 003B
T 0  GS 0000
D 0
O 0  LastErr

EFL 00000246

ST0 empty -U
ST1 empty 0.
ST2 empty 0.
ST3 empty 0.
ST4 empty 0.
ST5 empty 0.
ST6 empty 0.
ST7 empty 0.

FST 0000  Co
FCW 027F  Pr
```

```
Address  Hex dump
00422000 00 00 00 00 00 00 00 0
00422008 00 00 00 00 00 00 00 0
00422010 00 00 00 00 00 00 00 0
00422018 00 00 00 00 00 00 00 0
00422020 00 00 00 00 00 00 00 0
00422028 00 00 00 00 00 00 00 0
00422030 00 00 00 00 00 00 00 0
00422038 00 00 00 00 00 00 00 0
00422040 00 00 00 00 00 00 00 0
00422048 00 00 00 00 00 00 00 0
```

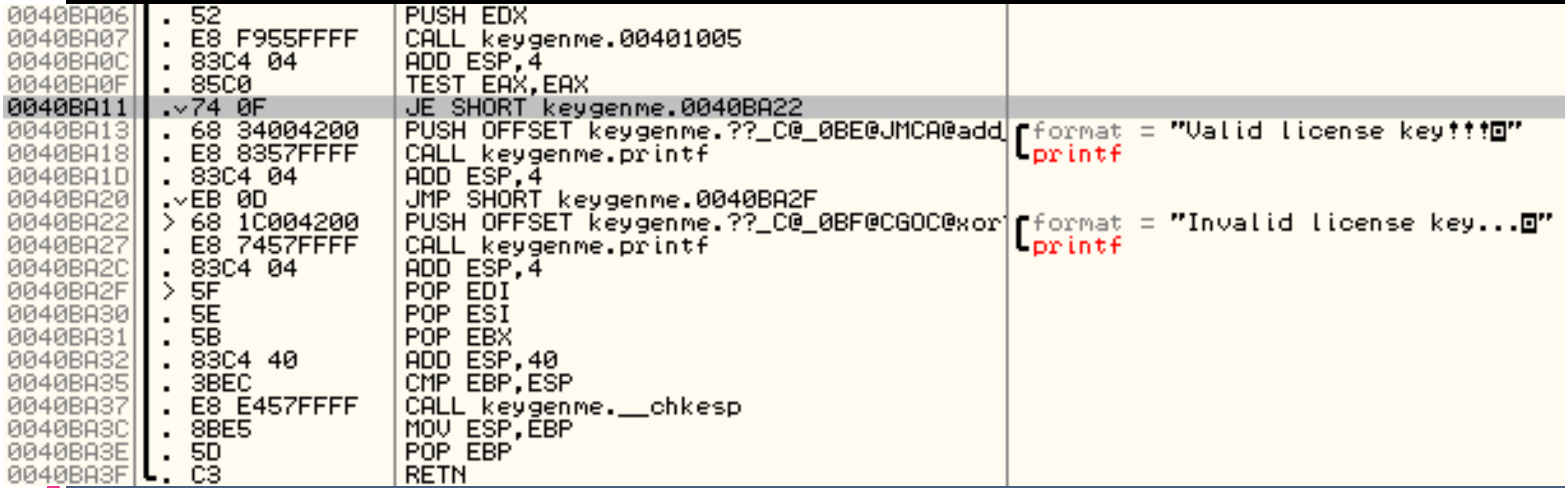
```
0012FF6C  85C4A2C8
0012FF70  AAF30C6C
0012FF74  00000000
0012FF78  E2C4D008
0012FF7C  00000000
0012FF80  85DE4FE0
0012FF84  00001FE0
0012FF88  85C4A200
0012FF8C  E2C4D408
0012FF90  00000001
0012FF94  00000000
0012FF98  00000408
```

Analysing keygenme: 161 heuristical procedures, 201 calls to known, 244 calls to guessed functions

Paused

OllyDbg
Assembly Patching Tutorial

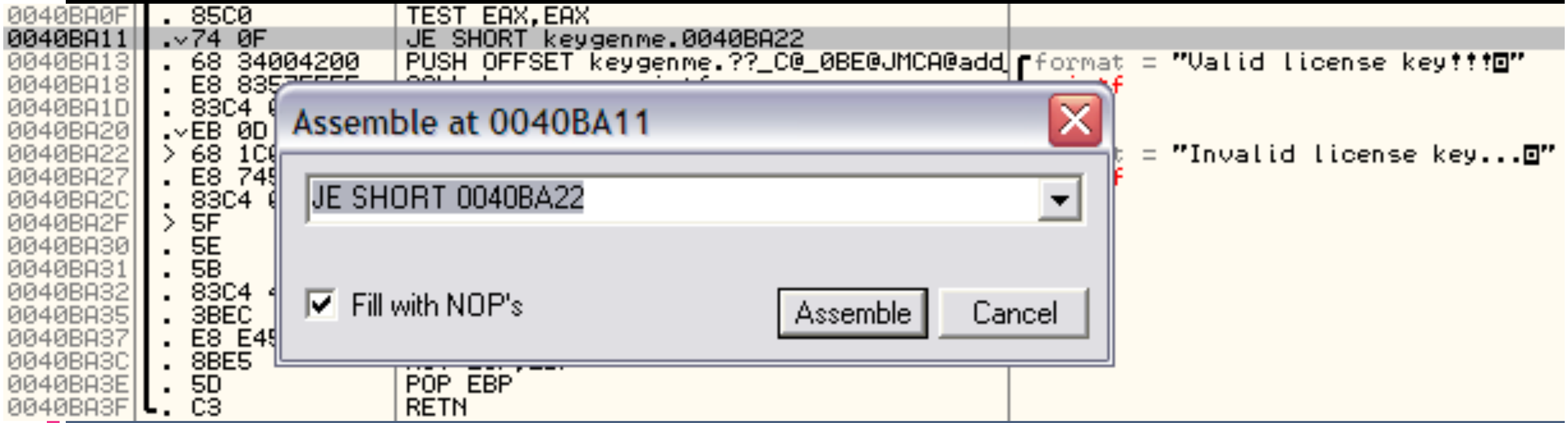
- Assembly Patching
  - Change the jump from “invalid” code to “valid” code



```
0040BA06  .  52              PUSH EDX
0040BA07  .  E8 F955FFFF     CALL keygenme.00401005
0040BA0C  .  83C4 04         ADD ESP,4
0040BA0F  .  85C0            TEST EAX,EAX
0040BA11  .v 74 0F           JE SHORT keygenme.0040BA22
0040BA13  .  68 34004200     PUSH OFFSET keygenme.??_C@_0BE@JMCA@add ┌format = "Valid license key!!!◘"
0040BA18  .  E8 8357FFFF     CALL keygenme.printf                    └printf
0040BA1D  .  83C4 04         ADD ESP,4
0040BA20  .v EB 0D           JMP SHORT keygenme.0040BA2F
0040BA22  >  68 1C004200     PUSH OFFSET keygenme.??_C@_0BF@CGOC@xor ┌format = "Invalid license key...◘"
0040BA27  .  E8 7457FFFF     CALL keygenme.printf                    └printf
0040BA2C  .  83C4 04         ADD ESP,4
0040BA2F  >  5F              POP EDI
0040BA30  .  5E              POP ESI
0040BA31  .  5B              POP EBX
0040BA32  .  83C4 40         ADD ESP,40
0040BA35  .  3BEC            CMP EBP,ESP
0040BA37  .  E8 E457FFFF     CALL keygenme.__chkesp
0040BA3C  .  8BE5            MOV ESP,EBP
0040BA3E  .  5D              POP EBP
0040BA3F  .  C3              RETN
```

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Double-click on the short jump



```
0040BA0F  . 85C0           TEST EAX,EAX
0040BA11  .v74 0F          JE SHORT keygenme.0040BA22
0040BA13  . 68 34004200    PUSH OFFSET keygenme.??_C@_0BE@JMCA@add  ┌format = "Valid license key!!!◘"
0040BA18  . E8 835
0040BA1D  . 83C4
0040BA20  .vEB 0D
0040BA22  > 68 1C                                                      = "Invalid license key...◘"
0040BA27  . E8 745
0040BA2C  . 83C4
0040BA2F  > 5F
0040BA30  . 5E
0040BA31  . 5B
0040BA32  . 83C4
0040BA35  . 3BEC
0040BA37  . E8 E45
0040BA3C  . 8BE5
0040BA3E  . 5D             POP EBP
0040BA3F  . C3             RETN
```

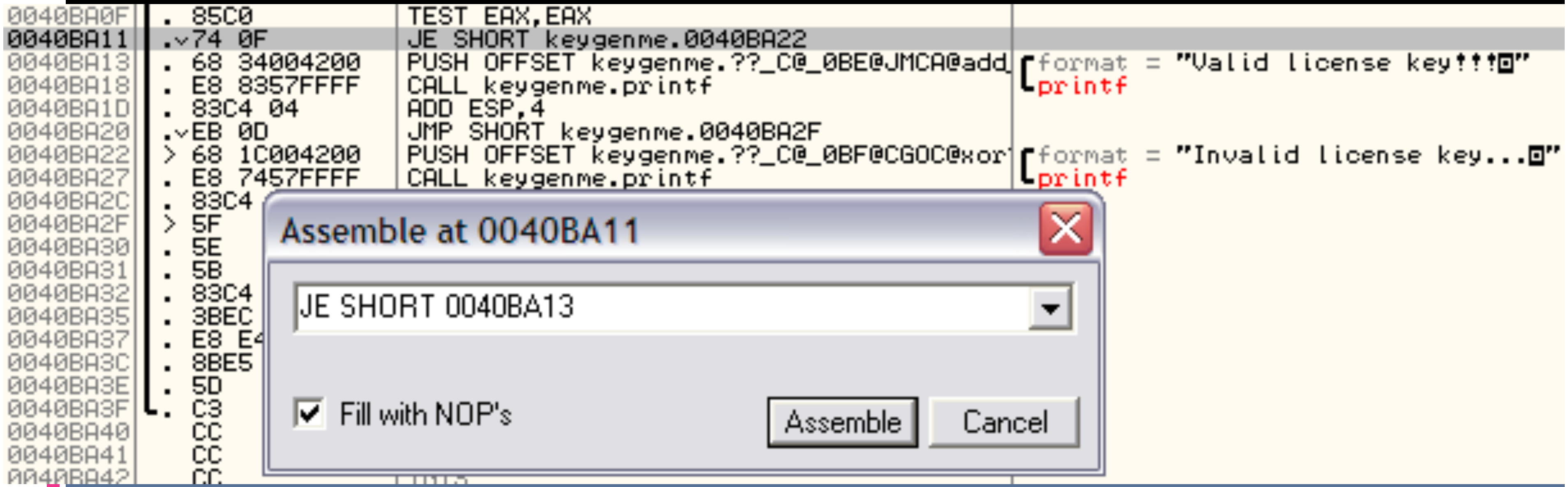
Assemble at 0040BA11

JE SHORT 0040BA22

☑ Fill with NOP's

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Change the jump address



```
0040BA0F  .  85C0          TEST EAX,EAX
0040BA11  .v 74 0F         JE SHORT keygenme.0040BA22
0040BA13  .  68 34004200   PUSH OFFSET keygenme.??_C@_0BE@JMCA@add  [format = "Valid license key!!!◘"
0040BA18  .  E8 8357FFFF   CALL keygenme.printf                     [printf
0040BA1D  .  83C4 04       ADD ESP,4
0040BA20  .v EB 0D         JMP SHORT keygenme.0040BA2F
0040BA22  >  68 1C004200   PUSH OFFSET keygenme.??_C@_0BF@CGOC@xor  [format = "Invalid license key...◘"
0040BA27  .  E8 7457FFFF   CALL keygenme.printf                     [printf
0040BA2C  .  83C4
0040BA2F  >  5F
0040BA30  .  5E
0040BA31  .  5B
0040BA32  .  83C4
0040BA35  .  3BEC
0040BA37  .  E8 E4
0040BA3C  .  8BE5
0040BA3E  .  5D
0040BA3F  .  C3
0040BA40     CC
0040BA41     CC
0040BA42     CC
```

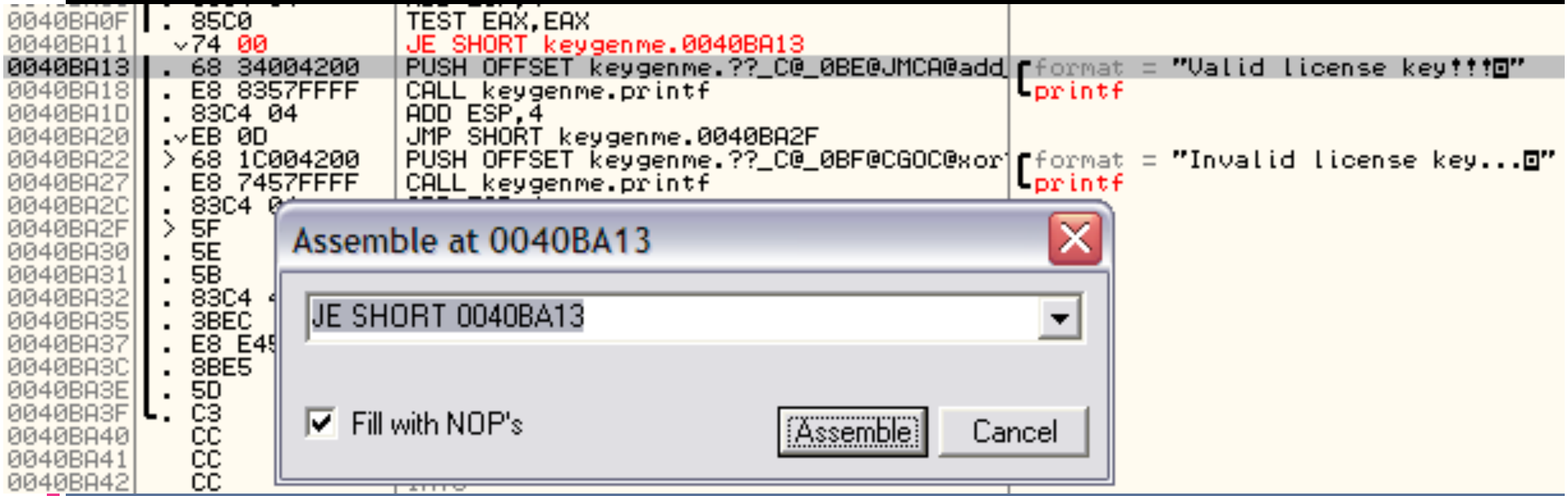
Assemble at 0040BA11

JE SHORT 0040BA13

☑ Fill with NOP's

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Hit assemble
  - Check that the size of the code hasn't changed



```
0040BA0F  . 85C0            TEST EAX,EAX
0040BA11   v74 00           JE SHORT keygenme.0040BA13
0040BA13  . 68 34004200     PUSH OFFSET keygenme.??_C@_0BE@JMCA@add  ┌format = "Valid license key!!!◘"
0040BA18  . E8 8357FFFF     CALL keygenme.printf                     └printf
0040BA1D  . 83C4 04         ADD ESP,4
0040BA20  .vEB 0D           JMP SHORT keygenme.0040BA2F
0040BA22  > 68 1C004200     PUSH OFFSET keygenme.??_C@_0BF@CGOC@xor  ┌format = "Invalid license key...◘"
0040BA27  . E8 7457FFFF     CALL keygenme.printf                     └printf
0040BA2C  . 83C4 0
0040BA2F  > 5F
0040BA30  . 5E
0040BA31  . 5B
0040BA32  . 83C4 4
0040BA35  . 3BEC
0040BA37  . E8 E45
0040BA3C  . 8BE5
0040BA3E  . 5D
0040BA3F  . C3
0040BA40    CC
0040BA41    CC
0040BA42    CC
```

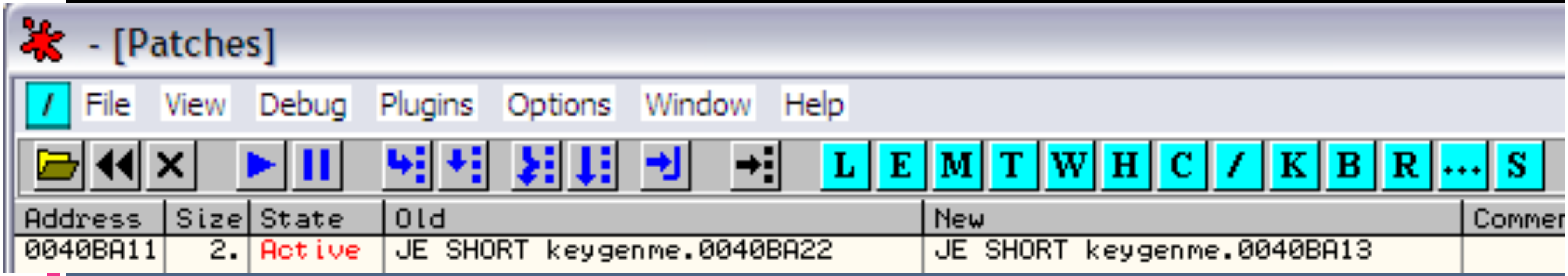
Assemble at 0040BA13

JE SHORT 0040BA13

☑ Fill with NOP's

OllyDbg
Assembly Patching Tutorial

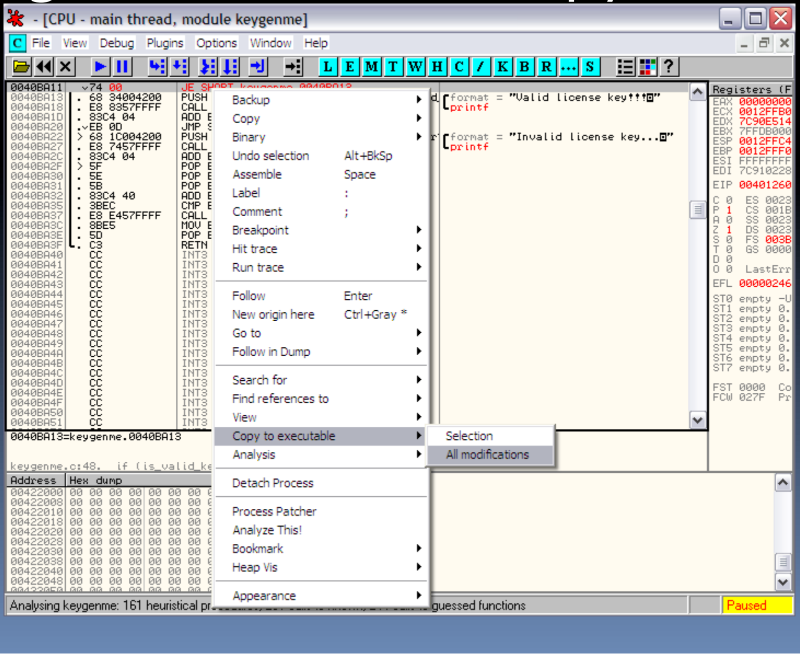
- Assembly Patching
  - View patches
    - Click on the "/" toolbar button or hit Ctrl+P
    - Right click on an entry and click "Follow in Disassembler" to return to the disassembler at the target address



Address	Size	State	Old	New	Commer
0040BA11	2.	Active	JE SHORT keygenme.0040BA22	JE SHORT keygenme.0040BA13	

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Right click on the patch
  - “Copy to executable” → “All Modifications”
  - OllyDbg 2.0: “Edit” → “Copy to executable”



```
0040BA11   v74 00          JE SHORT keygenme.0040BA13
0040BA13  . 68 34004200    PUSH
0040BA18  . E8 8357FFFF    CALL
0040BA1D  . 83C4 04        ADD E
0040BA20  .vEB 0D          JMP S
0040BA22  > 68 1C004200    PUSH
0040BA27  . E8 7457FFFF    CALL
0040BA2C  . 83C4 04        ADD E
0040BA2F  > 5F             POP E
0040BA30  . 5E             POP E
0040BA31  . 5B             POP E
0040BA32  . 83C4 40        ADD E
0040BA35  . 3BEC           CMP E
0040BA37  . E8 E457FFFF    CALL
0040BA3C  . 8BE5           MOV E
0040BA3E  . 5D             POP E
0040BA3F  . C3             RETN
0040BA40    CC             INT3
0040BA41    CC             INT3
0040BA42    CC             INT3
0040BA43    CC             INT3
0040BA44    CC             INT3
0040BA45    CC             INT3
0040BA46    CC             INT3
0040BA47    CC             INT3
0040BA48    CC             INT3
0040BA49    CC             INT3
0040BA4A    CC             INT3
0040BA4B    CC             INT3
0040BA4C    CC             INT3
0040BA4D    CC             INT3
0040BA4E    CC             INT3
0040BA4F    CC             INT3
0040BA50    CC             INT3
0040BA51    CC             INT3

d [format = "Valid license key!!!□"
  [printf
r [format = "Invalid license key...□"
  [printf
```

```
EAX 00000000
ECX 0012FFB0
EDX 7C90E514
EBX 7FFDB000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C910228

EIP 00401260

C 0  ES 0023
P 1  CS 001B
A 0  SS 0023
Z 1  DS 0023
S 0  FS 003B
T 0  GS 0000
D 0
O 0  LastErr

EFL 00000246

ST0 empty -U
ST1 empty 0.
ST2 empty 0.
ST3 empty 0.
ST4 empty 0.
ST5 empty 0.
ST6 empty 0.
ST7 empty 0.

FST 0000  Co
FCW 027F  Pr
```

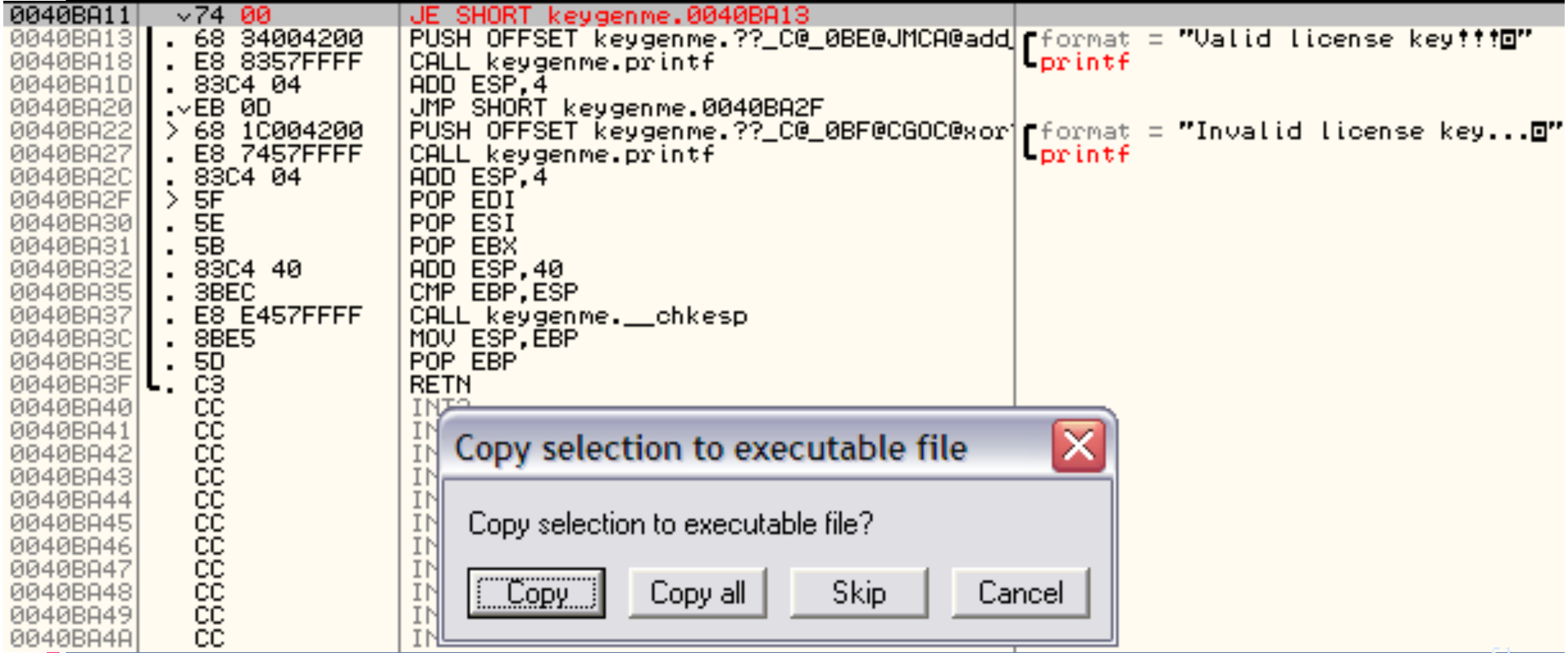
```
0040BA13=keygenme.0040BA13

keygenme.c:48.  if (is_valid_ke
```

```
00422000 00 00 00 00 00 00 00
00422008 00 00 00 00 00 00 00
00422010 00 00 00 00 00 00 00
00422018 00 00 00 00 00 00 00
00422020 00 00 00 00 00 00 00
00422028 00 00 00 00 00 00 00
00422030 00 00 00 00 00 00 00
00422038 00 00 00 00 00 00 00
00422040 00 00 00 00 00 00 00
00422048 00 00 00 00 00 00 00
```

Analysing keygenme: 161 heuristical pr guessed functions

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Select “Copy all” (OllyDbg 1.* only)



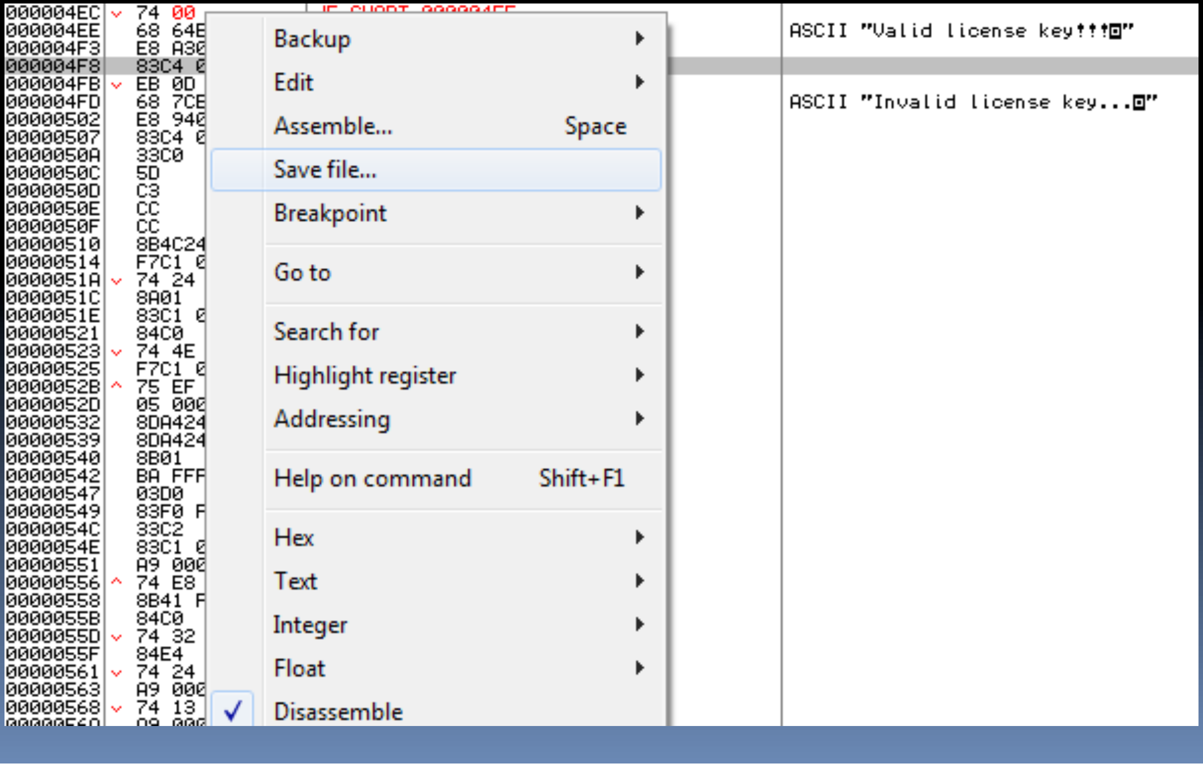
```
0040BA11    v74 00         JE SHORT keygenme.0040BA13
0040BA13  | . 68 34004200  PUSH OFFSET keygenme.??_C@_0BE@JMCA@add_  [format = "Valid license key!!!◘"
0040BA18  | . E8 8357FFFF  CALL keygenme.printf                     [printf
0040BA1D  | . 83C4 04      ADD ESP,4
0040BA20  | .vEB 0D        JMP SHORT keygenme.0040BA2F
0040BA22  | > 68 1C004200  PUSH OFFSET keygenme.??_C@_0BF@CGOC@xor'  [format = "Invalid license key...◘"
0040BA27  | . E8 7457FFFF  CALL keygenme.printf                     [printf
0040BA2C  | . 83C4 04      ADD ESP,4
0040BA2F  | > 5F           POP EDI
0040BA30  | . 5E           POP ESI
0040BA31  | . 5B           POP EBX
0040BA32  | . 83C4 40      ADD ESP,40
0040BA35  | . 3BEC         CMP EBP,ESP
0040BA37  | . E8 E457FFFF  CALL keygenme.__chkesp
0040BA3C  | . 8BE5         MOV ESP,EBP
0040BA3E  | . 5D           POP EBP
0040BA3F  L . C3           RETN
0040BA40      CC           INT3
0040BA41      CC           IN
0040BA42      CC           IN
0040BA43      CC           IN
0040BA44      CC           IN
0040BA45      CC           IN
0040BA46      CC           IN
0040BA47      CC           IN
0040BA48      CC           IN
0040BA49      CC           IN
0040BA4A      CC           IN
```

Copy selection to executable file

Copy selection to executable file?

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Right click in the newly generated executable
  - Select “Save file”

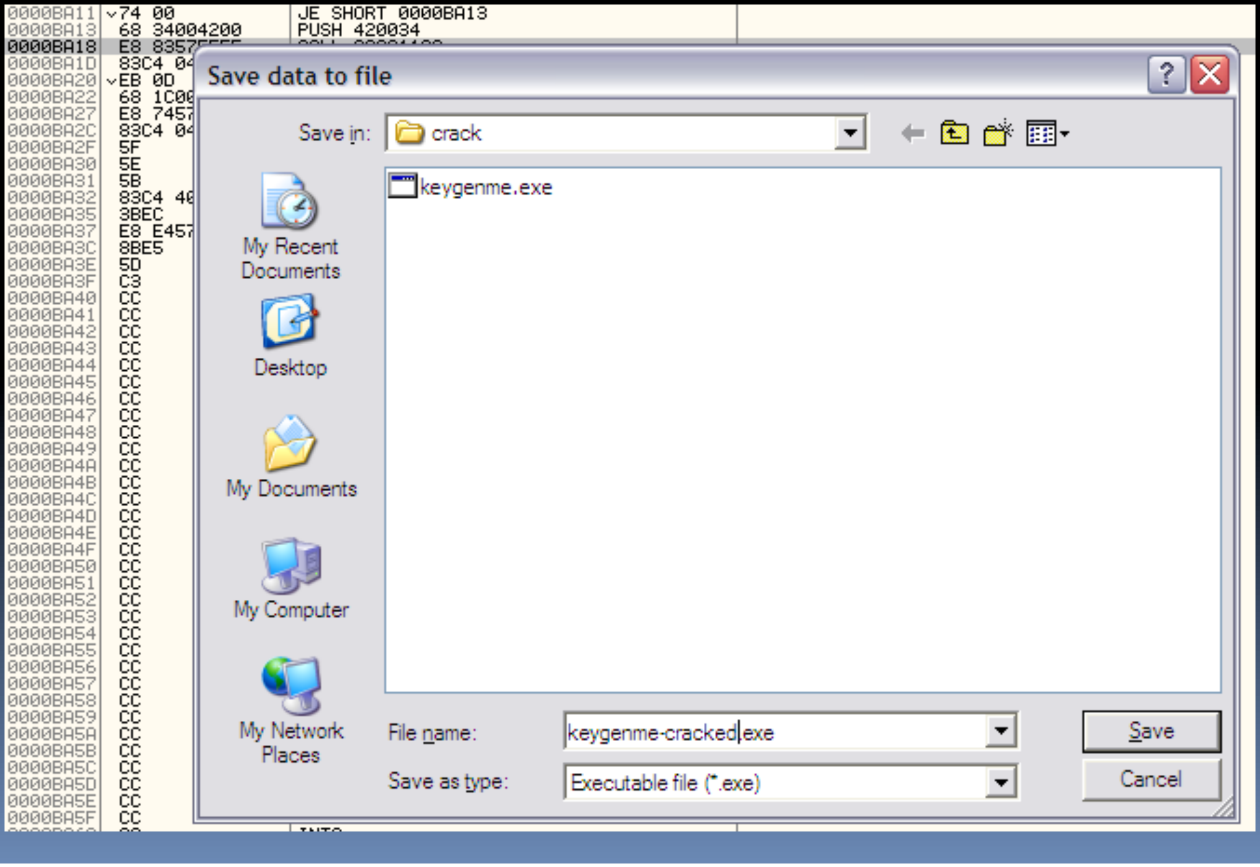


```
000004EC  v  74 00
000004EE     68 64E
000004F3     E8 A30
000004F8     83C4 0
000004FB  v  EB 0D
000004FD     68 7CE
00000502     E8 940
00000507     83C4 0
0000050A     33C0
0000050C     5D
0000050D     C3
0000050E     CC
0000050F     CC
00000510     8B4C24
00000514     F7C1 0
0000051A  v  74 24
0000051C     8A01
0000051E     83C1 0
00000521     84C0
00000523  v  74 4E
00000525     F7C1 0
0000052B  ^  75 EF
0000052D     05 000
00000532     8DA424
00000539     8DA424
00000540     8B01
00000542     BA FFF
00000547     03D0
00000549     83F0 F
0000054C     33C2
0000054E     83C1 0
00000551     A9 000
00000556  ^  74 E8
00000558     8B41 F
0000055B     84C0
0000055D  v  74 32
0000055F     84E4
00000561  v  74 24
00000563     A9 000
00000568  v  74 13
```

```
ASCII "Valid license key!!!◘"

ASCII "Invalid license key...◘"
```

OllyDbg
Assembly Patching Tutorial

- Assembly Patching
  - Save the new executable



```
0000BA11 v74 00         JE SHORT 0000BA13
0000BA13  68 34004200   PUSH 420034
0000BA18  E8 8357
0000BA1D  83C4 04
0000BA20 vEB 0D
0000BA22  68 1C00
0000BA27  E8 7457
0000BA2C  83C4 04
0000BA2F  5F
0000BA30  5E
0000BA31  5B
0000BA32  83C4 40
0000BA35  3BEC
0000BA37  E8 E457
0000BA3C  8BE5
0000BA3E  5D
0000BA3F  C3
0000BA40  CC
0000BA41  CC
0000BA42  CC
0000BA43  CC
0000BA44  CC
0000BA45  CC
0000BA46  CC
0000BA47  CC
0000BA48  CC
0000BA49  CC
0000BA4A  CC
0000BA4B  CC
0000BA4C  CC
0000BA4D  CC
0000BA4E  CC
0000BA4F  CC
0000BA50  CC
0000BA51  CC
0000BA52  CC
0000BA53  CC
0000BA54  CC
0000BA55  CC
0000BA56  CC
0000BA57  CC
0000BA58  CC
0000BA59  CC
0000BA5A  CC
0000BA5B  CC
0000BA5C  CC
0000BA5D  CC
0000BA5E  CC
0000BA5F  CC
```

Save data to file

Save in: crack

keygenme.exe

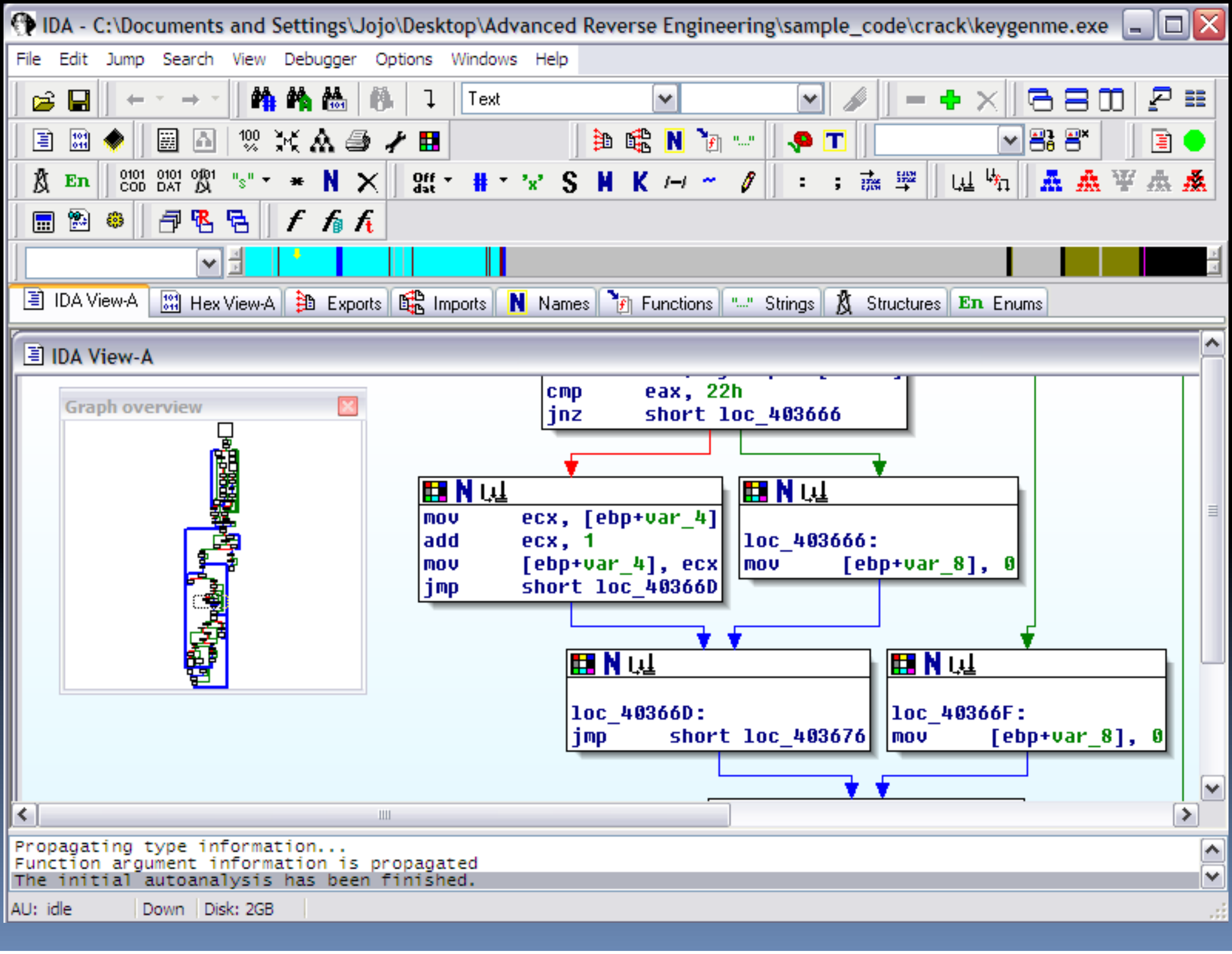
File name: keygenme-cracked.exe

Save as type: Executable file (*.exe)

IDA

- Disassembler, Decompiler*, Debugger
  - Commercial debugger
    - With freeware and demo versions
  - Now a Hex-Rays product
    - Formerly Datarescue
  - *Decompilers sold seperately (and is expensive)

IDA



```
cmp     eax, 22h
jnz     short loc_403666
```

```
mov     ecx, [ebp+var_4]
add     ecx, 1
mov     [ebp+var_4], ecx
jmp     short loc_40366D
```

```
loc_403666:
mov     [ebp+var_8], 0
```

```
loc_40366D:
jmp     short loc_403676
```

```
loc_40366F:
mov     [ebp+var_8], 0
```

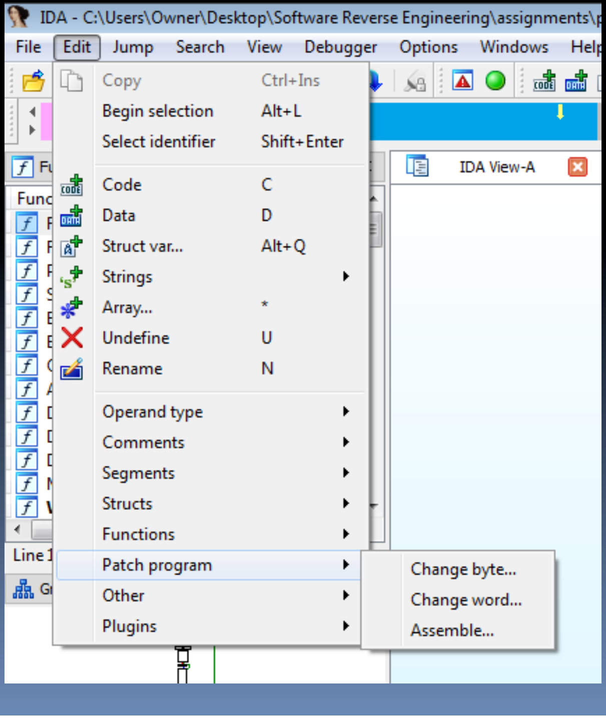
```
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
```

IDA

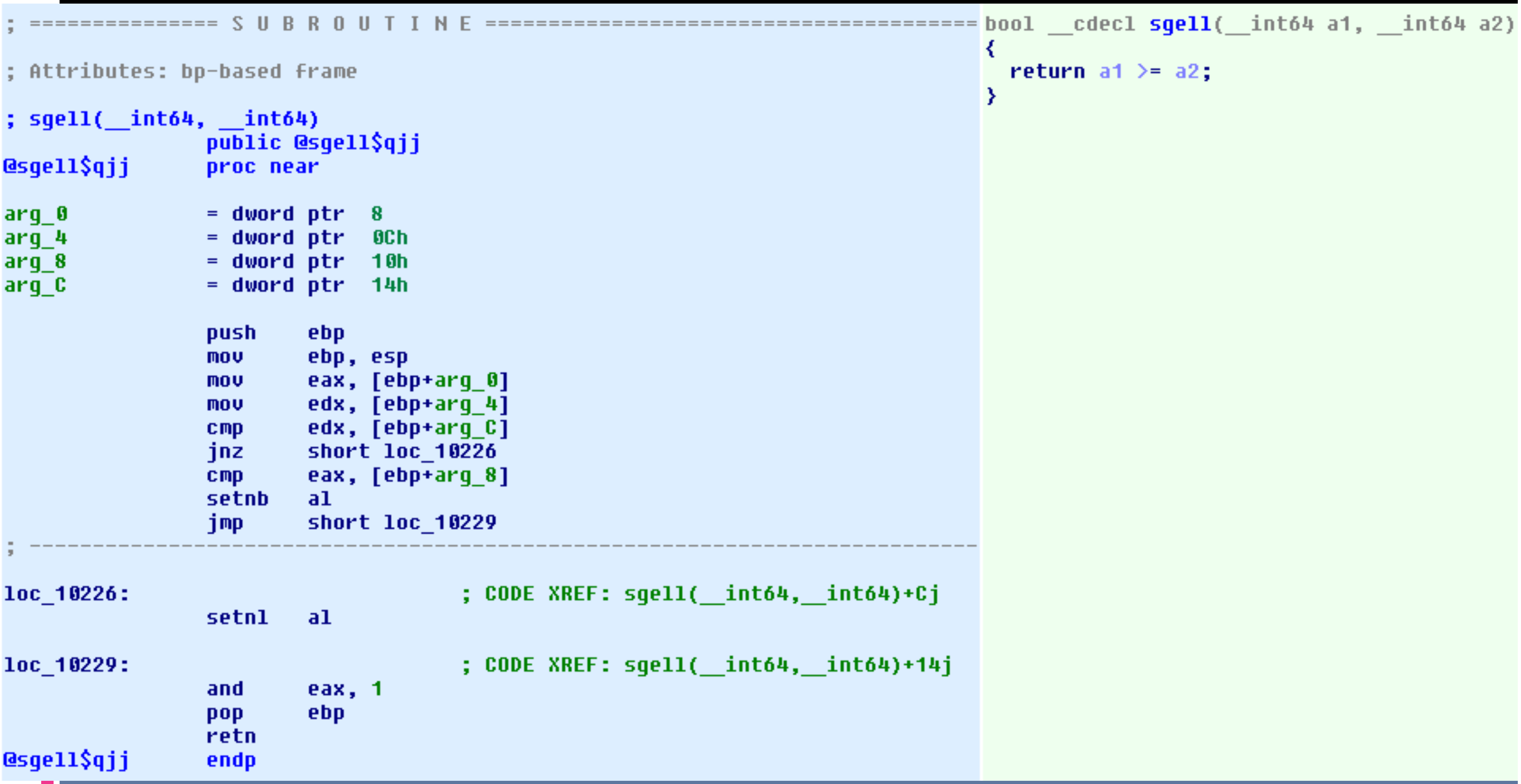
- Shortcuts
    - Run (F9), step into (F7), step over (F8)
    - Set/clear breakpoint (F2)
    - Apply name to an address (N)
    - Comment (:), repeatable comment (;)
    - Toggle graph view/assembly view (space)
    - Jump to name/address (G)
    - Follow reference (enter)
    - Display/jump to cross-references (X)
    - Return to previous location (esc)

IDA Patching

- Patching
  - Edit “cfg/idagui.cfg”
  - Change “DISPLAY_PATCH_SUBMENU” to “YES”



Hex-Rays Decompilers

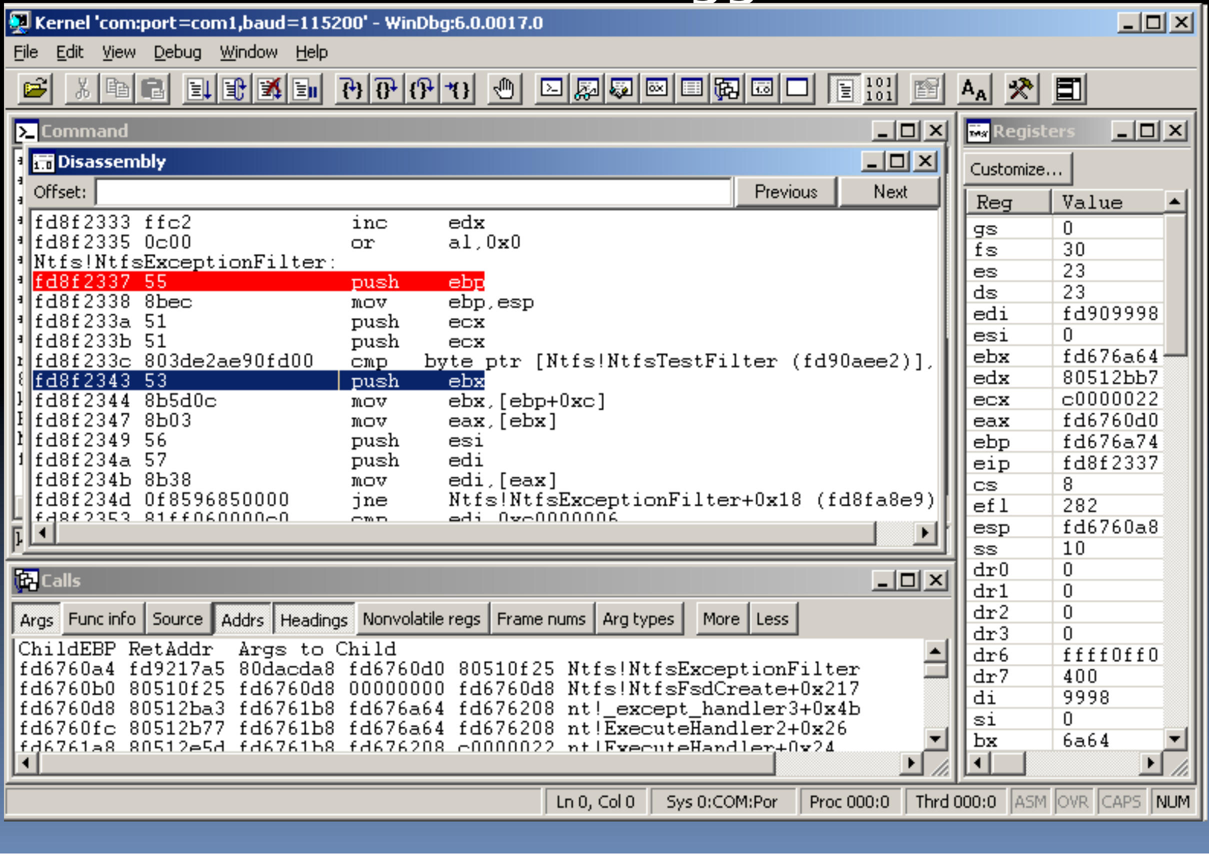


```
; =============== S U B R O U T I N E =======================================

; Attributes: bp-based frame

; sgell(__int64, __int64)
                public @sgell$qjj
@sgell$qjj      proc near

arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch
arg_8           = dword ptr  10h
arg_C           = dword ptr  14h

                push    ebp
                mov     ebp, esp
                mov     eax, [ebp+arg_0]
                mov     edx, [ebp+arg_4]
                cmp     edx, [ebp+arg_C]
                jnz     short loc_10226
                cmp     eax, [ebp+arg_8]
                setnb   al
                jmp     short loc_10229
; ---------------------------------------------------------------------------

loc_10226:                              ; CODE XREF: sgell(__int64,__int64)+Cj
                setnl   al

loc_10229:                              ; CODE XREF: sgell(__int64,__int64)+14j
                and     eax, 1
                pop     ebp
                retn
@sgell$qjj      endp
```

```
bool __cdecl sgell(__int64 a1, __int64 a2)
{
  return a1 >= a2;
}
```

WinDbg

- Disassembler, Debugger
  - User/kernel-mode debugger



```
fd8f2333 ffc2           inc     edx
fd8f2335 0c00           or      al,0x0
Ntfs!NtfsExceptionFilter:
fd8f2337 55             push    ebp
fd8f2338 8bec           mov     ebp,esp
fd8f233a 51             push    ecx
fd8f233b 51             push    ecx
fd8f233c 803de2ae90fd00 cmp   byte ptr [Ntfs!NtfsTestFilter (fd90aee2)],
fd8f2343 53             push    ebx
fd8f2344 8b5d0c         mov     ebx,[ebp+0xc]
fd8f2347 8b03           mov     eax,[ebx]
fd8f2349 56             push    esi
fd8f234a 57             push    edi
fd8f234b 8b38           mov     edi,[eax]
fd8f234d 0f8596850000   jne     Ntfs!NtfsExceptionFilter+0x18 (fd8fa8e9)
fd8f2353 81ff060000c0   cmp     edi,0xc0000006
```

```
Reg   Value
gs    0
fs    30
es    23
ds    23
edi   fd909998
esi   0
ebx   fd676a64
edx   80512bb7
ecx   c0000022
eax   fd6760d0
ebp   fd676a74
eip   fd8f2337
cs    8
efl   282
esp   fd6760a8
ss    10
dr0   0
dr1   0
dr2   0
dr3   0
dr6   ffff0ff0
dr7   400
di    9998
si    0
bx    6a64
```

```
ChildEBP RetAddr  Args to Child
fd6760a4 fd9217a5 80dacda8 fd6760d0 80510f25 Ntfs!NtfsExceptionFilter
fd6760b0 80510f25 fd6760d8 00000000 fd6760d8 Ntfs!NtfsFsdCreate+0x217
fd6760d8 80512ba3 fd6761b8 fd676a64 fd676208 nt!_except_handler3+0x4b
fd6760fc 80512b77 fd6761b8 fd676a64 fd676208 nt!ExecuteHandler2+0x26
fd6761a8 80512e5d fd6761b8 fd676208 c0000022 nt!ExecuteHandler+0x24
```

```
Ln 0, Col 0   Sys 0:COM:Por   Proc 000:0   Thrd 000:0   ASM   OVR   CAPS   NUM
```

Questions/Comments?