Introduction to Software Reverse Engineering with Ghidra Session 3: C to ASM II

Hackaday U

Matthew Alt

#Outline: C to ASM

- Class Admin
- SRE Tool Landscape Review
- Structures
- Enums
- Pointers
- File Operations
  - Open / Close / Read / Write
- Patching Binaries
- Ghidra Features
  - Bookmarks
  - Searching

#Course Administration

- Office hours will be Thursday at 6:00 ET
- Questions for office hours can be submitted via zoom
- Questions can also be submitted through:
  - Hackaday.io chat room
  - Hackaday messaging

#Session Goals

- Discuss SRE tool landscape
- Review Additional C Constructs / OS Concepts
  - Structures
  - Pointers
  - Enums
  - File operations
  - System calls
- Ghidra Feature Review
  - Binary patching
  - Searching
  - Bookmarks
  - Data type creation

#SRE Tool Landscape

- Many tools exist to perform SRE, below are a few:
  - Radare/r2 - disassembler
  - IDA Pro – disassembler/decompiler
  - Ghidra – disassembler/decompiler
  - Objdump - disassembler

- All of these tools have pros and cons
  - Many people use a combination of these tools

#SRE Tools: A Comparison

- IDA Pro, Pros:
  - Large amount of existing plugins and community tools
  - Handles C++ binaries well
  - PDB Parsing works well
  - In development for over a decade
- IDA Pro, Cons:
  - Costly
  - Documentation can sometimes be lacking (this is improving)
  - Can be difficult to modify core behavior

#SRE Tools: A Comparison

- Radare2, Pros:
  - Open source, regularly updated, active community
  - Forensic features (mount FS, detect partitions, etc)
  - Supports multiple architectures, file formats, Oses, etc
  - Extensible scripting interface
- Radare2, Cons:
  - Steep learning curve (initially)
  - No maintained GUI (3rd party GUIs have been developed)
  - No decompiler

#SRE Tools: Wrap up

- There are plenty of factors to consider when choosing SRE tools:
  - Usability
  - Architecture support
  - File format support
  - Extendibility
  - Price
- As you perform more SRE, experiment with as many tools as possible
  - Using other tools sometimes can help you refine how you approach SRE

#Structures

- User defined data type allowing for combining various data types
- Composite data type defining a grouped list of variables under one name in a block of memory
- Directly reference a contiguous block of memory
  - Padding bytes may be added for alignment purposes

#Structures: Padding

- Elements in structures need to be byte aligned
  - This is compiler dependent (most compilers will do this however)
- Fields that are not byte aligned with the CPU will be padded
  - This is to stop unaligned accesses from occurring
  - Used for compiler optimizations as well
- Ex: A byte sized element will occupy 4 bytes of memory but only use 1

#Structures: C to ASM

- C Code

```
typedef struct {
    int x;
    char y;
    int z;
}point;

int main(int argc, char * argv[])
{
    point test;
    test.x = 100;
    test.y = 'X';
    test.z = 300;
    return 0;
}
```

- Assembly Code

```
<main>:    push   rbp
<main+1>:  mov    rbp,rsp
<main+4>:  mov    DWORD PTR [rbp-0x14],edi
<main+7>:  mov    QWORD PTR [rbp-0x20],rsi
<main+11>: mov    DWORD PTR [rbp-0x10],0x64
<main+18>: mov    BYTE PTR [rbp-0xc],0x58
<main+22>: mov    DWORD PTR [rbp-0x8],0x12C
<main+29>: mov    eax,0x0
<main+34>: pop    rbp
<main+35>: ret
```

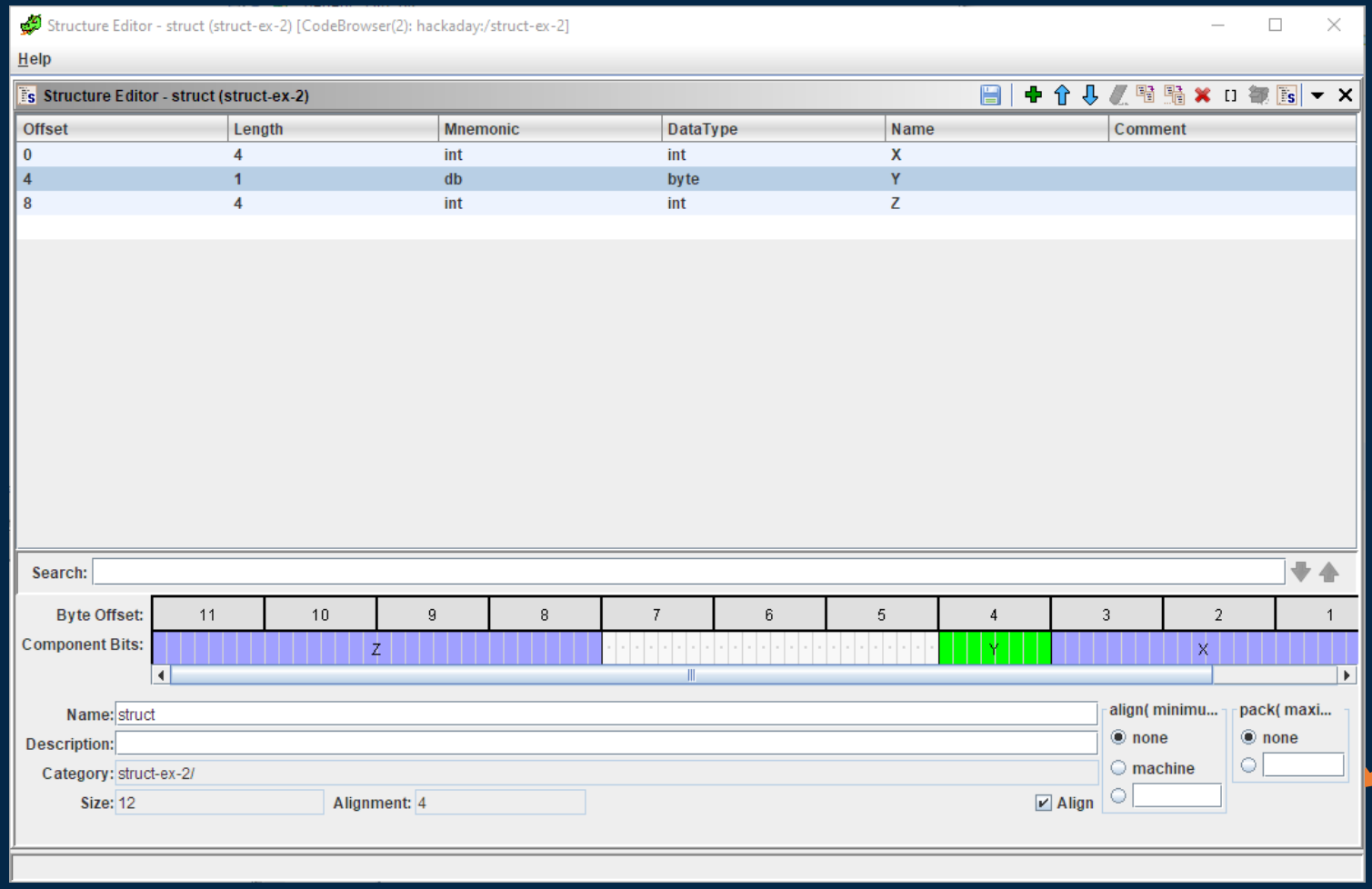
Stack Addr	Value
rbp-0x8	0x12C
rbp-0xC	0x58
rbp-0x10	0x64
rbp-0x14	RSI
rbp-0x20	EDI

#Structures: Ghidra

- Structures can be added to Ghidra in multiple ways
  - Import header files
  - Manual creation in struct creator
- To import a c file do the following:
  - File -> Import C Source
- Manual struct creation can be done from the data types manager

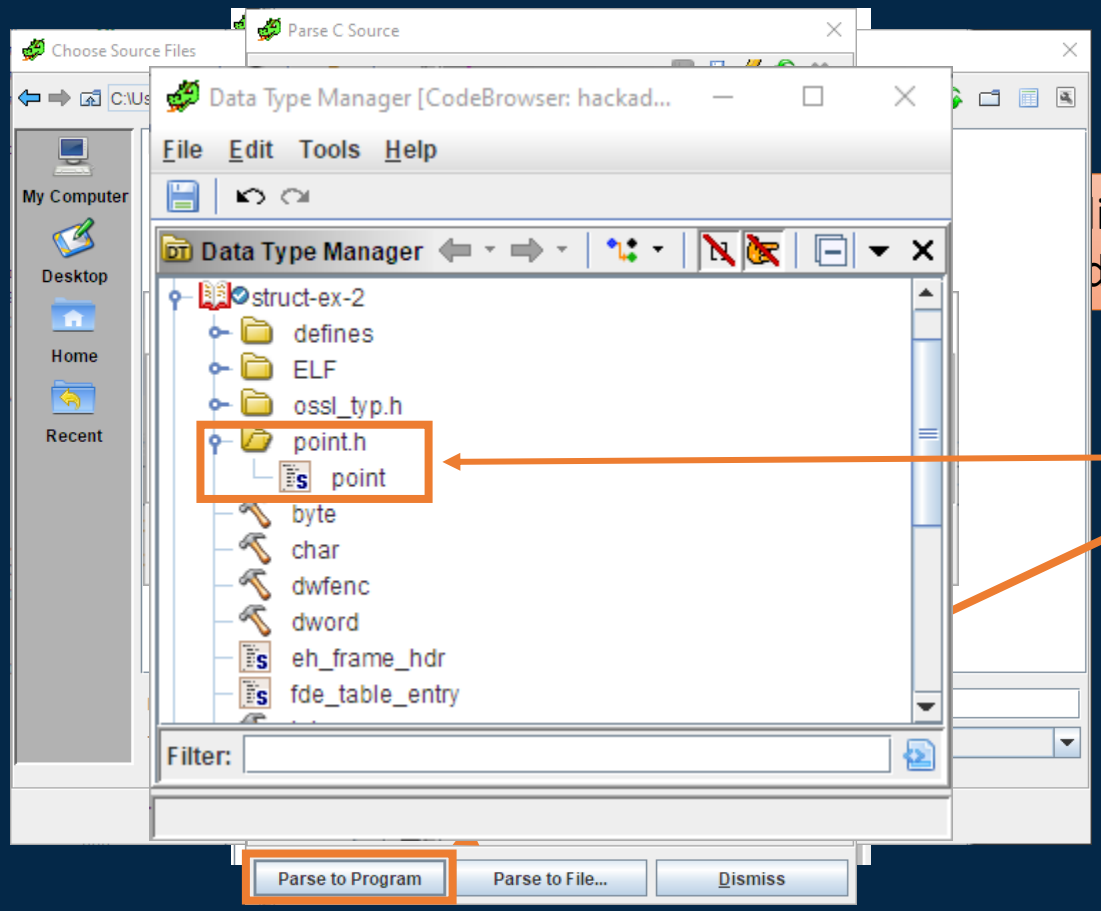
#Ghidra Tip: Structure Creation

Remember, the struct took up 12 bytes total on the stack

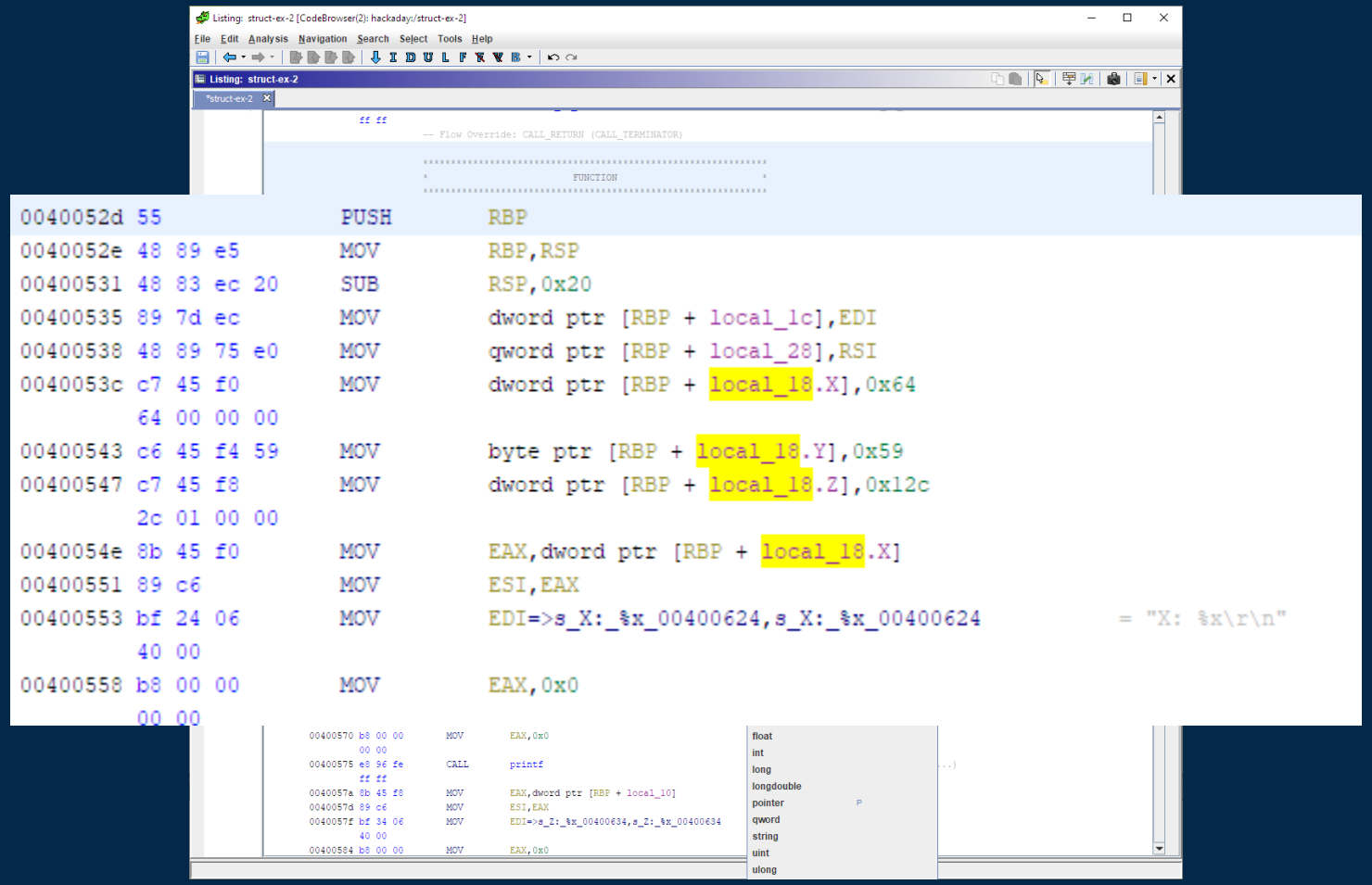


Select this option to fix the alignment!

#Ghidra Tip: Structure Creation



#Ghidra Tip: Appyling Structs



#Structures: Exercise

- Load the exercise from session-three/exercises/structs
- How many members does this struct have?
- What do the various members represent?
- Can you re-create the struct in Ghidra?

#Tip: Non Ascii from CMD line

- Thus far, the challenges for this course have worked on ASCII inputs/outputs
- Python can be used to provide non-ASCII input from the command line
- Example, if you wanted to provide the hex values 0x210010, you would perform the following:
  - `./exercise `python -c 'print “\x21\x00\x10" '`

#Pointers

- A pointer is a variable whose value is the address of another variable
  - Direct address of a memory location
- Pointers provide an indirect method of accessing variables
- Pointers are always the same size, despite the size of the data that they point to
  - This will be compiler / architecture dependent

#Pointeres: C to ASM

- C Code

```c
void swap(int *x, int *y) {
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}
int main() {
    int a = 2, b = 3;
    swap(&a, &b);
    return 0;
}
```

- Assembly Code: main

```
push   rbp
mov    rbp,rsp
sub    rsp,0x10
mov    DWORD PTR [rbp-0x8],0x2
mov    DWORD PTR [rbp-0x4],0x3
lea    rdx,[rbp-0x4]
lea    rax,[rbp-0x8]
mov    rsi,rdx
mov    rdi,rax
call   4004ed <swap>
mov    eax,0x0
leave
ret
```

#Pointeres: C to ASM

- C Code

```c
void swap(int *x, int *y) {
    int tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}
int main() {
    int a = 2, b = 3;
    swap(&a, &b);
    return 0;
}
```

- Assembly Code: swap

```
push   rbp
mov    rbp,rsp
mov    QWORD PTR [rbp-0x18],rdi
mov    QWORD PTR [rbp-0x20],rsi
mov    rax,QWORD PTR [rbp-0x18]
mov    eax,DWORD PTR [rax]
mov    DWORD PTR [rbp-0x4],eax
mov    rax,QWORD PTR [rbp-0x20]
mov    edx,DWORD PTR [rax]
mov    rax,QWORD PTR [rbp-0x18]
mov    DWORD PTR [rax],edx
mov    rax,QWORD PTR [rbp-0x20]
mov    edx,DWORD PTR [rbp-0x4]
mov    DWORD PTR [rax],edx
pop    rbp
ret
```

#Pointers: Ghidra

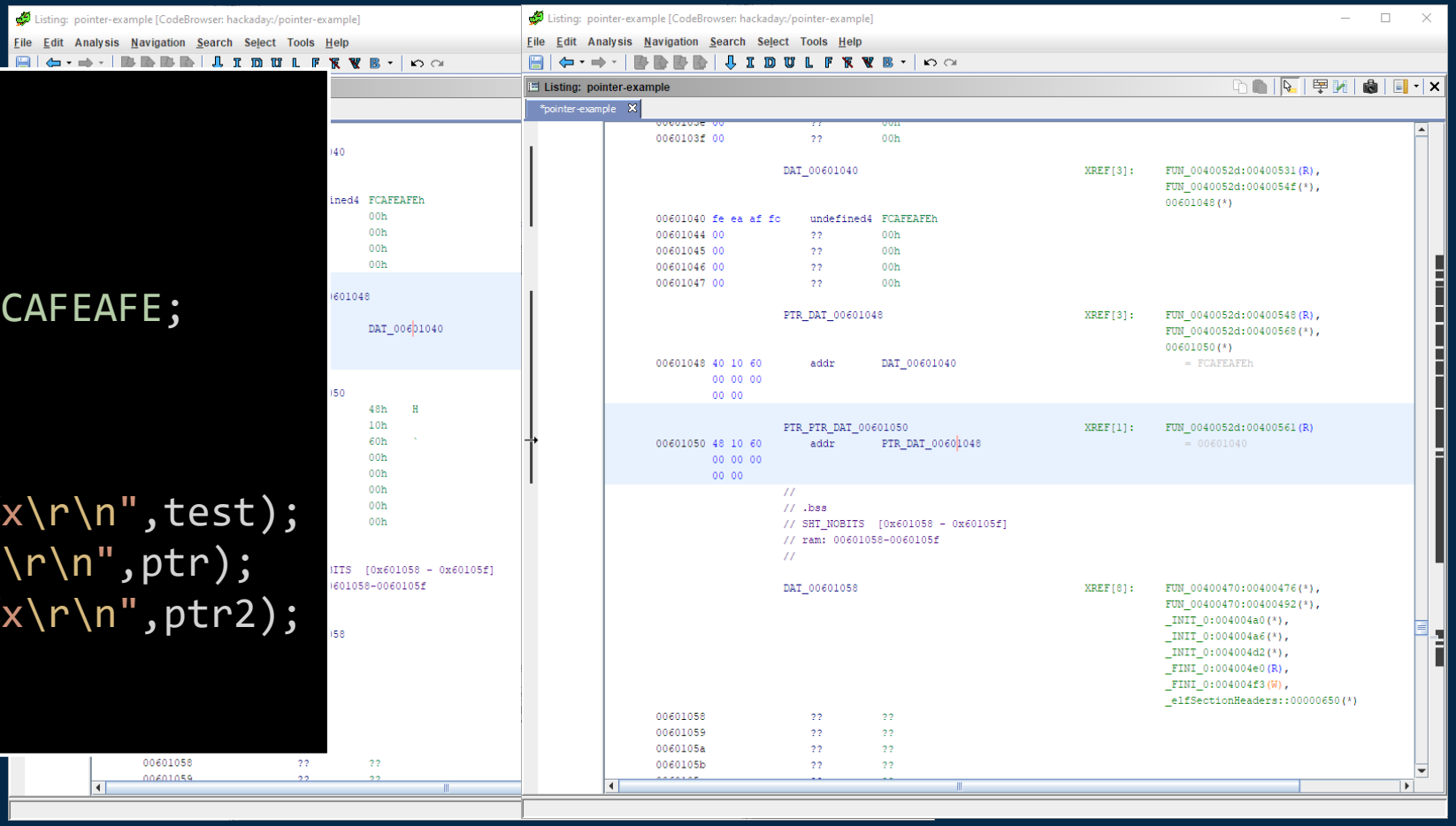
- Pointer to memory locations can be made in Ghidra using the “Pointer” datatype

- The unary operator can also be applied to any data type in the data types manager
  - For example, a pointer to the previously defined structure would be: point*

#Pointers: Ghidra

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

int test = 0xDEADBEEFCAFEAFE;
int *ptr = &test;
int * ptr2 = &ptr;
int main() {
    printf("test: 0x%x\r\n",test);
    printf("ptr: 0x%x\r\n",ptr);
    printf("ptr2: 0x%x\r\n",ptr2);
}
```



#Pointers: Exercise

- Load the exercise from session-three/exercises/pointers
- How many members does this struct have?
  - HINT: It will look familiar to the previous struct
- What do the various members represent?
- How are pointers being used here?

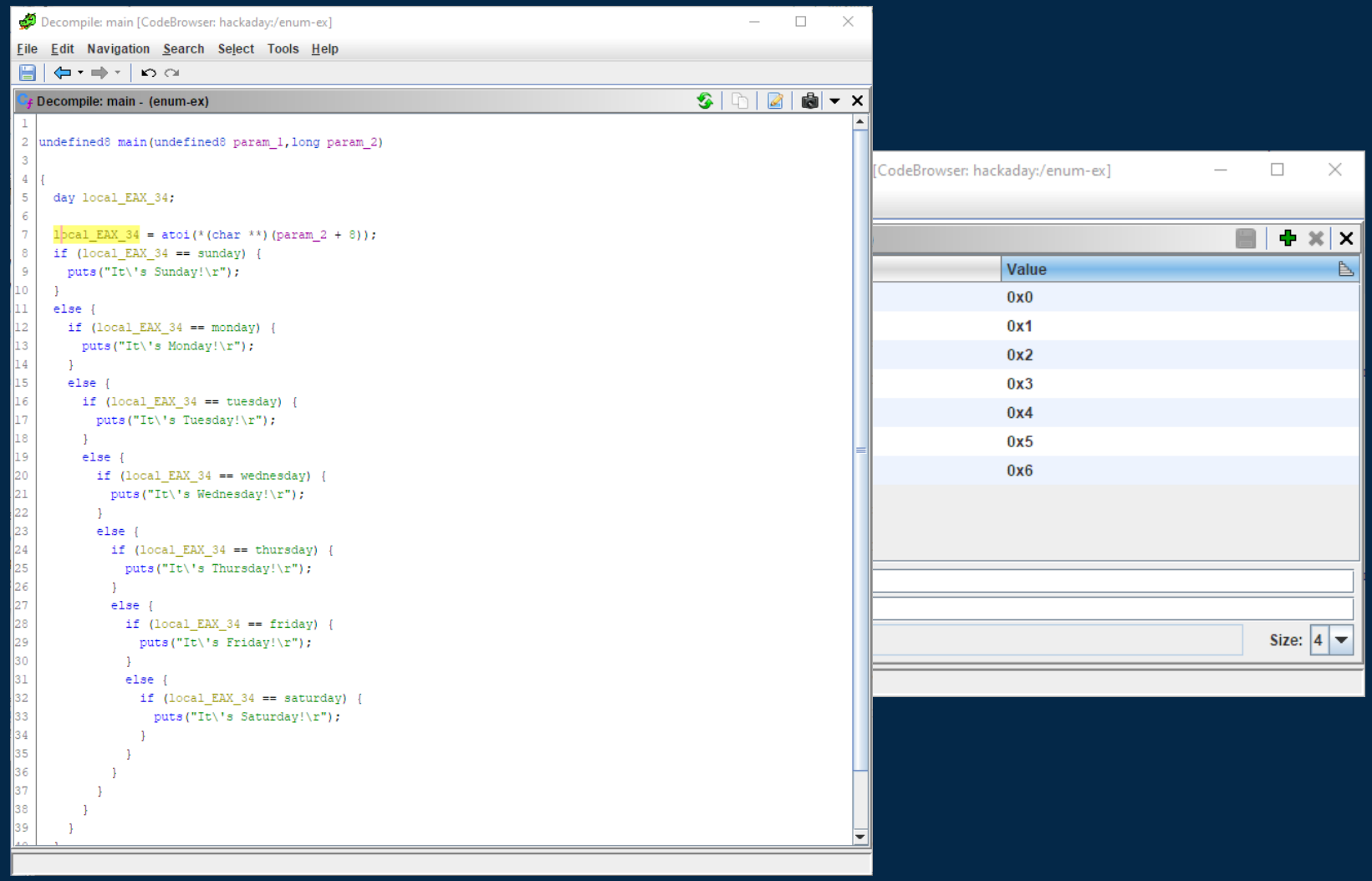
#Enums

- Enums are a user defined data type in C
- Enums assign names to integral constructs
  - Ex; `enum test  { pass = 1, fail = 0};`
- Often utilized to increase readability of code
- If values are not explicitly assigned to enum names, the compiler starts assignments at 0

#Enums: Ghidra

- Similar to structures, enums can be imported through C files or defined manually
- These imported/created enums can be applied to scalars in the disassembly listing
- Variables in the decompiler view can also be re-typed to enums that you have defined

#Enums: Ghidra



#x86_64: System Calls

- System calls are used to interface with the operating system
  - Used for operations which require privileged access
  - This is done through the `syscall` instruction
- The operating system implements a system call handler to catch when this instruction occurs
- Information about the requested syscall is passed through registers

#x86_64: System Calls

Register	Purpose
RAX	System Call Number
RCX	Return Address
R11	Saved RFLAGS
RDI	Argument 0
RSI	Argument 1
RDX	Argument 2
R10	Argument 3
R8	Argument 4
R9	Argument 5

#System Calls: ASM

```
section .rodata
    msg: db 'hackaday-u',10;

section .text
        global _start
    _start:
        mov rdi, 1
        mov rsi, msg
        mov rdx, 10
        mov rax, 1
        syscall
        mov rdi, 0
        mov rax, 60
        syscall
```

Register	Purpose	Val
RAX	System Call Number	60
RDI	Argument 0	0
RSI	Argument 1	NA
RDX	Argument 2	NA

System Calls: Exercises

- Load the exercise from session-three/exercises/syscall-exercise
- How many system calls are performed?
  - What syscalls are performed
- What does this program do?
  - Can you figure it out without running it?
- What is the entry point of this program?

#File Operations: C to ASM

- While system calls are used at a low level to perform file operations, you will often see higher level implementations
  - libc provides these for you
  - open, write, close, etc

- For the purposes of our exercises, these will just look like regular function calls!
  - Libc utilizes the system call functionality to provide these to you!

#File Operations: C to ASM

- C Code

```c
int main(int argc, char * argv[])
{
    int fd;
    char * output = "write to file";
    fd = open("testfile.txt",O_CREAT|O_WRONLY);
    int written = write(fd,output,13);
    int x = close(fd);
    return 0;
}
```

- Assembly Code

```
mov  DWORD PTR [rbp-0x14],eax
push rbp
mov  QWORD PTR [rbp-0x10],rax
mov  rbp,rsp
mov  eax,DWORD PTR [rbp-0x14]
mov  edx,0x0
call 400420 <open@plt>
mov  QWORD PTR [rbp-0x30],rsi
mov  edi,eax
mov  QWORD PTR [rbp-0x8],0x4006a4
call 400480 <write@plt>
```

```
Contents of section .rodata:
4006a0 01000200 77726974 6520746f 2066696c  ....write to fil
4006b0 65007465 73746669 6c652e74 78740000  e.testfile.txt..
```

#File Operations: Exercises

- Load the exercise from session-three/exercises/files
  - This exercise builds off the struct / pointer exercise
- What is different about this exercise?
- What file operations are being performed?
- Can you solve for the password?

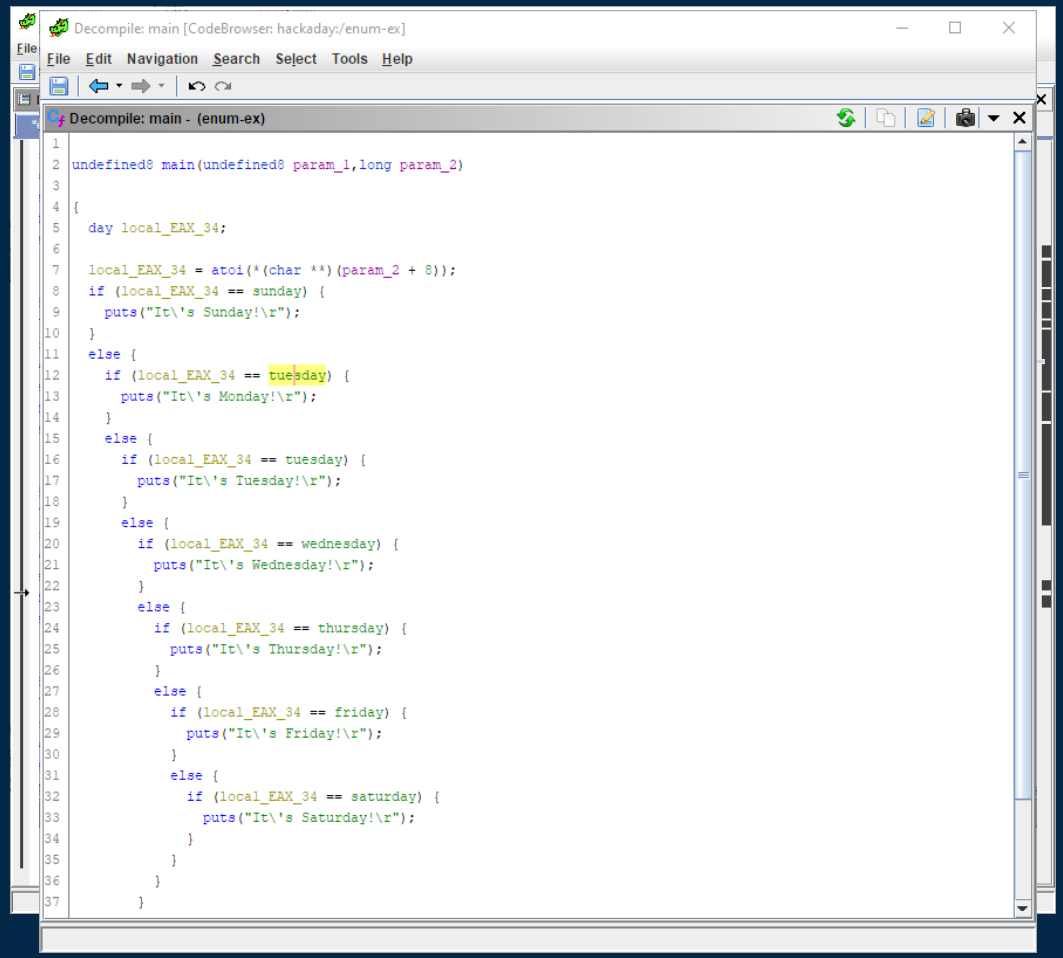
#Ghidra Tips

- During this next section, we will review various Ghidra features that may assist you with the challenges
  - Patching
  - Bookmarks
  - Searching
  - Checksumming
  - Adding additional Libraries

#Ghidra Tip: Binary Patching

- Ghidra can be used to patch binary files
  - Includes an assembler to allow use of actual instructions
  - Not just binary patching with hex values!
- The export functionality is used to dumps all initialized memory blocks within Ghidra
  - The result is not a valid ELF file!
- Note that this will not work on files such as ELF/EXE currently
  - This feature CAN be utilized on flat firmware images however

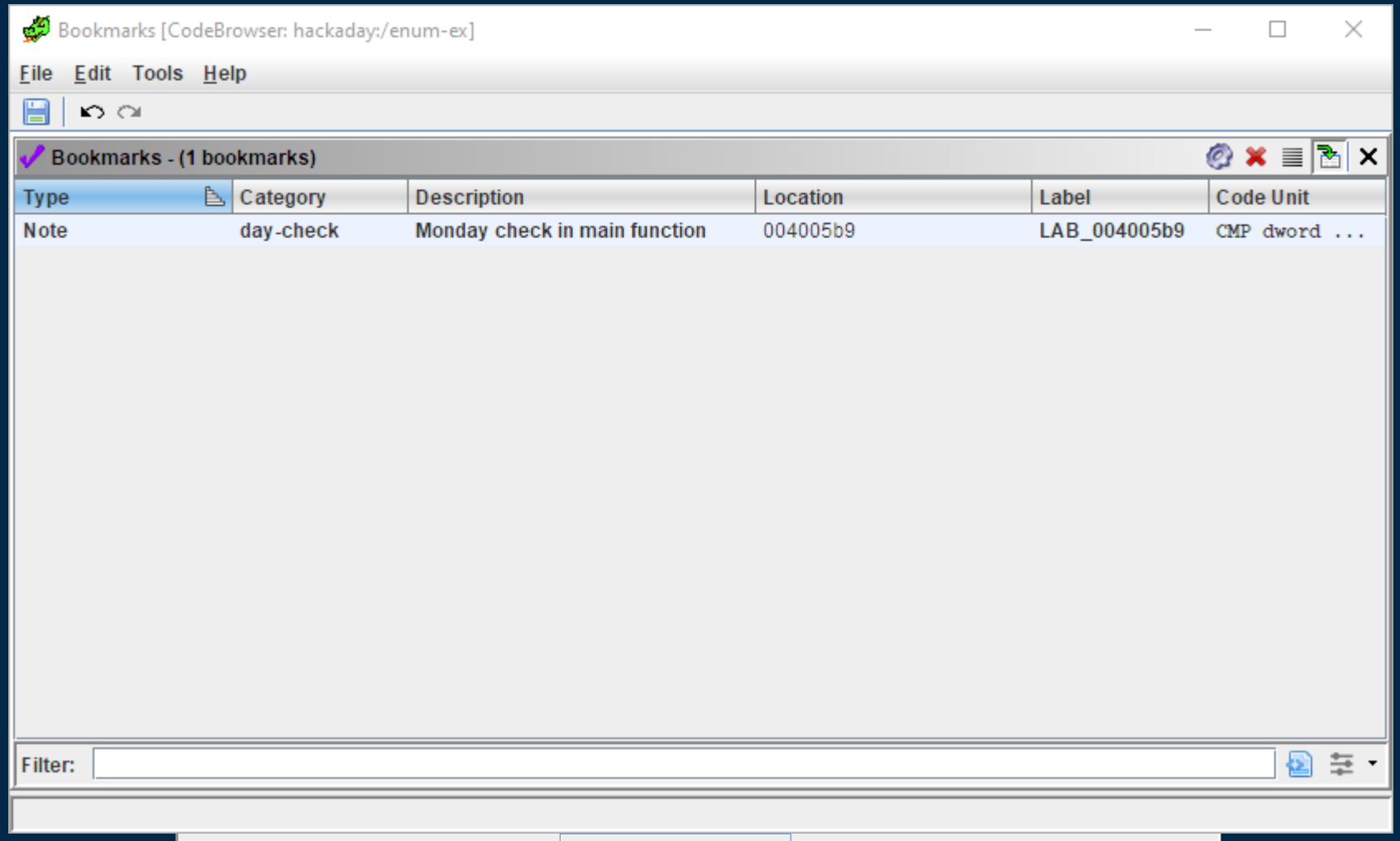
#Ghidra Tip: Binary Patching



#Ghidra Tip: Bookmarks

- When using Ghidra, locations of interest can be bookmarked
- Bookmarks can be viewed and searched within the bookmarks window
  - Window -> Bookmarks
- Bookmarks are tied to addresses, and can be categorized with custom fields

#Ghidra Tip: Bookmarks



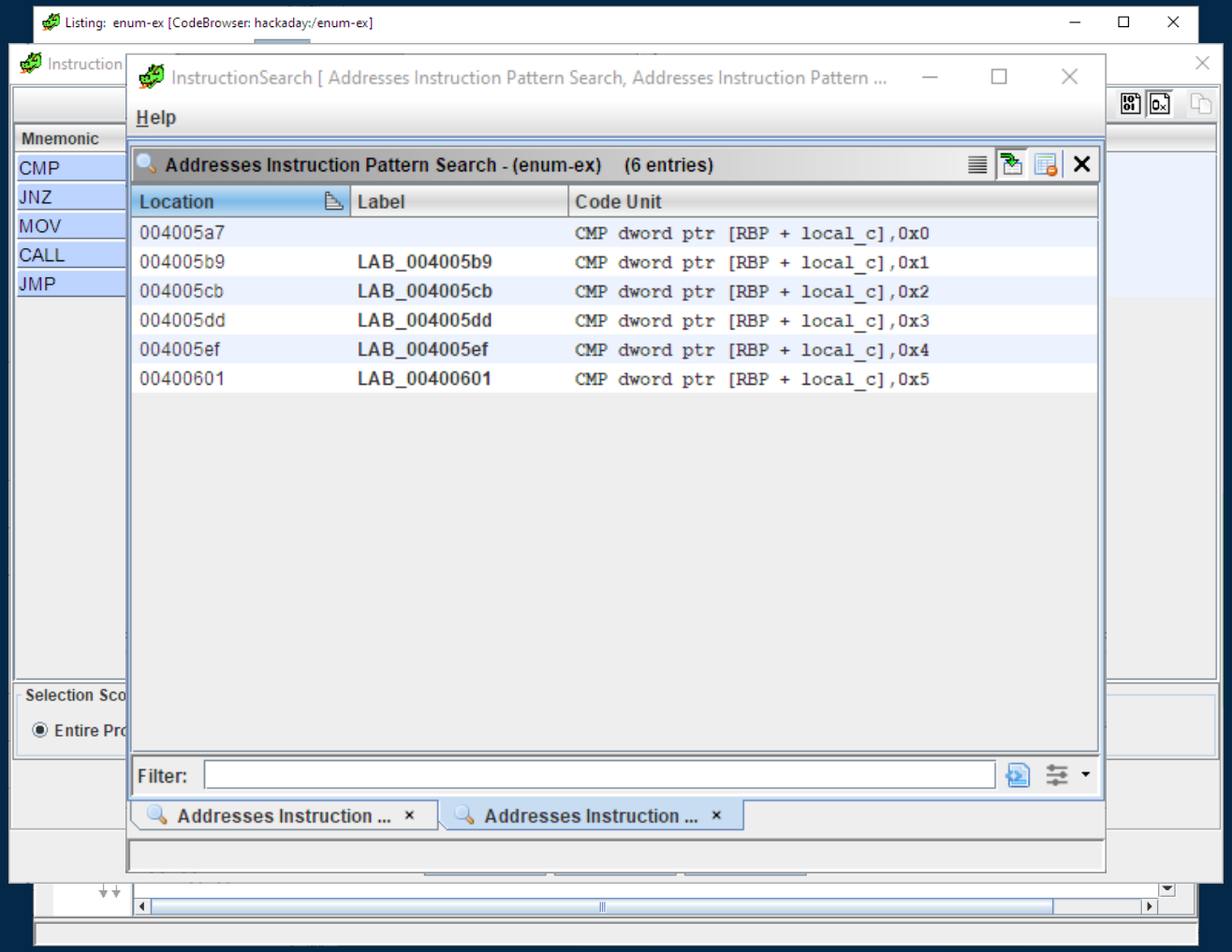
#Ghidra Tip: Searching

- When analyzing a program in Ghidra, you can search for various data types/patterns
  - Instructions
  - Scalars (immediate values)
  - Direct references
  - Instruction patterns
  - Strings
- Both program text and memory can be searched
  - Program Text = Program database or Listing display
  - Memory = entire address space

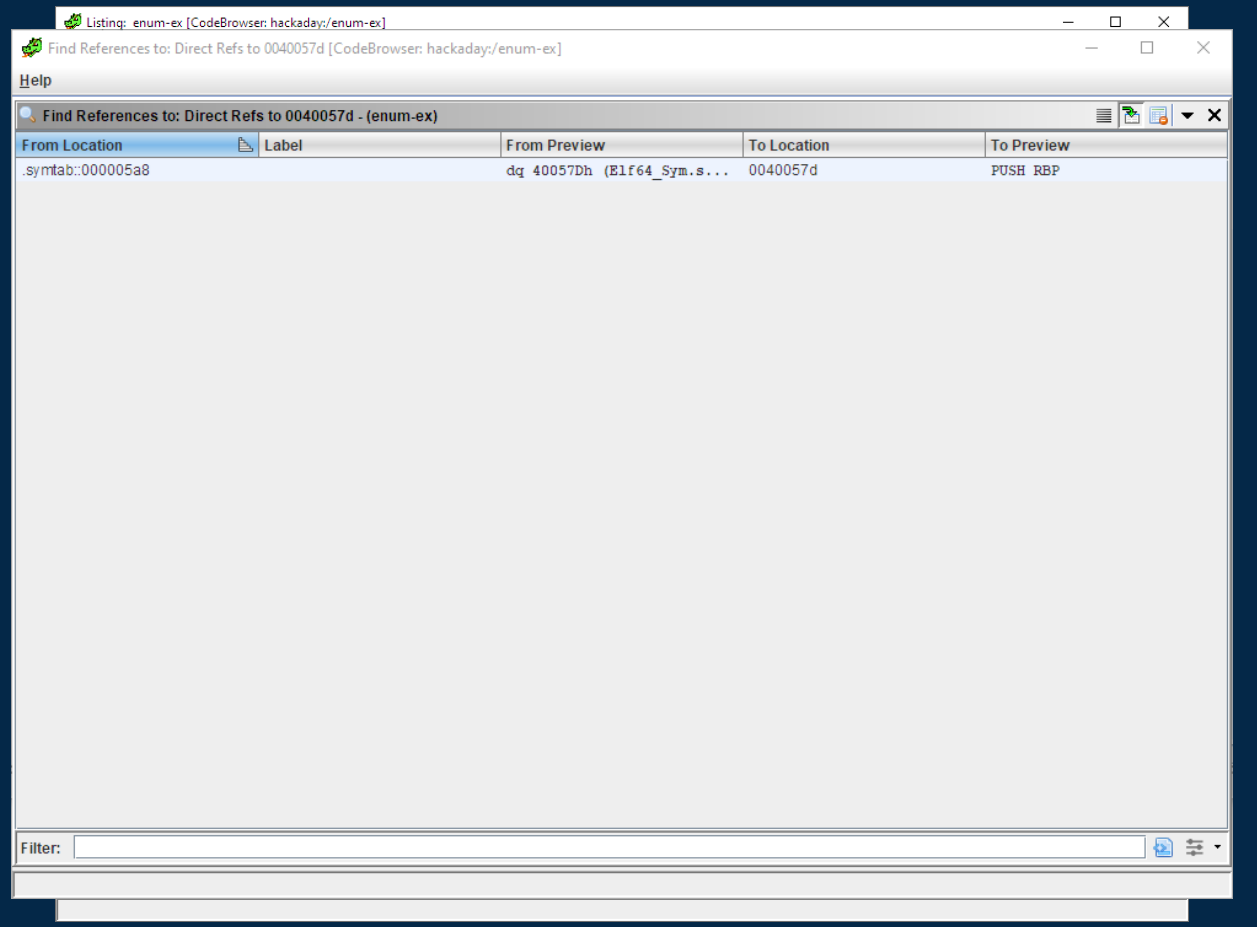
#Ghidra Tip: Searching

- Instructions
  - Allows the user to search for uses of a particular instruction
- Direct References
  - Searches the program for references to that particular address / location
- Instruction Patterns
  - Searches for user specified sequences of instructions in program
- Scalars
  - Search for immediate value
- Strings
  - Search for defined strings

#Search: Instruction Patterns



#Search: Direct References

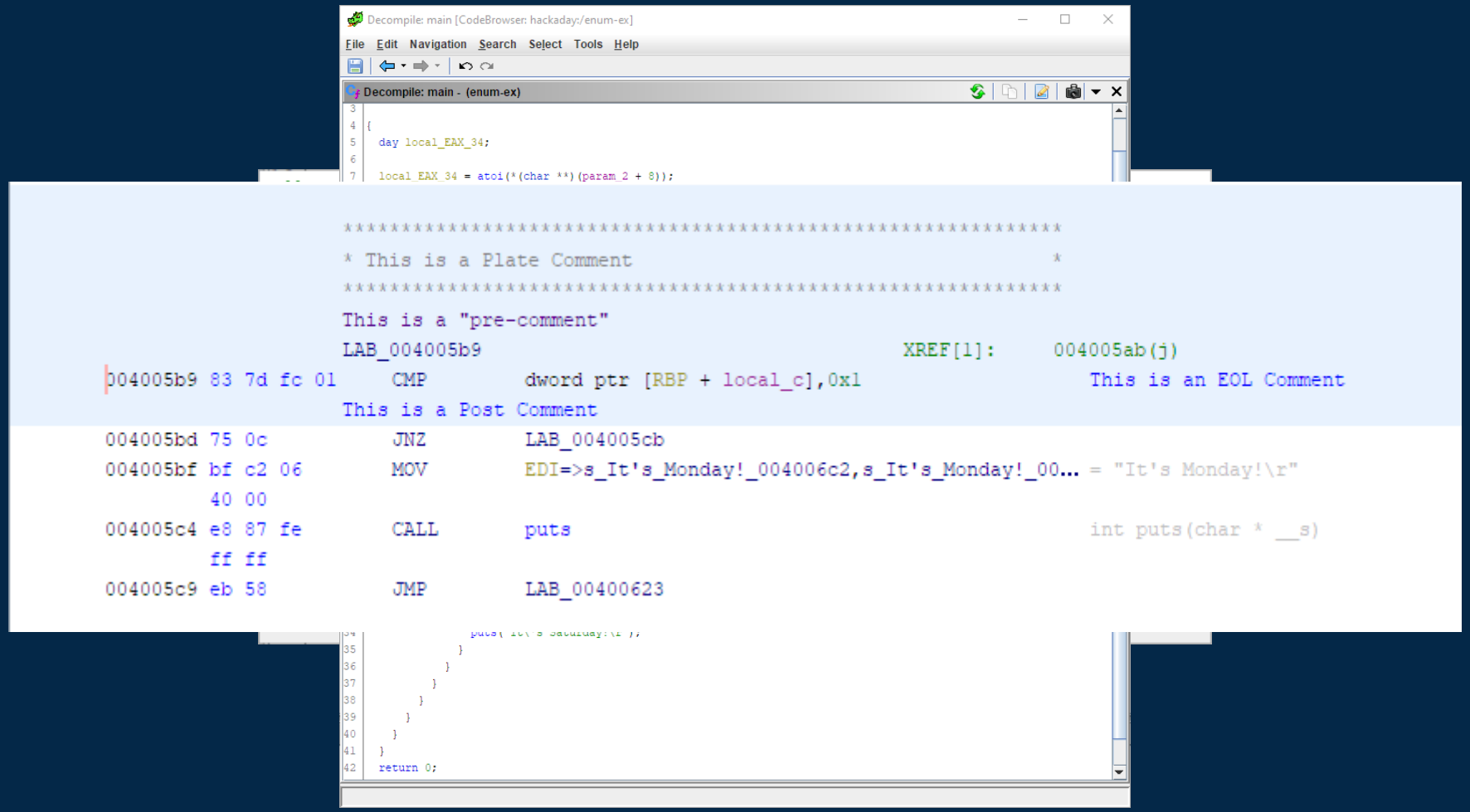


#Ghidra Tip: Comments

- Comments can be added to both the listing view and the decompiler view

- There are various types of comments that can be added:
  - Post
  - Plate
  - EOL
  - Pre
- To add a comment, right click and select “Comment” or press “;”

#Ghidra Tip: Comments



#Wrap Up

- Today we reviewed how to identify various C constructs in C
  - Even though you have a decompiler, it is important to be able to recognize these constructs!
- We also reviewed multiple Ghidra tips to make the reversing process more streamlined
- The exercises for this course are available on the github page!

#Questions

?

#Ghidra Tip: Checksum Tool

#External Libraries

#External Libraries: Ghidra