**Reverse Engineering Overview**

Reverse engineering is the process of taking compiled machine code program and converting it back into more human readable code. Essentially, we are trying to understand the functionality of the program.

Machine code or assembly code have been formatted for direct execution of a CPU. In general, converting to machine code is a one-way process for compiled languages. Machine code can be converted back into assembly however, it is more difficult to read and requires practice.

Assembly instructions perform various actions on registers. These include data movement, arithmetic operations and control-flow. Instructions are pieces of memory executed based on its address. Control flow is achieved by jumping or accessing instructions stored at a certain address. Addresses itself can be likened to the indices of an array, where the array is memory and memory addresses are the indices acting as a reference to an instruction.

The language to be used will be C and C++.

Disassemblers are tools which revert a machine code into assembly code. Various disassemblers are IDA, Binary Ninja and GNU Debugger (GDB). Binary Ninja was chosen mostly due to its aesthetics in presenting assembly code and it has a free version.

Decompilers attempt to convert compiled code back into pseudocode for further reconstruction. This gives a general outline of what the original source code looked like.

**Assembly Cheat Sheet**

For conditional jump instructions on signed data for arithmetic operations.

|  |  |
| --- | --- |
| **Instruction** | **Description** |
| JE/JZ | Jump Equal or Jump Zero |
| JNE/JNZ | Jump not Equal or Jump Not Zero |
| JG/JNLE | Jump Greater or Jump Not Less/Equal |
| JGE/JNL | Jump Greater/Equal or Jump Not Less |
| JL/JNGE | Jump Less or Jump Not Greater/Equal |
| JLE/JNG | Jump Less/Equal or Jump Not Greater |

For conditional jump instructions on unsigned data for logical operations.

|  |  |
| --- | --- |
| **Instruction** | **Description** |
| JE/JZ | Jump Equal or Jump Zero |
| JNE/JNZ | Jump not Equal or Jump Not Zero |
| JA/JNBE | Jump Above or Jump Not Below/Equal |
| JAE/JNB | Jump Above/Equal or Jump Not Below |
| JB/JNAE | Jump Below or Jump Not Above/Equal |
| JBE/JNA | Jump Below/Equal or Jump Not Above |

**Register Cheat Sheet**

Note: R prefix is for 64-bit, E prefix is for 32-bit, and neither in front is 16-bit.

**General Registers**

|  |  |
| --- | --- |
| Note: “H” and “L” suffix on 8 bit registers stand for high and low byte. | |
| RAX, EAX, AX, AH, AL | Called the accumulator register. It is used for I/O port access, arithmetic, interrupt calls, etc … |
| RBX, EBX, BX, BH, BL | Called the base register. It is used as a base pointer for memory access. Gets some interrupt return values. |
| RCX, ECX, CX, CH, CL | Called the counter register. It is used as a loop counter and for shifts. Gets some interrupt values. |
| RDX, EDX, DX, DH, DL | Called the data register. It is used for I/O port access, arithmetic, some interrupt calls. |

**Segment Registers**

|  |  |
| --- | --- |
| Segment registers gold segment address of various items. Only in available in 16 values and can only be set by a general register or special instruction. | |
| CS | Holds the code segment in which the program is run. Changing its value may make the computer hang. |
| DS | Holds the data segment that your program accesses. Changing its value might give erroneous data. |
| ES, FS, GS | These are extra segment registers available for far pointer addressing like video memory and such. |
| SS | Holds the stack segment the program uses. Sometimes has the same value as DS. Changing its value can give unpredictable results, mostly data related. |

**Indexes and Pointers**

|  |  |
| --- | --- |
| Indexes and pointers and the offset part of and address. They have various uses, but each register has a specific function. They are sometimes used with a segment register to point to far address. Register with an “E” prefix can only be used in protected mode. | |
| ES: RDI, EDI, DI | Destination index register. Used for string, memory array copying and setting and for far pointer addressing with ES. |
| DS: RSI, ESI, SI | Source index register. Used for string and memory array copying. |
| SS: RBP, EBP, BP | Stack base pointer register. Holds the base address of the stack. |
| SS: RSP, ESP, SP | Stack pointer register. Holds the top address of the stack. |
| CS: EIP, EIP, IP | Index pointer. Holds the offset of the next instruction. It can only be read. |

**Function Parameters**

|  |  |
| --- | --- |
| RDI | First argument. |
| RSI | Second argument. |
| RDX | Third argument. |
| RCX | Fourth argument. |
| R8 | Fifth argument. |
| R9 | Sixth argument. |

**Variables**

|  |  |  |
| --- | --- | --- |
| Bit | None |  |
| Byte or Octet | Char | 1 |
| WORD | Short | 2 |
| DWORD | Integer. | 4 |
| QWORD | Long. | 8 |