# Tools:

* 32bit debugger: <https://immunityinc.com/products/debugger/>
* 64bit debugger: <https://x64dbg.com/#start>

## GCC/G++:

* -c <file> - Build obj file
* -g - Build with extra symbols for gdb
* -o <name> - Output file
* -m32 - Compile 32bit executable (may need sudo apt install g++-multilib)

## gdb:

* set disassembly-flavor intel
* info proc mappings - list of mapped memory regions
* maintenance info sections - list sections
* info files - list sections of ALL loaded binaries
* eval "x/%dwx $esp", ($ebp-$esp)/4 – Print whole stack
* disas main, then: b \*1st Address - Set breakpoint at first instruction in main (AFTER START)

# Assembly:

Intel processor use Litte-Endian: Store bits in reverse order.

## Sections:

* .text: Stores all instructions. Access: Read, and Execute (Protect against hackers)
* .data: Stores global initialized variables. Access: Read, and Write
* .bss: Stores global UNINITIALIZED variables. Access: Read, and Write
* .rodata (read only data): Stores constant variables (const char\* msg = “hello”)
* Every dll/exe used by the program will have their own sections (text, data etc)

## Data types:

* Byte (DB): 1 byte (8bit)
* Word (DW): 2 bytes (16bit)
* Double Word (DD): 4 bytes (32bit)
* Quad Word (DQ): 8 bytes (64bit)

## Memory segments:

From low to high addresses:

* .text
* .data
* .bss
* Heap
* Stack (Grows up?)

# Registers:

## General Registers (EAX, ECX, EDX, EBX):

* EAX: Accumulator register, used to manipulate data (addition, subtraction etc) and interrupt calls.
* ECX: Counter register, used as a counter for loops and shifts.
* EDX: Data register, used together with EAX for calculations, and some interrupt calls.
* EBX: Base register, used as a base pointer for memory access, and get some return values from interrupt calls

## Stack Registers (ESP, EBP):

* ESP: Stack pointer register, holds the top address of the stack (top of the stack).
* EBP: Stack Base pointer register, holds the base address of the stack (bottom of the stack).
* EIP: Instruction pointer, stores the address of the next instruction.

## Pointer Registers (ESI, EDI):

* ESI: Source index register, used for string and memory array copying. Ex: Holds the location of a string.
* EDI: Destination index register, stores function pointers, and string addresses.