x86_32 Assembly + GDB Quick Reference

Registers (32-bit)

eax, **ebx**, **ecx**, **edx**, **esp** (stack pointer), **esi**
Each register holds 32 bits. You can access smaller parts of some registers:
eax = 32 bits
ax = lower 16 bits of eax
ah = upper 8 bits of ax
al = lower 8 bits of ax

Essential GDB (GEF) Commands

start → Creates a breakpoint at program start and runs until there.
si → Step into the next assembly instruction (line by line).
quit → Exit GDB.

Common Assembly Instructions

mov eax, 2 → Move value 2 into eax.
push eax → Push value in eax to the top of the stack.
pop eax → Pop top of stack into eax.
call <function> → Jump to function and store return address on stack.
ret → Return to the address stored on the stack.
jmp <label> → Jump to label, nothing done to stack.
cmp eax, 4 → Compare eax to 4 (sets flags for next jump).
je <label> → Jump if equal.
jz <label> → Jump if zero.
jnz <label> → Jump if not zero.
jl <label> → Jump if lower.
jg <label> → Jump if greater.
add ecx, ebx → ecx = ecx + ebx.
sub ecx, ebx → ecx = ecx - ebx.
dec ecx → Decrement ecx by 1.
inc ecx → Increment ecx by 1.
mul ecx → eax = eax * ecx.
div ecx → eax = eax / ecx; remainder in edx (zero edx first!).
idiv ecx → Same as div, but for signed integers.
xor ecx, ecx → Bitwise XOR; here used to zero a register (same as mov ecx, 0).

Functions

my_function
.my_inner_function
Nested or inner functions may appear inside larger ones.

Interrupts

An interrupt hands control to the operating system for privileged tasks.
Example:
int 0x80 → Used for system calls (e.g., write, exit).
mov eax, 1; int 0x80 → Exit program safely.