Hello World in x64 Assembly - A Beginner’s Gateway to the CPU

Table of Contents

1. Introduction: Why Learn Assembly?

Assembly is like learning the CPU’s secret language! It lets you give direct instructions to the processor.

Challenging? Yes. **Powerful?** Absolutely!

Unlike high-level languages, Assembly interacts directly with the hardware, allowing fine-grained control over performance and memory management. It’s widely used in low-level programming, reverse engineering, and cybersecurity.

How Assembly Works

Each instruction maps to physical CPU operations. For example:

mov rcx, -11

This command moves the value **-11** into a register (RCX), which acts like a small, ultra-fast storage location inside the CPU.

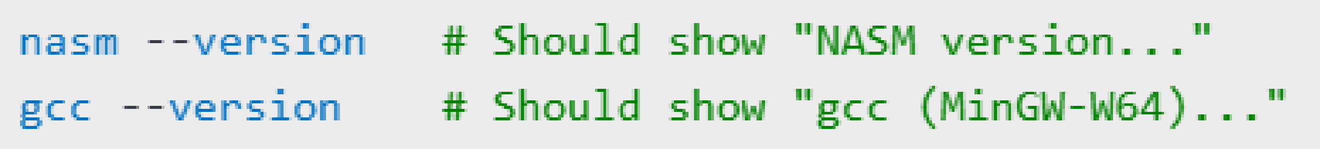
2. Tools & Setup

Required Tools

- NASM (Assembler) → Translates your code into machine language.
- MinGW-w64 (Compiler) → Converts object files into .exe executables.
- **Text Editor** (VS Code, Notepad++, etc.).

Configuration

1. Install NASM and add it to PATH during installation.
2. Install MinGW-w64 (select x86_64 architecture).
3. Verify in PowerShell:

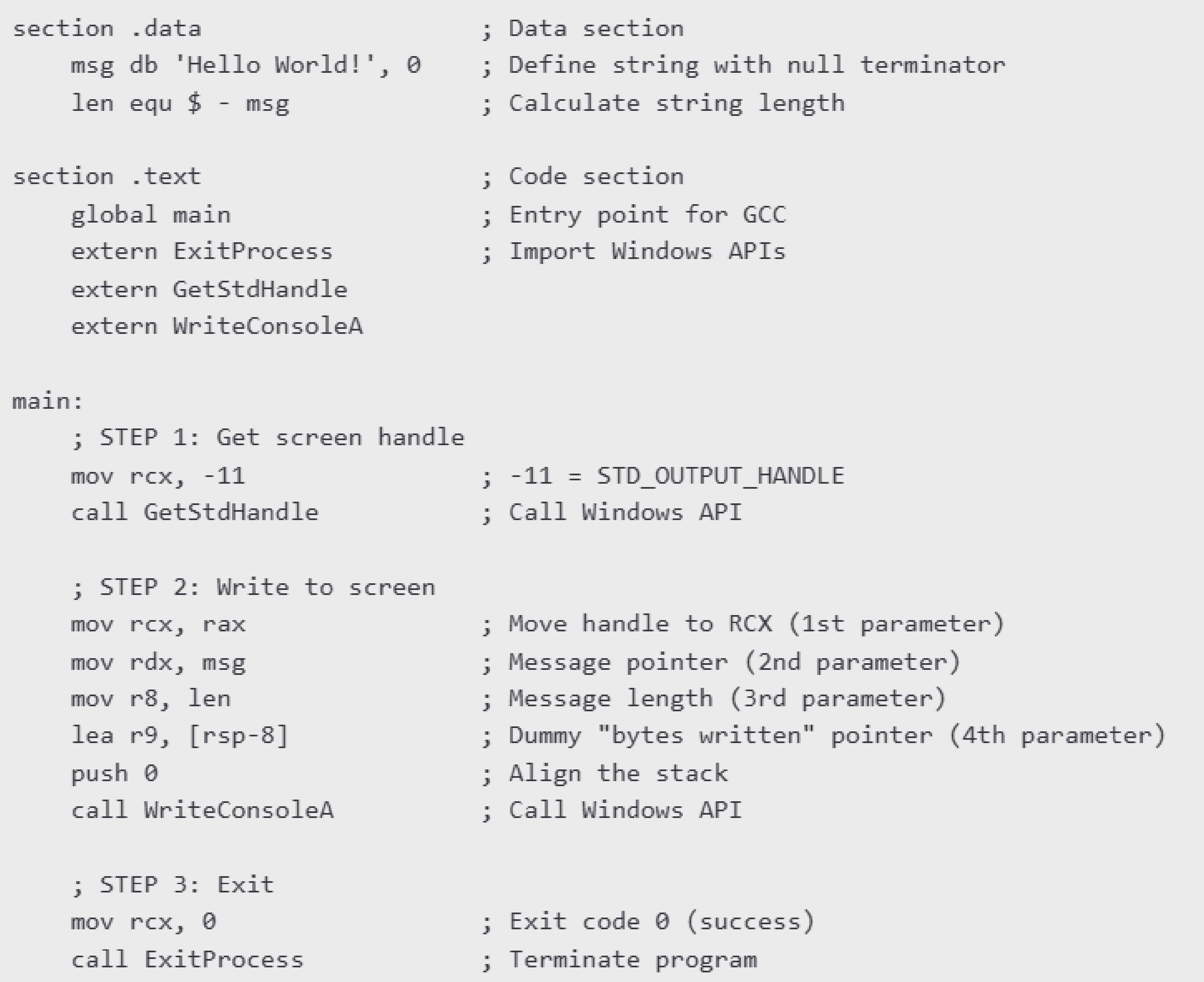


```
nasm --version    # Should show "NASM version..."
gcc --version     # Should show "gcc (MinGW-W64)..."
```

nasm --version   # Should show "NASM version..."

gcc --version   # Should show "gcc (MinGW-W64)..."

3. Line-by-Line Code Breakdown



```
section .data                  ; Data section
    msg db 'Hello World!', 0    ; Define string with null terminator
    len equ $ - msg             ; Calculate string length

section .text                  ; Code section
    global main                ; Entry point for GCC
    extern ExitProcess         ; Import Windows APIs
    extern GetStdHandle
    extern WriteConsoleA

main:
    ; STEP 1: Get screen handle
    mov rcx, -11               ; -11 = STD_OUTPUT_HANDLE
    call GetStdHandle          ; Call Windows API

    ; STEP 2: Write to screen
    mov rcx, rax               ; Move handle to RCX (1st parameter)
    mov rdx, msg               ; Message pointer (2nd parameter)
    mov r8, len                ; Message length (3rd parameter)
    lea r9, [rsp-8]            ; Dummy "bytes written" pointer (4th parameter)
    push 0                     ; Align the stack
    call WriteConsoleA         ; Call Windows API

    ; STEP 3: Exit
    mov rcx, 0                 ; Exit code 0 (success)
    call ExitProcess           ; Terminate program
```

CODE IN REPOSITORY!

Key Concepts:

- **-11:** A magic number Windows uses for the standard output (screen).
- **Registers:** RCX, RDX, R8, R9 are used for parameters in Windows x64.
- **Stack Alignment:** Windows requires the stack to be **16-byte aligned**.
- **Calling Conventions:** Windows uses the fastcall convention for function calls, passing parameters via registers.

4. Compilation & Execution

Step 1: Assemble the Code

```
nasm -f win64 hello.asm -o hello.obj
```

- -f win64: Specifies 64-bit Windows format.

Step 2: Link with GCC

```
gcc hello.obj -o hello.exe -lkernel32
```

- -lkernel32: Links the Windows API library.

Step 3: Run!

```
.\hello.exe
```

Expected Output:

```
Hello World!
```

5. Common Errors & Fixes

Error	Meaning	Fix
nasm: error: file not found	Missing .asm file	Use cd to navigate to the correct folder
undefined reference to 'WriteConsoleA'	Missing kernel32 link	Add -lkernel32 to the GCC command
Segmentation fault	Invalid memory access	Double-check register usage
Incorrect stack alignment	Unaligned function call	Ensure 16-byte alignment before calls

6. Glossary of Terms

- **Register:** Small, fast memory locations inside the CPU (e.g., RCX, RDX, RAX).
- **API (Application Programming Interface):** Pre-built functions provided by the OS, like WriteConsoleA.
- **Stack:** A special region of memory used for function calls, growing downwards.
- **Memory Addressing:** The method by which data is accessed in memory (direct, indirect, indexed, etc.).
- **Opcode:** The binary representation of an Assembly instruction.
- **Calling Convention:** The rules dictating how parameters are passed to functions in Assembly.

This document is designed for beginner-friendly learning in x64 assembly.