Summary

1 min

Congratulations on making it through the Assembly Language lesson!

Assembly languages play an important role in understanding how computer programs execute at the hardware level. Seemingly simple operations like our square() function actually require many more steps when translated into individual

Preview: Docs Loading link description

processor

tasks.

Learning how these larger programs increase or decrease a processor's workload can significantly affect the performance of a program.

As we write in a high-level programming language, our code leaves our editor and executes the steps of the compilation process in order to make our code functional.

Most of this process has been abstracted for general software developers but can still be accessed in order to optimize specific portions of your program. The translation to Assembly is one of the steps code takes on its journey to your

Preview: Docs Loading link description

computer hardware

.

While first written to make it easier for early programmers to write code, Assembly now primarily serves the purpose of testing hardware applications or providing precise control of limited capacity embedded systems.

Assembly languages very closely mimic the

Preview: Docs Loading link description

machine code

and ISAs will show references to both Assembly and

Preview: Docs Loading link description

Binary

instructions when defining functionality.

Instructions always start with an opcode that defines the operation to be performed and follows that with the operands, or memory locations, to be operated on.

Some of the basic types of operations that make up an Assembly program are:

- Arithmetic (Add/Subtract)
- Memory Access (Load / Store)
- Preview content is loading

Control Flow

(Branch/Jump)

Instructions

Looks like our code is shipped and ready for delivery to the hardware!