An Introduction

Here's a slightly correlated set of essays I've written. References are provided in the style of the American Mathematical Society This document is provided in , — mostly , and plain textual formats. The contains actual links at each reference, and a typeface much less exposed than Times New Roman.

This document is sel sh, though I'd be pleasantly surprised to receive either positive or negative feedback. My contact information is at the very end, past the long bibliography.

The Impact of the C Programming Language on the Past Years of Computing

The C Programming Language was initially developed between 1969 and 1979. It was further re ned when standardized by the American National Standards Institute in 1989. In the twenty years between birth and standardization, C became the dominant programming language, shaping the environments it entered and the programmers who used it. In the time between standardization and now, the majority of computing systems have been designed to leverage C.

A programming language is a format for expressing logic. The programming language that a computer natively understands, called an "instruction set," is composed of minuscule operations. Across all kinds of digital computers, these allow for logical evaluation, performance of arithmetic, management of information, and general hardware control [Intel], [Arm], and [Waterman].

These instructions are expressed in binary digits, commonly contracted to "bits" [Mackenzie]. Unlike a decimal digit, which may be a whole value between 0 and 9, a bit may be either 0 or 1. In digital computing's infancy, series of these were manually "toggled in" [Post], using physical switches and buttons. Save for theories put on paper, such as the works of Ada Lovelace [Menabrea] and Alan Turing [Turing], this was the rst form of computer programming.

As hardware and software both advanced, programmers became able to write programs textually. They would physically interact with a teleprinter, essentially a typewriter which would send key strokes to and print output from a computer. They would digitally interact with software to manage and store teleprinter input, naturally named "text editors." Then, the stored input from the teleprinter would be passed to a program called an "assembler," to further interpret the saved input. The textual representations of computer instructions, such as " ," would be encoded as real computer instructions, such as '0010000101101010' [Nelson], [Ritchie], [Thompson a], [Thompson b], [Thompson c]. The text which programmers would provide to assemblers came to be called "assembly language." [ ].

Assembly language directly correlates to a computer's instruction set, meaning that the programmer must textually manage each operation in a program. Doing so is less tedious than ip-ping switches hundreds of times over, but still tedious [Nather], [Post]. Computer scientists found that a computer may instead be provided a text which vaguely describes the intended logic of a program. Though the computer has been provided less information, it may have enough to produce a reasonable translation of the abstract program to its own instruction set. As opposed to an assembler, the software which performs this kind of interpretation is called a "compiler" [Aho].