A Structured Approach To FORTRAN 77 Programming: With WATFIV

C. Joseph Sass

A Structured Approach to FORTRAN 77 Programming: with WATFIV

C. Joseph Sass

University of Toledo

Allyn and Bacon, Inc. BOSTON • LONDON • SYDNEY • TORONTO

Library of Congress Cataloging in Publication Data

Sass, C. Joseph.
   A structured approach to Fortran 77 programming
with WATFIV.

   Includes index.
   1. FORTRAN (Computer program language) I. Title.
QA76.73.F25S28 1983   001.64'24   83-2548
ISBN 0-205-07918-0

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1 88 87 86 85 84 83

Preface

This textbook is designed for the introductory FORTRAN course in business, computer science, education, engineering, mathematics, and social science curricula. Its primary purpose is to teach students *how to write programs* in the FORTRAN language. It includes numerous samples and problem assignments from each of the topical areas. Although FORTRAN is formula-oriented, high school algebra is sufficient as a prerequisite to the material. The most current version of the language, FORTRAN 77, plus features available with the WATFIV compiler, are covered in the text in a fundamental format that is easy to read and comprehend.

Pedagogically, several features differentiate this text from others in the market. Each chapter is subdivided into sections, with each section followed by a group of review exercises that pertain to the material just introduced. At the end of each chapter, questions that pertain to the entire chapter are listed. A number of answers to the exercises and questions are included in an appendix at the end of the book. These answers permit the students to review their progress and knowledge of the material.

Within each chapter, numerous program statements are shown. Each of the statements is described along with the common uses and misuses of the instructions. By showing both valid and invalid entries, the novice programmer is guided to the proper implementation of the language rather than gaining the knowledge through a trial-and-error approach that is time-consuming and inefficient.

Logic development is important in an introductory course. An entire chapter is devoted to the subject early in the text. Both the traditional approach to logic development through flowcharting and the structured approach with pseudocode are covered in the chapter. In subsequent chapters, either a flowchart or pseudocode accompanies a complete program listing that represents the solution to a stated problem. As a result of incorporating both logic techniques, instructors are free to choose either method for their class projects.

A building block approach has been implemented in the writing. In the early chapters, relatively simple structured problems and examples are discussed that illustrate basic computer programming logic concepts. This approach allows the student to begin programming at an early stage. As more of the FORTRAN language is introduced, more difficult problems and examples are covered with an emphasis on structured solutions. Techniques and concepts common to most solutions are emphasized in all of the programs, whether they are simple or complex. All of the programs shown in the text have been run and tested thoroughly with different data sets.

Continuity from chapter to chapter is accomplished in several instances by enlarging a previously defined problem and solving or rewriting the older version with the new statements found in the current chapter.

All of the important, frequently used FORTRAN instructions, including file processing statements, are covered in the chapters. The instructions are discussed individually and incorporated in complete program solutions. A few of the less frequently employed statements are discussed individually in the last chapter. These instructions can be covered at an earlier point, if so desired. The appendices also contain other optional material, including a brief history of computers, technical information, a glossary of terms, and selected answers to the exercises and questions.

The author is indebted to many people who have made this work possible. My appreciation is extended to the reviewers, whose worthwhile suggestions have been incorporated throughout; to my graduate assistants Tom Blaser, Tod Burhans, and Ken Nowak, who spent a great deal of time testing and checking the programs included in the book; to my secretary, Betty Mangas, who helps keep things organized and in the proper form; and most of all to my family, without whom the manuscript would never have been completed.

C.J.S.

Contents

1 Introduction to Computers and FORTRAN Processing

The number of computer systems available for use by businesses, government, and society in general is increasing as rapidly as computer costs have been decreasing. In addition to the traditional employment of computers in business, engineering, research and development, and government, computers are now utilized in dissimilar fields such as crime prevention, traffic control, ecological studies, and even bookkeeping tasks at home.

Whenever information is gathered for the purpose of study and statistical investigation, a computer system is beneficial. It has the processing capability of compiling, storing, correlating, sorting, and selecting data rapidly and accurately. Instructions can be fed to the computer that permit simple or complex formulas and algorithms to be applied to voluminous data and transform it into concise, meaningful statistics.

This chapter will introduce the reader to the fundamental concepts of computer systems, hardware, and programming.[1]

1–1 Computer System Functions

The processing cycle common to all systems is comprised of three steps: the *input* cycle, the compute or *processing* cycle, and the *output* cycle. This sequence of operations is performed by a computer system. Five important physical components form the system (Fig. 1–1). These components are often classified singularly as *hardware.* To begin the normal processing cycle, information or data must be entered into the computer to permit mathematical operations and logical manipulations to be performed. The INPUT unit fulfills this need. Devices that function as input units include typewriterlike keyboards, card readers, paper tape readers, and several types of magnetic devices, such as tape and disk units. In general terms, the hardware that function in communication with the computer are identified as *peripheral devices.*

Whereas the input unit passes information into the system, the OUTPUT unit serves to transmit data, usually answers, from the computer to the user. Examples of output peripheral devices include the typewriterlike devices mentioned earlier, card punches, line printers (which print an entire row of charac-

[1] Appendix C contains a glossary of computer terms used frequently throughout the book. Refer to it for definition of new concepts mentioned.

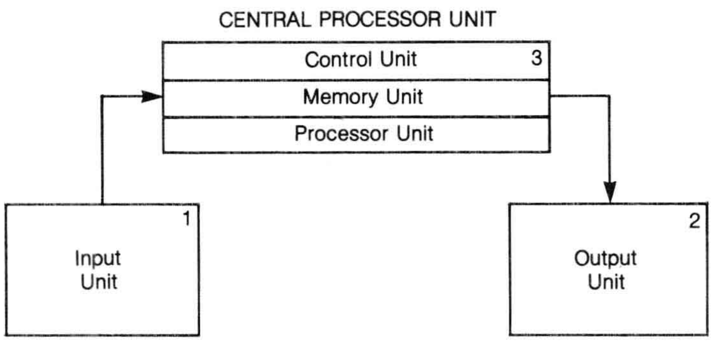


FIGURE 1–1. *Components of a Typical Computer System.*

ters at once), and the same magnetic devices available as input units. Notice that certain units, the keyboard and magnetic devices, may be used for input and/or output, adding to the flexibility of a computer system.

The final components of the system, the MEMORY unit, the CONTROL unit, and the PROCESSOR unit,[2] are called collectively the CENTRAL PROCESSOR UNIT (CPU). Usually, they are physically housed in the same unit, although they perform individual operations. The MEMORY unit is just that: it retains or stores for later use the program instructions, constant data, or data information that is transmitted to the computer by the input unit. It also stores the results of internal calculations carried out by the PROCESSOR unit. All instructions and data must be loaded in memory before they can be manipulated. The CONTROL unit performs the necessary function of coordination. The activities and operations of all the components are monitored and directed by this unit. Thus, it assures that the processing steps specified by the user are executed in the proper sequence. The last component is the PROCESSOR. Mathematical and logical instructions are carried out by this unit. For example, addition, subtraction, and logical or decision tests are performed within the PROCESSOR unit.

Collectively, these five entities operate as a computer system (Fig. 1–2). When combined with different computer programs, the system can be instrumental in the solution of a variety of different tasks. A computer system should be regarded on the same basis as other business equipment—as a specialized tool that can provide many answers for management and other users of the system.

1–2 Programming

What is programming? In general, it is supplying a program or a set of specific instructions or directions that, when performed in a precise, logical sequence, identify to the computer the steps necessary to solve a particular problem. The set of directions is normally written in a high-level, user-oriented programming language; for example, FORTRAN. An ominous trait of programs is that they must be precisely stated in one chosen programming language and written according to specified syntax rules for that language. This is of crucial importance since the computer does exactly what it is instructed to do and nothing else.[3]

[2] The PROCESSOR unit is often called the arithmetic and logical unit (ALU).

[3] Unfortunately for programmers, sometimes the instructions listed do not carry out the intent desired, resulting in the wrong answer to a defined problem.

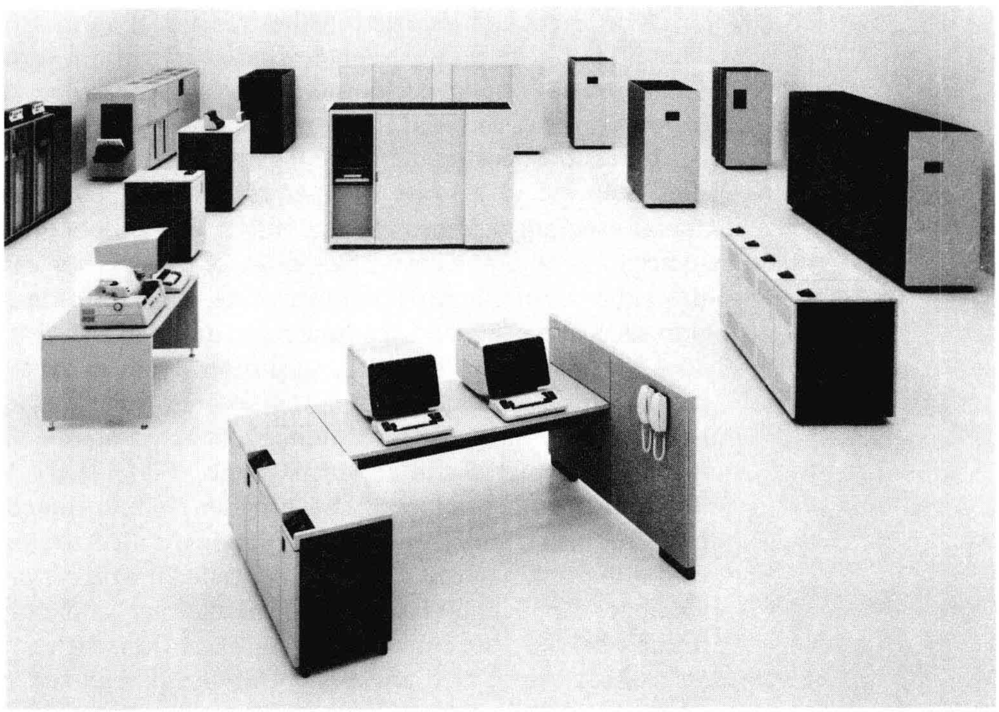


FIGURE 1–2a. *Large-Scale Computer System. (Courtesy of International Business Machines, Corp.)*

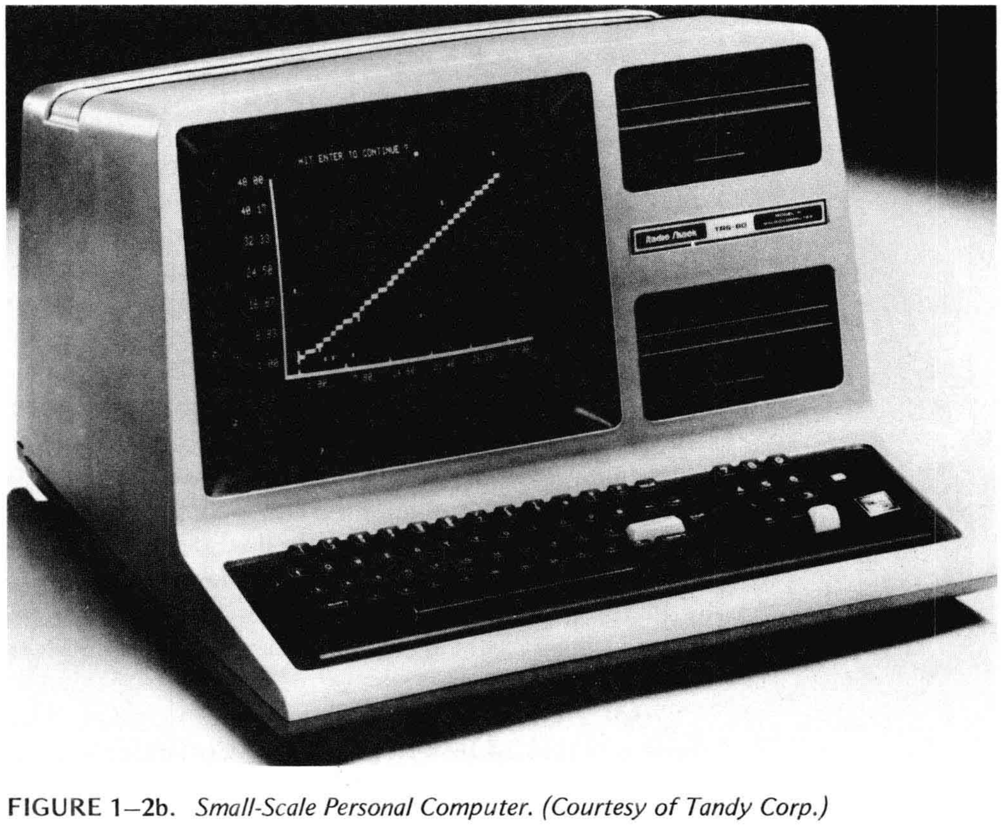


FIGURE 1–2b. *Small-Scale Personal Computer. (Courtesy of Tandy Corp.)*

What steps are involved in the traditional approach to writing a program? The first step is a clear, concise definition of the problem to be solved. An understanding of the task is essential before proceeding. An algorithm or method of solution that will lead to the answers identified earlier is the next step. The algorithm is the development of a logical sequence of directions leading to the desired solution in a finite number of operations. The third step is to start the actual program design or to establish a logical flow of instructions necessary to solve the problem. Either a *flowchart* or *pseudocode* can be the end product of this segment of the programming cycle. Both result in a nonprogramming solution to the problem and are covered in detail in the next chapter.

Once the nonprogramming solution guide is developed, it is the basis for "coding the program." Coding the program follows the logic of step three and involves writing actual programming language statements that conform to the format rules of the chosen language, i.e., FORTRAN. Finally, execution, testing, and documentation of the program follow the coding. Execution means that the programmer-written instructions, called the *source program*, are converted to machine-executable format, called the *object program*, and loaded into memory. Applicable test input, calculations, and output directions are performed next by the computer. After execution, the test results are evaluated to assure that the desired answers are obtained. And last, but not least, documentation is required. This is an ongoing step starting with the problem definition. Pertinent notes and references to the problem are maintained throughout the programming cycle for inclusion in the documentation file. In general, documentation covers two areas: the output and the program. The first pertains simply to identifying with labels or report and column headings the printed answers produced as output. The second is the collection of all the relevant information that applies to the problem. This includes the problem definition, algorithm, flowchart or pseudocode, source program listing with internal comments relating to the problem, and, finally, samples of both the input and the output.

When creating and executing a program for the first time, one of three types of errors can occur. *Syntax* errors are normally the easiest to correct. They involve a misuse of the programming language, for example, misspelled words, illegal symbols, incorrect formats, etc. The system usually identifies the statements in error and does not generate computer output. *Run-time* errors occur during execution of a program, usually resulting in an abnormal, premature end of job. They involve a variety of mistakes, such as division by zero, and are more difficult to find and correct. Output may be generated, determined by the type and location of the error. The most difficult types of errors to locate and correct are *logic* problems. With logic errors the program finishes and appears to have worked, except that upon closer examination the output contains one or more incorrect answers. For example, the programmer may have listed an addition operator when the original definition calls for multiplication. The incorrect answers result from a logic flaw, which must be found and corrected by the programmer. The flaw is often called a "program bug" and finding the error is called "debugging", i.e., removing the logic error. New programmers frequently have one or more errors on the first pass of a new program.

What are the desired objectives of a program? First, it should provide the answers requested by the user (finance officer, chief engineer, or whomever) in a presentable form. The task should be completed on time and within the estimated cost. It should be accurate and capable of processing any reasonable set of data supplied as input. Finally, it should be written in such a manner that it can be easily revised, since many programs undergo changes throughout their life cycle.

1–3 FORTRAN Principles

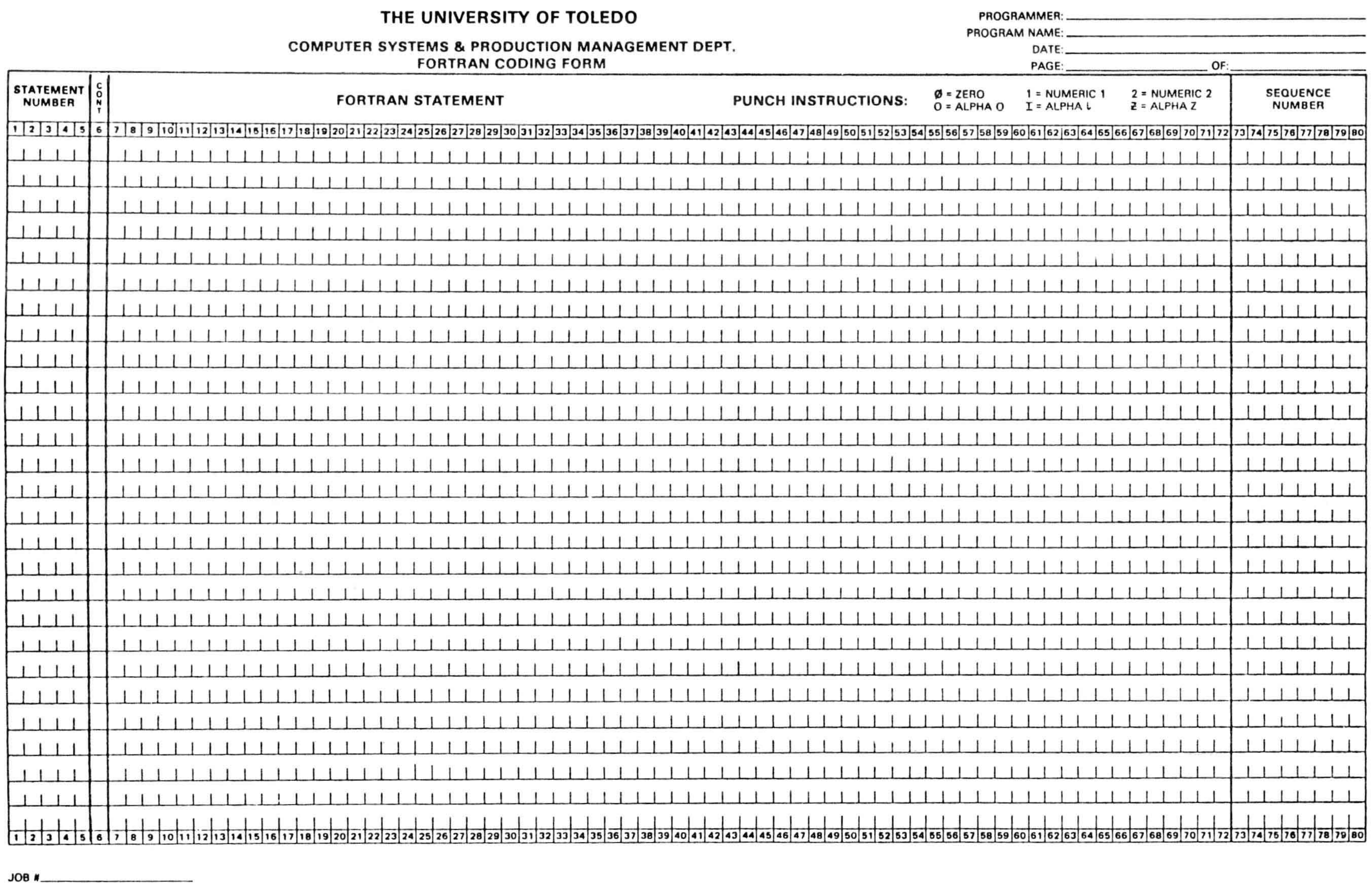
The programming language FORTRAN (an acronym for FORmula TRANSlation) was originally developed in the mid-1950s as a scientifically oriented language. It has been refined and revised a number of times since, with the latest changes resulting in new standards adopted in 1977. FORTRAN 77 refers to these language modifications adopted by the American National Standards Institute (ANSI). FORTRAN 77 and a companion version of the language, WATFIV,[4] are discussed concurrently throughout the book. Many of the statements are identical, but a few are different. When the differences occur, they will be identified. The FORTRAN language, in general, is universal in that it can be used on different manufacturers' systems and is employed not only in the United States but abroad as well. Its usage is not limited to engineers and mathematicians but is seen in business applications, research and statistical applications, and educational environments, too.

The language is made up of certain key words, symbols, and phrases that are placed on a coding sheet (Fig. 1–3) in predefined areas. The specific areas are defined in Fig. 1–4. Each FORTRAN instruction represents one of six different types of statements. *Arithmetic* statements assign a constant, a variable, or the result of a calculation to another variable.[5] *Input/Output* statements process data that must be entered into the system or produce the output that has been generated by the computer. A FORMAT statement specifies the exact appearance of the input or output data. *Control* statements direct the logic of a particular program. Decision routes, various options in a program, and even termination of the run are normally within the spectrum of the control statements. *Specification* entries identify the properties of other statements listed in a program. For example, integer data, real data, and even alphanumeric data can be manipulated. These three types of data have different properties as illustrated later on. Finally, separate routines or commonly used procedures can be established with the *subprogram* statements.

Within each statement are found *constants*, *variables*, and *expressions*. Numeric constants may be positive, negative, or zero. In Chapter 3 examples of the various types are listed. Nonnumeric constants are also permitted. They are covered later in the text, too. A variable represents a location in the memory unit of the computer. Numeric and nonnumeric data can be stored in these locations. An expression is a formula comprised of variables, constants, and special symbols that represent the standard math operations. These, too, are detailed in Chapter 3.

[4] WATFIV is an enhanced version of WATFOR, WATerloo FORtran, developed at the University of Waterloo. Both versions are similar to standard FORTRAN, except that they have enhancements appealing to the academic environment.

[5] Valid constants, variables, and calculations are defined in the next chapter.



THE UNIVERSITY OF TOLEDO

COMPUTER SYSTEMS & PRODUCTION MANAGEMENT DEPT.
FORTRAN CODING FORM

PROGRAMMER: ____________
PROGRAM NAME: ____________
DATE: ____________
PAGE: ________ OF: ________

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | PUNCH INSTRUCTIONS: Ø = ZERO, O = ALPHA O; 1 = NUMERIC 1, I = ALPHA I; 2 = NUMERIC 2, Ƶ = ALPHA Z | SEQUENCE NUMBER

1 2 3 4 5 | 6 | 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 | 73 74 75 76 77 78 79 80

JOB # ____________

FIGURE 1–3. *FORTRAN Coding Sheet*

FIGURE 1–4. *FOR* *oding Sheet Areas*

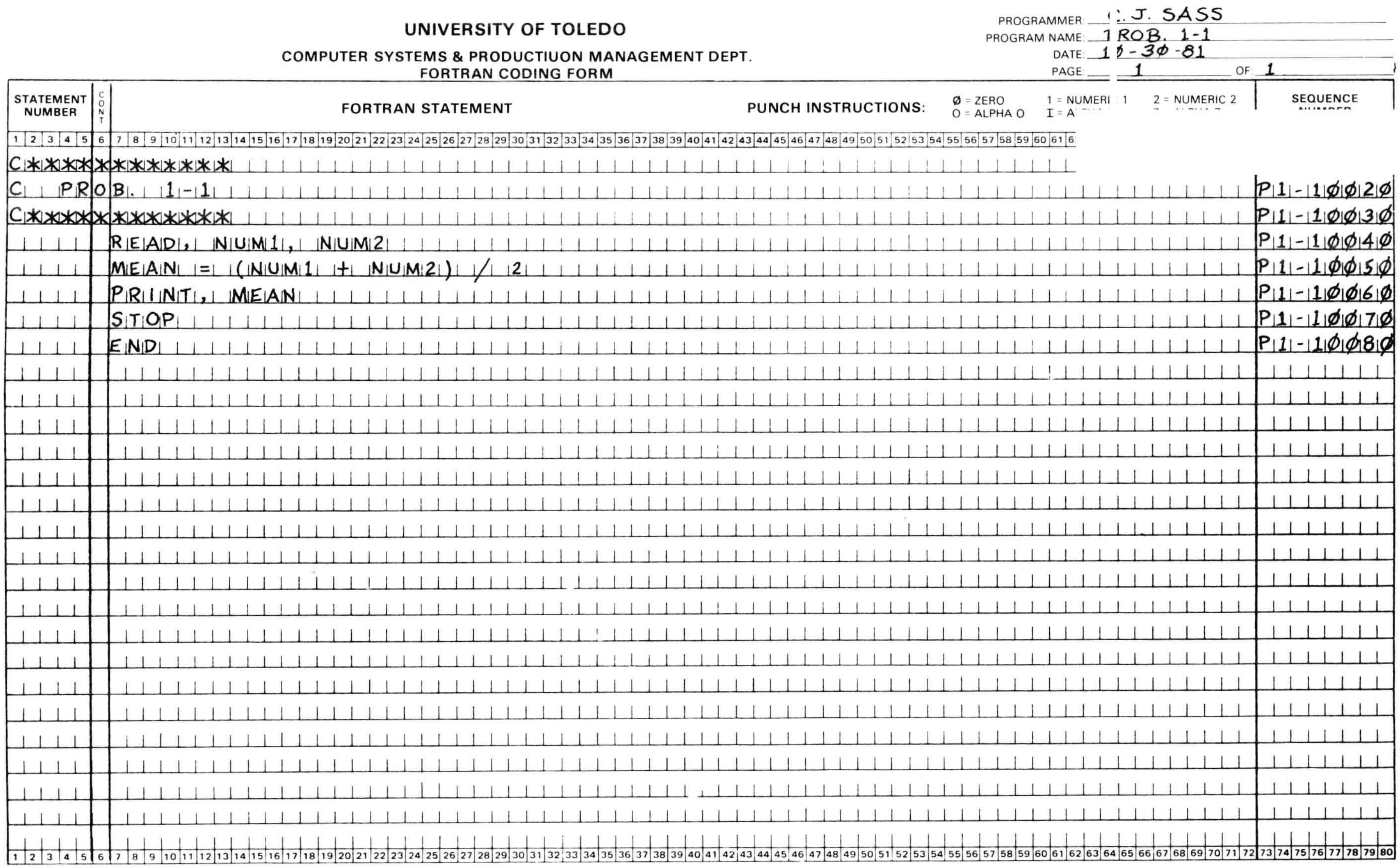
Columns	*Description*
1	When a C in column 1, the instruction is a comments entry. It provides the reader with information, but it issu rect action for the computer to follow. Thus, it is called a nonexecutable statement. The programm nter any notes or remarks after the C.
1-5	If there is in column 1, a statement label can be placed in columns 1-5. Leading and trailing blanks are ignore leading zeros. Only integer values are permitted in this field. Insertion of a label allows the statem e referenced by another FORTRAN statement.
6	FORTRA nents that cannot be completed on one line can be continued to subsequent lines by placing an ro character in column 6 of the succeeding lines.
7-72	The actual FORTRAN instruction is placed in columns 7-72. It is comprised of specific symbols, phrases, and key words, plus blanks. The blanks can be inserted anywhere in the area to improve readability.
73-80	An optional field that can name the program and sequence the instructions. When listed, columns 73-76 usually name the program and columns 77-80 number the statements consecutively.

1–4 A FORTRAN Sample

A short sample is given in Fig. 1–5. It processes two input data values, computes the numeric average of the two, and generates as output the result of the calculation. The first three statements are comment entries. Statements that begin with a C in column 1 are for documentation purposes only. In this case, the program is identified as being the first one in Chapter 1. Comment entries do not direct the computer to perform any action as do the remaining entries in the program. The READ statement causes the system to obtain values for the two variables labeled NUM1 and NUM2. This statement and the ones that follow must be placed within columns 7 through 72. Also note that the two commas in the READ statement are a required part of the syntax for the FORTRAN language. Or, in other words, without them a syntax error would occur and the program would not work. However, the blanks after the commas are not required. Optionally, they make the statement easier to read. After the READ, the next statement is an assignment entry that contains an expression. The formula to the right of the equal sign is evaluated, and the result is assigned to the variable MEAN. Output from this program is produced by the PRINT instruction that follows, while the STOP statement terminates the execution of the program. All FORTRAN programs must have the END statement as the last instruction in the source listing.

The instructions here illustrate that it is relatively easy to piece together simple FORTRAN programs. From this starting point, longer, more complex samples will be derived in later chapters.

Summary The computer is a flexible tool that can be controlled by creating effective programs in a high-level programming language. FORTRAN is one of these languages, used in engineering, math, business, and other environments. Several fundamental steps are a prerequisite to writing effective programs, with problem definition, design, and logic essential parts of the task.



UNIVERSITY OF TOLEDO

COMPUTER SYSTEMS & PRODUCTIUON MANAGEMENT DEPT.
FORTRAN CODING FORM

PROGRAMMER: C. J. SASS
PROGRAM NAME: PROB. 1-1
DATE: 1Ø-3Ø-81
PAGE: 1 OF: 1

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | PUNCH INSTRUCTIONS: Ø = ZERO, O = ALPHA O; 1 = NUMERI 1, I = A; 2 = NUMERIC 2 | SEQUENCE

```
C************                                                           P1-1ØØ2Ø
C   PROB. 1-1                                                           P1-1ØØ2Ø
C************                                                           P1-1ØØ3Ø
      READ, NUM1, NUM2                                                  P1-1ØØ4Ø
      MEAN = (NUM1 + NUM2) / 2                                          P1-1ØØ5Ø
      PRINT, MEAN                                                       P1-1ØØ6Ø
      STOP                                                              P1-1ØØ7Ø
      END                                                               P1-1ØØ8Ø
```

FIGURE 1–5. *FORTRAN Sample*