DESMONDE

Real-Time Data Processing Systems:

Introductory Concepts

PRENTICE-HALL SERIES IN AUTOMATIC COMPUTATION

REAL-TIME DATA PROCESSING SYSTEMS:

INTRODUCTORY CONCEPTS

WILLIAM H. DESMONDE

Research Staff Member
IBM Corporation

PRENTICE-HALL, INC.
ENGLEWOOD CLIFFS, N. J.

Current printing (last digit):
12 11 10 9 8 7 6 5 4 3

Library of Congress Catalog Card No. 64-19676
Printed in the United States of America
76709 C

ACKNOWLEDGMENTS

I would like to thank the numerous individuals who have helped me gather material on real-time systems, either by discussing problems with me, or by furnishing me with reports and manuals.

In studying the Sabre project, I was aided by Roy V. Bigelow, John H. Brummer, Roy E. Cicale, William B. Elmore, M. Gerson Ginzberg, Robert V. Head, James L. Kessler, Alfred A. Scaia, John Siegfried, and Cecil P. Webb. I obtained information concerning other airline reservation systems from Glen Albers, Mrs. Barbara Allen, Francis M. DeKalb, Donald Liebelt, Edward D. Liljegren, Patrick A. Petty, and Robert Sobecki. On many topics, explanations by H. F. Scheuer were particularly helpful.

Robert L. Hoffman, John P. Lamb, and Gerald M. Weinberg gave me assistance in understanding Project Mercury. The diagram I use in the text for presenting the Mercury control program is adapted from a chart in Project Mercury final reports.

Much of the material relating to the Programmed Transmission Control is drawn from Nicholas Sternad's "Programming Considerations for the 7750" (*IBM Systems Journal,* March 1963) and other manuals prepared by Dr. Sternad.

A great deal of the information in the chapter on testing real-time systems is contained in a special report prepared by Charles Barbel, James Egglezos, James Martin, and Thomas P. Taylor. Glen Albers, Edward D. Liljegren, and G. M. Weinberg provided data on both the testing and planning of real-time systems.

I have drawn considerable material on file organization from write-

ups, classes, and seminars provided by the IBM Department of Education.

William R. Elmendorf has read a large part of the book, and has given me a number of useful suggestions. Numerous individuals within IBM reviewed portions of the manuscript and made helpful recommendations.

The material for this book was obtained from a wide variety of sources, over a long period of time. My apologies if I have forgotten to thank anyone for his assistance.

The development of real-time systems has been the result of the creativity and labor of a very large number of persons. It would be impossible for me to attempt in this book to cite all of the contributors, major and minor, to the projects I have discussed. The basic ideas in the Sabre control program were created by John Siegfried and James L. Kessler. Gerald M. Weinberg and Mrs. Marilyn Scott originated the multiprogramming monitor at Mercury. As stated in the text, Geoffrey Gordon developed the general-purpose simulation program which I have described.

TABLE OF CONTENTS

1 INTRODUCTION, 1

Purpose of book, 1
The Mercury project, 2
Real-time commercial applications, 3
Multiple-user computational centers, 5
Other application areas, 6
Military real-time processing, 7
Industrial process control, 9

2 THE ENVIRONMENT OF REAL-TIME SYSTEMS, 11

Real-time data processing, 11
Throughput time, 12
The data communication channel, 13
Interrupt subroutines, 14
Terminal sets, 16
Terminal interchanges, 17
Operations of terminal interchanges, 18
The minimization of communication networks, 20

3 MULTIPROGRAMMING FOR REAL-TIME SYSTEMS, 23

Overlapping of input-output with internal processing, 23
Data channels, 24
Trapping, 25
Multiprogramming, 26
Control programs, 26
The CPU loop, 28

4 REAL-TIME STORAGE ALLOCATION, 33

Allocation of storage, 33
List structures, 34
Memory protection, 38
Program relocation, 39
Program fragmentation, 41
The one-level store, 44

5 THE LIFE OF A TRANSACTION IN SABRE, 47

Entry blocks, 47
Storing information in an entry block, 49
The life of transactions, 51
The FIND and WAIT macros, 52
File requests, 53
Allocation of storage to inputs, 54
Assembly of data from terminals, 55
Transfer of control to another segment, 57
Storage allocation with 7080's, 59
7080 entry blocks, 60

Simultaneous updating of the same record, 61
Machine errors and program errors, 62

6 PRIORITY PROCESSING IN PROJECT MERCURY, 65

Priorities among programs, 65
The priority mechanism, 66
The operation of the priority mechanism, 68
The control program, 70
Program space allocation, 72
Multiple requests for the same program, 73
The Gemini project, 74

7 ON-LINE COMPUTING CENTERS, 77

The problem of turn-around time, 77
Storage allocation, 79
System configuration, 81
The control program, 82
Replacement algorithm, 83
Other functions of the monitor, 86
Man-machine communication, 87

8 THE PROGRAMMED TRANSMISSION CONTROL, 89

Connecting a computer to terminals, 89
The Programmed Transmission Control, 90

Control storage and process storage, 91
Assembly of bits in channel words, 91
Transfer of characters to process storage, 93
Process storage, 95
Program modes, 96
Execution of the modes, 97
Control of the communication network, 98
Transmission delay, 99
Programming the PTC, 100
Use of program modes, 101
Servicing the terminals, 102
Communication with the processor, 104
Error procedures, 104

9 DISK FILE ORGANIZATION, 107

The 1301 disk file, 107
Seek times, 108
Record ready, 109
Interconnection of system units, 110
Objectives of file organization, 110
Factors influencing file organization, 111
The even spreading of unevenly distributed codes, 112
The generation of "synonyms," 113
Digit distribution in the control fields, 114
Direct addressing, 116
Extracting, 116
Multiplication and division, 117
A count of digits technique, 119

Re-evaluation of the conversion technique, 120
Packing, blocking, and repeating records, 121
Diagonal storage, 121
The distribution technique, 122
File indexes, 124
Chaining techniques, 124
Loading a chained file, 126
Another indexing method, 129
Additions to the file, 130
File organization in Sabre, 130
Inventory of seats, 132
File degradation, 134

10 PLANNING AND MANAGING A REAL-TIME SYSTEM, 137

Systems studies, 137
The problems of programmers, 139
Systems standards, 141
Program documentation, 143
Coordination functions, 144
Changes in specifications, 145
Programming restrictions, 145
Man-machine communication, 147
Peripheral units, 148
Program modification, 148
Post-processing of assembled programs, 149
Segmentation of programs, 149

11 THE TESTING OF PROGRAMS IN REAL-TIME SYSTEMS, 151

System testing, 151
Phase I, 153
Phase II, 154

Phase III, 154
Phase IV, 155
Phase V, 155
Phase VI, 156
Phase VII, 156
Phase I simulation program, 156
Testing aids in the control program, 157
Errors in dynamic program modification, 158
The discovery of circular holds, 159
Conversion to a new installation, 159

12 THE OPTIMIZING OF SYSTEMS THROUGH SIMULATION, 161

The effect of queues on throughput speed, 161
Analysis of operating characteristics, 163
Multiprogramming scheduling, 164
The need for simulation, 166
A general-purpose systems simulator, 167
The simulation of automobile traffic queues, 169
Use of the results, 175
Simulating a real-time system, 176
Simulation of channel assignment, 178
System design through simulation, 179

INDEX, 181

1 INTRODUCTION

PURPOSE OF BOOK

Electronic data processing is entering a new and dramatic phase. In the past, information has typically been gathered from large numbers of source documents, placed on punched cards, converted to magnetic tape, and then processed at some later date. A large time gap occurred between the origination of information and the printing of output documents. This procedure is satisfactory where an immediate response to a problem is not necessary, but is inadequate where rapid adjustments must be made to a large, fast-changing, complex situation.

To achieve a continual, dynamic interaction between a central processor and its environment, a *real-time* system is required. The number of such installations is increasing, and it is therefore becoming increasingly necessary for individuals in the data processing field to be familiar with real-time techniques. The aim of this book is to introduce the reader to many of the prevailing concepts.

It will be assumed that the reader is familiar with conventional programming. I will seek to avoid using any particular machine for explanations and illustrations. However, since two important real-time applications are using the IBM 7090, there will be a tendency for concepts unique to this computer to appear.

Practical problems in gathering information have caused me to confine my attention to IBM systems. From the limited material at my command when I prepared this manuscript, I have selected many of the most interesting details. But because of the immensity and

diversity of the projects under way, and the rapid development of the technology, this book does not seek to be comprehensive or profound. Where equipment I have described has since been modified or replaced, the reader will nevertheless find it valuable to understand the functions which any equipment must perform to solve the necessary problems. I believe that this book will be useful both to students preparing to enter the data processing field and to professionals interested in large-scale Sabre-type applications. I must emphasize here that many of the complexities and great difficulties of Sabre-type systems are not present in numerous simple real-time applications.

To give the reader a glimpse into areas where real-time systems can and are being used, this introductory chapter describes briefly the overall characteristics of several applications.

THE MERCURY PROJECT

The exploration of outer space by manned vehicles furnishes a vivid illustration of a dynamic system requiring instantaneous surveillance and control by a communication-based computer. For example, the data processing system for Project Mercury† consisted basically of a single computer receiving input from and sending output to distant terminals and an operations control center. The computer was duplexed, but only one machine provided output at any given time. The extra computer was present in case of a machine failure in which case there was a switchover to the backup machine.

During the launching of a manned satellite, radar and telemetry data are sent to the computer. The computer must immediately calculate whether the spacecraft will be properly inserted into orbit. In all stages of a mission, the machine maintains visual displays at a control center. While a spacecraft is being placed in orbit, inputs arrive at the rate of 1000 bits per second. Humans at the control center decide to abort the astronaut if the launch is unsuccessful. Prior to, and for, any abort situation, the machine computes the impact point of the spacecraft. Displays are changed every ½ second during launch, and every 1 second in abort.

In the course of a flight, inputs are transmitted to the computers

† The machine configuration for the Mercury Gemini system, described on page 74, differs from the Mercury system discussed here.

from numerous radar tracking stations located around the world. This information must be accepted by the data processing equipment whenever it arrives. The data must then be sorted, edited, and readied for mathematical programs which ascertain and refine the precise orbit of the satellite. Predictions of the spacecraft's position and velocity are transmitted to the tracking stations in advance of its appearance in each area. These data enable radar equipment to pick up the spacecraft before it crosses the station horizon. Information from radar stations is used by the machine to correct its prediction of the path of the astronaut. These calculations enable the computer to compute and display the time to fire the retrorockets to bring the capsule back to earth. During flight, the control center receives new displays of the spacecraft's location, speed, and condition every 6 seconds. Another function performed by the computer is to keep a log of all input data for later analysis.

When the retrorockets are fired for re-entry, the position of the astronaut must be monitored continually, and a prediction made of where he will come down. In the course of this phase, displays are changed every 3 seconds.

To carry out these procedures, a real-time system is needed. The central computer must be able to receive random inputs arriving at varying transmission rates. These entries must be edited simultaneously with the carrying out of mathematical and logical operations. At the same time, the machine must be able to send output messages to a large number of tracking stations and the control center. The computer must maintain numerous different programs with varying priorities in memory, and must have a procedure for deciding when to give control to each program.

REAL-TIME COMMERCIAL APPLICATIONS

Many businesses require a real-time data processing system to provide immediate decisions and to make available fast and accurate service to customers. The American Airlines Sabre system is an application of this type. Here, about 1000 agents, scattered through the United States, are busily engaged in making reservations for customers.

The main problem in a reservation system is to prevent underbooking or overbooking. If there are empty seats on an airplane, potential revenue is lost. On the other hand, if more seats are sold than are

available on a flight, extreme customer dissatisfaction may result. The number of seats available on future flights is an airline's inventory. If reservations and cancellations are not reflected immediately in the reservation records, this inventory may not be used with maximum effectiveness. In the absence of a real-time data processing system, agents must make reservations with a less accurate, more cumbersome method in which reservations are made on the basis of availability reports which lag behind actual conditions. In a large operation, underselling may make a substantial difference in a company's profits.

The magnitude of the American Airlines system is indicated by the anticipated load on the electronic data processing equipment which is being installed. This machinery is daily expected to handle about 40,000 passenger reservations, 30,000 seat availability inquiries, and 20,000 ticket sales, resulting from approximately 85,000 telephone calls. These inputs will pour in from many thousands of miles of linked telephone lines.

The procedure in making a reservation is for the agent to receive a call from a customer, inquiring as to what flights are scheduled for a given day. When the customer selects a particular flight on that date, the agent enters a *need* request to the central processor. If space is available, the machine adjusts the seat inventory for that flight and date. If seats are not open, the computer sends the agent an *availability* display. The same sequence is repeated for the return reservation. Finally, the agent enters the customer's name, address, telephone number, and sundry other items of information into the data processing system.

While the basic method for making a reservation is simple, there are a large number of complex variations on this procedure. For example, the customer may later call up to cancel his reservation, or to change the date or the time of day of certain segments of his itinerary. A large percentage of customers will cancel their reservations, so a wait list must be maintained by the processor. When a seat becomes available for a wait-list customer, the computer must notify his agent, who in turn must telephone the customer to find out if he still wishes to make the reservation. Many itineraries involve interconnections with other airlines, so that procedures must be established to communicate with and make requests from other companies.

The machine system needed for this application must enter into a conversation with the airlines agent at a remote terminal set. Since the

agent himself is holding a telephone conversation with a customer, the system is designed to answer 90% of the agent inquiries within 3 seconds to avoid irksome delays in the agent's response to the customer. At peak periods the equipment receives numerous inputs every second. Because of the great complexity of the reservation system, the processor must have available a large number of programs to handle all possible ramifications of customers' requests. Since all of these programs cannot be kept in core memory at the same time, equipment must be maintained for quick access storage, and a programming system devised for bringing in programs when needed. Another requirement is enormous random-access files for storing data. This need is not present in the Mercury computers, but is essential for American Airlines which at times has in the magnitude of 600,000 passenger records in its files. Any one of these reservation records for flights during the next year may be needed at any time by the processor to handle a customer's request. The real-time data processing system for airline reservations must be capable of simultaneously receiving inputs from agents, sending output messages to the remote terminals, receiving and sending teletype messages, executing operational programs, initiating requests for information from disks, drums, or tapes, and receiving data from disks, drums, or tapes.

MULTIPLE-USER COMPUTATIONAL CENTERS

Real-time systems may be used in the future by centralized data processing services. Numerous companies could be tied into a single information-handling center via communication lines from remote terminals. These users would be able to employ the center's computational and processing facilities, as well as large random-access files. The central computer could supply services ranging from mathematical calculation to the recording of individual retail sales and the transmission of output reports to companies' terminals. Ultimately, a number of these on-line service bureaus may spring up in every large city, making data services available on demand to any subscriber.

A start in this direction has been the development of multiconsole computer systems, where a number of programmers can simultaneously compile, debug, or run programs. Such installations reduce the time period between the entry of a program into the system and the time when printed results are available to the user. By enabling programs

entered at the consoles to be processed virtually immediately, the value of the computer to engineers and scientists is increased, and the time required for debugging programs is greatly decreased.

To achieve this objective, each user must be able to act as if he had his own, albeit slightly slower, machine. It is necessary, therefore, for the system to be fast enough to avoid keeping the user idle at his console for relatively long periods of time. Long delays would obviously occur if the programs were executed consecutively. Hence some technique such as the round-robin approach, or some variant of this method must be used. In these methods each program, in turn, is allotted a small amount of time on the machine, and this cycling procedure is repeated continually. If the allotted time is in the magnitude of a few seconds, the person at the console can remain in a conversational mode with the computer. Numerous complications arise if all of the programs cannot be kept in core memory simultaneously and if the running programs are permitted to store data on disks.

OTHER APPLICATION AREAS

Real-time data processing is potentially necessary wherever immediate decisions must be made concerning a large, complex, rapidly changing system. For example, whenever customers request credit or management takes action on the basis of current inventory, there is a necessity for a fast analysis of completely current information.

In banking, to cite one possible application, it might be possible to place terminals at each teller's window. These input-output sets would enable every clerk to communicate immediately with ledgers and records stored in a central file. Each teller would enter deposits and withdrawals in his terminal, and the central computer would automatically update the customer's account in a random-access storage. The date and the teller's identification would be included with the posting. The computer would calculate interest and enter it in the file. Record tapes produced at the terminal whenever the tellers entered information would provide an audit trail for every transaction.

Terminals could be located in branches spread over a wide geographical area, and could be connected to the central processor by leased lines. Records might be filed in the IBM 1311 disk storage, which uses removable disks, each storing about 3 million characters of alphanumeric information. Punched-card input-output, a printer,