MICROPROGRAMMING CONCEPTS and TECHNIQUES

Ben E. Cline


PBI
a petrocelli
book
new york / princeton

Typesetting by Backes Graphics
Designed by Diane L. Backes

Printed in the United States of America
1 2 3 4 5 6 7 8 9 10

DEC and PDP are trademarks of Digital Equipment Corporation, Maynard, Massachusetts.

Library of Congress Cataloging in Publication Data

Cline, Ben E.
Microprogramming concepts and techniques.

Bibliography: p.
Includes index.
1. Microprogramming. I. Title.
QA76.6.C56 001.64'2 81-15359
ISBN 0-89433-133-7 AACR2

MICROPROGRAMMING CONCEPTS and TECHNIQUES

This book is dedicated to my wife, Sue Ellen, and to my parents.

Preface

Microprogramming is a technique used in the implementation of the control circuits of digital computers. It is a topic of interest to all who desire to understand the fundamental workings of the modern digital computer. The ability of a single microprogrammable processor to emulate several different machines is also of interest. This book introduces the subject of microprogramming to the student and computer professional who have a basic knowledge of computer architecture and computer programming.

The number of individuals directly involved in microprogramming has been relatively small compared to the number involved in conventional computer programming. Computer vendors have been heavily involved in microprogramming, since it is a method used to implement the control circuits of many machines. Others have enhanced machine instruction sets for special-purpose applications or for research applications. With the increase in technology and the subsequent decrease in hardware costs, the number of individuals involved in the practice of microprogramming is on the rise. The bit slice integrated circuits discussed in Chapter 8 bring the construction of a sophisticated microprogrammable machine within the reach of the advanced computer hobbyist.

Structure

The best method of learning about microprogramming is to be exposed to several different microprogrammable machines. It is beyond the scope of this book to survey numerous machines, but the reader should be equipped to read current literature and technical manuals on the subject of microprogramming.

Most of the examples of hardware and microprogramming in this book use hypothetical machines to simplify explanations. To offset the use of hypothetical machines, the popular Digital Equipment Corporation LSI-11 microcomputer is

described in terms of its microprogrammable architecture. The Advanced Micro Devices (AMD) 2900 family of bit slice integrated circuits is also described to provide some insight into real hardware. In the cases of the LSI-11 computer and the AMD2900 integrated circuits, the discussions are not so involved as to make them unsuitable for the student.

Chapter 1 is a brief review of computer architecture. It emphasizes concepts and terminology that are essential in an introduction to microprogramming.

Chapter 2 describes hardwired control unit design. Since microprogrammable control unit design is an alternative to the hardwired approach, Chapter 2 provides the background needed to contrast the two types of design. Some of the hardwired control unit ideas are also needed to understand microprogrammable computers.

Chapter 3 introduces microprogramming as a method of implementing digital computer control units. Different approaches to microprogrammable control unit design are discussed. Chapter 4 expands the ideas presented in Chapter 3 by exploring emulation and comparing emulation and simulation.

Chapter 5 provides examples of microprogramming, using a simple hypothetical machine. The hypothetical machine is described, and a microprogrammable control unit is discussed along with its programming.

Chapter 6 describes the Digital Equipment Corporation LSI-11 microcomputer. The LSI-11 is a popular, inexpensive processor that executes the PDP-11 instruction set. The LSI-11 architecture, instruction set, and microprogrammable control unit are presented.

Chapter 7 explores microprogramming by nonvendors and the use of programmatically modifiable microprogrammable control units.

Chapter 8 discusses bit slice integrated circuits that may be used to construct inexpensive microprogrammable control units. The Advanced Micro Devices 2900 family of bit slice integrated circuits is used to illustrate bit slice concepts.

Chapter 9 presents tools for developing microprograms. Chapter 10 presents special topics from the subject of microprogramming.

A review of instruction execution, instruction formats, and operand addressing at the machine language level is provided as an appendix. In conjunction with Chapter 1, this appendix should provide the necessary background material for the student who does not feel comfortable with the subjects of computer architecture and machine language program execution. The student who has no previous study of these subjects is encouraged to consult additional sources before attempting to study the subject of microprogramming.

Acknowledgement

I wish to acknowledge the help of my wife, Sue Ellen J. Cline, in preparing the manuscript for this book.

BEN E. CLINE

Contents

1

Computer Architecture Overview

The focal point of this book is an implementation technique used in constructing the control circuits of the modern stored program digital computer. This technique, microprogramming, must be developed in the context of general digital computer architecture. Although the reader is assumed to have a background in the areas of computer architecture and computer programming, this chapter presents a brief overview of digital computer construction and provides terminology necessary in reading the following chapters.

The overview presented in this chapter is simplistic. There are numerous methods used to implement digital computers. Only those general architectural features needed to understand microprogramming are presented. More detail on the subject of general computer architecture may be gained from the study of texts on the subject and from the study of several different digital computers.

Parts of a Digital Computer and Communications Paths

Traditionally, a stored program digital computer is diagrammed as having four conceptual subsystems. These parts are the memory unit, the arithmetic and logic unit (ALU), the input/output (I/O) unit, and the control unit as illustrated in Figure 1.1. Data paths are represented with solid lines, while control paths are illustrated with broken lines.

The *memory unit* (or *primary memory*) is used to store both programs and data. The *arithmetic and logic unit* performs computations. The *input/output unit* provides communications with external devices. The *control unit* directs the other units to perform the actions implied by the execution of a program stored in the memory unit. These divisions are conceptual and may have no relation to the construction of a particular computer. As with some microprocessor integrated circuits, the control unit and arithmetic and logic unit (a central pro-

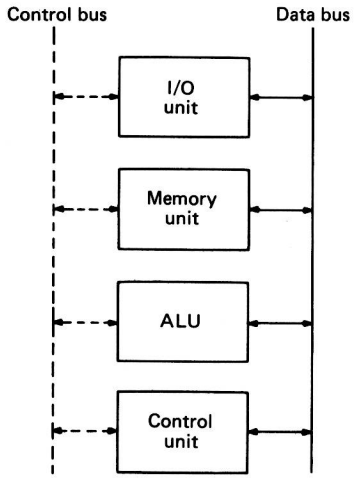

Figure 1.1. Conceptual parts of a digital computer

cessing unit–CPU) may be implemented as a single integrated circuit, or portions of each conceptual unit may be found on several printed circuit boards in a computer mainframe.

An information path in a digital computer is called a *bus*. The various bus implementation techniques are not important to this discussion. Each conceptual bus described in this chapter may carry information from a single information source to one or more information sinks during a specific transaction. Most buses are bidirectional. An information source during one bus transaction may be an information sink during another bus transaction. A unit connected to a bus may also ignore information on the bus during a transaction.

The data bus shown in Figure 1.1 carries information such as program instructions, results of computations, and data routed to or from external devices. The particular bus transaction at a given instant is selected by the control unit. Each of the four conceptual units may be a data source or a data sink to the data bus.

The control bus provides a mechanism for the control unit to direct the memory unit, the arithmetic and logic unit, and the input/output unit. Information concerning the status of these units is passed to the control unit over the control bus. Selection of a particular memory element or a particular input or output device register is affected by address information carried by the control bus.

Arithmetic and Logic Unit

The ALU of a stored program digital computer performs arithmetic operations such as addition and subtraction and logical functions such as "and" and "or".

The control unit selects the ALU function, ALU input sources, and ALU output destination for each ALU operation.

Figure 1.2 shows a typical ALU. Associated with this ALU is a register file consisting of n registers labeled 0 through $n - 1$. A computation network performs arithmetic and logic functions. A selection network allows any register or the data bus to be connected to either computation input (I1 and I2) and routes the computation network result (R) to the data bus or any register.

The computation function selected by the control unit may produce an output based on zero, one, or two of the ALU inputs. An output based on zero inputs is a constant function. A typical constant function produces a zero output, which can be stored in a register to clear it. A function that produces a result based on a single input simply ignores the other input. Functions that produce the one's complement and two's complement of a number are example functions based on a single input. Two input functions include operations such as addition and logical and.

The ALU output may be routed to the data bus or the register file. The ALU output may be ignored by not enabling the bus control logic and the register file store logic.

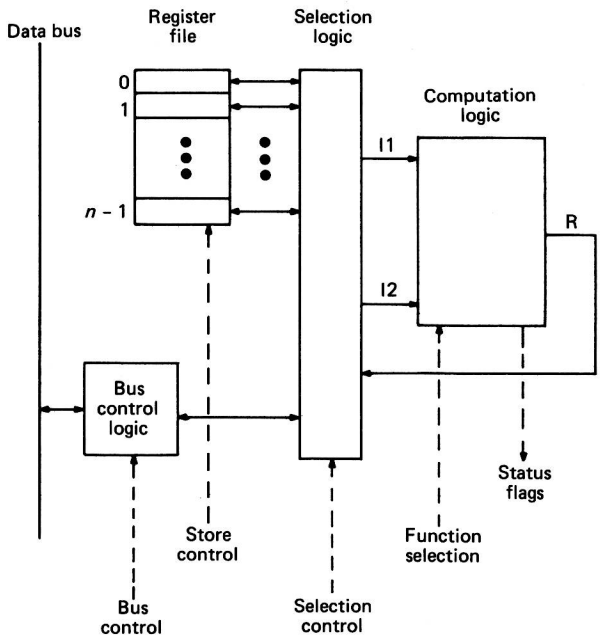

Figure 1.2. Diagram of an ALU

The ALU computation network produces several status flags that indicate properties of an ALU result such as "result is equal to zero." These flags may be inspected by the control unit directly or may be stored in a programmer accessible register to allow quick inspection and modification by a program, depending on the computer design. During machine operations where the ALU result is not used, the flags should not be affected. An ALU "no-operation" function may be implemented to allow flags to be protected from change during these periods.

The ALU may be synchronized to a clock; in this case a single primitive ALU operation will always take one clock cycle. As an alternative, the ALU may produce a signal that informs control circuits when the selected operation is complete. The synchronous ALU is presented for this discussion.

As an example of the operation of the ALU in Figure 1.2, suppose that for a particular operation the control unit must produce the signals to cause the ALU to add the data bus value to the contents of register zero and to store the result in register zero. It would direct the bus control logic to receive the data bus value, and it would direct the ALU selection logic to connect the output of register zero to the I1 input and to connect the data bus value to the I2 input. The control unit would select the ALU addition function. The ALU selection logic would also be directed to connect the ALU output to the input of register zero. At the end of the clock cycle, the stabilized ALU output would be strobed into register zero by the activation of the store control signal by the control unit.

Memory Unit

The primary memory unit of a modern digital computer is typically a *random access memory (RAM)*, meaning that access time is not dependent on the order of access. This memory usually allows memory cells to be written or read under control of the control unit; however, all or part of the memory may be constructed of *read-only memory (ROM)* circuits. Read-only memories are preprogrammed and may only be read, not written, under control of the control unit.

The heart of the typical memory unit of Figure 1.3 is a set of memory cells. Each cell is an addressable unit that consists of several bits of data. In some machines an eight-bit byte is the smallest addressable unit; other machines address words much wider than eight bits.

Figure 1.4 shows an addressing format for a typical memory unit. If n addressable cells are available, the cells can be numbered zero to $n - 1$. Before the control unit can read or modify a memory word, it must first select the desired cell via the address control lines.

The function control signals and bus control signals (Figure 1.3) allow the control unit to cause the memory unit to be idle, to put the data bus value in the selected memory cell (write operation), or to place the contents of the selected

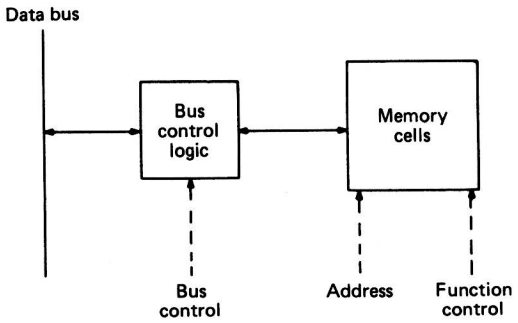

Figure 1.3. Diagram of a memory unit

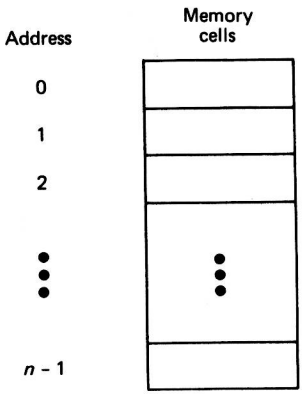

Figure 1.4. Typical memory addressing technique

cell on the data bus (read operation). Like the ALU, the memory unit may be designed to function synchronously or asynchronously. This discussion assumes a synchronous memory unit.

Memory accesses are generally slower than ALU operations. Several ALU cycles may occur during a single memory unit access cycle. The control unit may initiate a memory access operation; and while the memory unit is cycling, the control unit may effect one or more ALU operations before the memory unit completes its operations. Such parallel operations allow efficient use of processor resources.

Both data and program information are stored in the memory unit. To execute an instruction, the control unit must first read (or fetch) the instruction from the

memory unit. If the instruction implies that memory data be read or written as part of the operation it performs, the control unit must cause additional memory unit accesses during execution of the instruction.

Input/Output Unit

The I/O function may be implemented in a variety of ways. Three general methods are discussed in this chapter: I/O bus, memory-mapped I/O, and direct memory access. The direct memory access method is described in a separate section.

A dedicated I/O bus system is shown in Figure 1.5. Interfaces for I/O devices connect to the I/O bus. The I/O unit accepts control signals from the control unit and manipulates these interfaces via the I/O bus. The device interfaces are selected by an I/O address. The I/O unit can determine the status of each device and transfer data between the interfaces and the data bus under control of the control unit.

Characteristic (but not mandatory) of the dedicated I/O system is a set of machine language instructions that perform I/O operations. These instructions generally include an I/O address field to specify the device interface to be affected; however, some I/O instructions apply to all interfaces (such as an instruction to reset all devices). Typical instructions test the state of a device or effect a data transfer.

Figure 1.6 shows a memory-mapped I/O system. Unlike the dedicated I/O bus system where memory and I/O are in different address spaces, I/O and memory

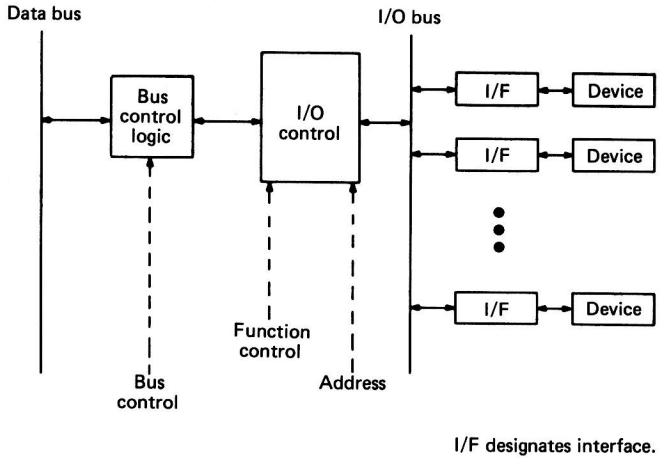

Figure 1.5. Dedicated I/O bus system

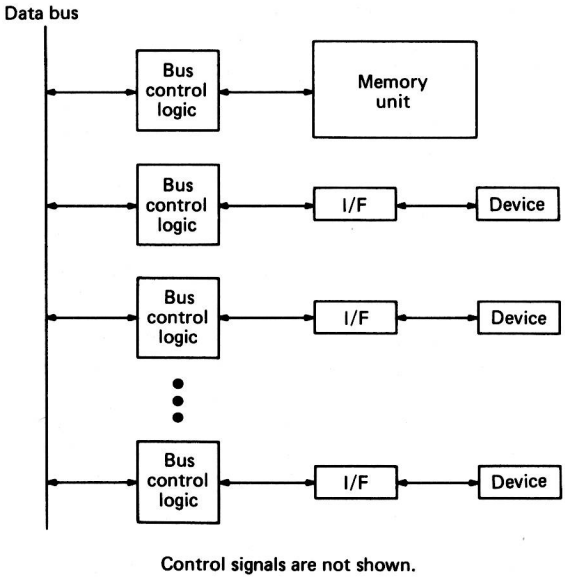

Figure 1.6. Memory-mapped I/O system

share the same address space in a memory-mapped I/O system. Each interface and the memory unit must respond in a similar manner when it is addressed by the control unit. The control unit directs both memory and I/O in a uniform way.

Characteristic of this type of I/O system is the lack of I/O instructions in the machine language instruction set. Since I/O and memory are accessed in a similar manner, memory reference instructions can also manipulate registers in device interfaces to effect I/O transfers.

Control Unit

The design of the control unit of digital computers is the subject of the remainder of this book. The discussion of the control unit in this chapter is expanded in subsequent chapters.

Figure 1.7 shows a control unit as a circuit which produces output signals based on certain input signals. Input to the control unit comes from the control bus and the data bus. The control bus inputs information to the control unit concerning the status of the ALU, memory unit, I/O unit, and environmental conditions (power supply state and front panel state). The data bus carries machine language