A.P.I.C. Studies in Data Processing
No. 15

COMPUTER ARCHITECTURE: A STRUCTURED APPROACH

R. W. DORAN

COMPUTER ARCHITECTURE: A STRUCTURED APPROACH

R. W. DORAN
Senior Computer Architect,
Amdahl Corporation,
Sunnyvale, California, USA

ACADEMIC PRESS
LONDON NEW YORK SAN FRANCISCO
A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS INC. (LONDON) LTD.
24/28 Oval Road
London NW1

United States Edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue
New York, New York 10003

British Library Cataloguing in Publication Data

Doran, R W
Computer architecture. – (Brighton College of Technology. *Automatic Programming Information Centre*. Studies in data processing).
1. Data processing
I. Title II. Series
001.6′4 QA76 78–73879

ISBN 0–12–220850–1

Printed in Great Britain by
Fletcher and Son Ltd, Norwich

Preface

This book originated from notes to accompany a course on Computer Architecture given at Massey University. At the time, the only texts available to be used for such a course were collections of papers describing many different computers or delving deeply into the logical design of one particular machine. Computer Architecture courses tend to be case studies looking at various machines at a not-too-deep level in order to find the best illustration for each topic of discussion. Such an approach is unfortunate because the student does not get to see how a computer architecture integrates the facilities it requires to accomplish a variety of purposes. However, the opposite approach of working through the complete description, usually the machine manual, of one computer leaves the student with detailed knowledge of one architecture but with no understanding of its rationale, its adequacy or the alternatives. One resolution of these two extremes is the approach taken here of discussing the principles behind computer architecture and at the same time developing a description of one example in detail.

In order to really understand a computer's architecture in detail, if you did not actually design it, you need to have access to field engineering descriptions, even logic diagrams, to make up for the lack of detail in programmer-oriented machine reference manuals—the low level documents are also a great source of clues as to the designer's real intentions. Having the computer available for student use makes the course more real and provides exercises such as examining the code produced by compilers. The choice of example computer was thus limited to those at hand, and, of those available, the largest and most modern was the Burroughs B6700.

The availability of the B6700 as an example was extremely fortunate because, at the time, it was the only general purpose computer about which one could say that it was clearly designed with its applications in mind. The B6700 is an extremely interesting, though complex, machine and, truth to tell, there would not be much point in writing a book of this nature without a machine like the B6700 to describe. The B6700 can serve as an example of one solution to many architectural problems but there are, of course, other alternative solutions. Where the alternatives are significant the approach taken here is to mention them but to not, usually, discuss them in detail and instead refer the reader to other sources. Note that the B6700 available for study was a small uniprocessor; details of the larger "reconfiguration systems" and the B7700 have been extracted solely from their reference manuals.

Every computer architecture is the concrete realization of a conceptual model of the computational process. Different computers embody quite different models which are not always easy to relate, especially in the case of the more complex, higher level computers. The main challenge when describing computers in a more general fashion than a case study is to "build" up a model that subsumes the examples being described so that they can be compared fairly and their inadequacies and strong points brought to light. In this book the first part of each chapter and section is spent in building a general model in terms of which the B6700 is then described. The model developed is definitely not that underlying the B6700 although very much based on the B6700's model (the B6700 model is described in the early papers on the machine and in Organick's monograph). One danger in developing a general model is that integration and generalization of the concepts behind different computers soon develops into research and speculation. An attempt has been made here to not go too far into the unknown although, admittedly, the model at times goes beyond what is widely accepted.

The level of this book reflects the particular mix of courses to which our students had previously been subjected. Most had looked at the detailed organization of a small computer and were also familiar with both high and low level programming. The text consequently assumes a reasonably high level of knowledge; however, there has been no hesitation in going right back to basics in order to clarify a principle. Although the level is mixed, the text should be of use to all computer professionals interested in computer architecture. The emphasis on details of the B6700/B7700 should make the book of particular interest to the users of such systems and to others who are intrigued by Burroughs' mysteries.

The Burroughs B6700/7700 has already been the subject of one monograph, that of E. I. Organick entitled "Computer System Architecture". It should be made clear that this book and Organick's are quite different in their scope. Organick's book is oriented towards the operating system of the B6700 whereas here it is the hardware architecture which is the main topic. Of course there is some overlap where the hardware which supports the operating system is described, but even then the approach taken here is one of the B6700 illustrating more general principles rather than being described solely for its own sake.

I would like to thank those who, directly or indirectly, assisted in this book's development. In particular; the students at Massey University who suffered through the courses where the book's material was gradually made intelligible; colleagues in the Computer Science Department, especially Mr L. K. Thomas; the Massey University Computer Unit, Professor G. Tate director, for providing access to their B6700; Burroughs Corporation (New Zealand) for allowing me to peruse detailed B6700 documentation; for assistance in reviewing—Dr H. C. Johnston, Professor C. A. R. Hoare and Mr R. D. Merrell; Mrs Verna Mullin for a sterling job of typing; finally, my wife and family for their patience and encouragement.

Palo Alto R. W. DORAN
January 1979

Contents

PAGE

Preface v

I. The General Purpose Computer 1

I.1. The User's Viewpoint 2
- I.1.1. The Calculator 2
- I.1.2. The Universal Machine 3
- I.1.3. The Data Processor 5
- I.1.4. The Algorithm Executor 7
- I.1.5. The Computing Utility 8
- I.1.6. The Data Bank 9
- I.1.7. The Communications Device 10

I.2. The System Viewpoint 11

I.3. The Burroughs B6700/B7700 13

I.4. Notes 15

II. Algorithms 17

II.1. Trees and Stacks 17

II.2. The Arithmetic Stack 21
- II.2.1. Evaluation of Expressions 22
- II.2.2. Implementation of Stacks 24
- II.2.3. Stacks in von Neumann Machines 25
- II.2.4. Automatic Speed-up of Stack Arithmetic 26

II.3. Machine Language Programs 29
- II.3.1. Branching Instructions 31
- II.3.2. Subroutine Structure 34
- II.3.3. Instruction Formats 36

II.4. Addressing Environments 36
- II.4.1. Environments and Relative Addresses 37
- II.4.2. Structured Environments 38
- II.4.3. Execution Records 40

II.4.4. Block Structure 45
II.4.5. The B6700 Stack 49

II.5. Operand-dependent Instructions 55
II.5.1. Tagged Memory 56
II.5.2. "Intelligent" Arithmetic 58

II.6. Data Structures and Algorithms 63
II.6.1. Escape Mechanisms 64
II.6.2. B6700 "Accidental Entry" 67

II.7. Notes 68

III. Data Structures 70

III.1. Basic Structures 71
III.1.1. Array Structures and Descriptors 72
III.1.2. Basic Operations on Array Structures 76

III.2. Strings 79
III.2.1. String Scanning 80
III.2.2. String Editing 84
III.2.3. Decimal Arithmetic 86

III.3. Vector Processing 89
III.3.1. Multidimensional Arrays 92
III.3.2. Special Vector Processing Facilities 95

III.4. Notes 99

IV. Operating Systems 101

IV.1. The Operating System Concept 101

IV.2. Unlimited Systems 104
IV.2.1. The Dynamic Structure of Tasks 104
IV.2.2. Conceptual Machines 112
IV.2.3. Interaction Between Tasks 116

IV.3. Processor Sharing 125
IV.3.1. Processor Allocation 125
IV.3.2. Processor Sharing by Interrupt 128
IV.3.3. Sleeping Tasks 133

IV.4. Virtual Memory 134
IV.4.1. The Structure of Virtual Memory 134
IV.4.2. Relative Addresses and Shared Segments 137
IV.4.3. Mapping Virtual Memory to Actual Memory . . . 141

IV.5. Protection 150
IV.5.1. Rings of Protection 152

IV.6. Notes 155

V. Input/Output 156

V.1. The Hierarchy of I/O Processors 157
V.1.1. Levels of I/O Tasks and Their Processors 157
V.1.2. Sharing and Allocation of Data Transfer Processors . . 160
V.1.3. Control of I/O Tasks and Processors 164

V.2. B6700/B7700 I/O Processors 167
V.2.1. B6700 I/O Control 167
V.2.2. B6700 I/O Processor 169
V.2.3. B6700 Data Communications Processor 179
V.2.4. B6700 Disk File Optimizer 186
V.2.5. B7700 I/O Module 190

V.3. Availability 196
V.3.1. Replication of Components 197
V.3.2. The B6700 Reconfiguration Systems 199

V.5. Notes 202

VI. High Performance 205

VI.1. Performance Improvement 205

VI.2. The Memory Bottleneck 207
VI.2.1. Program Buffer 209
VI.2.2. Stack Buffer 211
VI.2.3. Data Buffer 214

VI.3. The Processor Bottleneck 216
VI.3.1. The Pipelined Processor 216

VI.4. Notes 220
References 221
Index of B6700/B7700 Terms and Features 225
General Subject Index 231

I. The General Purpose Computer

Modern general purpose computer systems have become so complex that they are beyond the understanding of any one person. Although overall comprehension is impractical, it is still possible to come to grips with a computer at its architectural level. A computer's architecture is the interface between the designers of its hardware and software and is, after all, specifically intended to be fully understood by both teams of engineers. Unfortunately, the documentation provided with most systems does not facilitate understanding because, while it is exhaustive in detail, it is purely descriptive and lacking in rationale. It is the intention here to take one particular example computer and to set the matter to rights by describing it in such a manner that the reader can reach an understanding of what the computer's architecture is and why it is like it is.

As each successive aspect of the computer is described, we need, at the same time, to develop a model of what any general purpose computer is trying to accomplish in such circumstances. Before we can begin, we need, then, to have a feeling for what a computer is doing overall; indeed, to understand just what a computer *is*. In this introduction we will set the context for the later description by discussing just what "general purpose computer" means nowadays.

The epithet "general purpose" applied to computers has become so common that it is almost used automatically. However, here it is used with purpose and with a meaning which will be precisely defined. In fact, "general purpose" means different things to different people, but its imprecision appears slight when compared to the meanings associated with "computer" itself.

The most widely held concept of what a computer is has not remained constant but has changed over the years. Sometimes a change of viewpoint has come suddenly, as with the celebrated erratum for a computer manual in 1957—"Wherever the term 'computer' or 'digital computer' appears throughout the text, replace it by the term 'Data Processor'." [1]. Usually, however, the changes are gradual and concepts merge into one another, it being quite reasonable to consider computers from more than one viewpoint at the same time. However, when changes of viewpoint do occur there is a significant resulting impact on computer architecture itself. This is because a new aspect of the computer system assumes dominance and a major alteration to the system in order to make it fit in with the new view does not appear quite so radical or unreasonable as it may have done from the old viewpoint.

Computers as seen by users and as seen from an overall systems viewpoint present different appearances. In the following two sections we will consider first the user's view and then the modern system view. After this discussion of principle the last section of the chapter will consider the practical matter of the computer that we are going to discuss in detail.

I.1. THE USER'S VIEWPOINT

Although the total image of what a computer should be is compounded from the views seen by a whole range of users, there are cases where the view of one group of users is the dominant concept behind the computer's architecture. In these cases the computer is designed as if it were to be used permanently for the one particular purpose. Other applications may be recognized but these are accommodated by slight modifications or extensions rather than by a radical alteration of the architecture. We will consider below a number of these views that can be clearly isolated. They are, for emphasis, expressed in an extreme form, although few people would hold such singleminded opinions.

I.1.1. THE CALCULATOR

The oldest view of a computer is that it is a device for performing any numerical calculation; indeed, the term *computer* or *computor* was applied originally to any human calculator. Babbage's proposed analytic engine was designed specifically to perform arithmetic and it appears that Babbage knew that his machine could be programmed to perform *any* numerical task [2]. Most of the early model computer designs were also made from this viewpoint as is shown in the following extracts from the seminal 1946 report of Burks, Goldstine and von Neumann, which described a device which was to be the paradigm of the computer for many years and has come to be called the "von Neumann machine"[3]:

> "In as much as the completed device will be a general purpose computing machine it should contain main organs relating to arithmetic, memory storage, control and connection with the human operator.
> "In as much as the device is to be a computing machine there must be an arithmetic organ in it which can perform certain of the elementary arithmetic operations".

The above quotations introduce and effectively define the term "general purpose" as applied to computers, clearly assigning it the meaning of numerical calculator. This interpretation is confirmed by other early writings.

The first American computers had their origins in projects requiring vast amounts of calculation where the idea of a computer as a calculator was most natural. Nowadays most computers are used for commercial applications and computing professionals no longer consider their general purpose computers to be primarily calculating machines. There are computers designed for that purpose, such as great array processors like Illiac IV and small pocket computers more usually called programmable calculators, but these are considered to be very special purpose machines. Outside of the computing field, however, the public still has the simple view of a computer as a calculator.

Although we no longer think of a computer as primarily a calculator, this view still has the most profound influence on computer architecture. The majority of modern computers have been designed as if they were to be used as calculators, though it is recognized, implicit in the term "non-numeric computation", that some calculations will involve other than numbers. That this is true can be seen by inspecting manuals describing a computer to low level programmers. The important part of the machine is the central processor, other devices such as those for input and output are complications of detail and are left to the later chapters of the manual. The machine is defined by what the central processor can do.

Within the central processor of a computer-as-calculator each instruction is regarded as performing one complete step in a calculation. Each instruction is independent of others in the machine's repertoire and its effect is a meaningful, complete operation. There are many styles of instruction set possible (e.g. 1,2,3 or 4 addresses) but the format of the instruction always includes a unique operation code which defines the operation to be performed. Thus, apart from some preliminary details of the processor's organization, the description of the processor takes on the form of a list of instructions, giving for each its operation code and a description of what it does. Although, over the years, computers based on this line of reasoning have been augmented with special devices in order to make them more effective, the closed instruction form has remained in most machines and even the few that have not been designed from the calculator viewpoint are for the most part still described as if they were.

I.1.2. THE UNIVERSAL MACHINE

Although the development and production of modern computers has been in the main an American activity, many of the fundamental ideas of computer science have arisen outside the United States. Britain has been a major source of innovative ideas in computer architecture—after all, the whole subject was started by Babbage and the first modern computers to actually work were at Manchester and Cambridge. In modern times, perhaps the most important and definitely the most creative thinker in the computing field was the British mathematician A. M. Turing. In a paper published in 1946 Turing explored the limits of computation by defining and investigating the properties of a class of abstract machines which have now come to be called "Turing machines". Turing showed that it was possible to define a *universal* machine which could simulate the behavior of any other machine, i.e. a single universal machine could be constructed which could perform any mechanical algorithm.

Soon after the first modern computers were placed in operation it became accepted in the computing community that these machines were, in fact, universal. This is demonstrated by such names as UNIVAC (universal automatic computer) and DEUCE (digital electronic universal calculating engine). The term "general purpose", which always implied "universal" because a machine which can perform any numerical algorithm is certainly universal, became synonymous with the more general term. Throughout the fifties and sixties "general purpose" as applied to computers meant "universal" and it still has that connotation to some people today.

As well as being a theoretical genius, Turing was extremely interested in practical computing, particularly in non-numeric applications, for he was involved in using electronic computing equipment for cryptanalysis during the Second World War. In late 1945 Turing embarked on a project to produce a computer at the British National Physical Laboratory. Turing thought of his machine, although mainly to be used for numeric calculations, as being universal right from the start. In his first proposal of 1945 he lists chess and a variety of other possible applications [4]. In the following quote from his proposal he explains in simple terms how this can be:

> "... There will positively be no internal alterations to be made even if we wish suddenly to switch from calculating to energy levels of the neon atom to the enumeration of groups of order 720. It may appear somewhat puzzling that this can be done. How can one expect a machine to do all this multitudinous variety of things? The answer is that we should consider the machine as doing something quite simple, namely carrying out orders given to it in a standard form which it is able to understand".

Turing, because of his background in theory and non-numeric data processing (i.e. his involvement with wartime cryptanalysis), took a very different approach to the design of ACE (automatic computing engine) as his machine was called. He tried to design a universal machine by implementing a very basic architecture and providing application-oriented instruction sets as collections of subroutines. Now, although there is no direct connection, because Turing's work was largely ignored, this is just the approach taken with many modern computers where there is, "underneath" the outer computer seen by the programmers, a micro-machine or inner computer, which emulates the actions of the outer machine.

The essential features of a modern inner computer and Turing's 1945 design are much the same. An inner computer is extremely special purpose, being designed specifically for the limited application of emulation. As with the von Neumann machine the emphasis is on the central processor but an inner computer is designed at a lower level. The micro-programmer of such a machine sees a group of functional units each capable of performing some operation on data. The main purpose of instructions is to move data from unit to unit so that the correct sequence of operations is carried out. Instructions are not usually complete in themselves and many may have to be executed to perform a useful step in a calculation. Often the instructions contain fields of parameter bits each of which selects one option for controlling a particular unit. At one extreme, taken by Turing in his fifth design for ACE [5], it is possible to do away with instruction operation codes altogether, an instruction being solely for moving data from a source to a destination with the units and registers connected by data buses (a scheme used in earlier calculators such as the Harvard Mk I [6]).

One unfortunate misinterpretation of "general purpose" as universal is that a general purpose computer can do any calculation *practically*. No one would state such a blunder openly but it does seem to be a prevalent attitude and is partially responsible for the continuance of the von Neumann plus index registers model of a computer. It is hard to justify altering a computer's architecture when the feature desired can always be provided by software, particularly so at the design stage where the cost of the hardware to implement the feature is known, but its worth and the cost of software implementation are not so clear.

I.1.3. THE DATA PROCESSOR

The notion that a computer is fundamentally a data processing device came with the realization that commerce was to be the largest market for computers. Computers, starting with the UNIVAC and IBM 702 in the United States, and LEO in Britain, began to be designed specifically for the automation of business accounting.

The use of computers in business followed the techniques already evolved for unit record accounting equipment. The initial emphasis in data processor design was to increase the speed of the peripheral devices at handling unit records so that the fast electronic central processors could be used efficiently. As was stated in a paper introducing the IBM 702 [7]:

> "The characteristic of the 702 that soon becomes evident is its speed, particularly the speed of its input and output".

From the data processing viewpoint the peripherals are the really important devices and the flow of data through the machine the important concept: any computation is subsidiary and somewhat incidental to the machine's main purpose. The acceptance, though not necessarily the initial impetus, for many features of input/output was a result of data processing. Immediate examples are the use of buffers, I/O completion interrupts, and, eventually independent I/O processors. The central processor was not greatly influenced at all, the instruction set being slightly extended to deal with characters and to perform decimal arithmetic. These extra features were enough, however, for manufacturers to begin to market two lines of computers, commercial and scientific, the latter featuring binary arithmetic, maybe even floating-point hardware, but omitting decimal operations.

Later, computers designed for business started to develop in their own directions. The accumulator, which was included in the first data processors (the IBM 702 had a varying length accumulator up to 510 decimal digits long!), was eventually omitted with some machines, starting with the IBM 1401, which had character-addressed memories and instructions which referred to two or more variable-length fields of characters. There are many computers currently in use which are clearly of this special data processing type but most manufacturers have replaced them with series of machines (typified by the IBM System 360) which include both the commercial and scientific features. Most of these latter series include within their instruction sets a few extra instructions to help data processing, e.g. for translation and for editing in COBOL.

An interesting sidetrack in computer architecture was provided by the extreme data processing viewpoint of a computer as a "stream machine". The processor was viewed as a device which accepts input streams of characters and manipulates these to produce a continuous stream of output. This was implemented in the "Harvest" version of the IBM Stretch computer and as one of the two modes of operation of the Burroughs B5000 [8]. The idea was summed up in the paper describing the Harvest system:

> "The streaming mode is primarily a design attitude whose aim is to select bytes from memory according to some preassigned pattern and to keep a steady stream of such selected bytes flowing through a designated process or transformation and thence back to memory".

The concept seems to have died out, perhaps because it is stretching a point too far.

I.1.4. THE ALGORITHM EXECUTOR

It is implicit in the universality of a computer that it can be used to execute a wide variety of algorithms. However, the early concepts of a computer, as we have just seen, concentrated on the implementation of the basic steps of algorithms rather than on the overall form of programs or programming languages. The influence of programs on computer architecture did start very early with the introduction of instructions to facilitate subroutine linkage (first suggested in Turing's 1945 proposal) but, as far as the main manufacturers of computers were concerned, progress came to an equally early halt. In fact, most computer architectures nowadays incorporate few concessions to programmability, presumably because computers are not designed by programmers and at the time when most of the current lines of computers were conceived (in the early sixties) the importance of software was not fully realized.

The first computer in which high level programming languages were recognized as a significant component of the hardware/software system was the Burroughs B5000 [9]. One of the B5000's modes of operation was unashamedly designed to assist in the running of Algol programs, the other, the "streaming mode" mentioned above, was intended for string and data processing, in particular to assist with COBOL and compilation in general. The B5000 could fairly be termed an "Algol" machine, although many other factors were taken into consideration in its design. The design process was outlined in a 1961 paper [10]:

"As an example, for a computer to process applications expressible in the Algol 60 language, the system model group would interpret the language, specify a hardware representation and necessary language supplements, define speed or cost objects and the 'use image' the machine is to present".

A certain clue that the designers of a computer have had programming in mind is the appearance of instructions which are not complete in themselves but must work in concert with other instructions. For example, subroutine return must be preceded by subroutine entry, and the "add" instruction of a zero address machine must be preceded by instructions setting up the operands. One may also find instructions implementing quite complicated but often used subalgorithms. However, the form of a machine language program has otherwise been little changed even in computers designed expressly for high level languages. Programs are still represented as sets of subroutines each of which is a linear list of instructions, though explicit recognition of the structure within a routine has appeared recently [11].

Programming has also had little effect on the input/output side of machines, perhaps because high level languages have not themselves converged on an accepted approach to the process. However, there have been instructions included in some computers to help with the basic operations of scanning, formating and editing involved with I/O.

The majority of changes in computer architecture which can be attributed to the influence of programming languages have thus been made to central processors. The most significant influence is that many computers have been designed to make use of stacks for arithmetic, for subroutine linkage and for data storage. In these architectures the stack is regarded usually as an addendum to a more conventional architecture, but some have gone so far as to make the stack the fundamental feature of the machine. In the latter machines the von Neumann general registers and accumulators are replaced by registers for very specific purposes associated with the stack; in fact, there is no longer much resemblance to earlier architectures at all [12].

Since the software explosion of the late sixties, it has become apparent that the cost of software is the major expense in the development of a computer system. The importance of programs and programming implies that computers in future must be modified to help make the programmer's job easy; consequently, we can expect the view of the computer as an algorithm executor to become all-pervasive. It is significant that nearly all the non-imitative, larger computer lines announced since the mid-sixties have used stacks in one form or another.

I.1.5. THE COMPUTING UTILITY

This view of a computer's function is at a higher level than the previous, and is much more the attitude of an outside user. In its most blatant form, this view holds a computer system to be a source of a mysterious computational or control power.

Computers were put to work controlling real time devices very early [13] and it was not long before such use affected computer architecture. The artifice of a program *interrupt* was introduced in the UNIVAC 1103A expressly to assist with control. As was stated in a paper describing the feature [14]:

"The way Mr. Turner intended to run his machine, and in fact does run it, the wind tunnel has first claim on the services of the UNIVAC Scientific".

Of course, simple peripheral devices need to be controlled in real time and the interrupt concept spread to computers in general. In fact, the DYSEAC computer introduced interrupts for I/O completion somewhat earlier (though not called interrupts) [15].

Timesharing, where a computer's time is divided among several different users each of whom has the impression of using a dedicated machine, naturally invokes the image of computation power being shared. Coupled