Eric Mutch Memorial

The 'Eric Mutch Memorial' prize was offered, in August 1972, for the best paper to be submitted on a topic nominated by the Editorial Board.

Foremost among the objectives of the Board, when considering the topic to be chosen for the first competition, were the need to emphasise the role of the Society in bringing together the various disciplines involved in computing and the need to demonstrate to all members of the computing profession the relevance of the *Journal* to their work. The chosen topic, which highlighted these points, was:

'*How the pioneering work of yesterday relates to the computing practice of today*'

The paper which was chosen by the Board as the best on this topic is reproduced below.

Microprogramming and system architecture

J. K. Broadbent

Department of Computer Science, and Statistics, Queen Mary College, Mile End Road, London E1

This paper outlines the evolution of microprogramming from its position as a technique of logic implementation to its use in the implementation operating systems and high level programming languages. Although such uses are at the moment numerically relatively insignificant it is argued from both hardware and software considerations that this evolution will continue as an extrapolation of already well established trends in system architecture.

(Received February 1973)

1. Introduction

The term 'microprogramming' was first introduced by Wilkes (1951); since then a prolific and well documented literature has emerged. A recent bibliography (Jones *et alia*, 1972) covering only 1969–1972 contained one hundred and eighteen references. Wilkes's survey in 1969 contained fifty five references and Husson's book (1970) has a bibliography with over two hundred entries on microprogramming. Many definitions of microprogramming have been given (e.g. Husson, 1970; Flynn, 1967) and its actual meaning is often discussed at some length. Unfortunately as a help in understanding a principle, definitions can be opaque and misleading and can obsolesce even though the word defined continues in use. A brief example will probably be more helpful.

The execution of a typical machine instruction such as add the contents of register R to the contents of main store location S can be broken down into several discrete steps (or rather must be broken down into discrete steps to ensure that the instruction is executed correctly) as shown by the flowchart in **Fig. 1.** In a microprogrammed machine the setting of flip flops, loading of registers, connections to an arithmetic unit or whatever is required at each step to achieve the correct change of state in the CPU for that step are determined by the contents of a memory with one word of the memory usually controlling one step. Each step may take one or more processor cycles. A collection of step controls for implementing a specific machine instruction is a microprogram and the individual step controls are micro instructions—the parallel with programming or at least with machine code programming should be obvious. Microprograms may reside in main memory or in a special faster memory known as a control store with some processors being able to execute microprograms from either main memory or a control store. A collection of microprograms for implementing a specific machine code is a 'microinterpreter'. **Fig. 2** shows a block diagram for this organisation.

Sequencing of microinstructions can either be by keeping the address of its successor as part of each microinstruction or, in default of a branch microinstruction, assuming the next sequential location in the control memory. The micro prefix is also attached to many other familiar names—microassemblers, microcode, microsimulators, microdiagnostics, microexecutives.

The word length (or bit dimension as it is sometimes called) of the control memory is influenced by many factors. Basically there is a distinction between horizontal (or minimally encoded) and vertical (or highly encoded) formats for microinstructions. In a horizontal format the microinstruction would be controlling the opening and closing of logic gates largely in a bit significant manner, whereas in a vertical format the instruction is broken down into fields for selecting registers or functions. As a strict horizontal form would require one bit per control line in the processor microinstructions are always a compromise. The word length is determined in addition by the number of distinct operations which may be performed in achieving a processor's change of state and the degree of simultaneity which can be allowed or is required to give a satisfactory performance at the machine code level. For example, the Interdata model 4 (a minicomputer) microinstruction is 16 bits, whilst that of the 360/50 is 90 bits.

Subsequent sections in this paper will deal with various aspects of microprogramming and system architecture to give an overall impression of the movement in hardware complexity to meet programming requirements. Attempts will be made to justify or explain the use of microprogramming both as part of that trend and independently of it.

2. The engineering justifications for microprogramming

Although the current increase of interest in microprogramming is probably stimulated more by software problems it is worth considering the hardware justifications first to see how they have changed and whether they are still valid. Wilkes's original paper describes microprogramming as an orderly approach to the design of a control section of a computer. That is a sound design principle. But it is noticeable that at any one time the largest computers available have not been microprogrammed, e.g. CDC*STAR, 7600 and 6600 and IBM 360/195. The latter is a particularly important exception as other models of the 360 range were microprogrammed. In a computer with no cache memories or look ahead for operand fetches the main

Now at ICL, GmbH, Bereich Informatik, 4000 Dusseldorf, Immermannstrasse 7, West Germany.