PROCEEDINGS OF THE SYMPOSIUM ON
LANGUAGE DESIGN AND PROGRAMMING METHODOLOGY
SYDNEY, 10-11 SEPTEMBER, 1979

# A PERSONAL COMPUTER BASED ON A HIGH-LEVEL LANGUAGE

Niklaus Wirth
Institut für Informatik, ETH Zürich

Considerations of economy have in the past led to the so-called time-sharing of large computers. The premise of such systems is to project to each user the image of the entire computer being at his own exclusive disposal. This requires stringent measures of protection on programs and data of the individual participants against misbehaviour of programs and malfunctions of hardware. These measures, called overhead, are not only extensive, but also expensive.

Phenomenal advances in semiconductor technology have now increased the power of inexpensive micro-computers to the point where they can be used for tasks hitherto reserved for large scale systems. At this point, the strategy of sharing becomes of questionable value. The non-shared, personal computer is a genuine alternative and holds great promise for the future of many applications. It appears to be particularly attractive as the development tool of the software engineer; used in this function, we call it a work-station computer.

The power and usefulness of a computer does not only depend on its speed (of individual instructions) and the size of its store. It is equally much determined by the adequacy of the tools available for its use, in particular the programming language provided. It is now widely accepted that high-level languages are the only adequate tool for the development of complex systems, and these are typically the programs with which the professional software engineer is concerned. He should not have to regard his computer in terms of machine instructions and words of store, but as the mechanism implementing this high-level language.

As a consequence, a computer's design must not start with the specification of its (conventional) architecture, but rather with the

definition of the programming language. In order to achieve efficiency and economy of storage, the architecture and machine code must be chosen as an optimal interface between compiler and available hardware components.

We have undertaken a research and development effort to design and build a work station computer based on a high-level language. It consists of the following main parts:

- the language Modula-2
- a 4-pass compiler generating M-code
- a file system
- a basic executive system including linker and loader
- the hardware.

In this paper, we shall concentrate on the last two points. A basic premise of this project was and is that the computer is to be programmed in only one high-level language, Modula-2, which therefore has to be a sufficiently versatile system programming language. May it suffice here to characterize Modula-2 as a product of the ancestors Pascal and Modula [1], where the major addition to Pascal is the module which, in Modula-2, also may serve as a unit of separate compilation.

The other principal factor - besides the language - to determine the structure of the hardware, is the variety of peripheral devices to be connected. Our computer utilizes (in a typical configuration) a high-resolution display as principal output device, a keyboard and a pointing device (mouse) as input devices, a disk with exchangeable cartridge as file store, and it provides a serial line interface for connection to either a communication network or (at least) a hardcopy device.

The high-resolution display with 600 lines and about 800 dots per line requires a high-band width signal for its continuous refreshing. This led to the use of a multiport memory and a display-processor that operates independently from the main processor. Since the display reads bits from the store in purely sequential order, the width of the memory data bus was chosen to be four times the width of the CPU data bus, namely 64 bits. This measure reduces the interference of the display processor with the main computing process due to cycle stealing to a few percent only.

A consequence of using a high-level language for programming is that code is separate from data. Given a multiport memory, the use of a separate port for reading instructions is obvious. As instructions are mostly read in sequential order, the large memory bus width again appears as most advantageous. The action of instruction fetching appears as a merely slight interference with the data handling activities of the main processor.

The main processor consists of a conventional arithmetic/logical unit (built with a bit-slice micro-processor) augmented by a fast stack memory and a barrel shifter. The stack was motivated by the structure of Modula-2. It is called the hardware stack or expression stack and serves to hold intermediate results during expression evaluation. It coexists but is distinct from the main and conventional "software stack" used for procedure calls and to allocate data local to these procedures.

The barrel shifter is used mainly during the interpretation of M-code instructions constructing bitmaps for the display. In contrast with most other M-code instructions which correspond to a few microinstructions only, these display instructions represent themselves as fairly complex micro-programs. It is here that a powerful computing engine is most needed.

The processor is built with Shottky TTL technology using MSI and LSI components allowing for a cycle time of 150 ns. The fast progressing VLSI technology would permit moulding of this architecture into a small number of chips.

The project has shown that significant increases in computing power can be gained not only by the use of faster chips, but also by choosing a machine architecture appropriate for high-level languages and their compilers. The same holds for the density of compiled code, i.e. economy of storage.

---

[1] N. Wirth. Modula: A Language for Modular Multiprogramming. Software - Practice and Experience, 7, 3-55 (1977).