Workshop on general considerations on real-time programming

Chairman: **Prof. Dr R. LAUBER**
Stuttgart, W. Germany

The discussion included realistic assessments of the gains from using high-level languages, and the overhead penalties they carry. The benefits of standardisation were overshadowed by the practical difficulties of achieving it.

Comparisons of high level and low level programming

S. We should recognise that even when the goal is to program in a high-level language, some fraction of the programming will be in low-level. In my experience (an operating system for an automated radar system), the proportion of high-level to low-level programming is about 80: 20. This is not simply to achieve efficiency – high-level programs can be tuned up. A second look at high-level programs often produces a highly efficient object code, although the initial version may have been poor. In one case the high-level language version had 50% less code than the original assembly language version (perhaps because the compiler-writer himself coded it!).

L. In a real-time data gathering system we found that 85% of the code could be written in Real-time FORTRAN, supported by an operating system written in assembler language. Coding in assembly language may be necessary to reduce the size of a program. We had a 12 K FORTRAN program, of which 2.5 K words were data, and converted it into assembler. This brought the size down to 8 K still including the 2.5 K words of data, so indicates a reduction of code itself to 60% of the high-level version.

P. At the Royal Radar Establishment we developed programs for a computer controlled radar in assembly language. Later when a compiler became available the same problem was reprogrammed in CORAL 66. The expansion in code resulting from the use of high-level language was 20%. The total program length was 10 K and less than 10% of this was written as code inserts to optimise speed at critical points.

K. TPL1 at Essex has been used to program a stored program controlled telephone exchange. It had substantial overheads: 90% on space in main store, and 35% on running time. It has been studied to find the reasons for the inefficiencies, and a second language TPL2 has been designed to remove them. This iteration of language design is very necessary but elapsed time must exist to permit it.

H. CONTRAN has been used on a process control application (cement kilns) in England. Unfortunately, it introduced considerable overheads in additional storage requirements and execution time. This suggests the question whether application demands still make it necessary to squeeze the best performance possible from current hardware. There still appears to be no spare power or capacity in computers to tolerate the reduction in efficiency introduced by a programming in a high-level language.

Standardisation

S. It is desirable that we have a generally accepted real-time language. However, this is unlikely to happen until it is promoted and supported by one or more large computer manufacturers. In addition it must not be too specific to a particular computer. What is needed is a virtual processor whose primitives are generally agreed and can be realized on most hardware. PL/1 is too close to SYSTEM/360.

D. ANSI committee X3J1.4 is working on PL/1 Standardisation, and has admitted some important principles:

1. It starts from the user's point of view, not the manufacturer's.
2. You cannot standardise what is not developed.
3. You must not neglect the Europeans..

Y. Is standardisation by ANSI more successful than by a major manufacturer?

D. ANSI cooperate with ECMA, so it should be helpful for both sides.

Y. There are plenty of languages all competing for use. A sort of natural selection applies between them.

R. ALGOL 60 did not come into existence in this way.

F. A standard language would gradually lead to a standard operating system and then to a standard computer. This is not allowed!

Y. CORAL 66 was defined by users, and has been used for many applications, including an operating system.

P. CORAL 66 compilers are available on a number of machines in the UK now. These compilers conform to the definition of the language published by the Stationery Office. They have been implemented in some cases by the computer manufacturing firm and in other cases by the RRE. The compiler typically needs a simple machine with 16 K of store. It is planned in the future to allow modular extensions to the facilities of CORAL 66 so that where more space is available a more powerful compiler can be used.

General comments

T. There is a converse effect of machine architecture on programming languages. For example:

1. The Burroughs B6700 is an ALGOL machine and there is no assembler language for it: extended ALGOL is the lowest language that can be used. Because of the addressing structure which has to be implemented, there are extra core memory accesses, which make it a slower machine than the equivalent 360.
2. The DEC system 10 (formerly the PDP-10) has two pairs of protection/relocation (base/limit) registers in addition to its index registers. This means that the code and data can be addressed using different base registers, and so one can easily product re-entrant code. Making use of this facility, it is possible to produce a real-time system in which the applications programs are re-entrant, even though written in COBOL!

C. Concerning high-level languages and standard software packages, both are needed in different circumstances.

We should distinguish at least two stages of evolution:

In stage 1, the engineer has just a rough idea what the process control computer has to do, and he is experimenting to determine the appropriate algorithm. For this stage he needs a flexible general-purpose high-level language.

In stage 2, problems are well settled, and satisfactory algorithms known. Software packages containing these algorithms can be provided for the user.

J. The level of language is not important if you have good people, but it is when your staff are not so good. They will meet deadlines, but the quality of the programs cannot be guaranteed.