PORTABILITY-BASED SOFTWARE TECHNOLOGY FOR PROGRAMMING MICROPROCESSORS AND PDP-11 COMPUTERS

G. RÁCZ and J. SARBÓ

Department of Instrumentation and Metrology,
Technical University, H-1521 Budapest

Summary

The reasons of developing the software technological system applied at the Department of Instrumentation and Metrology, Technical University, Budapest, will be surveyed, as well as the problems in solving thereof the system gives support.

The system essentially relies on the CDL language and program portability based techniques. The main purpose of the system is to support the programming process of microprocessor equipments.

The efficiency of the programs developed has proved that, with appropriate technological background, reliable, quick, and small memory programs can be produced even in high level language.

In this paper the software technological system applied at the Department of Instrumentation and Metrology, Technical University, Budapest is described, as well as the problems to be solved by means the system.

The primary aim of the technological system is to support the programming process of microprocessor equipments. Several publications [2], [3] have been concerned with the elements and the most important characteristics of the system, to be summarized by the end of this paper.

Preliminaries

Development of the software technological system started in 1977. At that time the Department of Instrumentation and Metrology had dealt with applying minicomputers for about a decade, microcomputers and microprocessors for about five years.

Our research and development problems are featured by interfacing devices of general and special purpose. Programs of the systems built up of such units had earlier to be written — in lack of an appropriate high level language

—, in some assembly language, and often even the cross-assemblers had to be developed by ourselves.

Our programming tasks are characterized by the following:

1. Special hardware, varying as the case may be, has to be supported.
2. To develop programs the hardware tools (PDP11-34, PDP11-45) of the Department are the easiest to be used.
3. The developed systems were unique and mostly a single prototype was made, — with a few exceptions.
4. The type of the applied processor was dependent on:
   — the demands of the possible customer,
   — the possibilities of acquisition,
   — the technical requirements of the task to be realized.
5. Generally, not merely subroutine packages run under an existing operation system, but rather complete, stand-alone systems had to be developed.

The following conclusions have been drawn from our realized works:

1. In programs of very different purposes there are several program-parts realizing the same task, e.g.: number and code conversion routines, text-, table- and buffer-handling algorithms, as well as handlers. But, as a consequence of the application of different processors and of the different internal environments of programs, almost nothing but experience could be saved from the earlier programs into the newer ones, still less from one programmer's work into that of the other.
2. Programs are very much hardware-dependent, requiring the application of assembly languages. However, the construction of programs and a great part of the elaborated algorithms are independent of hardware.
3. The great variety of applied programming languages prevented the standardization of documentation thus, our programs were usually poorly documented and comprehensible only for the programmer who made them, still impairing, the adoption of programs and algorithms.
4. As mentioned above, there was no of other possibility but to specify by ourselves the assembly language of some processors and the appropriate cross-assemblers. As such programs were generally needed in a single or a few cases it was no worth making much effort to their elaboration. Thus, the cross-assemblers were prepared only for the most necessary functions and they did not help either the writing of programs, or the recognition of errors or the documentation to the degree expected from and advanced assembler (such as conditional assembling, macros, text-handling, vocabulary preparation, etc.).

These experiences induced us to look for languages and methods promoting a more effective program development. The CDL language (Compiler Description Language, — C. H. A. Koster, 1971) was chosen as such, successfully applied in SzÁMKI* and in other institutions. Namely:

1. CDL is an "open-ended language": the structure of the program can be defined on high level, the instructions to be executed can be written in an optional language, e.g.: on assembly level.
2. Programs written in CDL are portable: the same program part — if independent of the hardware —, can be run on different computers, may be with some restrictions.
3. The application of CDL language helps the top-down program design and program construction.
4. Programs written in CDL language can well be read by others, too, suiting documentation.

For what follows, it is considered necessary to outline some characteristics of microprocessor developments, of course, mainly, from the aspect of the software technology to be applied here.

Some characteristics of microprocessor software development

A computer program can be considered as an independent product. One of the most important requirements of the software is the accessibility of a computer (hardware–software ensemble) a given program can run on. This requirement is no doubt met for the "traditional" computer applications (data processing, scientific calculations, etc.). The universal computers, with their operating systems and high level, hardware-independent languages more or less permit the users to adapt their computers to a given task by purchasing (or ordering to develop) a new software product.

On the other hand, for microprocessor programs only part (and generally the less important one) of the microprocessor applications is "computer-like" application. Most are unique (dedicated) "device-like" applications. Although there are some resemblances between certain units, actually the differences seem to be more important, such as peripherals, memory spaces, or even processor types, different for each equipment. Another essential aspect is the technology requirement of the development of microprocessor-based equipment. According to national and international experience, the development of equipments is not simply a question of components or programming

* Research Institute for Applied Computer Sciences (Hungary).

languages, because these necessary tools now are usually available to the developer. (We return to this question later.)

These facts have several consequences, of them the most important for software technology will be detailed below.

Classification of microprocessor based equipments

The microprocessor based systems are classified according to their character as follows:

— a target system for solving simple, small problems, to be (worth being) produced in great series (over 1000 pieces), including e.g. pocket-calculators, washing-machine controllers, etc.;

— "professional" measuring and controlling devices and systems of relatively great complexity and value which can be produced in medium series (50 to 100 pieces), including e.g. machine-tool control systems, industrial robots, medical instruments, etc.;

— single (or nearly) measuring and control systems of a great complexity. Typical elements in this category are, e.g.: specific process control systems, automation of a line or of a complete factory, etc. It is still more characteristic of systems in this latter category, that the problem is solved by a small or big computer rather than by a microprocessor. With increasing microprocessor performances however, already at present solutions applying a micro- or multimicroprocessor can be imagined.

Considering the above grouping from the aspect of software technology, the following can be stated: For problems in the first category, the program is small and very simple (not more than some hundred machine instructions). In problems of this size, however, software technology has no much importance — in our opinion such a program has to be simply written in assembly. Anyhow, programs for computers in this category are optimum if of minimum length namely series production and hardware reliability aspects impose them to be produced on a single printed circuit board (or even integrated in a single chip).

In our opinion, computers in the second category above have the greatest importance. In Hungary there are several firms for telecommunication, instrumentation and mechanical industries, already suiting serial production of these computers. At the same time — provided of appropriate quality — they are marketable. This is the field where intellectual work predominates over the hardware technological level.

See more detailed description of computers belonging to this category later.

From another point of view, software technological systems are of great importance also in the third category, by their complexity and extension. Resulting from the character of tasks, the reliability is also of great importance.

Characteristics of devices of medium complexity

In the previous item microprocessors were classified according to complexity. Let's examine now some characteristics of computers of medium complexity in details.

Hardware

Computers usually consist of some (3 to 20) printed circuit boards. Among them some can be standardized (processor, memory, some peripheral interfaces), others are unique. In general, about 10 to 30% of hardware has to be unique in each dedicated equipments, the others are the same (or at least they can be selected from some boards of essentially the same purpose). New developments usually involve special sensors and peripherals and the interfacing thereof.

Software

Using standard hardware elements, the device is "individual" mostly by its software. (At present, standardization of software, — elaboration of software "components" and their application techniques — still is (not even) in initial stage.) The typical size of such software is 2 to 20 thousand machine instructions. Nowadays the software without operating system is typical.

Real-time character of problems and importance of relatively quick response time are usually emphasized.

Programs are "close-to-hardware" or at least they contain by all means such parts (there is no monitor, but if there is one, new peripherals have to be interfaced and the programmer has to solve individually, the handling of interrupts, etc.).

Life-cycle of microprocessor software

Remind that microprocessor developments are developments of equipments. Thus, instead of the life-cycle of software it is better to speak about the life-cycle of device.

Main steps of development are the following:

a) studying the user's demands, preparing device specification;
b) system design of the equipment (hardware–software rate, hardware and software specification);

c) detailed design, implementation and debugging (hardware and software separately);
d) debugging the equipment (hardware and software together);
e) setting up the computer at the user's, gathering of application experiences;
f) series production.

Below some characteristics of the above life-cycle — critical for software technology — will be stressed.

Detailed design, implementation and debugging (ref. item "c")

The parallel hardware–software development is crucial, if the equipment is to be produced in a reasonable time (before the planned equipment is outdated). It has, however, severe consequences; software has to be designed and written (or partly debugged) on a hardware which has not even been designed on board level yet! So application of cross methods is of essential importance. Remind, however, that "close-to-hardware" programs have to be debugged with cross method. The applied technology is expected to give some kind of solution to this problem.

Desing of the equipment (ref. item "b")

System design decides over the specification and task of hardware and software. Later modification (practically) is impossible, — more exactly: the hardware specification doubtless cannot be modified.

Software can be modified in principle (and in practice, unfortunately, it is likely to happen). An eventual modification demand arises, however, in phase "d" — i.e. when implementation of software may have been finished.

It has two important consequences:
— the applied technology has to stress system design and to make the designers "see" how the (still inexistent) system will behave;
— software technology has to provide, by all means, for a possibility of writing programs which can be modified (maybe in their basic specification).

Study, specification and application of the user's demands (ref. items "a" and "e")

The requirement above is supported by the following, mostly neglected circumstance: Abilities of the equipment are based on (supposed or real) user's demands. A great part of newly developed equipments are, however, not simply better for satisfying similar demands than a previous one but are applied in a field where no similar equipments has existed so far (in e.g.: medical applications). In such a case an expert unfamiliar with computer techniques (or even with electronics) cannot imagine what a microprocessor can offer. That is why the "user's demand", although it may be well specified, cannot be considered as an optimum. Putting in use the equipment, the user soon finds modifications increasing the efficiency, ease and handiness of application.

Our experience shows that implementation of similar demands usually requires minor but structural modifications in the program. (Hardware usually cannot be modified, the feedback time is prohibitive.)

It can of course, and in certain cases it has to be, stated, that the modification demand cannot be satisfied. But a factory which wants to be competitive in the market cannot a priori refuse modification demands of this type. It is more reasonable to accept that programs for a device may have to be written twice and to support it with proper technology.

Series production
(ref. item "f")

In the case of equipments designed for series production the software simply cannot contain errors. Programs of microprocessor equipments are stored in ROM. Users are not experts in computer techniques, mostly they do not even know that they have a microprocessor-controlled equipment.

Any kind of software modification in such an equipment is as complicated as to eliminate a hardware error — it requires a well-equipped laboratory. The equipment may be produced in series of several hundreds. All in all, a software error cannot be "repaired" in each product.

Thus, the emphasis on software maintenance functions is shifted to former remarks to items d), a), e) from the aspect of maintenance. On the other hand, a new element gets importance — the program library. Enterprises (departments) generally endeavour to develop several instruments within a limited application field. In this case, however, much of the software is expected to be the same — or nearly — in the different devices (and so is hardware, too!).

Thus, in developing a new equipment it is a decisive question how many new programs have to be written and how many of them can be taken from libraries.

Evaluation

The software technological system developed at this Department relies on CDL language and program portability techniques. As mentioned before, development of technology had begun in 1977. In the 1977 to '80 period the CDL(1) based system has been brought about. In the years 1980 to 82 the CDL(2) version of the system has been developed for programs of microprocessor systems to improve speed and size characteristics. The CDL(2) language is an improved version of CDL(1), able to optimize and to analyze static semantics.

A detailed description of elements of the software technology is found in [3]. The most important characteristics of the software technology can be recapitulated as:

— The system has been completed and in use efficiently for years;

— it is an up-to-date means for developing microprocessor equipment and small computers;

— the efficiency of developed programs — system programs, real-time applications, etc. — has proved that, with an appropriate technological background, reliable, quick, but small programs of assemblylike efficiency can be produced in a high level language.

It is also of importance that the work required by a development can be estimated more precisely than in the previous assembly technology.

References

1. KOSTER, C. H. A.: A Compiler Compiler. Matematisch Centrum Amsterdam, MR 127 (1971).
2. HANÁK, P.—RÁCZ, G.—SARBÓ, J.: Microprocessor Software Technological System. (In Hungarian) Programming Systems '81 Conference, Szeged (1981).
3. SARBÓ, J.—RÁCZ, G.: An Approach to Real-Time Microprocessor Programming. Period. Polytechn., El. Eng. *27* (1983).

Gábor RÁCZ
János SARBÓ } 1521 Budapest