SUPPORT TOOLS
FOR
MICROPROCESSOR APPLICATIONS

Hoo-Min D. Toong

December 10, 1978

CISR# 40
WP# 1045-79
SUPPORT TOOLS
FOR
MICROPROCESSOR APPLICATIONS

Hoo-Min D. Toong

December 10, 1978

CISR# 40
WP# 1045-79
ABSTRACT

The accelerating rate of advance of VLSI technology is challenging our ability to apply it or even evaluate it. There exist a host of problems that center around the application of this technology and how, when and where we use it. Methodologies for its evaluation and application must be developed.

This report attempts to outline the problems inherent in microprocessor software development and possibilities for future directions in the design and execution of support tools. A serious gap is growing between the hardware technology and the software tools available for its application. Software productivity is a key problem in this area of technology. The current situation of microprocessor development tools is first reviewed, their deficiencies highlighted, and then a specific plan for their evolution is outlined. This approach is explained from a functional viewpoint of desirable characteristics for development systems (Section I) and then from an implementation viewpoint of tool construction techniques (Section II). It is apparent that the microelectronics technology will continue to develop more powerful and more complex devices. Consequently, considerable effort must be made to improve productivity and help organizations apply this technology.
CONTENTS

Introduction .................................................. 1

I. Software/Hardware Microprocessor Development -
   A Functional Viewpoint ................................. 3
   A. Fundamental Issues of Development ................. 3
   B. Current Levels of Microprocessor Support ........ 5
      1. Generation 0 - SBC Systems ..................... 5
      2. Generation 1 - MDS Systems .................... 5
      3. Generation 2 - TRIAD/Host System .............. 9
      4. Generation 3 - Multiuser, Multimicroprocessor
         Full Support System .............................. 12

II. Tools for Microprocessor Development -
    An Implementation Viewpoint ........................ 15
   A. Generation 0 ........................................ 15
   B. Generation 1 ........................................ 17
   C. Generation 2 ........................................ 21
   D. Generation 3 ........................................ 25

III. Conclusion ............................................... 26

IV. Bibliography ............................................. 27
INTRODUCTION

Microprocessors are providing powerful tools for solutions to a wide range of problems. However, as the technology accelerates, the central problem becomes how to most effectively use it. How do we apply this technology? From the user's viewpoint the biggest roadblock to microprocessor design and application is the problem of software development and maintenance. Although the hardware is very cheap and becoming more powerful, the applications software is becoming much more complex and much more expensive. In fact, the software development effort can often cost five to ten times the hardware investment needed for the task.

The current generation of software development tools has several serious flaws. In general, they are very primitive in their approach, especially in light of techniques that historically have been used on large computer systems. In addition, they often restrict the user to one manufacturer, and in those few cases where they do not, the level of software support is very elementary. The hardware technology has moved extremely fast and has provided very powerful devices, often with little or no software support mechanisms. This has traditionally been the history of computers, hardware technology leading the software technology. Software productivity is lagging far behind.

In dealing with microprocessors, it is important that the user have equally powerful software and hardware evaluation tools. The user needs to be able to apply state-of-the-art software technology to his task at hand. He needs to be able to evaluate the current technology offerings without any omissions or constraints that might make the evaluation process incomplete. Having finished the evaluation, he must then apply the chosen technology in the quickest and the most cost-effective manner. Finally, from his viewpoint he wishes to retain local control over the project and its database. This desire for local project control is characteristic of the application process.

Support systems must provide the software tools to allow these evaluations and cost-effective implementations. At present it is difficult without considerable expense and time for an individual to effectively evaluate the current software and hardware technology for his particular
task and to then carry forward the application in the most cost-effective manner. This paper deals with the development aspects of microprocessor application and discusses frameworks for future generation software/hardware development systems.

DEVELOPMENT SYSTEMS

A wide range of skills must be employed in the application of microprocessor technology to a specific problem. Software, hardware, managerial, and financial skills are all necessary to the complete success, in the true sense of the word, of the application process. This paper will deal with the software and hardware issues.

The software and hardware tools that are used in the application process are called the development system. These tools may include traditional software support vehicles such as compilers, assemblers, loaders, and operating systems, and hardware vehicles such as processors, I/O controllers, and mass storage. In addition, microprocessors have engendered newer tools such as in-circuit-emulation, real-time logic analysis, and PROM/ROS memory devices. The totality of these tools that are brought to bear on a particular application form the development system. Development systems range from the simple to the complex. They can be categorized from Generation 0 to Generation 3, as discussed in the following sections.

The microprocessor is both a hardware and software device; within the microprocessor context the two are closely intertwined. This relationship has forced users with software backgrounds to understand hardware and vice-versa. Of course, this has made their application extremely difficult. For instance, in order to understand and apply the latest generation of micros, the user faces the software and hardware complexities of a mid-sized IBM 370 computer. To help him deal with this situation, which will only become more complex with time, we must provide him with new tools and methodologies that will increase his effectiveness and productivity.
I. SOFTWARE/HARDWARE MICROPROCESSOR DEVELOPMENT - A FUNCTIONAL VIEWPOINT

As the worldwide market for microprocessors and microprocessor systems climbs toward the billion dollar level, there exists an ever increasing need for microprocessor development tools to support the engineering and development efforts of this market. Conservative industry projections estimate the market for microprocessor development systems climbing to 200 million dollars by 1980. The growth rate in the development systems business is expected to be 25% to 30% annually, compounded for the next three or four years. Leading manufacturers in the United States are INTEL, Motorola, Tektronix, and Texas Instruments, with 1978 sales of development systems at $55 million, $12 million, $8 million, $5 million, respectively, and $30 million to a scattering of others.

This report is intended to outline the major issues in the design and use of such microprocessor development systems and to highlight opportunities for new approaches to these tools. Section I discusses the different levels of support that are desirable versus those that are available from the manufacturers for microprocessor development. Section II discusses the evolution and use of these tools; their problems and relative advantages are also discussed. First, however, it is important to outline the primary issues underlying the needs for development systems and the needs for increasingly more sophisticated development tools.

I.A. Fundamental Issues of Development

There are several factors that are focusing attention on the problems of applying the technology. These include the accelerating pace of technology, pricing policies, and the variety of software offerings. A few of the major factors are:

Hardware vs Software Costs

LSI and VLSI hardware prices at both the component and system level are decreasing rapidly as their products mature. However, with the increased complexity of the new LSI devices (for example the new generation of microprocessors that are of mid 370 complexity), the cost involved for application software is increasing at an enormous rate. These problems are compounded both by the severe shortage of qualified personnel and by the rapid introduction of new technology that obsoletes previous product development efforts.
Fast Pace of Technology

The introduction of new processors and support chips is occurring at an ever-increasing rate. This phenomenon forces design engineers and managers to constantly evaluate the new offerings in light of their own application needs. A good development system should allow for such a cross evaluation at both the hardware and software levels.

New Languages

In addition to new processors, a wide variety of new software and higher level language specifications are constantly being generated. For example, PLM/PLZ/PLC and their derivatives were very popular; currently Pascal is gaining much popularity and may replace these languages. The next generation of languages may be totally different from any of these. Realistically the design engineer must be constantly evaluating what language, assembly or higher level, to use for his particular application. A good microprocessor development system should allow him to evaluate these several different languages in their native hardware processor environments.

Documentation

A major cost of any software/hardware project is the need to document the work that has been accomplished at both the code level and report generation level. A good development system should facilitate this task.

These are a few of the problems which development systems are designed to solve. However, current generations of microprocessor development systems (0 and 1) have some serious shortcomings. These shortcomings and possible directions for improvement are outlined in Section II, "Tools for Microprocessor Development - An Implementation Viewpoint". The remainder of Section I will attempt to outline the functional levels of support that are offered or should be offered by microprocessor development systems.

In the following discussions in Section I, I will use the INTEL product line only as an example of a typical support system.
The issues raised can be extrapolated to other manufacturers such as Motorola, TI, and Fairchild. Each illustration functionally breaks the development system into a hardware portion and a software portion (separated by a dashed line). Under hardware are indicated the CPUs that are supported and under software are the languages that are supported by that generation development system.

I.B. Current Levels for Microprocessor Support

I.B.1. Generation 0 - SBC Systems

The most primitive level of support for developing microprocessor applications is the single board computer (SBC) or "kit" approach. Figure I.B.1 indicates this most primitive level of support. The hardware processor supported is the 8080/8085 and the language supported is only assembler, through hand assembly. The vehicle used to support this level is an SBC 80, a single board computer card. This level of support is very basic and generally unsuitable for any significant development effort. The SBC is generally used as a target system (please see Section II.A. for further details of target systems).

I.B.2. Generation 1 - MDS Systems

Generation 1 support tools are characterized by development systems such as the MDS 888 or Siemens SME system. At the hardware level the SME supports a wide range of 4 bit, 8 bit, and 16 bit processors. Indeed as new hardware devices are added to the product line of this manufacturer, they will be included as part of the support package under the MDS-888/SME. On the software side a large development effort is expended by the manufacturer to provide assembler and language support for all processors in the product line. In this example, as shown in Figure I.B.2a, the higher level languages are BASIC, FORTRAN, PLM, and possibly in the near future PASCAL. It is important to notice that Generation I systems provide a vertical support mechanism for developing microprocessor applications within one product group dedicated to a manufacturer. In other words, an INTEL MDS 888 system will not support Motorola or TI or Fairchild products. This "vertical support" situation for several different manufacturers is indicated in Figure I.B.2b.
MICROPROCESSOR DEVELOPMENT SYSTEMS

GENERATION 0

INTEL

HARDWARE SUPPORTED

\{ 

8080/8085 

SOFTWARE SUPPORTED

\{ 

Assembler (Hand Assembled) 

SBC 80 (SINGLE BOARD COMPUTER)

VERTICAL SUPPORT PATTERN OF GENERATION 0 SYSTEM

FIGURE I.B.1
MICROPROCESSOR DEVELOPMENT SYSTEMS

GENERATION I

INTEL

HARDWARE SUPPORTED

- 8086
- 8080, 8085
- 8748
- 4040

SOFTWARE SUPPORTED

- Assembler
- 8080, 8085
- 8748
- 4040
- 8086
- Basic
- Fortran
- PL/M

Example of a Full Development System (FDS)

INTEL MDS-888 (a FDS System)

VERTICAL SUPPORT PATTERN OF GENERATION I SYSTEM

FIGURE 1.B.2a
GROWING MARKET OF DEVELOPMENT SYSTEMS
VERTICAL SUPPORT BY MANUFACTURER  (Domestic U.S., over $150M, 1979)

<table>
<thead>
<tr>
<th>HARDWARE SUPPORTED</th>
<th>INTEL</th>
<th>MOTOROLA</th>
<th>ZILOG</th>
<th>T.I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxx</td>
<td></td>
<td>68000</td>
<td>Z-8000</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>8748</td>
<td>6809</td>
<td>Z-8</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>8086</td>
<td>6800</td>
<td>Z-80</td>
<td>9900</td>
</tr>
<tr>
<td>:</td>
<td>4040</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembler</td>
<td>8080,</td>
<td>Assembler</td>
<td>Assembler</td>
<td>Assembler</td>
</tr>
<tr>
<td>8085</td>
<td>6800</td>
<td>6800</td>
<td>Z-80</td>
<td>9900</td>
</tr>
<tr>
<td>8748</td>
<td>6809</td>
<td>Z-8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4040</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8086</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE SUPPORTED</td>
<td>BASIC</td>
<td>BASIC</td>
<td>Extended BASIC</td>
<td>FORTRAN IV</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>FORTRAN</td>
<td>FORTRAN IV</td>
<td>COBOL ANSI +</td>
<td></td>
</tr>
<tr>
<td>PL/M</td>
<td></td>
<td>PL/Z</td>
<td>BASIC</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td>:</td>
<td>CONCURRENT PASCAL</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
<td></td>
<td>PASCAL</td>
<td></td>
</tr>
</tbody>
</table>

INTEL MDS-888
MOTOROLA EXORCISOR
ZILOG ZDS-1
T.I. 990 SERIES

FIGURE I.B.2b
A slight enhancement to Generation 1 level of support is offered by several systems which support a selected number of hardware processors and assembler languages across a variety of manufacturers' offerings. Examples of this are the Tektronix 8002, Futuredata and Millenium systems as indicated in Figure I.B.2c. However, because of the large investment involved in developing support languages such as BASIC, Fortran, and PLM, these systems do not support any manufacturer's high level languages, and in the best case will only support their own version of a higher level language (for example, Tektronix is developing its own version of PASCAL - labelled T-PASCAL). In addition, because these Generation 1.5 systems cannot afford to support all products within a vertical product line, there are many processors and their support languages which are intentionally left out of these systems. This is indicated in Figure I.B.2c by the cross marks indicating products which will not be supported. An idealized view of what the Generation 1.5 systems can support and will support is given in Figure I.B.2d. The major weakness of these systems has been the lack of higher level language support, and incompatibility with individual manufacturer's source. Because of the large investment required to constantly generate new compilers for different languages on different manufacturers' systems, the Generation 1.5 people will not attempt to support these higher level languages. Instead they will develop their own, which in turn locks the user into using that particular cross system.

I.B.3. Generation 2 - Triad/Host System

The next generation of MDS type systems will offer a wider range of capabilities and many features which are found lacking in Generation 1 and 1.5 systems. A full discussion of Generation 2 and its approach using a timeshared host downloading system is discussed in Section II.C. - Tools for Microprocessor Development. If one tracks the direction that Generation 1 systems are approaching, one will find the concept of a multiuser host support system as being an eventual target. This includes systems from such firms as Tektronix, Hewlett-Packard, and Intel. However, such a host downloading system has a severe problem. Its principal shortcoming is how to successfully update new software cross assemblers
MICROPROCESSOR DEVELOPMENT SYSTEMS
GENERATION 1.5

INTEL

HARDWARE SUPPORTED

\{ 

8080, 8085

8086

\}

SOFTWARE SUPPORTED

\{ 

Assembler 8080, 8085

Basic

Fortran

PFLOW

\}

OTHER MANUFACTURER'S MICROPROCESSORS

HORIZONTAL SUPPORT PATTERN OF GENERATION 1.5 SYSTEM

NOTE THAT THERE IS LIMITED VERTICAL SUPPORT

FIGURE I.B.2c
GROWING MARKET OF DEVELOPMENT SYSTEMS
GENERATION 1.5
(Domestic U.S., over $150M, 1979)

INTEL
MOTOROLA
ZILOG
T.I.

xxxx
:

8748
8086

68000
Z-8000

HARDWARE
SUPPORTED

SOFTWARE
No
Mfg. HLL Support

BASIC
FORTTRAN
PL/M

TEK
F.D.
MIL

8748
4040
8086

6809
Z-80
Z-8000

Assembler 8080,8085
Assembler 6800
Assembler Z-80
Assembler 990

LIMITED HORIZONTAL SUPPORT PATTERN OF GENERATION 1.5 SYSTEMS

FIGURE I.B.2d
and cross compilers as new processors and languages are added from different manufacturers. There are several techniques available, including: the generation of new cross translators; the development of a universal cross assembler; the use of a compiler compiler; or the use of an intermediate target code for translation. This major problem will have to be solved if a truly sophisticated development system on the order of Generation 2 support capabilities is to be developed.

I.B.4. **Generation 3 - Multiuser, Multiple-microprocessor Full Support System**

The ideal support system should allow full capability of every manufacturer's full line of products. In other words, it should represent an integration of the MDS type system for every manufacturer into one coherent cost-effective universal development system. This situation is illustrated in Figure I.B.4a. The universal development system (UDS) will support all processors and languages that are offered by each individual manufacturer. More importantly, it will also provide for efficient update of software and hardware support tools as they become available from each manufacturer. This means that the user of such a universal development system should be able to use Fortran on an INTEL microprocessor as well as on a MOTOROLA, Zilog, and TI microprocessor with full compatibility at the source level with that manufacturer's Fortran language specifications. This is quite different from offering one Fortran language that will compile into many different processors. The latter approach loses all compatibility and support features that are available from each individual manufacturer. The contrast between the UDS and Generation 1.5 system is shown in Figure I.B.4b. It is important to note that the universal development system must be cost-effective and be priced on the range from $5K to $15K per user. At this price, the UDS system will be more cost-effective and have more extensive support capabilities than other present or near future development systems.

Section II will detail the tools that are available or should be made available for microprocessor development.
UDS (UNIVERSAL DEVELOPMENT SYSTEM)
Multiuser 5K-15K/User

GENERATION 3

INTEL

MOTOROLA

ZILOG

T.I.

AUTOMATIC UPDATE (new processors)

xxxx

:

8748 8086

68000

Z-8000

8040

6809

Z-8

8080,8085

6800

Z-80

9900

AUTOMATIC UPDATE (new languages)

HARDWARE SUPPORTED

SOFTWARE SUPPORTED

Assembler 8080,8085

8748

8086

Assembler 6800

6809

Assembler Z-80

Z-8

Assembler 990

Z-8000

BASIC

FORTRAN

PL/M

Extended BASIC

FORTRAN IV

PL/Z

COBOL ANSI +

CONCURRENT PASCAL

PASCAL

FIGURE I.B.4a
COMPARISON OF GENERATION 1.5 and 3 DEVELOPMENT SYSTEMS

UDS (UNIVERSAL DEVELOPMENT SYSTEM)

HARDWARE SUPPORTED

SOFTWARE SUPPORTED

AUTOMATIC UPDATE (new processors)

INTEL

MOTOROLA

ZILOG

T.I.

GENERATION 3
Multiuser
$5K-15K/User

GENERATION 1.5
$10K-25K/User

---

8748 8086 68000 Z-8000
4040 6809 Z-8

8080,8085 6800 Z-80 9900

Assembler 8080,8085 Assembler 6800 Assembler Z-80 Assembler 990

8748 6809 Z-8 Z-8000
4040
8086

BASIC BASIC Extended BASIC FORTRAN IV
FORTRAN FORTRAN IV COBOL ANSI +
PL/M PL/Z BASIC

... ...

PASCAL

CONCURRENT PASCAL

AUTOMATIC UPDATE (new languages)

FIGURE I.B.4b
II. TOOLS FOR MICROPROCESSOR DEVELOPMENT - AN IMPLEMENTATION VIEWPOINT

In order to effectively execute a microprocessor application the correct tools, both hardware and software, must be used. In fact, the development of a prototype microprocessor application involves the use of software and hardware tools at all levels of development. In order to understand the problems encountered during this application process it is best to trace the current use of these tools in the application process.

II.A. Generation 0

A large number of people begin their microprocessor application with a small target system often composed of a single board computer (SBC). This target system has been selected on the basis of a microprocessor CPU evaluation for the particular task and it is chosen as the most inexpensive, cost-effective way to develop the software application program. An example of this technique is shown in Figure II.A where the SBC is being used as a set-point controller in a simple feedback configuration to monitor and control flow in a pipe. These SBC target systems are offered by the manufacturers of nearly all microprocessor chips. They are designed to interface to a teletype or CRT with an on-board mini-monitor program. This approach is the most primitive in terms of development techniques and is generally abandoned after the first few weeks.

The problems that are encountered with this approach are manyfold. Primary among them is that the target system has almost no software support to do program development for the particular application. In most cases it has no on-board assembler and requires the user to hand-assemble his programs before entering them through the teletype. Secondly, the target system has very limited memory space, confined to the available on-board RAM and PROM memory. Thirdly, the RAM memory, which may be used to store prototype versions of the application program, cannot permanently retain data. Upon a power down condition, the RAM memory loses all data. The final problem of this approach is the limited editing and debugging facilities available through the mini-monitor on such a single-board target system.
MOST PRIMITIVE DEVELOPMENT TECHNIQUES INVOLVING TARGET SYSTEM

PROBLEMS WITH THIS APPROACH:

1. Target System too limited for software development
2. Limited Memory Space
3. No permanent storage of programs
4. Limited editing, debugging facilities

FIGURE II.A.
The principal reason why many individuals choose this approach is that the target system is very close to the microprocessor system that will be installed in the field to accomplish the control and monitoring functions. The size and complexities of the software development effort is nearly always underestimated. For the development of applications programs greater than several hundred lines of code in length, such a target system is severely inadequate.

II.B. Generation I

The development engineer rapidly outgrows the capabilities of such a primitive development system and subsequently takes the step to acquire a full development system (FDS), as shown in Figure II.B. Typical full development systems include dual floppy disc drives, line printers, PROM programmers, in-circuit emulators (ICE), and paper tape equipment. Their software consists of: editors to help prepare programs; assemblers and compilers to translate source into machine code; and a basic operating system to manage user files and I/O devices. The FDS is primarily a software development station that is used to develop trial programs that can be tested on the target system. The development procedure is as follows:

A. The user employs the editor and floppy disc to create the source level of his application programs.

B. The user invokes the assembler/compiler resident of the FDS to produce object code.

C. The user now has the option to either execute this program for debugging purposes or to put the object file in a format that is transportable to the target system for testing purposes. These two formats are generally in PROM or in paper tape.

D. The object programs to be tested are hand carried from the FDS to the TS (in some cases the ICE facility is used, however this is limited physically to a distance of a few feet).

E. The user then tries the test program in its target system environment with possibly a small on-board debugger to highlight program errors at the target system level.

F. Where errors are found, the user returns to step A for a re-editing, reassembling and transport of the next iteration of test programs to the target system from the FDS. Or, more often than not, the user starts to implement patches in his code at the target system level to avoid the delay in steps A through E. Depending on the utilization of the FDS system
ELEMENTARY DEVELOPMENT TECHNIQUES INVOLVING TARGET SYSTEM AND FDS
(GENERATION 1)

SOFTWARE

Editor
Assemblers
Compilers
Basic Operating System

HARDWARE

CPU
memory
Line Printer
PROM Programmer
ICE
Paper Tape

FULL DEVELOPMENT SYSTEM (FDS)

PROBLEMS WITH THIS APPROACH:
1. Primitive System Software
2. No Up-To-Date Documentation
3. Slow System Response
4. Limited File Space
5. Single User
6. No Sharing Among Users
7. No Transportability (exception: TEK 8002 class)
8. Extensive Maintenance
9. Non-Uniform User Interface

Solved by
Triad *
✓
✓
✓
✓
✓
✓
✓

*Please refer to Figure II.C.

FIGURE II.B.
by several groups, the typical delay for steps A through E can range from an hour to 1 day. This delay is generally intolerable for development engineers and forces a patching mode at the object code level. This of course leads to problems in keeping documentation up-to-date with the source code level.

The procedure as outlined using these two tools, FDS and TS, has a myriad of problems.

1. **Primitive system software.** The systems programs available on the FDS, such as the editor, the file system, and the disc operating system, are fairly primitive with respect to current programming techniques. This of course is an evolutionary phenomenon but the FDS software has always lacked those capabilities available on larger mini- or midi-computer systems.

2. **Difficulty of maintaining up-to-date documentation.** Because of the long iteration time to develop a new test program, development personnel often patch object programs in the target system. This leads to very severe documentation problems which can cause large amounts of after-project effort to update programs after their development.

3. **Slow system response.** Because the FDS is generally floppy disc based and because the CPU is generally an 8-bit processor, system response has been very slow. This includes file access time as well as searching, sorting and other data-base manipulations over user files.

4. **Limited file space.** A file space provided by the dual floppy discs is marginally adequate for small development efforts. As the complexity and size of the application effort grows, as the number of duplicate copies and historical backups that are kept grows, and as documentation grows, one finds the file space available per diskette in such a floppy-based system to be oftentimes inadequate. In addition, development personnel often like to segment project work per floppy. This cuts down any potential sharing between projects because of the separate diskettes used per project.

5. **Single-user.** The FDS system is a single-user system and consequently will present a bottleneck problem when several groups must schedule their time on one system. A common approach is to buy several systems, one for each project group. This is an expensive approach as well as engendering problems of sharing and maintenance.

6. **No sharing among users.** Because each user/project has his own floppy disc for his applications program under development, there are no easy mechanisms for sharing of programs, data, documentation, and experience of a project among several users. In other words, it is not easy for one user to have access on a constant and immediate basis to the programs and files of another user's diskette. This problem of sharing is very
crucial to prevent duplication of effort among project groups and to allow the experience gained in one project to be used in the next.

7. No transportability. Generally FDS systems are dedicated by manufacturer to a specific line of microprocessor products. For instance, a Motorola development system will not support Intel microprocessors. An exception to this is a class of universal FDS systems such as the Tektronix 8002. However, even these systems have the same problems as outlined above; in addition, they offer only limited higher level software support.

8. Extensive maintenance. The FDS system essentially represents a scale-down, low-cost minicomputer system. Because its components are not on the same quality as found on larger systems, more maintenance problems are experienced. This particularly applies to the floppy disc system and the line printer. In general, FDS manufacturers do not offer maintenance contracts.

9. Non-uniform user interface. When several FDS systems are used to accommodate different manufacturers' microprocessors, the user is faced with a variety of operating systems and systems programs that he must be familiar with, depending on the particular microprocessor he is using at that time. In general, the user interface offered by the manufacturers of FDS systems varies from manufacturer to manufacturer. This creates a problem of non-uniform documentation and the need to train personnel on different systems.

It is useful at this juncture to describe a progression of attitudes best categorized as the "seventh heaven" phenomenon. When the traditional hardware/digital engineer first acquires his target system (TS) to do the microprocessor application, he is in "seventh heaven." Having been used to hardwired digital circuits, he has now under his control a complete computer system that can execute programs and perform small control tasks. However, it takes the engineer only a matter of a few weeks to realize that the target system is woefully inadequate for his development task and he becomes rapidly disillusioned with this level of support. He then purchases an FDS system with its floppy disc, line printer, and other peripherals (for about $20 - 25K) and he is again in "seventh heaven."

The FDS system is more power than he ever envisioned. It is a matter of time before the problems outlined above (#1 - #9) appear and once again he finds the FDS system to be inadequate for his needs and again disillusionment sets in. The time constant for this second disillusionment is generally longer, on the order of a year to a year and a half, depending
on his and other groups' usage of the FDS. Because there is nothing better that is available commercially, he must often make do with this level of support. Generation 2 is his next "seventh heaven."

II.C. Generation 2

The FDS does not provide all the development features that many system designers would like (or should use) such as: large shared file-handling capabilities; quick response to system commands; powerful editing programs; and a library of software for the microprocessors of different manufacturers. Designers who require such features are finding it desirable to use minicomputer (the small but high-performance computing systems that first reached the market in the early 1960s) as a supplementary development tool.

Although some cross-software packages exist for enabling minicomputers to develop microprocessor programs, there are very few systems for closely coupling minicomputers, FDS's, and the microcomputer target system (TS). Such coupling is desirable because it enables a development engineer to move easily within this hierarchy as well as to exploit the distinctive features of each system: the minicomputer to provide efficient editing, mass storage, documentation tools and shared data bases; the FDS to emulate the real-time performance of the microcomputer as programs are evolving and to debug the hardware; and the microcomputer target system itself to evaluate the final programs and control routines under the actual environmental and electrical constraints of the application setting.

The cleanest integration of these three systems is the "Triad", a tool that enables the programmer or engineer to work at any level in achieving a desired program. See Figure II.C.1. The Triad closely couples three systems: the minicomputer, with its powerful software capabilities; the FDS, with its real-time emulation and hardware-debugging facilities; and the target system, with its application-defined construction. The Triad provides fast and direct access to all levels of the hierarchy.

The hardware designer and the software programmer would use the Triad as follows. The hardware engineer develops the target system, which might
FDS is OPTIONAL
(Triad minicomputer performs 90% of FDS functions, and better)

ADVANTAGES:
1) good system software
2) up-to-date code documentation & report generation capabilities
3) fast system response
4) large file space
5) capability for multiuser
6) sharing of programs, data, past project experiences through file system
7) transportable across micros with new cross software per \(\mu m\)
8) centralized, commercially available maintenance
9) single-user interface for all \(\mu m\), for all time

PROBLEM WITH THIS APPROACH:
Software update and maintenance effort as new micros become available: i.e., bringing on-line cross assemblers and cross Higher Level Languages, preferably compatible with manufacturer's offering. (For example: ASSUME 8080, 6800, Z-80 are all supported. If PASCAL were a new language to be offered, then the system should support INTEL PASCAL, MOTOROLA PASCAL, and ZILOG PASCAL.)
consist of a commercial general-purpose, single-board computer module in conjunction with an additional module of his own design that provides an interface to the control system to which the microprocessor is being applied. In order to debug the target system he will need some test programs. Such programs can be quickly prepared with the editor on the minicomputer, cross-assembled into the machine code of the microprocessor and loaded directly into the undebugged target hardware. As the engineer works out the errors in his hardware he will continually update his test programs. It takes only a few minutes to reassemble and load each change.

Concurrently with the hardware development, the programmer can be editing, assembling and simulating his programs on the minicomputer and on the FDS. He would maintain all his programs in the central file system of the minicomputer and share a library of applications software with other users of the Triad. System integration can proceed smoothly, because the same facilities the hardware engineer has been using to load test programs into the target system hardware can also be used by the programmer to load the final applications program downward into the hardware. Changes at this stage are readily achieved by a simple process of re-edit, assemble and download. In this way the Triad system can sharply reduce the time needed for development of a microprocessor application.

Several Triad systems can be supervised simultaneously by a single central minicomputer system. Such an arrangement enables a user to develop application packages utilizing several different microprocessors while he is still maintaining all his programs, documentation and engineering reports within the file system of the central minicomputer. Moreover, other users can share his programs, thus making the development cycle more efficient. As the next step the minicomputer operating system could be rewritten to allow time-sharing among simultaneous users. This gives rise to the most powerful variation of the Triad concept as shown in Figure II.C.2.

Not only can programmers and engineers work simultaneously on development efforts concerning different microprocessors with the time-shared Triad system, but responsibilities for development and testing can be split
WIDESPREAD USE OF A TRIAD SYSTEM FOR COORDINATED MICROPROCESSOR DEVELOPMENT WITHIN AN ORGANIZATION

ELABORATIONS OF "TRIAD" SYSTEM may be warranted in large organizations where a number of microprocessor applications are under development. The system is designed to give one programmer access to several different Triads, each dedicated to a different microprocessor. In the more powerful time-sharing system engineers and programmers can work simultaneously on development efforts involving different microprocessors.

FIGURE II.C.2
among several different parts of an organization. An example of such use of a Triad system for a coordinated microprocessor development within an organization is shown in Figure II.C.2. In particular, manufacturing could have its own set of terminal and target systems to exercise products coming down the assembly line. At the same time engineering could be developing new software or diagnostics for these products with its own set of terminals and Triads. Because both organizations share the same data base, the transferral of updated diagnostics to manufacturing could be immediate. Other parts of the organization such as field service and marketing could also share this data base. Naturally, such a time-shared system would implement protection and access mechanisms to allow selective sharing among users as defined by the project management.

II.D. Generation 3

Generation 3 development tools will include all the desirable features found in the host multiuser downloading approach of Generation 2 and in addition, it will eliminate the major drawback in terms of software updating and software support of that system. As the Universal Development System (UDS, Figure I.B.4a), it will give full capability for process support at all levels of languages that are offered by each individual manufacturer. This extensive software support will be capable of being automatically updated to include new offerings of software and hardware support as each individual manufacturer evolves their product line. This capability is crucial since the growth patterns of the microprocessor market have not been able to standardize on any one processor or on any one language. We expect this phenomena to continue for the foreseeable future. Finally, the third generation development tool must also be as cost-effective as current systems, while offering increased capabilities and desirable features that are not existant on any generation 1 or 1.5 system. Under current projections, the generation 3 system is achievable using current technology and advanced design techniques.
III. CONCLUSION

As pointed out in this paper, a serious gap is growing between the hardware technology and the software tools available for the application of microprocessors to industrial and commercial situations. The technology is outstripping our ability to use it. This "applications gap" will only grow as the hardware technology accelerates and our means for software development slowly evolve. The Universal Development System of generation III is a software/hardware development tool designed to help close this applications gap. As the cost of applying microprocessor systems becomes predominantly software, the need for such a development system becomes paramount.

It is important to note that the Universal Development System retains three important characteristics that are necessary for its acceptance by the microprocessor applications engineer. These are:

* The ability to apply state-of-the-art software/hardware technology to the task at hand. In order to do this the user must have an across-the-board evaluation mechanism that allows him to compare many different manufacturers' offerings without any omissions.

* Having chosen the hardware technology, the development system must allow the quickest and most cost-effective development cycle. This is crucial to reduce the costs of programming, debugging, and documentation. At any one time, the system must also support several different microprocessor development projects using a variety of possible manufacturers' devices. The third generation UDS will fulfill these measures.

* The user wishes to retain local control over his microprocessor project during its lifetime. The third generation UDS will allow for this and, in addition, will allow for sharing of the project database during and after its development.

This paper has attempted to outline the problems inherent in microprocessor development and possibilities for future directions in the design and execution of support tools. It is apparent that the microelectronics technology will continue to become more powerful and more complex. Consequently, efforts must be made to help users apply this technology.
IV. BIBLIOGRAPHY


