CERN-Data Handling Division
DD/84/9
July 1984

DEVELOPING PROGRAMS FOR THE MOTOROLA 68000 MICROPROCESSOR AT CERN

Julian Blake, Horst von Eicken and David Foster

(Presented at the Europhysics Conference on "Software Engineering, Methods and Tools in Computational Physics, Brussels, 21-24 August 1984)

Developing Programs for the Motorola 68000 Microprocessor at CERN

Julian Blake, Horst von Eicken and David Foster
Data Handling Division, CERN, CH-1211 Geneva 23

Abstract

This paper describes tools available at CERN for preparing and testing programs for the Motorola 68000 microprocessor. The repertoire extends from cross compilers running in host computers to an interactive debugging monitor running in 68000s under test.

The cross software is described first, with emphasis on portability. The rest of the paper deals with software running in the 68000: run time libraries, and the MoniCa monitor. The facilities provided for testing microprocessor programs are compared with testing facilities in multi-user computer systems.

1. Introduction

Since 1978, CERN has built up a range of cross software for microprocessor support. The cross software runs in a variety of host computer systems, and prepares programs for execution in one of the following target microprocessors:

Intel 8080, 8085
Motorola 6800, 6801, 6809, 68000
Texas Instruments 9900, 9995, 99000

The cross software for the Motorola 68000 is the most extensive, and is complemented by target software running in the 68000. These two (host and target software for the 68000) are the subject of this paper. Cross software for the other microprocessors is described in [1].

2. Cross software

The cross software supporting the Motorola 68000 consists of the following programs:

INCLUDE: a program pre-processor allowing file inclusion and conditional text.

PASC68K, MOD68K and XF77: compilers for Pascal, Modula-2, and Fortran.

M68MIL: a macro assembler [2].

CUFLINK, PREPUSH and UNIPUSH: a CUFOM link editor, pre-pusher and pusher.

CUFOM is CERN's format for object modules [3]. PASC68K, MOD68K and M68MIL generate CUFOM directly. XF77 generates assembly language for M68MIL. CUFLINK links CUFOM modules together to form a single CUFOM module. PREPUSH allocates addresses to CUFOM sections. UNIPUSH translates a CUFOM module into an absolute module in Motorola format for loading into the microprocessor, or in DataIO format for programming read-only memory.

With the exception of XF77 which is treated below, the cross software is written in Pascal, with a few small subroutines in other languages. It should run with little modification on computers whose Pascal implementations support 32-bit integers and 8-bit characters, and provide enough addressable memory for program execution (about half a megabyte; more for compiling or assembling large programs).

The Fortran compiler XF77 is based on Bell Laboratories' F77 compiler for the UNIX operating system. It is written in C.

Table 1 shows the computer systems on which the cross software has been installed at CERN. Others, outside CERN, have installed some of the cross software on a Prime computer, on the CNET SM 90, and (we hope) on the ModComp Classic.

Table 1: Cross software availability, by program and by computer system

	(1) IBM MVS	(2) VAX UNIX	(3) VAX VMS	(4) ND 500 SINTRAN	(5) CDC NOS/BE
A. INCLUDE	(yes)	yes	yes	no	no
B. PASC68K	yes	yes	yes	no	no
C. MOD68K	no	yes	no	no	no
D. XF77	no	yes	yes	no	no
E. M68MIL	yes	yes	yes	yes	(yes)
F. CUFLINK	yes	yes	yes	yes	(yes)
G. PREPUSH	(yes)	yes	yes	yes	(yes)
H. UNIPUSH	yes	yes	yes	yes	(yes)

Computers, operating systems, and compilers:

1. IBM 3081, MVS, AAEC Pascal 1.2.
2. DEC VAX 11, UNIX 4.2 BSD, Berkeley Pascal.
3. DEC VAX 11, VMS 3.5, VMS Pascal.
4. ND Nord 500, SINTRAN 3, ND Pascal.
5. CDC Cyber 835, NOS/BE 1.5, ETHZ/Minnesota Pascal 3.

Notes:

1A. On the IBM 3081, the INCLUDE program is written in BCPL, as AAEC Pascal does not allow a program to open a file by name.

1G. On the IBM 3081, PREPUSH is replaced by a Wylbur executable file, as very few users are authorized to run interactive programs.

5. The cross software on the CDC 835 is not up to date, and is restricted to a character set of 63 characters.

3. Target software

The following system software is provided for the Motorola 68000:

Run time libraries for programs in Pascal, Modula-2, Fortran and assembly language.

The debugging monitor MoniCa [4].

The run time library code for Pascal, Modula-2 and assembly language programs can be shared between concurrent tasks. In the common case of a single application program running under MoniCa, the library is shared between the application program and MoniCa. The code for this library, and for MoniCa, is placed in read-only memory. The library is written in Pascal and assembly language.

Run time library routines for Fortran programs are loaded with the Fortran program which requires them. The library code is not sharable (in the absence of memory mapping hardware), and may not be placed in read-only memory (without software to initialize data from read-only memory before starting a program). The library is written in C, with a few routines in assembly language. It uses the Pascal run time library for primitive input/output and for floating point arithmetic.

4. MoniCa

MoniCa is a debugging monitor for the Motorola 68000. It is written in Pascal and assembly language, and incorporates the run time library for programs in Pascal, Modula-2, and assembly language.

Unlike other debugging monitors (for example, Motorola's MACSBUG and TUTOR), MoniCa uses interrupt driven input and output. Device drivers have been written for a variety of asynchronous serial line interfaces, as well as for some other devices.

MoniCa is controlled by the user at his terminal, and loads programs from the host computer or, in one configuration, from disk. In the most usual configuration, both host computer and terminal are connected to the microprocessor by asynchronous serial lines (CCITT V24 / RS 232C). In this case the microprocessor appears to the host computer system as a terminal. In another configuration, programs are loaded through CERN's packet switching network CERNET [5].

MoniCa provides the usual facilities for testing programs at the machine code level, together with additional facilities to help testers of programs in assembly or higher level languages. There are commands for disassembling (translating machine code into assembly language), and for displaying the stack of active subroutine calls with the names of the subroutines. For programs in languages other than Fortran, the user may inspect and modify the contents of program variables specified by their names (as well as by their addresses in memory). Many commands accept a repeat count, indicating the number of times the command is to be obeyed or the number of items to be displayed, as appropriate. The user may set a conditional break point, at which program execution is to stop if the given condition holds; this may be combined with a repeat count if desired. A help command gives a brief display of command syntax and purpose. A list of MoniCa commands appears in table 2.

Table 2: MoniCa commands, by category

Program control:	
BREAK	Set or display break points
CLEAR	Clear break points
LOAD	Load program
GO	Execute program
STEP	Execute one instruction at a time
Memory and registers:	
COPY	Copy a block of memory
DISASSEMBLE	Disassemble memory into instructions
FILL	Fill a block of memory
LIST	List registers or memory
MODIFY	Modify registers or memory
SEARCH	Search for a pattern in memory
Utility services:	
CHANNELS	Display input/output channels
CONVERT	Convert a number
DISTINCT	Select treatment of lower case letters
HELP	Give help
NUMBERBASE	Set number base
TRANSPARENT	Set up communication between terminal and host
Symbolic debugging:	
BLOCK	Set or display symbolic debug blocks
BOTTOM	Move to bottom frame of stack
CALLS	Display active subroutine calls
DEFINE	Define a symbol
DOWN	Move down the stack
PURGE	Remove a symbol definition
SCOPE	Display visible procedures
SYMBOLS	List symbols in debug block
TOP	Move to top frame of stack
UP	Move up the stack

5. Program testing

In many respects the testing facilities provided by MoniCa are comparable to those of a good debugger in a multi-user computer system (for example, Control Data's Cyber interactive debugger and Digital Equipment's VMS debugger). There are however a number of ways in which the testing of microprocessor programs differs from program testing in a multi-user system. We indicate some of these in this section.

The typical microprocessor configuration lacks the following facilities which are present in multi-user systems:

> Memory protection (other than the protection provided by read-only memory). It is, for example, possible to disable MoniCa's input/output by overwriting buffer pointers.
>
> A storage device (or a network connection to a host) allowing data to be read from and written to files. This prevents systematic regression testing, in which specimen data is fed to a program and the program's output is compared with previously verified specimen output. In many cases, the microprocessor configuration does not even have a printer (or similar hard copy device).

The testing cycle, consisting of edit, build, load and test, is slower for microprocessor programs. Editing and building take place in a host computer, and use cross software which is relatively slow (this is one of the prices of portability). And program loading is slow, especially where terminal lines are used to connect host computer and target microprocessor. The use of local area networks should help here.

6. Conclusion

We have developed, and are continuing to develop, a coherent set of software for the Motorola 68000. This coherence shows itself in various ways, some visible to the user of the software, others of interest to the implementers:

A 68000 program may be built of routines written in different languages, which can be linked together and, subject to certain restrictions, can call each other.

The debugging monitor MoniCa provides program testing facilities for programs in all the supported languages.

MoniCa runs in several different hardware configurations, supplied by different manufacturers or combinations of manufacturers. The user does not have to learn a new command language when he moves to a new configuration.

The same run time library is used by programs in Pascal, Modula-2 and assembly language, and by MoniCa.

The Pascal and Modula-2 compilers use the same code generator.

The cross software programs, other than the Fortran compiler, are written in the same programming language (Pascal).

In practice the ideal of coherence is not achieved completely, as the Fortran implementation comes from a different background to the rest of the software.

The software presented here is only a part of the microprocessor activities at CERN. A wider view may be obtained by reading [6].

Acknowledgements

The Pascal 68000 compiler was developed by Dittmar Krall and Ulrike Weng-Beckmann at Siemens Research Laboratories in Munich [7]. It has been extensively modified at CERN.

The Fortran compiler XF77 and the corresponding run time library were prepared for CERN by Willem Wakker of Associated Computer Experts BV, Amsterdam [8]. The compiler was developed from the UNIX Fortran compiler F77 [9], and uses a Motorola 68000 version of the code generator of the portable C compiler [10]. The run time library is a slightly modified version of the run time library of F77.

CUFOM, the CERN universal format for object modules, was designed at CERN by Jean Montuelle, Ian Willers, and others [3].

References

1. Cross software for microprocessor program development at CERN. H. von Eicken, J. Montuelle, I. Willers, J. Blake. Proceedings of the Topical Conference on the Application of Microprocessors to High Energy Physics Experiments. CERN 81-07, July 1981.

2. Software support for the Motorola 68000 microprocessor at CERN: M68MIL cross macro assembler. Horst von Eicken. CERN 83-12, December 1983.

3. Cross software using a universal object module format CUFOM. Jean Montuelle and Ian Willers. EURO IFIP 79, edited by P. A. Samet. North-Holland Publishing Company, 1979.

4. Software support for the Motorola 68000 microprocessor at CERN: MoniCa symbolic debugger. Horst von Eicken. CERN report in preparation.

5. CERNET - a high speed packet switching network. Edited by J. M. Gerard. CERN 81-12.

6. CERN Mini and Micro Computer Newsletter. Produced at CERN about four times a year.

7. Pascal 68000 user's guide, release 2.1. Dittmar Krall and Ulrike Weng-Beckmann. Siemens AG, ZTI INF 21, Munich. December 1982.

8. The ACE compilers for MC 68000. Willem Wakker. Associated Computer Experts BV, Amsterdam. August 1983.

9. A portable Fortran 77 compiler. S. I. Feldman and P. J. Weinberger. UNIX programmer's manual. Bell Laboratories, August 1978.

10. A portable compiler: theory and practice. S. C. Johnson. Proceedings of the Fifth ACM Symposium on Principles of Programming Languages. 1978.