The MONADS II Computer System

David Abramson
Department of Computer Science
Monash University
Clayton 3168

ABSTRACT

In this paper we present the underlying design goals and main features of the MONADS II computer system, which was developed in the department of Computer Science at Monash University in 1980. Unlike many other processors MONADS II provides direct support for the decomposition of large software systems by recognizing and protecting software modules. We discuss the limitations of the system together with some solutions. The design of another processor, MONADS II/2 is also described.

This work was supported by grants from the Australian Research Grants Committee (ARGC Grant F80/15191), the Australian Computer Research Board and the Monash Special Research Fund.

1. INTRODUCTION

This paper describes the MONADS II and MONADS II/2 processors, which were built in the department of Computer Science at Monash University. These processors provide direct support for software modules, and include special hardware for protecting sharing and addressing the information within the modules.

The first section briefly describes the MONADS software methodology, and the special needs it places on a processor. We define the aims of the MONADS II processor and describe the key modules in the system. The last section outlines some of the limitations of the system, and shows the configuration of the MONADS II/2 processor, which was designed to remove some of these limitations.

2. THE MONADS PROJECT

2.1. Aims of the MONADS Project

The MONADS project (Keedy, 1978, 1981; Keedy, Abramson, Rosenberg and Rowe, 1982) began in 1976 with the intention of investigating methods for developing large software systems. The techniques used in the project are based on the information hiding principle, as advocated by Parnas (1971, 1972) and others (Wirth, 1977; Liskov and Zilles, 1974; Keedy and Rosenberg, 1981; Keedy and Rosenberg, 1982). Using this principle software systems are decomposed into small information hiding modules, each of which performs a specific task.

In the past, processors have neither recognized the existence of information hiding modules, nor protected the information within them. Consequently, ad hoc software techniques which are usually complex and inefficient have been used to provide the correct environment. The MONADS project aims to provide architectural support for software modules by using them as a basic unit of addressing and protection. For these reasons all objects within MONADS (both modules and the information within them) are addressed by capabilities (Fabry,1974).

2.2. History of the Project

The first phase of the project involved building an operating system (Keedy, 1978). An operating system was chosen because it was an example of a large software system which could be constructed using the information hiding principle, thereby demonstrating the effectiveness of this technique, and also because it would serve as a useful development tool for further work. Since conventional computer architectures are ill-suited to the MONADS methodology, the MONADS I processor was developed in about 1978 (Hagan and Wallace, 1979; Wallace, 1978; Hagan 1977) from a modified Hewlett Packard HP2100A.

Unfortunately, the MONADS I hardware did not understand software modules and could not protect sensitive information, such as call linkage data on the process stack. Also, since much of the support was provided in software the system was quite inefficient. Because of the limitations of the MONADS I system, the author developed the MONADS II processor in 1980, again from a Hewlett Packard HP2100A minicomputer. This paper describes the MONADS II computer system.

3. The MONADS II SYSTEM

3.1. Aims of the MONADS II System

The MONADS II hardware has a number of major aims:

(1) To provide a processor which understands and supports information hiding modules. This can then form a pilot system for future software development work on the MONADS project.

(2) To remove the restrictions of the MONADS I computer system. This required a new addressing scheme and special hardware support.

(3) In order to protect and address segments of a module a hardware capability register addressing scheme was devised (Abramson, 1982b). The MONADS II processor was considered a test bed for this scheme.

(4) In order that an working implementation could be realised by one post graduate student in a limited period of time, a technique for implementing architectural enhancements was devised (Abramson, 1982a). The MONADS II processor was considered a test bed for these ideas.

(5) The MONADS II computer uses a large paged uniform address space for all data and code (Keedy, 1980). To efficiently implement this space a new address translation scheme was designed (Abramson, 1981). The MONADS II system was used to demonstrate the effectiveness of this solution.

3.2. A System Overview

The MONADS II system, shown in Figure 1, is constructed around a HP2100A minicomputer. This processor provides all of the basic computing facilities, such as a standard instruction set, an input-output system, and a microprogrammed control unit. All of the complex addressing modes which are required by the MONADS architecture are provided by the intermediate processor, which is a fast microprogrammed processor. This unit develops 31 bit virtual addresses, which are translated into main memory addresses by the virtual memory manager. The current system configuration is connected to 400k bytes of semi-conductor memory.

The peripherals connected to the system include two 5M byte disk drives, one 80M byte Winchester disk, and a terminal multiplexor with up to 16 terminal lines. The processor may communicate with either another unmodified HP2100A for diagnostic purposes, or with a VAX 11/780 computer, which is currently used for program compilation.

4. THE MONADS II HARDWARE

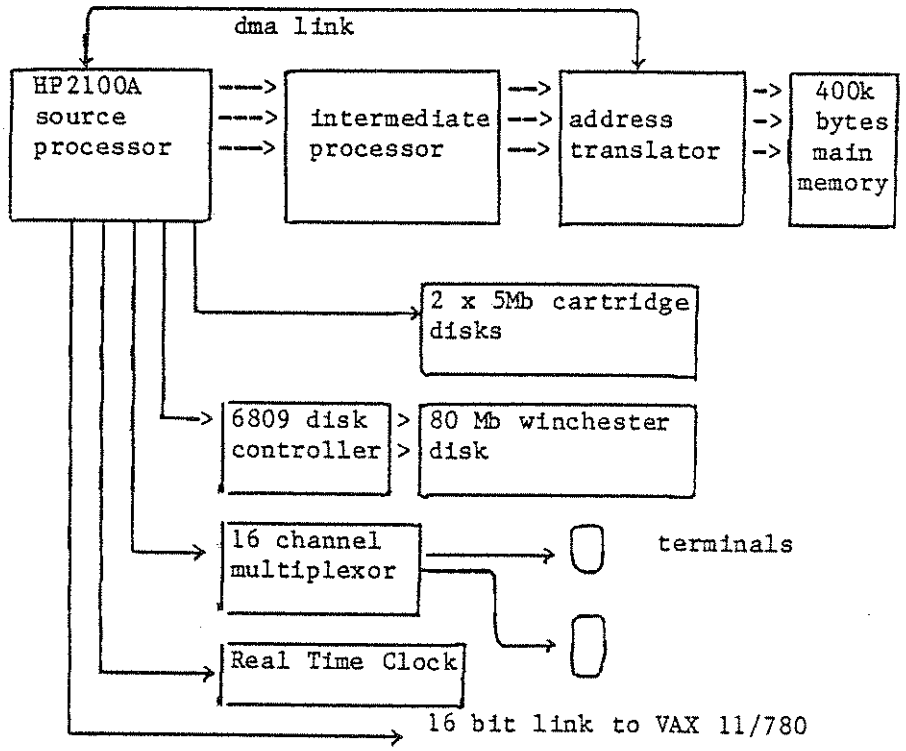

Figure 1 - the MONADS II Computer System

4.1. Functionality

4.1.1. Information Hiding Subsystems

The basic unit of software in the MONADS II system is an information hiding subsystem. This is composed of a number of memory segments, as described in Keedy (1982). Both subsystems and segments of memory are addressed by capabilities (Fabry, 1974), the format and nature of which are understood by system firmware. Attached to each active subsystem are a number of segment lists. These lists describe the location and nature of the data which may be addressed. In order that memory segments can be efficiently addressed, the firmware and segment lists are augmented by a special hardware unit, described in Abramson (1982b).

4.1.2. Addressing

All code and data are addressed by a uniform set of capability registers, as shown in Figure 2. A capability register contains a virtual base address of a segment, a segment length, and a set of access rights. When a process wishes to address a particular location within a segment, the capability must first be retrieved from the correct segment list, and placed in a capability register. This is performed by a special instruction, called 'Load Capability Register'. The capability register number may then be used together with an offset as the operand of any memory reference instruction, as shown in Figure 3. Special index registers can also be used to modify the contents of the capability register, and this can be used to access different locations within a segment of virtual memory.

A process can only address the segments which it is allowed to because the segment lists are protected from corruption and may only be addressed by special firmware. A number of capability registers are set aside for addressing data on a procedural stack. This allows MONADS II to support a stack structure suitable for protected procedure calls and block structured languages.

4.1.3. The Virtual Memory

The MONADS II system uses a large uniform virtual memory to hold all of its computational and permanent data (Rosenberg and Keedy, 1981). The

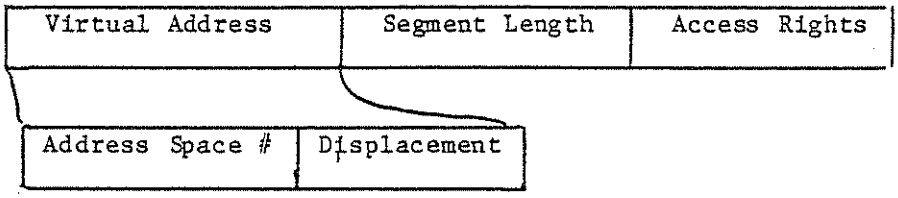

Figure 2 - a capability register

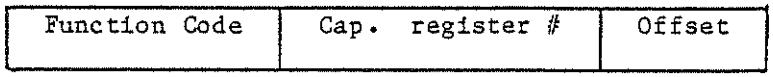

Figure 3 - a memory reference instruction

capability mechanism implements memory segments, as described in Keedy (1980) and a demand paging system facilitates easy virtual memory management. The scheme avoids many of the disadvantages of similar segmented and paged virtual memories, such as the high cost of transferring small segments and the space lost due to external fragmentation by actually mapping segments into paged virtual memory rather than into physical memory, as is common in many other segmented systems (e.g. the B6700 (Organick, 1973), Intel iAPX432 (Intel,1981)).

4.2. Implementation

4.2.1. The HP2100A

The main processor used in the MONADS II system is a Hewlett Packard 2100A minicomputer. This machine is used to provide all of the basic computer facilities, such as a basic instruction set, input-output devices, direct memory access channels, etc.

In order to implement the extra functions and addressing modes in the MONADS II system, the memory controller logic has been removed and an interface to the intermediate processor substituted. The address space of the HP2100 is then mapped into a number of MONADS II functions. In this way registers such as the new capability registers may be used as an operand for a conventional HP2100 memory reference instruction.

4.2.2. The Intermediate Processor

The intermediate processor implements many of the functions and addressing modes required by the MONADS software structures, but which are not provided by the basic HP2100 instruction set, or cannot be implemented in HP2100 microcode. These include the capability registers, the capability modifier registers, loop counter registers, extra addressing modes, such as indexed, immediate, etc. Functions provided are process switching, process timing etc. One of the major functions of the intermediate processor is to expand the addressing range of the HP2100 to a larger 31 bit address.

4.2.3. The Virtual Memory Manager

The virtual memory manager is a stand alone module of hardware which translates a 31 bit paged virtual address into a 22 bit main memory address. If the page is not resident in main memory then the hardware kernel is activated, and page tables consulted to find the page in secondary memory. The virtual memory manager hardware uses a hashing technique for associatively retrieving map entries for virtual pages. The table is held in high speed bipolar memory, and uses embedded link pointers to resolve overflow conditions. The scheme is further discussed in Abramson (1981).

5. THE MONADS II SOFTWARE

5.1. The Hardware Kernel

In order to provide the MONADS operating system with a high level hardware interface, a hardware kernel is provided (Rosenberg, 1979; Wallis, 1980). This body of code provides virtual memory management, process management, input-output management and various system functions, such as an 'inter-module call' instruction. Many of the important components of the hardware kernel are implemented in microcode, however, the remaining functions are provided by software.

5.2. The Operating System

The main operating system is composed of a number of information hiding subsystems. Unlike many other systems, there is no clear distinction between operating system services, and user programs. The capability mechanism allows some modules to address other modules simply because they have their capabilities, rather than any special operating system protection mechanism. An in-process structure (Ramamohanarao, 1980) is used to allow any user subsystem with a valid capability to call any system service. The protected stack allows calls to be safely nested to any depth.

6. CURRENT STATE

6.1. Hardware

The MONADS II hardware is complete and is being used for software development. New hardware (e.g. more memory, a hardware assisted debugger unit, etc.) may be added to the basic configuration at a later date.

6.2. Software

At present a number of compilers, utilities and a basic operating system have been developed. These include a modified MODULA II compiler (Dawson, 1982), a Pascal compiler, an Assembler and a C compiler. All of these are cross compilers and execute on a VAX 11/780. Code is then down line loaded to the MONADS II processor. When a complete operating system is provided they will execute on the processor itself. The operating system will consist of modules such as a process scheduler, a command line interpreter and a file directory manager, much of which is already written.

7. LIMITATIONS

The MONADS II system has a number of limitations.

(1) The virtual address is only 31 bits in length. In a capability based scheme addresses are never reused even when the objects they address are discarded. Unfortunately a 31 bit address does not provided enough unique names, and the total number would be exhausted quite quickly. This was not considered a problem in MONADS II as the system was only intended as a pilot study. Later in the paper we will discuss another processor, MONADS II/2, which has a larger address size, and thus solves this problem.

(2) Some of the special purpose capability registers in the processor differ from the model proposed in Abramson (1982b). These registers could easily be modified to conform to the format of the general capability registers and were only different for historical reasons.

(3) When the virtual memory manager was built a very simple algorithm was chosen for the internal hashing function which is used to locate virtual addresses in the hash table. Consequently the translation unit does not perform as well as possible. This function could easily be made more complex now that the basic design has been built and tested.

(4) The intermediate processor, which implements all of the extended addressing modes does not execute as fast as possible. This can easily be modified now that the basic processor has been built and tested, and was initially slower because the microcode control store was built from erasable memories rather than read only memories.

(5) Even after the extensive modifications to the HP2100A processor MONADS II still uses the basic HP2100A instruction set which is not ideally suited to the MONADS software methodology. The only solution to this

problem is to build a totally new processor.

(6) In the first version of the Hardware Kernel a very simple page discard algorithm was chosen which may not find the best page to remove from main memory. Provided that some extra support hardware is added (such as use and change bits (Morris, 1972)) this algorithm could be made more complex.

8. MONADS II/2

To alleviate some of the limitations of the MONADS II system another processor, the MONADS II/2 system, is being built. This processor will have a much larger addressing range (with a 47 bit address), more main memory, more secondary memory, and a different direct memory access system.

The larger virtual address means that more objects can be created before all of the available addressing regions are exhausted. The larger main memory and secondary memory (1/2 Mbyte of main memory and 240 Mbytes of secondary memory) will allow the system to support a much larger user community.

Because of problems with the basic input output system of MONADS II, the I/O system for MONADS II/2 has been redesigned. Instead of using the basic HP2100 I/O system and Direct Memory Access hardware, MONADS II/2 uses two Intel Multibuses controlled by a Z8000 micro-computer. One of the Multibuses allows the main processor to access memory. The other Multibus is used for Input output operations, and can be linked to the first bus when data must be transferred to or from main memory. This arrangement gives priority to the main MONADS processor, which must address memory to access operands and fetch instructions. All input output transfers are directed via the Z8000 which arranges the transfer between main memory without further intervention from the HP2100. Furthermore, the Z8000 is alerted when a page fault is generated, and is responsible for fetching the correct page into main memory and for finding a free page frame.

An intelligent terminal multiplexor is also attached to the second Multibus (Carra, 1982). This device controls the interactive terminals connected to the system, releaving the main processor of the time consuming task of handling terminal interrupts. The new input output scheme should leave the HP2100 free to execute more user programs, and also reduces the size of the hardware kernel significantly. A diagram of MONADS II/2 is shown in Figure 4.

9. CONCLUSION

The MONADS II system provides direct firmware support for information hiding modules and uses hardware to implement an efficient capability based addressing scheme. A protected procedural stack allows an in-process operating system to be safely constructed. Consequently, the hardware lends itself to the MONADS methodology far more than most other processors.

The capability register addressing scheme has been shown to be efficient and flexible. The enhancement technique has allowed a working processor to be constructed very quickly. The address translation scheme is also efficient, and has many advantages over alternative techniques.

The MONADS II/2 machine should provide a larger and more reliable implementation of the basic processor, and is scheduled for completion in 1983.

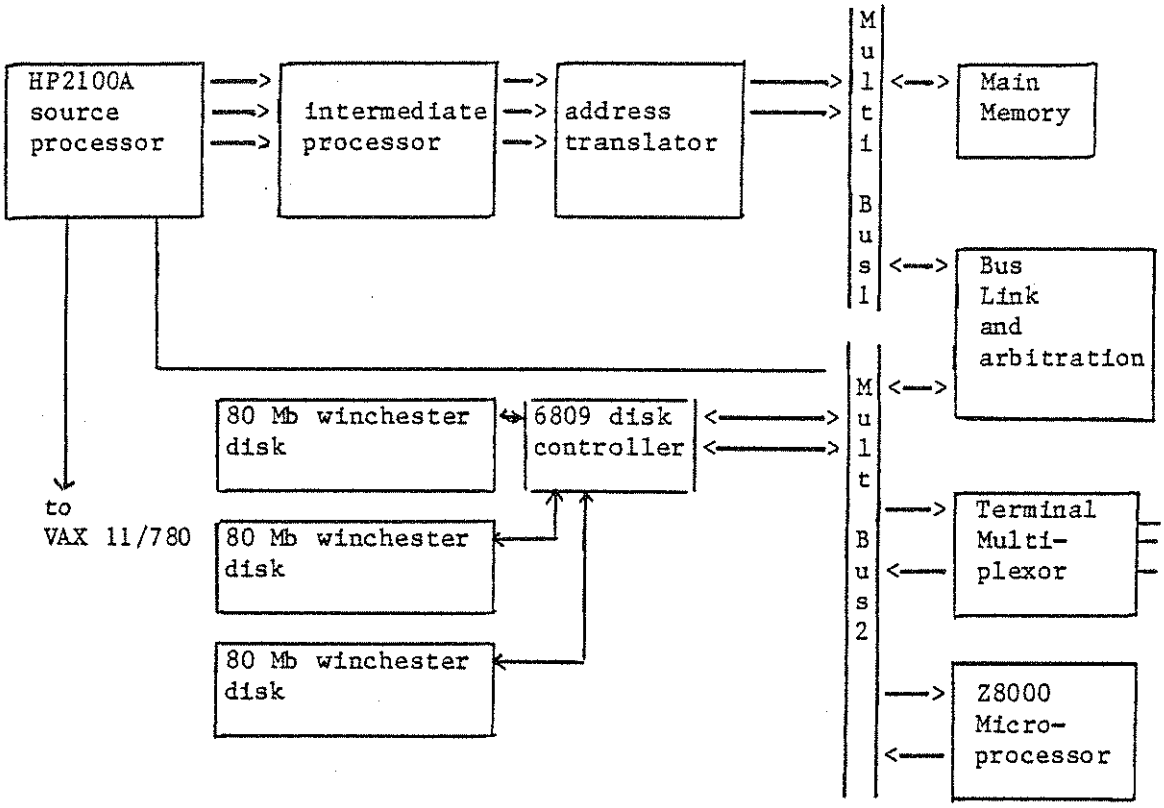

Figure 4 - the MONADS II/2 Computer System

ACKNOWLEDGEMENTS

We gratefully acknowledge funding from the Australian Research Grants Committee, the Australian Computer Research Board and the Monash Special Research Fund.

The MONADS II computer system has been the work of many people, in particular Dr. J.L. Keedy, Dr. J. Rosenberg, Mr. D.M. Rowe, Mr. P. Dawson, Mr. M. Evered, Dr. E.F. Gehringer, Mr. M. Halpern, Mrs. S. Rees, Mr. J.V. Thomson, Mr. B. Wallis, Mr. J. Wells. Thanks are also due to Brian Wallis, who read a draft of this paper.

REFERENCES

Abramson, D.A. (1981) "Hardware Management of a Large Virtual Memory", Proc. 4th Australian Computer Science Conference, Brisbane (Australian Computer Science Communications 3, 1, pp. 1-13).

Abramson, D.A. (1982a) "A Technique for Enhancing Processor Architecture", Proc. 5th Australian Computer Science Conference, Perth (Australian Computer Science Communications 4, 1, pp. 47-57).

Abramson, D.A. (1982b) "Hardware for Capability Based Addressing", Proc. 9th Australian Computer Conference, Hobart.

Carra, A.A. (1982) "A Front End Processor for MONADS III", Department of Computer Science, Monash University, Honour Report.

Fabry, R.S. (1974) "Capability Based Addressing" Comm. ACM, Vol 17, Num 7, pp. 403-412.

Hagan, R. (1977) "Virtual Memory Hardware for a HP2100A Minicomputer", M.Sc. Thesis, Monash University.

Hagan, R.A. and Wallace, C.S. (1979) "A Virtual Memory System for the Hewlett Packard 2100A", ACM Computer Architecture News, 6, 5, pp. 5-13.

Intel (1981) "Introduction to the iAPX432 Architecture", Intel Corp. Manual Order No. 171821-001.

Keedy, J.L. (1978) "The MONADS Operating System", Proc. 8th Australian Computer Conference, Canberra, pp. 903-910.

Keedy, J.L. (1980) "Paging and Small Segments: A Memory Management Model", Proc.8th World Computer Congress, IFIP-80, Melbourne, pp. 337-342.

Keedy, J.L. (1981) "A Progress Report on the MONADS Project" Australian Computer Science Communications, 3, 2, pp. 270-277.

Keedy, J.L. (1982) "The Internal Structure of Modules in the MONADS System", To be published

Keedy, J.L. and Rosenberg, J. (1981) "Information Hiding - A Key to Successful Software Engineering", Proc. Conference on Computers in Engineering, 1981, Institution of Engineers, Australia, Publication No. 81/8, pp. 1-5.

Keedy, J.L. and Rosenberg, J. (1982) "Architectural Support for Software in the MONADS III Computer Design", Proc. 12th Annual Conference Gesellschaft fuer Informatik, Kaiserslautern, 1982.

Keedy, J.L., Abramson, D., Rosenberg, J. and Rowe, D.M. (1982) "The MONADS Project Stage 2: Hardware Designed to Support Software Engineering Techniques", Proc. 9th Australian Computer Conference, Hobart.

Liskov, B. and Zilles, S. (1974) "Programming with Abstract Data Types", Proc. A.C.M. Sigplan Conf On Very High Level Languages, A.C.M., Sigplan Notices, Vol 9, Num 4, pp. 50-59.

Morris, J.B. (1972) "Demand Paging through Utilization of Working Sets in the MANIAC II", Comm. ACM, Vol 6, Num 1, pp 1-17.

Organick, E.I. (1973) "Computer Systems Organization, the B5700/6700 Series", Academic Press, New York.

Parnas, D.L. (1971) "Information Distribution Aspects of Design Methodology", Proc. 5th World Computer Congress, IFIP-71, pp. 339-344.

Parnas, D.L. (1972) "On the Criteria to be Used in Decomposing Systems into Modules", Comm. ACM, 15, 12, pp. 1053-1058.

Ramamohanarao, K. (1980) "A New Model for Job Management Systems", Ph.D. Thesis, Monash University.

Rosenberg, J. (1979) "The Concept of a Hardware Kernel and its Implementation on a Minicomputer", Ph.D. Thesis, Dept. of Computer Science, Monash University.

Rosenberg, J. and Keedy, J.L. (1981a) "Software Management of a Large Virtual Memory", Proc. 4th Australian Computer Science Conference, Brisbane, pp. 173-181.

Wallace, C.S. (1978) "Memory and Addressing Extensions to a HP2100A" Proc. 8th Australian Computer Conference, pp. 1796-1811, Canberra.

Wallis, B.R. (1980) "A Hardware Kernel for the MONADS II Computer", Honours Thesis, Department of Computer Science, Monash University.

Wirth, N. (1977) "Modula - A Language for Modular Programming", Software-Practice and Experience, 7, 1, pp. 3.

6th Australian Computer Science Conference

10–12 FEBRUARY 1983

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS

VOLUME 5, NUMBER 1

BASSER DEPARTMENT
OF COMPUTER SCIENCE
UNIVERSITY OF SYDNEY
N.S.W. 2006 AUSTRALIA

ACSC 6