THE FAST AMSTERDAM MULTIPROCESSOR (FAMP) OPERATING SYSTEM

D. GOSMAN, L.O. HERTZBERGER, D.J. HOLTHUIZEN, G.J.A. POR, M. SCHOOREL
NIKHEF-H, Amsterdam, The Netherlands*

Abstract:

The Fast Amsterdam MultiProcessor system (FAMP system) is developed for on-line filtering and second stage triggering. The system is based on the MC68000 microprocessor from MOTOROLA. In this report we will describe:

- the FAMP operating system software,
- the features of the slaves and supervisor in the FAMP operating system,
- the communication between supervisor and slaves using the dual port memories,
- the communication between user programs and the operating system.

The hardware as well as the application of the system will be described elsewhere [1,2,3].

1.0 Introduction

In this paper we describe some features of the software for the FAMP system, used for on-line filtering and second stage triggering. The FAMP system consists of a number of processing cells, each having its own program memory, I/O interfacing and data memory. These processing cells are connected using so-called dual port memories (fig. 1).

The operating system has to support separate processors as well as the multiprocessor environment. The system may be used in the "data acquisition" mode as well as in the "operating system" mode. The data acquisition mode supports the running of time critical user programs, e.g. trigger programs, whereas the operating system mode supports the not time critical functions, e.g. running of testprograms and system monitoring tasks. As a consequence user programs must perform with absolute priority, whereas the operating system tasks have to run as background programs. The two modes are selected by external interrupts to the supervisor.

In the data acquisition mode experimental data are fed to the slaves from the data memories. The slaves perform calculations on the data and give the results through dual port memories to the supervisor. There the information from all the slaves is combined and a decision is taken. For accepted events data are transmitted to the supervisor and from there on to the host computer. The fact that the system has to run in two distinct modes demands special software solutions, for instance concerning the management of the data stream through the dual port memories and the interaction of the user programs with the system. In the FAMP operating system the interruption facilities are provided by the system software, running in the operating system mode.

Cross-compilers like C and PASCAL now exist on some host computers, but are far from efficient. Therefore the software has been written in assembly language.

The multiprocessor operating system described here is intended for one supervisor cell (level 2) and N slave cells (level 1) (see fig. 1).

In section 2 the operating system is discussed, in section 3 more is said about the data acquisition mode. Finally in section 4 the message structure in the system communication through dual port memories is discussed in more detail.

2.0 The FAMP operating system mode [4,5]

In most applications of the FAMP system, the processing cells will form a tree, with one cell at the root of the tree, the so-called supervisor processing cell. The supervisor processing cell has certain control functions over the other slave processing cells. The operating system for the supervisor contains:

- the kernel,
- the minimum operating system,
- the extended operating system.

The operating system for the slave cells is in principle the same but contains no extended operating system for reason of memory space.

2.1 The kernel

The kernel provides the elementary system functions. It takes care of:

- system initialization,
- exception and interrupt processing.

System initialization includes the setting of some individual vectors, I/O initialization and loading of system programs in the RAM working space of the operating system. In each of these situations the MC68000 performs an exception processing sequence. Such a sequence also occurs in case of a bus error, a trap or in the case of tracing[6].

The MC68000 has two states of operation, one is the supervisor state and one is the user state. In the supervisor state privileged instructions are allowed, in the user state they are not. Our software is written such that in the operating system mode the MC68000 is always in the supervisor state, whereas during the data acquisition mode the processor

is in the user state. As user and supervisor have separate stackpointers it allows for protection of the stacks used by the operating system programs and those handled by the user programs. The switch from user to supervisor state is made automatically in any of the following circumstances:

- a user program calls on the operating system to execute some function requiring the use of a privileged instruction. Such a call is termed a supervisor call (the MC68000 TRAP instruction is used for this purpose),
- an interrupt occurs,
- an error condition occurs in a user program.

The switch from supervisor state to user state goes through a privileged instruction.

One of the essential features in a multiprocessor operating system is to monitor the behaviour of the various processing elements. For example, as a protection against hang up, a time-out interrupt is given when one of the processing cells is not responding within a certain time limit.

In the system described here memory protection is thought to be indispensible. Therefore the operating system itself resides in EPROM. Moreover, the first 2K Byte of the 32K Byte available program memory of each processing cell is only accessible in the supervisor state, thus preventing the user programs from reading or writing in it. All the programmable exception vector subroutine addresses and the scratch RAM used by the operating system, reside in this part of the program memory.

2.2 The minimum operating system

The minimum operating system contains the more elaborate functions, like:

- hardware testprograms,
- dual port memory input/output,
- debug facilities,
- the command interpreter,
- down line loading facilities.

The hardware test programs check the behaviour of memories and interfaces.

The communication between supervisor and slaves is accomplished in the operating system mode by exchanging messages and associated data. Messages are written in a dual port memory module by one process and read by another. The main problem is the synchronisation of these tasks. An evident problem arises when two processors want to write or read at the same memory location concurrently (the mutual exclusion problem). In section 4.0 we describe how these problems are solved and how the dual port memory I/O takes place.

Debug facilities are for instance: displaying and changing of memory and registers (including the stack pointers), and instruction by instruction tracing.

The command interpreter acts in the following way: a message can consist of a command entered at the console connected to the supervisor processor. If this command is not to be destinated for the supervisor processor, it has to transmit the message and associated data to the destination slave cell. The destination is specified by prefixing the message with the number of the slave cell. A check is made for the occurrence of errors in the command message. When it is correct it will be executed. Also messages without operator action can occur, for instance signals coming from the experiment can be translated into messages.

Down line loading of user programs can take place from the host computer to the supervisor and/or slaves, or from EPROM at the supervisor level to the slaves. The loading from a host computer is done via a serial line, loading from the supervisor is performed using the dual port memories.

2.3 The extended operating system

The extended operating system offers, in addition to the minimum operating system, the man-machine interface. So it contains I/O routines which can communicate with the outside world, via a terminal. In the future it should provide some mass storage.

3.0 The data acquisition mode

The software of the data acquisition mode has to provide the user the facilities to run his time critical programs on the system. The transition from the operating system mode into the data acquisition mode will always be caused by an interrupt from the experiment. As soon as the system is in the data acquisition mode communication goes without using interrupts because of the relatively long response time (interrupt processing takes approximately 25 micro seconds). The synchronisation between processing cells is realized now by setting and polling bits in status registers present in the dual port memory. Because this is experiment dependent the user should write this part of the communication. Only an interrupt is able to bring the system back in the operating system mode.

4.0 Dual port memory input/output

In our system the software has to control two distinct data flows, which originate from two asynchronous processes, i.e. the data of the experiment managed by the user programs and the information for the interprocessor communication managed by the operating system. Since these processes are not synchronized and both streams have to pass the same dual port memory, this memory is divided in two parts. The part used by the operating systems is hardware protected, i.e. only accessible in the supervisor state.

To solve the mutual exclusion problem, the fraction of the dual port memory used by the operating systems is divided in blocks. The data structure used to manage these blocks consists of three circular linked lists (fig. 2) [7]. One list contains messages. The other two lists are maintained to keep track of free memory blocks, one is used for free message space, the other for free data space. A message block consists of the following words:

- the address of the processor of destination,
- if necessary a pointer to a data block,
- reserved,
- a command,
- a pointer to the next message block.

When data is associated with a message a pointer in the message block will point to the data block, which is released from the circular linked list of free data blocks. Insertion and deletion of messages and linking data to a message is a matter of changing pointers in an obvious manner. The pointer is the only possible access to a block. The access to the pointer is protected by a semaphore. The testing, and when not set, the setting of a semaphore, must be an indivisible operation. The MC68000 provides such an instruction, which makes the processor suitable to operate in a multiprocessor environment. Blocks can be removed from the scope of other processes by changing one pointer value. This is illustrated in fig. 2.

An advantage of this approach is that it allows dynamical dual port memory management without the expensive need of garbage collection. Additionally the number and sizes of the two kind of blocks can be specified during system generation time, which adds to the flexibility of the system.

To keep the programming for the exchange of messages via the dual port memory simple, the following macro's have been written:

SULIST, sets up the circular linked lists for the dual port memories, during system initialization. This task is performed by the minimum operating system of the supervisor.

NEW, takes a message block from the empty list, and links the desired number of data blocks.

DISPOSE, puts a message block back in the empty list, and unlinks the data blocks.

FETCH, takes a message from the full message list.

SEND, puts a message block in the full message list.

Acknowledgements:

The authors would like to mention the work of P. Kuyer, especially on the dual port memory I/O system, and of M. Carels.

* This work is part of a joint program of FOM and ZWO.

References:

1) L.O. Hertzberger et al., FAMP system, these proceedings.

2) K. Eggert et al., Fast On-line Track finding for the Muon Trigger in the UA1 Experiment, these proceedings.

3) L.W. Wiggers et al.: Applications of the FAMP system in experiment NA11 in the SPS, these proceedings.

4) L.O. Hertzberger: The fast amsterdam multiprocessor (FAMP) system, NIKHEF-H/81-3.

5) L.O. Hertzberger et al., Proc. EPS Conf. on Computing in High Energy and Nuclear Physics, Bologna 1980, Comp. Phys. Comm. 22 (1981) nrs. 2, 3, p.253-260.

6) MC68000 microprocessor user's manual.

7) E. Horowitz and S. Sahni: Fundamentals of data structures.

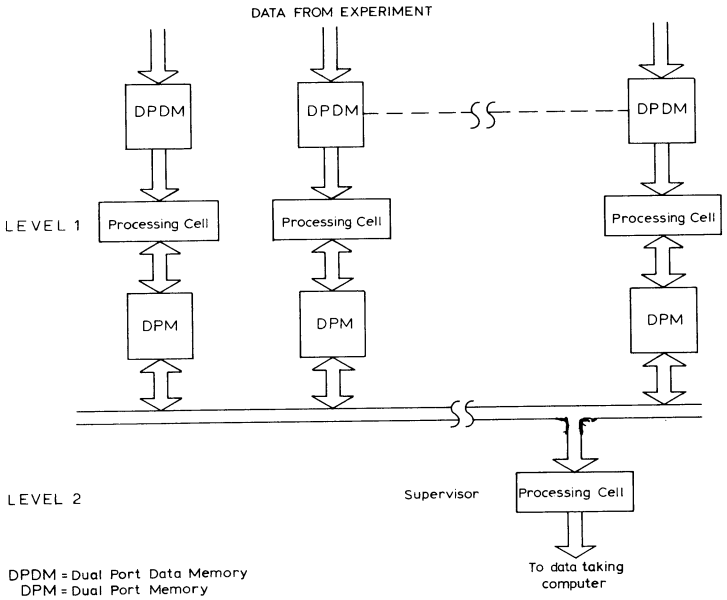

Fig. 1 Two level system

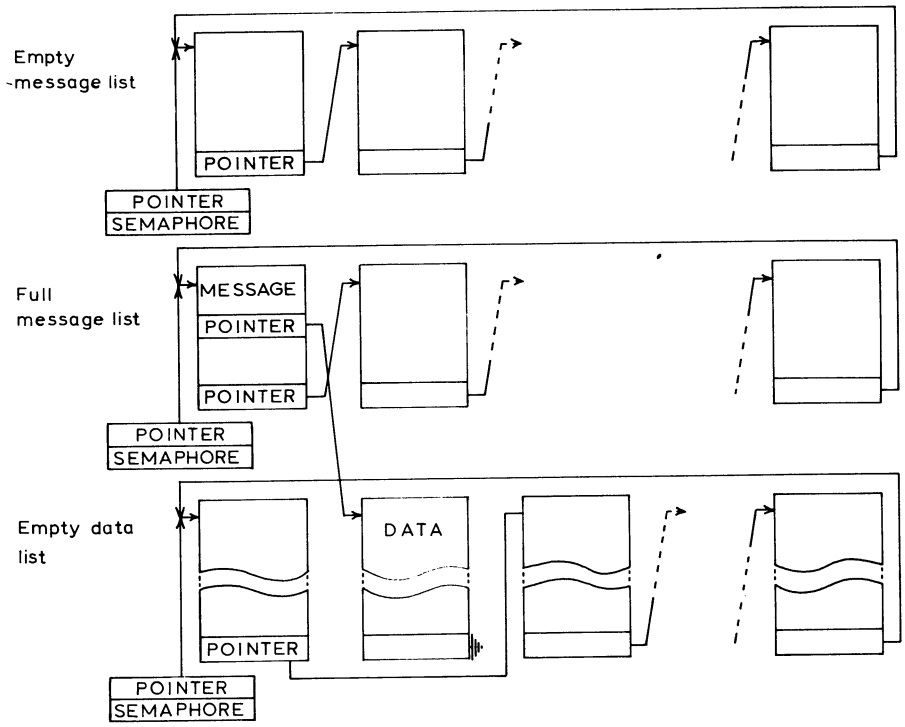

Fig. 2 Operating system message structure