AN INTRODUCTION TO

THE CR80 ADVANCED MULTIPROCESSOR OPERATING SYSTEM AMOS

791116/JHØ

1. INTRODUCTION TO CR80 AMOS

The Advanced Multiprocessor Operating System (AMOS) for the CR80 computers is a hierarchical structure of system components, where each component in the layer builds on the lower layers.

AMOS consists of the following components:

- an operating system kernel
- a memory management module
- a modular set of device drivers
- a file management system
- a unified input/output system

The modularity of AMOS makes it possible to build systems where not all components are used, where new modules are included or where certain components are replaced by customerized modules to fulfil very special requirements. In a small application with no need for backing store files the file management system need not be included. Not used device drivers can be left out and new device drivers are very easily included in the system.

2. AMOS Kernel

The Kernel is the lowest of the AMOS layers.

The Kernel is responsible for

- management of software-processes
- interprocess communication
- management of the CPUs
- the lowest level of device handling (interrupt handling).

2.1 Process Management

Process management is concerned with implementation of software processes as a data type and with implementation of the functions available for operating on processes.

A process may be defined as an incarnation of the data transformations obtained by execution of a program in a given context. A context is taken to mean a set of CPU registers (CPU resident or saved).

Processes are uniquely identified by symbolic names throughout a CR80 computer. A process may thus be referred to without a requirement to know in which memory locations it resides. The only instant at which such a knowledge need be specified is when a process is created.

The following functions are available for operating on processes:

- create process
- remove process
- start process
- stop process

2.2 Process Communication

Interprocess communication is based on exchange of messages and answers.

Two concurrent processes can cooperate by sending messages to each other. A message consists of a number of machine words. Messages are transmitted from one process to another by means of message buffers selected from a pool.

Efficiency is obtained by the queuing of buffers, which enables a sending process to continue immediately after delivery of a message or an answer regardless of whether the receiver is ready to process it or not.

2.3 Interrupt Handling

The Kernel controls the input/output interrupt facilities of the processor.

A process may reserve an interrupt and thereby establish monopoly of using that interrupt until it is released again.

On the arrival of an interrupt, hardware saves the executing process and loads the process the identification of which is fetched from an interrupt table.

If the interrupt is not waited for, a dummy interrupt process is loaded; it increments the occurrence count of the interrupt and reloads the former executing process.

If the interrupt is awaited by a process, the kernel interrupt process is loaded. It makes the waiting process enter the executing state.

2.4 CPU Management

CPU management is responsible for the allocation of CPUs to the preempted processes.

CPUs are handled by the Kernel as separately identifiable objects addressed by a symbolic name.

CPUs are grouped in pools. Each pool has its own ready list(s) of processes and is scheduled separately. When a process is created, it is determined which pool of CPUs it shall execute on.

The CPU allocation algorithm works independently for each CPU pool. The CPU allocation algorithm is invoked

- when a process calls a wait function to receive a not yet occurred event
- when a CPU has been allocated to a process a predefined amount of time (tunable).

3. Memory Management

The memory management module administers a shared pool of memory segments. Segments may dynamically be allocated to processes and deallocated after use.

4. Device Drivers

Each device connected to a CR80 system is controlled by a device driver.

The main purpose of a CR80 device driver is to bridge the gap between the hardware interface and the CR80 process concept.

Device drivers are implemented as CR80 processes establishing a generalized interface to the user processes fitting into the CR80 IO system. The generalized interface defines standard formats for driver operations and the command codes are unified where possible.

User processes communicate with drivers by using the send message and wait answer functions of the monitor. These functions may be used indirectly by calling the I/O system or directly from a user process (for instance to use a device in some specialized way).

Contrary to a user process, a driver process is authorized to perform IO operations.

A device driver has exclusive access to the corresponding device. Any use of a device must therefore be performed via the actual device driver which then is able to manage and control the access rights of the requesting user processes (granting mutual exclusion, etc. when necessary).

The device drivers check the hardware equipment, remedies errors if possible and reports serious failures.

A large number of different device drivers exist, interfacing a wide range of hardware equipment to the CR80:

- terminals
- large disks (12-300 MB)
- diskettes
- line printers
- card readers
- DMA channels
- magnetic tape transports
- communication lines for variety of protocols including X25 and BSC.

5. File Management System

The file management system offers a structuring of backing store into logical files and is responsible for storing, maintaining, and retrieving information on secondary storage devices (disks, diskettes, and magtapes).
The number and kind of devices(spindles, tapedrives) attached to the file management system is dynamically reconfigurable.

The file management system consists of a set of command processes each capable of performing one command at a time, communicating with the device drivers for disk, diskette or magtape by means of messages. The work is distributed among the command processes by a control process by means of messages, too. The control process, in turn, is activated by reception of messages sent to it by the I/O system on behalf of an application program. The interface between the I/O system and the control process corresponds to the interface between the I/O system and other driver processes.

5.1 Device and Volume Handling

Each device (disk controller, tape station, etc.) known to the file system is represented by a Device Control Block (DCB). A DCB contains the information which makes it possible for the file system to use the device. This information includes the process name of the corresponding driver process and the kind of device. When a volume is mounted on the device, its description is included in the DCB. Both devices and volumes may be referenced by symbolic names. The device control block is linked to all open files on the volume mounted on that device.

The file system may be given commands concerning:

- Management of peripheral devices. Devices may be assigned to and deassigned from the file system dynamically.

- Management of volumes. Volumes may be mounted on and dismounted from specific devices.

5.2 User Handling

Each process pair using the file management system is within the file system represented by a User Control Block (UCB). The name of the user is contained in the UCB. Several processes may be active under the same user name, thus having the same access rights. The UCB is linked to the file control blocks for files that are open to the user. The links show the access rights to that particular file.

There are commands to the file management system for creation and removal of user control blocks.

5.3 File Handling

Each open file is within the file system represented by a File Control Block (FCB). The FCB contains information which makes it possible to access the corresponding file. This information includes the address of the file on the volume, its size and its type. The FCB is linked to user control blocks for all users who operate on the file indicating at the same time their access rights to the file. The FCB is also linked to the device control block for the volume on which it resides.

symbolic name. Using that name it is possible to locate the file later on. The file may also be renamed or removed from the directory again.

- Change of access rights for a specific user (or the public) vis-a-vis a file. The right to change the access rights is itself delegatable.

- Transfer of data between files and buffers in the application programs. For transfer purposes a file is considered simply as a string of bytes. It is, therefore, a byte string that is transferred between a file and a buffer. The user can directly access any byte sequence in a file. The operations which are implemented by the file system are read, modify, and append.

5.4 Security

The protection of data entrusted to the file management system is handled at the file level.

The mechanism for access control is based on the use of Access Control Lists (ACL). There is an ACL connected to each file. The ACL is a table which describes the access rights of each individual user (one being the public) to the corresponding file. When ever a user tries to access a file, the ACL is used to verify that he is indeed allowed to perform this access.

6. Input/Output System

The AMOS I/O system provides the application programs with a unified interface to peripheral devices including disk volumes.

The I/O system may either be used to call directly to peripheral device driver process or it may call the DAMOS file management system. In the former case the device is treated as a single file in itself, in the latter case the file management system will structure the device into logical files organized in a hierarchical directory structure.

The AMOS input/output system is a set of procedures which may be called from all application programs. The input/output system code is shared by all programs and invoked by MON-instructions. However, no context switch takes place. The input/output system checks the validity of parameters passed to it, and the legality of a requested operation. If the check is satisfied a message is prepared for the appropriate device driver and sent to it. User code and classification information are specified to the driver, enabling it to perform any relevant authentication.

The commands to the I/O system may be divided into four groups:

- Environment Control
- Direct Input/Output
- Sequential Input/Output
- Input/Output Utilities.

The first group which to a certain extent is device dependent covers such operations as:
create file, assign terminal, etc.

The direct I/O operations directly reflect the data transfer commands of peripheral device drivers and of the file management system. They may be performed either with or without suspension of the calling process. It is the responsibility of the user to allocate and specify to the I/O system the necessary buffers. The buffers used for one command must be specified to the I/O system by a list of buffer references. Each of these references may specify either a local buffer or an external buffer requested from the buffer manager. Local and external buffers may be mixed freely in a buffer list.

The sequential I/O procedures are implemented by the I/O system on top of the direct I/O procedures. For the purpose of sequential I/O, the I/O system will allocate the necessary buffers for implementation of a read-ahead/write-behind strategy. The I/O system copies data between the user area and the buffers. Single bytes or records of any length may be transferred. Blocking/Deblocking of records are handled by the I/O system.

The input/output utility procedures are built on top of the sequential I/O procedures. These procedures include formatted input/output of numbers, identifiers, text strings, etc.

6.1 Input/Output Via the File Management System

The device, volume, user, and file manipulation commands of the file mangement system are directly available through the I/O system. The necessary data structures are allocated and deallocated when files are opened and closed, but apart from that parameters are just passed on to the file management system.

6.2 Input/Output to Peripheral Drivers

All commands are passed on to the driver process but some file related commands may turn out to be irrelevant and therefore ignored by the driver. By the I/O system terminals are handled in the same way as files. However, for sequential input only one buffer is used in stead of the usual double buffer.