68020 coprocessors

Using tree-structured memory management tables saves large amounts of memory

DAVID BURNS and DAVID JONES

In paged memory-management systems a description of how a logical address is transferred into a physical address needs to be stored in memory. This description is usually placed in a descriptor table containing information about where the physical address is in memory and other information pertinent to the requirements of a modern virtual-memory operating system.

Linear tables, which contain a single contiguous table with an individual entry for each logical page, are by far the simplest. For logical to physical translations the logical address is simply used to index a large linear table containing the physical address.

With linear tables however, a large portion of the table must reside in memory. Pages as small as 256 bytes are sometimes used; a 68020 system with this type of management and 4 Gbyte addressing would need many megabytes of memory to accommodate several million pages.

An alternative is to translate addresses in a tree structure, Fig.1. With this method only a portion of the logical address space is mapped at each level. At the highest level of the tree, logical address space is subdivided into relatively large blocks; each stage of the tree further subdivides the address space until the page address emerges at the bottom. The M68851 manages memory in this way, accommodating pages as small as 256 bytes.

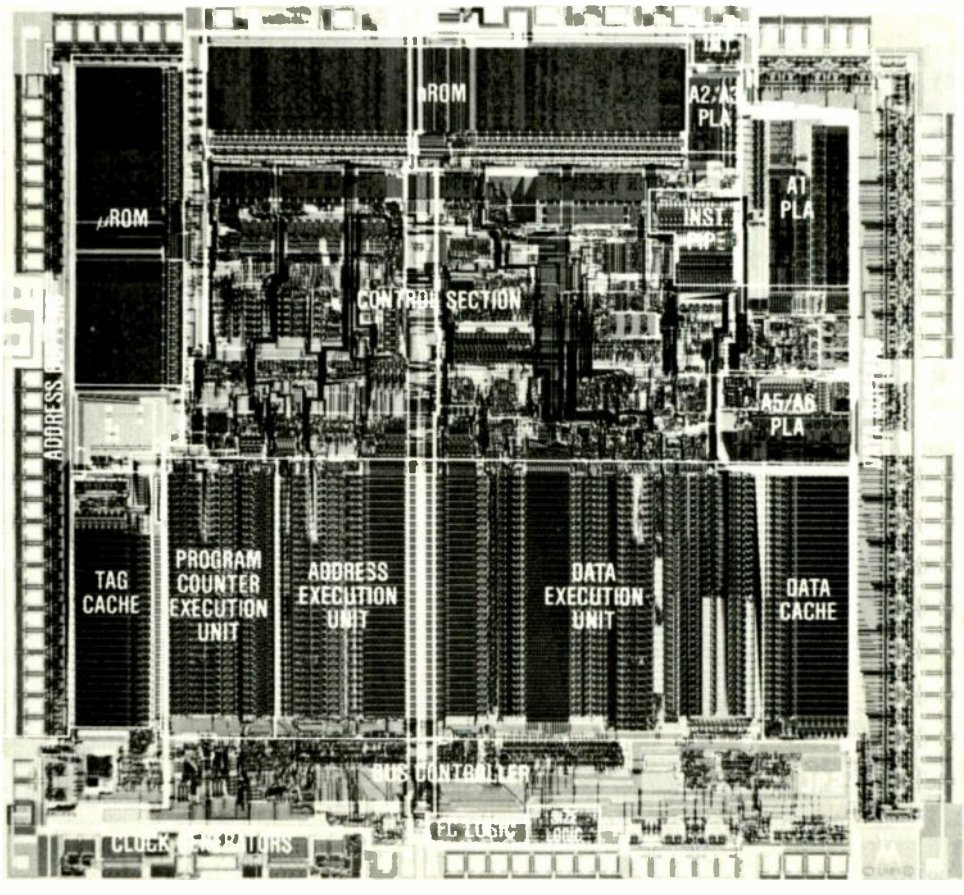
Tree structures allow large portions of logical address space to be allocated to a single entry at a higher level of the tree. In addition, portions of the tree itself may reside on secondary-storage devices, or may not exist at all, until they are required by the processor. As a result, more system memory is available to the user.

Being more complex than linear tables, tree structure translation tables may require several table searches to locate a translation descriptor. In the 68851, a fully-associative address-translation cache with 64 entries and an efficient bus overcome this disadvantage (the translation cache has a hit rate of less than 95%). When a physical map is found, an entry is made in the translation cache. On each subsequent access of this physical page the translation cache responds, making external table searching unnecessary.

TREE STRUCTURE TABLES

Translation tables in the tree structure are indexed from a particular section of the logical address input. How the logical address is divided depends on the contents of the translation-control register.

Within the control register the size and level of the table tree can be defined. This register also defines initial shift, page size, supervisor translations and function-code look-up. Figure 2 shows how these indices are used as index pointers to different levels of table within the structure.



Two types of tables exist within this process – descriptor tables and page tables. Descriptor tables indicate addresses of the next level of tables beneath the present one and page tables point to the base address of the translated page. During normal address translation, the memory-management unit steps down through several levels of descriptor tables before it reaches the page-table reference at the bottom of the tree. At this level the page pointer can also be used as an indirect pointer to the final page address so there is the potential for five levels of table within the translation process.

Table entries can be in either long or short form. Table descriptors contain the base address of the next table (descriptor or page) and status information relating to write protection, user and descriptor type. Long-form descriptors have a limit field to restrict the size of a particular table and additional information to accommodate access levels. Page descriptors hold the base address of the physical page in memory, i.e. the translated logical address. In Fig.3, the complete translation process using all four levels of tables is shown.

Table structures of this type increase flexibility. By using different translation paths down through the tree structure it is possible to end up at the same physical address translation. This feature can be used in multi-tasking to give different tasks different access rights to a particular physical page. By mapping different tasks down through different descriptor tables it is possible to assign read/write protection to the lower levels of tables. One task may be allowed to read and write to the page whereas the other task may only be allowed to read from it simply by manipulating the write-protect bit within the relevant table descriptor.

68851 PROGRAMMERS' MODEL

Being designed as a coprocessor for the 68020, the 68851 follows the concepts of the floating-point arithmetic processors described last month. These coprocessors work

Table 1. In 68020 assembly language, memory-management instructions for the 68857 are preceded by P.

Mnemonic	Operation
PMOVE	Move data to or from the 68851 registers using the addressing modes of the processor.
PVALID	Test access level permission of a calling module.
PTEST	Search a table using specified address and function code values. Results of the search are recorded in the PSR. Optionally, the physical address of the last descriptor fetched may be returned.
PLOAD	Performs a search as PTEST but loads an entry into ATC.
PFLUST/PFLUSHS	Invalidate an entry in the ATC; PFLUSHS invalidates only share global entry in ATC.
PFLUSHR	Invalidate an entry in root-pointer table.
PFLUSHA	Invalidates all ATC entries.
PBcc, PDBcc, PSCC, PTRAPcc	These instructions operate in the same manner as the 68020 conditional instructions except they operate on the 68851 status register.
PSAVE	Save entire machine state information on the supervisor stack.
PRESTORE	Restore all machine-state information in the 68851 register.

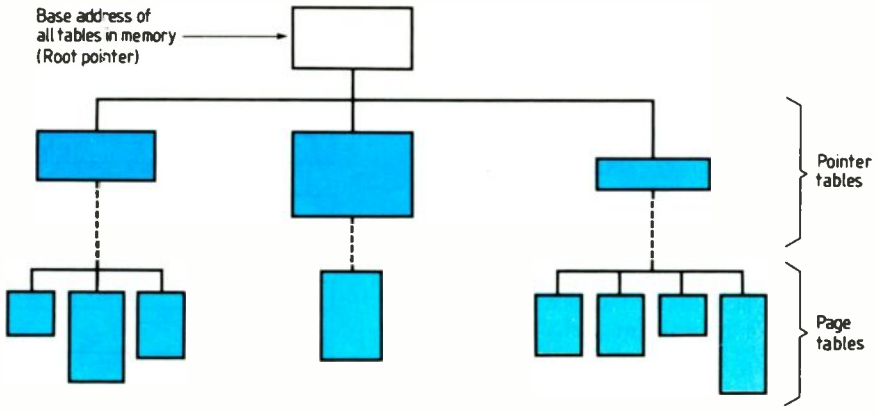


Fig.1. Translation mechanism of the 68851 uses a tree structure to find physical mapping of the logical address. Method allows only a portion of the current tree structure to be resident in memory whereas the linear approach requires the whole contiguous table to be stored in memory.

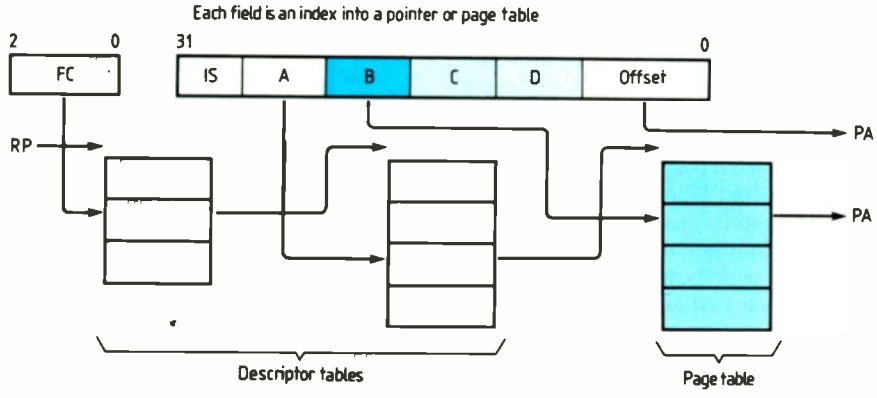


Fig.2. Translation control register, TC, is the main element in the 68851 and controls how all translations in memory are performed, table size and the page size.

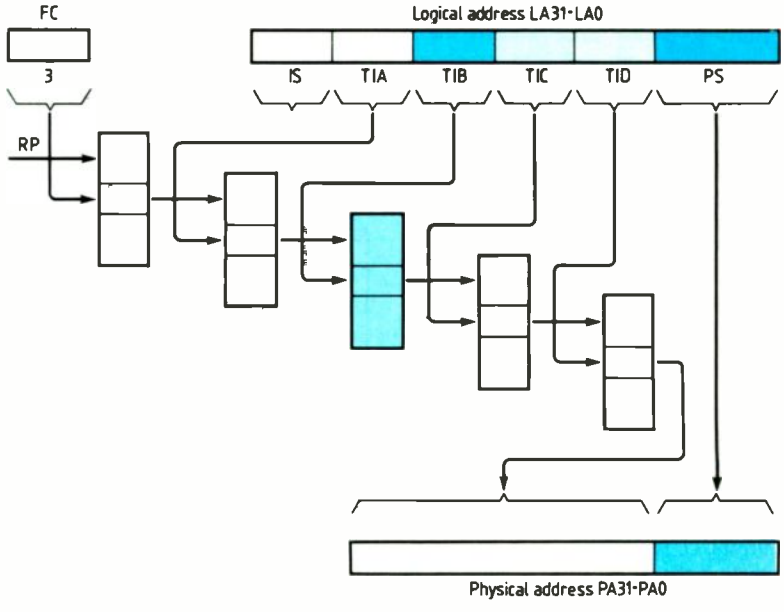


Fig.3. Logical addresses are divided into a number of blocks. At each level there is a table with a reference to another table finally ending in a page descriptor. Page-descriptor information is used by the 68851 to produce the final physical address. This information is loaded into the address translation cache.

so closely with the main processor that the programmer need not be aware that extra hardware has been added; all that the programmer needs to know is that there are extra instructions and registers available. Figure 4 shows the 68851 memory-management unit programmers' model and Table 1 shows the instructions for programming these registers.

Within the 68851 are three root pointers each containing the base address of a set of table and page descriptors for a task, the descriptor type at the next level, a limit field and a bit for implementing shared global memory.

Most translations are performed through the c.p.u. root pointer and each time a new task is executed a new value is written into the root pointer. A record of the eight most recently used root pointers is kept in an on-chip root-pointer table.

The root-pointer table exists so that if a context switch is made, i.e. if the time-slice given to that particular task expires, a new root-pointer value is simply written to the c.p.u. root-pointer register. If this new root pointer also appears in the root pointer table, the task has already run and it is likely that the translation cache will contain entries for it. This being the case the 68851 may not need to perform many, if any, table searches to fetch the pages for this task into the address-translation cache.

If the supervisor root-pointer enable bit is set in the translation control register then all supervisor logical address accesses are translated via a separate root pointer called the supervisor root pointer. This adds flexibility and increases system integrity.

Since the 68851 accommodates both logical and physical bus arbitration the direct memory access root pointer is used to allow translations for other logical bus masters; a disc controller for example can have its own translation mechanism.

How translations occur is controlled by the translation control register. It enables translations to occur once all the table entries have been entered into processor memory or other storage devices. In addition the translation control register can enable a function-code to be taken from the first table pointed to by the root pointer.

Remaining parts of the control register program page size, initial shift and up to four table index fields. How many upper logical-address bits are to be ignored by the 68851 during table search operations is determined by the initial shift field. During a table search operation, the index fields specify the number of bits of the logical address to be used as an index to the translation-table level.

Status information for the translation cache is held in the memory management unit's cache status register, PCSR. This information helps the operating system to maintain the computer system by for example allowing the translation cache to be flushed when the F bit is set. Information in a second status register, PCR, is used by the operating system to determine the cause of faults like bus errors and access-level violation. To simplify fault correction, this status also contains information about the table or page descriptor at the time that the fault occurred.

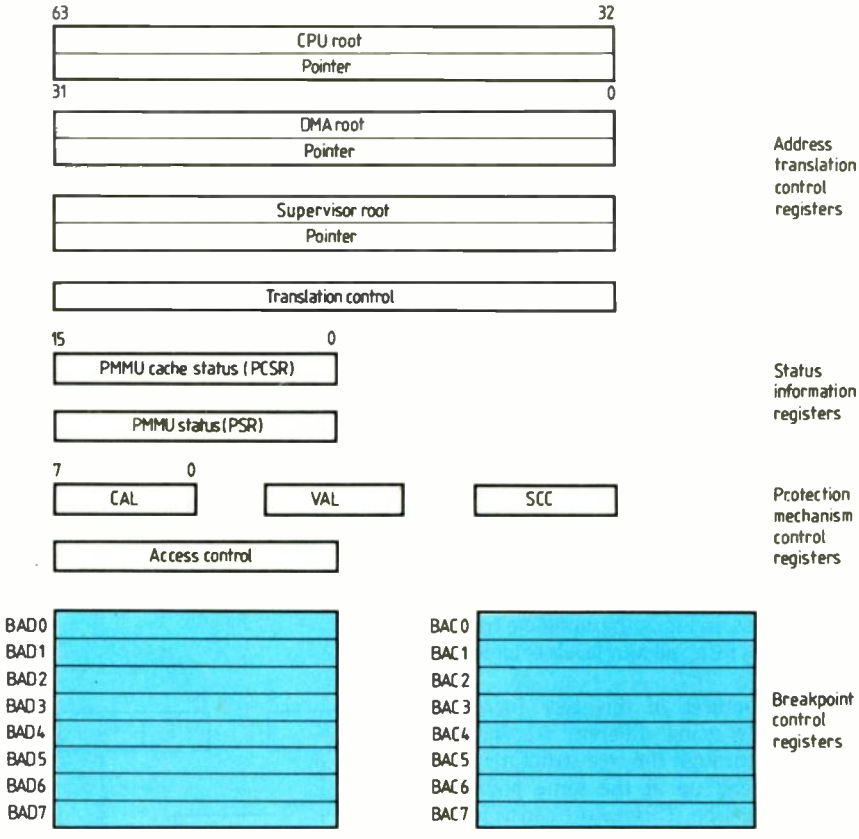


Fig.4. As a 68020 coprocessor, the 68851's internal registers can be considered as extensions to those of the 68020. As far as the user is concerned, executing instructions for the 68851 is no different to executing 68020 instructions.

Four protection registers, CAL, VAL, SCC and ACCESS CONTROL are used in conjunction with the 68020 module support facility described in the February issue to implement a hardware and software protection scheme for the memory system.

Registers BAD_{0-7} and BAC_{0-7} are used in conjunction with the 68020 breakpoint instruction BKPT #<data>. Breakpoint-acknowledge data registers BAD_{0-7} hold opcodes displaced by the breakpoint instruction and breakpoint-acknowledge control registers BAC_{0-7} contain enable and count functions for the breakpoint-acknowledge mechanism.

Communication between the 68020 and coprocessor is not apparent to the programmer. All that the programmer needs to know is that there are extra registers and instructions to make the system more flexible. Adding a coprocessor to the 68020 is quite straightforward and once it is in position it can be used as much or as little as is necessary.

David Burns and David Jones are applications engineers at Motorola's East Kilbride plant.

Why have memory management?

Memory management is needed in most large systems today for several reasons.

- **Multi-tasking operating systems require a degree of inter-task protection. This is to prevent the situation whereby one task may accidentally try and overwrite another task's data area. The provision of a memory management unit will confine a task to permitted areas of memory.**

- **In a virtual memory system of the address space is contained on disc and only the areas that are currently being used are actually resident in main memory. A memory management unit allows the operating system to keep track of memory areas (pages) which are active and those which are on disc.**