A DESIGN FOR A MULTIPLE USER MULTIPROCESSING SYSTEM

James D. McCullough
Kermith H. Speierman
and
Frank W. Zurcher

Burroughs Corporation
Paoli, Pennsylvania

INTRODUCTION

The B8500 system is designed to deal with the following situation. A large number of active programs requiring various services are present in the system and their current status and required service are recorded. When some component of the system becomes available, e.g., processor, memory space, peripheral device, it is assigned to the active job of highest priority that requires this service. The important concept is that no component of the system belongs to any program but rather provides a service and then goes on to service another program. The main function of the executive scheduling program is to keep track of the services required by programs and to schedule the services when equipment becomes available.

This mode of operation requires that the system have the following properties:

1. The equipment should consist of independent modules that can function concurrently; e.g., processors, memories, I/O, etc.
2. A bulk memory system that is a logical extension of main memory.
3. Segmentation of data and programs to make more effective use of memory and permit a large number of active programs to be present in the system.
4. Dynamic allocation of memory.
5. Memory Protection to prevent interference between programs.
6. An Executive Scheduling Program (ESP) that controls and schedules the entire system.

The Burroughs B8500 is a modular processing system designed for a multiprocessing and multiprogramming mode of operation. In addition to the concept of multiple central processors, the B8500 also functions with multiple input/output processors which operate nearly independently of the central processors. A high-speed fast-access disk storage unit is provided as an extension to the main memory and is used concurrently by the input/output modules for storing input from external communications and retrieving required programs and data for the central processor.
STRUCTURE OF PROGRAMS

Program segmentation is a basic requirement of a multiple user system to provide effective use of memory by permitting a large number of active programs to be present in the system. A B8500 program may be considered as the output of one compilation consisting of program segments, data segments, an operand stack segment, a working storage segment, and a program reference table (PRT). The minimum structure that a program must retain in main memory to remain active is one program segment, an operand stack, a working storage segment, and a program reference table. A large program may also require additional program and data segments at various periods of program execution. The allocation of required segments is provided at the time the segments are referenced through descriptors in the program reference table. The descriptors define the segments as they appear on disk storage and when the segments are allocated in main memory the descriptors provide communication between the separately allocated segments of the program.

Tag Bits

A memory word contains 52 bits, 48 data bits and 4 tag bits that may only be modified in a protected control mode. In addition to a parity bit, three tag bits are provided to construct and control memory words used as descriptors. One of these tag bits is a presence bit that is used by ESP to define the presence in main memory of the segment that is represented by the descriptor. A reference through a descriptor to a segment not yet allocated in main memory causes a presence bit interrupt and invokes ESP control of allocation of the required segment. The two remaining tag bits are encoded as program descriptor, data descriptor and indirect memory reference. A typical program is shown in Fig. 1.

Program Segments

Program Segments contain instructions and constants and may have a maximum length of 4096 words. Programs are location-independent and all internal addressing of constants and jump instructions is relative to the Base Program Register (BPR) which contains the absolute address of the segment base. Jump instructions may reference any syllable within a word. Syllables are six bits long and instructions contain from one to four syllables. The Program Counter Register (PCR) is a 15-bit register relative to the BPR: 12 bits are required to address the relative word in the segment and 3 bits to address one of the 7 syllables contained in the word. The PCR was designed to enhance dynamic memory reallocation and allows the ESP to move a program segment which has been partially executed, simply by changing BPR to the new base location of the segment.

Program segments can only be read and are protected from accidental modification since they are allocated outside the area bounded by the memory bounds registers. Since program segments are referenced by program descriptors which have the appropriate tag bit configuration, they may never be accidentally read as data segments. Program segments may contain internal subroutines which are referenced via a program descriptor in the PRT. While individual program segments are restricted to the 4096-word limit, a large program may contain many program segments. Transfers between segments are directed by program descriptors in the PRT.

Program Reference Table

The program reference table is a read-only segment and contains descriptors for program communication with data segments and other program segments. Descriptors are addressed relative to the PRT base register and addressing beyond the limits of the table is prohibited. A program is filed on the disk with its PRT and program and data segments.

The filed PRT contains the information required by ESP to construct the descriptors which must be present in the PRT when the program is placed in active status. For each descriptor, this information includes the name of the object to which it refers, the type of object (procedure, simple subroutine, data, etc.), the mode of use (global, own, read-only, etc.) and the descriptor required for its input from the disk. When the PRT is input to memory, ESP decodes this information and sets the necessary tag bit configuration required for processor recognition. One tag bit configuration is set for descriptors which refer to program segments, procedures, and subroutines and another configuration is set for
Figure 1. Typical Program Structure.
those descriptors which reference data sets or memory space.

Words zero and one relative to the PRT base register are reserved for special use on a procedure call. A procedure is a program which requires its own PRT, and is provided to permit calls on programs which have been compiled separately from the calling program. A procedure call is executed by a reference to a program descriptor in the caller's PRT. This program descriptor contains the address of the procedure's PRT instead of the program segment of the procedure. Word zero relative to the new PRT is used to save the contents of the caller's PRT base register such that it may be restored to its correct value when the procedure returns to the caller. Word one relative to the new PRT contains the BPR value of the procedure's initial program segment.

Data Segments

Data segments are addressed relative to data descriptors which contain the absolute addresses of the segments. The tag bits of the descriptor determine the memory bounds. The next instruction which executes a memory fetch or store is compared with these memory bounds, providing both read and write memory protection. Any reference to a data descriptor relative to the PRT base register causes that descriptor to be placed in the processor's local high-speed memory such that subsequent references to that descriptor will not require a main memory fetch if it is among the last 16 descriptors referred to.

Working Memory Segment

The working memory area comprises two logical segments, common, and the subroutine control stack, allocated in a contiguous memory block and bounded by the processor memory bounds registers. The common, or global data area, is addressed relative to the Base Data Register (BDR). It should be noted that we have not provided any direct method of setting or saving the BDR on subroutine jumps or procedure calls because of its intended use for common data.

The subroutine control stack provides dynamically allocated space for subroutines and procedures and is used to contain local variables and index words and for passing parameters between procedures and subroutines. Addressing in the control stack area is relative to the Base Index Register (BXR). When a subroutine or procedure is called, the BXR is incremented beyond the calling program's control stack area; parameters and return information are stored relative to the new BXR value; and the called program addresses relative to the new BXR. Word zero relative to BXR is used to save the relative BPR, PCR, BXR increment, and jump control bits of the calling program. The subroutine return instruction refers to this location for its information. This structure provides an automatic mechanism for subroutine nesting and recursion. Any of the 4096 directly addressable words relative to BXR may be used as index words; the most recently used are kept in the processor's local high speed memory.

Operand Stack Segment

The operand stack segment is used by the processor to hold operands and results for the arithmetic and logical instructions. Programs for expression evaluation are Polish strings which are directly executed by the arithmetic unit using a push down stack implemented in the processors hardware. The stack segment discussed here is in main memory and is a logical extension of the processor's stack.

Memory Protection

Memory space is allocated by the ESP and given to the user program by setting bounds registers in the processor and descriptors in the PRT. These registers and descriptors can only be set by ESP (in the control mode) which prohibits the user from having any control over the assignment of memory.

The working segment and operand stack segment are read-write areas and each are defined by memory bounds registers. The program segments are read-only objects and are not contained within the limits of bounds registers. The PRT, which is a directory of all program and data segments used by a program is a read-only object and is contained within the limits of bounds registers which prevent using any descriptors that are not in this PRT. If a user tries to change his PRT he will be interrupted and ESP given control. Data segments are referred to by data descriptors in the PRT. Each time such a data reference is made the descriptor sets up bounds around the data segment being referenced for the
duration of the data reference. Any attempt by the user to read or write any other areas of memory will cause an interrupt and entry to ESP.

It is possible to branch outside of a program segment without detection but the program is still restricted to its own data and working storage segments so it can’t effect another user by accidentally branching to his program. I/O operations are controlled by the ESP to prevent a user from interfering with another’s space. This combination of ESP and hardware conventions allows any number of user programs to be executed together in a multiprocessing-multiprogramming mode without the danger of accidental interference.

THE EXECUTIVE SCHEDULING PROGRAM

The Executive Scheduling Program (ESP) schedules both hardware and software services for all programming tasks or jobs that are present in the system. Many of the services of the ESP are themselves programs that are structured as normal user programs and therefore may also be scheduled in the normal manner. The intent is an organization of the ESP consisting of many subprograms which are separately constructed and therefore may be executed concurrently. Each of these subprograms is extended system privileges according to the functions it is to perform, e.g., initiate I/O, manipulate tag bits, etc. The ultimate requirement of the ESP is to efficiently schedule all services, both hardware and software, to effectively establish maximum throughput of the system.

Scheduling

Jobs may be introduced to the system from various sources. Prestored production tasks are entered by the ESP without any external request; requests may be entered from external remote stations; input streams from peripheral devices are interpreted for batched requests; or a program or job may request the execution of another job during execution. All jobs presented to the system are retained on bulk storage and descriptions of these candidates for scheduling are kept in a Cold Job Table which is also kept on bulk storage. A Cold Job entry remains in the system from the time it is introduced for execution until its final outputs are delivered. Each Cold Job entry contains information required by the scheduling program to efficiently introduce tasks to the system. Information required in each Cold Job entry includes class and priority of the task, estimated processor time, memory requirements, input files required, dependence upon other jobs and accounting information.

Prior to a job’s introduction to the scheduling program, the availability of program and data files must be established. This fundamental requirement is established to insure that once a program is entered for execution, its completion will not be delayed by the unavailability of a program or data file when required. Therefore, prior to successful scheduling, a collection program is invoked to accumulate the job’s external files on bulk storage and present to the system the required information concerning those files.

When a job is acceptable for execution, the scheduling program generates a Hot Job Table entry in main memory and requests the allocation and readying of the program’s initial requirements. Initial requirements for all programs are the program reference table, the working storage area, the operand stack, and at least one initial program segment. Other required program and data segments are allocated and readied when they are first referenced through the descriptor which describes them. These Hot Job Table entries establish a path of control which the processor is to execute, and contain the processor state information (hardware register values) recorded at the program’s last suspension of execution. The entries also contain the program status and are linked in priority order. The program status may be ready to execute, awaiting I/O, awaiting memory, being executed, awaiting time, or being terminated, and is used by the internal scheduling program for determining the next useful task to assign to the processor.

Classes and Priorities

Class is defined as a mode of operation; e.g., real time, batch, conversational, etc. Priority is defined as relative importance within a class. Classes possess a relative priority to each other. It is a desirable feature that the system be self-regulating to prevent a group of users in one mode from monopolizing the system’s resources. The philosophy employed is to get done what must be done in a timely fashion, but always to maintain a lower limit of re-
sources below which the sum of users in a class cannot fall. A good example is the conversational mode in which remote terminal users may experience a decrease in system response time but will not experience a complete blackout due to higher priority requirements.

**Memory Allocation**

Memory allocation is controlled entirely by ESP since no hardware directed technique is attempted. The memory allocation program is responsible for the maintenance of all main memory. Its basic functions include obtaining a block of available space to satisfy a request and to assume responsibility for space being relinquished by its prior owner. Allocation performs its function through the mechanism of linked tables which include all blocks of memory. All blocks of memory, whether available or assigned, are linked by address in a memory map.

All blocks of memory which are available are linked by size in available space map. An attempt to allocate space for a caller is governed by the priority and class of the caller and the amount of space which has previously been committed to callers of that class.

The allocation routine will first try to allocate by scanning the available space map to find the smaller block which is large enough. If a block of sufficient size cannot be found, the overlay program is called.

The function of the overlay program is to find a currently committed block of memory which can be reassigned to the caller of allocation. Using the priority and class of the requestor, the overlay program will scan the Memory Map for a block of memory to be reassigned. The considerations to be applied at each block will be:

1. Does the block belong to a running program?
2. Is I/O going on in the block?
3. Size of block and surrounding blocks.
4. Is the block program or data?
5. Priority and class of block.
6. The number of disk operations required.
7. Number of users.
8. Size required.

If a program segment is overlaid, the appropriate program reference tables are updated to cause interrupt on access by the callers of the segment. If a data segment is overlaid, the data segment is saved in bulk storage before the space is reassigned. User PRT segments and stored register values are updated appropriately.

In the event that a request for space cannot be granted by these means, a deferred space request will be put into an unallocated request chain. This chain will be scanned periodically to allocate the deferred request.

In the event that the system finds itself bound by having too many things to do, and not enough space to do it in, it will (based on priority, class and percent completion) choose a job which has been introduced into the system and terminate processing on that job. The Cold Job Table will be restored to a previous state so that the job can be rescheduled at a later time.

The PRT contains the descriptors through which programs address separately allocated segments without knowledge or regard to the absolute memory allocation. Location independent addressing allows ESP to dynamically change the contents of memory by releasing segments not currently in use and replacing them with other required segments. The presence bit of a descriptor is used to indicate the presence of a segment in main memory and a reference to a descriptor representing an absent segment causes a processor interrupt invoking ESP. ESP must then read the required segment referenced by that descriptor. The descriptor is interpreted for type and when a global type is required, the memory map is scanned to attempt to locate the desired segment in active memory.

If the segment is not in memory or a local copy had been requested, the memory requirements and disk address for the segment are available from the PRT, and ESP places the requesting program in a suspended status, initiates an input request for the segment, and assigns the processor to some other useful task. When the requested input is completed, the descriptor(s) which addresses it are marked present and those programs may then be scheduled for the processor.

REFERENCES

1. "Burroughs B5500 Information Processing"

