HITAC 8700/8800 Operating System (OS7)

Isao Ohnishi, Hidehiko Akita, Shingi Domen and Kazumasa Kaneko

1. Introduction

OS7 is a general-purpose commercial base operating system incorporating multiple virtual spaces and multiprocessing modes up to 4 CPUs.

OS7 enables each processor of the system to operate in a symmetric mode or jointly with the other models in a non-symmetric mode.

It is designed to support closed batch, open batch, remote batch, time sharing (interactive) and real time processing simultaneously under the single installed operating system.

The same command languages (JCL) are used commonly to describe jobs among the 5 types of processings.

The file subsystem such as the structure, catalogue designs, and file access methods are identical and common among these processings, thus the user at the TSS terminal can manipulate data created in a batch processing.

2. Main features

2.1 Virtual memory

2.1.1 Virtual space organization

Virtual spaces in OS7 has the organization shown in Fig. 1.

Address 0~M-1 is the system space and is used commonly by all users, while M~2^{31}-1 is the user space and is given in each user job.

Each user has the same beginning address M in the virtual, in spite of the different address in the real space (multi-virtual space).

Therefore the address translation table (shown in Fig. 2) is given for each user, and the user space never be overlapped with the others.

The multiplicity of the job is infinite in design and the available address space for each user is 2^{31} bytes.

Besides, a ring memory protection mechanism is implemented in hardware on which all segments are classified into 7 rings and the program in the lower ring area cannot write the data on the upper ring.

Rings 0~3 are used by the control program, ring 4 is for compilers and utility programs, and ring 5~6 are assigned to user programs which are available for making hierarchial codings.

The ring mechanism has made it easy to locate the illogical system bugs during the development phase of OS itself.

This paper first appeared in Japanese in Joho—Shori (Journal of the Information Processing Society of Japan), Vol. 14, No. 10 (1973), pp. 769~777.

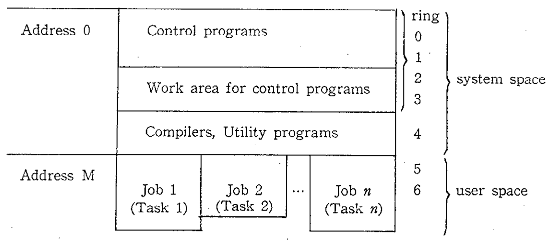

Fig. 1 Virtual space organization in OS 7

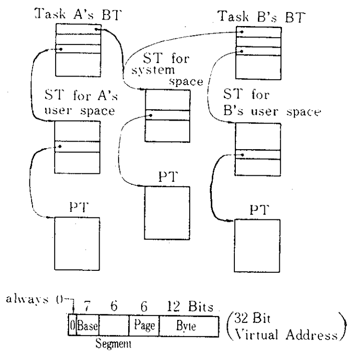

Fig. 2 Address translation tables and virtual address format

2.1.2 Memory scheduling implementation

In OS7 memory requirements are generally notified by mapping fault interruptions.

If there is enough unused main storage then the page issuing the mapping fault is allocated immediately, otherwise a proper page in the main storage has to be swapped out on to the auxiliarly memory to give the main memory to the page.

On the memory scheduling the "algorithm to select pages to be swapped out" is the most fundamental policy to increase performance and reduce response time.

The purpose of finding the optimal replacement algorithm is supposed to minimize the number of page faults.

In general the best "unrealizable" algorithm is "to select and swap out pages to be referred to in the most far future".

However it is impossible only to predict which is referred to in the nearest future.

The Least Recently Used (LRU) algorithm, in which the page referred to the least recently is swapped out is based on the assumption that the page referred to in the least recently will not be referred to in the nearest future.

The strict LRU rule is also very difficult to be implemented, so that the scheduling algorithm of OS7 is classified as the First In Not Used First Out (FINUFO) policy in which the unused page not referred to in the interval is swapped out first.

The FINUFO policy is realized as follows:

(i) Each page on the main storage has R (Refer) bit and C (change) bit.
Initially R and C are zeros.
If the page is referred to then R=1, If the page is changed then C = 1 (in hardware).

(ii) There is a pointer to the pages which may be swapped out (ring list).

(iii) If it becomes neccessary to swap out a page, R bit of the pointed page is tested.
If R = 0, the page is swapped out. If R = 1, then after R is changed to zero the pointer is updated to the next page.

(iv) After a page is selected to be swapped out, its C bit is examined.
If C = 0, and the same image is on the drum, then the page is skipped and not to be copied on the drum. Otherwise it is always copied.

The selection of a ring list is applied in OS7 as follows (local policy):

(1) Either the 'system space' or the 'user space' is selected depending on the less of the number of the mapping fault/page in seconds.

(2) If the 'user space' is selected, then a user job is found. The user job should be swapped out based on the following criterion:-

Job's current memory size should be within the user declared space limit, which is defined for each job class at system generation.

When the multiplicity of tasks is too heavy, because of insufficient stabilizing mechanism in the attached job function, the number of swapping-in/out may increase excessively (thrashing).

When the phenomenon of the thrashing is detected by the probe, the control program makes a certain task inactive (roll-out).

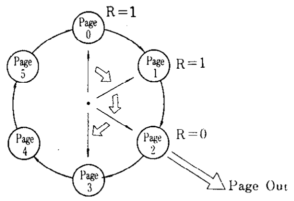

Fig. 3 First In Not Used First Out (FINUFO) Algorithm

2.2 Multiprocessing

2.2.1 Locks

In a multiprocessing environment, the same code of control program is concurrently executed by several CPUs.

To prevent one CPU from referring or updating control information while other CPU is updating it, the synchronization among CPUs is necessary.

The simplest method is to introduce a single lock which serializes the execution of the kernel part of control program in which external interruption or mapping fault is inhibited.

But a single lock may become a performance bottleneck when heavy services are necessary.

In OS7, in order to minimize the loss time due to waiting for a lock to be released, more than 50 locks are allocated.

To prevent deadlocks, all the resources are designed to be locked in the same sequence.

2.2.2 Task Scheduling

OS7 has two types of task scheduling algorithms, one of them is selected at system generation time. This is simple priority scheduling algorithm based on task priority. The other is the algorithm based on task types as well as task priority.

There is one ready task queue for each task type - system task, real time task, TSS task and batch task. The scheduling priority of each task type is normally in the order described above - system task has the highest priority and batch task has the lowest priority. Among tasks with the same task type, each task's priority is used for scheduling.

In this algorithm, task type priorities for TSS task and batch task are determined by the CPU serviced - time ratio. When TSS tasks have spent more CPU time than the ratio which is specified at system generation time, batch tasks will have the higher priority than TSS tasks.

In both scheduling algorithms, time-slicing feature is available. For n CPU, there is a set of ready task queues and n tasks with the highest priority are selected and executed. In the multiprocessing environment, one CPU informs other CPUs by means of direct control in the following procedure:-

(i) When a task becomes ready in one CPU, there is some other CPU which is in the idle state.

(ii) When a task becomes ready in one CPU, there is some other CPU executing a task whose priority is lower than that of the new ready task.

The CPU, which has received the direct control signal, will switch to an appropriate task.

2.2.3 Multiprocessing with different models of processors

Multiprocessor configuration containing both H-8800 and H-8700 is supported.

(i) H-8800 is much more powerful than H-8700 on arithmetic operations rather than logical operations, memory to memory operations and I/O operations. Therefore, in the 8700-8800 multiprocessing system, task scheduler is designed to execute user tasks preferentially on H-8800 and system tasks on H-8700.

(ii) H-8800 does not become idle when H-8700 is executing a task. When H-8800 is in idle, H-8800 takes over the task which H-8700 is executing.

(iii) It is possible to make all I/O operations to be performed only by H-8700.

(iv) For a job which must be processed with the shortest turn-around time, one can utilize an express task which has the highest priority and is processed only by H-8800.

2.2.4 Performance of Multiprocessing

Performance of multiprocessor system, as compared with single processor system, is affected by the following factors.

(i) In multiprocessor system, system resources can be utilized more efficiently. This improves the total throughput of the system.

(ii) Effect of sharing programs becomes prominent. Only one copy of control programs and other systems programs have to reside in memory. One can save main memory space and reduce swapping overhead.

(iii) Average main memory access time becomes longer due to the conflict of main memory access by CPUs.

(iv) There is some delay due to the locks of serially reusable resources.

The magnitude of the effect of these factors varies depending on the characteristics of user programs and environments.
Table 1 shows the performance of multi-processor system measured on FORTRAN (compile, link and go) bench-mark jobs in October 1973.

Table 1 Throughput of multi-processing on OS 7
(Measured on 20 FORTRAN programs)

No.	Configuration	Thrupt Ratio	CPU Time Ratio
1	8,700×1, 1 MB Memory	1 (0.8)	1
2	8,700×1, 2 MB Memory	1.2 (1)	1
3	8,700×2, 2 MB Memory	2.3 (1.84)	1.1

2.3 PROGRAM SHARING

2.3.1 Reentrant program

A program must be reentrant if it is to be shared. In OS7 user can write a reentrant program easily by using PSECT (Private Section) and COPY macro in addition to hardware stack features. User can define constants and work areas in PSECT and get a copy of PSECT for each task, issuing COPY macro. Compliers (FORTRAN, COBOL and PL/I) can produce reentrant object codes, if so specified.

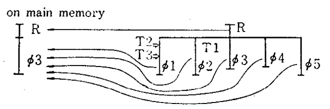

Fig. 4 Overlay structure in real memory system

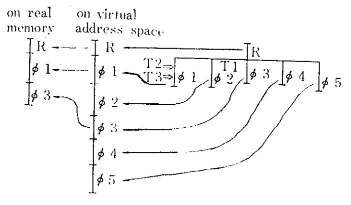

Fig. 5 Logical overlay structure in virtual memory system

2.3.2 Logical overlay structure

Very large programs such as compilers have had overlay structure in conventional real memory systems. In virtual memory system, one can execute a very large program without overlay structure. But if programmer notifies that these pages will not be used any longer, the control program can schedule main memory better. On the other hand, if a program has overlay structure, it cannot be shared between users at the same time. Therefore, we have introduced the concept of logical overlay structure in OS7.

Fig. 4 shows the example using overlay structure in real memory system. Fig. 5 shows the same example using logical overlay structure. In Fig. 5, every load module (R, ϕ1, ϕ2, ϕ3, ϕ4, ϕ5) is given different virtual addresses.

Task T1, T2 and T3 are now executing in ϕ3 and ϕ1. Whenever a task tries to link a load module, link fault occurs. Thus the control program always knows which tasks are executing in the load module. The load modules, which are being used at least by one task, are tried to keep in main memory. The load modules, which are not being used by any task, are swapped out.

3. Conclusion

OS7 was installed and operated at TOKYO Institute of Technology in October of 1972 as the first version incorporating single processor system.

The 2nd version which supports multiprocessor system started its operation at TOKYO University Computer Center in January of 1973.

The system consists of 2-8800 CPUs and 2-8700 CPUs. It supports the services, open batch, closed batch, TSS, and remote batch processing.

As we are still tuning up software, the total throughput may improve in the future.

Acknowledgement

In the end we express our gratitude to the staffs in the center of the University of TOKYO and TOKYO Institute of Technology for the proper suggestions or advices from the user point of view.

REFERENCES

1 K. Nakazawa et al, The development of the high speed national project computer system, the 1st. USA-JAPAN Computer Conference Proceedings, October 1972, 173-181.

2 I. Ohnishi et al, Command Languages in OS7, Proceedings of the IFIP Working Conference on Command Languages, July 1974.

3 S. Domen et al, HITAC 8700 Operating System, HITACHI HYORON, Vol. 54 (1972).

4 P. J. Denning, Virtual Memory, Computing Surveys, Vol. 2, No.3 September 1973.