Advanced Computer Architecture

Lecture No. 40

Reading Material

Vincent P. Heuring & Harry F. Jordan — Chapter 7
Computer Systems Design and Architecture — 7.6

Summary

- Virtual Memory
- Virtual Memory Organization

Virtual Memory

Introduction

Virtual memory acts as a cache between main memory and secondary memory. Data is fetched in advance from the secondary memory (hard disk) into the main memory so that data is already available in the main memory when needed. The benefit is that the large access delays in reading data from hard disk are avoided.

Pages are formulated in the secondary memory and brought into the main memory. This process is managed both in hardware (Memory Management Unit) and the software (The operating systems is responsible for managing the memory resources).

The block diagram shown (Book Ch.7, Section 7.6, and figure 7.37) specifies how the data interchange takes place between cache, main memory and the disk. The Memory Management unit (MMU) is located between the CPU and the physical memory. Each memory reference issued by the CPU is translated from the logical address space to the physical address space, guided by operating system controlled mapping tables. As address translation is done for each memory reference, it must be performed by the hardware to speed up the process. The operating system is invoked to update the associated mapping tables.

Memory Management and Address Translation

The CPU generates the logical address. During program execution, effective address is generated which is an input to the MMU, which generates the virtual address. The virtual address is divided into two fields. First field represents the page number and the second field is the word field. In the next step, the MMU translates the virtual address into the physical address which indicates the location in the physical memory.

Advantages of Virtual Memory

- Simplified addressing scheme: the programmer does not need to bother about the exact locations of variables/instructions in the physical memory. It is taken care of by the operating system.
- For a programmer, a large virtual memory will be available, even for a limited physical memory.
- Simplified access control.

Virtual Memory Organization

Virtual memory can be organized in different ways. This first scheme is segmentation.

Segmentation:

In segmentation, memory is divided into segments of variable sizes depending upon the requirements. Main memory segments identified by segments numbers, start at virtual address 0, regardless of where they are located in physical memory.

In pure segmented systems, segments are brought into the main memory from the secondary memory when needed. If segments are modified and not required any more, they are sent back to secondary memory. This invariably results in gap between segments, called external fragmentation i.e. less efficient use of memory. Also refer to Book Ch.7 , Section 7.6, Figure 7.38.

Addressing of Segmented Memory

The physical address is formed by adding each virtual address issued by the CPU to the contents of the segment base register in the MMU. Virtual address may also be compared with the segment limit register to keep track and avoiding the references beyond the specified limit. By maintaining table of segment base and limit registers, operating system can switch processes by switching the contents of the segment base and limit register. This concept is used in multiprogramming. Refer to book Ch.7, Section 7.6, and Figure 7.39

Paging:

In this scheme, we have pages of fixed size. In demand paging, pages are available in secondary memory and are brought into the main memory when needed.

Virtual addresses are formed by concatenating the page number with the word number. The MMU maps these pages to the pages in the physical memory and if not present in the physical memory, to the secondary memory. (Refer to Book Ch.7, Section 7.6, and Figure 7.41)

Page Size: A very large page size results in increased access time. If page size is small, it may result in a large number of accesses.

The main memory address is divided into 2 parts.

- Page number: For virtual address, it is called virtual page number.
- Word Field

Virtual Address Translation in a Paged MMU:

Virtual address composed of a page number and a word number, is applied to the MMU. The virtual page number is limit checked to verify its availability within the limits given in the table. If it is available, it is added to the page table base address which results in a page table entry. If there is a limit check fault, a bound exception is raised as an interrupt to the processor.

Page Table

The page table entry for each page has two fields.

- Page field
- Control Field: This includes the following bits.
  - Access control bits: These bits are used to specify read/write, and execute permissions.

- Presence bits: Indicates the availability of page in the main memory.
- Used bits: These bits are set upon a read/ write.

If the presence bit indicates a hit, then the page field of the page table entry contains the physical page number. It is concatenated with the word field of the virtual address to form a physical address.
Page fault occurs when a miss is indicated by the presence bit. In this case, the page field of the page table entry would contain the address of the page in the secondary memory. Page miss results in an interrupt to the processor. The requesting process is suspended until the page is brought in the main memory by the interrupt service routine.
Dirty bit is set on a write hit CPU operation. And a write miss CPU operation causes the MMU to begin a write allocate (previously discussed) process. (Refer to book Ch.7, Section 7.6, and Figure 7.42)

Fragmentation:
Paging scheme results in unavoidable internal fragmentations i.e. some pages (mostly last pages of each process) may not be fully used. This results in wastage of memory.

Processor Dispatch -Multiprogramming
Consider the case, when a number of tasks are waiting for the CPU attention in a multiprogramming, shared memory environment. And a page fault occurs. Servicing the page fault involves these steps.

1. Save the state of suspended process
2. Handle page fault
3. Resume normal execution

Scheduling: If there are a number of memory interactions between main memory and secondary memory, a lot of CPU time is wasted in controlling these transfers and number of interrupts may occur.
To avoid this situation, Direct Memory Access (DMA) is a frequently used technique. The Direct memory access scheme results in direct link between main memory and secondary memory, and direct data transfer without attention of the CPU. But use of DMA in virtual memory may cause coherence problem. Multiple copies of the same page may reside in main memory and secondary memory. The operating system has to ensure that multiple copies are consistent.
Page Replacement
On a page miss (page fault), the needed page must be brought in the main memory from the secondary memory. If all the pages in the main memory are being used, we need to replace one of them to bring in the needed page. Two methods can be used for page replacement.
Random Replacement: Randomly replacing any older page to bring in the desired page.
Least Frequently Used: Maintain a log to see which particular page is least frequently used and to replace that page.
Translation Lookaside buffer
Identifying a particular page in the virtual memory requires page tables (might be very large) resulting in large memory space to implement these page tables. To speed up the process of virtual address translation, translation Lookaside buffer (TLB) is implemented

as a small cache inside the CPU, which stores the most recent page table entry reference made in the MMU. It contents include

- A mapping from virtual to physical address
- Status bits i.e. valid bit, dirty bit, protection bit

It may be implemented using a fully associative organization

Operation of TLB

For each virtual address reference, the TLB is searched associatively to find a match between the virtual page number of the memory reference and the virtual page number in the TLB. If a match is found (TLB hit) and if the corresponding valid bit and access control bits are set, then the physical page mapped to the virtual page is concatenated. (Refer to Book Ch.7, Section 7.6, and Figure 7.43)

Working of Memory Sub System

When a virtual address is issued by the CPU, all components of the memory subsystem interact with each other. If the memory reference is a TLB hit, then the physical address is applied to the cache. On a cache hit, the data is accessed from the cache. Cache miss is processed as described previously. On a TLB miss (no match found) the page table is searched. On a page table hit, the physical address is generated, and TLB is updated and cache is searched. On a page table miss, desired page is accessed in the secondary memory, and main memory, cache and page table are updated. TLB is updated on the next access (cache access) to this virtual address. (Refer to Book Ch.7, Section 7.6, and Figure 7.44).

To reduce the work load on the CPU and to efficiently use the memory sub system, different methods can be used. One method is separate cache for data and instructions.

Instruction Cache: It can be implemented as a Translation Lookaside buffer.

Data Cache: In data cache, to access a particular table entry, it can be implemented as a TLB either in the main memory, cache or the CPU.