CS 340
Lecturer: Dr. Simina Fluture

MEMORY MANAGEMENT

Read the material before lectures so we can keep up with the necessary lecture-pace. Do the homework (even if I will not collect it). If the time allows, we will be able to go in class over some of the homework. Most of the final will consist of problems based on the taught material.

Topics: Memory Management
Logical v. s. Physical Address Space
Types of Contiguous Allocation

Logical versus Physical Address Space
logical address: address generated by the CPU.
logical address space is the set of all logical addresses generated by a program.

physical address: the address loaded into the memory address register of the memory. Physical address or absolute address is an actual location in main memory.
physical address space is the set of all physical addresses.

The user program deals only with logical address and never sees the physical address.
The mapping from logical to physical address is done by *memory-management unit (MMU)* which is a hardware device.

Contiguous Allocation

In most schemes for memory management, the operating system occupies some fixed portion of main memory and the rest of main memory is available for use by multiple processes. Most of the time the operating system is placed in low memory. The core task of any memory management system is to bring programs into main memory for execution.

Single Partition Allocation
We need to protect the operating-system code and data from changes by user processes and also need to protect user processes from one another.
This is done using:
- relocation register (contains the value of the smallest physical address).
- limit register (contains the range of logical addresses).
The MMU maps the logical address dynamically by adding the value in the relocation register.

Fig: Hardware support for relocation and limit registers

Multiple-Partition Allocation
We need to consider the problem of how to allocate available memory to various processes that are in the job queue waiting to be brought into memory.
Fixed Partitioning
Equal-Size Partitioning
Divide main memory into equal fixed-sized partitions.
Disadvantages:

CS 340
Lecturer: Dr. Simina Fluture

Unequal-Size Partitioning
Divide main memory by using unequal-size partitions.
Disadvantages:

a) Equal-size partitions b) Unequal-size partitions

Dynamic Partitioning (OS/MVT - Multiprogramming with a Variable Number of Tasks)
The partitions used are of variable length and number.
When a process is brought into main memory, it is allocated exactly as much memory as it requires.
The OS keeps a table indicating which parts of memory are available and which are occupied. Initially, memory is considered as one large block of available memory, a *hole*.
As processes enter the system, there are put into an input (job) queue. At any given time, there is a list of available block sizes and the input queue.
The OS can order the input queue according to a scheduling algorithm. When a process arrives and needs memory, we search for a large enough memory hole.
If the hole is too large, it is split in two: one part is allocated to the process, the other is returned to the set of available blocks.
When a process terminates, it releases its block of memory.

Dynamic Storage Allocation Problem: How to satisfy a request of size *n* from a list of free holes.
Strategies used:
First-fit: allocate the first big enough hole. Searching can start either at the beginning of the set of holes, or where previous first-fit search ended.
Best-fit: allocate the *smallest* and big enough hole.
Worst-fit: allocate the *largest* hole. This strategy produces the largest leftover hole.

External and Internal Fragmentation
External Fragmentation: External fragmentation exists when enough total memory space exists to satisfy a request but it is not contiguous.
Solution: Compaction: The goal is to shuffle the memory contents to place all free memory together in one large block.

Compaction is possible only if relocation is dynamic and is done at execution time.
- selecting an optimal compaction strategy is difficult.
- swapping can be combined with compaction.

Internal Fragmentation: Internal Fragmentation occurs when there is available memory internal to a partition and cannot be used.

Homework: 9.5

CS 340
Lecturer: Dr. Simina Fluture

Topics: Paging
Read: Class Notes

A solution to the external fragmentation problem is to permit the logical address to be non-contiguous.
This can be implemented through the use of a paging scheme.
Physical memory is broken into fixed-sized blocks called **frames**.
Logical memory is broken into blocks of the same size called **pages** (usually between 2 and 8K)
The backing store is divided into fixed-sized blocks that are of the same size as the memory frames.
The page (same as frame) size is defined by the hardware depending on the computer architecture.

Translation between logical address and physical address
Page Table is a mapping that assigns to each virtual page a page frame number or an auxiliary storage address number.
The page table contains the base address of each page in physical memory.
Frame Table contains information about the physical memory: how many total frames are, which frames are allocated and to which page of which process, which frames are available.
When a process arrives to be executed, its size expressed in pages is examined. Each page of the process needs one frame.
If the process requires *n* pages, there must be at least *n* frames available in memory.
The first page of the process is loaded into one of the allocated frames, and the frame number is entered in the page table for this process.
The second page is loaded in the same manner and so on until the entire process is loaded.
The user's program address space will be scattered throughout the physical memory.

logical address:	**page number (p)**	**page offset (d)**
	m - n	**n**

p: is an index into the page table
d: is the displacement within the page

Figure Paging model of logical and physical memory

If **S** page size in bytes
R logical reference
f the frame number
then **Physical Address = f x S + offset = f x S + R mod S**
This is a mapping of the logical address into physical address using software computations.
More convenient is the address translation that uses the **Hardware Support**.

physical address: frame number (f) | frame offset (displacement) d

The OS maintains a copy of the page table for each process. This copy is used by the OS for address mapping.

CS 340
Lecturer: Dr. Simina Fluture

Structure of the Page Table
1) page table is implemented as a set of **dedicated registers**.
The use of registers for the page table is satisfactory if the page table is reasonably small.

2) the page table is kept in main memory and a **page-table-base-register (PTBR)** points to the page table. Changing page tables requires changing only this one register reducing the context switch time.

3) use a special, small fast-lookup hardware cache, called **associative registers** or **translation look-aside buffers (TLBs)**
Usually the associative registers contain only a few of the page-table entries. If the page number is found in the associative registers, its number is immediately available. If the page number is not in the associative registers, a memory reference to the page table must be made.
Search is fast, the hardware is expensive.

hit ratio the percentage of times that a page number is found in the associative registers.

Address Translation with Hardware Support
Case 1: DIRECT MAPPING
The page table is in primary memory.

Two memory accesses are needed to access a byte.
Slowdown: the page table has to be accessed in memory.

Case 2: ASSOCIATIVE MAPPING
Each register contains a key and a value (the frame number). Every entry in the associative storage is searched simultaneously.
Extra cost: the time necessary to search the associative registers. One memory reference is made.

Case 3: Combined ASSOCIATIVE/DIRECT
If the frame number is in the associative mapping the searching process stops, otherwise search further on P.T. In addition we add the page number and frame number to the associative registers, so that they will be found quickly in the next reference.

If **h = TLB hit ratio**
then **effective access time = h * TLB hit access time + (1 - h) * TLB miss access time**

hit access time =
miss access time =

Protection:
Memory protection in a paged environment is accomplished by protection bits that are associated with each frame.
One bit can define a page to be read-write or read-only or execute-only.

valid-invalid bit: if the bit is set to valid then the reference is legal.
if the bit is set to 'invalid' the page is not in the process's logical address space.

CS 340
Lecturer: Dr. Simina Fluture

Shared Pages
In multiprogrammed computer systems, especially in timesharing systems, it is common for users to execute same programs. If individual copies of these programs were given to each user, then too much of primary storage would be wasted.
Nonmodifiable code is named reentrant code. Nonmodifiable code can be shared.
If the code is reentrant then it never changes during execution.
Heavily used programs can be shared: compilers, database systems etc.
Modifiable code cannot be shared.
All this points to the need to identify each page as either sharable or nonsharable.

Homework: (I will not collect it)
9.8, 9.10.

Segmentation
The user prefers to view memory as a collection of variable-size segments, with no necessary ordering among segments.
Segmentation is a memory-management scheme by which a program is allowed to occupy many separate segments (blocks) of physical memory. The segments (blocks) themselves need not be the same size, must consist of contiguous locations but the separate segments need not to be adjacent to one another.
The logical address will be a collection of segments. These segments can represent procedures, functions or various data structures.
Each segment has a length and a name. Elements within a segment are identified by their offset from the beginning of the segment.

logical address < segment number, offset >

Hardware Implementation / Implementation of Segment Tables
The mapping from user-defined address into physical address is done using the segment table.
A segment table can be implemented either in fast registers or in memory.
The segment table has variable length, that's why it is more feasible to be in main memory.
Each entry of the segment table base a segment *base* and a segment *limit*.
When a particular process is running, a special register (segment table base register) will hold the Starting address of the segment table for that process.
Because the number of segments used by a program may vary, a segment table length register is used (STLR).

CS 340
Lecturer: Dr. Simina Fluture

Topics: Virtual Memory
Demand Paging
Page Replacement algorithms

<u>Virtual Memory</u>
Two characteristics of paging and segmentation are the keys to a breakthrough in memory management:
- all memory references within a process are logical addresses that are dynamically translated into physical addresses at run time.
A process may be swapped in and out of main memory such that occupies different regions of main memory at different times.
- a process may be broken up into a number of pieces (pages or segments) and these pieces need not to be contiguously located in main memory during the execution.
If these two characteristics are present, then it is not necessary that all the pages or all the segments of a process be in main memory during execution.
Virtual Memory is a technique that allows the execution of processes that may not be completely in memory.

Benefits:
- more processes may be maintained in main memory; this will bring an increase in CPU utilization and throughput.
- it is possible for a process to be larger than all the main memory. A program would no longer be constrained by the amount of physical memory that is available.
- less I/O would be needed to load or swap each user program into memory.
The 'Virtual memory' concept allows large virtual memory to be provided for programmers when only a smaller physical memory is available.

Virtual memory is implemented by:
- demand paging
- demand segmentation

<u>Demand Paging</u>
Demand Paging never swaps a page into memory unless that page will be needed (is demanded). Instead of swapping-in a whole process, the pager brings only those necessary pages into memory.
If a page was brought in main memory then the page is 'memory resident'.

valid - invalid bit:
'v' the associated page is both legal and in memory. (1)
'i' the page is either not valid or is valid but not in main memory. (0)

The page table entry might contain either the frame number, or the address of the page on the disk.
Access to a page marked invalid causes a 'page-fault' trap. Most of the time this trap is the result of the operating system's failure to bring the desired page into memory.
In case of a 'page fault' the O.S. reads the desired page into memory, into a free frame, and restarts the process from the interrupt point, as the page had always been in memory.

Figure: Page table when some pages are not in main memory.
Figure: Steps in handling a page fault

CS 340
Lecturer: Dr. Simina Fluture

Page Replacement
There exists the possibility that there are no 'free' frames in main memory.
The operating system has several actions at this point:
- terminate the process (not the best choice)
- swap out a process, freeing all its frames, and reducing the level of multiprogramming.
- page replacement: if no frame is free, we find one that is not currently being used and free it. The freed frame can now be used to hold the page for which the process faulted. All the tables has to be updated accordingly.

modify (dirty) bit:
0 if the page has not been modified.
1 if the page has been modified. (whenever any byte or word in the page is written into)

A possibility for deciding what frame will be replaced:
When we select a page for replacement, we examine its 'modify bit'.
If the bit is set, then that page has been modified since it was read in from the disk. In this case, we must write back the page to the disk. If the copy of the page on the disk has not been overwritten we can avoid writing the memory page to the disk.
This will shorten the page-fault service time.

Performance of Demand Paging
p - the probability of a page fault.

effective access time = (1-p) x ma with no page fault + p x page fault time

OBS: ma with no fault depends on the used mapping.

There are three major components of the page-fault service time:
1. Service the page-fault interrupt
2. Read in the page
3. Restart the process
It is important to keep the page fault rate low in a demand-paging system.
An average page-fault service is of order of milliseconds.
A memory access time is of order of nanoseconds.

CS 340
Lecturer: Dr. Simina Fluture

Page Replacement Algorithms How do you select a particular replacement algorithm? We want the one with the lowest *page-fault* rate.
An algorithm is evaluated by running it on a particular string of memory reference and computing the number of page faults.
reference string - the string of memory reference.
- to determine the number of page faults we need to know the number of page frames available.
- initially all frames are considered to be empty. When a specific page is demanded for the first time, it has to be brought from the disk into the main memory.
- if we have a reference to a page *p*, then any immediately following reference will not cause a page fault.

Frame Locking:
Some of the frames in main memory may be locked. When a frame is locked, the page currently stored in that frame may not be replaced. Much of the kernel, as well as the key control structures of the operating system is held on locked frames.
I/O buffers may be locked into main memory frames. Locking is done by associating a lock bit with each frame. This bit may be kept in a frame table.

The Locality Principle:
This principle states that the memory tends to cluster. Over a long period, the cluster in use changes, but over a short period, the processor is primarily working with fixed clusters of memory reference.
Note: Examples of algorithm behavior will be given in class.

Optimal Algorithm
Replace the page that will not be used for the longest period of time.

FIFO Algorithm
When a page must be replaced, the oldest page in memory is chosen.

Least Recently Used (LRU)
Replace the page that has not been used for the longest period of time.
- the LRU algorithm relies on the existence of *locality* in the page frame stream. Since programs are usually written as loops, then it is likely that the loop body will fit within some small number of frames. If a page has been referenced recently, it is likely to be referenced again soon.
- the LRU does nearly as well as the optimal policy.
- the major problem is *how* to implement the LRU algorithm.

Enhanced Second-Chance Algorithm (used in Macintosh virtual-memory management scheme).
We consider both the reference bit and the modify bit. There are 4 classes:
(0,0) - neither recently used, nor modified
(0,1) - not recently used but modified
(1,0) - recently used but clean
(1,1) recently used and modified
Replace the page that is in the lowest class.

Allocation of Frames
Number of Frames: The minimum number of frames is decided by the instruction set architecture. There must be enough frames to hold all the different pages that any different instruction can reference. The worst-case occurs in the architecture that allows multiple levels of indirection. To overcome this case, a limit on the levels of indirection must be considered. In the worst case the entire virtual memory must be in physical memory.
The maximum number of frames is defined by the amount of available physical memory.

CS 340
Lecturer: Dr. Simina Fluture

Allocation Algorithms
Global Versus Local Allocation
Global Replacement: allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process.
(the number of frame allocated to a specific process may change)
The process cannot control its own page-fault rate.

Local Replacement: each process selects from only its own set of allocated frames.
Local replacement might slow down a process by not making available to it other, less used pages of memory.
In general Global Replacement results in greater throughput and represents the common used method.

Thrashing
Other Considerations

Thrashing If a process spends more time paging than executing, then that process is thrashing.
As the degree of multiprogramming increases, CPU utilization increases until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in and CPU utilization drops sharply.

Fig. Thrashing

Trying to limit trashing:
- use a **local replacement algorithm**. At least the process that will start thrashing will not steal frames from other processes. Anyway, because any process will have to wait longer on the paging queue, the effective access time will increase even for a process that is not thrashing.
- use the ***Locality Principle***.

Locality is a set of pages that are actively used together. Localities are defined by the program structure and its data structures.
If we allocate fewer frames than the size of the current locality the process will trash.

Other Considerations
Prepaging In general, at the beginning of execution, all the frames allocated to a process are empty.
In the process of bringing the initial locality into memory, with each frame being filled up we will have a page fault.
When a swapped-out process is restarted, all its pages on the disk must be brought in by its own page fault.
By prepaging, a set of pages is brought in advance into main memory, before the process starts its execution.
These pages should be the ones most likely to be used in the next future.

Page Size (10.7.2)
Small page size:
- for a given virtual memory space, if the number of pages is increased, then the size of the page table is increased.
- by choosing a small page size, memory has a better utilization, the internal fragmentation is decreased.
Considering the I/O time:
- a smaller page size allows each page to match program locality more accurately.
- by increasing the page size, the I/O time also increase but at a much slower rate. (this is because the seek and latency time take considerable time compare to transfer time). This argues for a larger page size.

Considering the amount of reference needed:
- with a smaller page size there is a better resolution.
- with a larger page we must allocate and transfer also data that is not actually needed.

Considering the page-fault rate:
- large size page means more data in main memory, less page faults.

The problem has no best answer.
There are systems that are using two different pages sizes. (MULTICS 64 words and 1024 words), IBM/370 (2K and 4K)

Program Structure (10.7.4)

The structure of the program might increase or lower the page-fault rate and the number of pages in the working set.

Real-Time Processing

Real-time systems almost never have implemented the concept of virtual memory. Page faults might introduce a too big overhead that in such cases cannot be accepted.

Homework 10.5, 10.10, 10.11