CS 390
Chapter 9 Homework Solutions

9.2 `Why are page sizes always ...`

Page sizes that are a power of two because that makes it possible for the kernel to quickly determine the page number and offset of a logical address. The page number can be determined by right-shifting the logical address. The offset can be determined by masking out some number of the most-significant bits of the logical address.

9.4 `Consider a logical address space ...`

a. We need to calculate the total number logical addresses: There are 2^6 pages $\cdot 2^{10}$ words per page $= 2^{16}$ words in the logical address space. Thus, logical addresses need to be at least 16 bits.

b. 32KB of physical memory contains 2^{15} addresses, so there need to be a minimum of 15 bits in the physical address.

9.5 `What is the effect of allowing ...`

If two page table entries point to the same frame, then the two pages will map to the same frame. If we need to copy the contents of a large number of pages, we can allocate a range of logical addresses and then make the page table entries for those addresses point to the data to be copied. To the process, it looks like the data has been copied, when in reality, both "copies" of the data refer to the same physical memory. (We use this same scheme in object-oriented programming when we avoid copying an object and instead just copy the reference or pointer to the object.)

This is the technique used in many kernels to share memory; when two processes want to share a segment of memory, the kernel allocates pages for the shared segment, then adjusts the page tables of the two processes so that the specified logical addresses both map to the same shared physical pages.

On Linux, threads created using pthreads are separate processes, but these processes share their data segments. The sharing is actually accomplished through this page table magic.

9.7 `Assuming a 1-KB page size, ...`

Address	Page number	Offset
3085	3	13
42095	41	111
215201	210	161
650000	634	784
2000001	1953	129

9.9 `Consider a logical address space ...`

Remember that pages and page-frames are always the same size.

a. The size of the logical address space is
256 pages $\cdot$ 4-KB per page $= 2^8 \cdot 2^{12}$ bytes $= 2^{20}$ bytes. Logical addresses must be 20 bits wide.

b. The size of the physical address space is
64 frames $\cdot$ 4-KB per frame $= 2^6 \cdot 2^{12}$ bytes $= 2^{18}$ bytes. Physical addresses must be at least 18 bits wide.

Additional Exercises

A Explain the difference between internal and external fragmentation.

B Compare the memory organization schemes of contiguous memory allocation, contiguous memory allocation with variable-sized partitions, and paging, with respect to:

1. External fragmentation
2. Internal fragmentation
3. Ability to share text segments between processes

C Consider a paging system with the page table stored in memory.

1. If the system does not have a TLB, and a memory reference takes 50 ns, how long does a reference to paged memory take?
2. If we add a TLB, and 75% of all page-table references are found in the TLB, what is the effective memory reference time? Assume that a TLB lookup takes 2 ns, if the entry is present.

Answers to Additional Exercises

A Internal fragmentation is free space *inside* of a partition that can not be allocated to another process. External fragmentation is free space *outside* of a partition that cannot be used by any process because it is too small.

B **1. External fragmentation**

Contiguous Allocation with Fixed-Size Partitions: does not suffer from external fragmentation.

Contiguous Allocation with Variable-Size Partitions: suffers from external fragmentation.

Paging: does not suffer from external fragmentation.

2. Internal fragmentation

Contiguous Allocation with Fixed-Size Partitions: suffers from internal fragmentation.

Contiguous Allocation with Variable-Size Partitions: does not suffer from internal fragmentation.

Paging: suffers from internal fragmentation.

c. Ability to share text segments across processes

Contiguous Allocation with Fixed-Size Partitions: no support for code sharing across processes.

Contiguous Allocation with Variable-Size Partitions: no support for code sharing across processes.

Paging: supports text segment sharing across processes. This is accomplished by making the different logical addresses of the code to be shared map to the same physical addresses. However, we must make sure that the processes do not mix code and data in the same page. Otherwise, sharing the last page of the text segment between two processes might inadvertently share some variables.
When a new process is created in Linux, the kernel does not load the data segment immediatly after the last byte of the text segment. Instead, a gap is left so that the text segment starts on a new page.

C 1. 100 ns. 50 ns to access the page table entry, plus another 50n s to access the desired memory location. Storing the page table in

memory effectively doubles the time required to fetch a word from memory.

2. The effective access time is
$0.75 \cdot (2 + 50) + 0.25 \cdot (2 + 2 \cdot 50) = 64.5$ns.