Figure 4.64 shows how a page directory can be used to map linear addresses to 4-MB pages. The entries in the page directory point to page tables, and the entries in a page table point to 4-MB pages in physical memory. This paging method can be used to map up to 1024 pages into a 4 GB linear address space. In this case, the linear address is divided into two sections:

- *Directory*: Bits 22 through 31 contain an offset to an entry in the page directory. The selected entry provides the base physical address of a 4-MB page.
- *Offset*: Bits 0 through 21 contain an offset in the page.

It is possible to access both page tables for 4-KB pages and 4-MB pages from the same page directory. A typical example of mixing 4-KB and 4-MB pages is to place the operating system or executive's kernel in a large page to reduce TLB misses and thus improve overall system performance. The processor maintains 4KB-page entries and 4-MB page entries in separate TLBs. So, placing often used code, such as the operating system's kernel, in a large page, frees up 4 KB-page TLB entries for application programs and tasks and for less frequently used utilities.

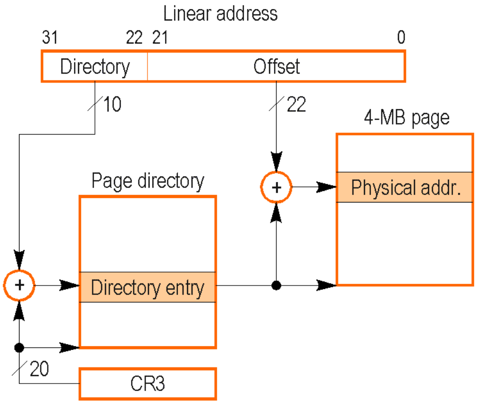

Figure 4.64. Linear address translation for the *Intel Architecture* (4-MB pages).

4.9. Problems

4.9.1. Using memory units of 4 × 2 bits of the type shown in Figure 4.65, design a 16 × 4 random-access memory unit.

4.9.2. Consider the generic 1D RAM organization illustrated in Figure 4.6. Assume the storage array is implemented by the DRAM cell of Figure 4.4(b). Describe three ways in which the RAM can be modified to double its transfer rate.

4.9.3. Using the 64-Mbit DRAM (8E1) presented in Section 4.4.4 as the basic component, design a 256M × 32-bit DRAM. Draw a diagram of the memory in the style of Figures 4.9 and 4.10.

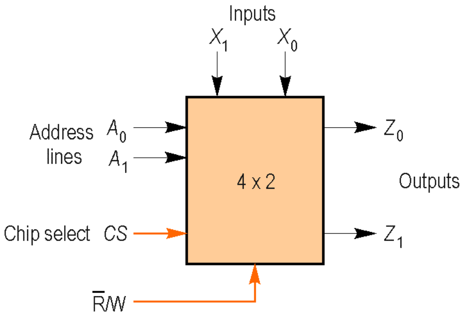

Figure 4.65. Simplified block diagram of a memory unit of 4 × 2 bits.

4.9.4. A 16-Mbits DRAM chip has a word size w = 8 bits. Like the 8E1 memory in Section 4.4.4 (Figure 4.11), it has a 2D organization with multiplexed row-column addressing. (*a*) If the column address is 10 bits, what is the size of the row address? (*b*) How many chips are needed to construct a 1G × 32-bit memory?

4.9.5. DRAM manufacturers sometimes sell RAM chips that are defective, but contain parts that are fully operational. Such units are used in low-cost applications such as toys. Describe how the 64-Mbit DRAM in Figure 4.11 can be used as a 512K × 4-bit DRAM.

4.9.6. A DRAM with 1M × 16-bit organization has four-way address interleaving with four memory banks M_0, M_1, M_2, and M_3. Identify the bank to which each of the following addresses is assigned: (*a*) 01234h; (*b*) ABCDEh; (*c*) 91272h; (*d*) FFFFFh.

4.9.7. Using associative memory cells of the type shown in Figure 4.35(b), design a 4 × 4-bit associative memory unit.

4.9.8. Build a small associative memory using as basic building block the 4 × 4-bit memory unit designed for Problem 4.9.7. The memory has to store 10 words of 8 bits each, with the format shown in Figure 4.66. Any one of the fields *A*, *B*, and *C* may be selected as the key. Assume that all stored keys are unique. When a match occurs, the entire matching word is to be fetched (read operation) or replaced (write operation). Draw a logic diagram for the memory including all access circuitry.

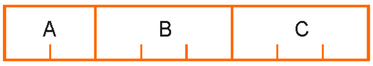

Figure 4.66. Word format for Problem 4.9.8.

4.9.9. (*a*) What is the access time of the four-level memory system specified in Table 4.9? (*b*) Suppose that, as a cost-saving measure, the second-level cache memory is eliminated from the system. Determine the new access time of the memory system.

Table 4.9. Characteristics of the memory system for Problem 4.9.9.

Memory	Capacity	Access time	Hit ratio
Cache Level 1	16 KB	10 ns	0.99000
Cache Level 2	256 KB	20 ns	0.99990
Main memory	32 MB	50 ns	0.99999
Disk memory	8 GB	10 ms	1.00000

4.9.10. A memory hierarchy has four levels M_1, M_2, M_3, and M_4 with hit ratios of 0.8, 0.95, 0.99, and 1.0, respectively. A program P makes 3000 references to this memory system. Calculate the number of references R_i made by P that are satisfied by an access to level M_i.

4.9.11. Why is set-associative cache organization better than direct-mapping and associative-mapping?

4.9.12. Consider a processor with a cache and main memory. Out of 100 requests that the processor sends to the cache memory, assume that the number of address references satisfied by the cache memory is N_c = 10, and the number of address references satisfied by the main memory is N_m = 90. The access time for the cache memory is t_c = 10 ns, and the access time for the main memory is t_m = 50 ns. (*a*) Compute the hit ratio, H, and the average access time, t_a, for these assumptions. (*b*) Assume that N_c = 60 and N_m = 40. Recompute H and t_a and compare the result with that in part (*a*).

4.9.13. (*a*) Suppose the cache memory in Figure 4.44 has its address lines labeled A_{31}:A_0, where A_{31} is the high-order address bit. Identify the 15 lines used to address the data RAM of the cache memory. (*b*) Assume that a single-word transfer over the system bus takes 10 ns. Estimate the time needed for the system to fully respond to a memory access when a cache miss occurs.

4.9.14. Draw a register-level diagram for the IDT 71B74 cache-tag RAM circuit used in Figure 4.44.

4.9.15. Redesign the direct-mapped cache memory described as an example in Section 4.7.4.2 (Figure 4.44) with the following changes: the capacity of the cache memory is to be reduced to 64 KB, and the block size as well as the width of the system data bus are to be 32 bits.

4.9.16. Suppose that a cache memory with a capacity of 2 KB has set-associative address mapping. There are 16 sets, each containing 4 blocks (lines). The memory-address size is 32 bits, and the smallest addressable unit is the byte. (*a*) To which set of the cache memory is the address 000010AFh assigned? (*b*) If the addresses 000010AFh and FFFF7*xyz*h can be simultaneously assigned to the same cache set, what values can the address digits *xyz* have?

4.9.17. An eight-way set-associative cache memory is used in a computer in which the physical memory size is 2^{32} bytes. The block size is 16 bytes, and there are 2^{10} blocks per set. Calculate the size of the cache memory and the length of the tag.

4.9.18. Design a four-way set-associative cache memory in the style of the example presented in Section 4.7.4.3 (Figure 4.47). The following parameters will be used: the capacity of the cache memory is 64 KB; the block size is 32 bytes; and the width of the system data bus is 32 bits.

4.9.19. Consider three small cache memories, each consisting of 4 one-word blocks. The first cache memory is fully associative, the second is direct mapped, and the third is two-way set associative. Assuming that the replacement policy is LRU, show in a table the contents of each cache memory and determine the number of misses for the following reference sequence: 0, 8, 0, 6, 8.

4.9.20. Assume a direct-mapped cache memory with 16 one-word blocks that is initially empty. Consider the following reference sequence given as word addresses: 0, 3, 7, 4, 19, 16, 18, 55, 8, 10, 3, 42, 4, 5, 8, 16. Label each reference in the list as a hit or miss and show the final contents of the cache memory.

4.9.21. Show the hits and misses and the final contents for a direct-mapped cache memory with a total size of 16 words and blocks of 4 words. Use the reference sequence listed in Problem 4.9.20.

4.9.22. Show the hits and misses and the final contents for a fully associative cache memory with a total size of 16 words and one-word blocks. Assume the LRU replacement policy. Use the reference sequence listed in Problem 4.9.20.

4.9.23. Show the hits and misses and the final contents for a fully associative cache memory with a total size of 16 words and blocks of 4 words. Assume the LRU replacement policy. Use the reference sequence listed in Problem 4.9.20.

4.9.24. Show the hits and misses and the final contents for a two-way set-associative cache memory with a total size of 16 words and one-word blocks. Assume the FIFO replacement policy. Use the reference sequence listed in Problem 4.9.20.

4.9.25. Some processors have cache memories that use a random page-replacement policy (RANDOM). The page to be replaced is selected by a fast process that approximates truly random selection and does not use any data on the page's reference history. Determine whether or not RANDOM is a stack replacement policy.

4.9.26. Consider a paged virtual memory system with the following characteristics:

- 40-bit virtual address;
- 16-KB pages;
- 36-bit physical address.

What is the total size of the page table for each process on this computer, assuming that the presence, access, and dirty bits take a total of 3 bits and that all the virtual pages are in use. Assume that disk addresses are not stored in the page table.

4.9.27. A memory reference by the PowerPC processor generates a 32-bit effective address A_{eff} that contains a 16-bit virtual address to a page of size 4 KB. Address A_{eff} also contains a pointer to a small set of segment registers that store segment descriptors. (*a*) How many segment registers does the PowerPC processor have? (*b*) Each segment descriptor includes a 24-bit segment address, called the virtual segment identifier (VSID). What is the size of the PowerPC's virtual address space?

4.9.28. A variation of the LRU replacement policy, called *simplified* LRU (SLRU), has been used in some virtual-memory systems. Every page P_i in an SLRU page table has a reference bit R_i associated with it. Whenever P_i is accessed, its reference bit R_i is set to 1. If the access request for P_i causes a page fault, then R_i is reset to 0 for all $j \neq i$ and P_i is loaded into main memory M_1. When a page in M_1 must be selected for replacement, the algorithm scans all the reference bits in a fixed order. The first page encountered with a reference bit of 0 is replaced. If all the reference bits are 1, then the page with the smallest logical address is replaced. (*a*) For the following reference sequence, determine the page-hit ratio for the SLRU and LRU policies, assuming that M_1 has a capacity of 3 pages and is initially empty: 1, 3, 1, 2, 4, 0, 2, 3, 0, 1, 4, 5. (*b*) Is SLRU a stack replacement policy? Justify the answer.

4.9.29. Consider the following reference sequence: 1, 4, 10, 4, 3, 6, 1, 1, 8, 6, 9, 4, 3, 3, 3, 1. What will be the number of page faults for a system that allocates to each process a maximum of four frames, using the following replacement policies: (*a*) FIFO; (*b*) LRU; (*c*) An approximation of the least frequently used policy that keeps a counter for each page. For each reference, the counter for that page is incremented. The page with the lowest count will be replaced. The system also uses a FIFO algorithm in cases when the page counts are the same.

4.9.30. Consider a two-dimensional integer array x defined as:

```
integer x [50][50];
```

Also consider a paged memory system with 4 page frames and a size of 100 words each. Assume that page frame 0 is used for the program, and the other three are initially empty. What will be the number of page faults for each of the following array initialization loops when using the LRU replacement policy?

(*a*)

```
for (i = 0; i < 50; i++)
    for (j = 0; j < 50; j++)
        x[i][j] = 0;
```

(*b*)

```
for (j = 0; j < 50; j++)
    for (i = 0; i < 50; i++)
        x[i][j] = 0;
```