Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

SKU: 9781498772716-SOLUTIONS

2 Computer Memory Systems

1. Consider the various aspects of an ideal computer memory discussed in Section 2.1.1 and the characteristics of available memory devices discussed in Section 2.1.2. Fill in the columns of the table below with the following types of memory devices, in order from most desirable to least desirable: magnetic hard disk, semiconductor DRAM, CD-R, DVD-RW, semiconductor ROM, DVD-R, semiconductor flash memory, magnetic floppy disk, CD-RW, semiconductor static RAM, semiconductor EPROM.

Cost/bit (will obviously fluctuate somewhat depending on market conditions): CD-R, DVD-R, CD-RW, DVD-RW, magnetic hard disk, magnetic floppy disk, semiconductor DRAM, semiconductor flash memory, semiconductor ROM, semiconductor EPROM, semiconductor static RAM.

Speed (will vary somewhat depending on specific models of devices): semiconductor static RAM, semiconductor DRAM, semiconductor ROM, semiconductor EPROM, semiconductor flash memory, magnetic hard disk, DVD-R, DVD-RW, CD-R, CD-RW, magnetic floppy disk.

Information Density (again, this may vary by specific types of devices): Magnetic hard disk, DVD-R and DVD-RW, CD-R and CD-RW, semiconductor flash memory, semiconductor DRAM, semiconductor ROM, semiconductor EPROM, semiconductor static RAM, magnetic floppy disk.

Volatility: Optical media such as DVD-R, CD-R, DVD-RW, and CD-RW are all equally nonvolatile. The read-only variants cannot be erased and provide secure storage unless physically damaged. (The same is true of semiconductor ROM.) The read-write optical disks (and semiconductor EPROMs and flash memories) may be intentionally or accidentally erased, but otherwise retain their data indefinitely in the absence of physical damage. Magnetic hard and floppy disks are nonvolatile except in the presence of strong external magnetic fields. Semiconductor static RAM is volatile, requiring continuous application of electrical power to maintain stored

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

data. Semiconductor DRAM is even more volatile since it requires not only electrical power, but also periodic data refresh in order to maintain its contents.

Writability (all memory is readable): Magnetic hard and floppy disks and semiconductor static RAM and DRAM can be written essentially indefinitely, and as quickly and easily as they can be read. DVD-RW, CD-RW, and semiconductor flash memory can be written many times, but not indefinitely, and the write operation is usually slower than the read operation. Semiconductor EPROMs can be written multiple times, but only in a special programmer, and only after a relatively long erase cycle under ultraviolet light. DVD-R and CD-R media can be written once and only once by the user. Semiconductor ROM is pre-loaded with its binary information at the factory and can never be written by the user.

Power Consumption: All types of optical and magnetic disks as well as semiconductor ROMs, EPROMs, and flash memories can *store* data without power being applied at all. Semiconductor RAMs require continuous application of power to retain data, with most types of SRAMS being more power-hungry than DRAMs. (Low-power CMOS static RAMs, however, are commonly used to maintain data for long periods of time with a battery backup.) While data are being read or written, all memories require power. Semiconductor DRAM requires relatively little power, while semiconductor ROMs, flash memories, and EPROMs tend to require more and SRAMs, more still. All rotating disk drives, magnetic and optical, require significant power in order to spin the media and move the read/write heads as well as to actually perform the read and write operations. The specifics vary considerably from device to device, but those that rotate the media at higher speeds tend to use slightly more power.

Durability: In general, the various types of semiconductor memories are more durable than disk memories because they have no moving parts. Only severe physical shock or static discharges are likely to harm them. (CMOS devices are

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

particularly susceptible to damage from static electricity.) Optical media are also very durable; they are nearly impervious to most dangers except that of surface scratches. Magnetic media such as floppy and hard disks tend to be the least durable as they are subject to erasure by strong magnetic fields and also are subject to "head crashes" when physical shock causes the read-write head to impact the media surface.

Removability/Portability: Flash memory, floppy disks, and optical disks are eminently portable and can easily be carried from system to system to transfer data. A few magnetic hard drives are designed to be portable, but most are permanently installed in a given system and require some effort for removal. Semiconductor ROMs and EPROMs, if placed in sockets rather than being soldered directly to a circuit board, can be removed and transported along with their contents. Most semiconductor RAM devices lose their contents when system power is removed and, while they could be moved to another system, would not arrive containing any valid data.

2. Describe in your own words what a hierarchical memory system is and why it is used in the vast majority of modern computer systems.

A hierarchical memory system is one that is comprised of several types of memory devices with different characteristics, each occupying a "level" within the overall structure. The higher levels of the memory system (the ones closest to, or a part of, the CPU) offer faster access but, due to cost factors and limited physical space, have a smaller storage capacity. Thus, each level can typically hold only a portion of the data stored in the next lower level. As one moves down to the lower levels, speed and cost per bit generally decrease, but capacity increases. At the lowest levels, the devices offer a great deal of (usually nonvolatile) storage at relatively low cost, but are quite slow. For the overall system to perform well, the hierarchy must be managed by hardware and software such that the stored items

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

that are used most frequently are located in the higher levels, while items that are used less frequently are relegated to the lower levels.

3. What is the fundamental, underlying reason why low-order main memory interleaving and/or cache memories are needed and used in virtually all high-performance computer systems?

The main underlying reason why speed-enhancing techniques such as low-order interleaving and cache continue to be needed and used in computer systems is that main memory technology has never been able to keep up with the speed of processor implementation technologies. The CPUs of each generation have always been faster than any devices (from the days of delay lines, magnetic drums, and core memory all the way up to today's high-capacity DRAM ICs) that were feasible, from a cost standpoint, to be used as main memory. If anything, the CPU-memory speed gap has widened rather than narrowed over the years. Thus, the speed and size of a system's cache may be even more critical to system performance than almost any other factor. (If you don't believe this, examine the performance difference between an Intel Core series processor and an otherwise similar Celeron processor.)

4. A main memory system is designed using 15 ns RAM devices using a 4-way low-order interleave.

(a) What would be the effective time per main memory access under ideal conditions?

Under ideal conditions, four memory accesses would be in progress at any given time due to the low-order interleaving scheme. This means that the effective time per main memory access would be (15 / 4) = 3.75 ns.

(b) What would constitute "ideal conditions"? (In other words, under what circumstances could the access time you just calculated be achieved?)

The ideal condition for best performance of the memory system would be

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

continuous access to sequentially numbered memory locations. Equivalently, any access pattern that consistently used all three of the other "leaves" before returning to the one just accessed would have the same benefit. Examples would include accessing every fifth numbered location, or every seventh, or any spacing that is relatively prime with 4 (the interleaving factor).

(c) What would constitute "worst-case conditions"? (In other words, under what circumstances would memory accesses be the slowest?) What would the access time be in this worst-case scenario? If ideal conditions exist 80% of the time and worst-case conditions occur 20% of the time, what would be the average time required per memory access?

The worst case would be a situation where every access went to the same device or group of devices. This would happen if the CPU needed to access every fourth numbered location (or every eighth, or any spacing that is an integer multiple of 4). In this case, access time would revert to that of an individual device (15 ns) and the interleaving would provide no performance benefit at all.

In the hypothetical situation described, we could take a weighted average to determine the effective access time for the memory system: (0.80)(3.75 ns) + (0.20)(15 ns) = (3 + 3) = 6 ns.

(d) When ideal conditions exist, we would like the processor to be able to access memory every clock cycle with no "wait states" (that is, without any cycles wasted waiting for memory to respond). Given this requirement, what is the highest processor bus clock frequency that can be used with this memory system?

In part (a) above, we found the best-case memory access time to be 3.75 ns. Matching the CPU bus cycle time to this value and taking the reciprocal (since $f = 1/T$) we obtain:

$f = 1/T = (1 \text{ cycle}) / (3.75 * 10^{-9} \text{ seconds}) \approx 2.67 * 10^{8} \text{ cycles/second} = 267 \text{ MHz}.$

(e) Other than increased hardware cost and complexity, are there any potential

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

disadvantages of using a low-order interleaved memory design? If so, discuss one such disadvantage and the circumstances under which it might be significant.

The main disadvantage that could come into play is due to the fact that under ideal conditions, all memory modules are busy all the time. This is good if only one device (usually the CPU) needs to access memory, but not good if other devices need to access memory as well (for example, to perform I/O). Essentially all the memory bandwidth is used up by the first device, leaving little or none for others.

Another possible disadvantage is lower memory system reliability due to decreased fault tolerance. In a high-order interleaved system, if one memory device were to fail, 3/4 of the memory space would still be usable. In the low-order interleaved case, if one of the four "leaves" fails, the entire main memory space is effectively lost.

5. Is it correct to refer to a typical semiconductor integrated circuit ROM as a "random access memory"? Why or why not? Name and describe two other logical organizations of computer memory that are not "random access."

It is correct to refer to a semiconductor ROM as a "random access memory" in the strict sense of the definition – a "random access" memory is *any* memory device that has an access time independent of the specific location being accessed. (In other words, any randomly chosen location can be read or written in the same amount of time as any other location.) This is equally true of most semiconductor read-only memories as it is of semiconductor read/write memories (which are commonly known as "RAMs"). Because of the commonly-used terminology, it is probably better not to confuse the issue by referring to a ROM IC as a "RAM", even though that is technically a correct statement.

Besides *random access*, the other two logical memory organizations that may be found in computer systems are *sequential access* (typical of tape and disk

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

memories) and *associative* (or content-addressable).

6. Assume that a given system's main memory has an access time of 6.0 ns, while its cache has an access time of 1.2 ns (five times as fast). What would the hit ratio need to be in order for the effective memory access time to be 1.5 ns (four times as fast as main memory)?

Since effective memory access time in such a system is based on a weighted average, we would need to solve the following equation:

$t_{a\ \text{effective}} = t_{a\ \text{cache}} * (p_h) + t_{a\ \text{main}} * (1 - p_h)$

for the particular values given in the problem, as shown:

1.5 ns = (1.2 ns)(p_h) + (6.0 ns)(1 - p_h)

Using basic algebra we solve to obtain p_h = 0.9375.

7. A particular program runs on a system with cache memory. The program makes a total of 250,000 memory references; 235,000 of these are to cached locations.

(a) What is the hit ratio in this case?

p_h = number of hits / (number of hits + number of misses) = 235,000 / 250,000 = 0.94

(b) If the cache can be accessed in 1.0 ns but the main memory requires 7.5 ns for an access to take place, what is the average time required by this program for a memory access assuming all accesses are reads?

$t_{a\ \text{effective}} = t_{a\ \text{cache}} * (p_h) + t_{a\ \text{main}} * (1 - p_h)$ = (1.0 ns)(0.94) + (7.5 ns)(0.06) = (0.94 + 0.45) ns = 1.39 ns

(c) What would be the answer to part (b) if a write-through policy is used and 75% of memory accesses are reads?

If a write-through policy is used, then all writes require a main memory access and write hits do nothing to improve memory system performance. The average write access time is equal to the main memory access time, which is 7.5 ns. The average read access time is equal to 1.39 ns as calculated in (b) above. The overall average time per memory access is thus given by:

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

$t_{a\ \text{effective}}$ = (7.5 ns)(0.25) + (1.39 ns)(0.75) = (1.875 + 1.0425) ns = 2.9175 ns

8. Is hit ratio a dynamic or static performance parameter in a typical computer memory system? Explain your answer.

Hit ratio is a dynamic parameter in any practical computer system. Even though the cache and main memory sizes, mapping strategy, replacement policy, etc. (which can all affect the hit ratio) are constant within a given system, the proportion of cache hits to misses will still vary from one program to another. It will also vary widely within a given run, based on such factors as the length of time the program has been running, the code structure (procedure calls, loops, etc.) and the properties of the specific data set being operated on by the program.

9. What are the advantages of a set-associative cache organization as opposed to a direct-mapped or fully associative mapping strategy?

A set-associative cache organization is a compromise between the direct-mapped and fully associative organizations that attempts to maximize the advantages of each while minimizing their respective disadvantages. Fully associative caches are expensive to build but offer a higher hit ratio than direct-mapped caches of the same size. Direct-mapped caches are cheaper and less complex to build but performance can suffer due to usage conflicts between lines with the same index. By limiting associativity to just a few parallel comparisons (two- and four-way set-associative caches are most common) the set-associative organization can achieve nearly the same hit ratio as a fully associative design at a cost not much greater than that of a direct-mapped cache.

10. A computer has 64 MB of byte-addressable main memory. It is proposed to design a 1 MB cache memory with a refill line (block) size of 64 bytes.

(a) Show how the memory address bits would be allocated for a direct-mapped cache organization.

Since 64M = 2^{26}, the total number of bits required to address the main

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

memory space is 26. And since 64 = 2^6, it takes 6 bits to identify a particular byte within a line. The number of refill lines in the cache is 1M / 64 = 2^{20} / 2^6 = 2^{14} = 16K. Since there are 2^{14} lines in the cache, 14 index bits are required. 26 total address bits – 6 "byte" bits – 14 "index" bits leaves 6 bits to be used for the tag. So the address bits would be partitioned as follows: Tag (6 bits) | Index (14 bits) | Byte (6 bits)

(b) Repeat part (a) for a four-way set-associative cache organization.

For the purposes of this problem, a four-way set-associative cache can be treated as four direct-mapped caches operating in parallel, each one-fourth the size of the cache described above. Each of these four smaller units would thus be 256 KB in size, containing 4K = 2^{12} refill lines. Thus, 12 bits would need to be used for the index, and 26 – 6 – 12 = 8 bits would be used for the tag. The address bits would be partitioned as follows: Tag (8 bits) | Index (12 bits) | Byte (6 bits)

(c) Repeat part (a) for a fully associative cache organization.

In a fully associative cache organization, no index bits are required. Therefore the tags would be 26 – 6 = 20 bits long. Addresses would be partitioned as follows: Tag (20 bits) | Byte (6 bits)

(d) Given the direct-mapped organization, and ignoring any extra bits that might be needed (valid bit, dirty bit, etc.), what would be the overall size ("depth" by "width") of the memory used to implement the cache? What type of memory devices would be used to implement the cache (be as specific as possible)?

The overall size of the direct-mapped cache would be: (16K lines) * (64 data bytes + 6 bit tag) = (16,384) * ((64 * 8) + 6) = (16,384 * 518) = 8,486,912 bits. This would be in the form of a fast 16K by 518 static RAM.

(e) Which line(s) of the direct-mapped cache could main memory location $1E0027A_{16}$ map into? (Give the line number(s), which will be in the range of 0 to (n-1) if there are n lines in the cache.) Give the memory address (in hexadecimal) of another location that could not reside in cache at the same time as this one (if

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

such a location exists).

To answer this question, we need to write the memory address in binary. 1E0027A hexadecimal equals 0111100000000000001001111010 binary. We can break this down into a tag of 011110, an index of 00000000001001 and a byte offset within the line of 111010. In a direct-mapped cache, the binary index tells us the number of the only line that can contain the given memory location. So, this location can only reside in line 1001_2 = 9 decimal.

Any other memory location with the same index but a different tag could *not* reside in cache at the same time as this one. One example of such a location would be the one at address $2F0027A_{16}$.

11. Define and describe virtual memory. What are its purposes, and what are the advantages and disadvantages of virtual memory systems?

Virtual memory is a technique that separates the (virtual) addresses used by the software from the (physical) addresses used by the memory system hardware. Each virtual address referenced by a program goes through a process of translation (or mapping) that resolves it into the correct physical address in main memory, if such a mapping exists. If no mapping is defined, the desired information is loaded from secondary memory and an appropriate mapping is created. The translation process is overseen by the operating system, with much of the work done in hardware by a memory management unit (MMU) for speed reasons. It is usually done via a multi-level table lookup procedure, with the MMU internally caching frequently- or recently-used translations so that the costly (in terms of performance) table lookups can be avoided most of the time.

The principal advantage of virtual memory is that it frees the programmer from the burden of fitting his or her code into available memory, giving the illusion of a large memory space exclusively owned by the program (rather than the usually much more limited physical main memory space that is shared with other resident

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

programs). The main disadvantage is the overhead of implementing the virtual memory scheme, which invariably results in some increase in average access time vs. a system using comparable technology with only physical memory. Table lookups take time, and even when a given translation is cached in the MMU's Translation Lookaside Buffer, there is some propagation delay involved in address translation.

12. Name and describe the two principal approaches to implementing virtual memory systems. How are they similar and how do they differ? Can they be combined, and if so, how?

The two principal approaches to implementing virtual memory (VM) are *demand-paged* VM and *demand-segmented* VM (*paging* and *segmentation*, for short). They are similar in that both map a virtual (or logical) address space to a physical address space using a table lookup process managed by an MMU and overseen by the computer's operating system. They are different in that paging maps fixed-size regions of memory called pages, while segmentation maps variable-length segments. Page size is usually determined by hardware considerations such as disk sector size, while segment size is determined by the structure of the program's code and data. A paged system can concatenate the offset within a page with the translated upper address bits, while a segmented system must translate a logical address into the complete physical starting address of a segment and then add the segment offset to that value.

It is possible to create a system that uses aspects of both approaches; specifically, one in which the variable-length segments are each comprised of one or more fixed-sized pages. This approach, known as segmentation with paging, trades off some of the disadvantages of each approach to try to take advantage of their strengths.

13. What is the purpose of having multiple levels of page or segment tables rather than a single table for looking up address translations? What are the disadvantages, if any, of

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

this scheme?

The main purpose of having multiple-level page or segment tables is to replace one huge mapping table with a hierarchy of smaller ones. The advantage is that the tables are smaller (remember, they are stored in main memory, though some entries may be cached) and easier for the operating system to manage. The disadvantage is that "walking" the hierarchical sequence of tables takes longer than a single table lookup. Most systems have a TLB to cache recently-used address translations, though, so this time penalty is usually only incurred once when a given page or segment is first loaded into memory (or perhaps again later if the TLB fills up and a displaced entry has to be reloaded).

14. A process running on a system with demand-paged virtual memory generates the following reference string (sequence of requested pages): 4, 3, 6, 1, 5, 1, 3, 6, 4, 2, 2, 3. The operating system allocates each process a maximum of four page frames at a time. What will be the number of page faults for this process under each of the following page replacement policies?

a) LRU **7 page faults**

b) FIFO **8 page faults**

c) LFU (with FIFO as tiebreaker) **7 page faults**

15. In what ways are cache memory and virtual memory similar? In what ways are they different?

Cache memory and virtual memory are similar in several ways. Both involve the interaction between two levels of a hierarchical memory system – one larger and slower, the other smaller and faster. Both have the goal of performing close to the speed of the smaller, faster memory while taking advantage of the capacity of the larger, slower one; both depend on the principle of locality of reference to achieve this. Both operate on a demand basis and both perform a mapping of addresses generated by the CPU.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

One significant difference is the size of the blocks of memory that are mapped and transferred between levels of the hierarchy. Cache lines tend to be significantly smaller than pages or segments in a virtual memory system. Because of the size of the mapped areas as well as the speed disparity between levels of the memory system, cache misses tend to be more frequent, but less costly in terms of performance, than page or segment faults in a VM system. Cache control is done entirely in hardware, while virtual memory management is accomplished via a combination of hardware (the MMU) and software (the operating system). Cache exists for the sole reason of making main memory appear faster than it really is; virtual memory has several purposes, one of which is to make main memory appear larger than it is, but also to support multiprogramming, relocation of code and data, and the protection of each program's memory space from other programs.

16. In systems which make use of both virtual memory and cache, what are the advantages of a virtually addressed cache? Does a physically addressed cache have any advantages of its own, and if so, what are they? Describe a situation in which one of these approaches would have to be used because the other would not be feasible.

All else being equal, a virtually mapped cache is faster than a physically mapped cache because no address translation is required prior to checking the tags to see if a hit has occurred. The appropriate bits from the virtual address are matched against the (virtual) tags. In a physically addressed cache, the virtual-to-physical translation must be done before the tags can be matched. A physically addressed cache does have some advantages, though, including the ability to perform task switches without having to flush (invalidate) the contents of the cache. In a situation where the MMU is located on-chip with the CPU while a cache is located off-chip (for example a level-2 or level-3 cache on the motherboard) the address is already translated before it appears on the system bus and, therefore, that cache would have to be physically addressed.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

17. Fill in the blanks below with the most appropriate term or concept discussed in this chapter:

Information density - A characteristic of a memory device that refers to the amount of information that can be stored in a given physical space or volume.

Dynamic Random Access Memory (DRAM) - A semiconductor memory device made up of a large array of capacitors; its contents must be periodically refreshed in order to keep them from being lost.

Magnetic RAM (MRAM) - A developing memory technology that operates on the principle of magnetoresistance; it may allow the development of "instant-on" computer systems.

Erasable/Programmable Read-Only Memory (EPROM) - A type of semiconductor memory device, the contents of which cannot be overwritten during normal operation, but can be erased using ultraviolet light.

Associative memory - This type of memory device is also known as a CAM.

Argument register - A register in an associative memory that contains the item to be searched for.

Locality of reference - The principle that allows hierarchical storage systems to function at close to the speed of the faster, smaller level(s).

Miss - This occurs when a needed instruction or operand is not found in cache and thus a main memory access is required.

Refill line - The unit of information that is transferred between a cache and main memory.

Tag - The portion of a memory address that determines whether a cache line contains the needed information.

Fully associative mapping - The most flexible but most expensive cache organization, in which a block of information from main memory can reside anywhere in the cache.

Write-back - A policy whereby writes to cached locations update main memory only

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

when the line is displaced.

Valid bit - This is set or cleared to indicate whether a given cache line has been initialized with "good" information or contains "garbage" due to not yet being initialized.

Memory Management Unit (MMU) - A hardware unit that handles the details of address translation in a system with virtual memory.

Segment fault - This occurs when a program makes reference to a logical segment of memory that is not physically present in main memory.

Translation Lookaside Buffer (TLB) - A type of cache used to hold virtual-to-physical address translation information.

Dirty bit - This is set to indicate that the contents of a faster memory subsystem have been modified and need to be copied to the slower memory when they are displaced.

Delayed page fault - This can occur during the execution of a string or vector instruction when part of the operand is present in physical main memory and the rest is not.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition Dumas II Solutions Manual

Courtesy of CRC Press/Taylor & Francis Group

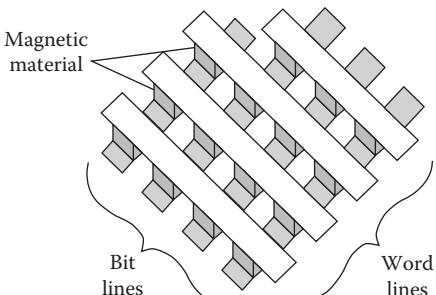


Figure 2.1 Magnetic RAM construction.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

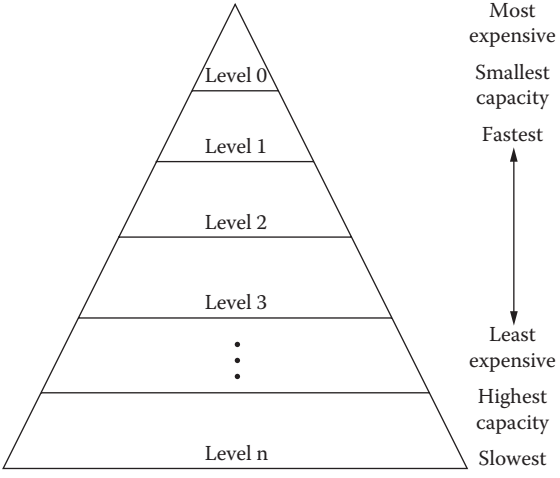


Figure 2.2 Memory hierarchy (conceptual).

002x002.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

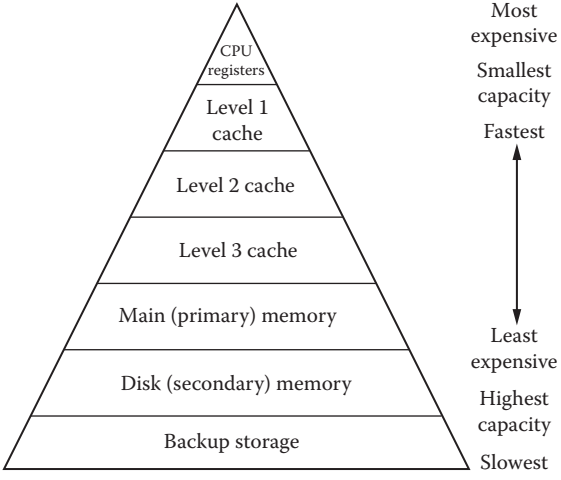


Figure 2.3 Memory hierarchy (typical of modern computer systems).

002x003.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

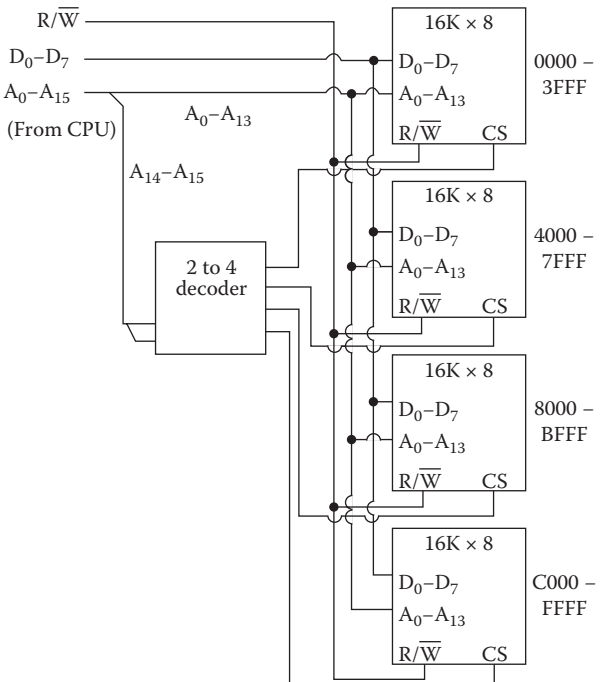


Figure 2.4 Simple memory system using high-order interleaving.

002x004.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

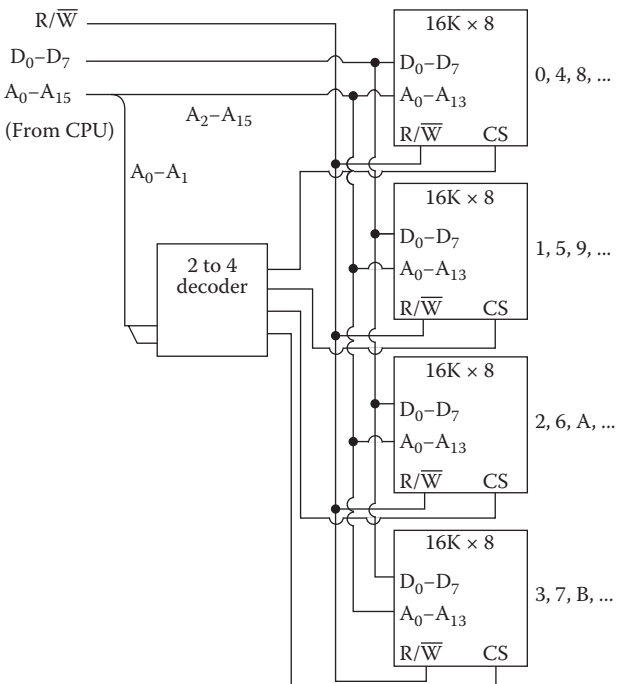


Figure 2.5 Simple memory system using low-order interleaving.

002x005.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

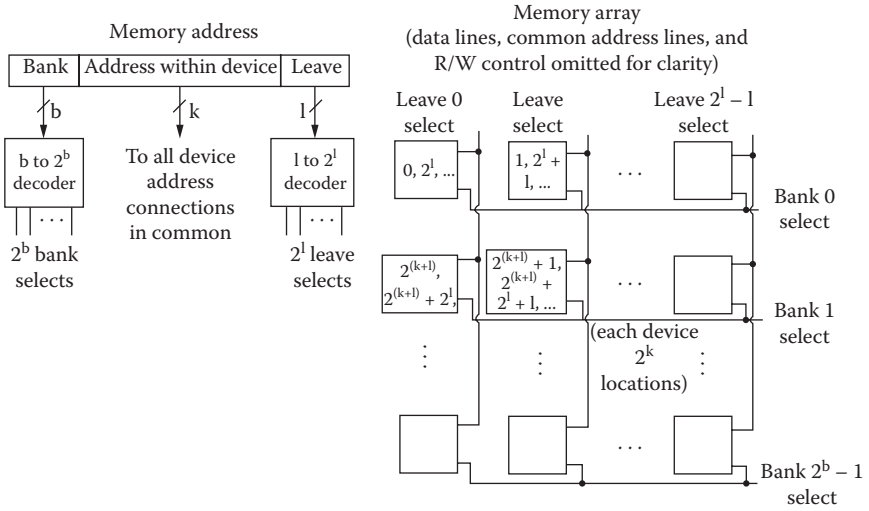


Figure 2.6 Memory system using both high- and low-order interleaving.

002x006.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

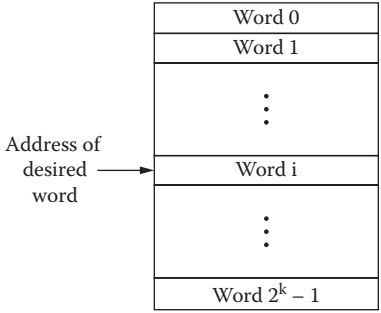


Figure 2.7 Memory access by word slice.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

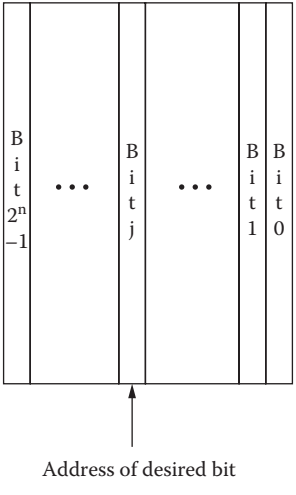


Figure 2.8 Memory access by bit slice.

002x008.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

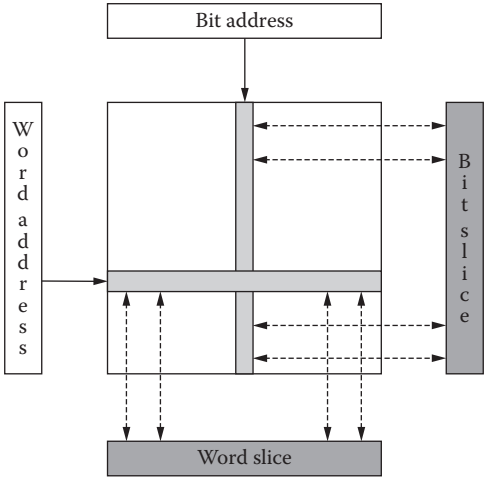


Figure 2.9 Orthogonal memory.

002x009.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

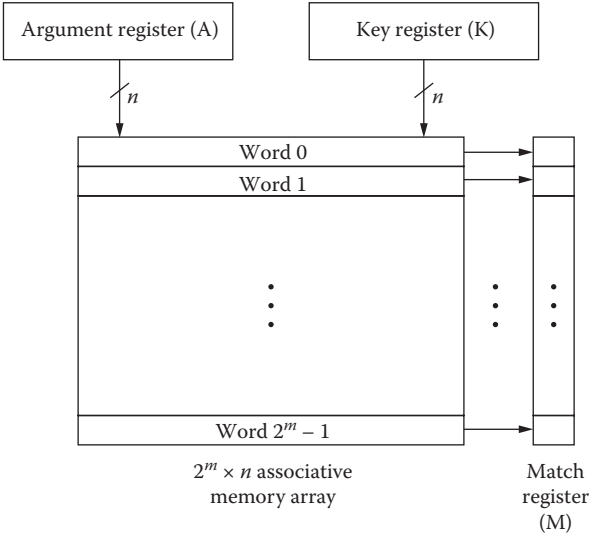


Figure 2.10 Associative memory block diagram.

002x010.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

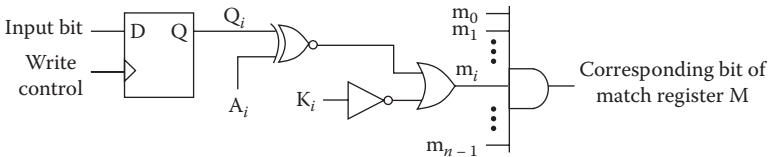


Figure 2.11 Associative memory cell logic.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

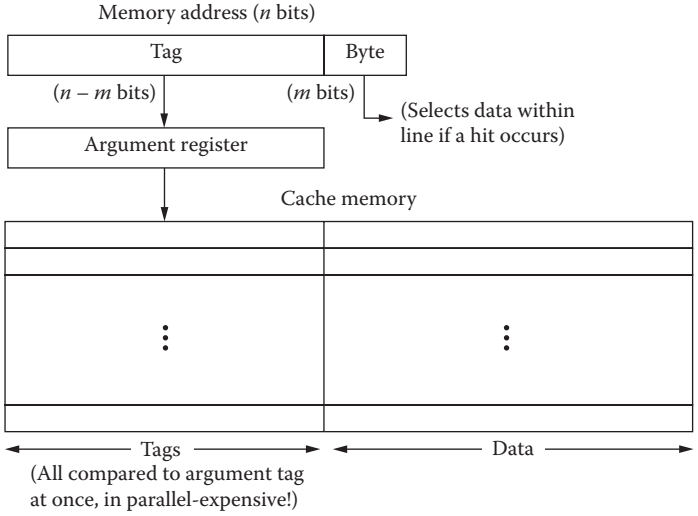


Figure 2.12 Fully associative cache.

002x012.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

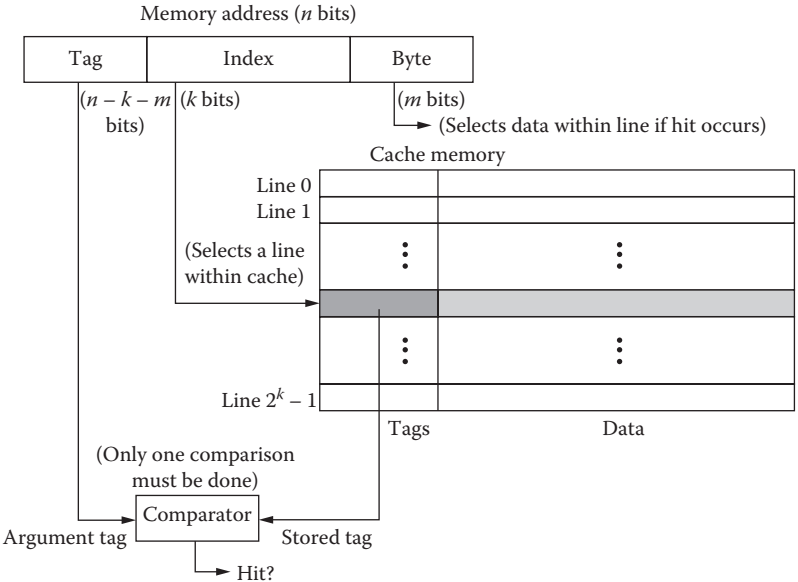


Figure 2.13 Direct-mapped cache.

002x013.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

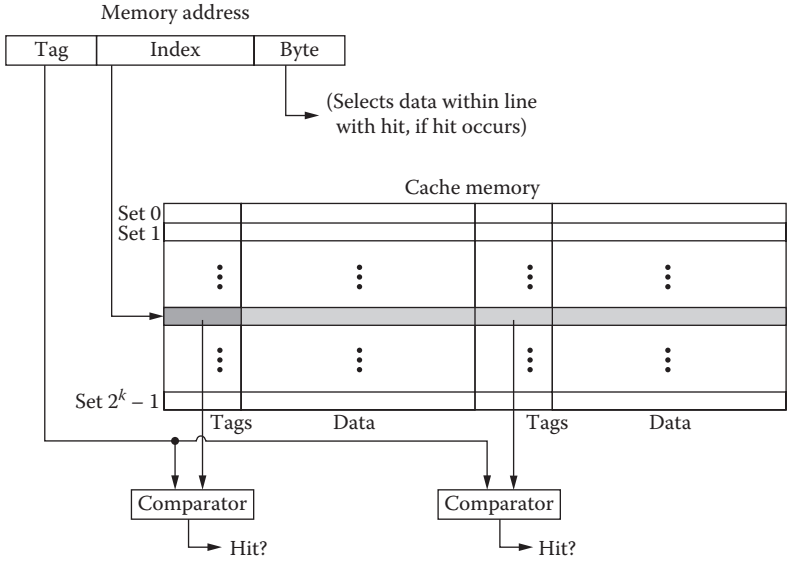


Figure 2.14 Two-way set-associative cache.

002x014.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

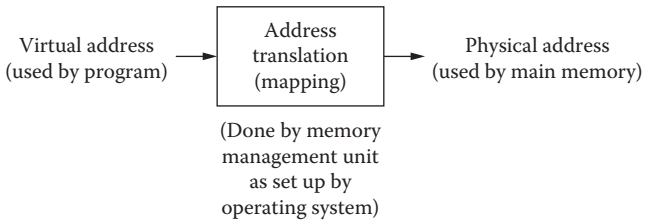


Figure 2.15 Address translation in a system with virtual memory.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

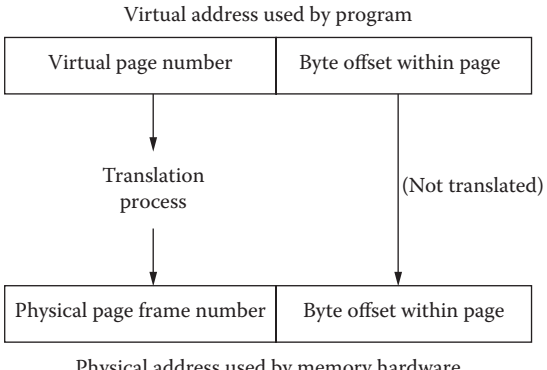


Figure 2.16 Address translation in a system with demand-paged virtual memory.

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

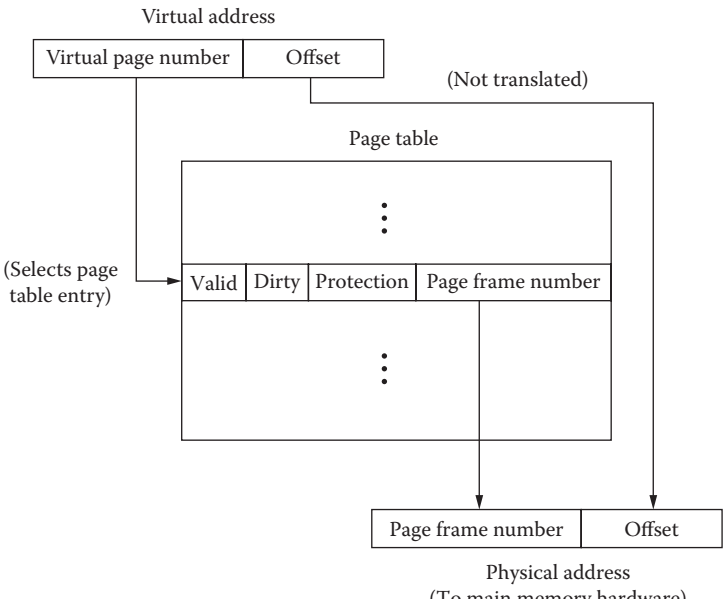


Figure 2.17 Table lookup for address translation in a paged system.

002x017.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

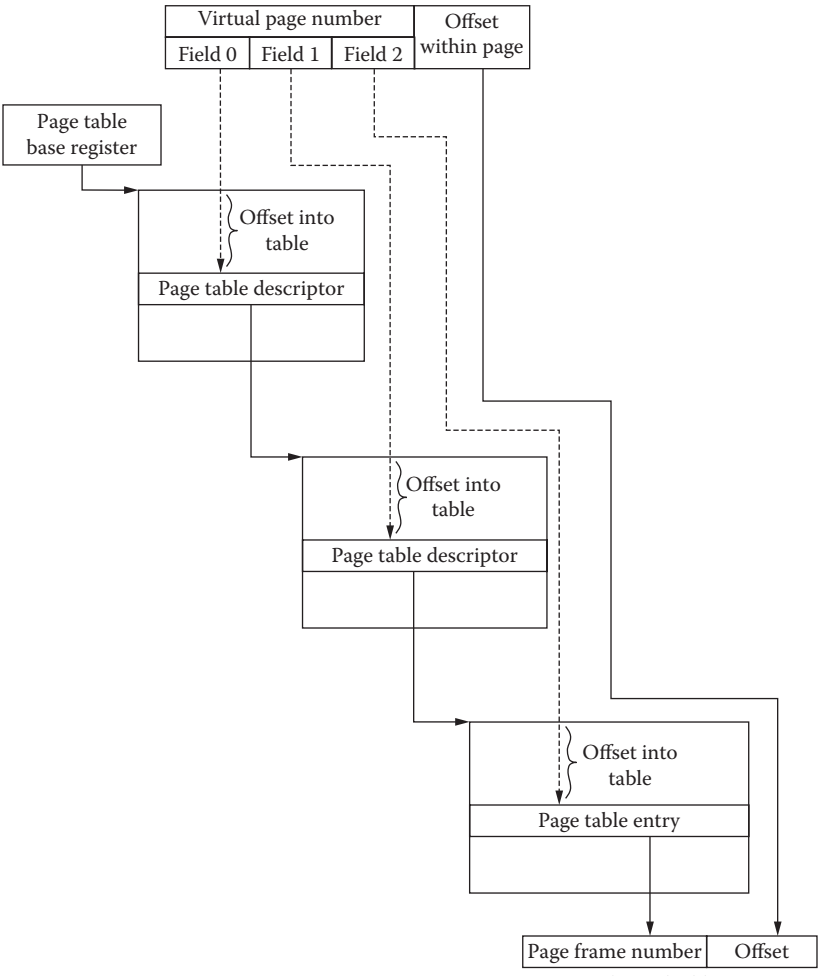


Figure 2.18 Multiple-level table lookup in paged system.

002x018.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

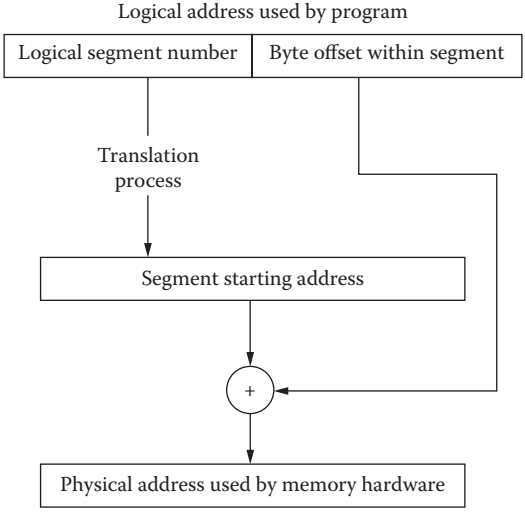


Figure 2.19 Address translation in a system with demand-segmented virtual memory.

002x019.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

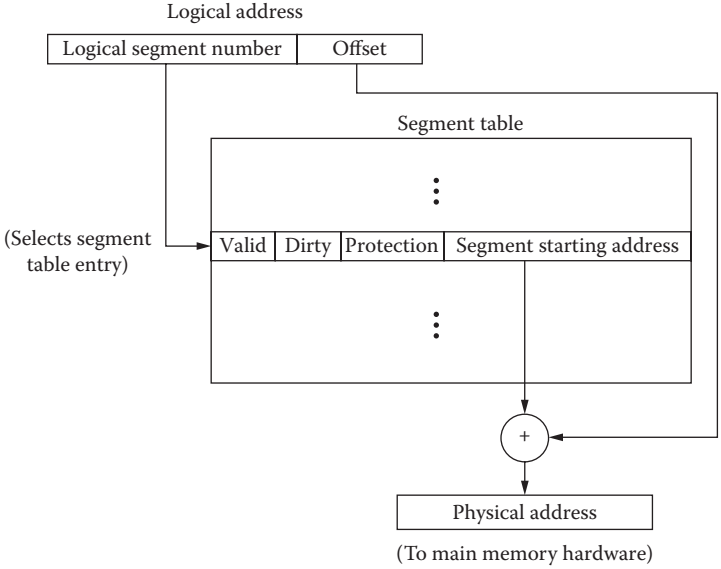


Figure 2.20 Table lookup for address translation in a segmented system.

002x020.eps
Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D

Courtesy of CRC Press/Taylor & Francis Group

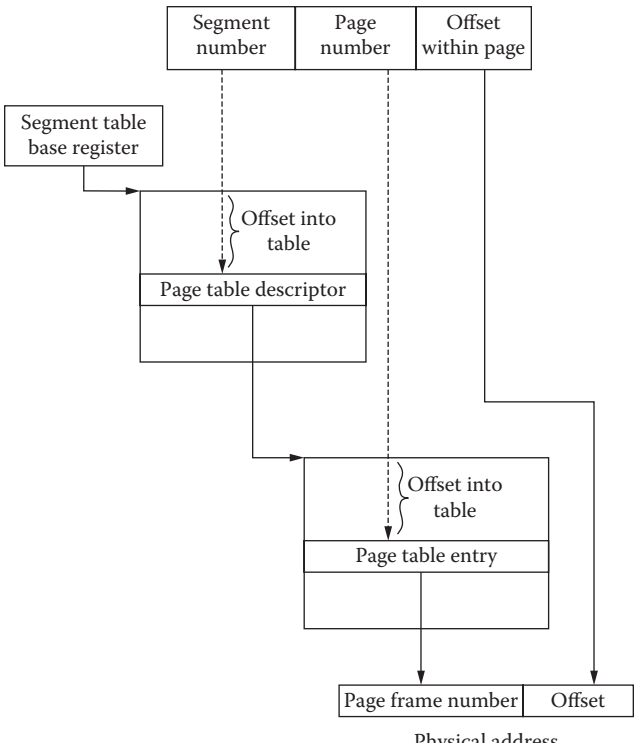


Figure 2.21 Address translation in a system using segmentation with paging.

002x021.eps

Computer Architecture Fundamentals and Principles of Computer Design 2nd edition D