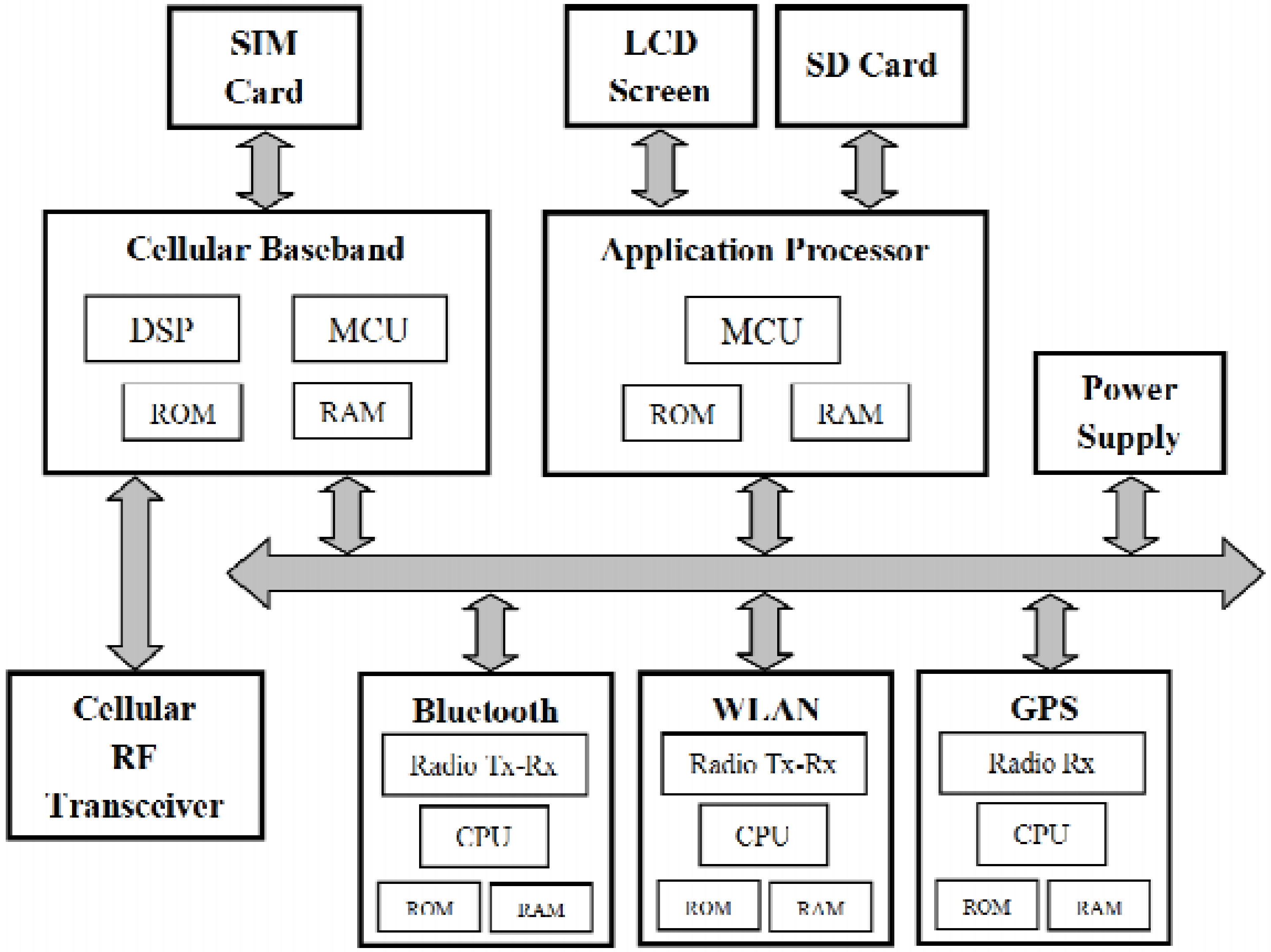
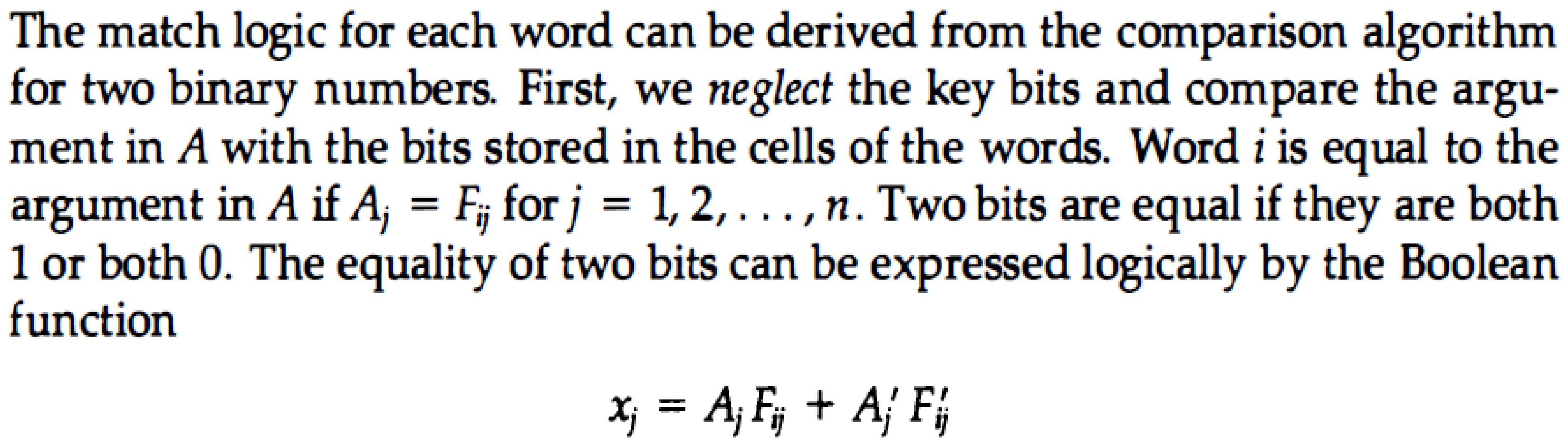
Basic Smartphone architecture



Associative Memory

❑ Match Logic



The match logic for each word can be derived from the comparison algorithm for two binary numbers. First, we *neglect* the key bits and compare the argument in A with the bits stored in the cells of the words. Word i is equal to the argument in A if $A_j = F_{ij}$ for $j = 1, 2, \ldots, n$. Two bits are equal if they are both 1 or both 0. The equality of two bits can be expressed logically by the Boolean function

$$x_j = A_j F_{ij} + A_j' F_{ij}'$$

It consists of a flip-flop storage element Fij

Associative Memory

- where $x_j = 1$ if the pair of bits in position j are equal; otherwise, $x_j = 0$.
  For a word i to be equal to the argument in A we must have all x_j variables equal to 1. This is the condition for setting the corresponding match bit M_i to 1. The Boolean function for this condition is

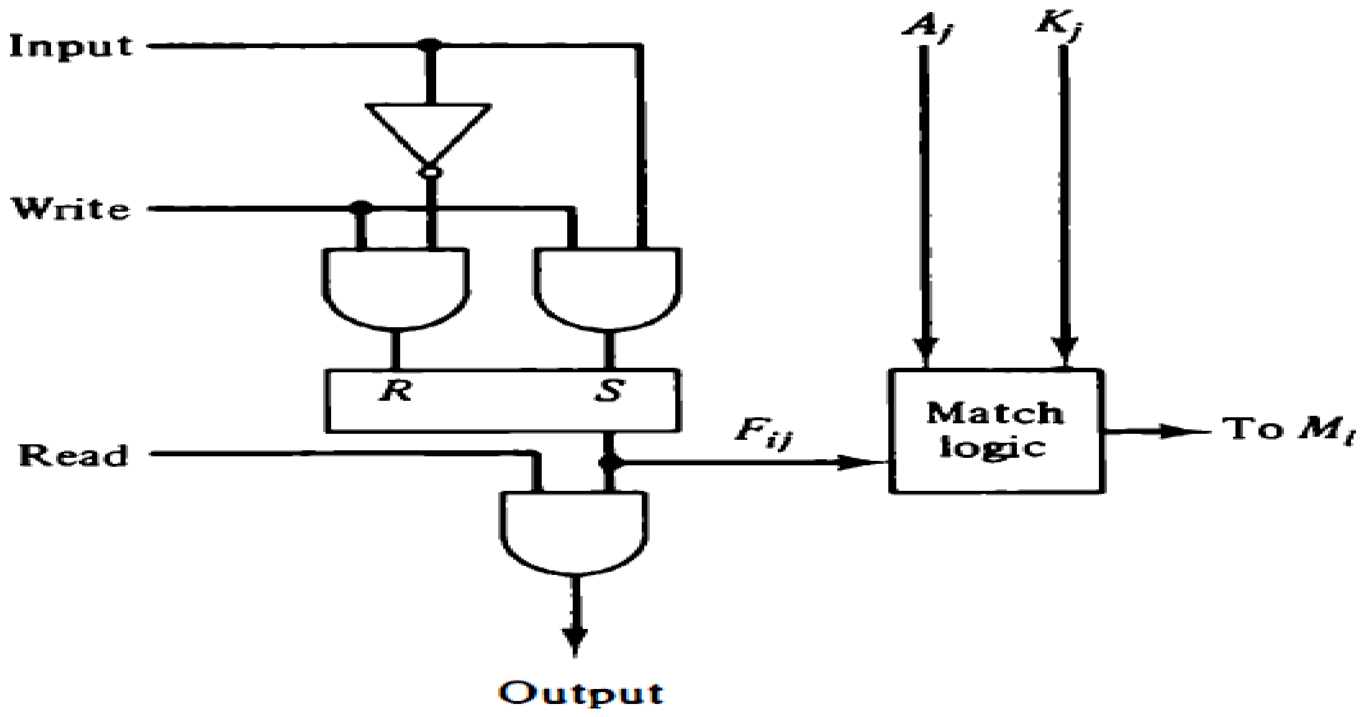
$$M_i = x_1 x_2 x_3 \cdots x_n$$

and constitutes the AND operation of all pairs of matched bits in a word.

Associative Memory

❑

Figure 12-8 One cell of associative memory.



Associative Memory

- We now include the key bit Kj in the comparison logic.

$$M_i = \prod_{j=1}^{n} (A_j F_{ij} + A_j' F_{ij}' + K_j')$$

where Π is a product symbol designating the AND operation of all n terms. We need m such functions, one for each word $i = 1, 2, 3, \ldots, m$.

Associative Memory

- ❑ The circuit for matching one word is shown in Fig. 12-9. Each cell requires two AND gates and one OR gate.

- ❑ The inverters for Aj and Kj are needed once for each column and are used for all bits in the column.

- ❑ The output of all OR gates in the cells of the same word go to the input of a common AND gate to generate the match signal for Mi .
- ❑Mi will be logic 1 if a match occurs and 0 if no match occurs.

Associative Memory

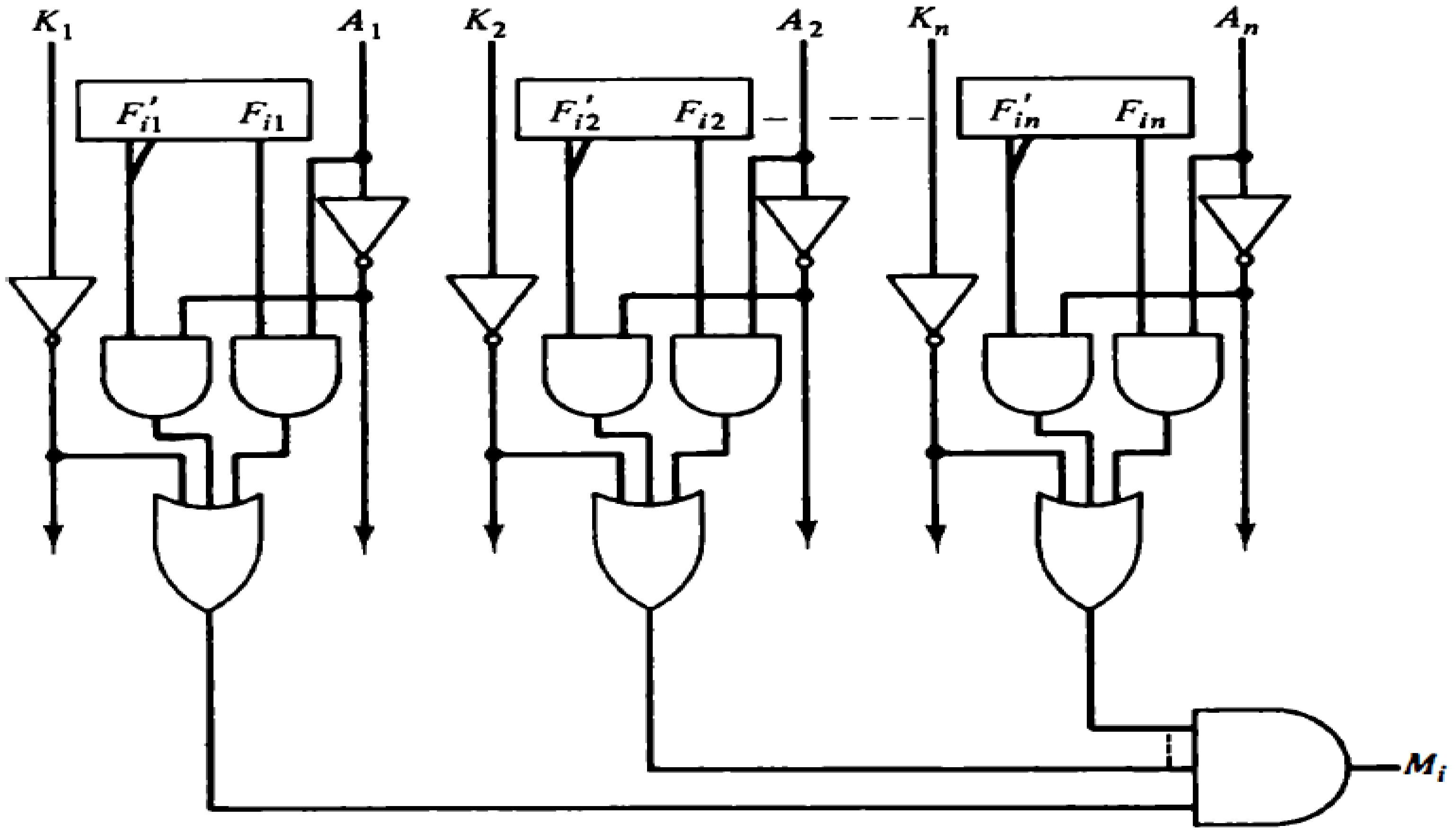


Figure 12-9 Match logic for one word of associative memory.

Direct Mapping-Cache Memory

- ❑ The possibility of using a random-access memory for the cache is investigated in Fig. 12-12.

- ❑The CPU address of 15 bits is divided into two fields.

- ❑ The nine least significant bits constitute the index field and the remaining six bits form the tag field.

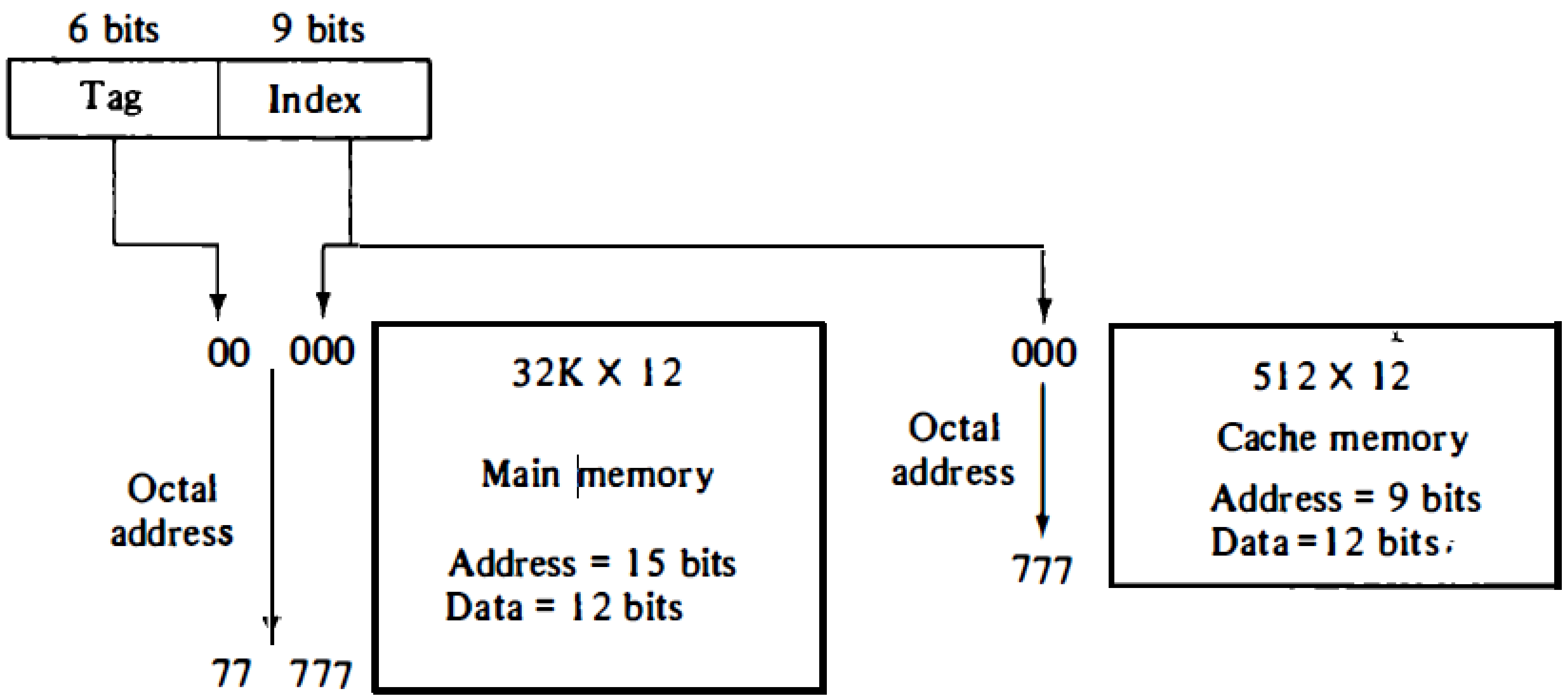
Direct Mapping-Cache Memory

- ❑ The figure shows that main memory needs an address that includes both the tag and the index bits.

- ❑The number of bits in the index field is equal to the number of address bits required to access the cache memory.

Direct Mapping-Cache Memory

Figure 12-12 Addressing relationships between main and cache memories.



Direct Mapping-Cache Memory

- In the general case, there are 2^k words in cache memory and 2^n words in main memory.

- The n-bit memory address is divided into two fields: k bits for the index field and n - k bits for the tag field.

- The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

Direct Mapping-Cache Memory

- The internal organization of the words in the cache memory is as shown in Fig. 12-13(b).

- Each word in cache consists of the data word and its associated tag.

- When a new word is first brought into the cache, the tag bits are stored alongside the data bits.

- When the CPU generates a memory request, the index field is used for the address to access the cache.

Direct Mapping-Cache Memory

- The tag field of the CPU address is compared with the tag in the word read from the cache.
- If the two tags match, there is a hit and the desired data word is in cache.
- If there is no match, there is a miss and the required word is read from main memory.
- It is then stored in the cache together with the new tag, replacing the previous value.

Direct Mapping-Cache Memory

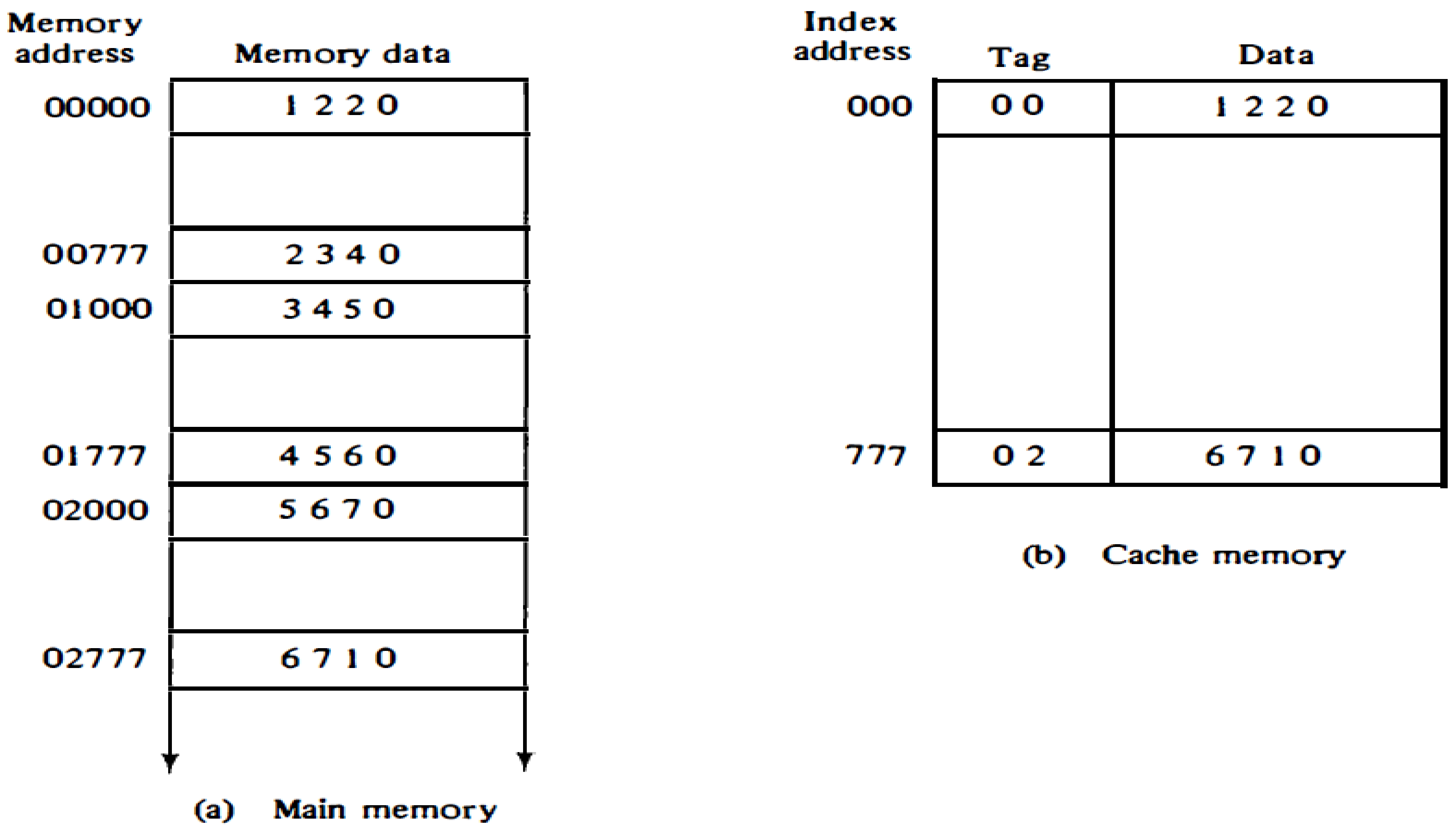


Figure 12-13 Direct mapping cache organization.

Set-Associative Mapping

- A third type of cache organization, called set-associative mapping-in that each word of cache can store two or more words of memory under the same index address.

- Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set.

Set-Associative Mapping

- An example of a set-associative cache organization for a set size of two is shown in Fig. 12-15.

- Each index address refers to two data words and their associated tags.

- Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits.

Set-Associative Mapping

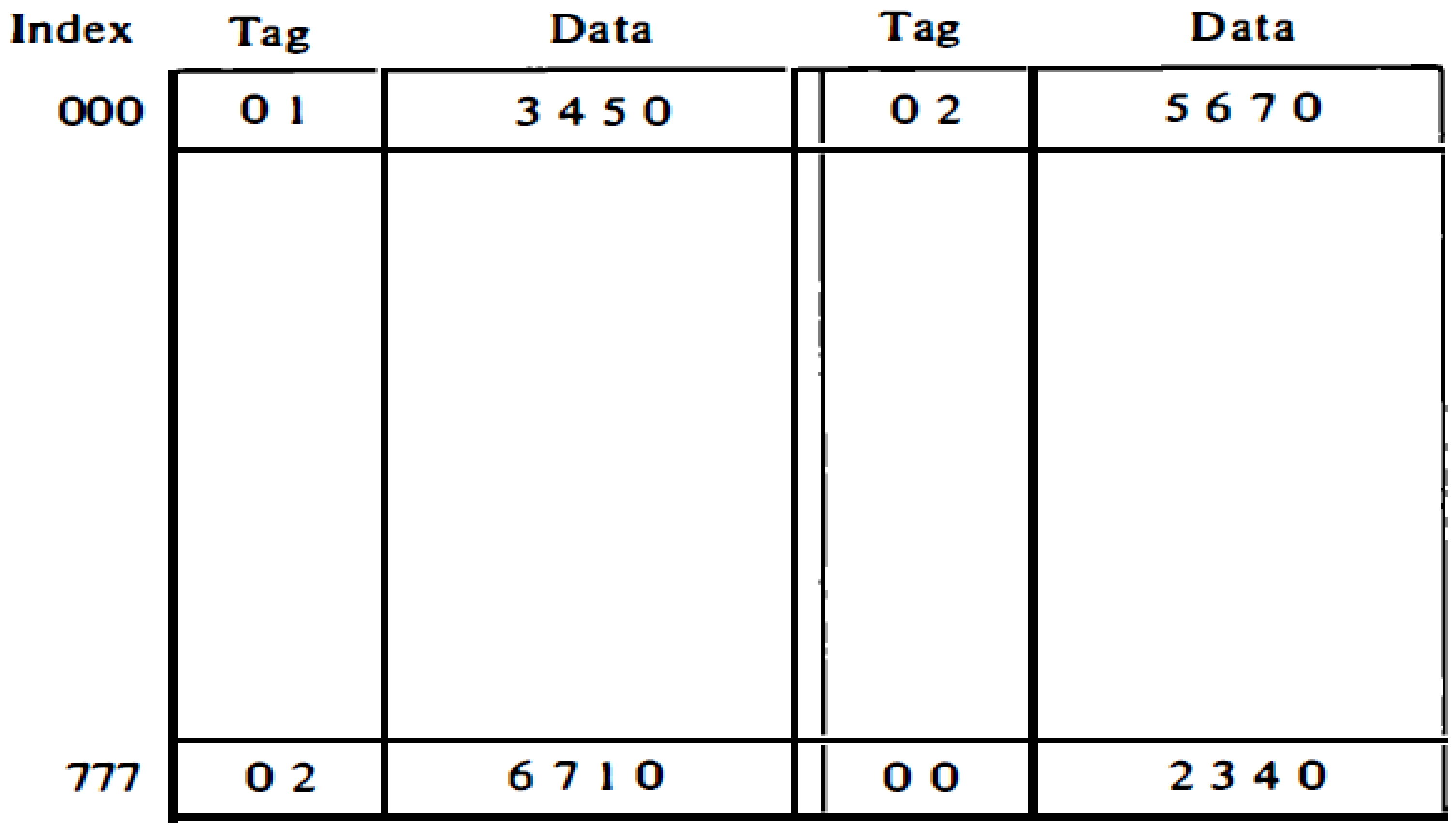
- An index address of nine bits can accommodate 512 words.

- Thus the size of cache memory is 512 x 36.

- It can accommodate 1024 words of main memory since each word of cache contains two data words.

- In general, a set-associative cache of set size k will accommodate k words of main memory in each word of cache.

Set-Associative Mapping



Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

Figure 12-15 Two-way set-associative mapping cache.

Associative Memory

- ❑ The octal numbers listed in Fig. 12-15.

- ❑The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000.

- ❑ Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777.

Associative Memory

- ❑ When the CPU generates a memory request, the index value of the address is used to access the cache.

- ❑The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs.

- ❑The comparison logic is done by an associative search of the tags in the set similar to an associative memory search: thus the name "set-associative."

Replacement Algorithms

- ❑When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value.

- ❑ The most common replacement algorithms used are:

- ❑random replacement, first-in, firstout (FIFO), and least recently used (LRU).

Replacement Algorithms

- With the random replacement policy the control chooses one tag-data item for replacement at random.

- The FIFO procedure selects for replacement the item that has been in the set the longest.

- The LRU algorithm selects for replacement the item that has been least recently used by the CPU.

Writing into Cache

- ❑ The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel if it contains the word at the specified address. This is called the write-through method.

- ❑This method has the advantage that main memory always contains the same data as the cache.

Writing into Cache

- This characteristic is important in systems with direct memory access transfers.

- It ensures that the data residing in main memory are valid at all times so that an I/O device communicating through DMA would receive the most recent updated data.

- The second procedure is called the write-back method.

Writing into Cache

- In this method only the cache location is updated during a write operation.

- The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory.

Writing into Cache

- The reason for the write-back method is that during the time a word resides in the cache, it may be updated several times; however, as long as the word remains in the cache,

- it does not matter whether the copy in main memory is out of date, since requests from the word are filled from the cache.

- It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory.

Writing into Cache

- **Cache Initialization**

- The cache is initialized when power is applied to the computer or when the main memory is loaded with a complete set of programs from auxiliary memory.

- After initialization the cache is considered to be empty, but in effect it contains some nonvalid data.

- It is customary to include with each word in cache a valid bit to indicate whether or not the word contains valid data.

Writing into Cache

- The cache is initialized by clearing all the valid bits to 0.

- The valid bit of a particular cache word is set to 1 the first time this word is loaded from main memory and stays set unless the cache has to be initialized again.

- The introduction of the valid bit means that a word in cache is not replaced by another word unless the valid bit is set to 1 and a mismatch of tags occurs.

- If the valid bit happens to be 0, the new word automatically replaces the invalid data.

Writing into Cache

- Thus the initialization condition has the effect of forcing misses from the cache until it fills with valid data.

Address Mapping Using Pages

- The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each.
- The term page refers to groups of address space of the same size.
- Page refers to the organization of address space, while a block refers to the organization of memory space.

Address Mapping Using Pages

- The programs are also considered to be split into pages.

- Portions of programs are moved from auxiliary memory to main memory in records equal to the size of a page.

- The term "page frame" is sometimes used to denote a block.

Address Mapping Using Pages

- ❑ Consider a computer with an address space of 8K and a memory space of 4K.

- ❑ If we split each into groups of 1K words we obtain eight pages and four blocks as shown in Fig. 12-18.

- ❑ At any given time, up to four pages of address space may reside in main memory in any one of the four blocks.

Address Mapping Using Pages

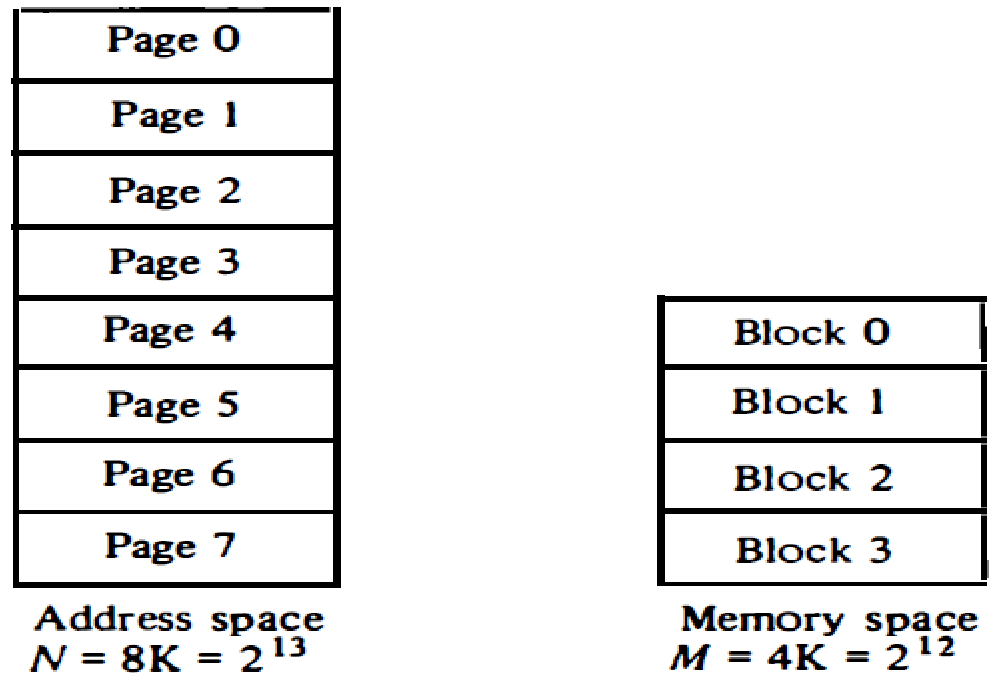


Figure 12-18 Address space and memory space split into groups of 1K words.

Address Mapping Using Pages

- The organization of the memory mapping table in a paged system is shown in Fig. 12-19.

- The memory-page table consists of eight words, one for each page. The address in the page table denotes the page number and the content of the word gives the block number where that page is stored in main memory.

Address Mapping Using Pages

- The table shows that pages 1, 2, 5, and 6 are now available in main memory in blocks 3, 0, 1, and 2, respectively.

- A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory.

Address Mapping Using Pages

- A 0 in the presence bit indicates that this page is not available in main memory. The CPU references a word in memory with a virtual address of 13 bits.

- The three high-order bits of the virtual address specify a page number and also an address for the memory-page table.

- The content of the word in the memory page table at the page number address is read out into the memory table buffer register.

- If the presence bit is a 1, the block number thus read is transferred to the two high-order bits of the main memory address register.

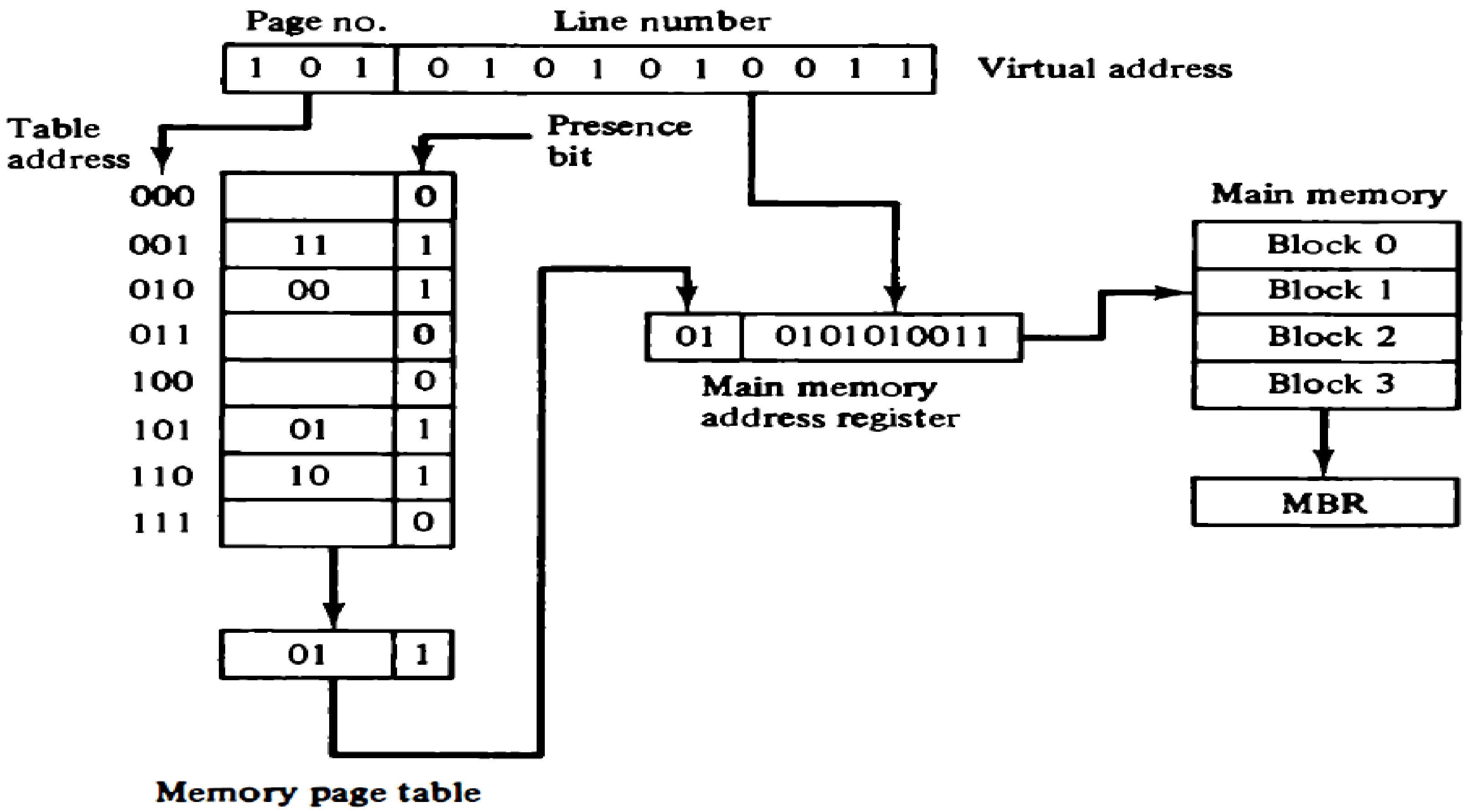
Address Mapping Using Pages

- ❑ The line number from the virtual address is transferred into the 10 low-order bits of the memory address register.
- ❑ A read signal to main memory transfers the content of the word to the main memory buffer register ready to be used by the CPU. If the presence bit in the word read from the page table is 0, it signifies that the content of the word referenced by the virtual address does not reside in main memory.
- ❑A call to the operating system is then generated to fetch the required page from auxiliary memory and place it into main memory before resuming computation.

Address Mapping Using Pages

- The

Figure 12-19 Memory table in a paged system.



Associative Memory Page Table

- In Fig. 12-20. Each entry in the associative memory array consists of two fields.
- The first three bits specify a field for storing the page number.
- The last two bits constitute a field for storing the block number. The virtual address is placed in the argument register.
- The page number bits in the argument register are compared with all page numbers in the page field of the associative memory. If the page number is found, the 5-bit word is read out from memory.

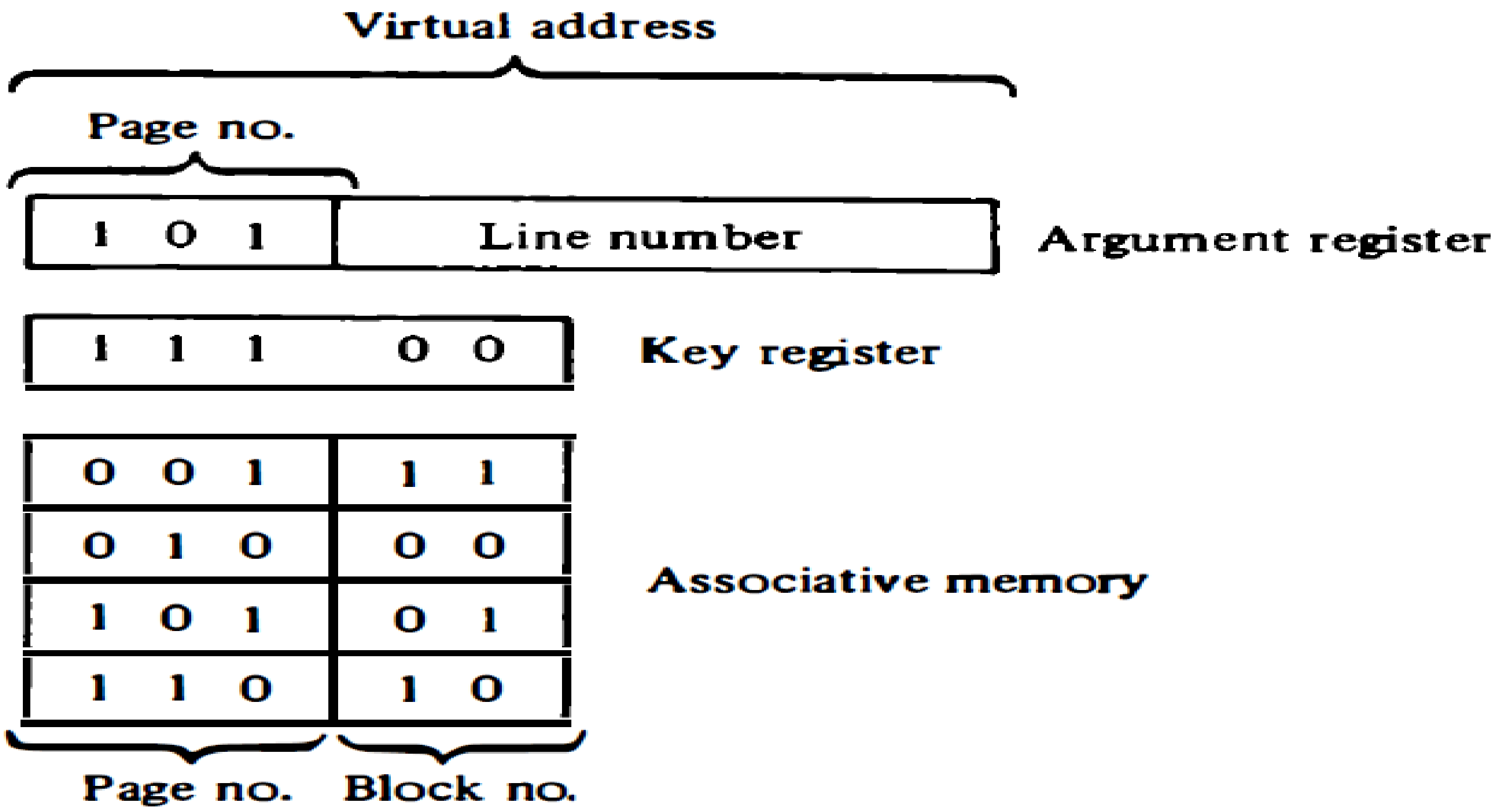
Associative Memory Page Table

- The corresponding block number, being in the same word, is transferred to the main memory address register.

- If no match occurs, a call to the operating system is generated to bring the required page from auxiliary memory.

Associative Memory Page Table

Figure 12-20 An associative memory page table.



Page Replacement

- A virtual memory system is a combination of hardware and software techniques.

- The memory management software system handles all the software operations for the efficient utilization of memory space.

- It must decide (1) which page in main memory ought to be removed to make room for a new page, (2) when a new page is to be transferred from auxiliary memory to main memory, and (3) where the page is to be placed in main memory.

Page Replacement

- When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position.

- The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called page fault.

- When page fault occurs, the execution of the present program is suspended until the required page is brought into main memory.

Page Replacement

- Since loading a page from auxiliary memory to main memory is basically an I/O operation, the operating system assigns this task to the I/O processor.

- In the meantime, control is transferred to the next program in memory that is waiting to be processed in the CPU.

- Later, when the memory block has been assigned and the transfer completed, the original program can resume its operation.

Page Replacement

- When a page fault occurs in a virtual memory system, it signifies that the page referenced by the CPU is not in main memory.

- A new page is then transferred from auxiliary memory to main memory.

Page Replacement

- If main memory is full, it would be necessary to remove a page from a memory block to make room for the new page.

- The policy for choosing pages to remove is determined from the replacement algorithm that is used.

- The goal of a replacement policy is to try to remove the page least likely to be referenced in the immediate future.

Associative Memory Page Table

- Two of the most common replacement algorithms used are the first-in first-out (FIFO) and the least recently used (LRU).

- The FIFO algorithm selects for replacement the page that has been in memory the longest time. Each time a page is loaded into memory, its identification number is pushed into a FIFO stack.

- FIFO will be full whenever memory has no more empty blocks.

- When a new page must be loaded, the page least recently brought in is removed. The page to be removed is easily determined because its identification number is at the top of the FIFO stack.

least recently used (LRU)

- The LRU policy is more difficult to implement but has been more attractive on the assumption that the least recently used page is a better candidate for removal than the least recently loaded page.

- The LRU algorithm can be implemented by associating a counter with every page that is in main memory.

- When a page is referenced, its associated counter is set to zero. At fixed intervals of time, the counters associated with all pages presently in memory are incremented by 1.

least recently used (LRU)

- The least recently used page is the page with the highest count.

- The counters are often called aging registers, as their count indicates their age, that is, how long ago their associated pages have been referenced.

Discussions

- ❑ Discuss FIFO buffer ?

First-In, First-Out Buffer

- ❑A first-in, first-out (FIFO) buffer is a memory unit that stores information in such a manner that the item first in is the item first out.

- ❑FIFO buffer comes with separate input and output terminals.

- ❑The important feature of this buffer is that it can input data and output data at two different rates and the output data are always in the same order in which the data entered the buffer.

First-In, First-Out Buffer

- ❑When placed between two units, the FIFO can accept data from the source unit at one rate of transfer and deliver the data to the destination unit at another rate.

- ❑If the source unit is slower than the destination unit, the buffer can be filled with data at a slow rate and later emptied at the higher rate.

First-In, First-Out Buffer

- Thus a FIFO buffer can be useful in some applications when data are transferred asynchronously.

- It piles up data as they come in and gives them away in the same order when the data are needed.

Discussions

- Discuss Data transfer to and from peripherals may be handled ?
- Programmed I/O ?

Modes of Transfer

- ❑**Programmed I/O** operations are the result of I/O instructions written in the computer program.

- ❑ Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral.

- ❑Other instructions are needed to transfer the data to and from CPU and memory.

Modes of Transfer

- Transferring data under program control requires constant monitoring of the peripheral by the CPU.

- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

Modes of Transfer

- ❑It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.

- ❑In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer.

- ❑This is a time-consuming process since it keeps the processor busy needlessly.

Modes of Transfer

- ❑It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

- ❑In the meantime the CPU can proceed to execute another program.

- ❑ The interface meanwhile keeps monitoring the device.

Modes of Transfer

- ❑When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.

- ❑Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing.

- ❑Transfer of data under programmed I/O is between CPU and peripheral.

Discussions

- ❑ Discuss Daisy-Chaining Method?

Daisy-Chaining Method

- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.

- The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain.

Daisy-Chaining Priority

- This method of connection between three devices and the CPU is shown in Fig. 11-12.

- The interrupt request line is common to all devices and forms a wired logic connection.

- If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU.

Daisy-Chaining Priority

- ❑When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU.

- ❑The CPU responds to an interrupt request by enabling the interrupt acknowledge line.

- ❑This signal is received by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt

Daisy-Chaining Priority

- ❑If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.
- ❑It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.

Daisy-Chaining Priority

- ❑A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked.

- ❑A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output.

- ❑If the device does not have pending interrupts, it transmits the acknowledge signal to the next device by placing a 1 in its PO output.

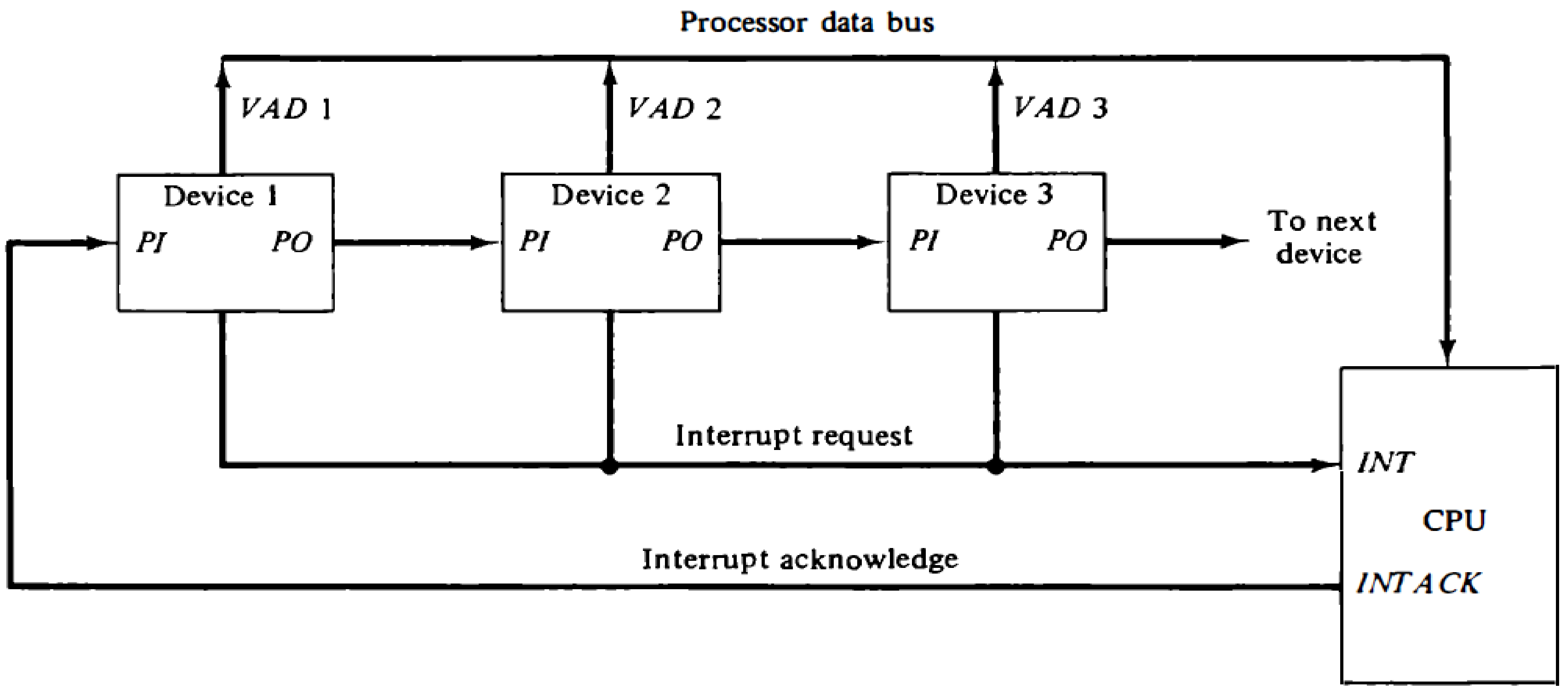
Daisy-Chaining Priority

- Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus.

- The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU.

- The farther the device is from the first position, the lower is its priority.

Daisy-Chaining Priority



Discussions

- Discuss peripherals?

Input-Output Organization

- ❖ Devices that are under the direct control of the computer are said to be connected on-line.

- ❖ These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be part of the total computer system.

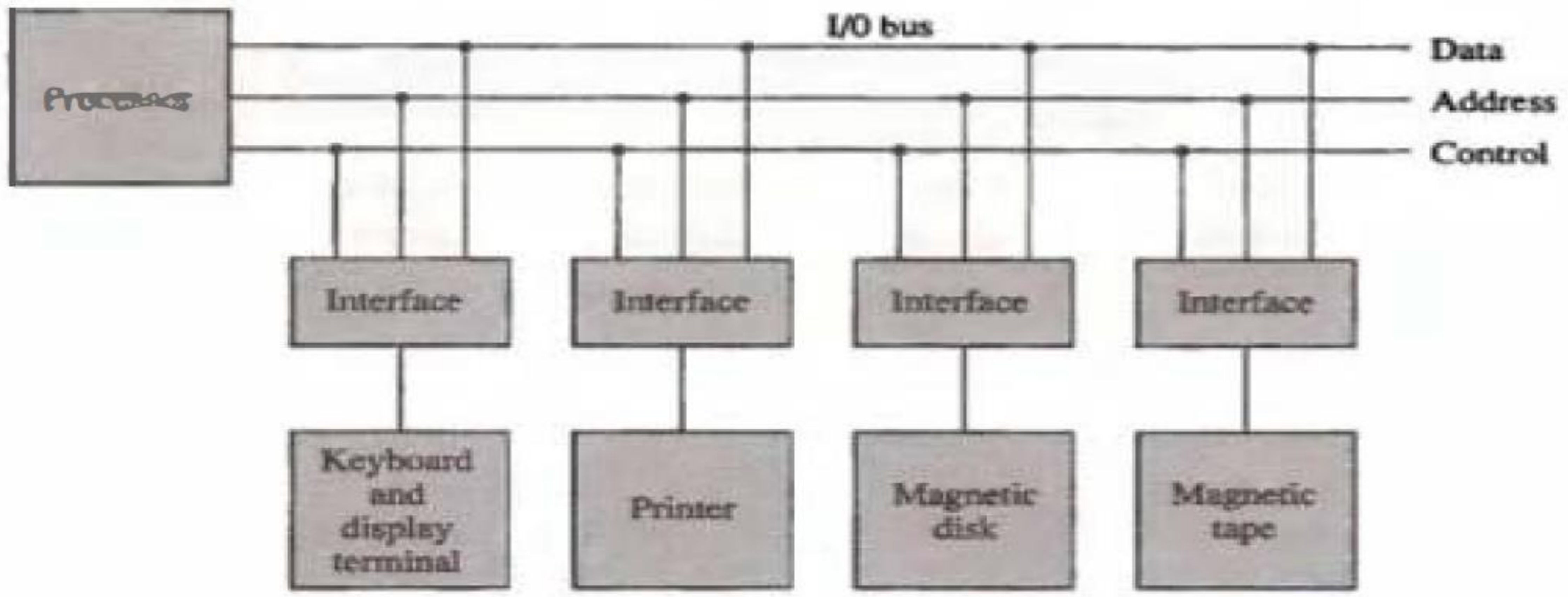
- ❖ Input or output devices attached to the computer are also called **peripherals.**

Discussions

- ❑ Discuss connections of I/O bus with input output devices ?

I/O Bus and Interface Modules

Figure 11.1 Connection of I/O bus to input-output devices


I/O bus
Processor
Data
Address
Control
Interface
Interface
Interface
Interface
Keyboard and display terminal
Printer
Magnetic disk
Magnetic tape