

Memory Unit

- ❑ In a multiprogramming system, when one program is waiting for input or output transfer, there is another program ready to utilize the CPU.
- ❑ The part of the computer system that supervises the flow of information between auxiliary memory and main memory is called the memory management system.

Main Memory

- ❑ The main memory is the central storage unit in a computer system.
- ❑ It is a relatively large and fast memory used to store programs and data during the computer operation.
- ❑ The principal technology used for the main memory is based on semiconductor integrated circuits.

Main Memory

- ❑ Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.
- ❑ The static RAM consists essentially of internal flip-flops that store the binary information.
- ❑ The stored information remains valid as long as power is applied to the unit.

Main Memory

- ❑ The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.
- ❑ The capacitors are provided inside the chip by MOS transistors.
- ❑ The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.

Main Memory

- ❑ Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge.
- ❑ The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.
- ❑ The static RAM is easier to use and has shorter read and write cycles.

Main Memory

- ❑ Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.
- ❑ Originally, RAM was used to refer to a random-access memory, but now it is used to designate a read/write memory to distinguish it from a read-only memory, although ROM is also random access.

Main Memory

- ❑ RAM is used for storing the bulk of the programs and data that are subject to change.
- ❑ ROM is used for storing programs that are permanently resident in the computer and for tables of constants that do not change in value once the production of the computer is completed.

Main Memory

- ❑ ROM portion of main memory is needed for storing an initial program called a bootstrap loader.
- ❑ The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.

Main Memory

- ❑ Since RAM is volatile, its contents are destroyed when power is turned off.
- ❑ The contents of ROM remain unchanged after power is turned off and on again.
- ❑ The startup of a computer consists of turning the power on and starting the execution of an initial program.

Main Memory

- ❑ Thus when power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.
- ❑ The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the operating system, which prepares the computer for general use.

Main Memory

- ❑ RAM and ROM chips are available in a variety of sizes.
- ❑ If the memory needed for the computer is larger than the capacity of one chip, it is necessary to combine a number of chips to form the required memory size.

Main Memory

❑ RAM and ROM Chips

- ❑ A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip only when needed.
- ❑ Another common feature is a bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation, or from CPU to memory during a write operation.

Main Memory

- ❑ A bidirectional bus can be constructed with three-state buffers. A three-state buffer output can be placed in one of three possible states:
- ❑ a signal equivalent to logic 1, a signal equivalent to logic 0, or a high impedance state.
- ❑ The logic 1 and 0 are normal digital signals. The high impedance state behaves like an open circuit, which means that the output does not carry a signal and has no logic significance.

Main Memory

- ❑ The block diagram of a RAM chip is shown in Fig. 12-2. The capacity of the memory is 128 words of eight bits (one byte) per word.
- ❑ This requires a 7-bit address and an 8-bit bidirectional data bus.
- ❑ The read and write inputs specify the memory operation and the two chips select (CS) control inputs are for enabling the chip only when it is selected by the microprocessor.

Main Memory

- ❑ The availability of more than one control input to select the chip facilitates the decoding of the address lines when multiple chips are used in the microcomputer.
- ❑ The read and write inputs are sometimes combined into one line labeled R/W.
- ❑ When the chip is selected, the two binary states in this line specify the two operation of read or write.

Main Memory

- ❑ The function table listed in Fig. 12-2(b) specifies the operation of the RAM chip.
- ❑ The unit is in operation only when $CS1 = 1$ and $CS2(\text{bar}) = 0$.
- ❑ The bar on TOP of the second select variable indicates that this input is enabled when it is equal to 0.

Main Memory

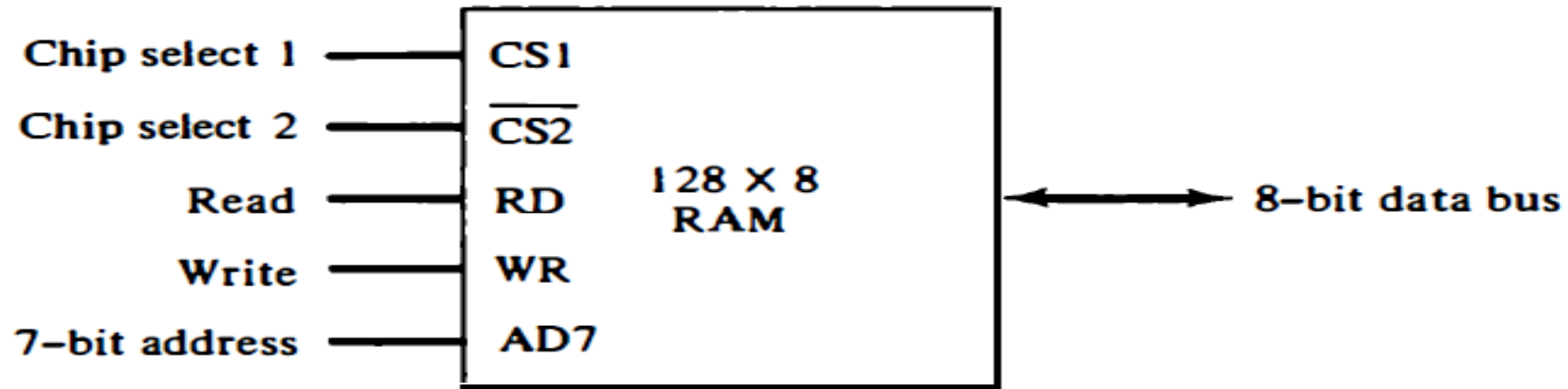
- ❑ If the chip select inputs are not enabled, or if they are enabled but the read or write inputs are not enabled, the memory is inhibited and its data bus is in a high-impedance state.
- ❑ When $CS1 = 1$ and $CS2(\text{bar}) = 0$, the memory can be placed in a write or read mode.
- ❑ When the WR input is enabled, the memory stores a byte from the data bus into a location specified by the address input lines.

Main Memory

- ❑ When the RD input is enabled, the content of the selected byte is placed into the data bus.
- ❑ The RD and WR signals control the memory operation as well as the bus buffers associated with the bidirectional data bus .

Main Memory

Figure 12-2 Typical RAM chip.



(a) Block diagram

CS1	$\overline{\text{CS2}}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

(b) Function table

Main Memory

- ❑ A ROM chip is organized externally in a similar manner.
- ❑ However, since a ROM can only read, the data bus can only be in an output mode.
- ❑ The block diagram of a ROM chip is shown in Fig. 12-3.

Main Memory

- ❑ For the same-size chip, it is possible to have more bits of ROM than of RAM, because the internal binary cells in ROM occupy less space than in RAM.
- ❑ For this reason, the diagram specifies a 512-byte ROM, while the RAM has only 128 bytes.
- ❑ The nine address lines in the ROM chip specify any one of the 512 bytes stored in it.
- ❑ The two chip select inputs must be $CS1 = 1$ and $CS2 (\text{bar}) = 0$ for the unit to operate.

Main Memory

- ❑ Otherwise, the data bus is in a high-impedance state.
- ❑ There is no need for a read or write control because the unit can only read.
- ❑ Thus when the chip is enabled by the two select inputs, the byte selected by the address lines appears on the data bus.

Main Memory

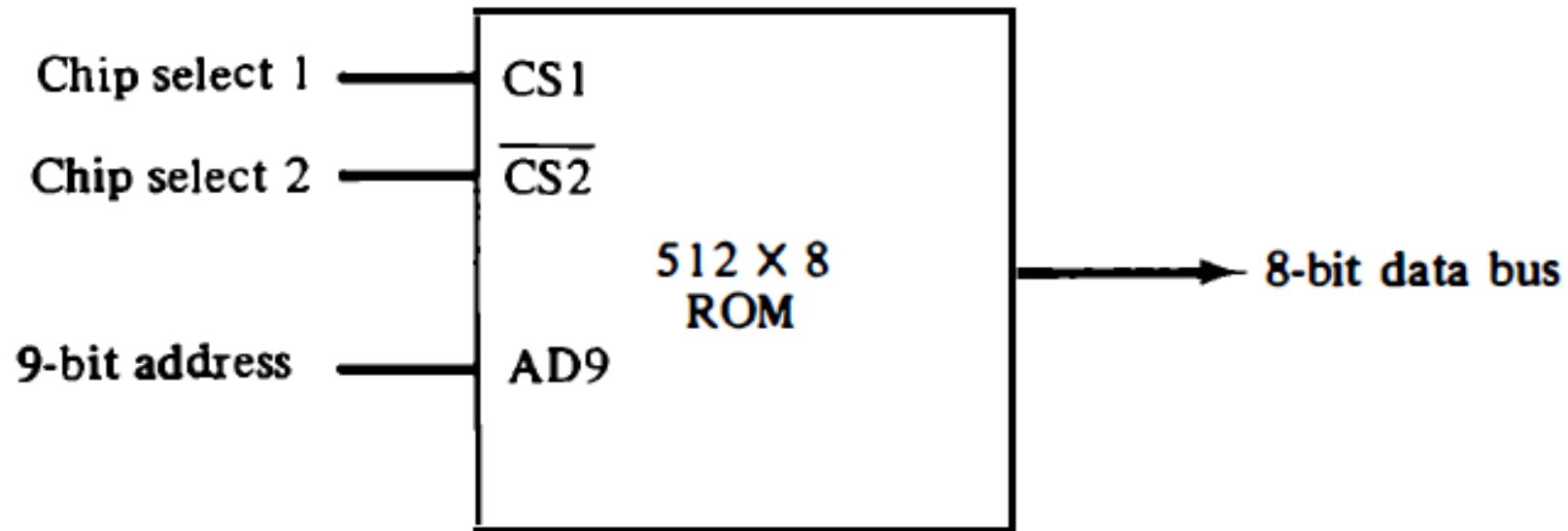


Figure 12-3 Typical ROM chip.

Memory Address Map

- ❑ The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM.
- ❑ The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available.

Memory Address Map

- ❑ The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip.
- ❑ The table, called a **memory address map**, is a pictorial representation of assigned address space for each chip in the system.

Memory Address Map

- ❑ To demonstrate with a particular example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.
- ❑ The memory address map for this configuration is shown in Table 12- 1 .

Memory Address Map

- ❑ The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip.
- ❑ The address bus lines are listed in the third column.
- ❑ Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero.

Memory Address Map

- ❑ The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.
- ❑ The RAM chips have 128 bytes and need seven address lines.
- ❑ The ROM chip has 512 bytes and needs 9 address lines.

Memory Address Map

- ❑ The x's are always assigned to the low-order bus lines: lines 1 through 7 for the RAM and lines 1 through 9 for the ROM.
- ❑ It is now necessary to distinguish between four RAM chips by assigning to each a different address.
- ❑ For this particular example we choose bus lines 8 and 9 to represent four distinct binary combinations.

Memory Address Map

- ❑ The table clearly shows that the nine low-order bus lines constitute a memory space for RAM equal to $2^9 = 512$ bytes.
- ❑ The distinction between a RAM and ROM address is done with another bus line.
- ❑ Here we choose line 10 for this purpose.

Memory Address Map

- ❑ When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it selects the ROM.
- ❑ The equivalent hexadecimal address for each chip is obtained from the information under the address bus assignment.
- ❑ The address bus lines are subdivided into groups of four bits each so that each group can be represented with a hexadecimal digit.

Memory Address Map

- ❑ The first hexadecimal digit represents lines 13 to 16 and is always 0.
- ❑ The next hexadecimal digit represents lines 9 to 12, but lines 11 and 12 are always 0.
- ❑ The range of hexadecimal addresses for each component is determined from the x's associated with it.
- ❑ These x's represent a binary number that can range from an all-0's to an all-1's value.

Memory Address Map

TABLE 12-1 Memory Address Map for Microcomputer

[illegible]

Memory Connection to CPU

- ❑ RAM and ROM chips are connected to a CPU through the data and address buses.
- ❑ Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes.
- ❑ The particular RAM chip selected is determined from lines 8 and 9 in the address bus .

Memory Connection to CPU

- ❑ This is done through a 2 x 4 decoder whose outputs go to the CS1 inputs in each RAM chip.
- ❑ Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is selected, and so on.
- ❑ The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip.

Memory Connection to CPU

- ❑ The selection between RAM and ROM is achieved through bus line 10.
- ❑ The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1.

Memory Connection to CPU

- ❑ The other chip select input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a read operation.
- ❑ Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder.
- ❑ This assigns addresses 0 to 511 to RAM and 512 to 1023 to ROM.
- ❑ The data bus of the ROM has only an output capability, whereas the data bus connected to the RAMs can transfer information in both directions .

Memory Connection to CPU

- ❑ The example just shown gives an indication of the interconnection complexity that can exist between memory chips and the CPU.

Memory Connection to CPU

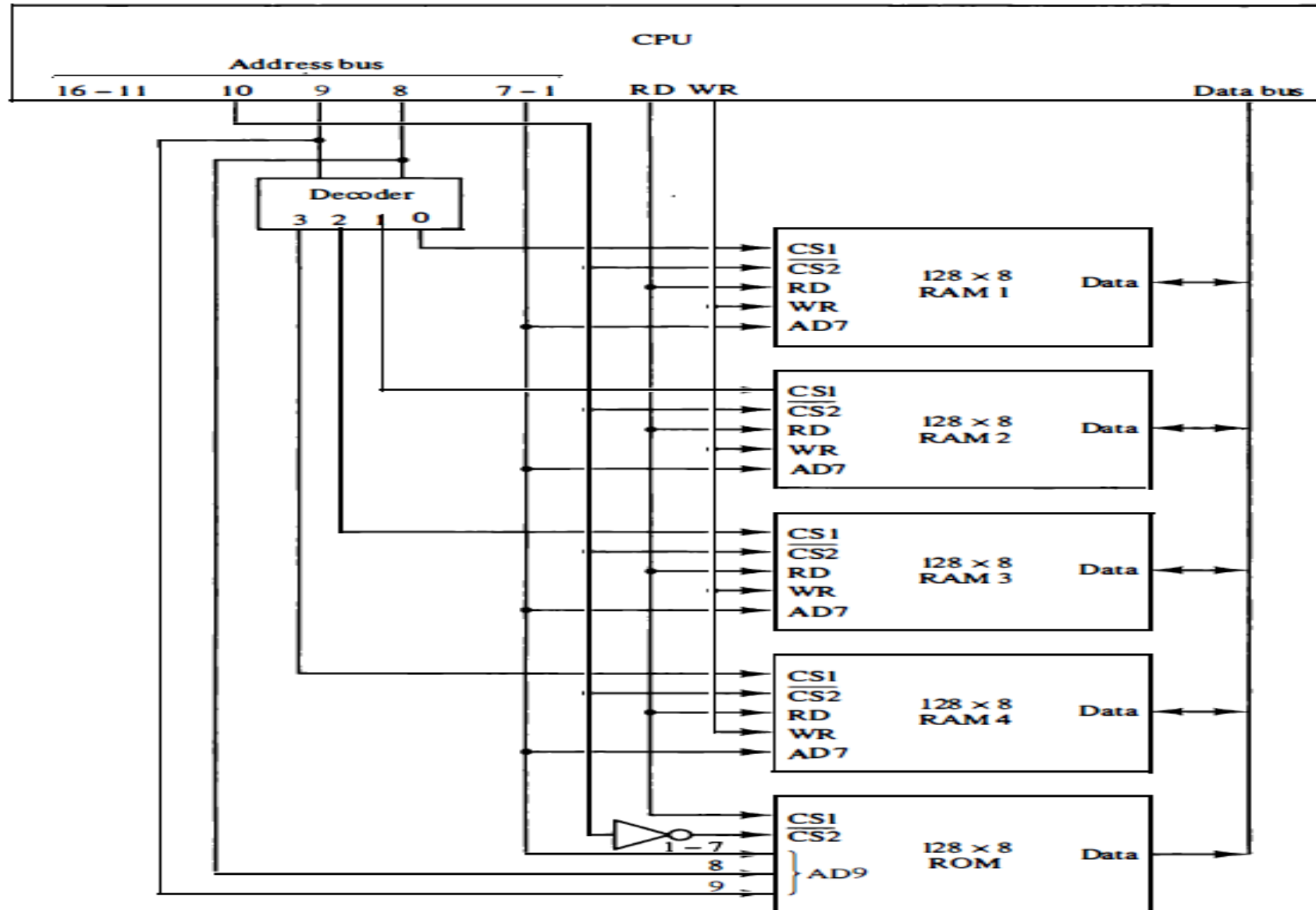


Figure 12-4 Memory connection to the CPU.

Auxiliary Memory

- ❑ The most common auxiliary memory devices used in computer systems are magnetic disks and tapes.
- ❑ The average time required to reach a storage location in memory and obtain its contents is called the **access time**.

Auxiliary Memory

- ❑ In electromechanical devices with moving parts such as disks and tapes, the access time consists of a seek time required to position the read-write head to a location and a transfer time required to transfer data to or from the device.
- ❑. Because the seek time is usually much longer than the transfer time, auxiliary storage is organized in records or blocks.

Auxiliary Memory

- ❑ A record is a specified number of characters or words. Reading or writing is always done on entire records.
- ❑ The transfer rate is the number of characters or words that the device can transfer per second, after it has been positioned at the beginning of the record.

Auxiliary Memory

- ❑ Magnetic drums and disks are quite similar in operation. Both consist of high-speed rotating surfaces coated with a magnetic recording medium.
- ❑ The rotating surface of the drum is a cylinder and that of the disk, a round flat plate.
- ❑ The recording surface rotates at uniform speed and is not started or stopped during access operations.

Auxiliary Memory

- ❑ Bits are recorded as magnetic spots on the surface as it passes a stationary mechanism called a write head.
- ❑ Stored bits are detected by a change in magnetic field produced by a recorded spot on the surface as it passes through a read head.

Auxiliary Memory

- ❑ The amount of surface available for recording in a disk is greater than in a drum of equal physical size.
- ❑ Therefore, more information can be stored on a disk than on a drum of comparable size.
- ❑ For this reason, disks have replaced drums in more recent computers.

Auxiliary Memory

☐ Magnetic Disks

- ☐ A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.
- ☐ Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.

Auxiliary Memory

- ❑ All disks rotate together at high speed and are not stopped or started for access purposes.
- ❑ Bits are stored in the magnetized surface in spots along concentric circles called tracks. The tracks are commonly divided into sections called sectors.
- ❑ In most systems, the minimum quantity of information which can be transferred is a sector.
- ❑ The subdivision of one disk surface into tracks and sectors is shown in Fig. 12-5.

Auxiliary Memory

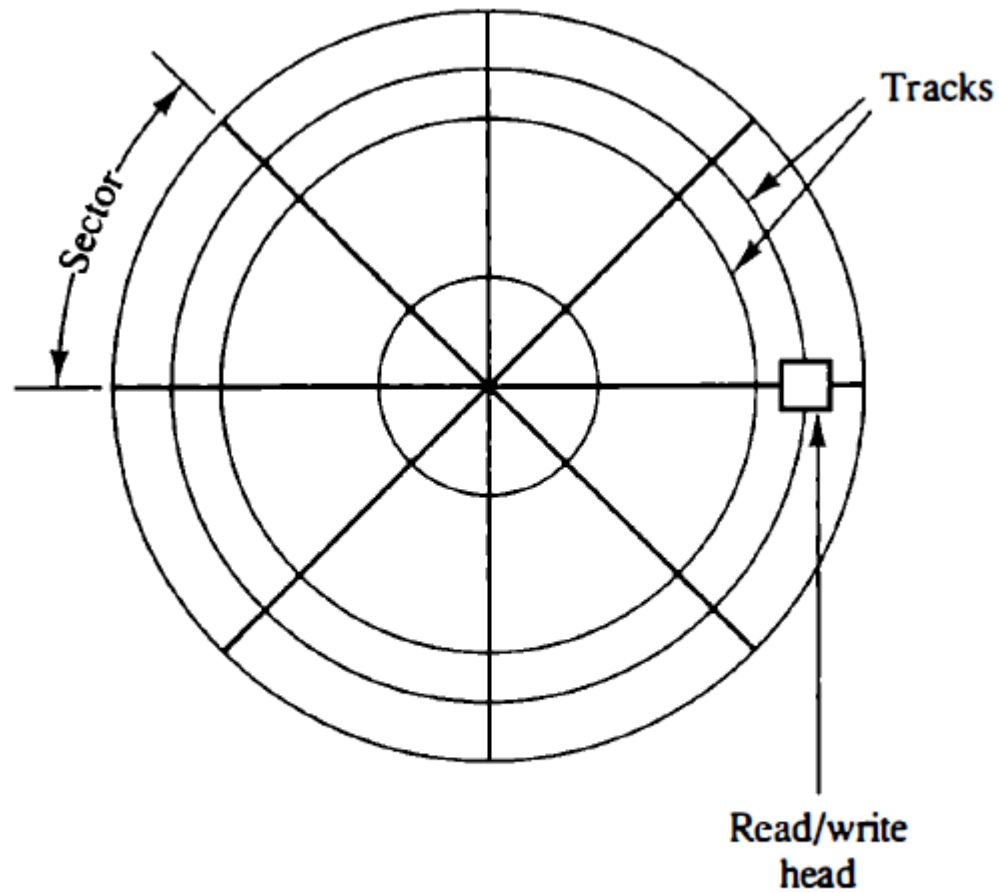


Figure 12-5 Magnetic disk.

Auxiliary Memory

- ❑ Some units use a single read/write head for each disk surface.
- ❑ In this type of unit, the track address bits are used by a mechanical assembly to move the head into the specified track position before reading or writing.
- ❑ In other disk systems, separate read/write heads are provided for each track in each surface.
- ❑ The address bits can then select a particular track electronically through a decoder circuit.

Auxiliary Memory

- ❑ Permanent timing tracks are used in disks to synchronize the bits and recognize the sectors.
- ❑ A disk system is addressed by address bits that specify the disk number, the disk surface, the sector number and the track within the sector.

Auxiliary Memory

- ❑ After the read/write heads are positioned in the specified track, the system has to wait until the rotating disk reaches the specified sector under the read/write head.
- ❑ Information transfer is very fast once the beginning of a sector has been reached.
- ❑ Disks may have multiple heads and simultaneous transfer of bits from several tracks at the same time.

Auxiliary Memory

- ☐ A track in a given sector near the circumference is longer than a track near the center of the disk.
- ☐ If bits are recorded with equal density, some tracks will contain more recorded bits than others.
- ☐ To make all the records in a sector of equal length, some disks use a variable recording density with higher density on tracks near the center than on tracks near the circumference.
- ☐ This equalizes the number of bits on all tracks of a given sector.

Auxiliary Memory

- ❑ Disks that are permanently attached to the unit assembly and cannot be removed by the occasional user are called hard disks.
- ❑ A disk drive with removable disks is called a floppy disk.
- ❑ The disks used with a floppy disk drive are small removable disks made of plastic coated with magnetic recording material.

Auxiliary Memory

- ❑ Floppy disks are extensively used in personal computers as a medium for distributing software to computer users.

Auxiliary Memory

❑ Magnetic Tape

- ❑ A magnetic tape transport consists of the electrical, mechanical, and electronic components to provide the parts and control mechanism for a magnetic-tape unit.
- ❑ The tape itself is a strip of plastic coated with a magnetic recording medium.
- ❑ Bits are recorded as magnetic spots on the tape along several tracks.

Auxiliary Memory

- ❑ Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.
- ❑ Read/write heads are mounted one in each track so that data can be recorded and read as a sequence of characters.

Auxiliary Memory

- ❑ Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewind.
- ❑ However, they cannot be started or stopped fast enough between individual characters.
- ❑ For this reason, information is recorded in blocks referred to as records.

Auxiliary Memory

- ❑ Gaps of unrecorded tape are inserted between records where the tape can be stopped.
- ❑ The tape starts moving while in a gap and attains its constant speed by the time it reaches the next record.
- ❑ Each record on tape has an identification bit pattern at the beginning and end.
- ❑ By reading the bit pattern at the beginning, the tape control identifies the record number.

Auxiliary Memory

- ❑ By reading the bit pattern at the end of the record, the control recognizes the beginning of a gap.
- ❑ A tape unit is addressed by specifying the record number and the number of characters in the record.
- ❑ Records may be of fixed or variable length.

Associative Memory

- ❑ Many data-processing applications require the search of items in a table stored in memory.
- ❑ An assembler program searches the symbol address table in order to extract the symbol's binary equivalent.
- ❑ An account number may be searched in a file to determine the holder's name and account status.

Associative Memory

- ❑ The established way to search a table is to store all items where they can be addressed in sequence.
- ❑ The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing the information read with the item being searched until a match occurs.

Associative Memory

- ❑ The number of accesses to memory depends on the location of the item and the efficiency of the search algorithm.
- ❑ Many search algorithms have been developed to minimize the number of accesses while searching for an item in a random or sequential access memory.

Associative Memory

- ❑ The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.
- ❑ A memory unit accessed by content is called an associative memory or content addressable memory (CAM).

Associative Memory

- ❑ This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.
- ❑ When a word is written in an associative memory, no address is given.
- ❑ The memory is capable of finding an empty unused location to store the word.
- ❑ When a word is to be read from an associative memory, the content of the word, or part of the word, is specified.
- ❑ The memory locates all words which match the specified content and marks them for reading.

Associative Memory

- ❑ Because of its organization, the associative memory is uniquely suited to do parallel searches by data association.
- ❑ Moreover, searches can be done on an entire word or on a specific field within a word.

Associative Memory

- ❑ An associative memory is more expensive than a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external argument.
- ❑ For this reason, associative memories are used in applications where the search time is very critical and must be very short.

Associative Memory

❑ Hardware Organization

- ❑ The block diagram of an associative memory is shown in Fig. 12-6.
- ❑ It consists of a memory array and logic for m words with n bits per word.
- ❑ The argument register A and key register K each have n bits, one for each bit of a word.

Associative Memory

- ❑ The match register M has m bits, one for each memory word.
- ❑ Each word in memory is compared in parallel with the content of the argument register.
- ❑ The words that match the bits of the argument register set a corresponding bit in the match register.

Associative Memory

- ❑ After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.
- ❑ Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

Associative Memory

- ☐ The key register provides a mask for choosing a particular field or key in the argument word.
- ☐ The entire argument is compared with each memory word if the key register contains all 1's.
- ☐ Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.

Associative Memory

- ❑ Thus the key provides a mask or identifying piece of information which specifies how the reference to memory is made.
- ❑ To illustrate with a numerical example, suppose that the argument register A and the key register K have the bit configuration shown below.
- ❑ Only the three leftmost bits of A are compared with memory words because K has 1's in these positions

Associative Memory

- Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

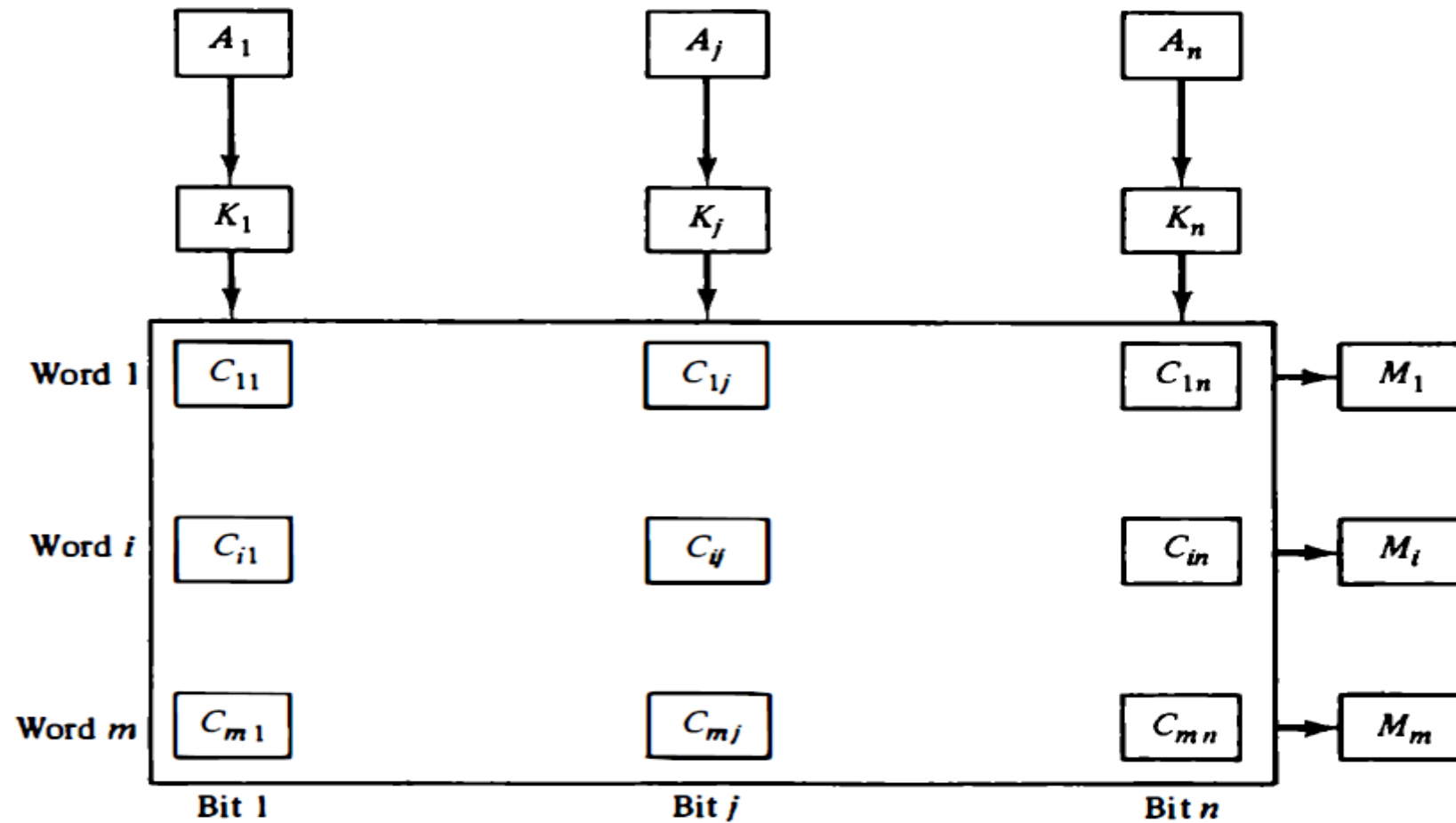
<i>A</i>	101 111100	
<i>K</i>	111 000000	
Word 1	100 111100	no match
Word 2	101 000001	match

Associative Memory

┌ The relation between the memory array and external registers in an associative memory is shown in Fig. 12-7. The cells in the array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. Thus cell C_{ij} is the cell for bit j in word i . A bit A_j in the argument register is compared with all the bits in column j of the array provided that $K_j = 1$. This is done for all columns $j = 1, 2, \dots, n$. If a match occurs between all the unmasked bits of the argument and the bits in word i , the corresponding bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

Associative Memory

Figure 12-7 Associative memory of m word, n cells per word.



Cache Memory

- ❑ Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory.
- ❑ This phenomenon is known as the property of locality of reference .
- ❑ The reason for this property may be understood considering that a typical computer program flows in a straight-line fashion with program loops and subroutine calls encountered frequently.

Cache Memory

- ❑ When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop.
- ❑ Every time a given subroutine is called, its set of instructions are fetched from memory.
- ❑ Thus loops and subroutines tend to localize the references to memory for fetching instructions.

Cache Memory

- ❑ To a lesser degree, memory references to data also tend to be localized.
- ❑ Table-lookup procedures repeatedly refer to that portion in memory where the table is stored.
- ❑ Iterative procedures refer to common memory locations and array of numbers are confined within a local portion of memory.

Cache Memory

- ❑ The result of all these observations is the locality of reference property,
- ❑ which states that over a short interval of time, the addresses generated by a typical program refer to a few localized areas of memory repeatedly,
- ❑ while the remainder of memory is accessed relatively infrequently.

Cache Memory

- ❑ If the active portions of the program and data are placed in a fast small memory,
- ❑ the average memory access time can be reduced, thus reducing the total execution time of the program.
- ❑ Such a fast small memory is referred to as a cache memory.

Cache Memory

- It is placed between the CPU and main memory as illustrated in Fig. 12-1 .

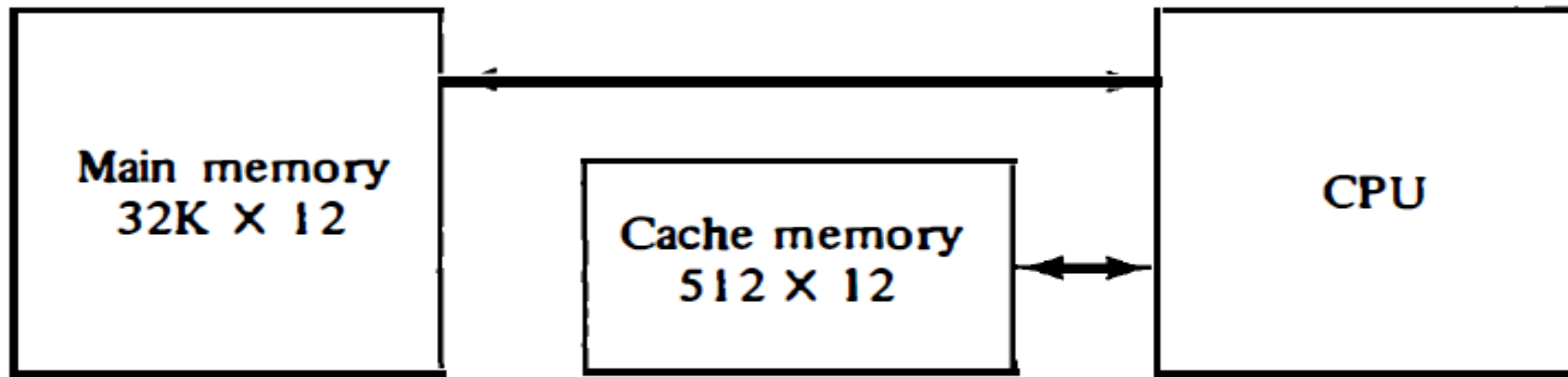


Figure 12-10 Example of cache memory.

Cache Memory

- ❑ The main memory can store 32K words of 12 bits each.
- ❑ The cache is capable of storing 512 of these words at any given time.
- ❑ For every word stored in cache, there is a duplicate copy in main memory.

Cache Memory

- ❑ The CPU communicates with both memories.
- ❑ It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache.
- ❑ If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

Cache Memory

- ❑ The cache memory access time is less than the access time of main memory by a factor of 5 to 10.
- ❑ The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components a duplicate copy in main memory.
- ❑ The CPU communicates with both memories. It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache.
- ❑ If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

Cache Memory

- ❑ The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory,
- ❑ the average memory access time will approach the access time of the cache.
- ❑ Although the cache is only a small fraction of the size of main memory,
- ❑ A large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

Cache Memory

- ❑ The basic operation of the cache is as follows.
- ❑ When the CPU needs to access memory, the cache is examined.
- ❑ If the word is found in the cache, it is read from the fast memory.

Cache Memory

- ❑ If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
- ❑ A block of words containing the one just accessed is then transferred from main memory to cache memory.
- ❑ The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
- ❑ In this manner, some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

Cache Memory

- ❑ The performance of cache memory is frequently measured in terms of a quantity called hit ratio.
- ❑ When the CPU refers to memory and finds the word in cache, it is said to produce a hit.
- ❑ If the word is not found in cache, it is in main memory and it counts as a miss .

Cache Memory

- ❑ The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio.
- ❑ The hit ratio is best measured experimentally by running representative programs in the computer and measuring the number of hits and misses during a given interval of time.

Cache Memory

- ❑ The average memory access time of a computer system can be improved considerably by use of a cache.
- ❑ If the hit ratio is high enough so that most of the time the CPU accesses the cache instead of main memory,
- ❑ the average access time is closer to the access time of the fast cache memory.

Cache Memory

- ❑ The basic characteristic of cache memory is its fast access time.
- ❑ Therefore, very little or no time must be wasted when searching for words in the cache.
- ❑ The transformation of data from main memory to cache memory is referred to as a mapping process.

Cache Memory

❑ Associative Mapping

- ❑ The fastest and most flexible cache organization uses an associative memory. This organization is illustrated in Fig. 12-11.

Cache Memory

- ❑ The associative memory stores both the address and content (data) of the memory word.
- ❑ This permits any location in cache to store any word from main memory.

Cache Memory

- ❑ The diagram shows three words presently stored in the cache.
- ❑ The address value of 15 bits is shown as a five-digit octal number and its corresponding 12 -bit word is shown as a four-digit octal number.
- ❑ A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.

Cache Memory

- ❑ If the address is found, the corresponding 12-bit data is read and sent to the CPU.
- ❑ If no match occurs, the main memory is accessed for the word.

Virtual Memory

- ❑ Virtual memory is a concept used in some large computer systems that
- ❑ permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory.
- ❑ Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.

Virtual Memory

- ❑ Virtual memory is used to give programmers the
- ❑ illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.
- ❑ A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations.
- ❑ This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table.

Virtual Memory

☐ Address Space and Memory Space

- ☐ An address used by a programmer will be called a virtual address, and the set of such addresses the address space.
- ☐ An address in main memory is called a location or physical address .

Virtual Memory

- ❑ The set of such locations is called the memory space.
- ❑ Thus the address space is the set of addresses generated by programs as they reference instructions and data;
- ❑ the memory space consists of the actual main memory locations directly addressable for processing.

Virtual Memory

- ❑ In most computers the address and memory spaces are identical.
- ❑ The address space is allowed to be larger than the memory space in computers with virtual memory.

Virtual Memory

- ❑ As an illustration, consider a computer with a main-memory capacity of $32K$ words ($K = 1024$). Fifteen bits are needed to specify a physical address in memory since $32K = 2^{15}$.
- ❑ Suppose that the computer has available auxiliary memory for storing $2^{20} = 1024K$ words.
- ❑ Thus auxiliary memory has a capacity for storing information equivalent to the capacity of 32 main memories.
- ❑ Denoting the address space by N and the memory space by M , we then have for this example $N = 1024K$ and $M = 32K$.

Virtual Memory

- ❑ In a multiprogram computer system, programs and data are transferred to and from auxiliary memory and main memory based on demands imposed by the CPU.
- ❑ Suppose that program 1 is currently being executed in the CPU.
- ❑ Program 1 and a portion of its associated data are moved from auxiliary memory into main memory as shown in Fig. 12-16.

Virtual Memory

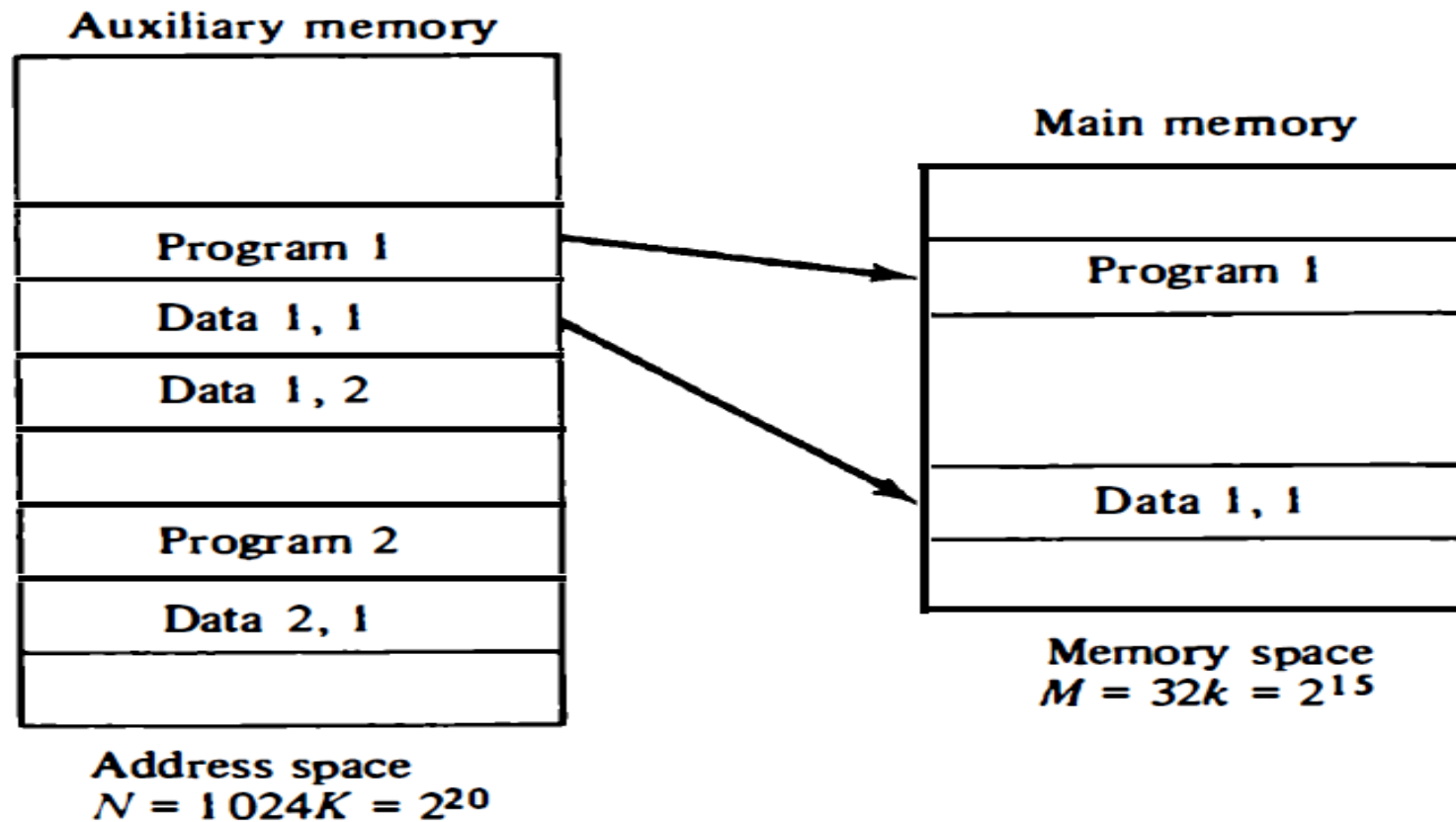


Figure 12-16 Relation between address and memory space in a virtual memory system.

Virtual Memory

- ❑ In a virtual memory system, programmers are told that they have the total address space at their disposal.
- ❑ Moreover, the address field of the instruction code has a sufficient number of bits to specify all virtual addresses.
- ❑ In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits.

Virtual Memory

- ❑ Thus CPU will reference instructions and data with a 20-bit address,
- ❑ but the information at this address must be taken from physical memory
- ❑ because access to auxiliary storage for individual words will be prohibitively long.

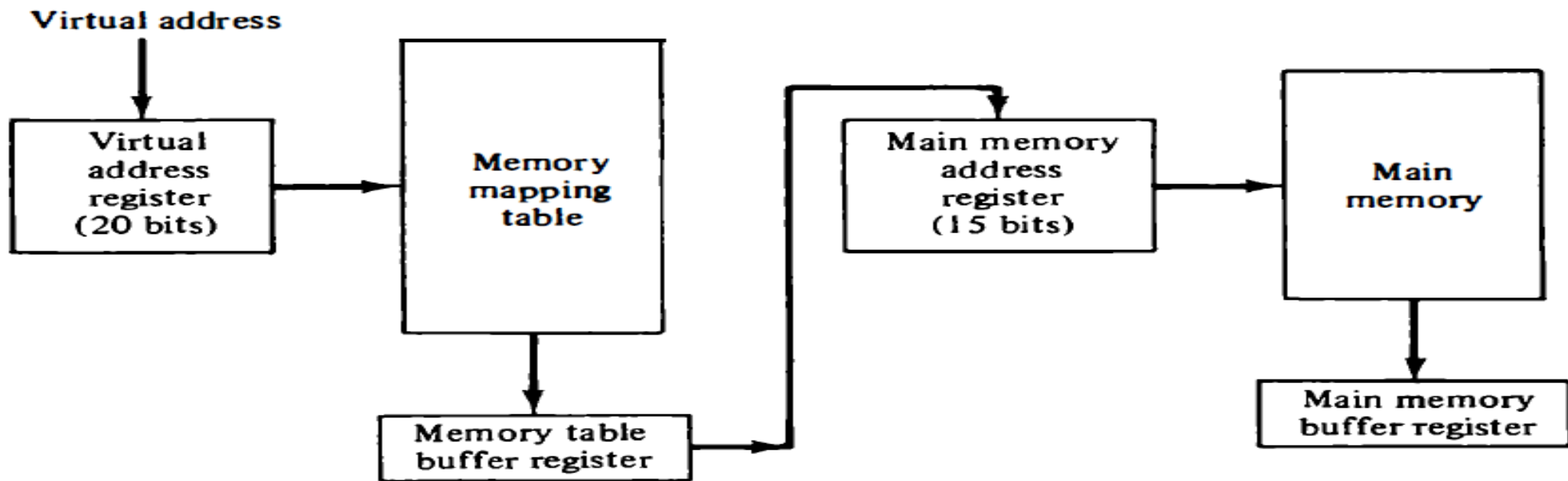
Virtual Memory

- ❑ (Remember that for efficient transfers, auxiliary storage moves an entire record to the main memory.)
- ❑ A table is then needed, as shown in Fig. 12-17, to map a virtual address of 20 bits to a physical address of 15 bits.
- ❑ The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.

Virtual Memory

- ❑ The mapping table may be stored in a separate memory as shown in Fig. 12-17 or in main memory.

Figure 12-17 Memory table for mapping a virtual address.



Memory Management Hardware

- ❑ Memory management system is a collection of hardware and software procedures for managing the various programs residing in memory.
- ❑ The memory management software is part of an overall operating system available in many computers.
- ❑ Here we are concerned with the hardware unit associated with the memory management system.

Memory Management Hardware

□ The basic components of a memory management unit are:

1. A facility for dynamic storage relocation that maps logical memory references into physical memory addresses
2. A provision for sharing common programs stored in memory by different users
3. Protection of information against unauthorized access between users and preventing users from changing operating system functions

Memory Management Hardware

- ❑ It is more convenient to divide programs and data into logical parts called segments.
- ❑ A segment is a set of logically related instructions or data elements associated with a given name.
- ❑ Segments may be generated by the programmer or by the operating system.
- ❑ Examples of segments are a subroutine.

Memory Management Hardware

- ❑ The sharing of common programs is an integral part of a multiprogramming system.
- ❑ For example, several users wishing to compile their Fortran programs should be able to share a single copy of the compiler rather than each user having a separate copy in memory.

Memory Management Hardware

- ❑ Other system programs residing in memory are also shared by all users in a multiprogramming system without having to produce multiple copies.
- ❑ The third issue in multiprogramming is protecting one program from unwanted interaction with another.
- ❑ An example of unwanted interaction is one user's unauthorized copying of another user's program.

Memory Management Hardware

- ❑ Another aspect of protection is concerned with preventing the occasional user from performing operating system functions and
- ❑ Thereby interrupting the orderly sequence of operations in a computer installation.
- ❑ The secrecy of certain programs must be kept from unauthorized personnel to prevent abuses in the confidential activities of an organization.

Memory Management Hardware

- ❑ The address generated by a segmented program is called a logical address.
- ❑ This is similar to a virtual address except that logical address space is associated with variable-length segments rather than fixed-length pages.
- ❑ The logical address may be larger than the physical memory address as in virtual memory, but it may also be equal, and sometimes even smaller than the length of the physical memory address.

Memory Management Hardware

- ❑ Each segment has protection information associated with it.
- ❑ Shared programs are placed in a unique segment in each user's logical address space so that a single physical copy can be shared.
- ❑ The function of the memory management unit is to map logical addresses into physical addresses.

Computer Arithmetic

- ❑ Arithmetic instructions in digital computers manipulate data to produce results necessary for the solution of computational problems.
- ❑ The basic arithmetic operations are
- ❑ addition, subtraction, multiplications etc.

Computer Arithmetic

- ❑ An arithmetic processor is the part of a processor unit that executes arithmetic operations.
- ❑ The data type assumed to reside in processor registers during the execution of an arithmetic instruction is specified in the definition of the instruction.

Computer Arithmetic

- ❑ The solution to any problem that is stated by a finite number of well-defined procedural steps is called an algorithm.
- ❑ A convenient method for presenting algorithms is a flowchart.

Computer Arithmetic

- ❑ The computational steps are specified in the flowchart inside rectangular boxes.
- ❑ The decision steps are indicated inside diamond-shaped boxes from which two or more alternate paths emerge.

Computer Arithmetic

☐ Addition and Subtraction

☐ there are three ways of representing negative fixed-point binary numbers:

☐ signed-magnitude,

☐ signed-1's complement, or

☐ signed-2's complement.

Computer Arithmetic



Example: Represent +9 and -9 in 7 bit-binary number

Only one way to represent

+ 9 ==> 0 001001

Three different ways to represent - 9:

In signed-magnitude: 1 001001

In signed-1's complement: 1 110110

In signed-2's complement: 1 110111

Computer Arithmetic

❑ Addition and Subtraction with Signed-Magnitude Data

❑ We designate the magnitude of the two numbers by A and B.

❑ When the signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed.

Computer Arithmetic

- ❑ These conditions are listed in the first column of Table 10-1.
- ❑ The algorithms for addition and subtraction are derived from the table.

Computer Arithmetic

- ❑ Addition (subtraction) algorithm: when the signs of A and B are identical (different), add the two magnitudes and attach the sign of A to the result.
- ❑ When the signs of A and B are different (identical), compare the magnitudes and subtract the smaller number from the larger.
- ❑ Choose the sign of the result to be the same as A if $A > B$ or the complement of the sign of A if $A < B$. If the two magnitudes are equal, subtract B from A and make the sign of the result positive.

Follow this one:-

3+3

3-2;

2-3;

3-3;

Computer Arithmetic

TABLE 10-1 Addition and Subtraction of Signed-Magnitude Numbers

Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

Here subtract

Here subtract

Follow this one:-

3+3

3-2;

2-3;

3-3;

Computer Arithmetic

❑ Hardware Implementation

- ❑ To implement the two arithmetic operations with hardware, it is first necessary that the two numbers be stored in registers.
- ❑ Let A and B be two registers that hold the magnitudes of the numbers, and A_s and B_s be two flip-flops that hold the corresponding signs.

Computer Arithmetic

- ❑ The result of the operation may be transferred to a third register: however, a saving is achieved if the result is transferred into A and As.
- ❑ Thus A and A, together form an accumulator register.
- ❑ we know that subtraction can be accomplished by means of complement and add.
- ❑ Careful investigation of the alternatives reveals that the use of 2's complement for subtraction and comparison is an efficient procedure that requires only an adder and a complementer.

Computer Arithmetic

- ❑ Figure 10-1 shows a block diagram of the hardware for implementing the addition and subtraction operations.
- ❑ It consists of registers A and B and sign flip-flops A_s and B_s . Subtraction is done by adding A to the 2's complement of B.

Computer Arithmetic

- ❑ The output carry is transferred to flip-flop E, where it can be checked to determine the relative magnitudes of the two numbers.
- ❑ The add-overflow flip-flop AVF holds the overflow bit when A and B are added.

Computer Arithmetic

- ❑ The addition of A plus B is done through the parallel adder.
- ❑ The S (sum) output of the adder is applied to the input of the A register.
- ❑ The complements provides an output of B or the complement of B depending on the state of the mode control M.

Computer Arithmetic

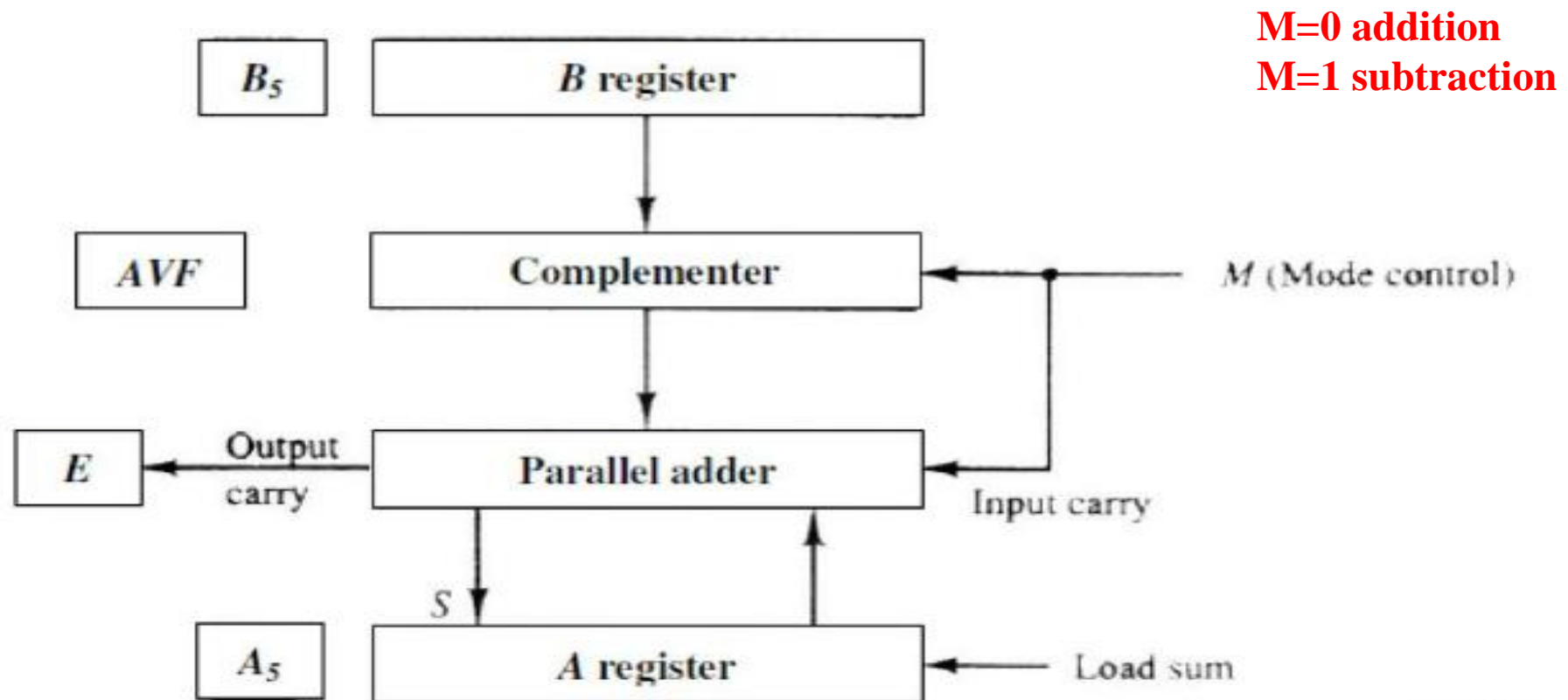
- ❑ The complementer consists of exclusive-OR gates and the parallel adder consists of full-adder circuits as shown in Fig. 4-7 in Chap. 4.
- ❑ The M signal is also applied to the input carry of the adder. When $M = 0$, the output of B is transferred to the adder, the input carry is 0, and the output of the adder is equal to the sum $A + B$.

Computer Arithmetic

- ❑ When M is 1, the 1's complement of B is applied to the adder, the input carry is 1, and output $S = A + \bar{B} + 1$.
- ❑ This is equal to A plus the 2's complement of B, which is equivalent to the subtraction $A - B$.

Computer Arithmetic

Hardware for signed magnitude addition and subtraction

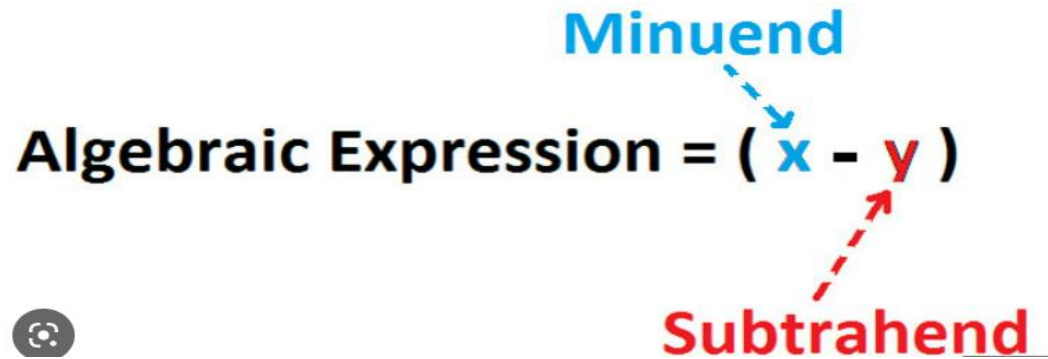


Computer Arithmetic

Algebraic Expression = ($x - y$)

Minuend

Subtrahend

A diagram showing the algebraic expression (x - y). A blue dashed arrow points from the word 'Minuend' to the variable 'x'. A red dashed arrow points from the word 'Subtrahend' to the variable 'y'.

- $A + B$
 - A is the augend
 - B is the addend
- $C - D$
 - C is the minuend
 - D is the subtrahend

Computer Arithmetic

☐ Hardware Algorithm

- ☐ The flowchart for the hardware algorithm is presented in Fig. 10-2.
- ☐ The two signs A, and B, are compared by an exclusive-OR gate.
- ☐ If the output of the gate is 0, the signs are identical; if it is 1, the signs are different.

Computer Arithmetic

- ❑ The magnitudes are added with a microoperation
- ❑ $EA \leftarrow A + B$. The carry in E after the addition constitutes an overflow if it is equal to 1.
- ❑ The value of E is transferred into the add-overflow flip-flop AVF.

Computer Arithmetic

- ❑ The magnitudes are subtracted by adding A to the 2's complement of B.
- ❑ No overflow can occur if the numbers are subtracted so AVF is cleared to 0.
- ❑ A 1 in E indicates that $A \geq B$ and the number in A is the correct result.
- ❑ If this number is zero, the sign A, must be made positive.

Computer Arithmetic

- ❑ A 0 in E: indicates that $A < B$.
- ❑ For this case it is necessary to take the 2's complement of the value in A.
- ❑ This operation can be done with one microoperation $A = A(\text{bar}) + 1$.

Computer Arithmetic

- ❑ However, when $A < B$, the sign of the result is the complement of the original sign of A. It is then necessary to complement A, to obtain the correct sign.
- ❑ The final result is found in register A and its sign in As
- ❑ The value in AVF provides an overflow indication. The final value of E is immaterial.

Computer Arithmetic

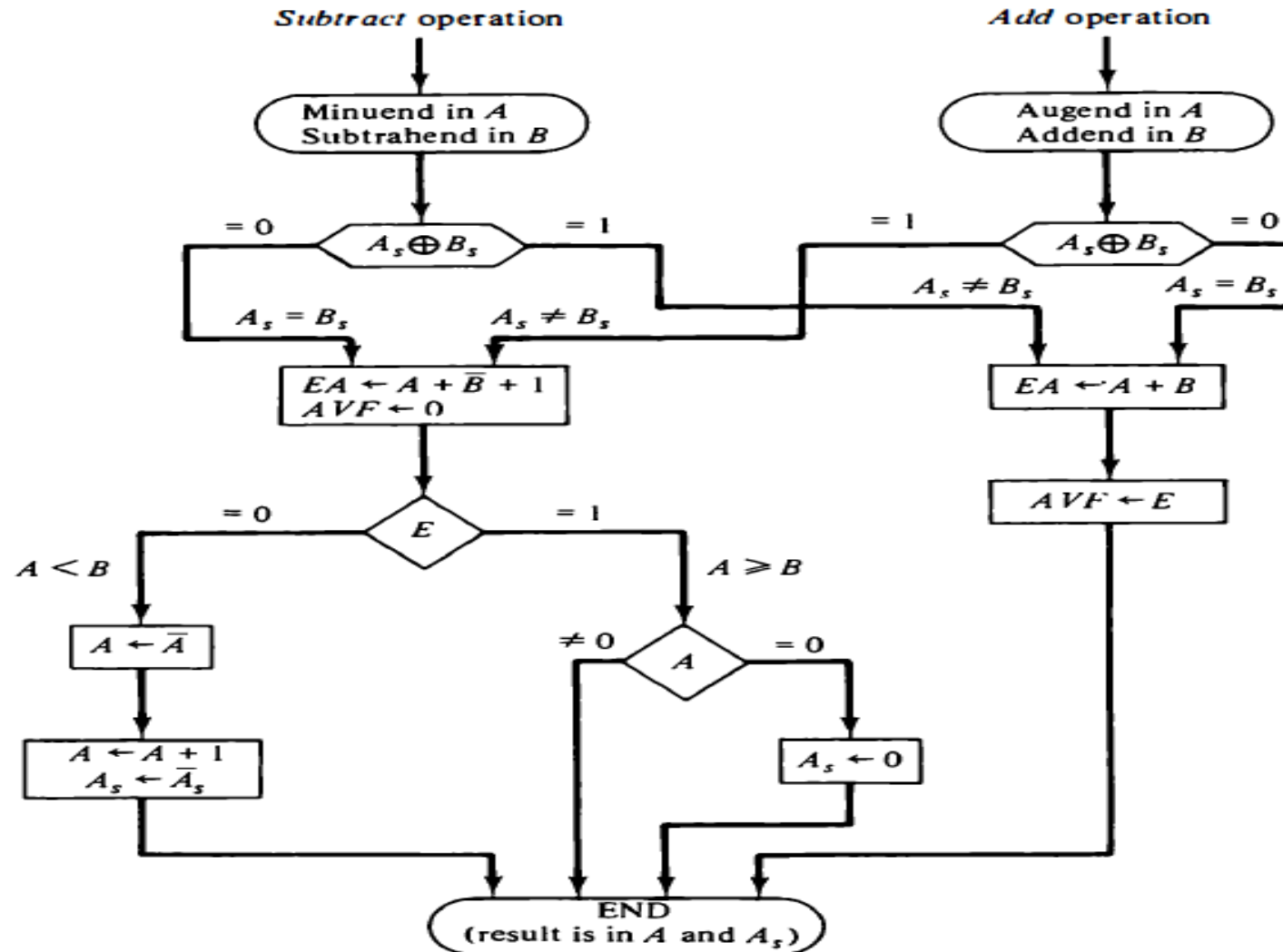


Figure 10-2 Flowchart for add and subtract operations.