ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

CASH MEMORY OF THE COMPUTER, MEMORY ADDRESS AND THEIR ORDER

Ibrokhimjon Tojalievich Tojiboev
Fergana branch of TUIT named after Muhammad al-Khwarizmi
Deputy Director for Academic Affairs

Muhammadamin Mavlonjonovich Mamirkhojayev
Students of the Fergana branch of TUIT named after Muhammad Al-Khwarizmi

Abbosjon Eraliyevich Toychibaev
Students of the Fergana branch of TUIT named after Muhammad Al-Khwarizmi

Jamshidbek Tokhtasinovich Umaraliyev
Students of the Fergana branch of TUIT named after Muhammad Al-Khwarizmi

ABSTRACT

The main content of this article is information about the computer memory device, the basic unit of data storage in memory bits, instructions for the memory device and cache memory in a deep sense.

Keywords: memory block, addressing, cache memory, random access memory, bits, bytes, number systems, memory address, static memory, cache memory level.

INTRODUCTION

Memory is the part of a computer that stores programs and data. The term “storage device” is also common. Without digital memory, where processors can read and write data, there would be no digital computers.

The basic unit of data storage in memory is a binary number called a bit. A bit can contain the number 0 or 1. This is the smallest block of memory. (It was difficult to have a device that only stores zeros, a memory backup - at least two values are required).

It is often said that the use of binary numbers in computers is related to their "efficiency". This means (although speakers themselves rarely realize this) that digital storage can be based on differences between different values of certain physical properties, such as voltage or current. The more differences you have, the less the difference between adjacent values and the less reliable the memory.

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

In binary, only two quantities need to be separated, so this is the most reliable way to encode digital data. If you are unfamiliar with the binary number system, see Appendix A.

MATERIALS AND METHODS

Some computers, such as IBM mainframes, are believed to use decimal and binary arithmetic. In fact, binary decimal code is used here. 4 bits are used to store one bit fraction. These 4 bits provide 16 combinations to accommodate 10 different values (0 to 9). In this case, the remaining 6 combinations are not used. This is what it looked like in 1944, BCD and pure binaries (both use 16 bits):

- BCD representative - 0001 1001 0100 0100; You don't need to know about this yet. "
- binary representation - 0000011110011000.

In binary decimal notation, 16 bits is enough to store a number from 0 to 9999, which means that there are only 10,000 different combinations, and in binary, the same 16 bits allow you to get 65656 combinations. That is why the binary system is called more efficient.

But imagine what would happen if some master engineer brought in a very reliable electronic device that stores the numbers 0-9, dividing the voltage range of 0-10 volts into 10 intervals. These four devices can store decimal numbers from 0 to 9999 or 10,000 combinations. And if the same devices were used to store binary numbers, they could only hold 16 combinations. Of course, a decimal system would be more efficient.

Memory consists of cells, each of which can store part of the data. Each cell has a number called an address. Targeted applications can refer to a specific cell. If there are n cells in memory, they have addresses from 0 to n - 1. All memory cells have the same number of bits. If a cell consists of k bits, it can contain any 2k connections. In fig. 2.8 shows 3 different ways of organizing 96-bit memory. Note that adjacent cells have a number of addresses.

On computers that use the binary system (including octagonal and hexadecimal representations of binary numbers), memory addresses are also represented by binary numbers. If the address consists of m bits, the number of cells included in the address is 2m. For example, the memory access address shown in the picture. 2.8, a, there must be at least 4 bits to represent all numbers from 0 to 11 with the storage device shown in the figure. 2.8b and 2.8, c, 3-bit addresses are enough.

The number of bits in an address determines the maximum number of memory cells that can be reached, and does not depend on the number of bits in a cell. 12-bit

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

addresses are required for the memory of 212 locations, 8-bits and 212 locations, each of which is 64 bits.

A cell is the unit of memory with the smallest address. In recent years, almost all manufacturers have developed computers with 8-bit cells called bytes (sometimes called "octets"). Bytes are grouped into words. A computer with 32-bit words has 4 bytes for each word, and a computer with 64-bit words has 8 bytes.

You need a unit of measure, such as a word, because most commands work with whole words (for example, adding two words). So, a 32-bit machine has 32-bit registers and instructions.

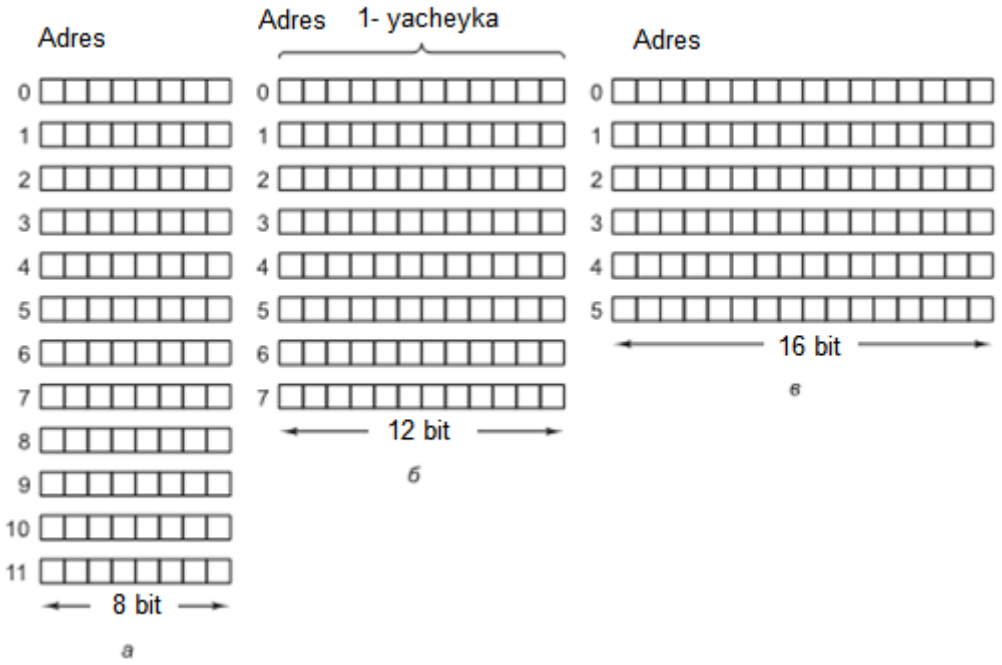


Figure 2.8. Three options for organizing 96-bit memory

32-bit word manipulation, 64-bit machine has 64-bit registers for 64-bit records, and instructions for copying, adding, subtracting, and other operations.

The bytes in a word can be numbered from left to right or right to left. At first glance, it may seem that there is no difference between the two options, but we will soon see that the choice is very important. In fig. 2.9, a shows a 32-bit area of computer memory, the bytes of which are numbered from left to right (as in SPARC computers or IBM base systems). In fig. 2.9, b the memory area of a 32-bit computer is numbered from right to left (as in Intel computers).

In the first of these systems, numbering starts with a higher order and therefore falls into the category of computers that are more orderly than the larger, endangered systems.

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

Incidentally, these terms ("big endian" and "big endian") are taken from Swift's Travels of Gulliver, which, as we recall, refer to the controversy among politicians who end up breaking the egg. They were first introduced into scientific circulation in an excellent article by Cohen (1981).

RESULTS AND DISCUSSION

For simplicity, we assume that the bits do not reverse during transmission. So 0 bytes from the first machine to the second byte 0 and so on, as shown in the picture. 2.10, v.

The computer to log into the system pronounces the name correctly, but its age is 21 x 224 and the partition number does not work either. This is because when transmitting text, the order of the letters in the words changes as needed, but it also changes the order of the bytes in the whole number, which leads to incorrect results.

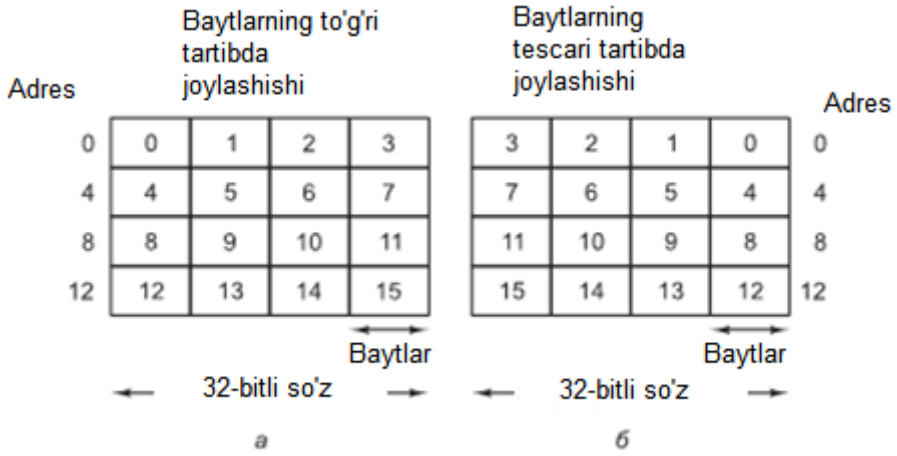


Figure 2.10. The employee data records for the machine are now the same for the machine in the large order (s) in the large Indian order (b); the result of converting a record from a little-endian machine to a little-endian machine (c); The result of the byte replacement in the previous case (d).

The obvious solution to the problem is to use a program that changes the bytes in the word after copying. The result of such an operation is shown in Fig. 2.11, d. We saw that the numbers were correct, but the line became "MIJTIMS" and the letter "H" was usually placed separately. The line is rotated because the computer first reads byte 0 (space), then 1 byte (M), and so on.

There is no easy solution. There is only one way, but it is ineffective.
(Before each block of data should be placed before any block of data that provides information about a string, integers, etc. This will allow the receiving computer to make only the necessary changes.)

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

It is clear that the lack of a byte ordering standard is a serious problem when exchanging data between different machines.

Processors have always been faster than memory. This incompatibility will persist as processors and memory evolve in parallel. Because more transistors can be placed on a chip, processor designers create tubular and ultra-large architectures that further improve CPU performance. Memory designers are using new technologies to increase power, not speed, which increases space. In practice, this mismatch in operating speeds forces the processor to spin several machines until the requested word is received when it enters memory. The slower the memory, the more cycles the processor will have to wait.

As mentioned above, there are two ways to solve the problem. The simplest one is to start reading data from memory and continue executing commands at the same time, but if any command tries to use a word before reading from memory, the processor must stop. The slower the memory, the more often this situation occurs and the longer the processor idle time increases. For example, if one instruction and the access time are 5 cycles out of five input memory, the execution time will be twice as long as the RAM access time. However, if the memory access time is 50 cycles, the execution time increases by 11 times (5 cycles per instruction + 50 memory samples). Another solution to the problem is to design the machine non-stop, but compiler programs don't use words before reading from memory. In practice, however, this is not easy. Often, when processing a boot command, the machine cannot do anything else, so the compiler is forced to enter empty instructions that do not perform any operations, but take up memory space. In fact, this approach results in latency at the software level, not at the hardware level, but the performance degradation is the same.

In fact, the problem is economic, not technological. Engineers know how to create fast memory like a processor.

However, it must be placed directly on the processor chip (since data travels over the bus very slowly). Placing a large amount of memory on a processor chip increases its size and hence its cost. However, even if the cost is low, there are still limitations on the number of processors that can be created. Therefore, you need to choose fast memory with small size and slow memory with large size. (We also prefer fast, large, and inexpensive memory.)

Interestingly, there are technologies that combine small and fast memory with large and slow memory, which allows you to get both high-speed and large memory at a low cost. Small memory is called high-speed cache memory (from the French

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

word "cacher" - "to hide"; it reads "porridge"). Next, we will briefly describe how the cache is used and how it works.

The basic idea behind the cache is simple: it contains the most frequently used words. If the processor needs a word, it is first put into the cache. If there is no word, it belongs to main memory. If a significant portion of the words are in the cache, the average access time is significantly reduced.

Thus, success or failure depends on how many words are in the cache. It is already known that programs do not have random access to memory.

If a program needs access to address A, it needs access to an address near A, since almost all of the commands in a typical program (except for traversal and procedure calls) are called sequentially from memory locations. In addition, the program spends a lot of time outdoors when a limited set of instructions is repeated. Likewise, when working with matrices, the program will often refer to the same matrix before moving on to something else.

The situation that occurs when a small piece of memory is used sequentially over a period of time is called the layout principle. This principle is at the heart of all caching systems. The fact is that when a particular word is called from memory, it is transferred to the cache along with adjacent words, which allows the next request to quickly access the next words. The general arrangement of the processor, cache memory, and main memory is shown in Fig. 2.13.

If a word is read or written more than once, the computer must slowly make one write to main memory, and the fast cache must receive k-1 calls. The higher the K, the higher the performance.

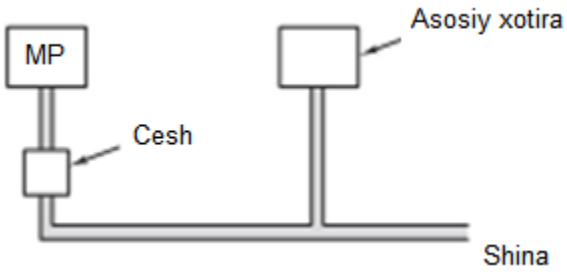


Figure 2.13. Logically, the cache should be located between the processor and main memory. There are actually three possible cache modes.

CONCLUSION

We can make more accurate calculations. Let the access time to the cache memory be equal to m, the ratio of the access time to the main memory m and the

ISSN: 2181-1385
Scientific Journal Impact Factor (SJIF) 2021: 5.723
DOI: 10.24412/2181-1385-2021-7-37-43

number of levels of access to the cache memory to the total number of all entries in the memory. In our example:

$$h = (k - 1) / k.$$

Some authors point the cache miss coefficient to 1 - h.

So we can calculate the average time of entry:

Average access time = c + (1 - h) m.

If h → 1, that is, all calls are made only in cache memory, the access time is C. On the other hand, if

h → 0, that is, each time you need to access the main memory, the access time will be equal to c + m: first you need c time to check the cache memory (in this case, without success), and then you need m time to access the main memory. On some systems, access to the main memory can be started parallel with a cache test, so that if the cache is not missed, the main memory access cycle has already begun. At the same time, this strategy requires the ability to block access to the main memory when the cache is hit, which complicates the design of such a computer.

REFERENCES

1. Aripov MM; Yakubov A.Kh; Sagatov M.V; Irmukhammedova R. m and others. Informatics Information technology textbook. Part 1 Tashkent 2005.
2. Nazirov Sh.A. R.V. Kobulov Use of computers and office equipment. Token 2007.
3. Scott Mueller "PC Upgrade and Repair" Peter 2001
4. A. Dzharov "Jalezo" IBM 2000 Moscow "Micro Art".
5. R. Nasimov. Text of the 19th lecture on the science of computer organization.