

Memory Architecture of a Computer

Main memory

Main memory is the second major subsystem in a computer (Figure 4.1). It consists of a collection of storage locations, each with a unique identifier, called an address. Data is transferred to and from memory in groups of bits called words. A word can be a group of 8 bits, 16 bits, 32 bits, or 64 bits (and growing). If the word is 8 bits, it is referred to as a byte. The term ‘byte’ is so common in computer science that sometimes a 16-bit word is referred to as a 2-byte word, or a 32-bit word is referred to as a 4-byte word.

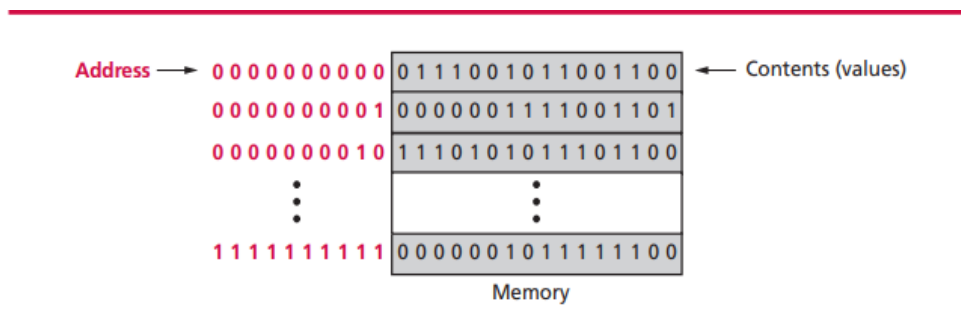


Figure (4.1) Main Memory

Address space

To access a word in memory requires an identifier. Although programmers use a name to identify a word (or a collection of words), at the hardware level each word is identified by an address. The total number of uniquely identifiable locations in memory is called the address space. For example, a memory with 64 kilobytes and a word size of 1 byte has an address space that ranges from 0 to 65535. Table 4.1 shows the units used to refer to memory. Note that the terminology is misleading: it approximates the number of bytes in powers of 10, but the actual number of bytes is in powers of 2. Units in powers of 2 facilitates addressing.

Table (4.1) Memory Units

<i>Unit</i>	<i>Exact Number of Bytes</i>	<i>Approximation</i>
kilobyte	2^{10} (1024) bytes	10^3 bytes
megabyte	2^{20} (1 048 576) bytes	10^6 bytes
gigabyte	2^{30} (1 073 741 824) bytes	10^9 bytes
terabyte	2^{40} bytes	10^{12} bytes

The main memory organization

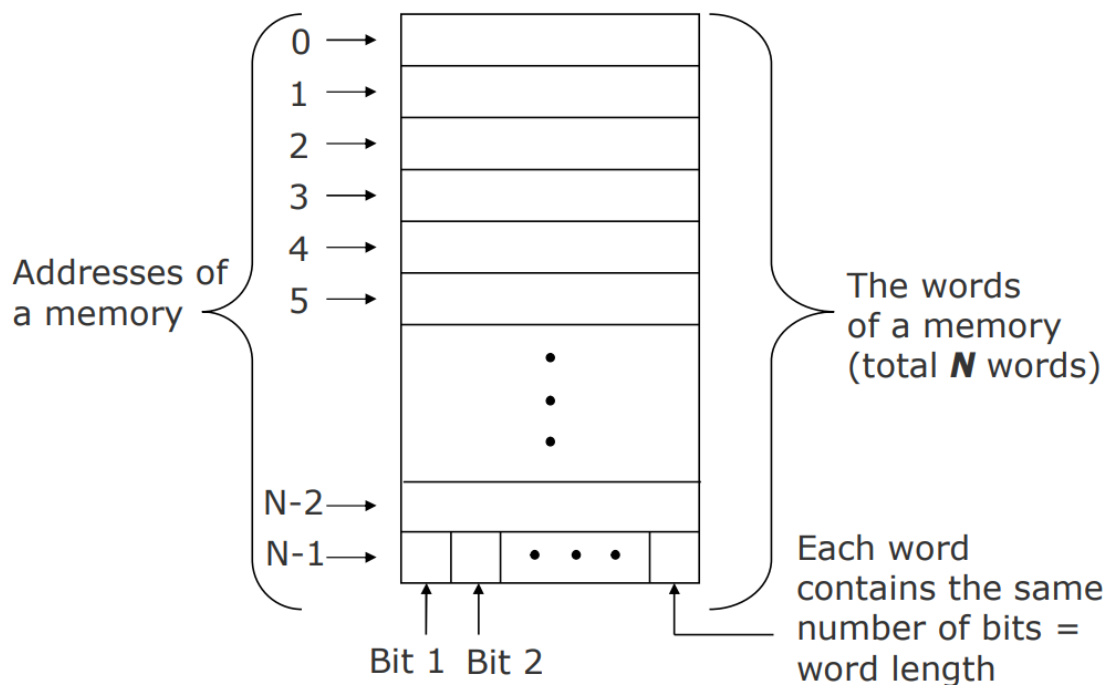


Figure (4.2) The main memory organization

Addresses as bit patterns

Because computers operate by storing numbers as bit patterns, a memory address is also represented as a bit pattern. So if a computer has 64 kilobytes (2^{16}) of memory with a word size of 1 byte, we need a bit pattern of 16 bits to define an address. In other words, the first location is referred to as address 0000000000000000 (address 0), and the last location is referred to as address 1111111111111111 (address 65535). In general, if a computer has N words of

memory, we need an unsigned integer of size $\log_2 N$ bits to refer to each memory location.

Example

A computer has 32 MB (megabytes) of memory. How many bits are needed to address any single byte in memory?

Solution

The memory address space is 32 MB, or 2^{25} ($2^5 \times 2^{20}$). This means that we need $\log_2 2^{25}$, or 25 bits, to address each byte.

Memory types

Two main types of memory exist: RAM and ROM

1- RAM

Random access memory (RAM) makes up most of the main memory in a computer. In a random access device, a data item can be accessed randomly—using the address of the memory location—without the need to access all data items located before it. However, the term is confusing, because ROM can also be accessed randomly. What distinguishes RAM from ROM is that RAM can be read from and written to. Another characteristic of RAM is that it is volatile: the information (program or data) is lost if the computer is powered down. RAM technology is divided into two broad categories:

- SRAM(Static)
- DRAM (Dynamic)

2- ROM

The contents of read-only memory (ROM) are written by the manufacturer, and the CPU can read from, but not write to, ROM. Its advantage is that it is nonvolatile—its contents are not lost if you turn off the computer. Normally, it is used for programs or data that must not be erased or changed even if you turn off the computer. For example, some computers come with ROM that holds the boot program that runs when we switch on the computer.

- PROM (programmable)

-EPROM (erasable programmable)

-EEPROM (electrically erasable programmable)

Memory hierarchy

Computer users need a lot of memory, especially memory that is very fast and inexpensive. This demand is not always possible to satisfy—very fast memory is usually not cheap. A compromise needs to be made. The solution is hierarchical levels of memory (Figure 4.3). The hierarchy is based on the following:

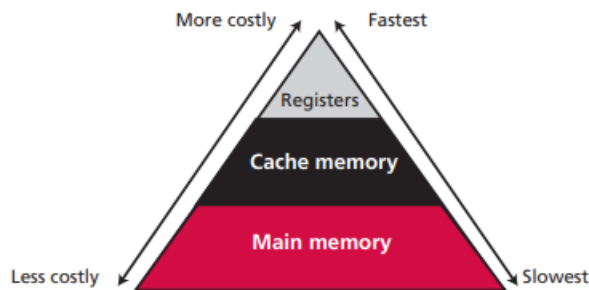


Figure (4.3) Memory hierarchy

- ✓ Use a very small amount of costly high-speed memory where speed is crucial. The registers inside the CPU are of this type.
- ✓ Use a moderate amount of medium-speed memory to store data that is accessed often. Cache memory, discussed next, is of this type.
- ✓ Use a large amount of low-speed memory for data that is accessed less often. Main memory is of this type

Cache memory

Cache memory is faster than main memory but slower than the CPU and its registers. Cache memory, which is normally small in size, is placed between the CPU and main memory.

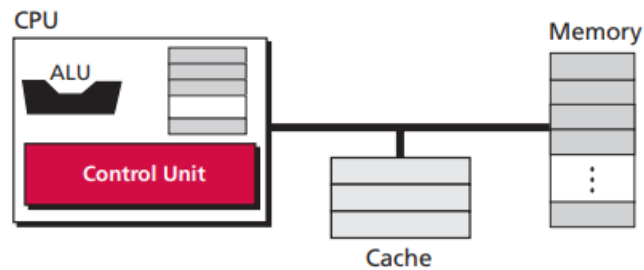


Figure (4.4) Cache Memory

Cache memory at any time contains a copy of a portion of main memory. When the CPU needs to access a word in main memory, it follows this procedure:

1. The CPU checks the cache.
2. If the word is there, it copies the word: if not, the CPU accesses main memory and copies a block of memory starting with the desired word. This block replaces the previous contents of cache memory.
3. The CPU accesses the cache and copies the word.

This procedure can expedite operations; if the word is in the cache, it is accessed immediately. If the word is not in the cache, the word and a whole block are copied to the cache. Since it is probable that the CPU, in its next cycle, will need to access the words following the first word, the existence of the cache speeds processing. We might wonder why cache memory is so efficient despite its small size. The answer lies in the '80–20 rule'. It has been observed that most computers typically spend 80 per cent of their time accessing only 20 per cent of the data. In other words, the same data is accessed over and over again. Cache memory, with its high speed, can hold this 20 per cent to make access faster at least 80 per cent of the time.