Computer storage and data types

Basics for data storage

Daniel Giovanni Martínez Sandoval

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS
UNIVERSIDAD DE GUADALAJARA

I7266 - Programación de Sistemas Embebidos - 2024B

Table of Contents

Table of Contents

Brief history of data storage

Data storage in computers has evolved from mechanical methods to faster and more reliable storage without moving parts.

Brief history of data storage

Data storage in computers has evolved from mechanical methods to faster and more reliable storage without moving parts.

Let us explore some of these methods.

Punch cards

One of the first methods for storage were *Punch Cards* mainly used between the 1890s and 1960s:

Punch cards

One of the first methods for storage were *Punch Cards* mainly used between the 1890s and 1960s:

- In the 1890s, Herman Hollerith developed a punch card system (based on Joseph-Marie Jacquard's idea, who used punched cards to control textile looms) for data storage and processing for the U.S. Census Bureau.

Punch cards

One of the first methods for storage were *Punch Cards* mainly used between the 1890s and 1960s:

- In the 1890s, Herman Hollerith developed a punch card system (based on Joseph-Marie Jacquard's idea, who used punched cards to control textile looms) for data storage and processing for the U.S. Census Bureau.
- His machine was able to read and sort punched cards, revolutionizing data processing at the time and leading to the founding of IBM (International Business Machines), which became a major producer of punch card technology.

Punch cards (cont.)

- A punch card is a piece of stiff paper with holes punched into it. Each hole represents a piece of data, typically a bit (binary digit) of information.

Punch cards (cont.)

- A punch card is a piece of stiff paper with holes punched into it. Each hole represents a piece of data, typically a bit (binary digit) of information.
- The card is divided into rows and columns, with each row typically representing one character or value, and each column a bit position (0 or 1).

Punch cards (cont.)

- A punch card is a piece of stiff paper with holes punched into it. Each hole represents a piece of data, typically a bit (binary digit) of information.
- The card is divided into rows and columns, with each row typically representing one character or value, and each column a bit position (0 or 1).
- Machines would ”read” the punch cards by passing them through an array of mechanical or electrical sensors that detected the holes.

Punch cards (cont.)

- A punch card is a piece of stiff paper with holes punched into it. Each hole represents a piece of data, typically a bit (binary digit) of information.
- The card is divided into rows and columns, with each row typically representing one character or value, and each column a bit position (0 or 1).
- Machines would ”read” the punch cards by passing them through an array of mechanical or electrical sensors that detected the holes.
- Punch cards were widely used in the first half of the 20th century, particularly for business and government applications.

Punch cards (cont.)

- Early computers like the ENIAC, IBM 1401, and UNIVAC relied on punch cards for both data input and programming.

Punch cards (cont.)

- Early computers like the ENIAC, IBM 1401, and UNIVAC relied on punch cards for both data input and programming.
- COBOL, FORTRAN, and other early programming languages were often coded and compiled using punch cards.

Punch cards (cont.)

- Early computers like the ENIAC, IBM 1401, and UNIVAC relied on punch cards for both data input and programming.
- COBOL, FORTRAN, and other early programming languages were often coded and compiled using punch cards.
- During the 1960s and 1970s, punch cards were a staple in computer labs and industries like banking, manufacturing, and government.

Magnetic drums

Magnetic drums were also one of the earliest forms of data storage, they were used in computers from the 1930s through the 1950s:

Magnetic drums

Magnetic drums were also one of the earliest forms of data storage, they were used in computers from the 1930s through the 1950s:

- Magnetic drum memory was a cylindrical storage device coated with a magnetic material, used to store data as magnetic patterns. The concept was developed in the 1930s by Gustav Tauschek in Austria, and later improved and used in early digital computers.

Magnetic drums

Magnetic drums were also one of the earliest forms of data storage, they were used in computers from the 1930s through the 1950s:

- Magnetic drum memory was a cylindrical storage device coated with a magnetic material, used to store data as magnetic patterns. The concept was developed in the 1930s by Gustav Tauschek in Austria, and later improved and used in early digital computers.
- Magnetic drums became widely used in the 1950s for main memory in many early computers like the IBM 650, UNIVAC I, and the ERA 1103.

Magnetic drums (cont.)

- A magnetic drums are metal cylinders, typically about 30 to 60 centimetre long, coated with a magnetic oxide, which rotate continuously, and data is written to or read from the surface by fixed read/write heads, positioned along the length of the drum. As the drum rotates, each head reads or writes data on a specific track (a circular path on the drum's surface).

Magnetic drums (cont.)

- A magnetic drums are metal cylinders, typically about 30 to 60 centimetre long, coated with a magnetic oxide, which rotate continuously, and data is written to or read from the surface by fixed read/write heads, positioned along the length of the drum. As the drum rotates, each head reads or writes data on a specific track (a circular path on the drum's surface).
- The drum operated at a consistent rotational speed, and the timing of the read/write operations was critical, as data could only be accessed when the desired part of the drum surface was under the read/write head.

Magnetic drums (cont.)

- They were capable of Random Access although not very efficient.

Magnetic drums (cont.)

- They were capable of Random Access although not very efficient.
- They stored close to 64KB at max.

Magnetic drums (cont.)

- They were capable of Random Access although not very efficient.
- They stored close to 64KB at max.
- They were used as both as primary or secondary storage.

Magnetic cores

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

Magnetic cores

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Magnetic core memory uses tiny magnetized rings (cores) made of ferrite material to store data, where each ring stores one bit of information: either a 0 or a 1, depending on the magnetic polarity of the core.

Magnetic cores

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Magnetic core memory uses tiny magnetized rings (cores) made of ferrite material to store data, where each ring stores one bit of information: either a 0 or a 1, depending on the magnetic polarity of the core.
- The memory is arranged in a grid or matrix of these cores.

Magnetic cores

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Magnetic core memory uses tiny magnetized rings (cores) made of ferrite material to store data, where each ring stores one bit of information: either a 0 or a 1, depending on the magnetic polarity of the core.
- The memory is arranged in a grid or matrix of these cores.
- It was a type of random access memory (RAM), meaning any part of the memory could be accessed directly and independently, unlike earlier sequential storage methods like punch cards or magnetic tape.

Magnetic cores (cont.)

- Each core is made of a small ferrite ring, and wires are threaded through the core to read, write, and reset the magnetic state. The cores are organized in a grid or matrix, with rows and columns of wires. Addressing a specific core requires sending current through specific row and column wires to access that bit.

Magnetic cores (cont.)

- Each core is made of a small ferrite ring, and wires are threaded through the core to read, write, and reset the magnetic state. The cores are organized in a grid or matrix, with rows and columns of wires. Addressing a specific core requires sending current through specific row and column wires to access that bit.
- Data is written to memory by sending electrical currents through the wires, and reading data involves changing the magnetic state of a core and detecting if the change has occurred, which tells the system what the stored bit was. This process of reading destroys the data, so it had to be rewritten after every read.

Magnetic cores (cont.)

- They were good for Random Acces, very reliable and long-lasting, non-voaltile and easily scalable.

Magnetic cores (cont.)

- They were good for Random Acces, very reliable and long-lasting, non-voaltile and easily scalable.
- Their main disadvantage was that manufacturing them was labour intensive, and thus, they were quite expensive.

Magnetic cores (cont.)

- They were good for Random Acces, very reliable and long-lasting, non-voaltile and easily scalable.
- Their main disadvantage was that manufacturing them was labour intensive, and thus, they were quite expensive.
- They were used as both as primary or secondary storage.

Hard Disk Drives

In 1956, IBM introduced the *Hard Disk Drives* or HDD, which are still in use today:

Hard Disk Drives

In 1956, IBM introduced the *Hard Disk Drives* or HDD, which are still in use today:

- HDDs store data on rotating disks (platters) coated with magnetic material.

Hard Disk Drives

In 1956, IBM introduced the *Hard Disk Drives* or HDD, which are still in use today:

- HDDs store data on rotating disks (platters) coated with magnetic material.
- The HDD stores information in binary form (0s and 1s), using a magnetic read/write head which changes in magnetic polarity to represent bits of data.

Hard Disk Drives

In 1956, IBM introduced the *Hard Disk Drives* or HDD, which are still in use today:

- HDDs store data on rotating disks (platters) coated with magnetic material.
- The HDD stores information in binary form (0s and 1s), using a magnetic read/write head which changes in magnetic polarity to represent bits of data.
- Rotational speed and data density on the platters are the main factors determining the speed and storage capacity of the HDD.

Hard Disk Drives

In 1956, IBM introduced the *Hard Disk Drives* or HDD, which are still in use today:

- HDDs store data on rotating disks (platters) coated with magnetic material.
- The HDD stores information in binary form (0s and 1s), using a magnetic read/write head which changes in magnetic polarity to represent bits of data.
- Rotational speed and data density on the platters are the main factors determining the speed and storage capacity of the HDD.

Hard Disk Drives (cont.)

- The first HDD, the IBM 350, was introduced in 1956 as part of the IBM RAMAC system. It used 50 24-inch platters and had a storage capacity of about 5 MB.

Hard Disk Drives (cont.)

- The first HDD, the IBM 350, was introduced in 1956 as part of the IBM RAMAC system. It used 50 24-inch platters and had a storage capacity of about 5 MB.
- Early HDDs were large and expensive, making them accessible only to businesses and research institutions.

Hard Disk Drives (cont.)

- The first HDD, the IBM 350, was introduced in 1956 as part of the IBM RAMAC system. It used 50 24-inch platters and had a storage capacity of about 5 MB.
- Early HDDs were large and expensive, making them accessible only to businesses and research institutions.
- Modern HDDs are much smaller and offer thousands of times more capacity. A typical consumer HDD today can store anywhere from 1 TB to 10 TB.

Hard Disk Drives (cont.)

- The first HDD, the IBM 350, was introduced in 1956 as part of the IBM RAMAC system. It used 50 24-inch platters and had a storage capacity of about 5 MB.
- Early HDDs were large and expensive, making them accessible only to businesses and research institutions.
- Modern HDDs are much smaller and offer thousands of times more capacity. A typical consumer HDD today can store anywhere from 1 TB to 10 TB.

Floppy disks

Floppy disks were a popular form of removable storage from the 1970s through the early 2000s:

Floppy disks

Floppy disks were a popular form of removable storage from the 1970s through the early 2000s:

- A floppy disk is a thin, flexible magnetic storage medium encased in a square or rectangular plastic shell. They were widely used to store and transfer files, as well as to distribute software, operating systems, and games.

Floppy disks

Floppy disks were a popular form of removable storage from the 1970s through the early 2000s:

- A floppy disk is a thin, flexible magnetic storage medium encased in a square or rectangular plastic shell. They were widely used to store and transfer files, as well as to distribute software, operating systems, and games.
- The actual storage medium is a circular, thin disk made of flexible plastic and coated with a magnetic material (usually iron oxide) where data is stored on concentric tracks, and the magnetic properties of the disk are altered to represent binary data (0s and 1s).

Solid-State Drives

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

Solid-State Drives

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Solid-State Drives (SSDs) are data storage devices that use flash memory to store information, meaning they have no moving parts (hence the term ”solid-state”).

Solid-State Drives

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Solid-State Drives (SSDs) are data storage devices that use flash memory to store information, meaning they have no moving parts (hence the term ”solid-state”).
- SSDs are significantly faster and more durable than HDDs, making them a popular choice for modern computing systems.

Solid-State Drives

Magnetic core memory was a type of non-volatile memory used in computers from the early 1950s until the mid-1970s:

- Solid-State Drives (SSDs) are data storage devices that use flash memory to store information, meaning they have no moving parts (hence the term ”solid-state”).
- SSDs are significantly faster and more durable than HDDs, making them a popular choice for modern computing systems.
- Data in SSDs is stored in memory cells, which are organized into pages and further into blocks, where different types of flash cells store from 1 bit (SLC) up to 4 bits (QLC).

Optical storage

Optical storage is a type of storage which is still in use, it begun to be used in the 1980s:

Optical storage

Optical storage is a type of storage which is still in use, it begun to be used in the 1980s:

- CDs, DVDs, and Blu-ray discs emerged in the 1980s and 1990s as optical storage solutions.

Optical storage

Optical storage is a type of storage which is still in use, it begun to be used in the 1980s:

- CDs, DVDs, and Blu-ray discs emerged in the 1980s and 1990s as optical storage solutions.
- Data is stored using laser technology to read and write to the discs. While CDs and DVDs are less commonly used today, Blu-ray discs remain relevant for large media storage.

Cloud storage

Cloud storage is a recent type of storage which started to gain popularity in the 2010s:

Cloud storage

Cloud storage is a recent type of storage which started to gain popularity in the 2010s:

- In the 2000s, cloud storage became prevalent, allowing data to be stored remotely on distributed servers.

Cloud storage

Cloud storage is a recent type of storage which started to gain popularity in the 2010s:

- In the 2000s, cloud storage became prevalent, allowing data to be stored remotely on distributed servers.
- It enables access to data from anywhere with an internet connection and offers scalable, cost-efficient storage for individuals and organizations.

Cloud storage

Cloud storage is a recent type of storage which started to gain popularity in the 2010s:

- In the 2000s, cloud storage became prevalent, allowing data to be stored remotely on distributed servers.
- It enables access to data from anywhere with an internet connection and offers scalable, cost-efficient storage for individuals and organizations.
- Although it is irrelevant for embedded systems as means of primary storage, it is becoming really important for many embedded systems since we are growing towards IoT (Internet of Things) technologies.

Similarities in storage methods

Actually, all these storage methods followed a simple principle: to store binary data. At the most fundamental level, all storage methods encode data in binary form (0s and 1s). Whether using magnetic fields, electrical charges, or optical properties, the data is ultimately represented as binary information that the computer processes.

Similarities in storage methods

Actually, all these storage methods followed a simple principle: to store binary data. At the most fundamental level, all storage methods encode data in binary form (0s and 1s). Whether using magnetic fields, electrical charges, or optical properties, the data is ultimately represented as binary information that the computer processes.

Also, storage methods have always been part of a broader memory hierarchy. This hierarchy includes multiple levels, ranging from fast, volatile memory (like RAM) to slower, non-volatile storage (like HDDs and SSDs). This hierarchy helps balance speed and capacity in computing systems.

We now need to dive deeper into how bits are stored in the current storage technologies.

Table of Contents

Volatile memory

In modern microcontrollers we can usually find:

Volatile memory

In modern microcontrollers we can usually find:

- RAM (Random Access Memory): Used for temporary data storage during program execution, such as stack and heap.

Volatile memory

In modern microcontrollers we can usually find:

- RAM (Random Access Memory): Used for temporary data storage during program execution, such as stack and heap.
- SRAM (Static RAM): Used in smaller amounts for faster, low-power memory. Commonly used for caches and real-time data storage.

Volatile memory

In modern microcontrollers we can usually find:

- RAM (Random Access Memory): Used for temporary data storage during program execution, such as stack and heap.
- SRAM (Static RAM): Used in smaller amounts for faster, low-power memory. Commonly used for caches and real-time data storage.
- DRAM (Dynamic RAM): Used for larger memory capacity but is slower than SRAM. Typically found in more resource-intensive embedded systems (e.g., smartphones).

Non-volatile memory

In modern microcontrollers we can usually find:

- Flash Memory: Ued to store the program code and persistent data. Flash memory can be divided into two types:

Non-volatile memory

In modern microcontrollers we can usually find:

- Flash Memory: Ued to store the program code and persistent data. Flash memory can be divided into two types:
  - NOR Flash: Used for code execution in place (XIP) because of its fast read speeds.
  - NAND Flash: Primarily used for bulk data storage due to its higher density and cost-effectiveness.

Non-volatile memory

In modern microcontrollers we can usually find:

- Flash Memory: Ued to store the program code and persistent data. Flash memory can be divided into two types:
  - NOR Flash: Used for code execution in place (XIP) because of its fast read speeds.
  - NAND Flash: Primarily used for bulk data storage due to its higher density and cost-effectiveness.
- EEPROM (Electrically Erasable Programmable Read-Only Memory): Used for storing small amounts of data that must persist across power cycles, such as configuration settings or calibration data.

Memory map

The way the memory in a microcontroller is organized is defined (partially, from the software perspective) by the *memory map*.

Memory map

The way the memory in a microcontroller is organized is defined (partially, from the software perspective) by the *memory map*. This is a structured arrangement of different memory areas within the embedded system, defining how the CPU accesses RAM, ROM, Flash, and peripherals.

Memory map

The way the memory in a microcontroller is organized is defined (partially, from the software perspective) by the *memory map*. This is a structured arrangement of different memory areas within the embedded system, defining how the CPU accesses RAM, ROM, Flash, and peripherals. Memory is usually divided into two big groups:

- Program Memory: Non-volatile memory (e.g., Flash) that stores the embedded system's firmware or application code.
- Data Memory: Volatile memory (e.g., RAM) used for temporary data storage during program execution.

Memory map (cont.)

It is important to note that the way the memory is organized in a computer system (the memory architecture) is defining the way the memory map will work, and, for example, how many memory maps we will have.

Memory map (cont.)

It is important to note that the way the memory is organized in a computer system (the memory architecture) is defining the way the memory map will work, and, for example, how many memory maps we will have.
Now, how can we use that information to efficiently store information in an embedded system? And why should we care about that?

Table of Contents

Why do data types exist in programming?

Embedded systems have limited resources, and although dependant on the intended application, they usually have really small amounts of memory space compared to regular computers.

Why do data types exist in programming?

Embedded systems have limited resources, and although dependant on the intended application, they usually have really small amounts of memory space compared to regular computers. Since we have limited memory, we want to have exceptional control over how the memory is used.

Why do data types exist in programming?

Embedded systems have limited resources, and although dependant on the intended application, they usually have really small amounts of memory space compared to regular computers. Since we have limited memory, we want to have exceptional control over how the memory is used.
Most programming languages include some level of control over this, specially the mid and low-level languages. This enables programmers to choose better ways to manage the memory, but also sometimes makes it harder to program, since you have to take into account more details about it.

Data types in C

C was invented as a high-level language, meaning it provided a high level of abstraction and was easily understandable by most people related to the area. With time, computer systems have evolved so complex that it does not seem that easy to understand nowadays, but it is still considered a mid-level language.

Data types in C

C was invented as a high-level language, meaning it provided a high level of abstraction and was easily understandable by most people related to the area. With time, computer systems have evolved so complex that it does not seem that easy to understand nowadays, but it is still considered a mid-level language.
It is important to remark that this language is widely used in embedded systems due to its efficiency and control over hardware, thus, it is crucial to understand data types and memory in order to write optimized code and manage limited resources.

Data types in C (cont.)

In the C language, we have different ways to describe how we want to use memory in our microcontroller or microprocessor. There are, as well, multiple ways to classify data types, but among them we will care mainly about two things:

- What kind of data can they store?
- What size are they?

Data types in C (cont.)

Primary data types are the building blocks for more complex methods of storing data. Our most basic components are the following:

Name	Bytes	Range	Format specifier
char	1	-128 to 127	%c
int	4	$-2\,147\,483\,648$ to $2\,147\,483\,647$	%d
float	4	1.21×10^{-38} to 3.4×10^{38}	%f
double	8	1.7×10^{-308} to 1.7×10^{308}	%lf

Data types in C (cont.)

The usual way to use them is like this:

- **char:** to store characters, sin ASCII uses 1 byte per character.
- **int:** to store integers.
- **float:** to store numbers with decimals.
- **double:** to store really big numbers with certain precision.

Integers in C

That doesn't mean we're limited to those use cases, in fact, let us explore a little bit further how integers are stored:

Integers in C (cont.)

Integers in C (cont.)

Floating point numbers)

Floating point numbers (cont.)

Floating point numbers (cont.)

Arrays

Arrays (cont.)

Enums (cont.)

Enums (cont.)

Structs

Structs (cont.)

Type qualifiers

Memory addresses

We have discussed about the parts that make up the memory of a microcontroller. In simple terms:

- **Program Flash (ROM):** Stores the firmware.
- **RAM:** Stores data generated and used by the firmware.
- **EEPROM:** Stores persistent data.

We have also discussed about the memory map and how it depends on the system architecture.

Memory addresses

A memory address is a unique identifier for a specific location in the microcontroller's memory.

Memory addresses

A memory address is a unique identifier for a specific location in the microcontroller's memory. We use addresses to identify addressable units. In most systems the smallest addressable unit is the byte, so, each memory cell can store a byte.

Memory addresses

A memory address is a unique identifier for a specific location in the microcontroller's memory. We use addresses to identify addressable units. In most systems the smallest addressable unit is the byte, so, each memory cell can store a byte.
The address space is the range of possible memory addresses that out control unit can generate.

Memory addresses

A memory address is a unique identifier for a specific location in the microcontroller's memory. We use addresses to identify addressable units. In most systems the smallest addressable unit is the byte, so, each memory cell can store a byte.

The address space is the range of possible memory addresses that out control unit can generate. Word Addressing is accessing to memory with the addresses for words, this means, groups of bytes of a predefined size.

Memory addresses

A memory address is a unique identifier for a specific location in the microcontroller's memory. We use addresses to identify addressable units. In most systems the smallest addressable unit is the byte, so, each memory cell can store a byte.
The address space is the range of possible memory addresses that out control unit can generate. Word Addressing is accessing to memory with the addresses for words, this means, groups of bytes of a predefined size.
Also, endianness refers to the order of multi-byte datatypes. We have Little-Endian and Big-Endian.

Pointers

Pointers (cont.)

Pointers (cont.)