**Memory**

* Memories are stored in cells, each cells having their own addresses in **bytes (8 bits)**.
* Memory addresses are written in hexadecimal numbers (easier for human to read).
* Total space of the memory for a program is **264 bytes** (limit for a program)

Diagram

Description automatically generated with medium confidence

* Instructions are not allowed to be changed, after compilation 🡪 Read-only
* **Sp** - stack pointer - point to the memory region on top of the stack (nothing there)
* Stack will grow downwards (shrink upwards) 🡪 value of sp will be smaller and smaller

Diagram

Description automatically generated with medium confidence

**QUESTION: Why do we need dynamic memory in the first place?any**

The primary reason for using dynamic memory is to create objects with a dynamic size. Unlike objects created on the stack, where the size must be known at compile-time, dynamic memory allocation allows you to allocate memory based on runtime conditions. This flexibility is especially useful when dealing with data structures such as arrays or linked lists, where the size may vary during program execution or when the required size is too large to fit in the limited stack memory.

Another reason for using dynamic memory is to control the object's lifetime. Objects created on the stack have automatic storage duration, meaning they are automatically destroyed when they go out of scope. On the other hand, dynamic memory allows you to manage the lifetime of objects explicitly by allocating and deallocating memory as needed. This control is important when you want to create objects that persist beyond the scope of a single function or when you want to share objects between different parts of your program.

Stack-based memory allocation is generally faster and more efficient than dynamic memory allocation because it involves simple pointer manipulations and does not require heap management.