Topic 1-3

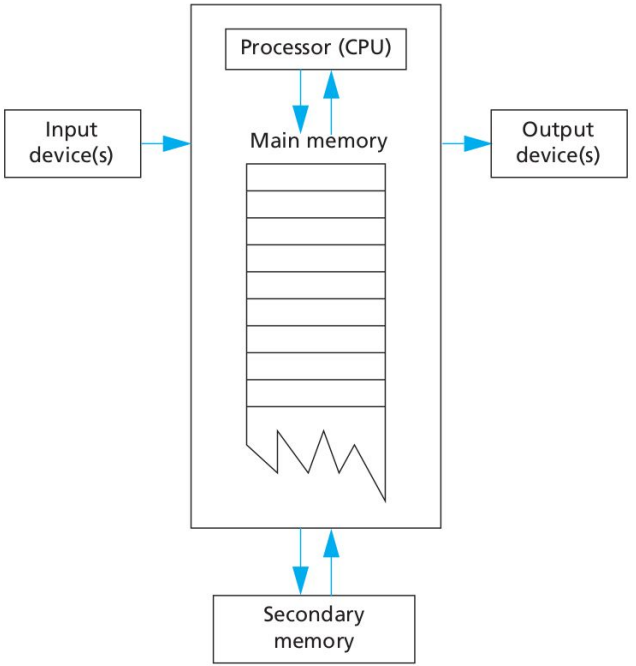
Memory and Computer Architecture

In this lecture

- Basic definitions
- Bits, Bytes and Words
- First look at built-in data types and their sizes
- Memory fragmentation
- First look at the view of memory from a c++ program point of view

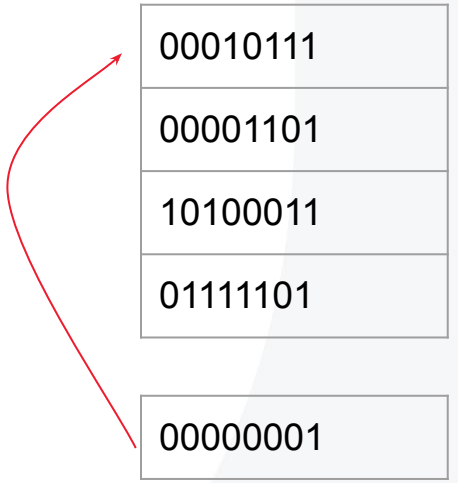
Basics

- A **program** is a set of instructions for a computer to follow.
- **Software** is the collection of programs used by a computer.
- **Hardware** is the actual physical machines that make up a computer.
- There are 5 **main components** that makes a computer.
- Your C++ program lives in the main memory.
- Files stored in the secondary memory (e.g. on Hard Disks).



The Main memory

- Long list of **locations** each with an **address**.
- Each memory location contains a set of zero/one digits.
- **bits** comes from binary digits.
- **byte** is 8 bits
- **word** is the size of data a processor can manipulate at once. The most common word sizes encountered today are 8, 16, 32 and 64 bits.



Memory location that contains the address of another memory location

The Built-in data types and their sizes.

Follow the example in the demo video and fill in the size for each data type.

Data Type	Size (in bytes)	Range
short int		-32,768 to 32,767
unsigned short int		0 to 65,535
unsigned int		0 to 4,294,967,295
int		-2,147,483,648 to 2,147,483,647
long int		-2,147,483,648 to 2,147,483,647
unsigned long int		0 to 4,294,967,295
long long int		-(2^63) to (2^63)-1
unsigned long long int		0 to 18,446,744,073,709,551,615
signed char		-128 to 127
unsigned char		0 to 255
float		
double		
long double		
wchar_t		1 wide character

Memory locations are Contiguous

- **If you have more than one program sharing memory, they must be protected from each other.**
- **If a program can refer to memory outside its allocated area it could**
  - Damage other running programs
  - Control other running programs
  - Steal information from other running programs

THE UNIVERSITY of ADELAIDE

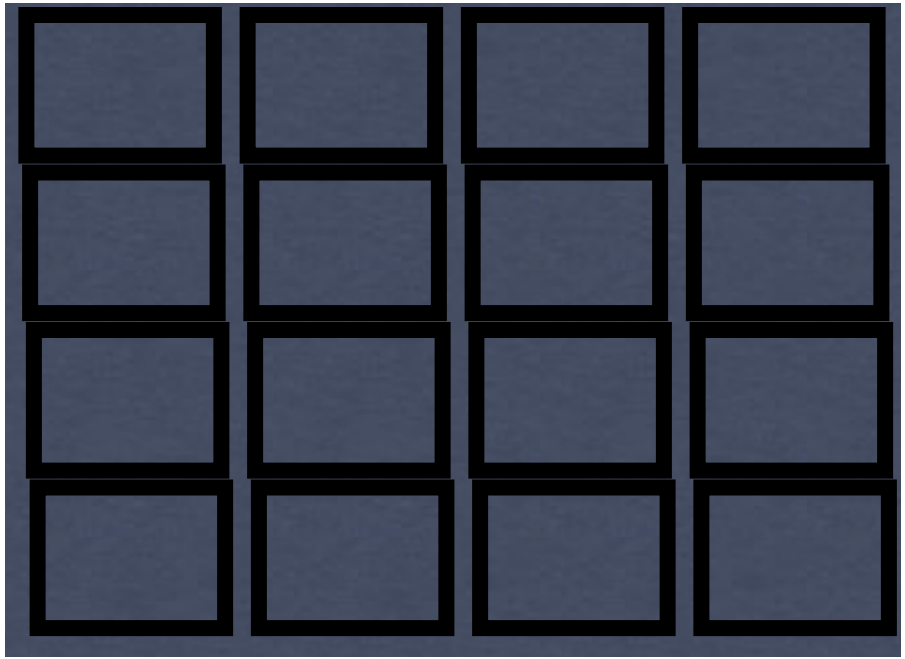
MEMORY

0-3

4-7

8-11

12-15



THE UNIVERSITY of ADELAIDE

Fragmentation

Over time, memory allocation and deallocation can make it hard to allocate large contiguous blocks of memory.

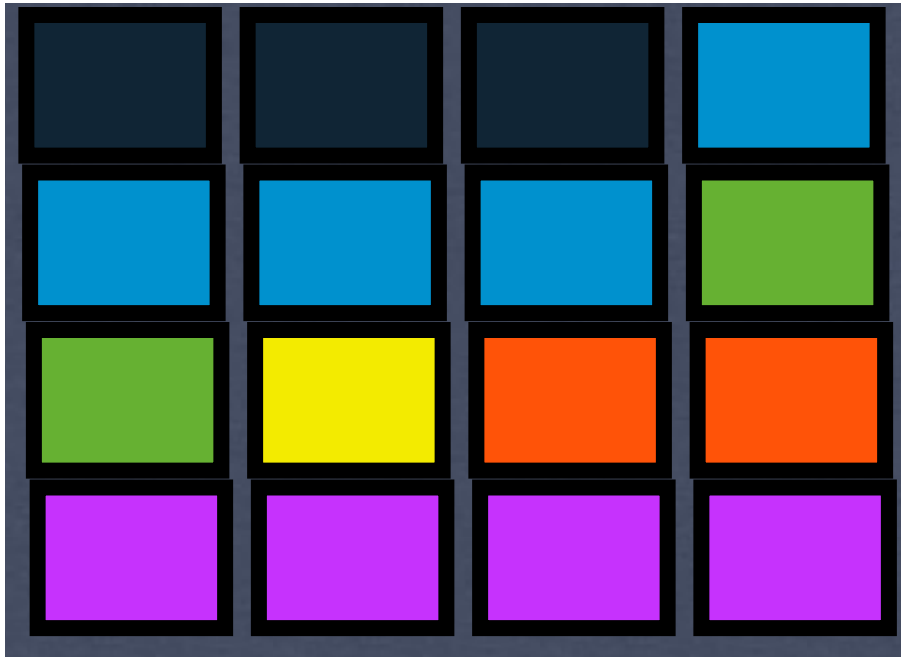
MEMORY

0-3

4-7

8-11

12-15



THE UNIVERSITY of ADELAIDE

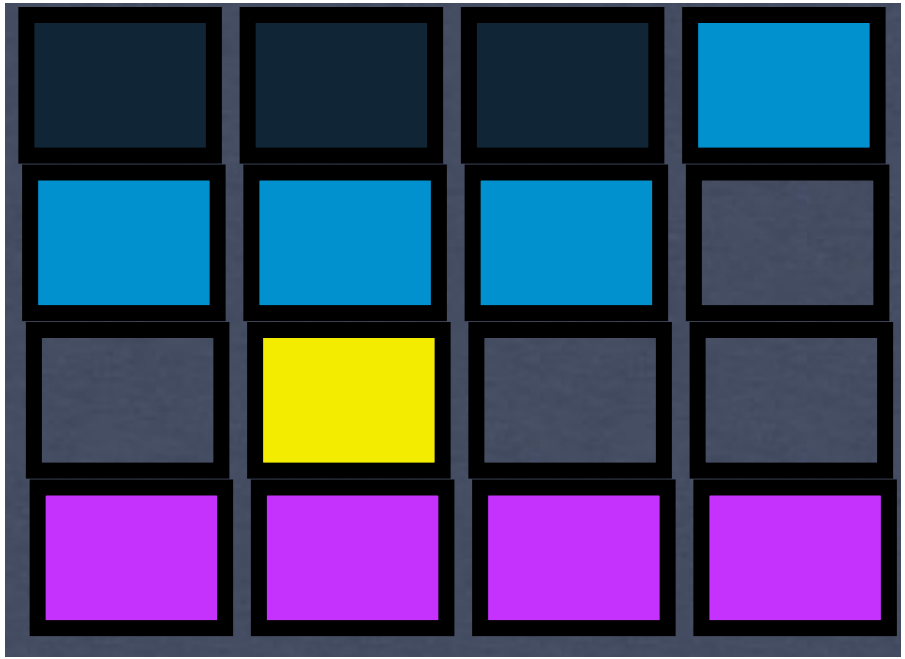
MEMORY

0-3

4-7

8-11

12-15



THE UNIVERSITY of ADELAIDE

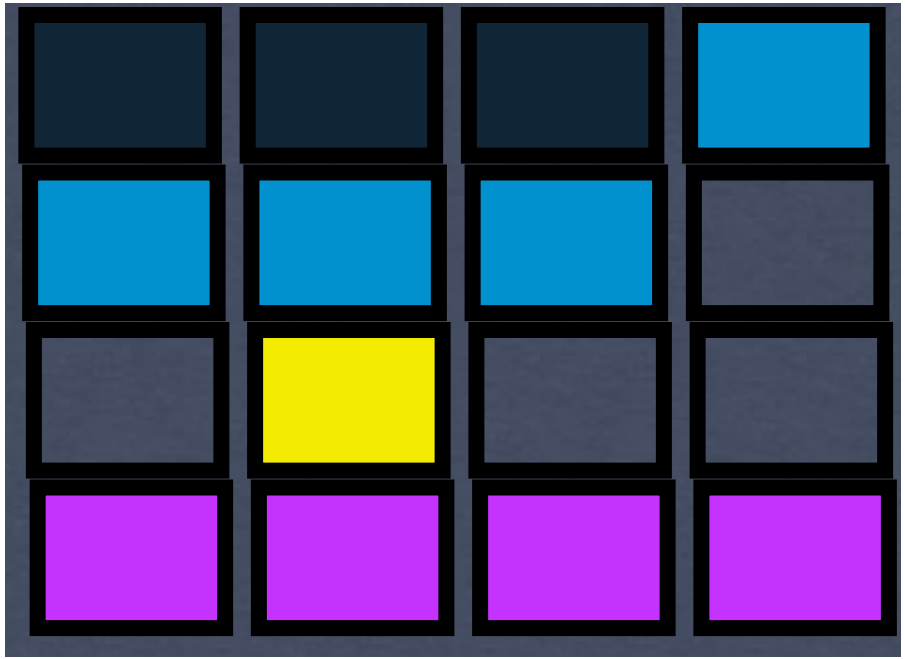
MEMORY

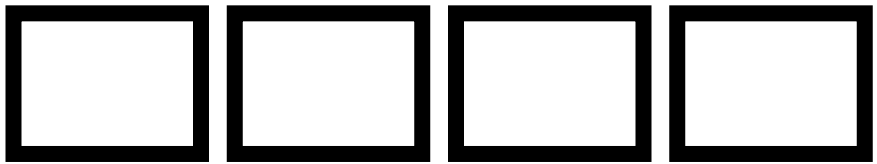
0-3

4-7

8-11

12-15





THE UNIVERSITY of ADELAIDE

Fragmentation

- **Finding a large enough contiguous area of free space requires us to defragment the memory space.**

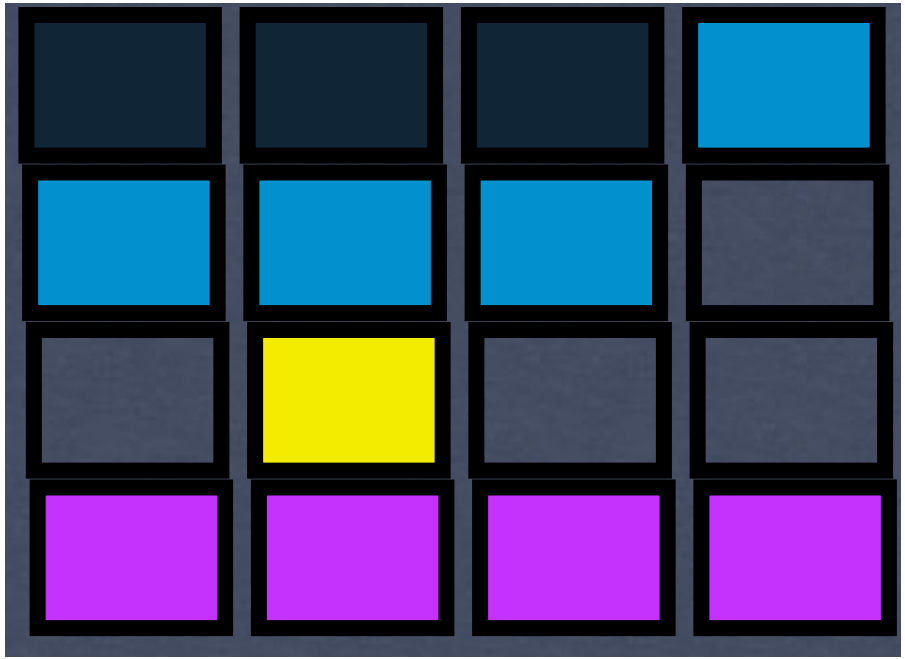
MEMORY

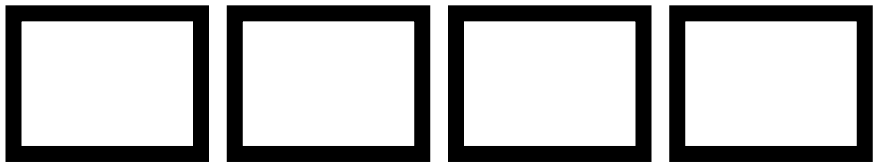
0-3

4-7

8-11

12-15





THE UNIVERSITY of ADELAIDE

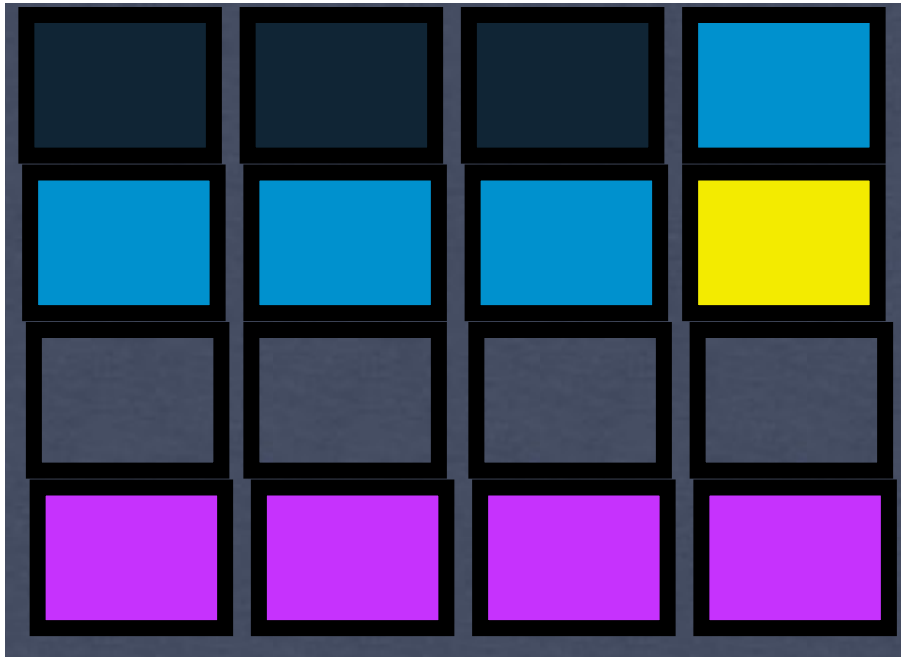
MEMORY

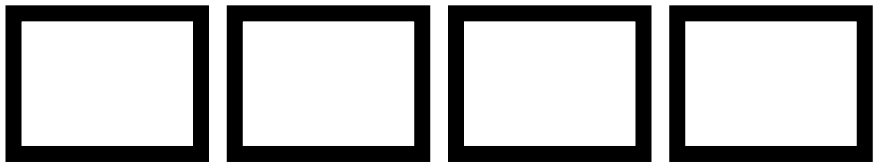
0-3

4-7

8-11

12-15





THE UNIVERSITY of ADELAIDE

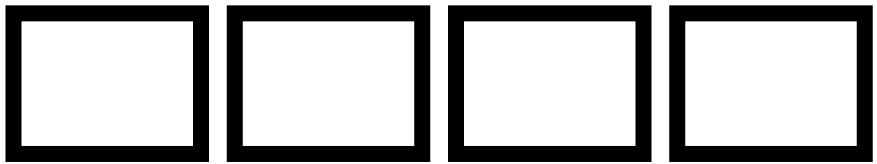
MEMORY

0-3

4-7

8-11

12-15



THE UNIVERSITY of ADELAIDE

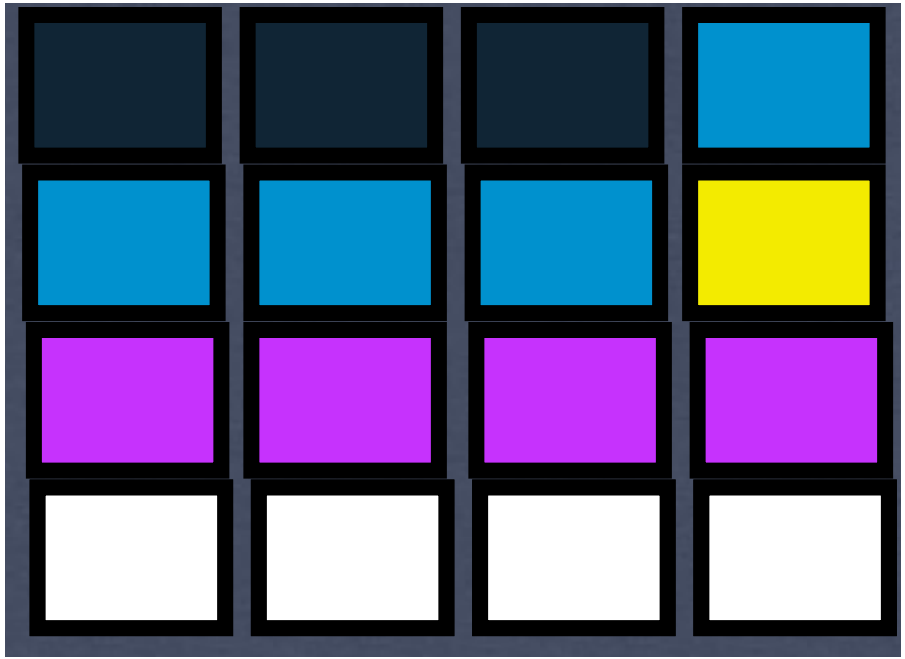
MEMORY

0-3

4-7

8-11

12-15



THE UNIVERSITY of ADELAIDE

Using bits

- **We don’t all work in binary.**
- **We want to store things like**
  - 4
  - 100.45
  - “Hello, World”
  - and so on

Integers

- **Integers are easy because every digit in a binary number can represent a power of 2.**
  1010
  is read as “2 to the 3rd power” plus “2 to the 1st power”, which is 8+2, so this is the number
  10

What?

- **Two questions immediately arise:**
  - How can I tell the decimal number 10 from the binary number 10?
  - Why is 2^0 on the right hand side?

Alignment

- Most data types are made up of more than one byte but, as we know, that’s less than one word.
- We should always align our memory access with the system's word size, around some sensible number of bytes.

Alignment examples

- Characters are stored in one byte, short integers (shorts) are stored in two bytes.
- On a 32-bit system, each word is four bytes.
- Whenever we try to pull the char and the short out of memory, the system will most likely pad it out with an extra byte to maintain alignment.

Bus errors

- The memory is accessed by words, but it is stored as bytes.
- If you address a byte that doesn't start a word and try to pull data out that is bigger than a byte, you will often generate a bus error and your program will crash.

Kinds of memory

- **There is more than one type of memory in the computer:**
  - Main memory (located in the computer)
  - Cache memory (located on the chip)

Caching

- If we've used a value once, chances are we'll use it again.
- Memory is fast but, compared to the chip talking to itself, it's slow.
- The cache stores frequently used data and gives a massive speed up when used correctly.

Types of Cache

- There is often a cache hierarchy, where Level 1 (L1) cache is the fastest, L2 is the next fastest and so on.
- Generally, the faster the cache, the smaller and it only stores the most current data.

How does it work?

- When we want to use some data, we check to see if it's in (one of) our cache(s).
- If it's there, then it's a hit and we retrieve it at high speed.
- If it's not there, it's a miss, and we get it (slowly) from main memory and then it gets stored in a cache.

Virtual Memory

- Memory used to be very expensive.
- Keeping memory images on (cheaper) disk drives made sense.
- Much slower but much larger.
- Memory was paged in and paged out as required, using a page table.

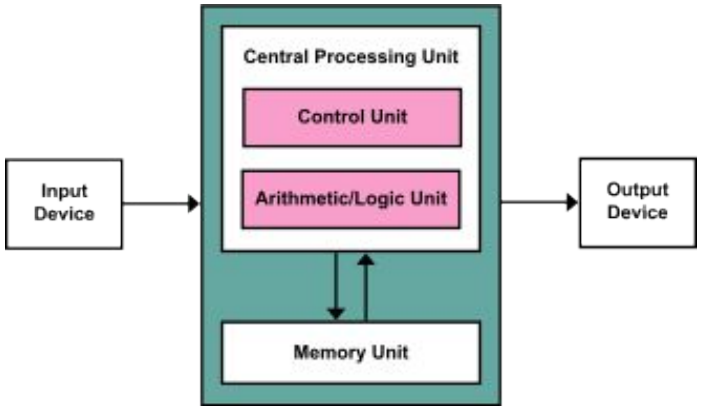
Page Faults

- Virtual memory gives us a virtual address space that is effectively larger than our real memory size.
- Each processor might have its own view of something the size of main memory, which is secretly stored on disk.
- If a program tries to access a part of the address space that is still on disk, we generate a page fault, and have to wait for the page.

THE UNIVERSITY of ADELAIDE

Von Neumann Architecture

- **Almost all of the computers you will use are Von Neumann architecture machines**



- **Most importantly for us, Code and Data are stored in the same memory. If it's not in memory, we can't perform operations on it.**

THE UNIVERSITY of ADELAIDE

How do we use this?

- **If Code and Data are in the same memory, we probably don't want to accidentally overwrite our code or data.**
- **The C++ compiler (which turns C++ into executable code) makes life easier for us.**

Basic layout

THE STACK AND THE HEAP

WHERE VARIABLES THAT ARE ACCESSIBLE TO EVERYTHING (GLOBAL) RESIDE

WHERE THE COMPILED CODE RESIDES, USUALLY READ ONLY

THE UNIVERSITY of ADELAIDE

Next Topic

More on memory and introduction of pointers and arrays.