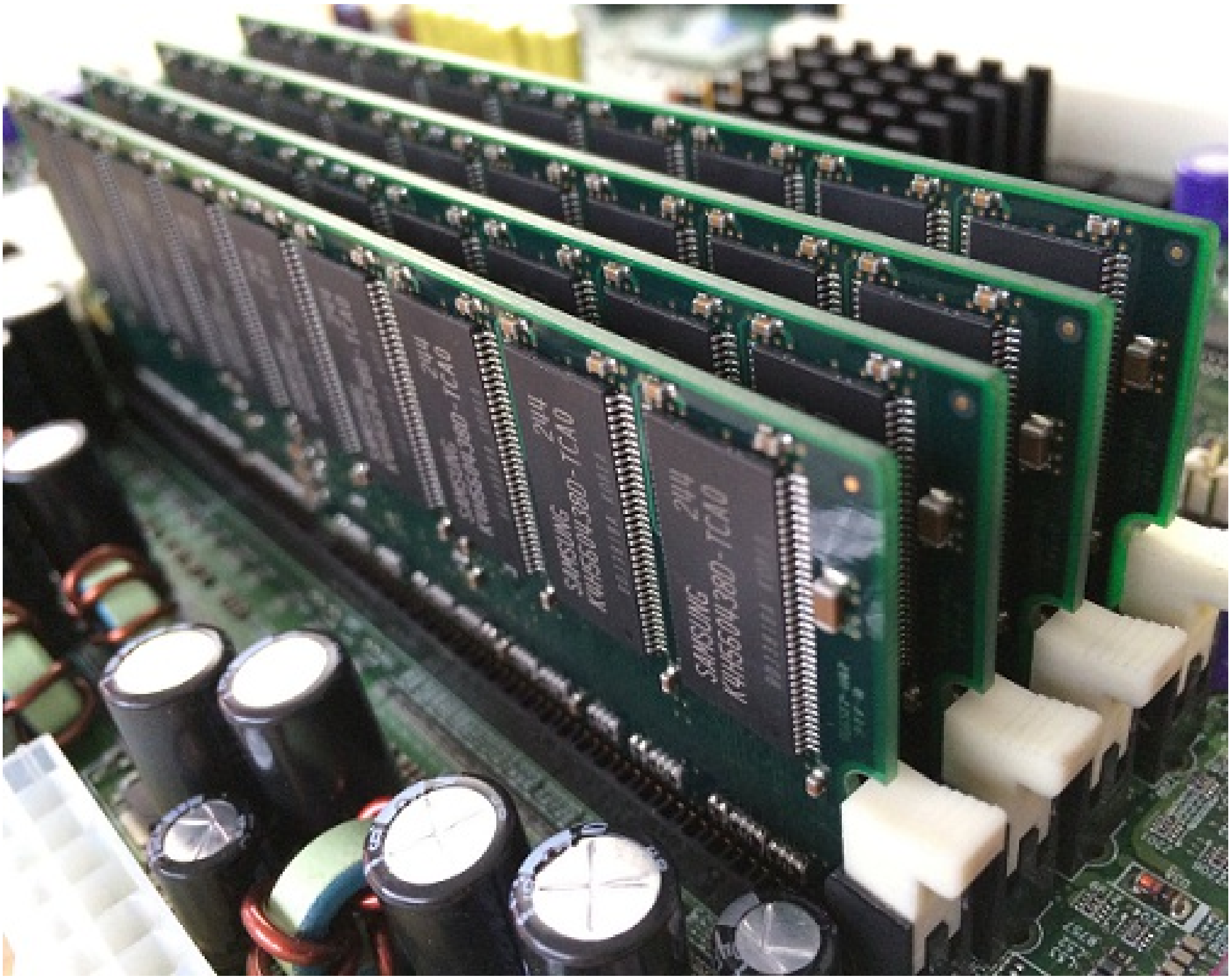
Stack in Memory Management



Most important application of stacks: stack memory

- Memory RAM
- A call stack is an abstract data type that stores information about the active subroutines / methods / functions of a computer program
- The details are normally hidden and automatic in high-level programming languages
- Stores temporary variables created by each function

Call stack

Every time a function declares a new variable it is pushed onto the stack

Every time a function exits all of the variables - pushed onto the stack by that function - are freed all of its variables are popped off of the stack // and lost forever

Local variables: they are on the stack, after function returns they are lost

Stack memory is limited

Heap memory

The heap is a region of memory that is not managed automatically for you

This is a large region of memory // unlike stack memory

C: malloc() and calloc() function // with pointers

```
malloc()
```

```
calloc()
```

Java: reference types and objects are on the heap

We have to deallocate these memory chunks: because it is not managed automatically

If not: memory leak

Slower because of the pointers

Stack memory VS heap memory

Stack memory	**Heap memory**

Limeted in size	No size limits
Fast access	Slow access
Local variables	Objects
Space is managed efficiently by CPU	Memory may be fragmented
Variables cannot be resized	Variable can be resized - realloc()

Bruna Santos - March 29, 2018 11:22 am