**Principle of programming review**

By Richard Liao

Pointers ::

—Pointer is the memory address.

—Pointer variable holds the address of a value somewhere in the memory

—Size of pointer depends on machine, typically 64 bits.

—Benefits of pointer-based data structures.

—\* and / are not pointer’s operator.

Memory ::

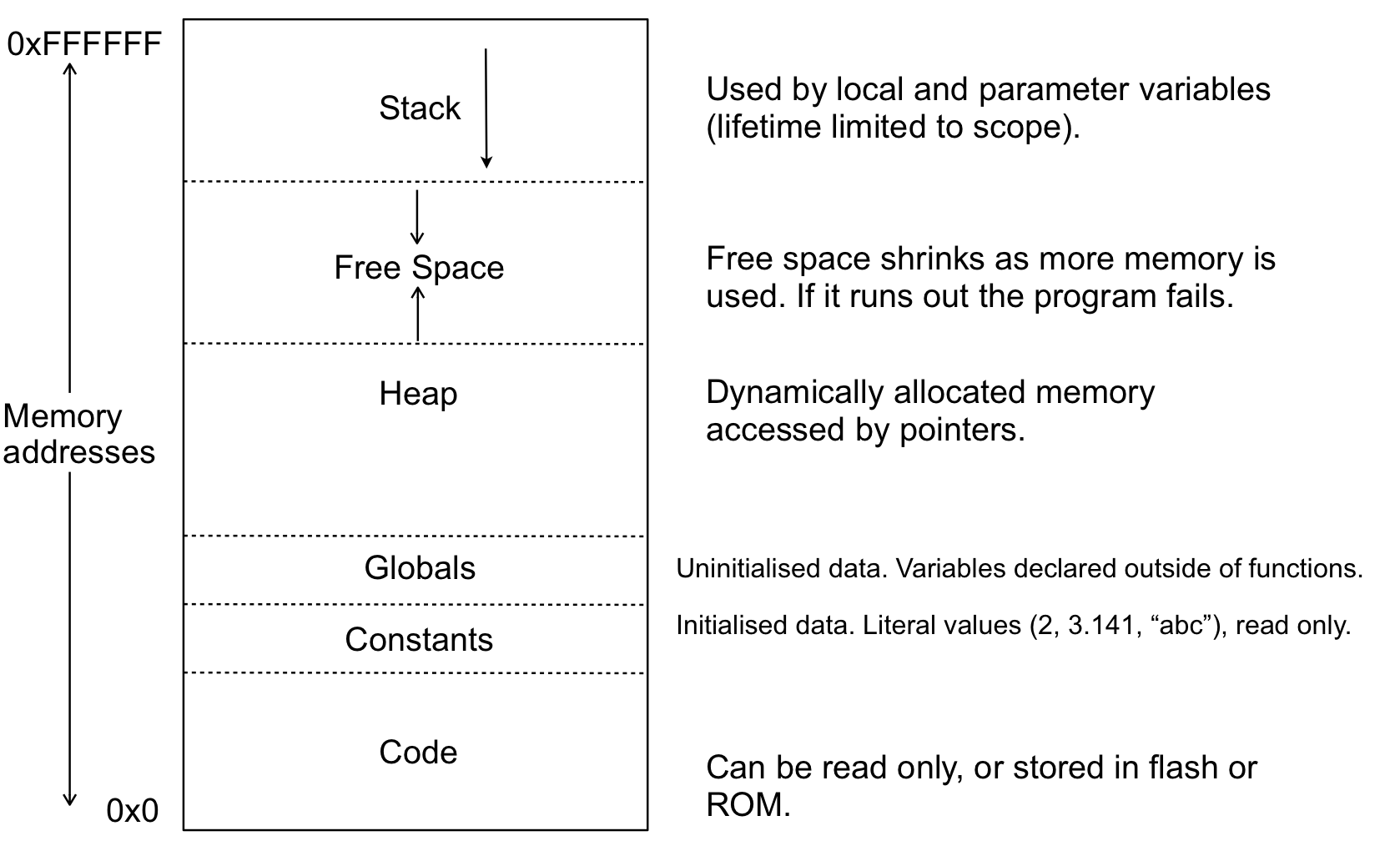
—(RAM)Random access memory|volatile(content lost when not powered)

—(Rom)Read only memory|non-volatile

—(EEROM)Erasable read only memory|non-volatile

—Flash memory|non volatile

—Cache memory|volatile



-Two many function calls without returning will exhaust the memory because the stack grow too large

Arduino ::

—Arduino has no operation system

—A Bootloader allows program to be uploaded via the serial over the USB port

THREADS ::

—A thread is a sequence of control

(A sequence of function calls in C)

—C program can be multi-threaded

—Can have multi sequence of control

—Within the same process

—All threads share the same memory space, and that is why multi-thread is difficult to handle

Object File ::

—Called .o files(on Unix)

—Usually each .o file Is compiled into a .c file with the same name

—Contain relocatable binary code

—Processor and operation system specific

Linking ::

—Join all the .o and library files into a single executable program

—The address of all functions and local variables must be resolved (This is called identified and “linked up”)

—Every variable and function must be defined once

—Any undefined functions or variables will result in a link error

—All modern Unix implementations use dynamic linking

—linking Is done at runtime as functions in libraries are called

Library ::

—Contains precompiled but not linked code

—Intended to be reusable , for example, the standard C library

—Name lib<something>.so

Array and memory ::

char word[] = “hello”

—“hello” is a string literal stored in the constants section

of memory (read only).

—word[] is allocated memory In the stack if local, else in the globals section

—Initialisation copies the literal string Into the array memory

(Hence it is ok to change the value like word[1]=‘#’)

Char \*word=“hello”

—Word is a pointer to the read only memory

—So \*word+1 = ‘#’ is not allowed due to trying to write to a read only memory

Dynamic memory allocation

—Allocate memory for variables, arrays and data structures on the heap

—Memory will remain allocated until explicitly freed, regardless of function calls and scope

—The programmer has to manually manage the memory allocation

—Allocate when needed

—Free when no longer needed(Forget will cause memory leak)

The linked list ::

—Each node In the list store a value and a pointer to the next node

—The last node store a null pointer ‘\0’ or zero or NULL

—Struct allows a group of variables to be composed together into a single unit

—Implementation: Slide 9 Page 47

—Code: <https://paste.ofcode.org/MifviQuzbGvYLw3tEaRLXm>

Scope ::

—A name must be declared before it can be used

—The scope of a name determines where in the source code it is valid

—Use of a name not “in scope” potentially results in an error

—File scope:

-Declared outside of a function

-Within a file

-sort of global scope

-Start from the point of declaration

—Local scope:

-within a compound statement

-Start from the point of declaration

—Parameter variable can only be used inside the function body(a compound statement)

—Only exist while the function is running

—A parameter variable is a kind of local variable with special initialization rule

—Lifetime of local/parameter variable determined by scope

-Created when scope entered at runtime

-Destroyed when scope existed

-The stack behaviour

Heap Data

—The lifetime of data in the heap is determined by the programmer

-Using malloc/free

—Accessible only via pointers

-But pointer variables have their own scope and life time

—Lose a pointer will cause the data to be inaccessible but the memory remains allocated