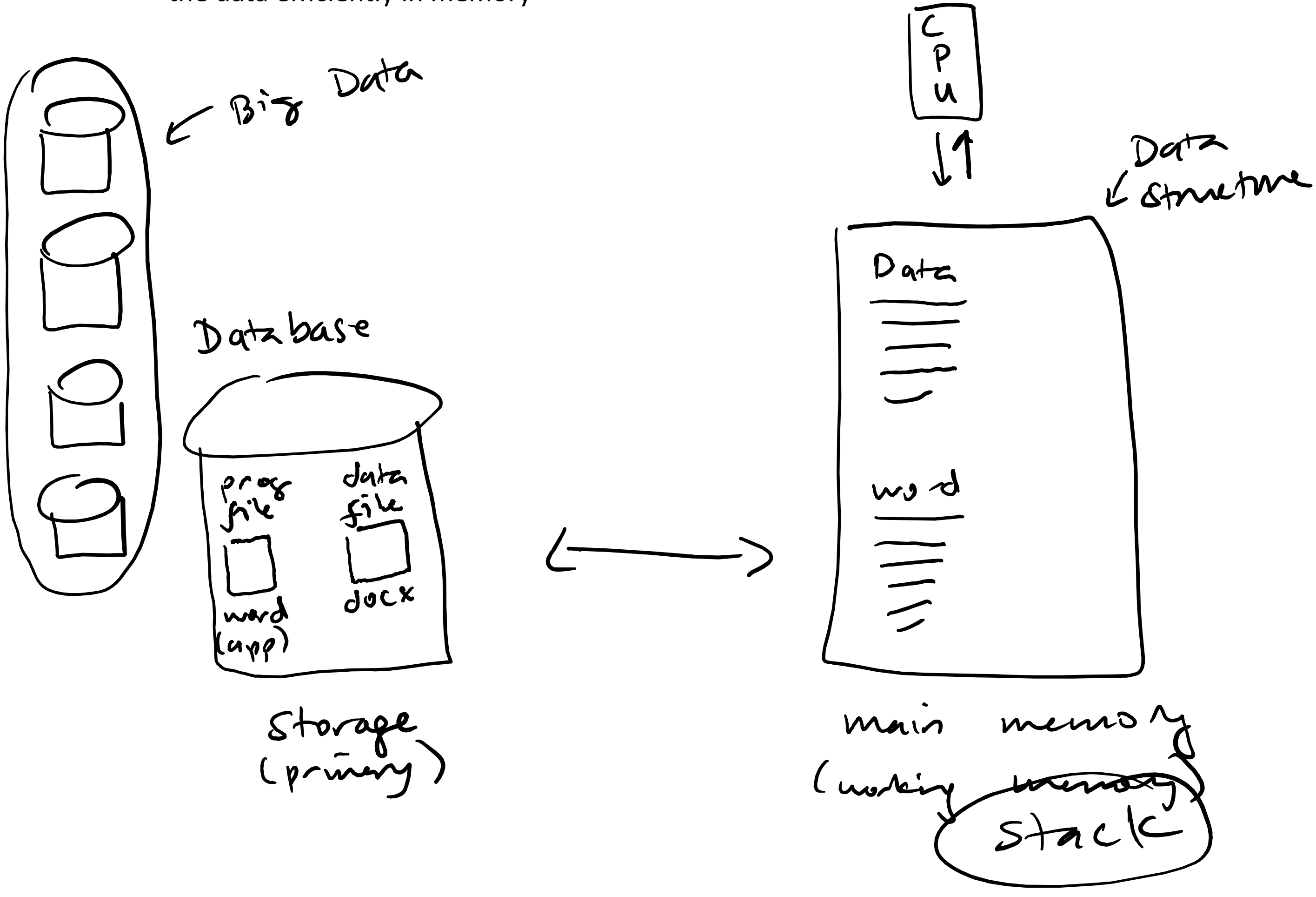
Data Structures Introduction:

1. Data Structures
2. Database – arranging data in primary storage so it can be retrieved or accessed by applications easily
3. Datawarehouse – operational and legacy data, can be kept on array of storage drives
4. Big Data

Data Structures: arrangement of the collection of data items so you can perform operations on the data efficiently in memory

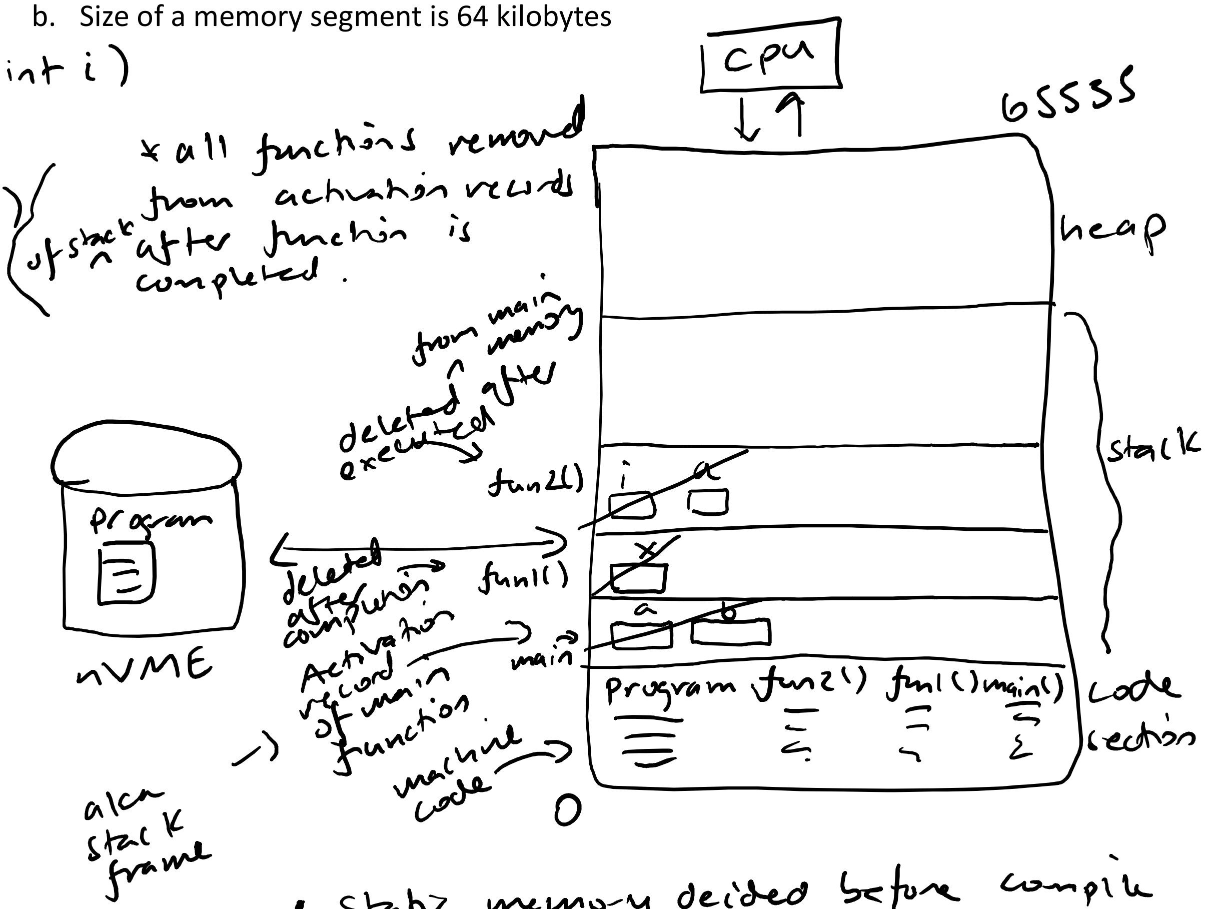


Static & Dynamic memory allocation:

1. About Main Memory
    a. Memory divided into bytes, each byte has an address (addresses are linear, i.e. only one coordinate, not x & y)
    b. Size of a memory segment is 64 kilobytes

```
void fun2(int i)
{ int a; }

void fun1()
{ int x;
  fun2(x); }

void main()
{ int a;
  float b;
   {
  fun1(); }
```

* all functions removed from activation records of stack after function is completed.



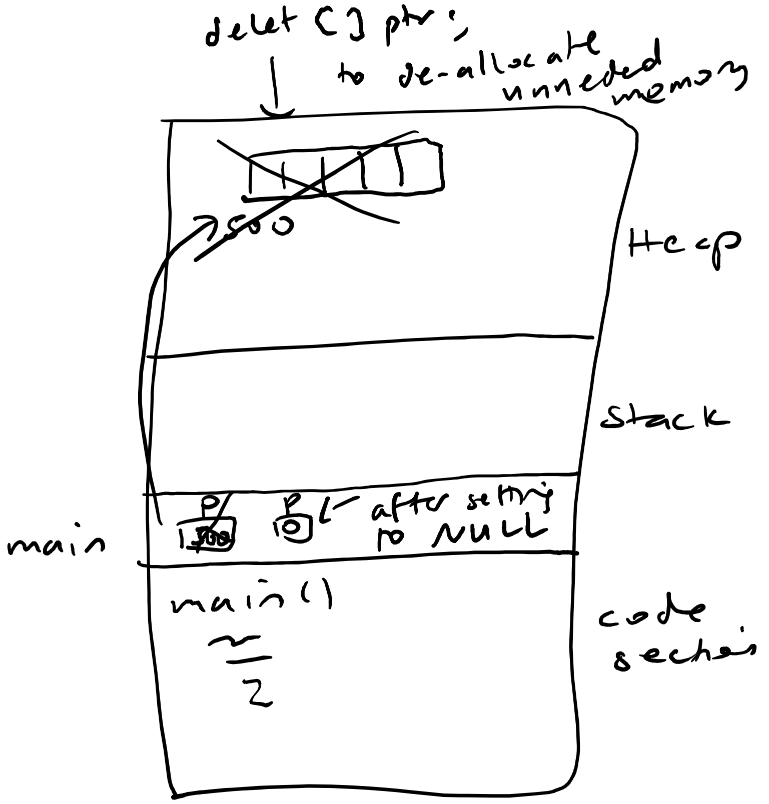
- static memory decided before compile time
- stack automatically created when function starts & ends.

2. How a program uses memory
3. Static Allocation
4. Dynamic Allocation

<u>Heap</u> - used as resource

- use mem when required, released when not

- program <u>can't</u> access <u>heap</u> memory directly
  - accessed using <u>pointer</u>



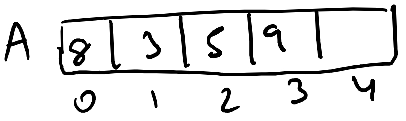
```
void main ()  ← 4 bytes
{   int *ptr;
                        ← C++ code
    ptr = new int[5];
                                  ← C code to access memory
    ptr = (int *)malloc(4 * 5);

    delete [] ptr;  ← de-allocates
                       memory

    ptr = NULL;  ← no longer points,
       ↓            but previous still
    sets pointer    in memory, need
    value to 0      to delete to de-allocate
                    memory
}
```

- <u>memory leak</u>: if you don't delete no longer needed memory, new memory can't be stored, so you lose that new data

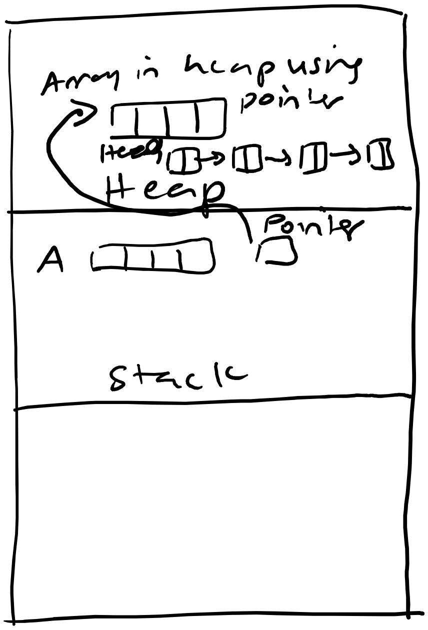
(1) Physical & (2) Logical Data structures:

(1) Physical Data Structures: -> Stores data in memory

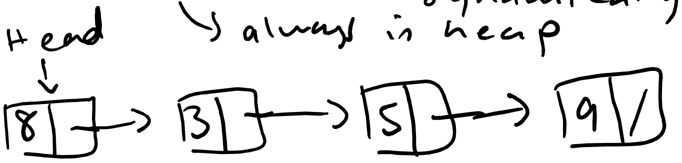
1. Array -> fixed size, static
   -> stack or heap



(heap using pointer)



2. Linked list - can grow & reduce dynamically
   -> always in heap



(2) Logical Data Structures: implemented using physical data structures (array &/or linked list)

linear
1. Stack : LIFO
2. Queues: FIFO

non-linear
3. Trees : Hierarchy
4. Graph : link between nodes

5. Hash table
Tabular or linear

Abstract Data Types:

1. Representation of Data → how data is stored

2. Operation on data

add (index, element)

List: 8, 3, 9, 4, 6, 10, ~~12~~, ~~15~~, ~~20~~

20 12 15

0 1 2 3 4 5 6, 7, 8

↑

20

Data: 1. space for storing element. → 1. array → 2. linked list

2. capacity

3. size

Operations:

add(x)
remove()
search(key)
⋮

also called append(element)

- add (element) → add element to end of list
- add(index, element) → add element to specific index
- remove(index) → remove one element from given index
- set(index, element) → change element @ specific index
  ↳ also called replace(index, element)
- get(index) → find what element is in specific element
- search(key) → search for element to find its element
  ↳ also called contain(key)

- sort () → arrenge elements so you can sort them in order