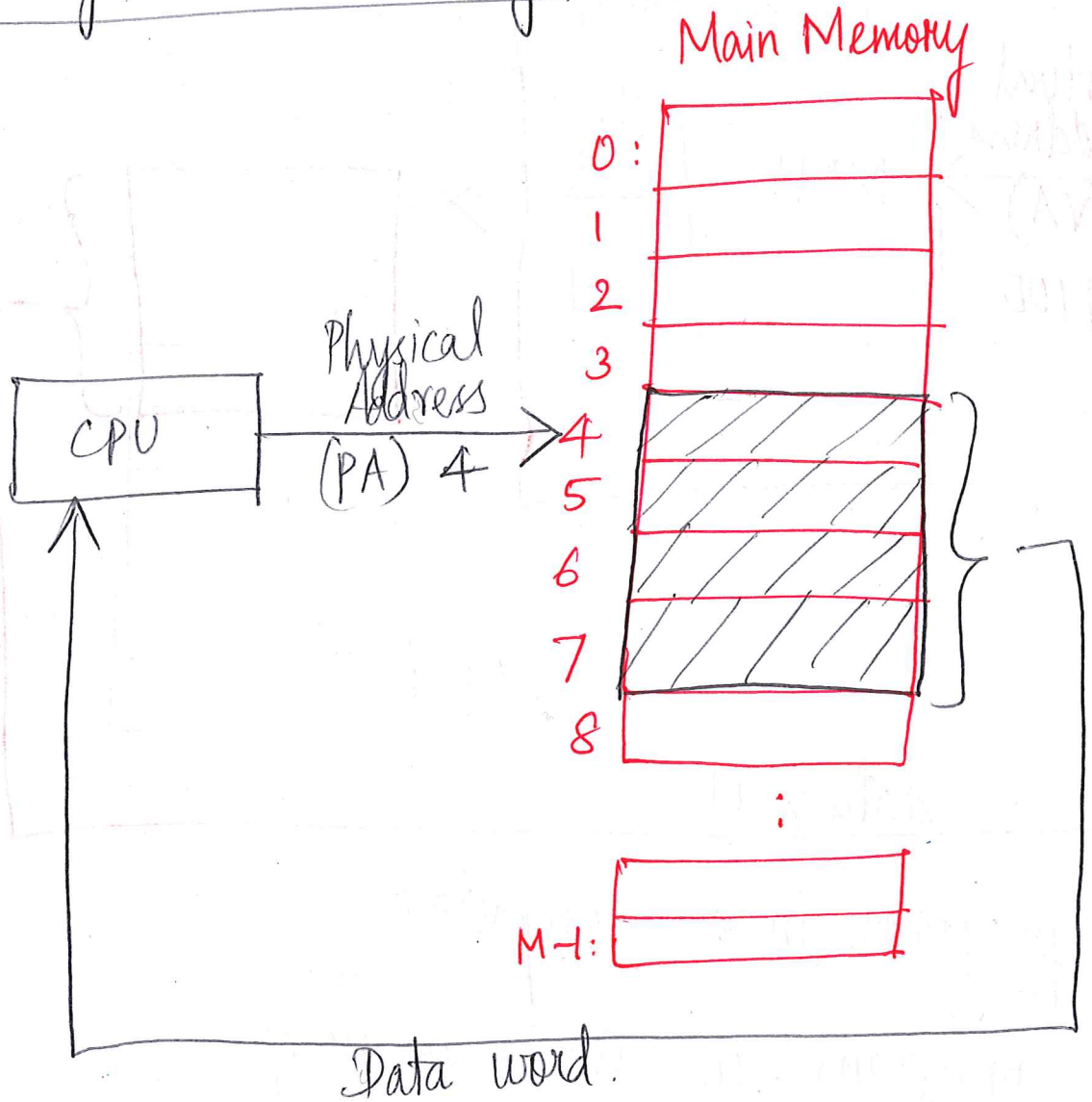
Virtual Memory

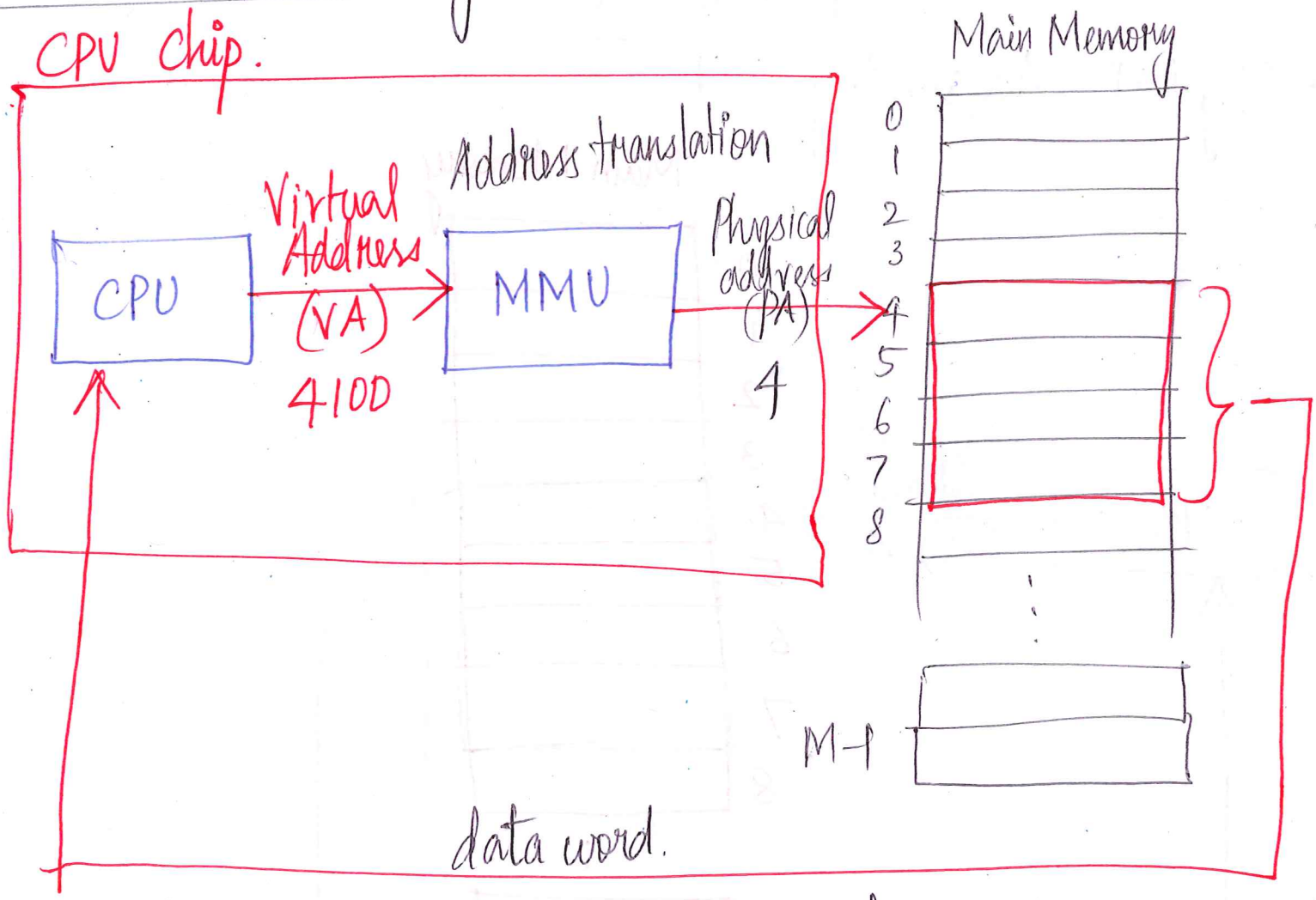
Lecture - 30.

* Physical Addressing (1)



(X) Address Space

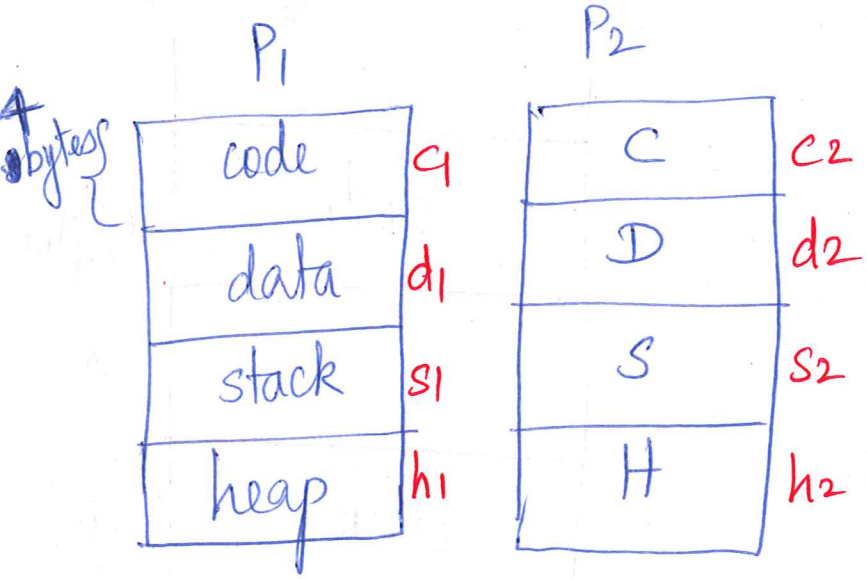
* Virtual Addressing



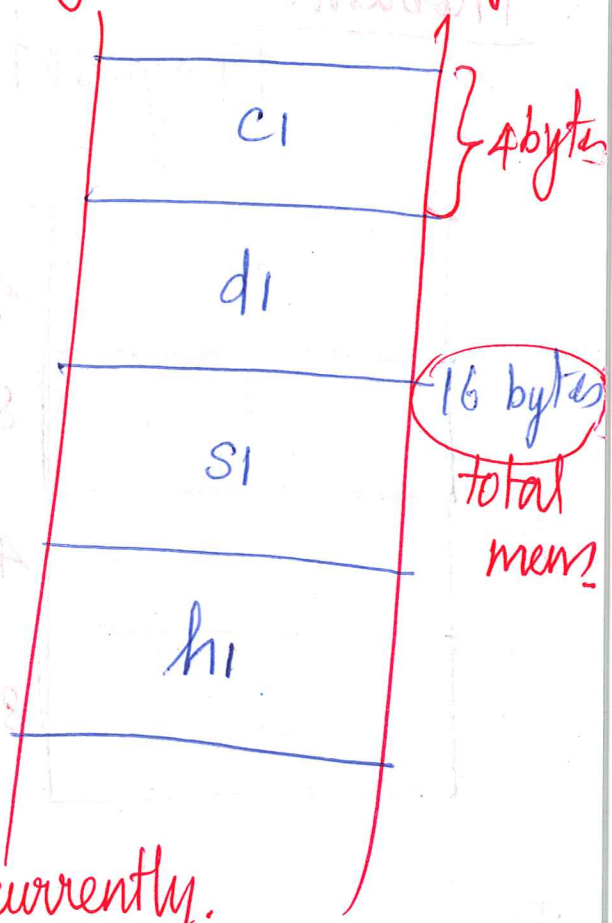
* Executing a program in a computer

1. Entire program in main memory.
2. Segmentation.
3. Paging.

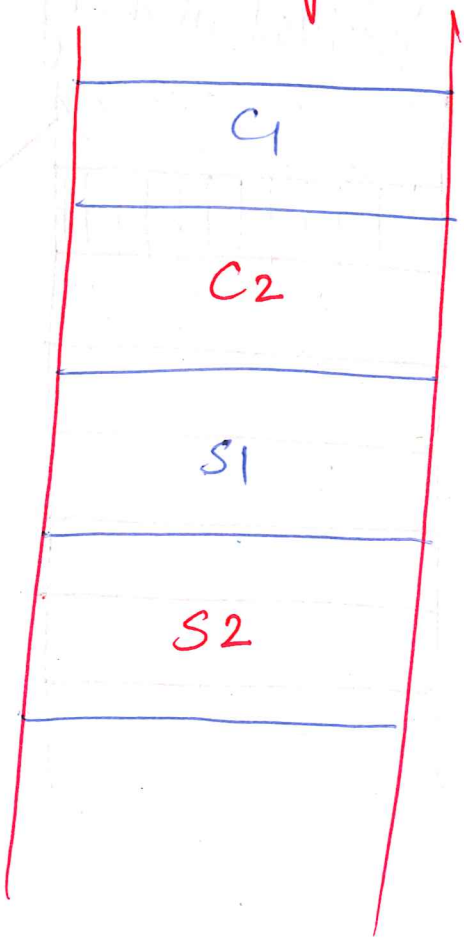
Segmentation



Only P_1 is executing

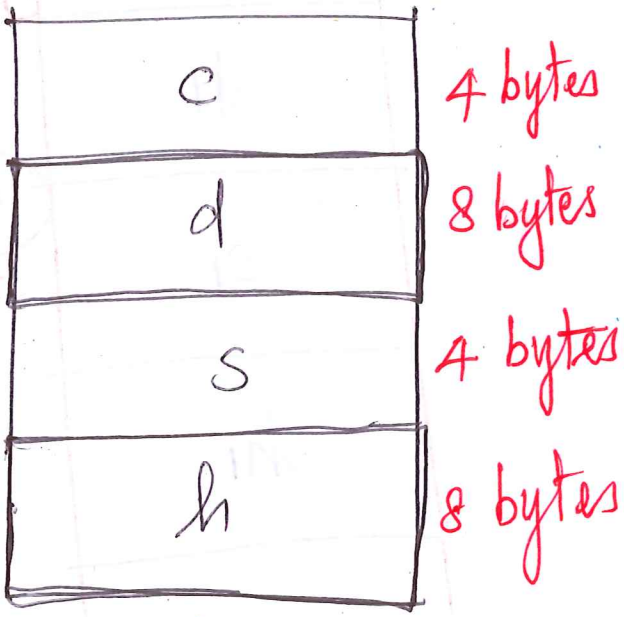


If P_1 and P_2 executing concurrently.

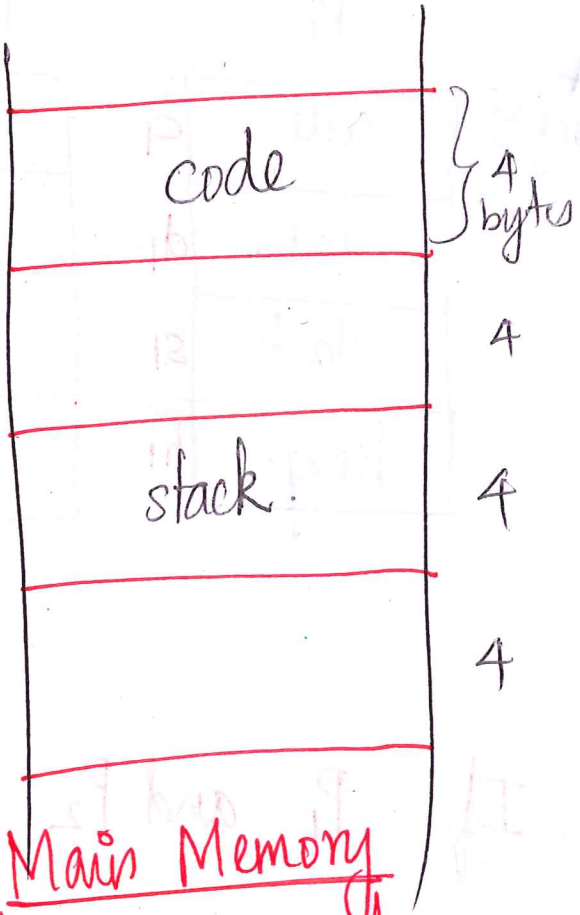


Problem:

Fragmentation

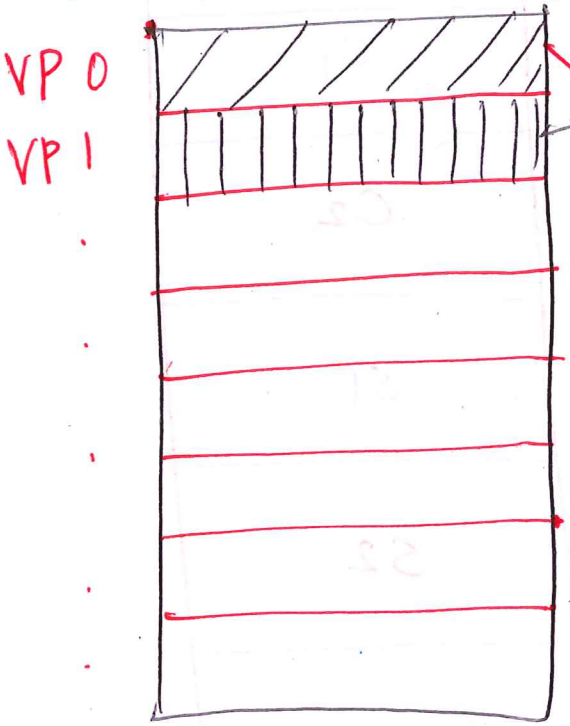


Memory

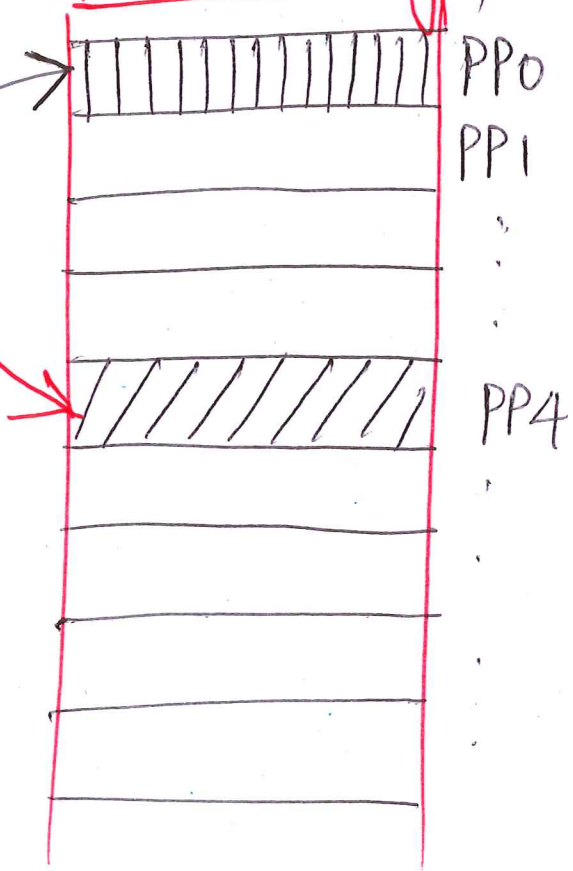


Paging

Virtual memory

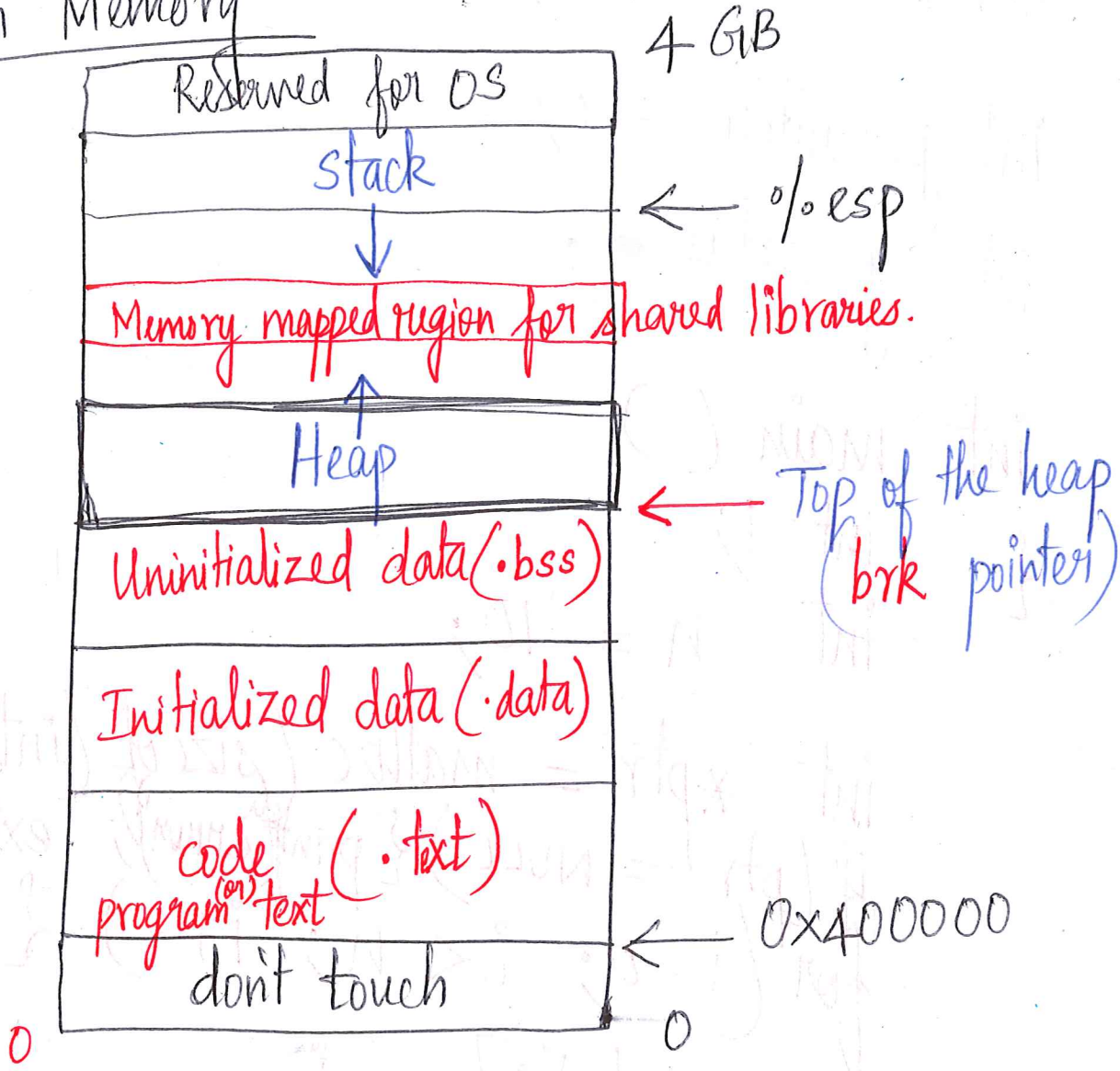


Main Memory



Dynamic Memory Allocation. (5)

Program Memory



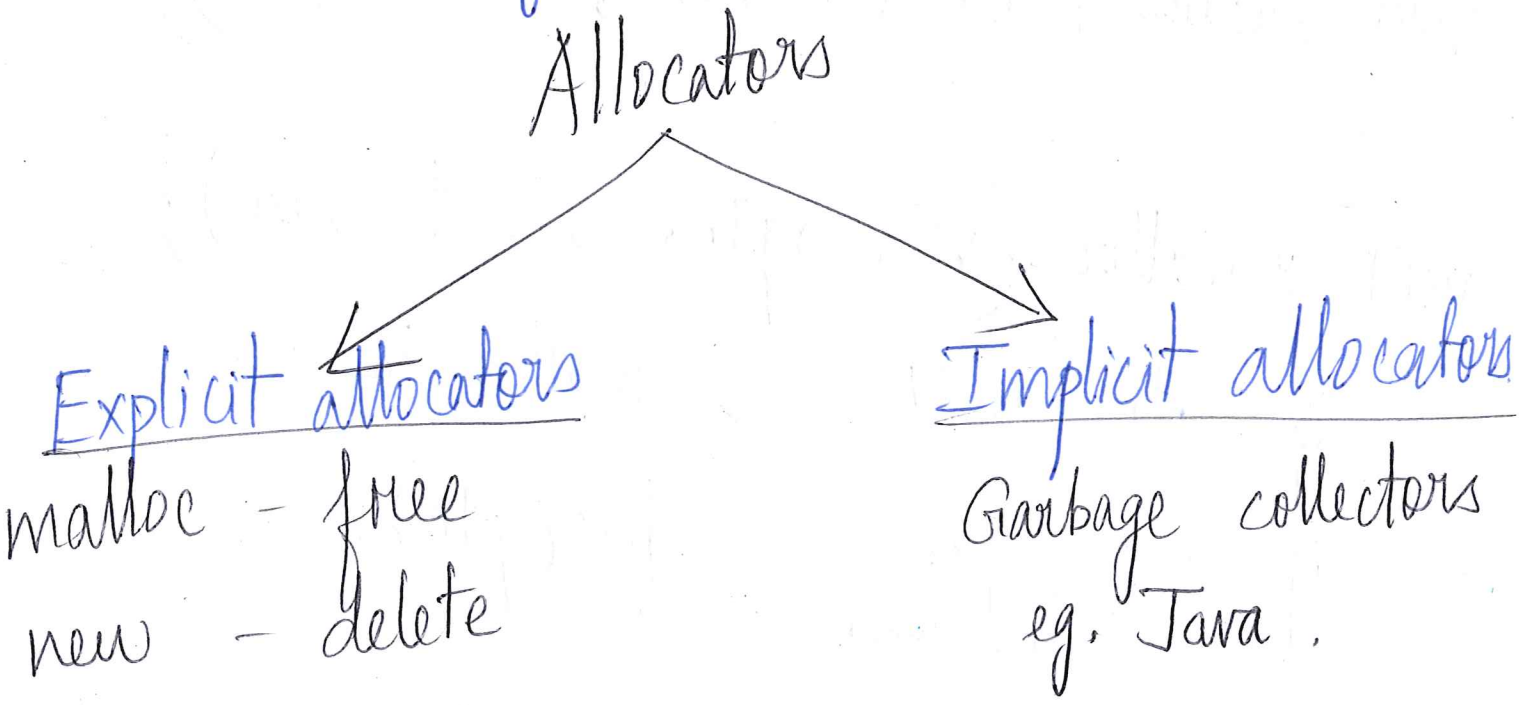
'.bss = Block started by Symbol.

```
#include <stdio.h>
#include <stdlib.h>

int g_counter = 0;
int g_status = ;

int main ()
{   int i;
    int n = 10;

    int *ptr = malloc (sizeof (int) * n);
    if (ptr == NULL) { printf("error"); exit(1); }
    for (i=0; i<n; i++) {
            ptr[i] = i;
    }
    free(ptr);
}
```

Allocator

A dynamic memory allocator maintains an area of a process's virtual memory known as the heap.

* heap → collection of various-sized blocks.
* block → contiguous chunk of virtual memory.
* block → allocated / free.



Functions to access the heap memory.

```
#include <stdlib.h>

void *malloc(size_t size);
                ↓
            unsigned int
```

Returns: ptr to allocated block if OK, NULL on error.

```
void free(void *ptr);
```

Returns: nothing.

```
free(p);
free(p);   } undefined behaviour.

free(NULL); ⇒ no operation is performed.
```

```
void *calloc(size_t nmemb, size_t size);

void *realloc(void *ptr, size_t size);
```

if ptr == NULL ⇒ malloc(size)

if size == 0 && ptr != NULL ⇒ free(ptr).

```
#include <unistd.h>

int brk(void *addr);

void *sbrk(intptr_t increment);
```

intptr_t → typedef of long or int

brk() and sbrk() change the location of the program break.

brk() – sets the end of the "data segment" (or) top of the heap to the value specified by addr.

"data segment" = heap + data in traditional UNIX.

Returns: 0 on success
-1 on error.

sbrk() – increments the program's data space by 'increment' bytes.

Returns: On success, returns the previous program break.
On error, (void *) -1 is returned.

(void *) -1 → 0xFFFFFFFF

Address Space

An ordered set of non-negative integer addresses.

eg. 32-bit address space

$\{0, 1, 2, \ldots, 2^{32}-1\}$

In a 32-bit address space, there are 2^{32} addresses.

Linear address space – if the integer addresses are consecutive.

Address space

Virtual Address Space

$\{0, 1, 2, \ldots N\}$

$N = 2^n$

Physical Address Space

$\{0, 1, 2, \ldots M\}$

$M = 2^m$

(X) Each byte of main memory has a virtual address chosen from the virtual address space, and a physical address chosen from the physical address space.