void Pointer

- void pointer is also known as generic pointer.
- because in void pointer, we can store address of any type of variable.

- we can assign address of any variable to void pointer directly (without typecasting).
- at the time of derefeancing, we need to type cast void pointer by respective pointer of data.
- eg.

```
int num;
void *ptr = &num;
*(int *)ptr;
```

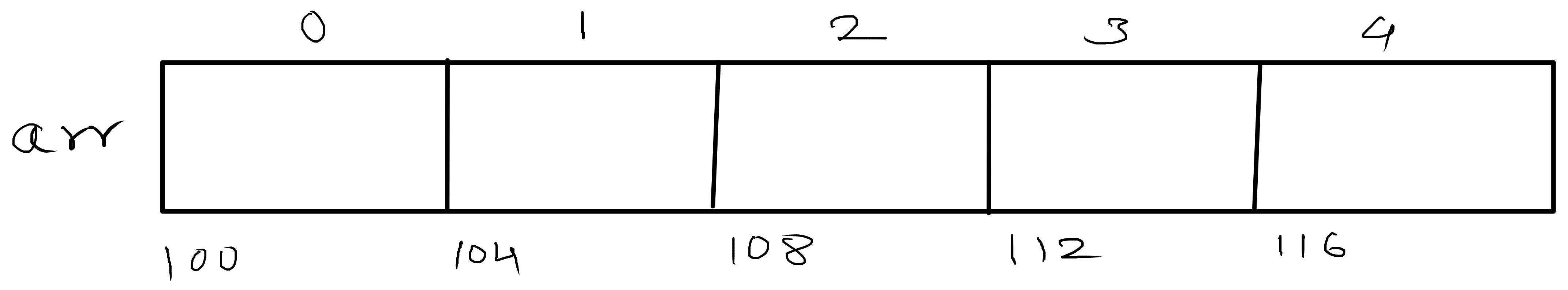
- sizeof(ptr) = sizeof(void *) = 8 bytes
- void pointer do not have scale factor thats why we can not directly do pointer arithmetic on it.

- NULL ---> (void *)0

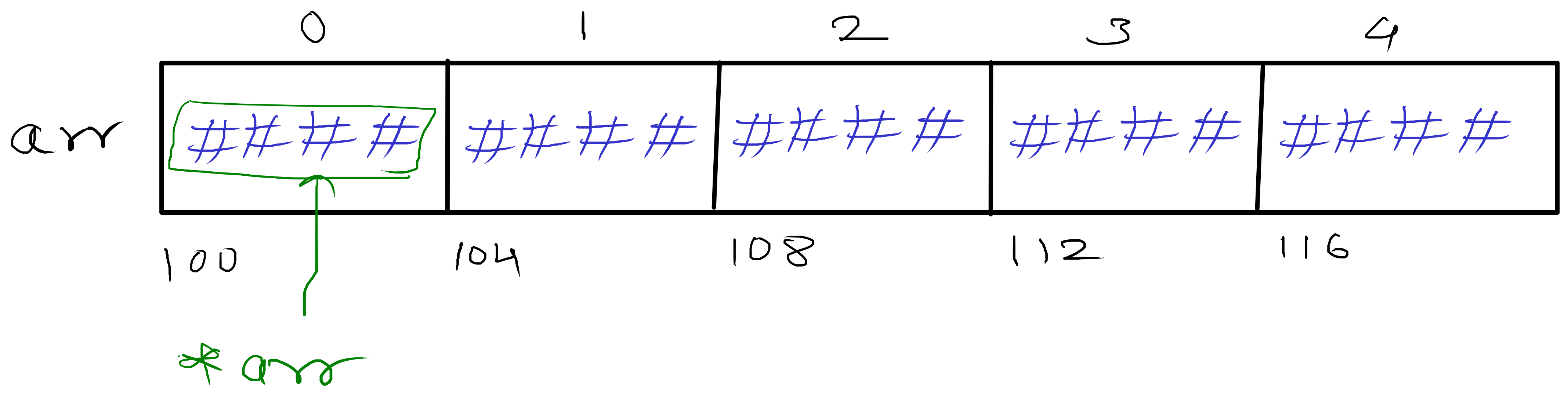
```
#define NULL (void *)0
```

- void pointer is used for generic programming (qsort, memset, memcpy, memmove)
- void pointer is used in dynamic memory allocation (malloc, calloc and realloc)

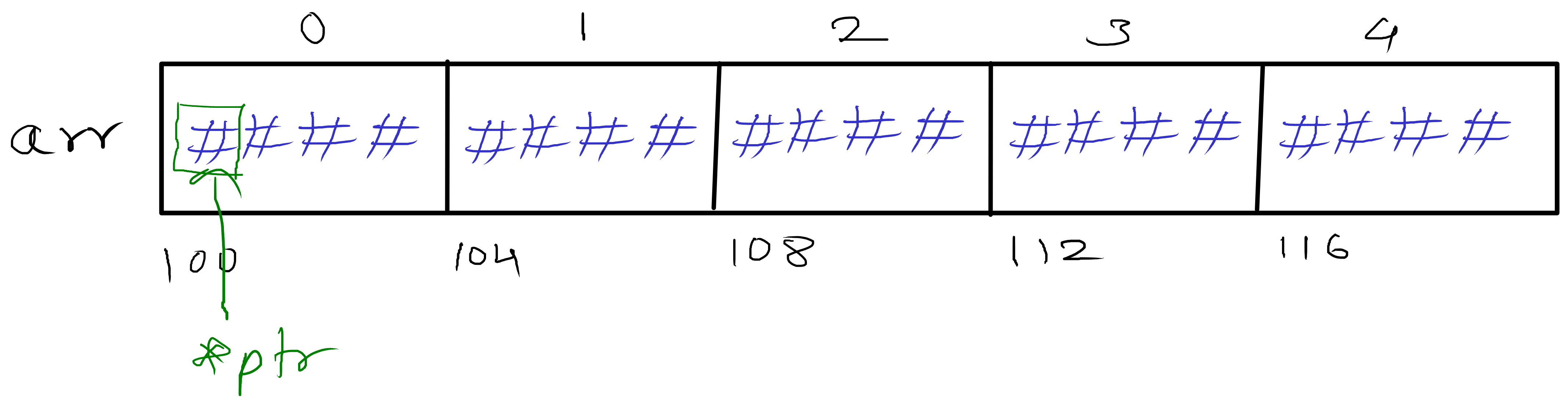
int arr[5];

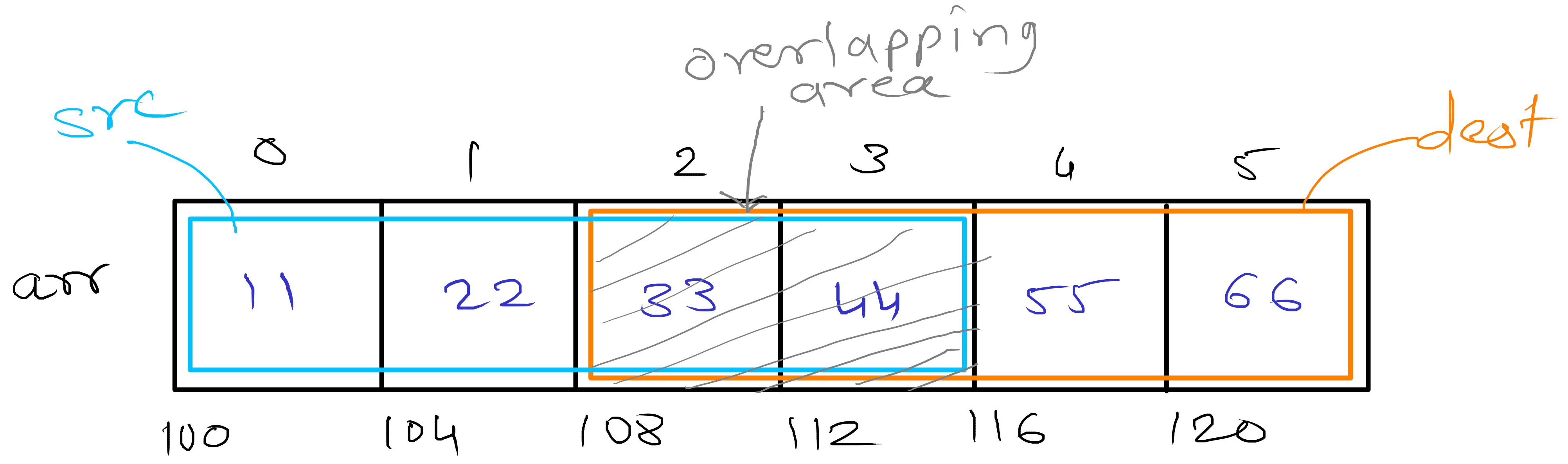


memset(arr, '#', sizeof(arr))



char *ptr = (char *)arr;





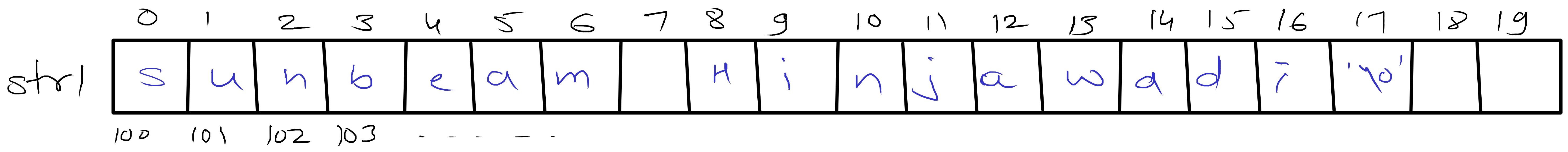
memmove(arr+2, arr, 16)

dest = arr+2 = 108

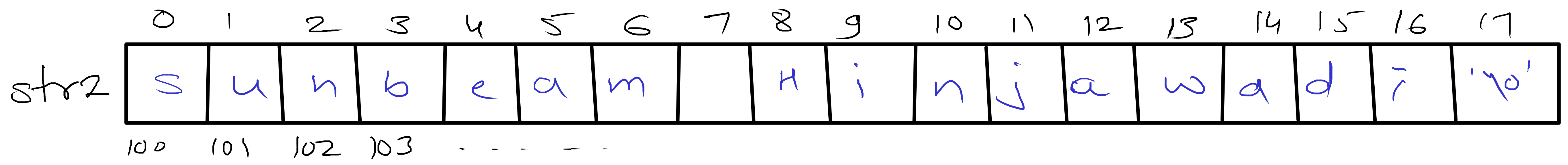
src = arr = 100

n = 16 bytes

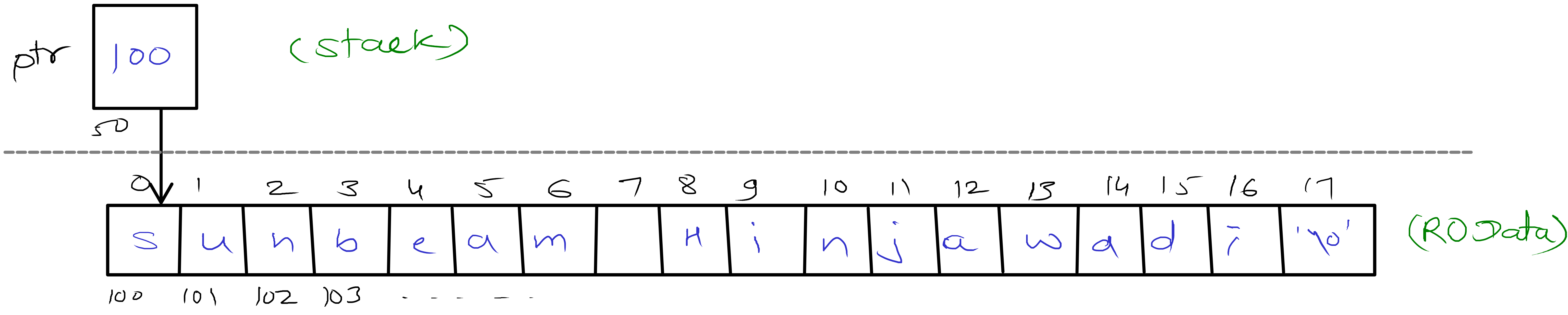
char str1[20] = "sunbeam Hinjawadi"; (stack)

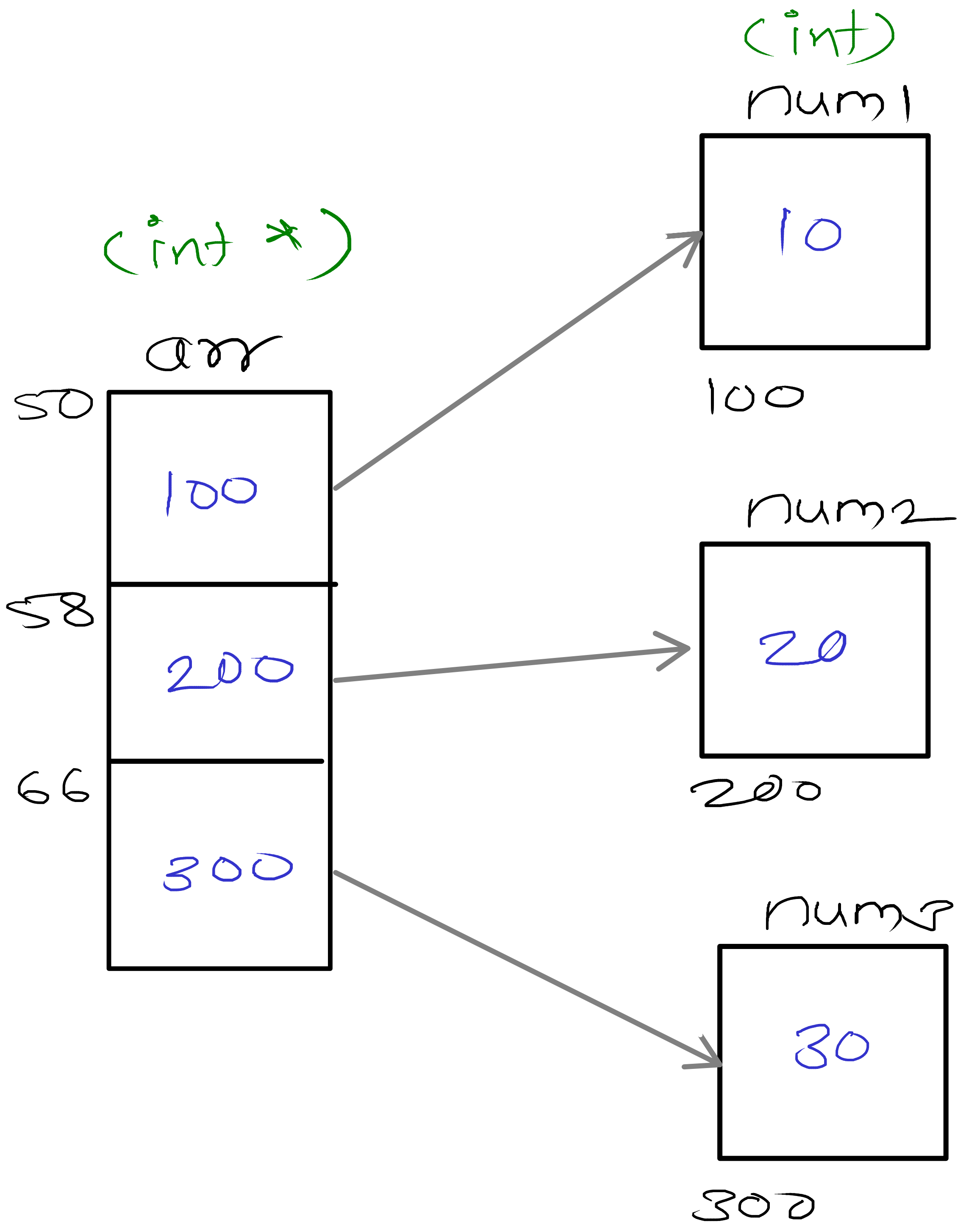


char str2[] = "sunbeam Hinjawadi"; (stack)



char *ptr = "sunbeam Hinjawadi";





```
int num1=10, num2=20, num3=30;
int *arr[3]={&num1,&num2,&num3};
```

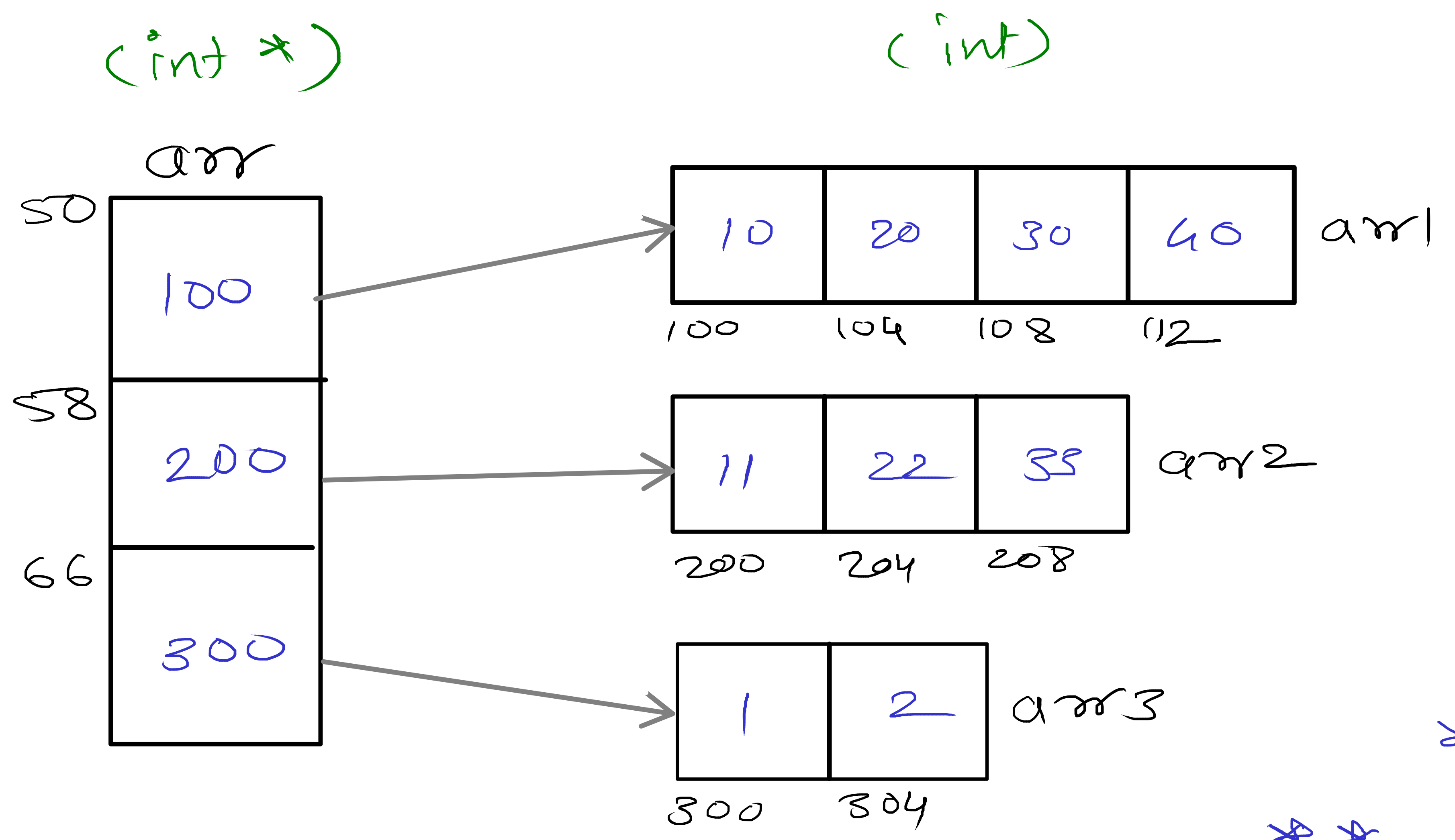
arr[0] => 100
arr[1] => 200
arr[2] => 300

*arr[0] => 10
*arr[1] => 20
*arr[2] => 30

arr+0 = 50
arr+1 = 58
arr+2 = 66
*(arr+0) = 100
*(arr+1) = 200
*(arr+2) = 300
**(arr+0) = 10
**(arr+1) = 20
**(arr+2) = 30

arr[1]+2 = 200+2
= 200+2*SF(200)
= 200+2*4
= 200+8
= 208

```c
int arr1[]={10,20,30,40};
int arr2[]= {11,22,33};
int arr3[]={1 , 2 };
int*arr[]={arr1,arr2,arr3};
```



arr = 50

arr[0] = 100

arr[1] = 200

arr[2] = 300

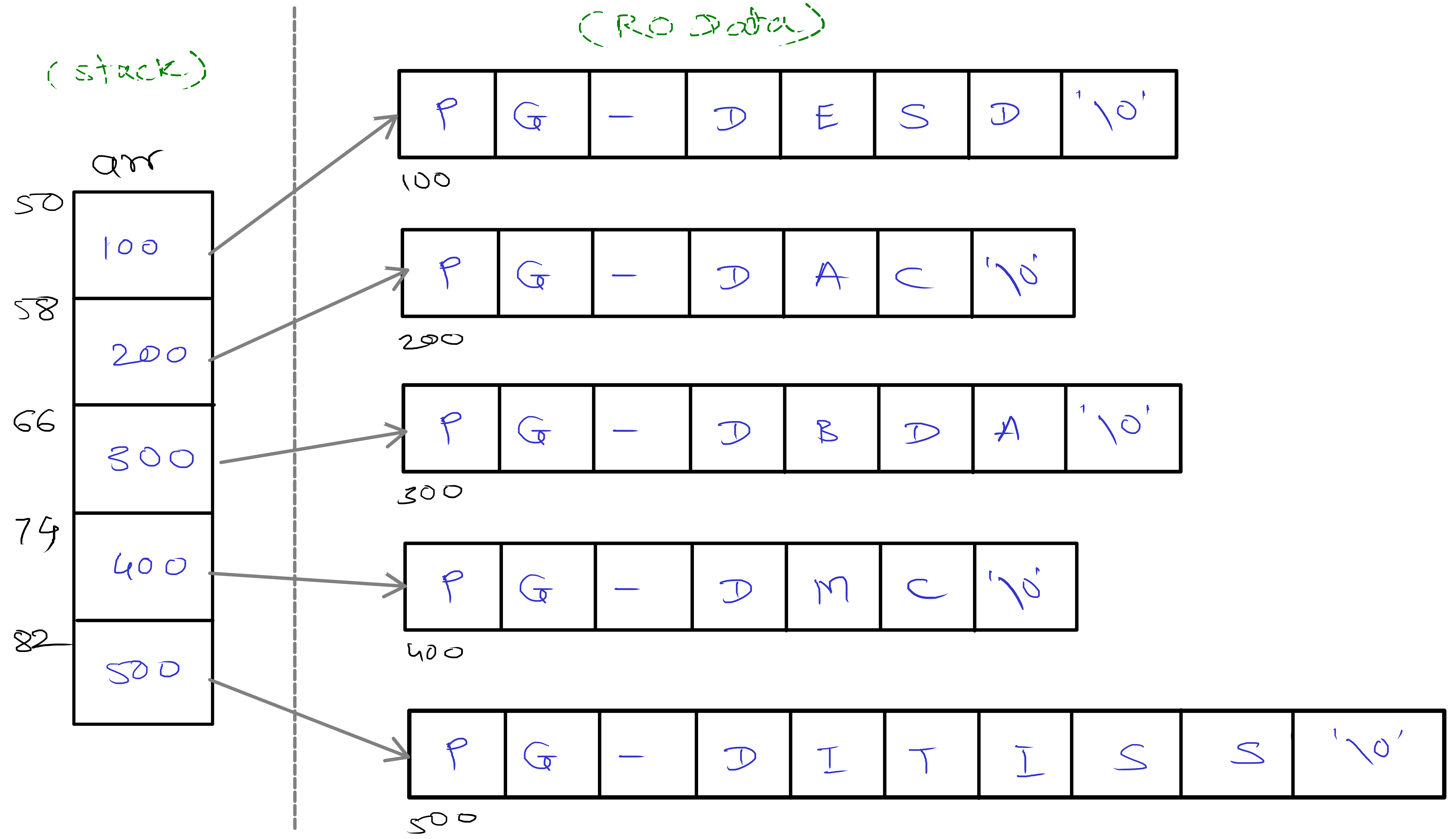
*arr = *50 = 100

**arr = **50 = *100 = 10

**arr[1] = **200 = *11 ← error

***(arr+1)

char *arr[ ] = { "PG-DESD", "PG-DAC", "PG-DBDA", "PG-DMC", "PG-DITISS"};



Command Line Arguments

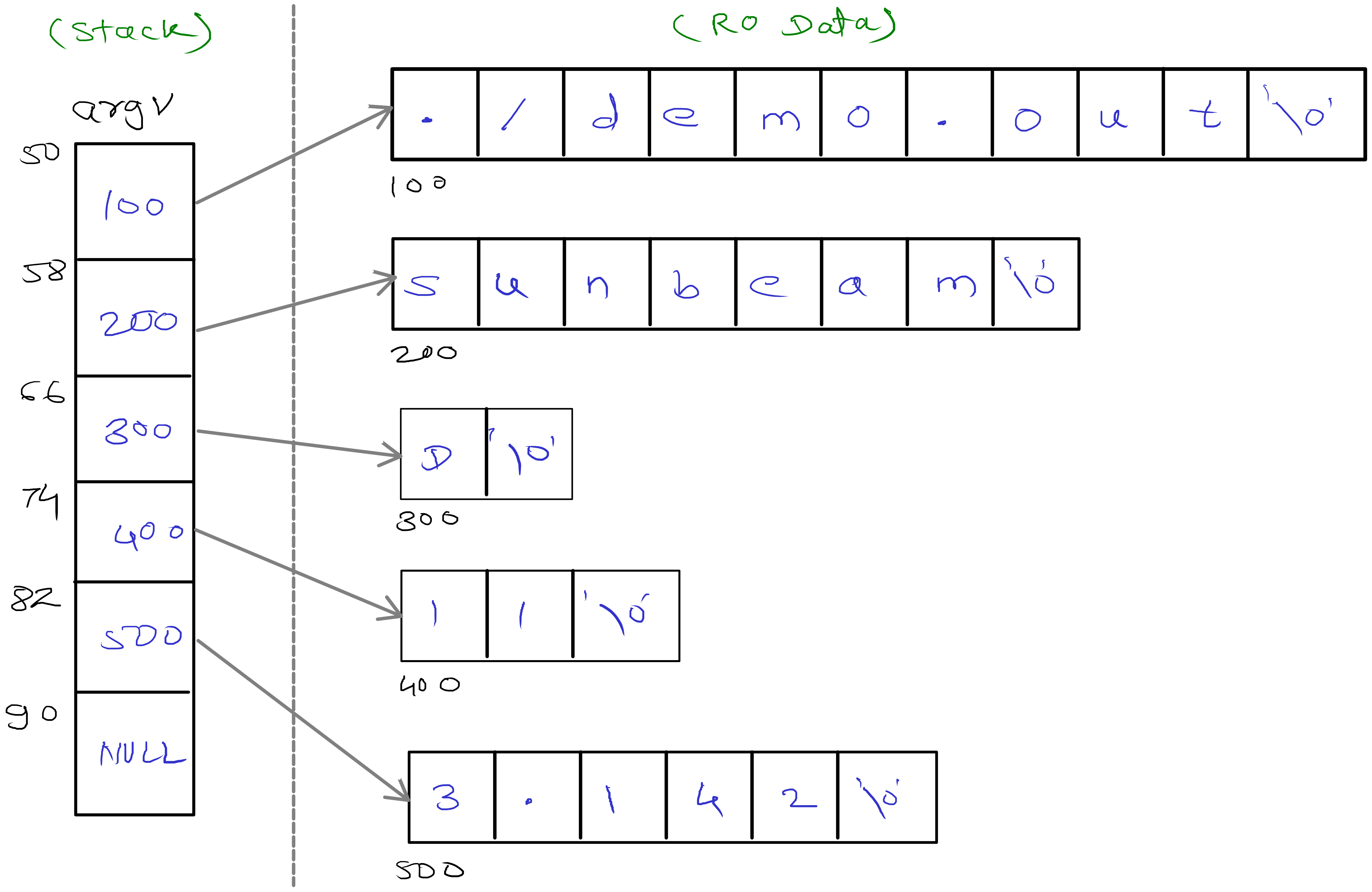
- Arguments passed to the program from command line at the time of running.
- Argruments are passed to the main function
- if we are giving cmd line args then signature/declaration of main should be

```
int main(int argc, char *argv[])   {}
        argc - no of command line arguments
        argv - list(array) of command line arguments
```

- By default, name of executable file is passed as first command line arguments.
- While running your program you can pass cmd line arg
- eg we have created demo.out

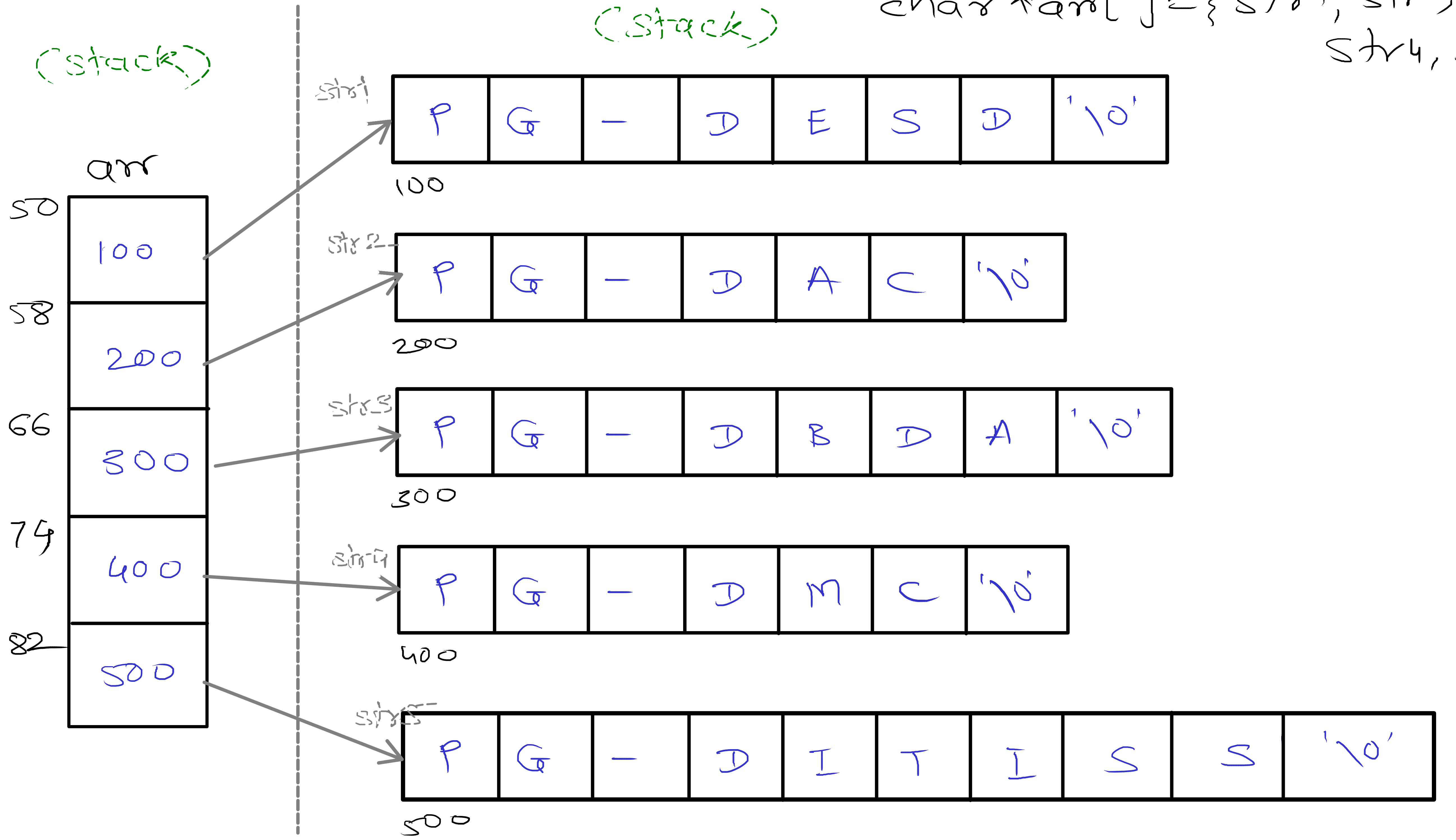
```
$ ./demo.out sunbeam D 11 3.142
argc  - 5
argv = {"./demo.out", "sunbeam", "D", "11", "3.142", NULL}
argv[0] = "./demo.out"
argv[1] = "sunbeam"
argv[2] = "D"
argv[3] = "11"
argv[4] = "3.142"
argv[5] = NULL                     -- indicates end of cmd line args
```

```
$ ./demo.out sunbeam D 11 3.142
```



```
char str1[]="PG-DESD";
char str2[]="PG-DAC";
      :
char str5[]="PG-DITISS";

char *arr[]={str1, str2, str3,
             str4, str5};
```


(stack)
(stack)
arr
50
58
66
74
82
100
200
300
400
500
str1
str2
str3
str4
str5
P G - D E S D '\0'
P G - D A C '\0'
P G - D B D A '\0'
P G - D M C '\0'
P G - D I T I S S '\0'
100
200
300
400
500