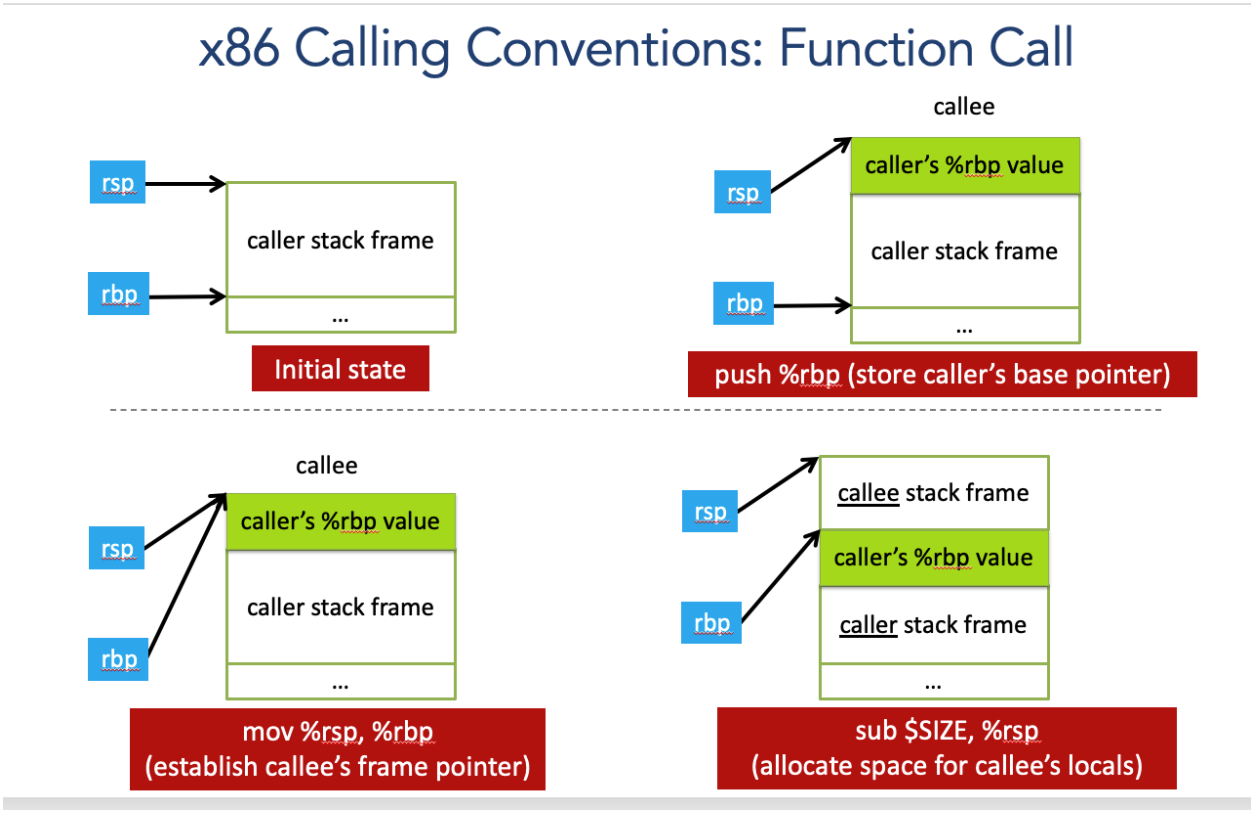
# CS31 Worksheet: Week 7: Arrays and Pointers




Given the figure above, can you describe in figures, and words the sequence of instructions to return from a function call?

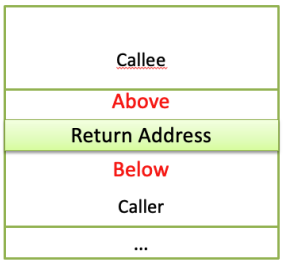
## If we need to place arguments in the caller's stack frame, should they go above or below the return address?

A. Above

B. Below

C. It doesn't matter

D. Somewhere else




## Which expression would compute the address of iptr[3]?

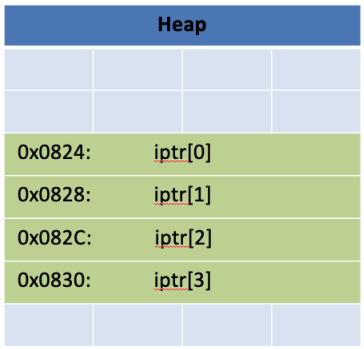
A. 0x0824 + 3 * 4

B. 0x0824 + 4 * 4

C. 0x0824 + 0xC

D. More than one (which?)

E. None of these



## Let's try an example

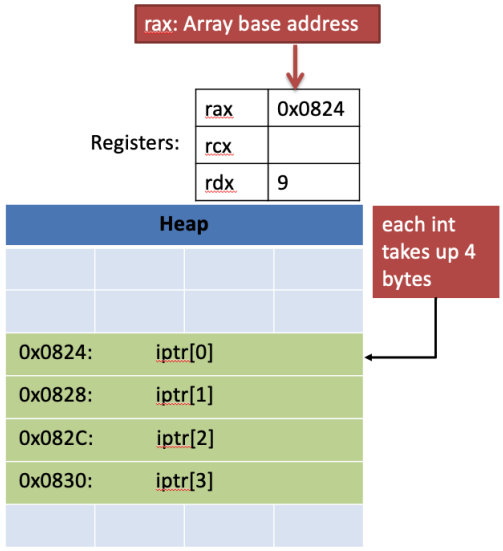
Suppose:

```
int iptr = malloc(4*sizeof(int));
//iptr is stored in register %rax.
int i=2; is stored at %rbp-8
```

C code says:

```
iptr[i] = 9;
```

Using what we just learnt, what does the C code above translate to, in assembly?




## Why do we want to align data on multiples of the data size?

A. It makes the hardware faster.

B. It makes the hardware simpler.

C. It makes more efficient use of memory space.

D. It makes implementing the OS easier.

E. Some other reason.

How much space do we need to store one of these structures? Why?

```
struct student{
  char name[11];
  short age;
  int id;
};
```

A. 17 bytes
B. 18 bytes
C. 20 bytes
D. 22 bytes
E. 24 bytes

## Struct field syntax…

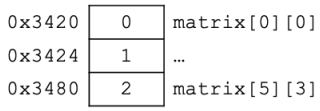
```
struct student {
  int id;
  short age;
  char name[11];
};
struct student *s = malloc(sizeof(struct student));
```

What about this?

How do we get to the id and age?

If we declared `int matrix[5][3];`, and the base of matrix is 0x3420, what is the address of `matrix[3][2]`?

A. 0x3438
B. 0x3440
C. 0x3444
D. 0x344C
E. None of these




```
base addr
+ row offset (# rows * row_size * sizeof(data_type))
+ col offset
```