CS 354
Machine Organization and Programming

Week 3a: Pointers and Arrays

Michael Doescher
Spring 2021

Review

Control Flow

- Conditionals
- Loops
- Functions

- switch

Data Structures

- Primitives: char, int, float, double
- 1d arrays
- char arrays (strings)
- pointers

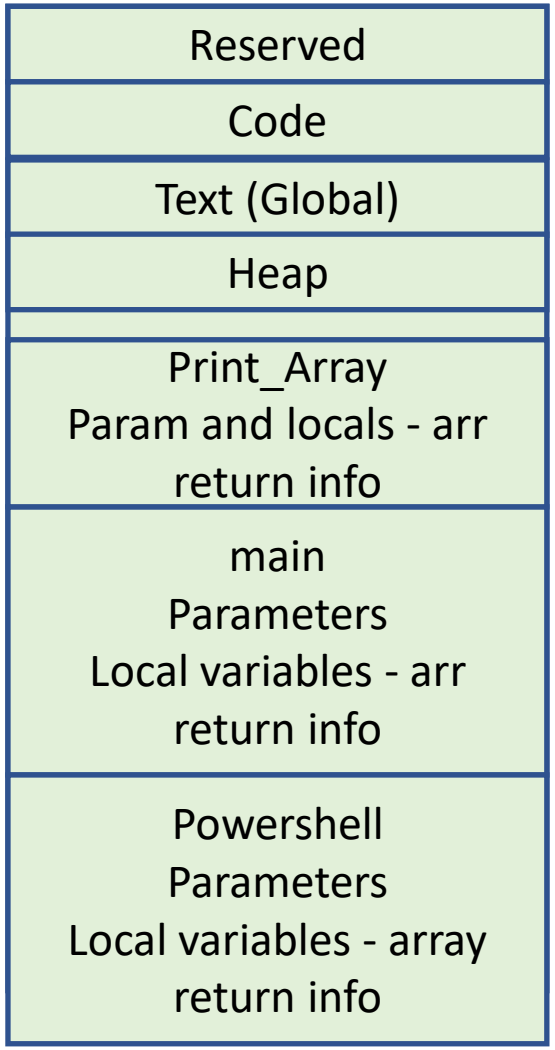
- enum
- multidimensional arrays
- structures

Call Stack Diagram

```
Welcome to Week 3 of CS354!

a    = 3; address of a = 0061FF1C
b    = 4; address of b = 0061FF18
*pa = 3; pa = 0061FF1C; address of pa = 0061FF14
*pb = 4; pb = 0061FF18; address of pb = 0061FF10
```

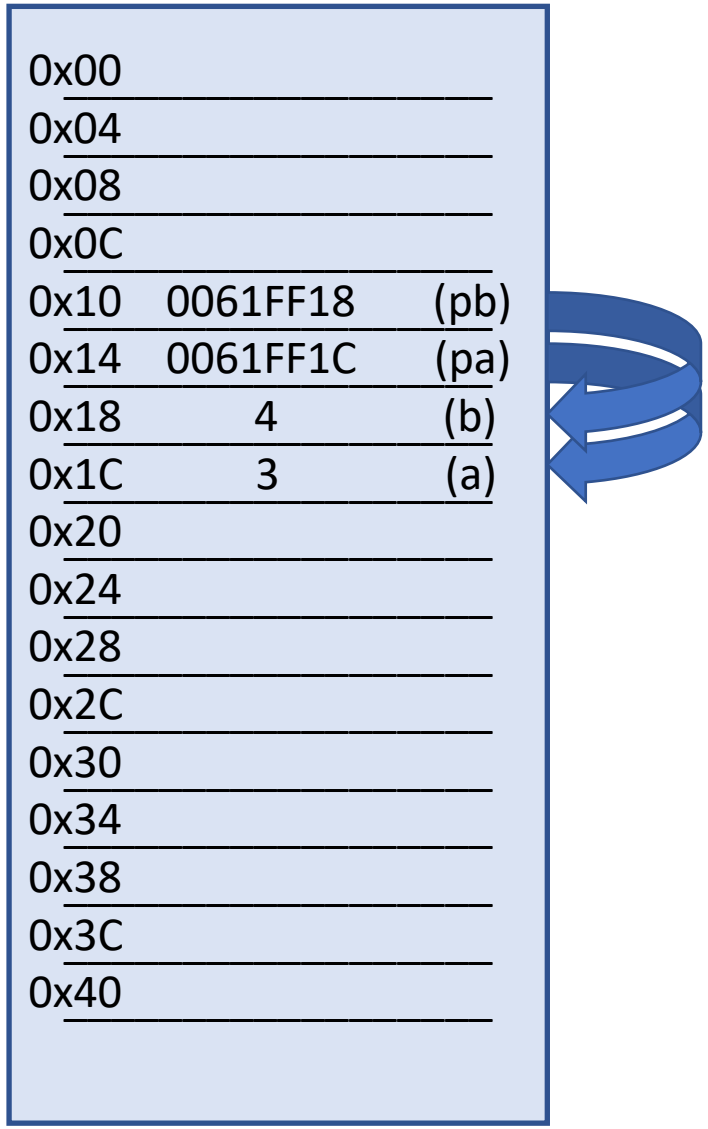
best-linux.gslb.cs.wisc.edu - PuTTY

```
[doescher@snares-10] (15)$ gcc -o demo demo.c -m32 -Wall
[doescher@snares-10] (16)$ demo
Welcome to Week 3 of CS354!

a    = 3; address of a = 0xffb34c8c
b    = 4; address of b = 0xffb34c90
*pa = 3; pa = 0xffb34c8c; address of pa = 0xffb34c94
*pb = 4; pb = 0xffb34c90; address of pb = 0xffb34c98
```



Memory Model

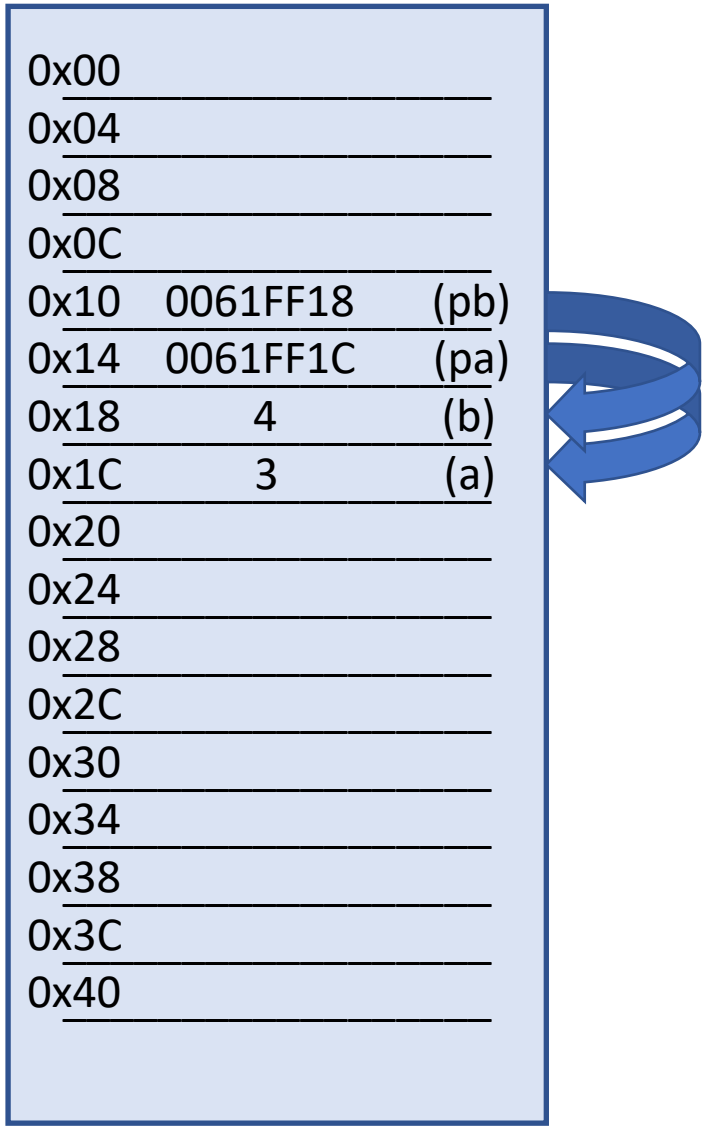
```
Welcome to Week 3 of CS354!

a   = 3; address of a = 0061FF1C
b   = 4; address of b = 0061FF18
*pa = 3; pa = 0061FF1C; address of pa = 0061FF14
*pb = 4; pb = 0061FF18; address of pb = 0061FF10
```



Memory Model

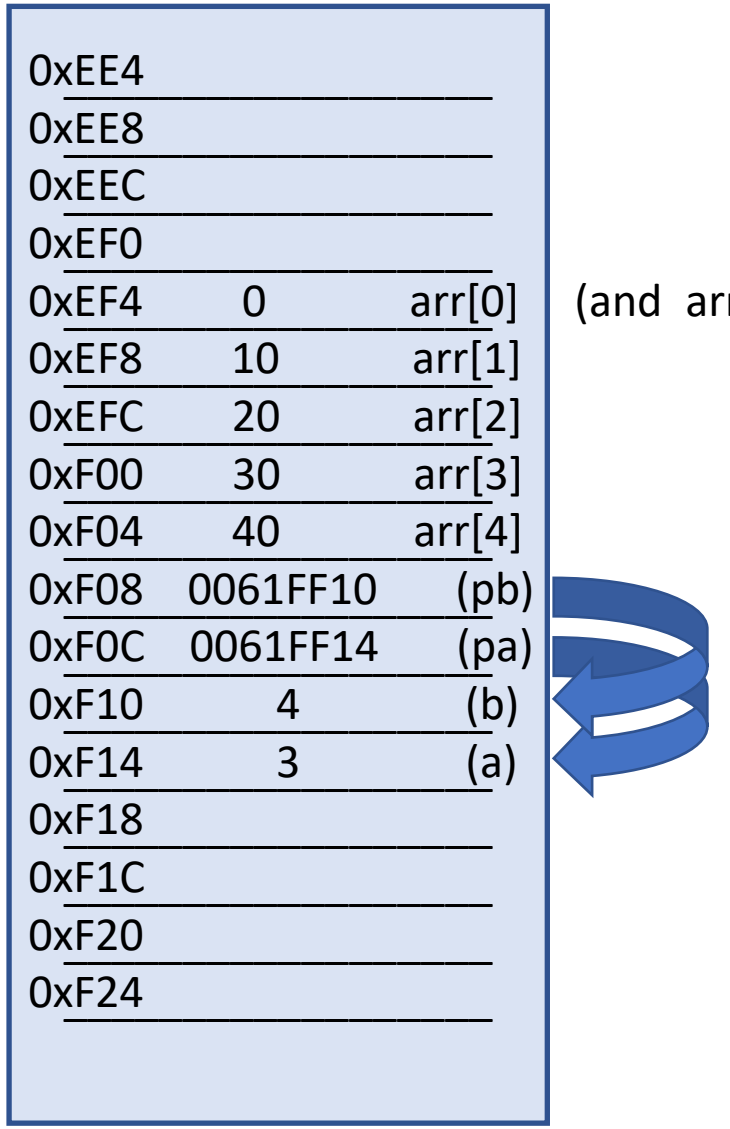
```
Welcome to Week 3 of CS354!

a   = 3; address of a = 0061FF1C
b   = 4; address of b = 0061FF18
*pa = 3; pa = 0061FF1C; address of pa = 0061FF14
*pb = 4; pb = 0061FF18; address of pb = 0061FF10
```

```
Welcome to Week 3 of CS354!

a   = 3; address of a = 0061FF14
b   = 4; address of b = 0061FF10
*pa = 3; pa = 0061FF14; address of pa = 0061FF0C
*pb = 4; pb = 0061FF10; address of pb = 0061FF08

arr[0] = 0 address of arr[0] = 0061FEF4
arr[1] = 10 address of arr[1] = 0061FEF8
arr[2] = 20 address of arr[2] = 0061FEFC
arr[3] = 30 address of arr[3] = 0061FF00
arr[4] = 40 address of arr[4] = 0061FF04
```



Memory Model

```
Welcome to Week 3 of CS354!

a   = 3; address of a = 0061FF14
b   = 4; address of b = 0061FF10
*pa = 3; pa = 0061FF14; address of pa = 0061FF0C
*pb = 4; pb = 0061FF10; address of pb = 0061FF08

arr[0] = 0 address of arr[0] = 0061FEF4
arr[1] = 10 address of arr[1] = 0061FEF8
arr[2] = 20 address of arr[2] = 0061FEFC
arr[3] = 30 address of arr[3] = 0061FF00
arr[4] = 40 address of arr[4] = 0061FF04
```

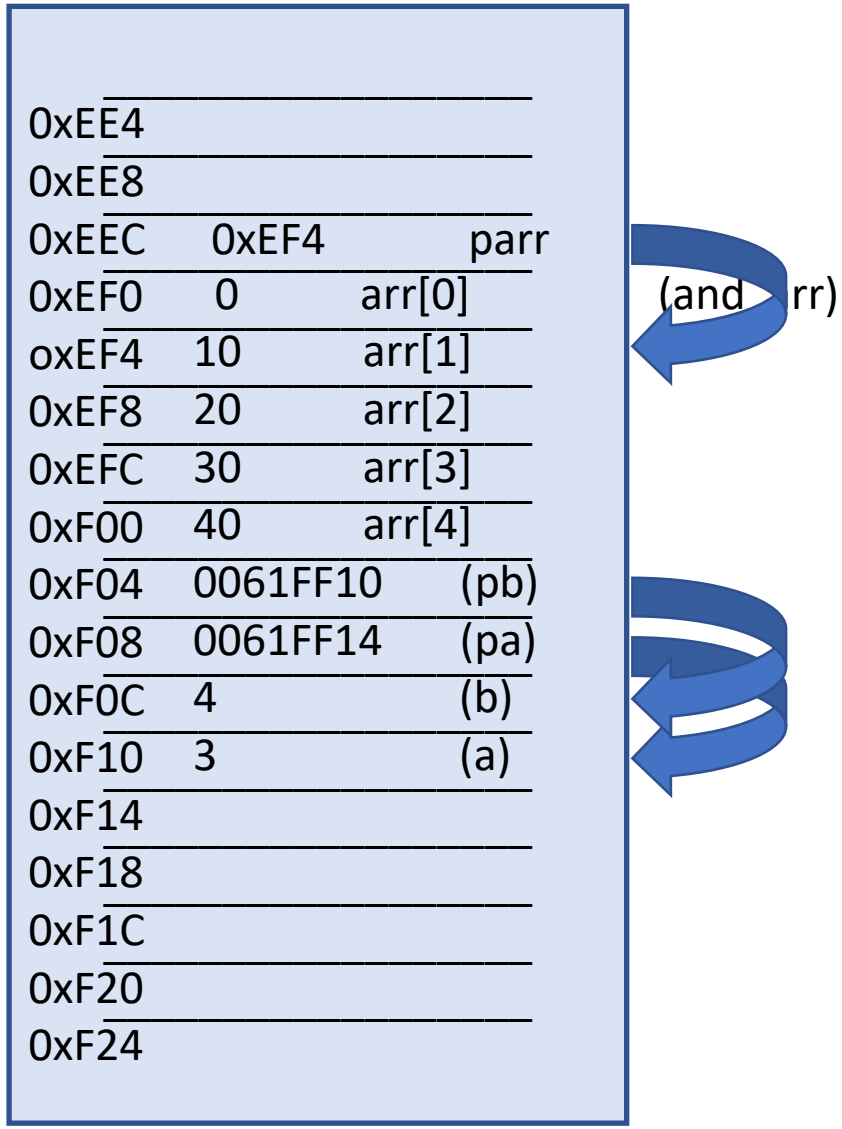


How Does Array Access Really Work

arr[3] translates to *(arr + 3*sizeof(int))
arr = *(0xEF0 + 3*4) = *0xEFC

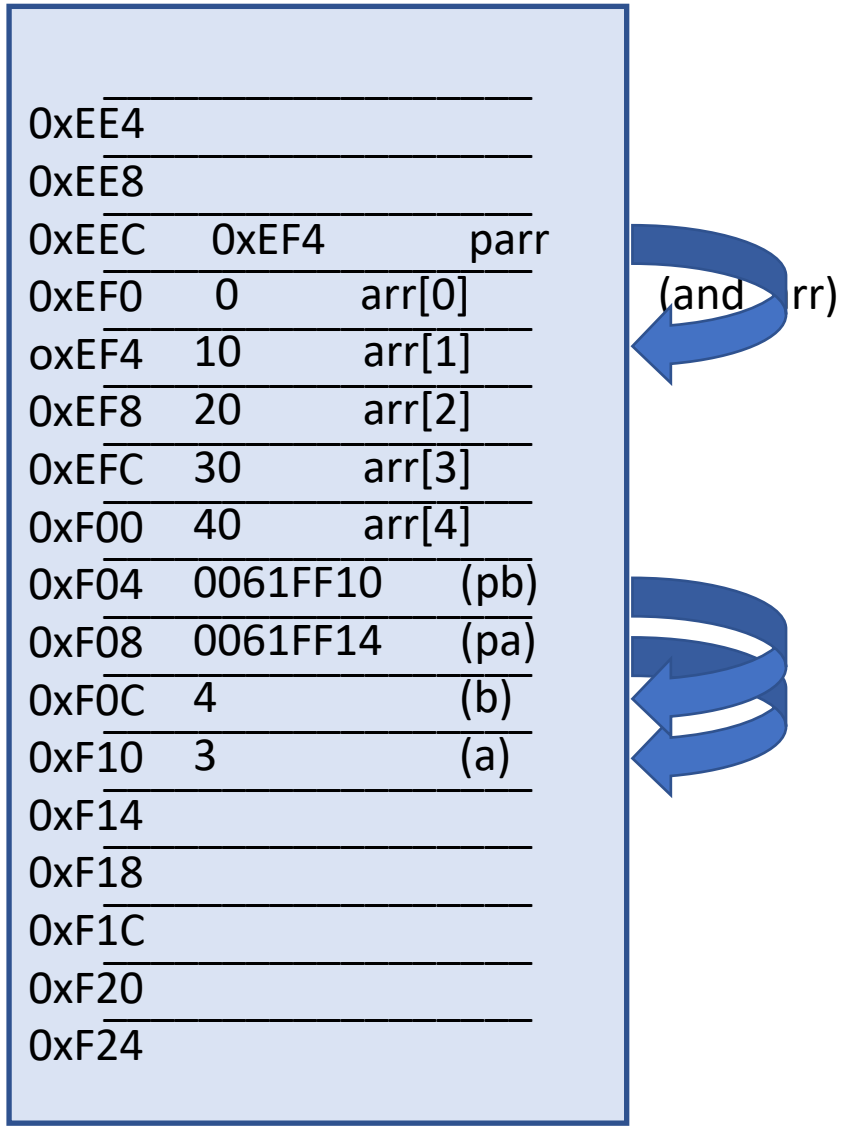
we don't type that as programmers
arr[3] = *(arr+3)
sizeof is taken care of by the compiler

in general
arr[i] -> *(arr + i * sizeof(int))
or we type
arr[i] -> *(arr+i) = *(i + arr)



Memory Model

```
a    = 3; address of a = 0061FF10
b    = 4; address of b = 0061FF0C
*pa = 3; pa = 0061FF10; address of pa = 0061FF08
*pb = 4; pb = 0061FF0C; address of pb = 0061FF04

arr[0] = 0 address of arr[0] = 0061FEF0
arr[1] = 10 address of arr[1] = 0061FEF4
arr[2] = 20 address of arr[2] = 0061FEF8
arr[3] = 30 address of arr[3] = 0061FEFC
arr[4] = 40 address of arr[4] = 0061FF00
arr    = 0061FEF0; address of arr = 0061FEF0
parr[0] = 0 address of parr[0] = 0061FEF0
parr[1] = 10 address of parr[1] = 0061FEF4
parr[2] = 20 address of parr[2] = 0061FEF8
parr[3] = 30 address of parr[3] = 0061FEFC
parr[4] = 40 address of parr[4] = 0061FF00
parr    = 0061FEF0; address of parr = 0061FEEC
```



CS 354
Machine Organization and Programming

Week 3b: 2D Arrays

Michael Doescher
Spring 2021

2d arrays

```
int m[3][5];
// 3 rows
// 5 columns
```

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35

2d arrays and 1d memory

```
int m[3][5];

// initialize
for (int i=0;i<3;i++)
for (int j=0;j<5;j++)
m[i][j] = 10*(i+1) + j+1
```

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35

2d arrays and 1d memory

```
m[0][0] = 11 at address 0061FECC
m[0][1] = 12 at address 0061FED0
m[0][2] = 13 at address 0061FED4
m[0][3] = 14 at address 0061FED8
m[0][4] = 15 at address 0061FEDC
m[1][0] = 21 at address 0061FEE0
m[1][1] = 22 at address 0061FEE4
m[1][2] = 23 at address 0061FEE8
m[1][3] = 24 at address 0061FEEC
m[1][4] = 25 at address 0061FEF0
m[2][0] = 31 at address 0061FEF4
m[2][1] = 32 at address 0061FEF8
m[2][2] = 33 at address 0061FEFC
m[2][3] = 34 at address 0061FF00
m[2][4] = 35 at address 0061FF04
```

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35

0xEC0		
0xEC4		
0xEC8		
0xECC	11	m[0][0]
0xED0	12	m[0][1]
0xED4	13	m[0][2]
0xED8	14	m[0][3]
0xEDC	15	m[0][4]
0xEE0	21	m[1][0]
0xEE4	22	m[1][1]
0xEE8	23	m[1][2]
0xEEC	24	m[1][3]
0xEF0	25	m[1][4]
0xEF4	31	m[2][0]
0xEF8	32	m[2][1]
0xEFC	33	m[2][2]
0xF00	34	m[2][3]
0xF04	35	m[2][4]

2d arrays and 1d memory

Address Computation

arr[3] -> *(arr + 3 * sizeof(int))

m[1][2] -> *(m + (5*1 + 2)*sizeof(int))
m[i][j] -> *(m + (MAX_COL * i + j)*sizeof(int))

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35

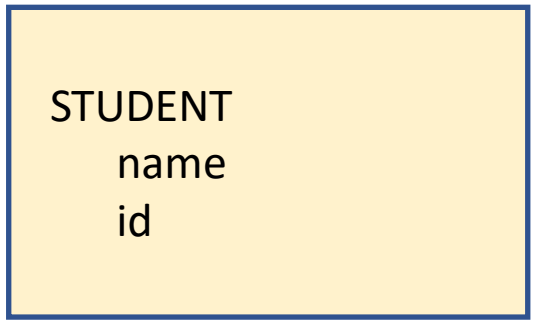
0xEC0		
0xEC4		
0xEC8		
0xECC	11	m[0][0]
0xED0	12	m[0][1]
0xED4	13	m[0][2]
0xED8	14	m[0][3]
0xEDC	15	m[0][4]
0xEE0	21	m[1][0]
0xEE4	22	m[1][1]
0xEE8	23	m[1][2]
0xEEC	24	m[1][3]
0xEF0	25	m[1][4]
0xEF4	31	m[2][0]
0xEF8	32	m[2][1]
0xEFC	33	m[2][2]
0xF00	34	m[2][3]
0xF04	35	m[2][4]

CS 354
Machine Organization and Programming

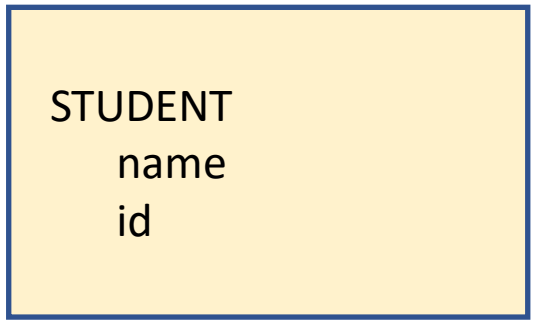
Week 3c: Structs

Michael Doescher
Spring 2021

Structures

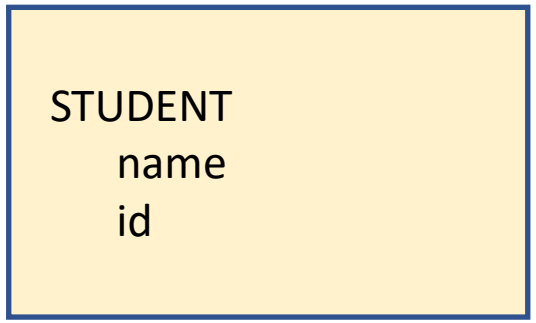


Structures




```
struct STUDENT {
    char *name;
    int id;
};
```

Structures



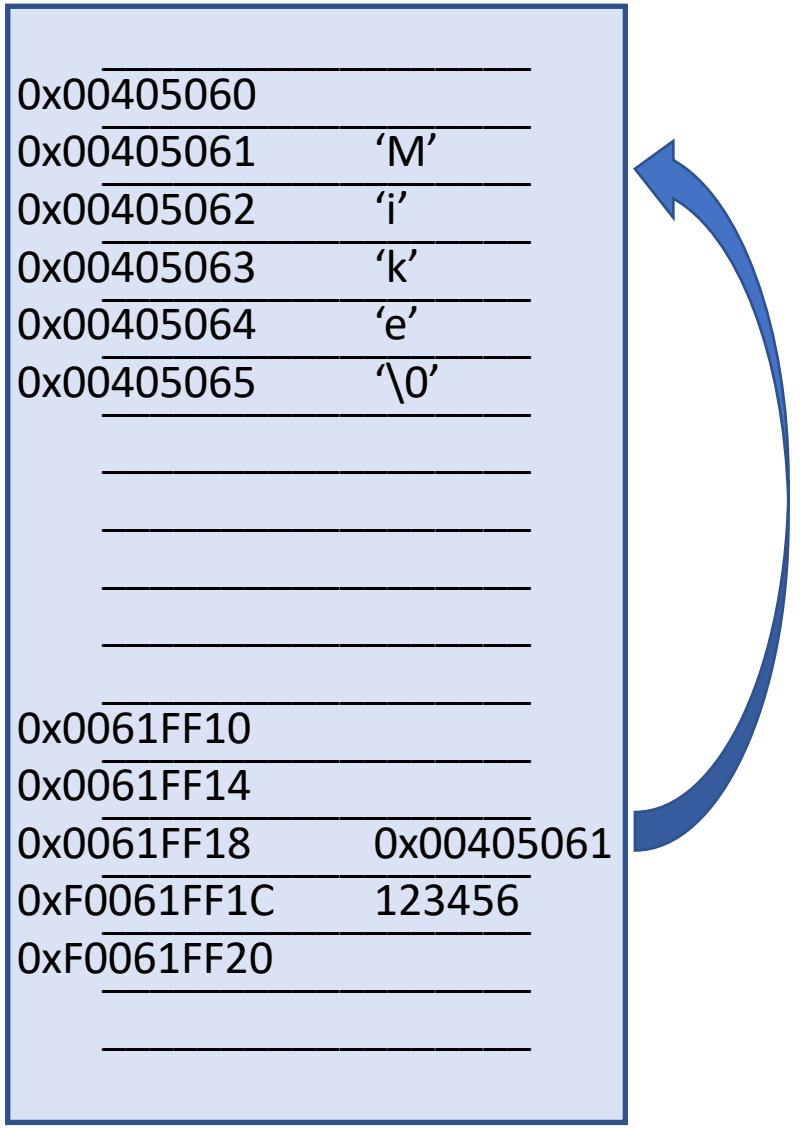

```
struct STUDENT {
    char *name;
    int id;
};

struct STUDENT s1;
s1.name = “Mike”
s1.id = 123456;
```

Structures

STUDENT
- name
- id

```
struct STUDENT {
    char *name;
    int id;
};

struct STUDENT s1;
s1.name = “Mike”
s1.id = 123456;
```

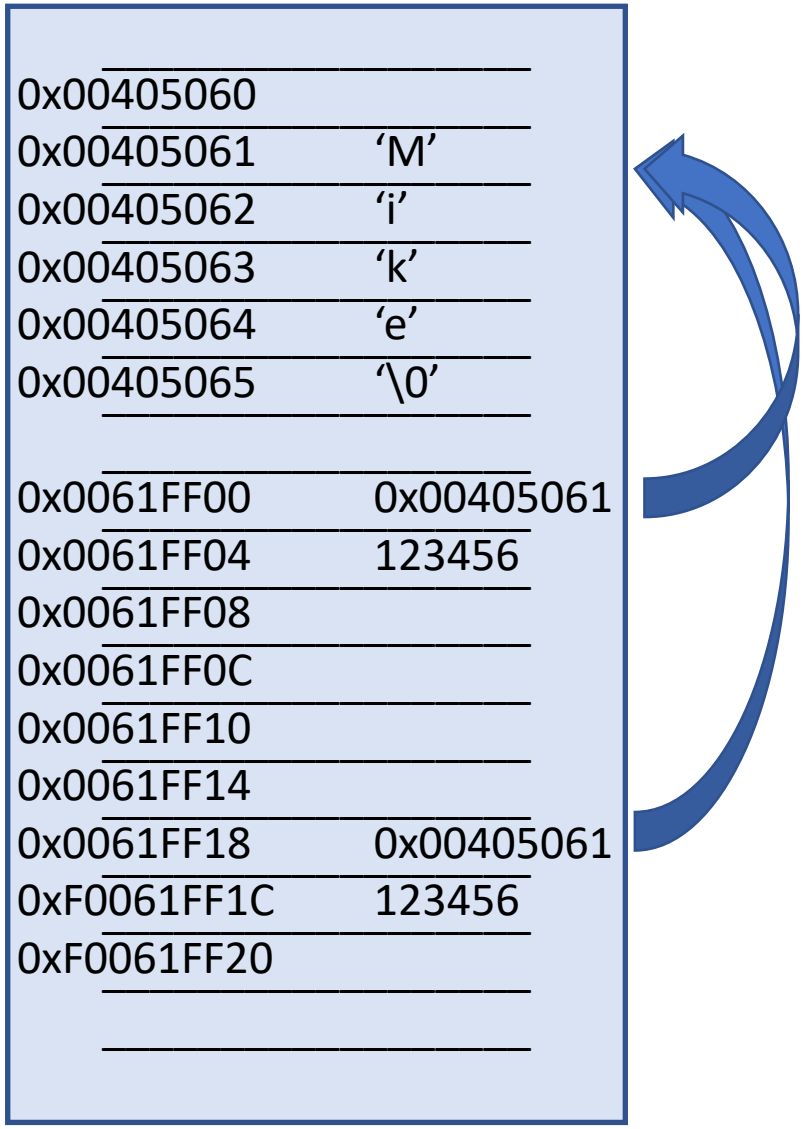
```
address of student = 0061FF18
name: Mike at address 0061FF18
id: 123456 at address 0061FF1C
address of string mike = 00405061
```



Structures

```
struct STUDENT {
    char *name;
    int id;
};

struct STUDENT s1;
s1.name = “Mike”
s1.id = 123456;
```

```
address of student = 0061FF18
name: Mike at address 0061FF18
id: 123456 at address 0061FF1C
address of string Mike = 00405061

address of s = 0061FF00
name: Mike at address 0061FF00
id: 123456 at address 0061FF04
address of string Mike = 00405061
```



Print_Student_With_Pointer

```
struct STUDENT {
    char *name;
    int id;
};

struct STUDENT s1;
s1.name = “Mike”
s1.id = 123456;
```

```
address of student = 0061FF18
name: Mike at address 0061FF18
id: 123456 at address 0061FF1C
address of string Mike = 00405061

address of s = 0061FF00
name: Mike at address 0061FF00
id: 123456 at address 0061FF04
address of string Mike = 00405061
```