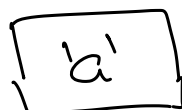




Pointer → variable that stores address.

char ch = 'a'; ⇒ 
↓
1 byte

10000 ← address in RAM at which memory is allocated for ch.

int i = 10; ⇒ 
↓
1 word

Memory allocated for a variable depends on its type.

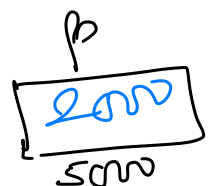
float f = 1.0 ⇒ 
↓
4 bytes

3000

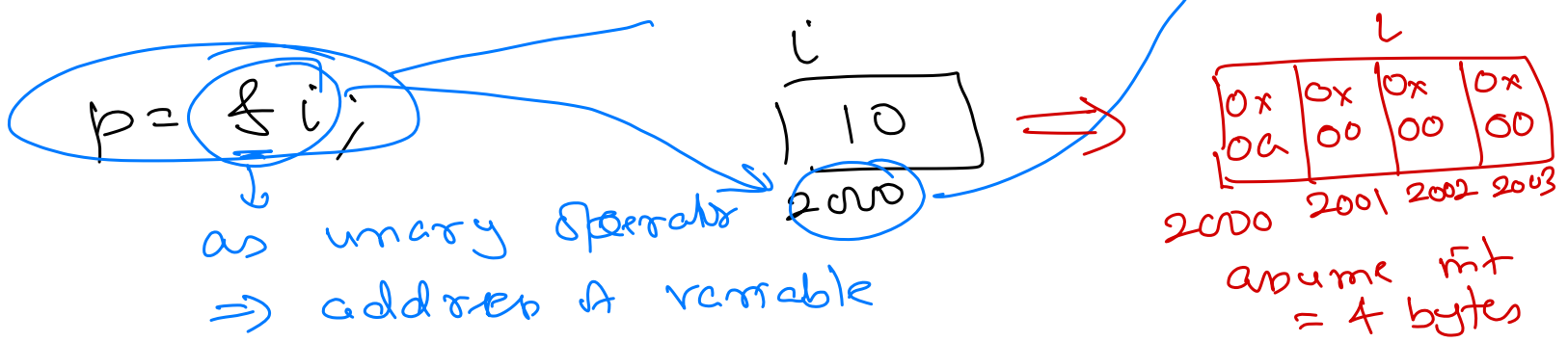
short \rightarrow type modifiers can be used
long \rightarrow with int

address + type of value stored at that address \Rightarrow to find what value is stored at address.

int $\underbrace{\quad\quad\quad}_{\text{to int}}$ $\underbrace{*p;}_{p \text{ is a pointer}} \Rightarrow p \text{ stores address of int variable.}$
int $i = 10;$



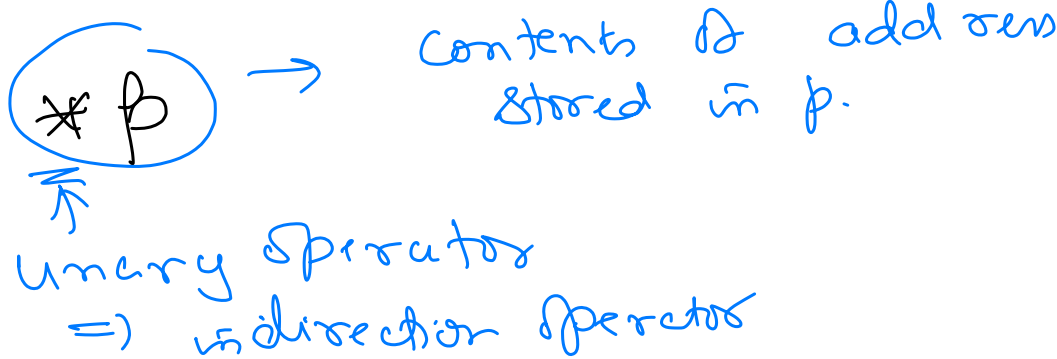
The diagram shows a pointer variable p pointing to a memory box. Inside the box is the value 2000 . Below the box is the address 5000 . An arrow points from the text $i = 10;$ to the memory box.

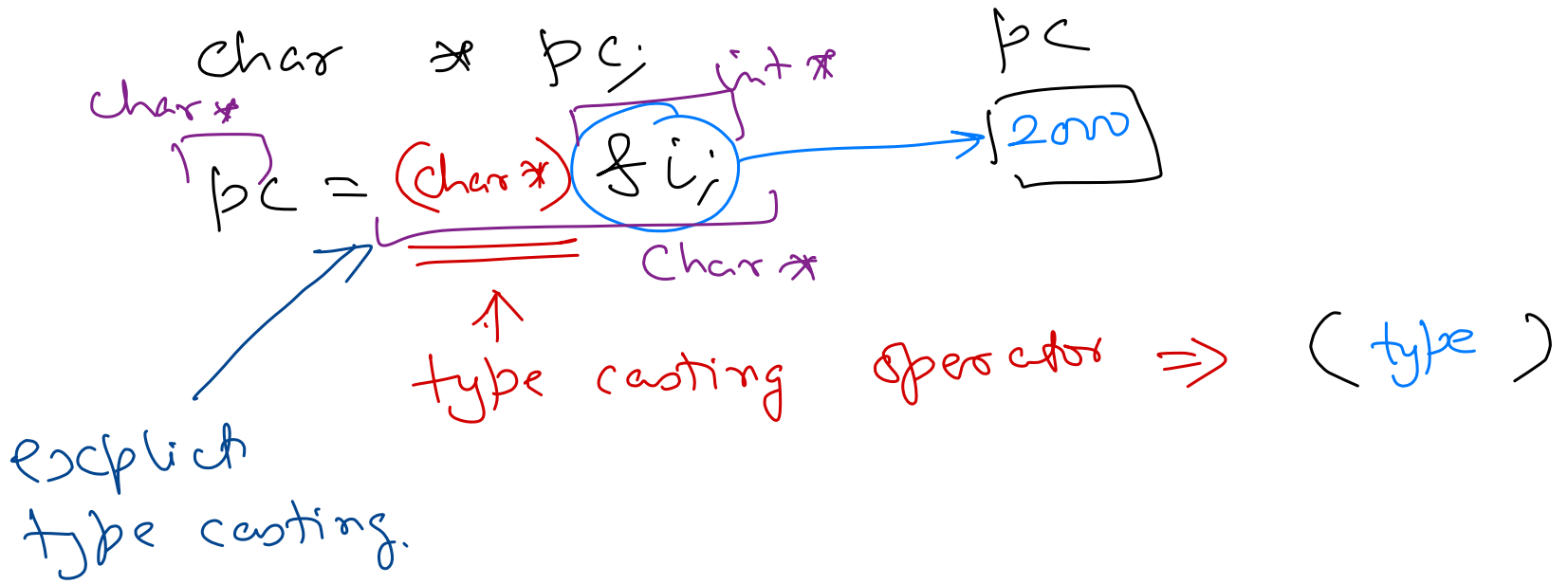


size of \leftarrow keyword
 \leftarrow operator
 \Downarrow

size of (p) \Rightarrow 4/8

size of a type



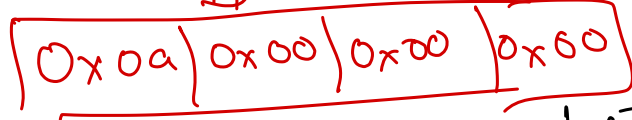


`*p` \Rightarrow 10

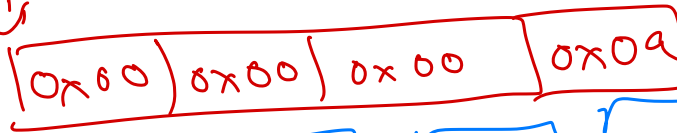
`*pc` \Rightarrow 10

endianism of CPU

little
endian



big
endian (network byte order)



10

10

base

= 0000 0000 ... 000 01010₂

0x 0000 000a₁₆

integer
value/
type

signed

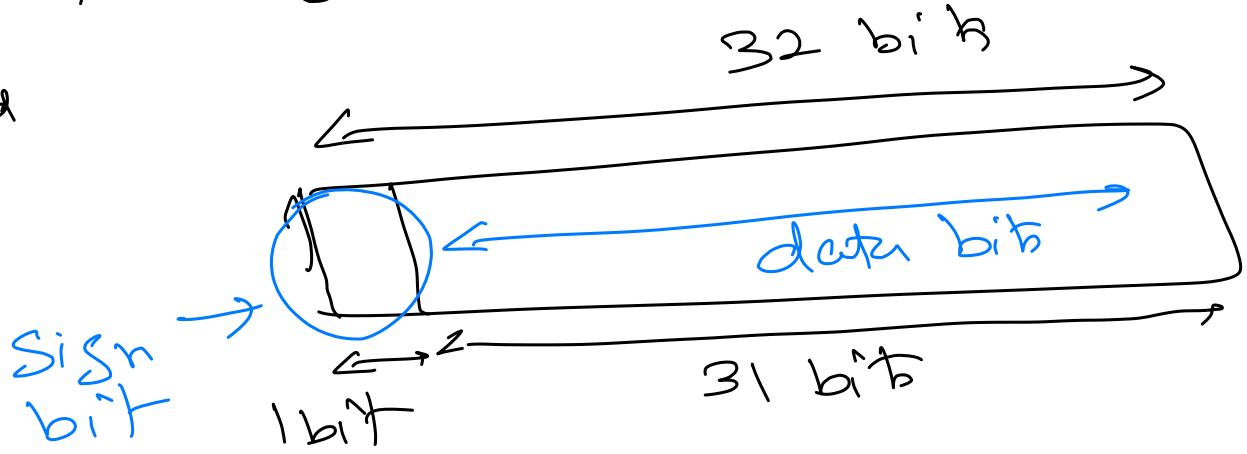
or

unsigned

keywords as type modifiers

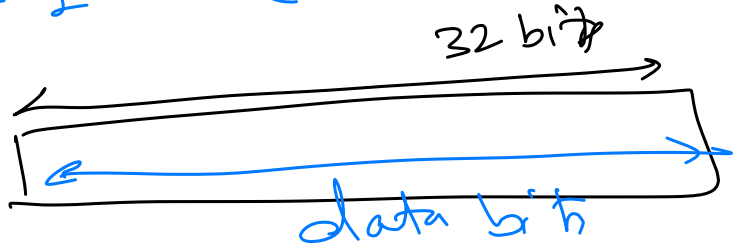
int \rightarrow 4 bytes \Rightarrow 32 bits

signed



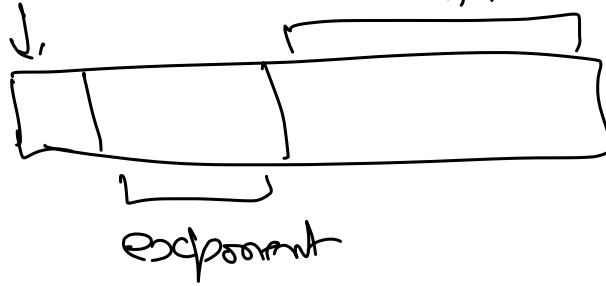
$\begin{cases} 0 & +ve \\ 1 & -ve \end{cases}$

unsigned int \Rightarrow



float, double \Rightarrow floating point type

Sign bit

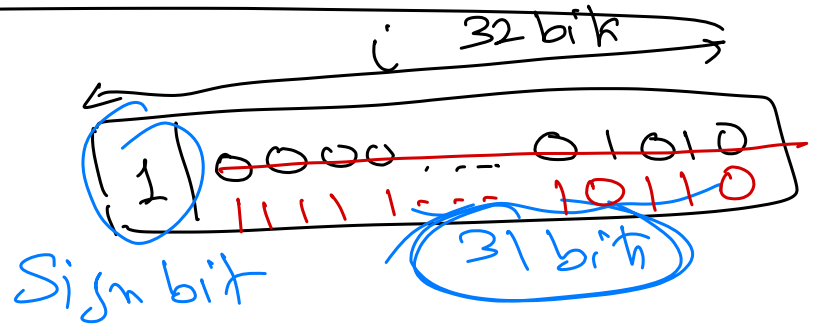


are always signed.

IEEE

int $i = -10$

\downarrow
-ve numbers
are stored in
2's complement



0000 ... 01010

11111 ... 10101 \leftarrow 1's complement

+ 1

11111 ... 110110 \leftarrow 2's complement

$$3 - 2 \Rightarrow 3 + (-2)$$

char ch = 1024; $\textcircled{\text{ch}} \rightarrow ?$


```
int main() {
    int arr[3] = {1, 2, 3};
```

arr		
0	1	2
1	2	3
1000	1004	1008

```
f1(arr);
```

1000

array name
gives us starting address
of first element

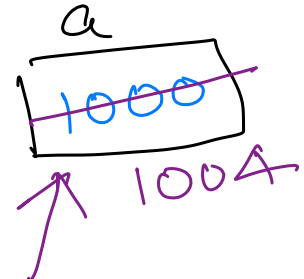
```
}
```

```
void
```

```
f1(int a[]) {
    int *a
    int a[3]
    int a[100] }
```

*a ⇒ 1

a is a
pointer



$$*(a+1) \Rightarrow 2$$

$$1000 + 1$$

address of next
of type int

value

1004

1004

$$\Rightarrow 2$$

$$* \underline{\underline{++a}}$$

\Downarrow

\Downarrow

$*1004$
int*

$$a = a + 1$$

$a[1]$

\Downarrow

$*(a+1)$

int arr[10];

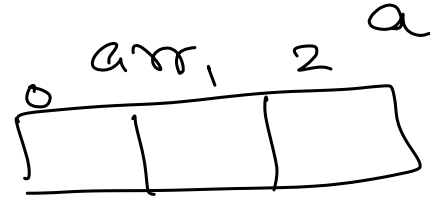
⋮

How many elements in arr?

$$\underbrace{\text{sizeof}(arr)}_{10 * \text{sizeof}(\text{int})} / \underbrace{\text{sizeof}(\text{int})}_{\text{arr}[0]} \Rightarrow 10$$

```
void f1( int (&a)[3] ) {
```

```
}  
int main() {  
    int arr[3] = ...;  
    f1(arr);
```



~~int &x[3];~~
x is an array of 3
elements
each of type reference to int

int (&y) [3];

y is a reference to

array of 3 elements each of
type int

C-strings \Rightarrow char array that
are null terminated.

char s[10] = { 'a', 'b', '0' } \Rightarrow 'ab0'

\downarrow \uparrow
s is a string. null terminator

char s1[10] = "ab";

↓
string constant

'a', 'b', '\0'

f1(s1); \Rightarrow int f1(char s[])
{
:
}

```
void printString (char s[]) {
```

```
    int i = 0;
```

```
    while (s[i] != '\0') {
```

```
        std::cout << s[i];  
        ++i;
```

```
    }
```

```
int main() {
```

```
    char s1[] = "hello";
```

```
    printString (s1);
```

```
    return 0;
```

```
}
```

8
100

s1

0	1	2	3	4	5
h	e	l	l	o	\0
100	101	102	-	-	-

int strlen(char s[]);

void strcpy(char dest[], char src[]);

bool compare(char s1[], char s2[]);

void ToUpper(char s[]);

void ToLower(char s[]);