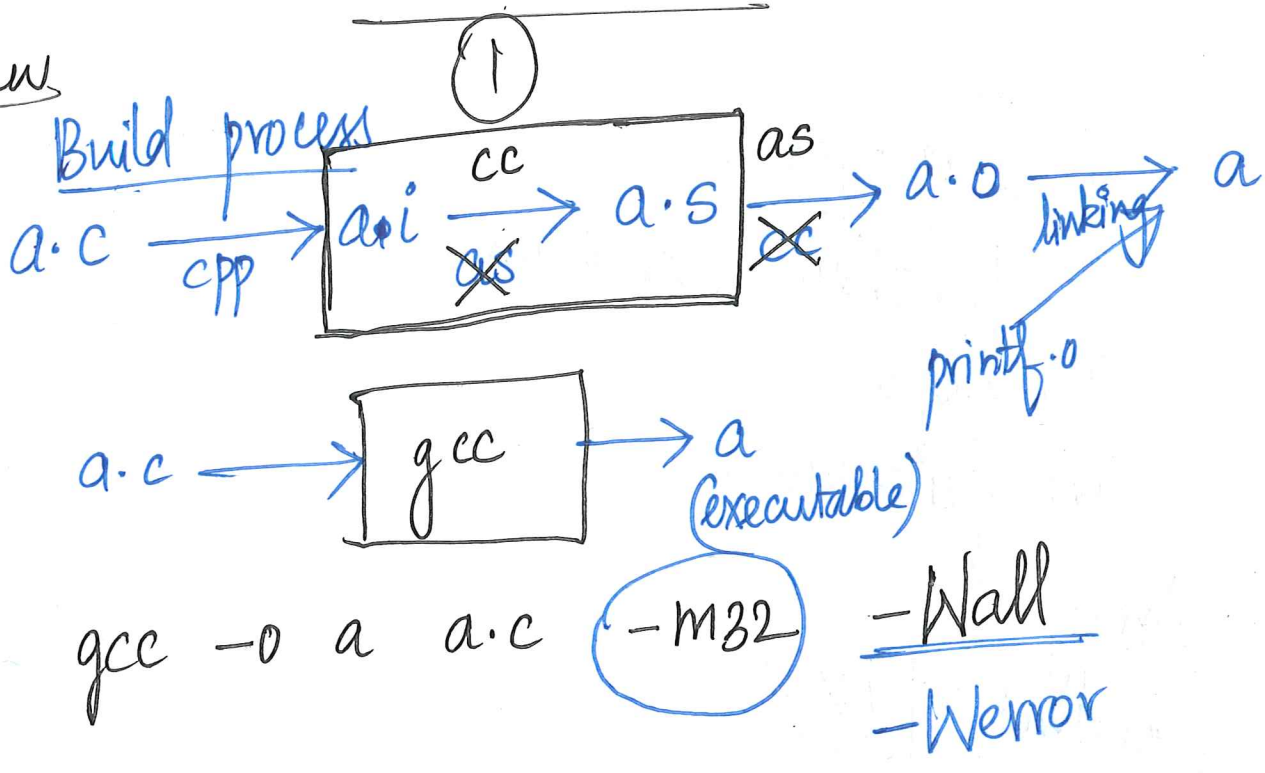
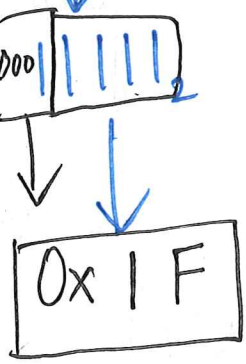
CS 354 @ Epic

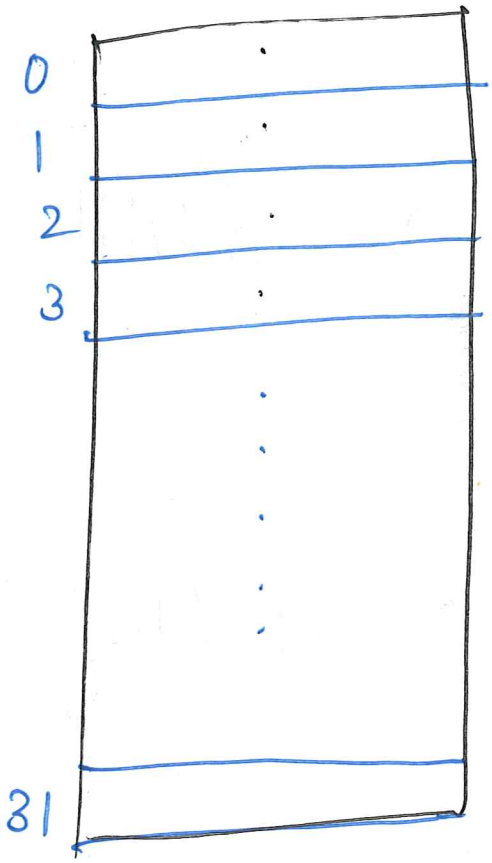
Lecture 2

Review

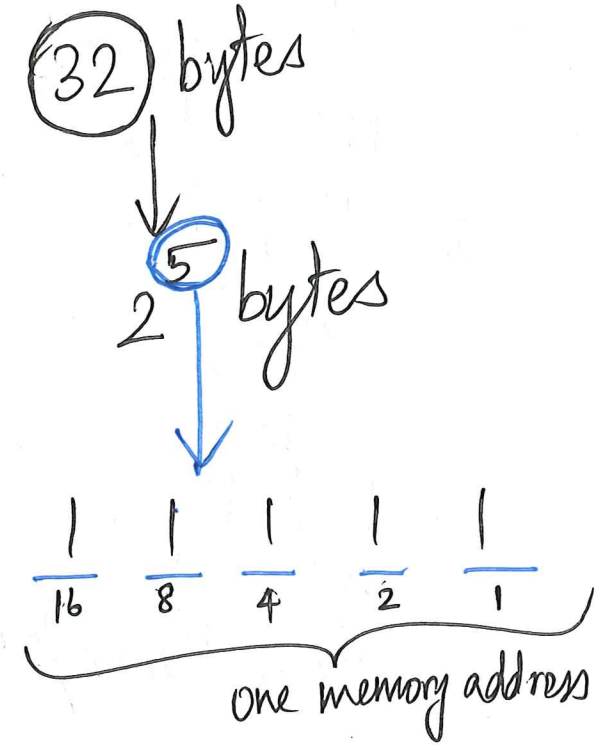


gcc -o a a.c -m32 -Wall -Werror





Memory



Today: C Programming.

1. C Basics
   - i) data types
   - ii) if/then _ decisions
   - iii) loops
   - iv) functions.
2. Arrays
3. Pointers (X.)
4. Structures.

History

Dennis Ritchie → C → UNIX → Machine / System

@ Bell Labs. (1972)

K&R C (1972) → ANSI C (1989)

K&R C (1972) → Brian Kernighan

ANSI C (1989) → ISO C99 → C18

1. Data types

- int
- float (3.14) OR double (3.14325972)
- char 'A'

Size qualifiers

1. short
2. long

short int [atleast 16 bits] ≤ int [atleast 16 bits] ≤ long int [atleast 32 bits]

short int: 16 bits OR 2 bytes

int: 32 bits OR 4 bytes

long int: ~~4~~ 8 bytes

sizeof(int) ⟹ 4 bytes

Char

```
char ch = 'A';
```

char ⟶ 1 byte (8 bits)

ASCII

A - 65
B - 66
:

0 - 48
1 - 49
:

a - 97
b - 98
:

A → 65_{10} → 0x41 → 0 1 0 0 | 0 0 0 1

16 | 65
4 - 1

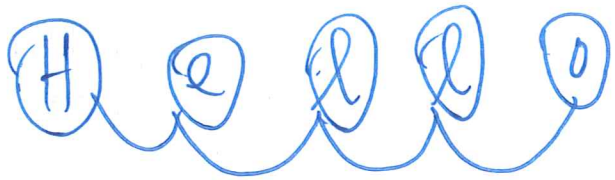
a → 97 → 0 1 1 0 0 0 0 1
128 64 32 16 8 4 2 1

0 → 0x30 = 0 0 1 1 0 0 0 0

16 | 48
3 - 0

Strings

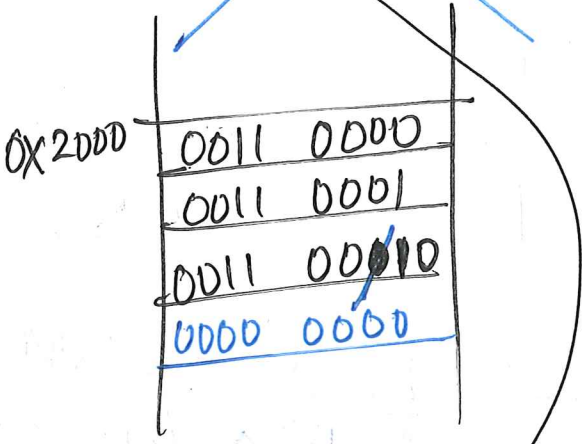
"Hello"

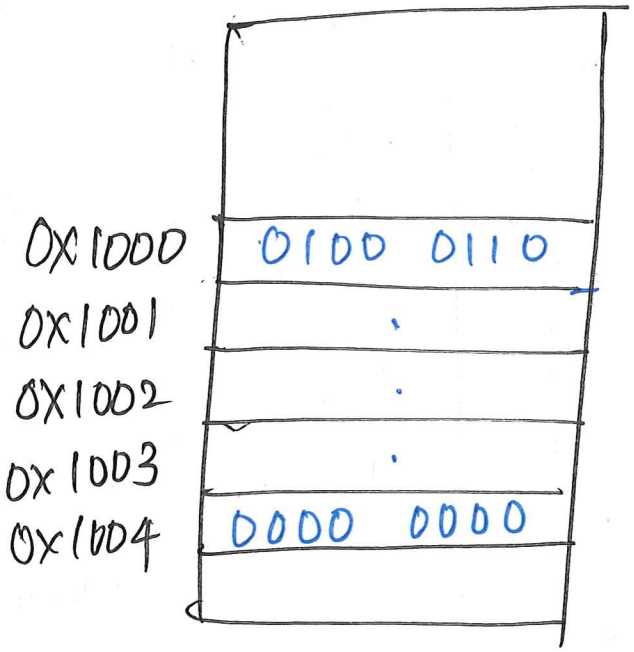



```
char s[5] = "EPIC";
```

	0	1	2	3	4
S:	'E'	'P'	'I'	'C'	'\0'

```
char n[3] = "012";
```






```
char n[4] = "012";
```

8 bits → 2^8 → 256 characters.

⑥

Operators

- Arithmetic: +, -, *, /, %
- Relational: <, <=, >, >=, ==, !=
- Logical: &&, ||, !

X	Y	X && Y	X || Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Operators

- inc/dec: ++, --
- assignment: =, +=, -=, *=, /=, %=
- bitwise operators: &, |, ^, ~, <<, >>

2. Conditionals

```
if (expr) {
}
```

Java → boolean ← True / False

C → No boolean type.

→ 0 ⟹ F

anything other than zero → T

```
if (0) {
}
```

→ False

```
if (1) {
}
```

→ True

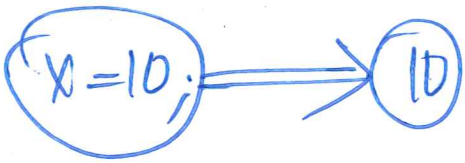
NULL → 0

```
if (n1 - n2) {
}
```

n1 = 1
n2 = 2

1 − 2 = −1

```
x = 0;
if (x = 10) {
        y = 20;
}
```



x = 10 y = 20

```
if (c1) {
} else if (c2) {
} else if (c3) {
} else {
}
```

```
switch (expr) {
    case const-expr:
        break;
        .
        .
        .
    default:
        break;
}
```

int denom = 0; ⑨ num = 12;

if (denom != 0 && num / denom) {

F

short circuiting

}

denom != 0 ⟹ 3 != 0 ⟹ 1

denom = 0

denom != 0 ⟹ 0 != 0 ⟹ 0

c1 || c2

Loops

- do-while
- for
- while

```
while (expr) {
    ____
    ____
    ____
}
```

```
for (init expr ; condition ; post-block stmt) {
    ____
    ____
    ____
}
```

```
int i = 0;
for (i = 0; i <= 99; i++) {
    printf("Hi");
}
```

```
do {
    ____
    ____
    ____
} while (cond);
```

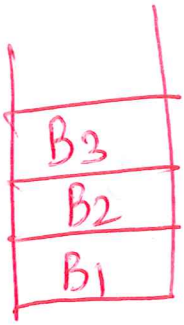
3. Functions

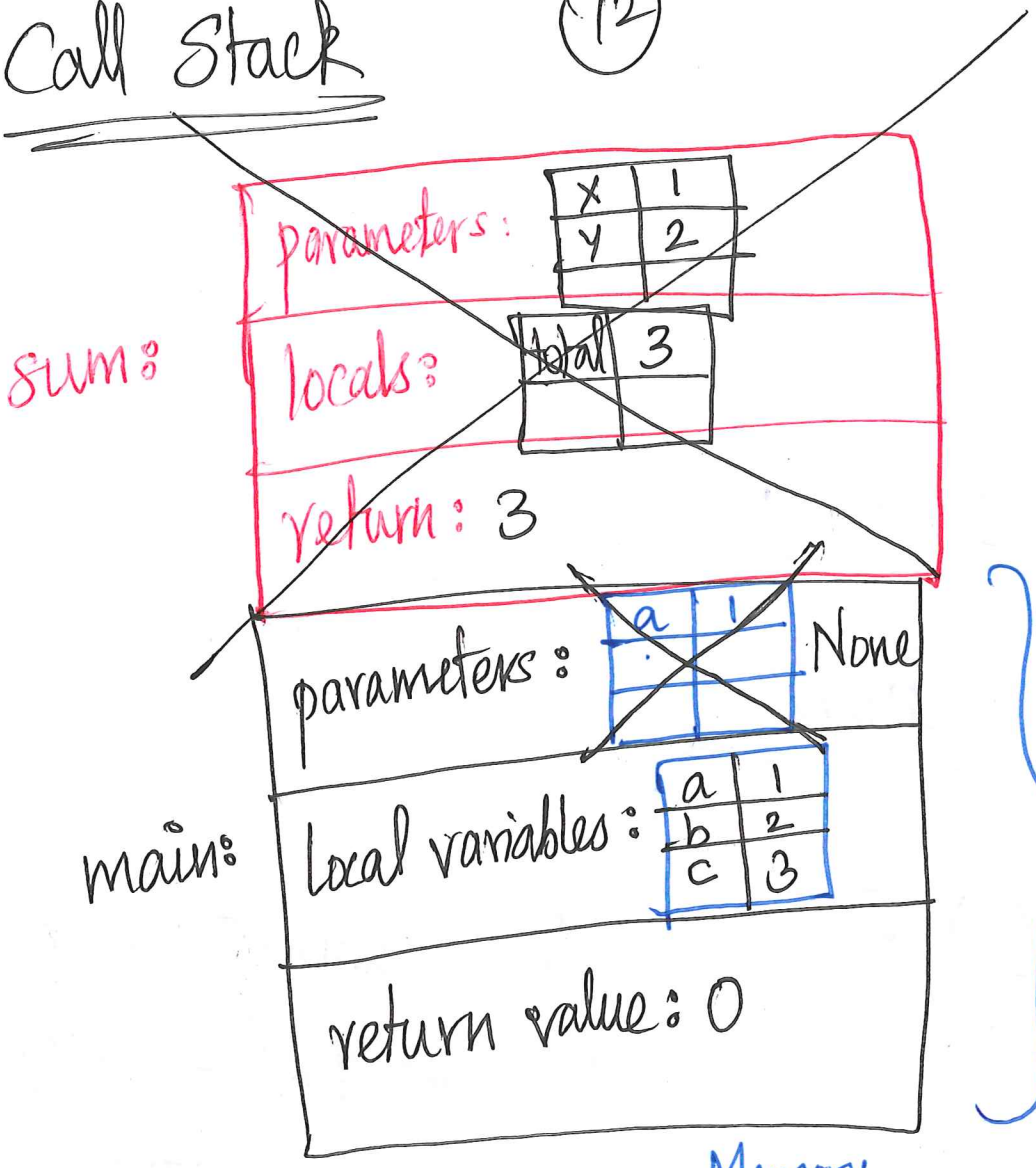
```
int g = 1;
int main() {
    int a = 1;
    int b = 2;
    int c = a+b;
    printf("c = %d\n", c);
    return 0;
}
```

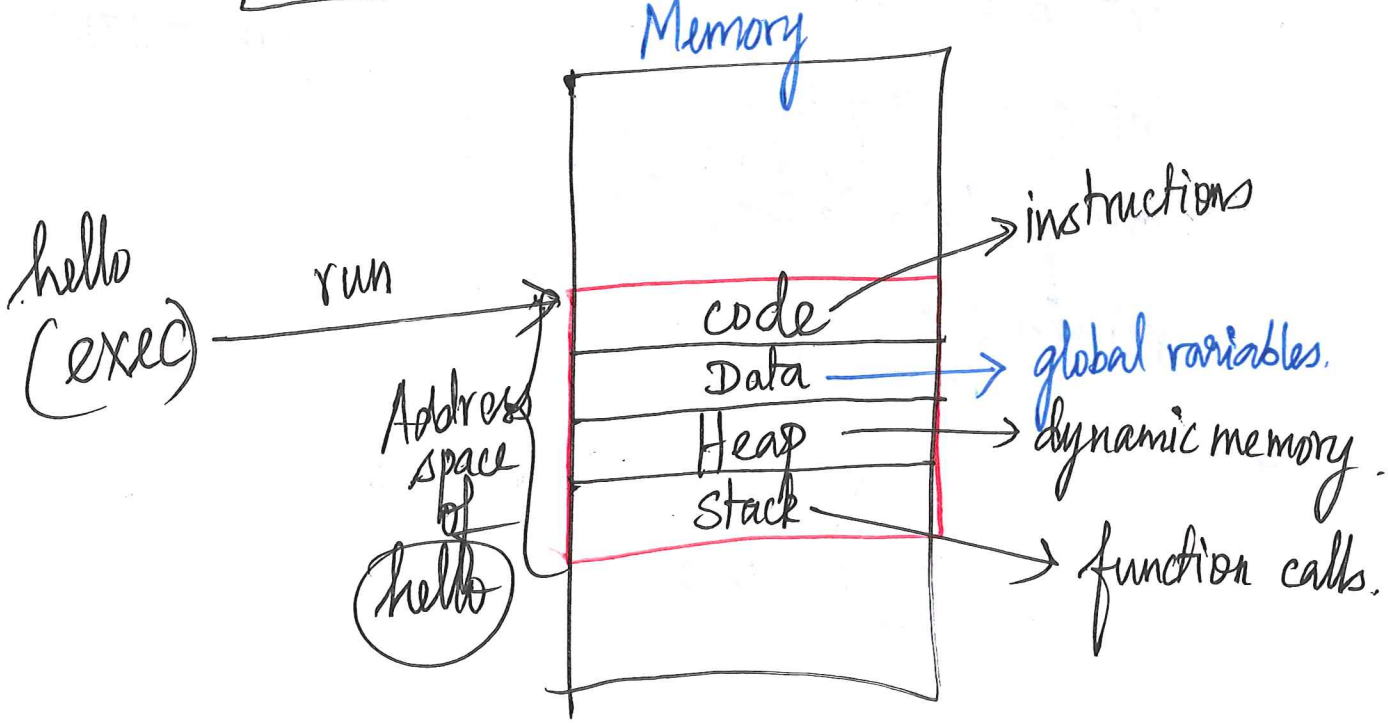
sum(a, b);

```
int sum(int x, int y) {
    int total = x+y;
    return total;
}
```

Call Stack


B3
B2
B1


sum:
parameters:
x 1
y 2
locals:
total 3
return: 3
main:
parameters:
a 1
None
Local variables:
a 1
b 2
c 3
return value: 0
stack frame

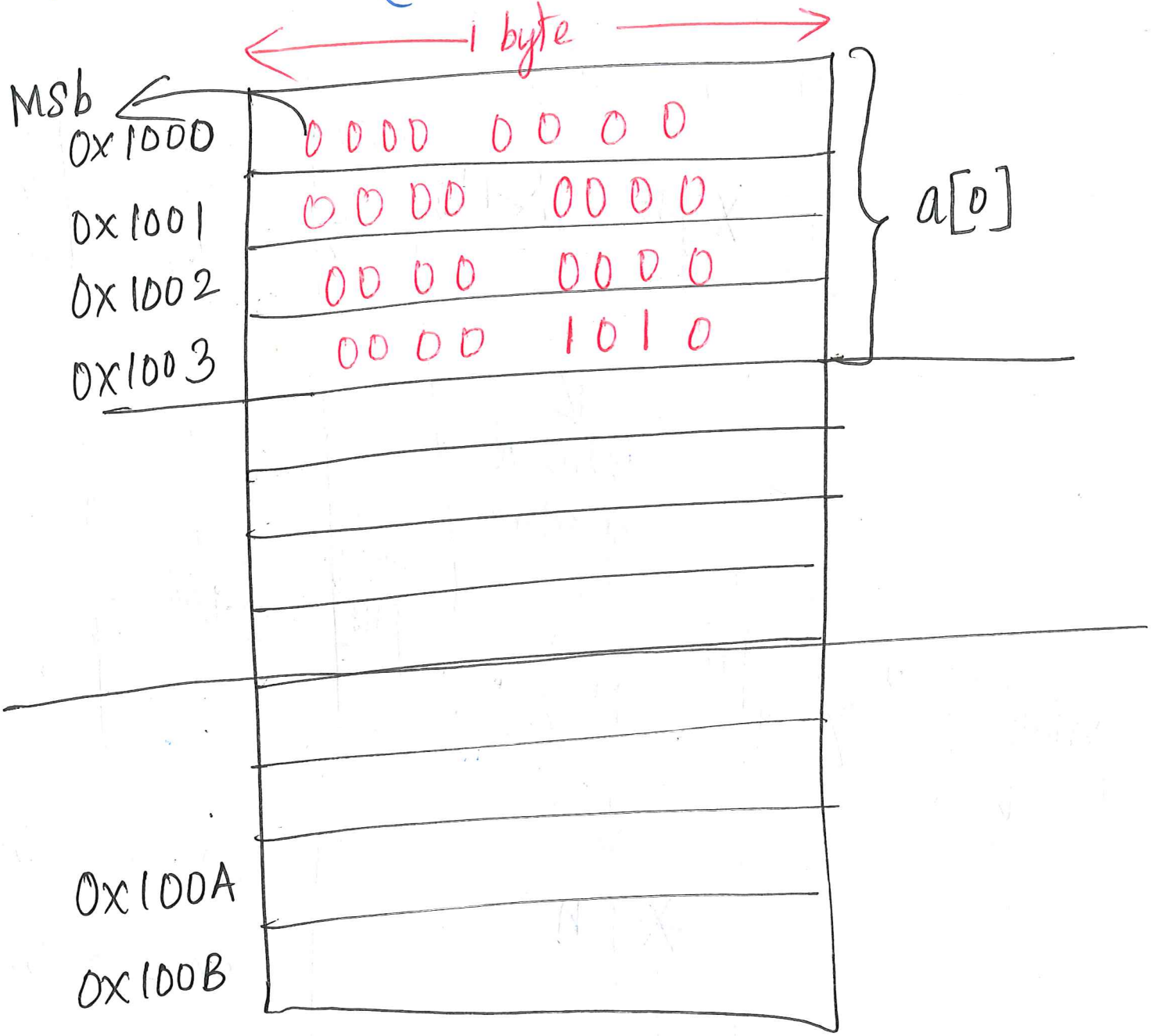

Memory
hello
(exec)
run
Address space of hello
code
Data
Heap
Stack
instructions
global variables.
dynamic memory.
function calls.

Arrays + Pointers!

int → 4 bytes

0 1 2 3 4

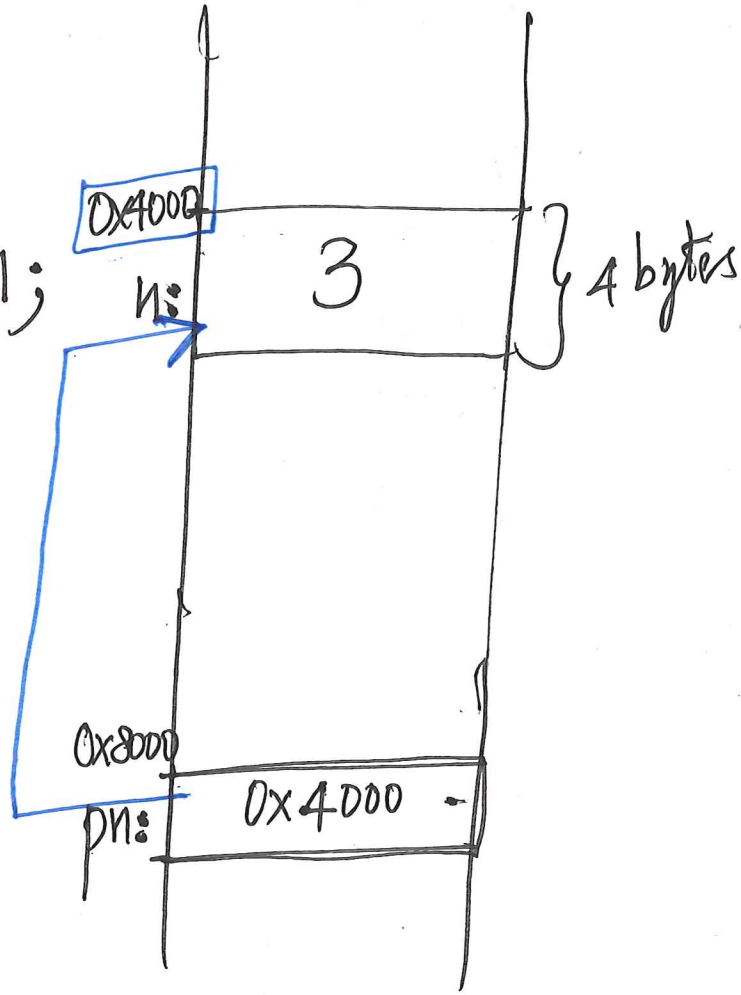
```
int a[5] = {10, 20, 30, 40, 50};
```



Pointers

A pointer is a variable that stores the address of a variable.

```
int n = 3;
int *pn = &n;
```



address of operator

```
printf("%d", n);
```

*pn

indirection operator or dereferencing

&n → 0x4000

pn → 0x4000