The Secret to Mr C's Success

ESC101: Fundamentals of Computing

Purushottam Kar

Announcement

- Advanced Track group meeting
- This Friday evening – details over email
- Please think carefully about your project and come
- Will discuss tools, tips, techniques during meeting
- Short meeting 10 – 15 minutes only ☺

Number Systems

Number Systems

We grow up learning the decimal number system

Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

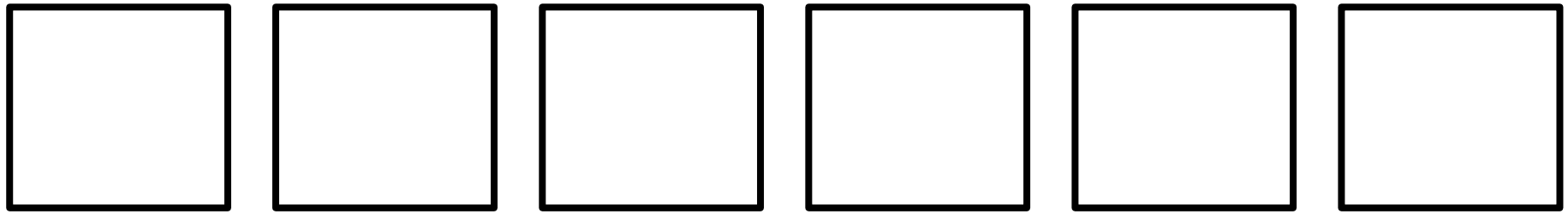
Based on the place value system

Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

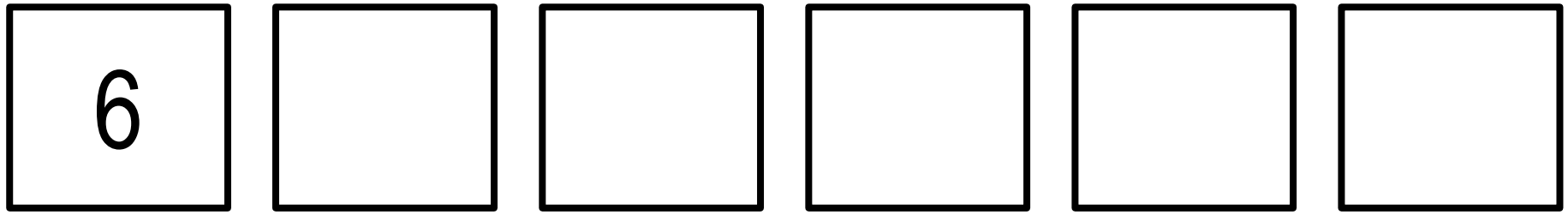


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

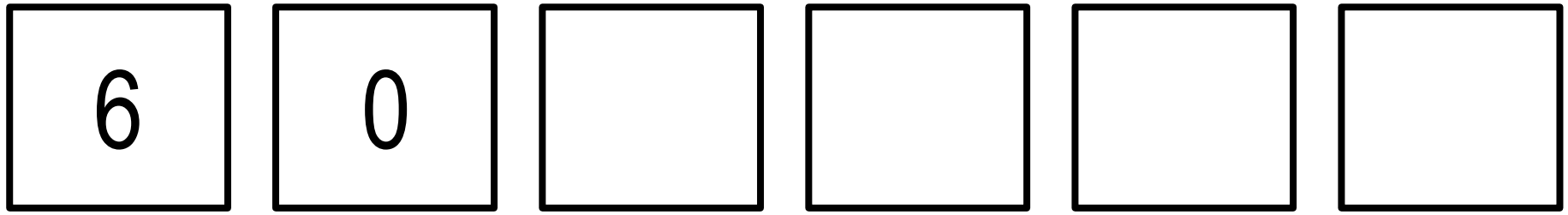


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

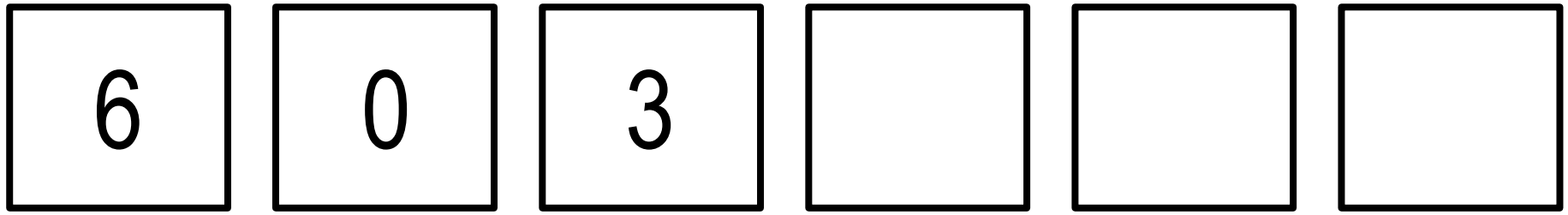


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

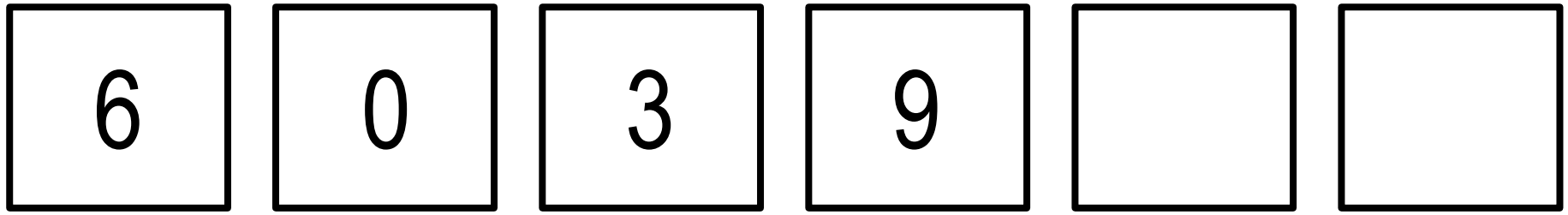


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

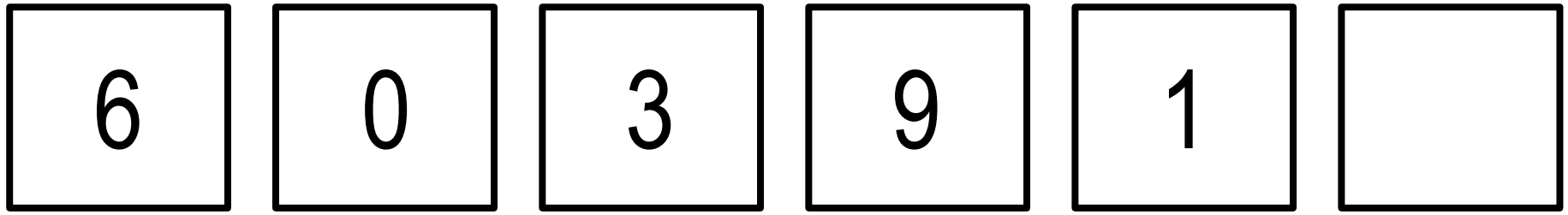


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

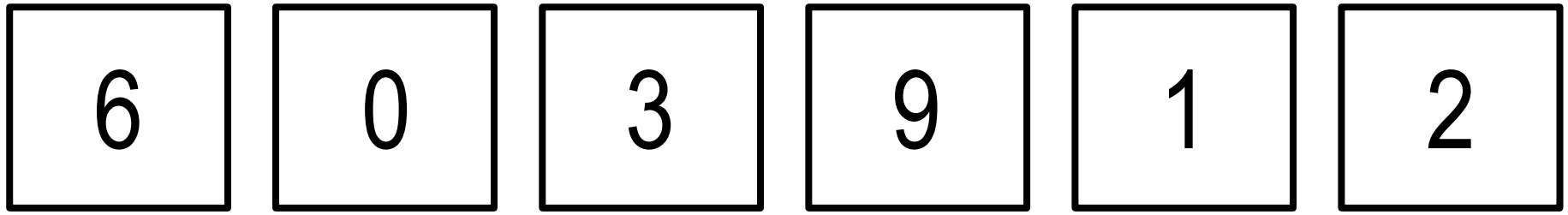


Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system



Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

Number Systems

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

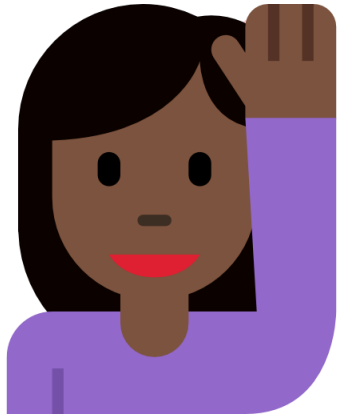
6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Number Systems



We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Is there anything special about having 10 digits?

Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

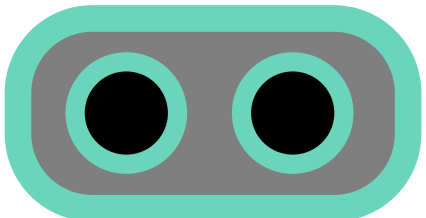
6	0	3	9	1	2

$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Is there anything special about having 10 digits?



Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.

We grow up learning the decimal number system

Developed hundreds of years ago by wise people

Based on the place value system

6	0	3	9	1	2

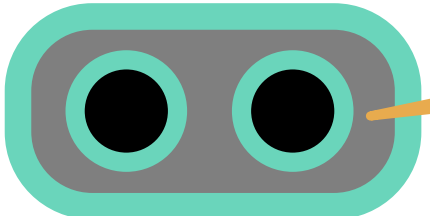
$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

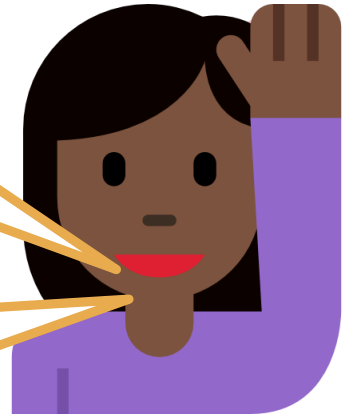
Is there anything special about having 10 digits?

Nope! In fact, I prefer a number system with just 2 digits!



Number Syster

It is called the *decimal* system since it has 10 digits 0 … 9 from the Latin word *decimus* which means *the tenth*.



The word *binary* comes from the Latin word *bini* which means *two together*

We grow up learning t

Developed hundreds

Based on the place value system

6	0	3	9	1	2

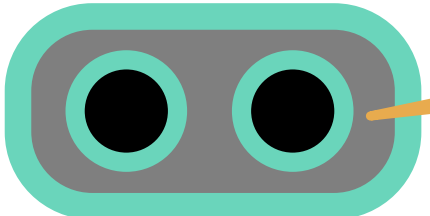
$= 2 \times 10^0 + 1 \times 10^1 + 9 \times 10^2 + 3 \times 10^3 + 0 \times 10^4 + 6 \times 10^5$

$= 2 + 10 + 900 + 3000 + 0 + 60000$

$= 603912$

Is there anything special about having 10 digits?

Nope! In fact, I prefer a number system with just 2 digits!



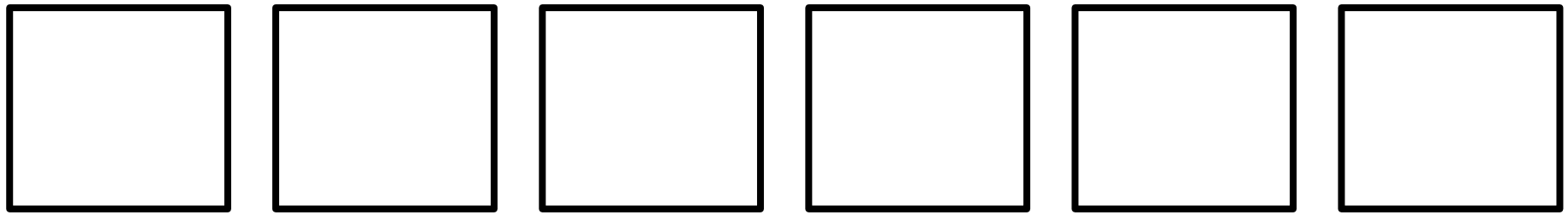
The Octal Number System

The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7

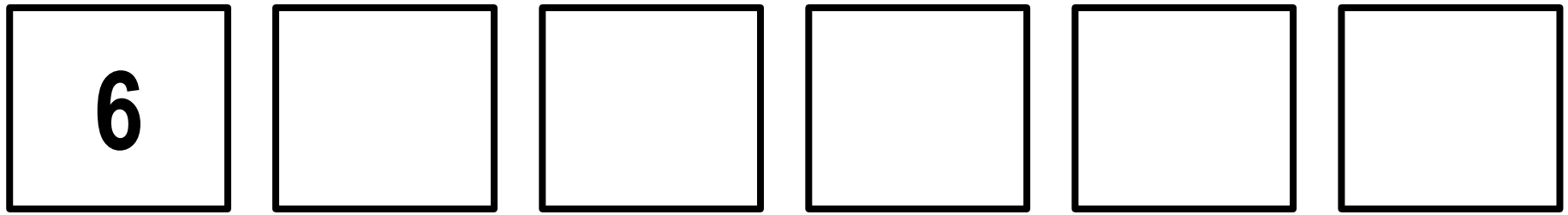
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



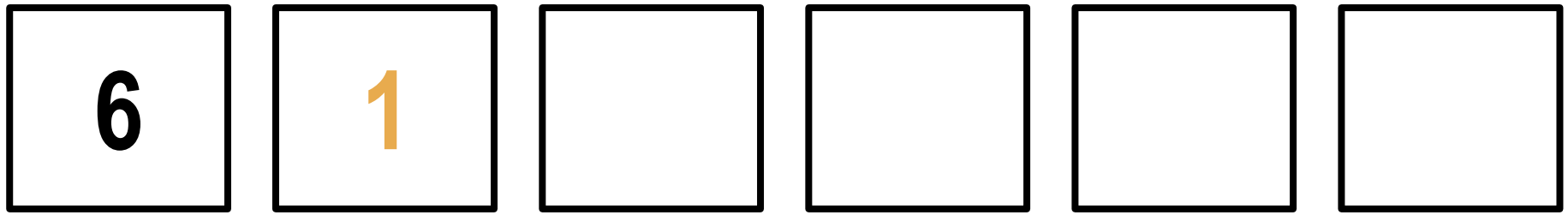
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



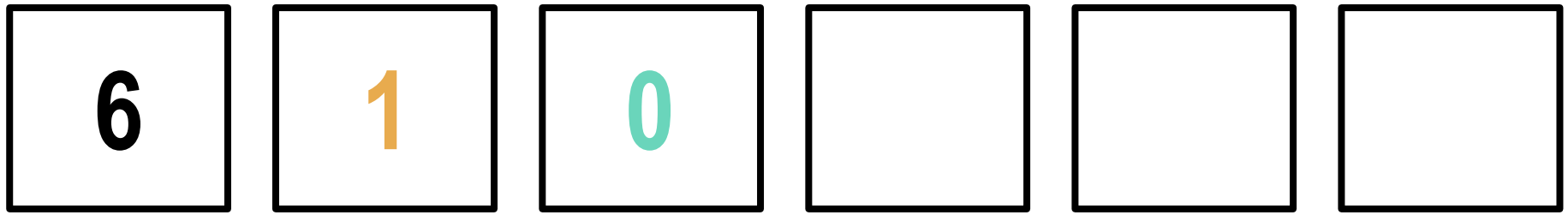
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



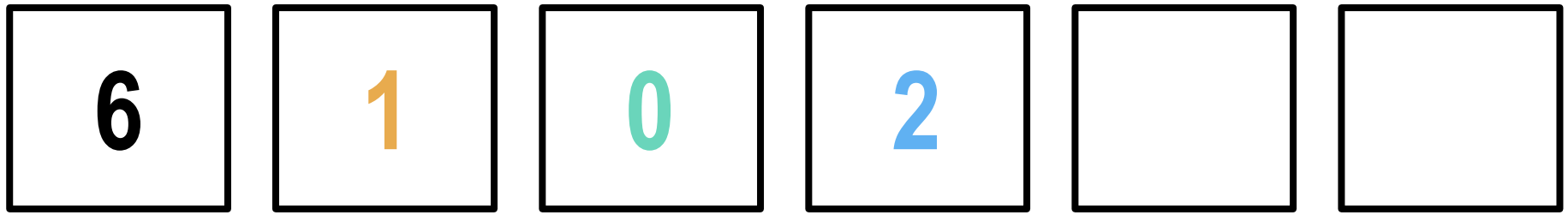
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



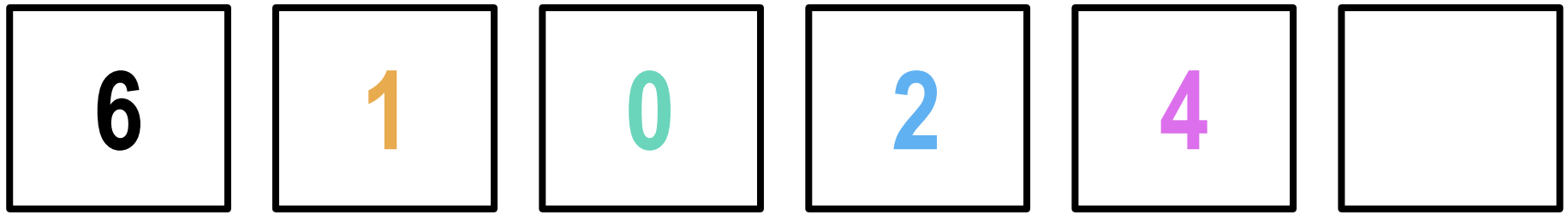
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



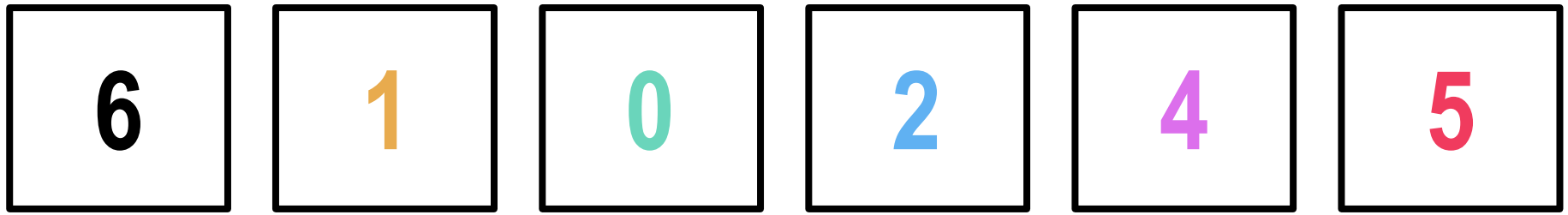
The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7

6	1	0	2	4	5

$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7

6	1	0	2	4	5

$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

$= 5 + 32 + 128 + 0 + 4096 + 196608$

The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7

6	1	0	2	4	5

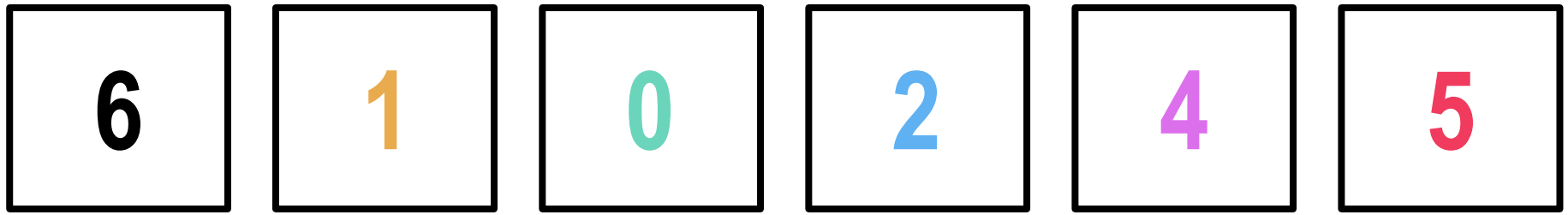
$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

$= 5 + 32 + 128 + 0 + 4096 + 196608$

$= 200869$

The Octal Number System

Just eight digits – 0,1,2,3,4,5,6,7



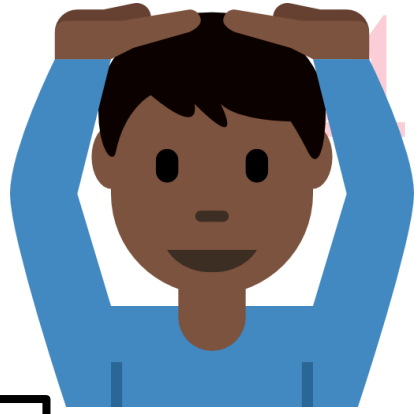
$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

$= 5 + 32 + 128 + 0 + 4096 + 196608$

$= 200869$

Can read and print integers in octal format directly using scanf and printf – use the format specifier %o

The Octal Number System



Just eight digits – 0,1,2,3,4,5,6,7

6	1	0	2	4	5

$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

$= 5 + 32 + 128 + 0 + 4096 + 196608$

$= 200869$

Can read and print integers in octal format directly using scanf and printf – use the format specifier %o

The Octal N

Aha! That is why we have been using %d for integers so far – d for decimal ☺

Just eight digits – 0,1,2,3,4,5,6,7

6	1	0	2	4	5

$= 5 \times 8^0 + 4 \times 8^1 + 2 \times 8^2 + 0 \times 8^3 + 1 \times 8^4 + 6 \times 8^5$

$= 5 + 32 + 128 + 0 + 4096 + 196608$

$= 200869$

Can read and print integers in octal format directly using scanf and printf – use the format specifier %o

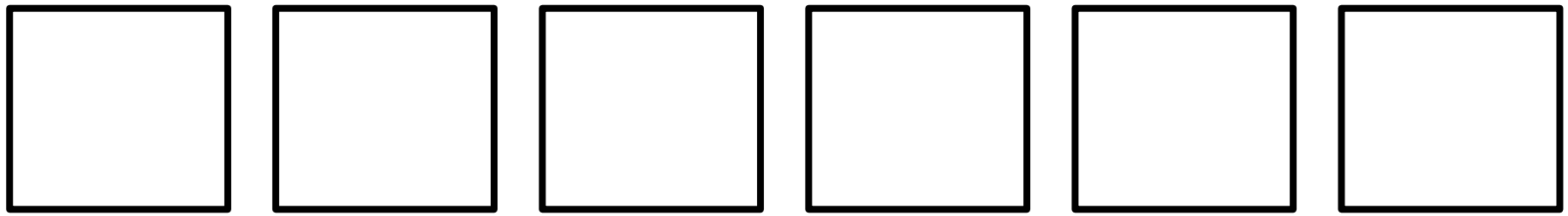
The Hexadecimal Number System

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

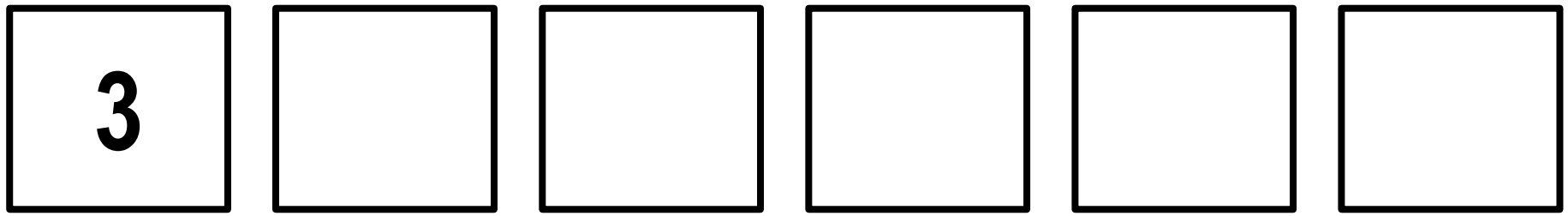
The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F



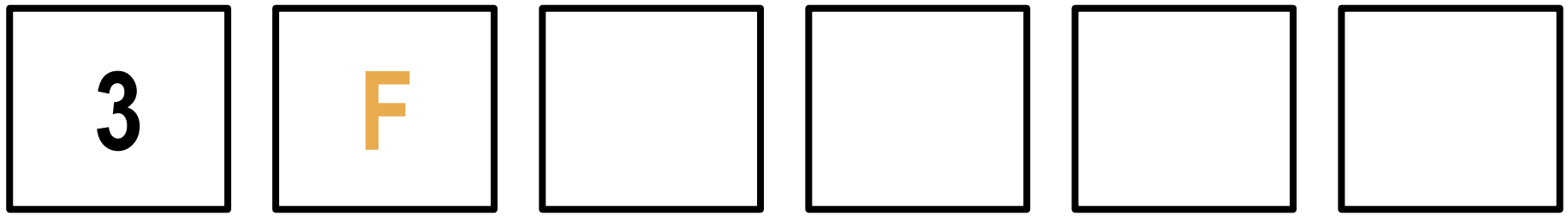
The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F



The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F



The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + 10 \times 16^3 + 15 \times 16^4 + 3 \times 16^5$

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + 10 \times 16^3 + 15 \times 16^4 + 3 \times 16^5$

$= 9 + 0 + 1536 + 40960 + 983040 + 3145728$

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + 10 \times 16^3 + 15 \times 16^4 + 3 \times 16^5$

$= 9 + 0 + 1536 + 40960 + 983040 + 3145728$

$= 4171273$

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + 10 \times 16^3 + 15 \times 16^4 + 3 \times 16^5$

$= 9 + 0 + 1536 + 40960 + 983040 + 3145728$

$= 4171273$

Can read and print integers in hex format directly %X, %x

The Hexadecimal Number System

"Just" sixteen digits – 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3	F	A	6	0	9

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + A \times 16^3 + F \times 16^4 + 3 \times 16^5$

$= 9 \times 16^0 + 0 \times 16^1 + 6 \times 16^2 + 10 \times 16^3 + 15 \times 16^4 + 3 \times 16^5$

$= 9 + 0 + 1536 + 40960 + 983040 + 3145728$

$= 4171273$

Can read and print integers in hex format directly %X, %x

%X if you want A, B, etc and %x if you want a, b, etc.

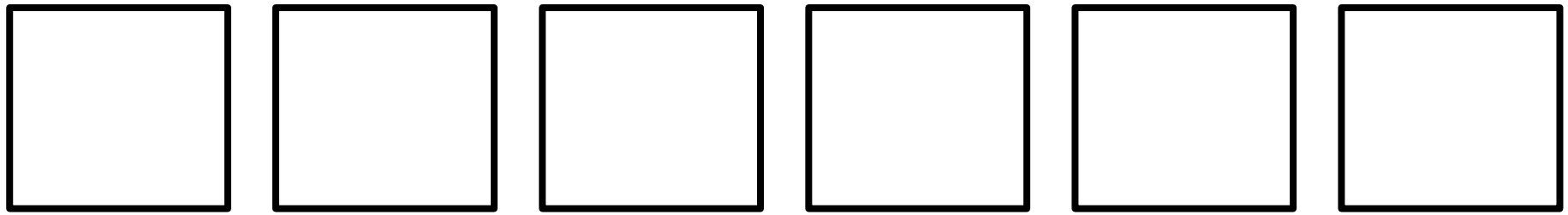
The Binary Number System

The Binary Number System

Just two digits – 0 and 1

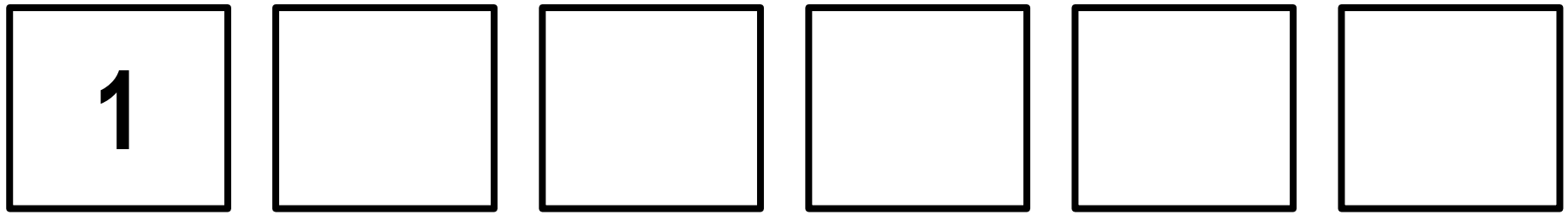
The Binary Number System

Just two digits – 0 and 1



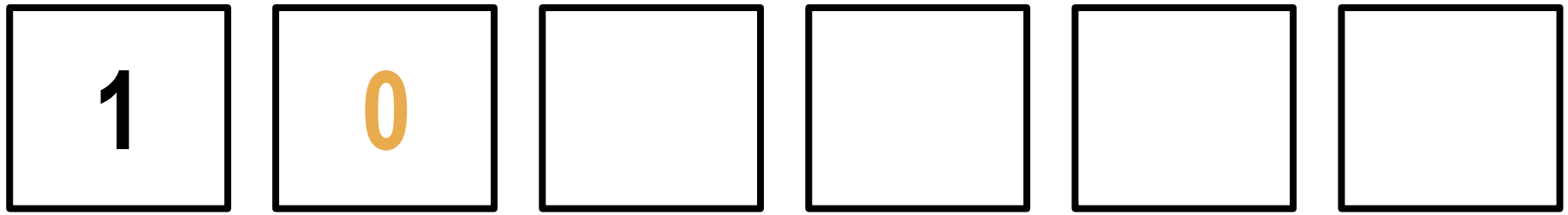
The Binary Number System

Just two digits – 0 and 1



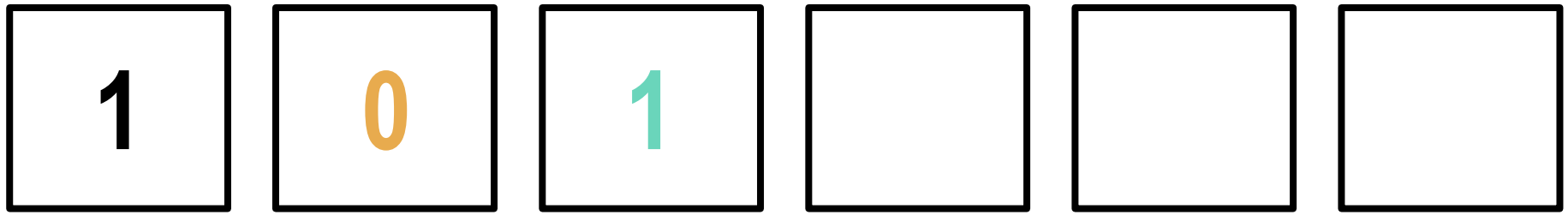
The Binary Number System

Just two digits – 0 and 1



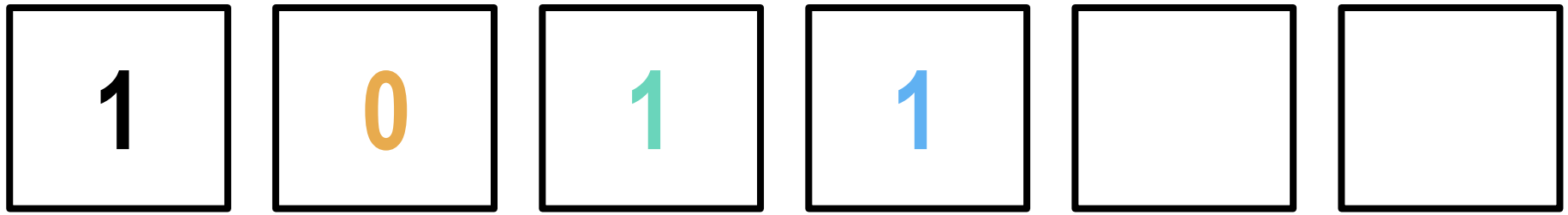
The Binary Number System

Just two digits – 0 and 1



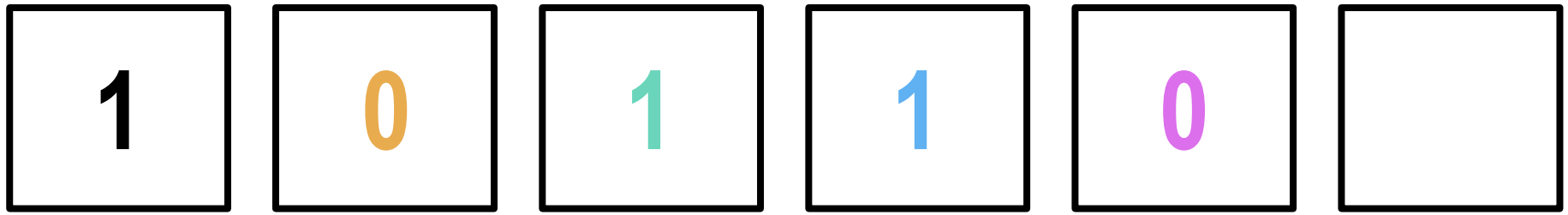
The Binary Number System

Just two digits – 0 and 1



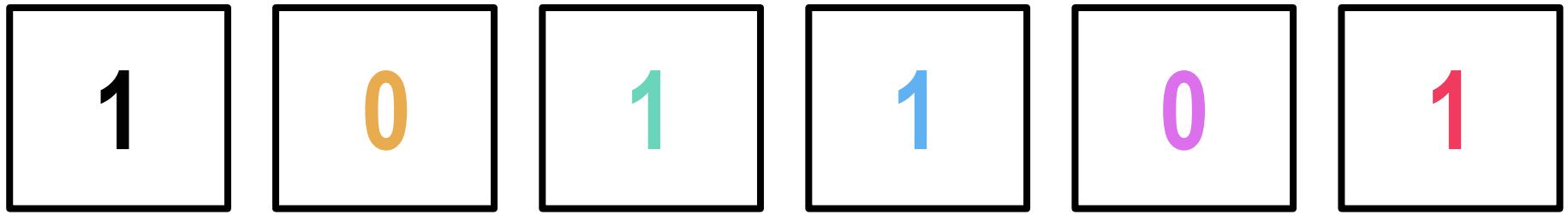
The Binary Number System

Just two digits – 0 and 1



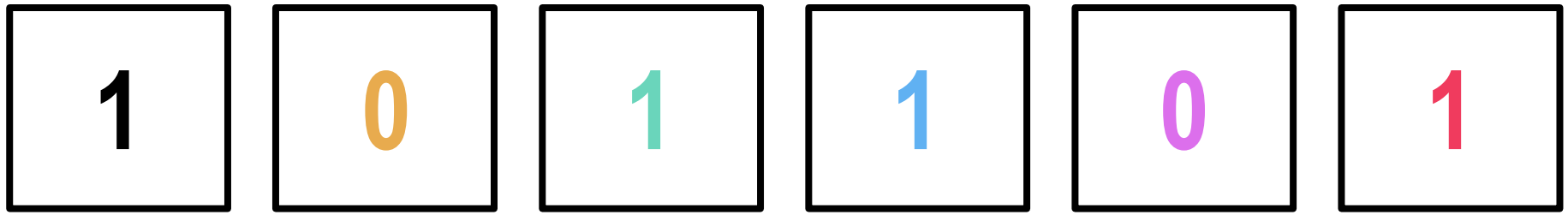
The Binary Number System

Just two digits – 0 and 1



The Binary Number System

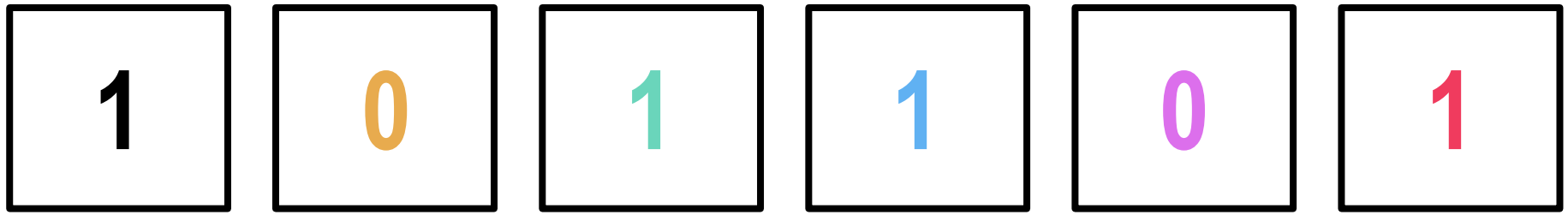
Just two digits – 0 and 1



$= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$

The Binary Number System

Just two digits – 0 and 1

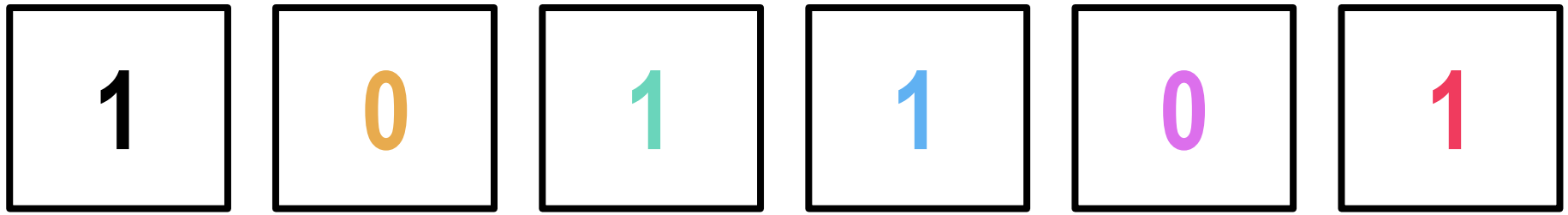


$= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$

$= 1 + 0 + 4 + 8 + 0 + 32$

The Binary Number System

Just two digits – 0 and 1



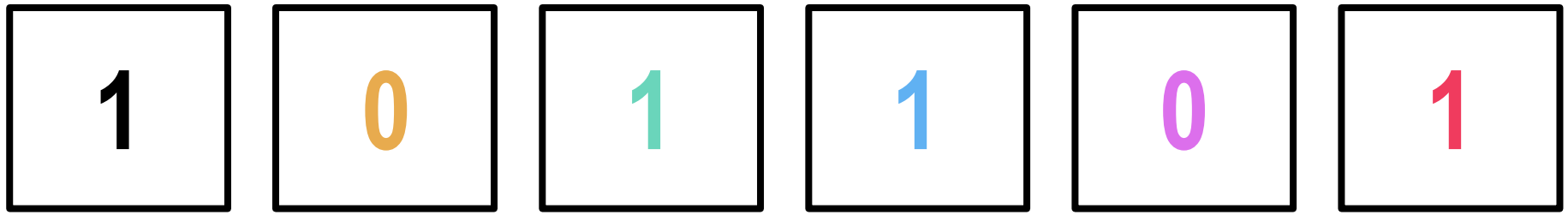
$= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$

$= 1 + 0 + 4 + 8 + 0 + 32$

$= 45$

The Binary Number System

Just two digits – 0 and 1



$= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$

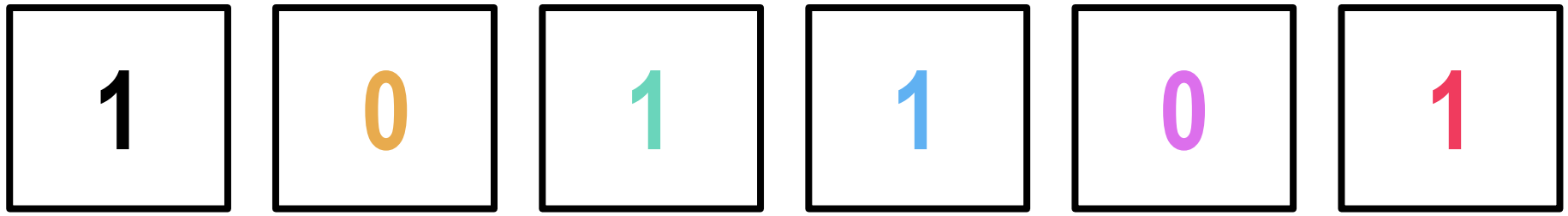
$= 1 + 0 + 4 + 8 + 0 + 32$

$= 45$

No direct way to read or print integers in binary format ☺

The Binary Number System

Just two digits – 0 and 1



$= 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$

$= 1 + 0 + 4 + 8 + 0 + 32$

$= 45$

No direct way to read or print integers in binary format ☺

Write a program to take an integer in decimal system and print its binary representation

A word about number systems

A word about number systems

Octal, Binary, Decimal, Hexadecimal are just different systems of representing integers

A word about number systems

Octal, Binary, Decimal, Hexadecimal are just different systems of representing integers

The same integer can be represented using any system

A word about number systems

Octal, Binary, Decimal, Hexadecimal are just different systems of representing integers

The same integer can be represented using any system

Can add, subtract, divide, multiply, take remainder with numbers in any of the systems – use them as plain int

A word about number systems

Octal, Binary, Decimal, Hexadecimal are just different systems of representing integers

The same integer can be represented using any system

Can add, subtract, divide, multiply, take remainder with numbers in any of the systems – use them as plain int

Can represent negative integers using all these systems as well – more tricky, will be covered later in course

A word about number systems

Octal, Binary, Decimal, Hexadecimal are just different systems of representing integers

The same integer can be represented using any system

Can add, subtract, divide, multiply, take remainder with numbers in any of the systems – use them as plain int

Can represent negative integers using all these systems as well – more tricky, will be covered later in course

Can represent fractional numbers as well – explore yourself if you are interested

A word about number systems

A word about number systems

13895 can be interpreted as a

A word about number systems

13895 can be interpreted as a

Valid hexadecimal number – will have value 80021

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

Number systems have digits till one less than their base

A word about number systems

13895 can be interpreted as a

- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

Number systems have digits till one less than their base

- Binary: base 2, doesn't have a digit with value 2. Value 2 represented as 10

A word about number systems

13895 can be interpreted as a
- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

Number systems have digits till one less than their base
- Binary: base 2, doesn't have a digit with value 2. Value 2 represented as 10
- Octal: base 8, doesn't have a digit with value 8. Value 8 represented as 10

A word about number systems

13895 can be interpreted as a
- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

Number systems have digits till one less than their base
- Binary: base 2, doesn't have a digit with value 2. Value 2 represented as 10
- Octal: base 8, doesn't have a digit with value 8. Value 8 represented as 10
- Decimal: base 10, doesn't have a digit with value 10

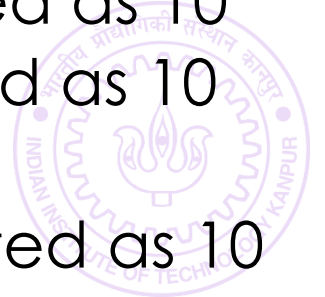
A word about number systems

13895 can be interpreted as a
- Valid hexadecimal number – will have value 80021
- Valid decimal number – will have value 13895
- Not a valid octal number (octal numbers cannot have digit 8)
- Not a valid binary number (binary numbers cannot have digit 3,8,9,5)

1011 is valid in binary, octal, decimal and hex systems

1653 is valid in octal, decimal and hex systems

Number systems have digits till one less than their base
- Binary: base 2, doesn't have a digit with value 2. Value 2 represented as 10
- Octal: base 8, doesn't have a digit with value 8. Value 8 represented as 10
- Decimal: base 10, doesn't have a digit with value 10
- Hex: base 16, doesn't have a digit with value 16. Value 16 represented as 10



Limits in various number systems

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

- Largest number in binary system will be 1111 … 1111 with value $2^k - 1$

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

Largest number in binary system will be $\overbrace{1111 \ldots 1111}^{k}$ with value $2^k - 1$

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

Largest number in binary system will be 1111 … 1111 (k) with value $2^k - 1$

Largest number in octal system will be 7777 … 7777 with value $8^k - 1$

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

Largest number in binary system will be	1111 … 1111 (k)	with value $2^k - 1$
Largest number in octal system will be	7777 … 7777	with value $8^k - 1$
Largest number in decimal system will be	9999 … 9999	with value $10^k - 1$

Limits in various number systems

Just as we have decimal digits 0,1,…,9, the other number systems also have digits

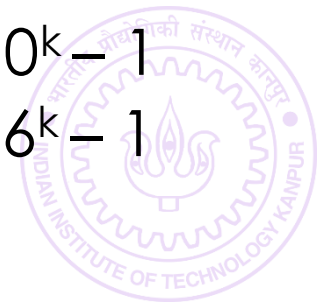
Octal number system has 8 *octal digits* 0,1,…,7

Hexademical system has 16 *hex digits* 0,1,…,9,A,B,…F

Binary system has two *binary digits* 0 and 1

Suppose we allow only k digits

Largest number in binary system will be	1111 … 1111 (k)	with value $2^k - 1$
Largest number in octal system will be	7777 … 7777	with value $8^k - 1$
Largest number in decimal system will be	9999 … 9999	with value $10^k - 1$
Largest number in hex system will be	FFFF … FFFF	with value $16^k - 1$



How Mr C stores your variables

How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

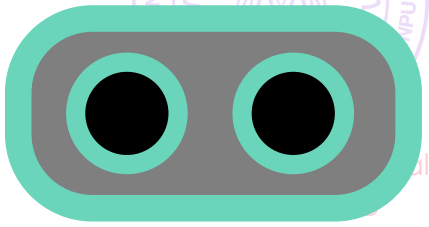
*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*

How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*



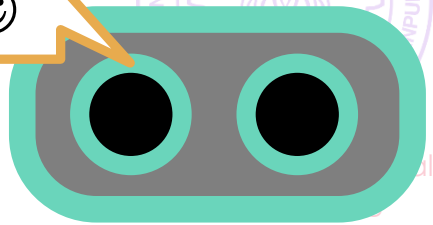
How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*

I lead an otherwise very boring life – I need to amuse myself ☺



How Mr C stores your variables

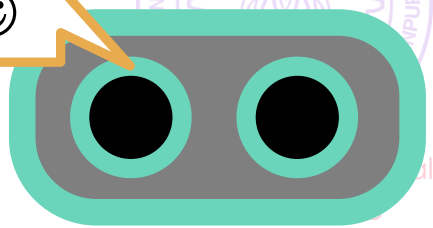
Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*

All variables, int, long, char, float, double, arrays are stored in binary format using one or more bytes

I lead an otherwise very boring life – I need to amuse myself ☺



How Mr C stores your variables

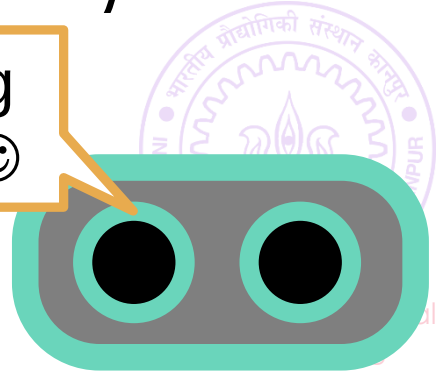
Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*

All variables, int, long, char, float, double, arrays are stored in binary format using one or more bytes

We have already seen how integers (int, long, char) can be stored in binary format



How Mr C stores your variables

Mr C loves binary digits so much I gave them a cute nickname bit – short for **b**inary dig**it**

A set of 8 bits has an even cuter nickname *byte*

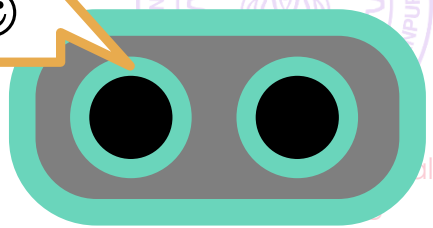
*John just **bit** into his sandwich but Harry was so hungry he took a real big **bite** out of his sandwich*

All variables, int, long, char, float, double, arrays are stored in binary format using one or more bytes

We have already seen how integers (int, long, char) can be stored in binary format

Float/double later

I lead an otherwise very boring life – I need to amuse myself ☺



The sizeof various variable types

The sizeof various variable types

8 bits □ make a byte □□□□□□□□

The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

 = □□□□□□□□

The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$

= □□□□□□□□

The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$

= □□□□□□□□

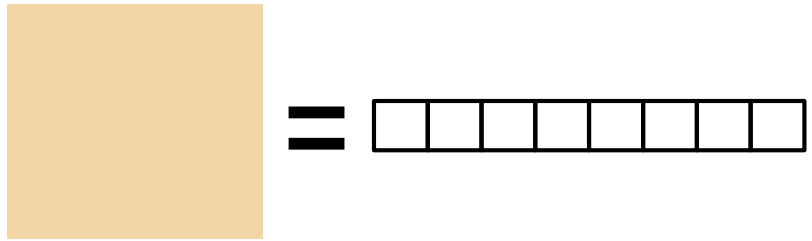
int/float takes 4 bytes = 32 bits

The sizeof various variable types

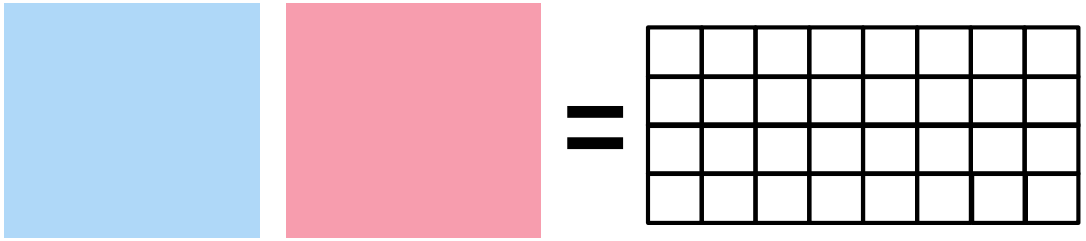
8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

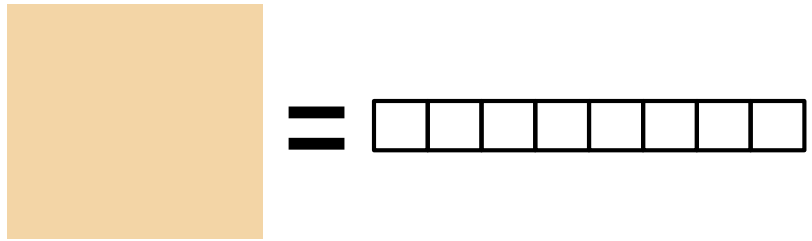


The sizeof various variable types

8 bits □ make a byte □□□□□□□□

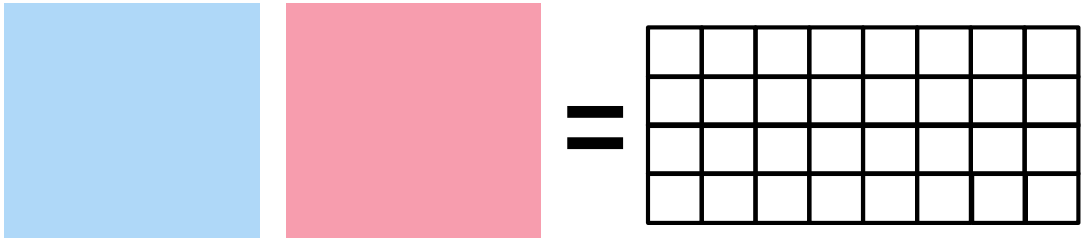
char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

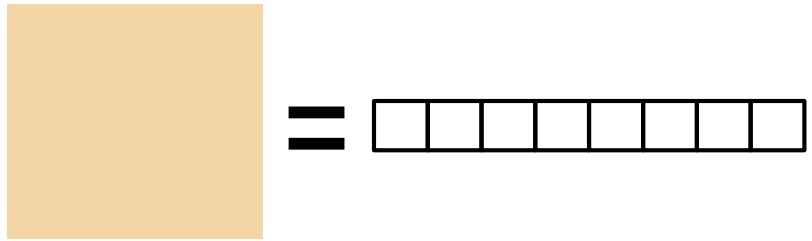


The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

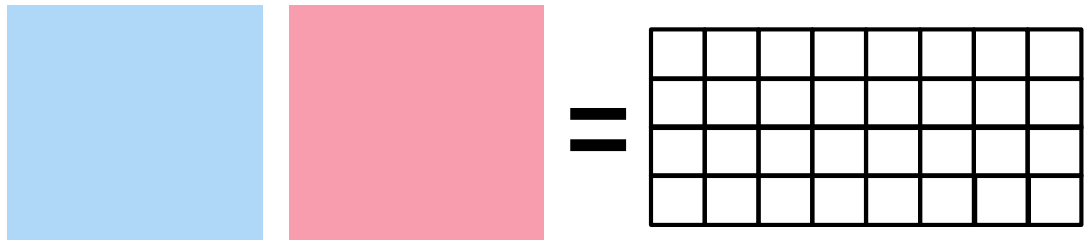
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

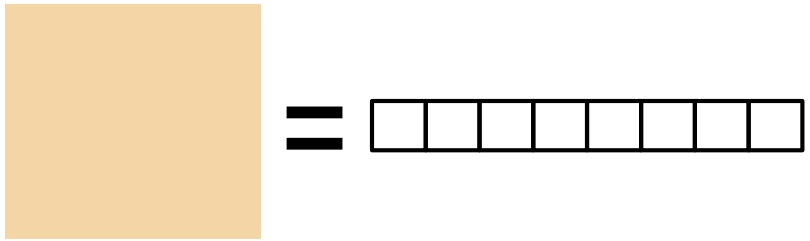


The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

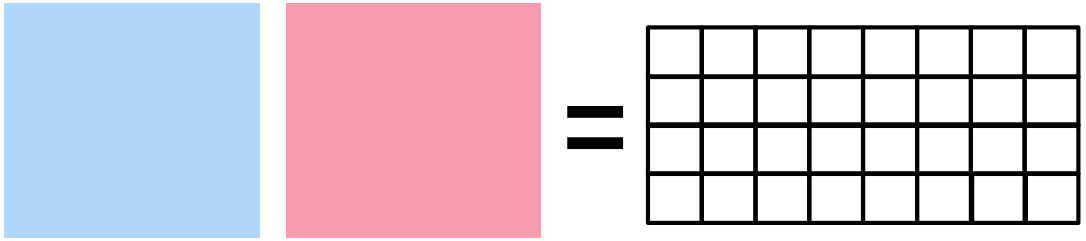
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

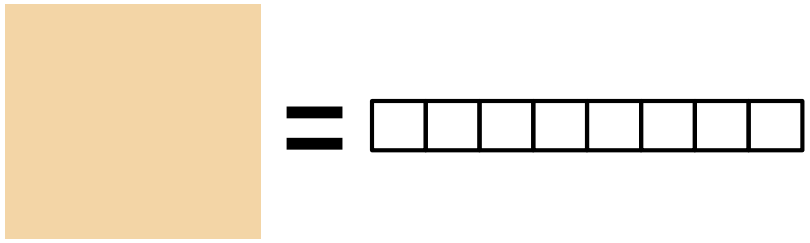
Max value of float discussed later



long/double takes 8 bytes = 64 bits

The sizeof various variable types

8 bits □ make a byte □□□□□□□□



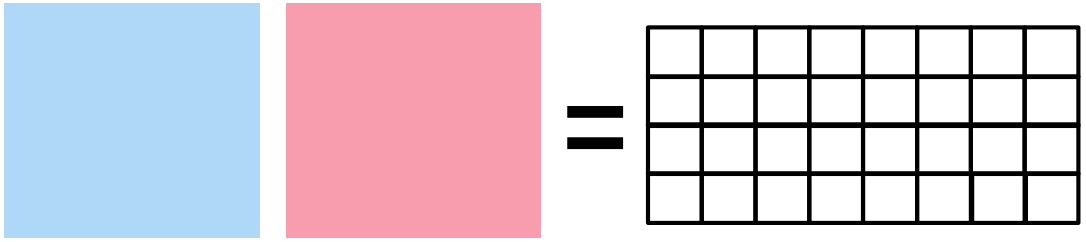
char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$

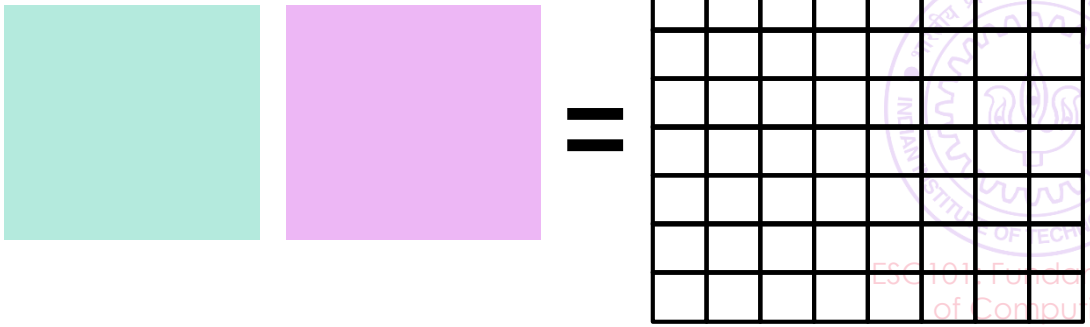
int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later



long/double takes 8 bytes = 64 bits

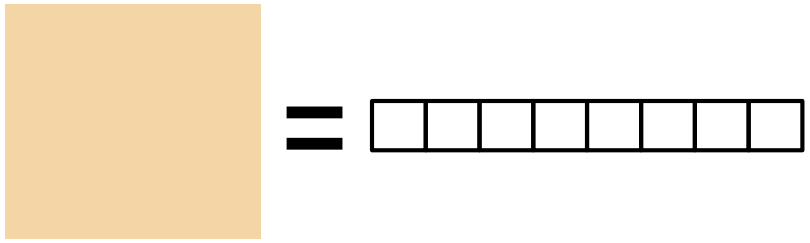


The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

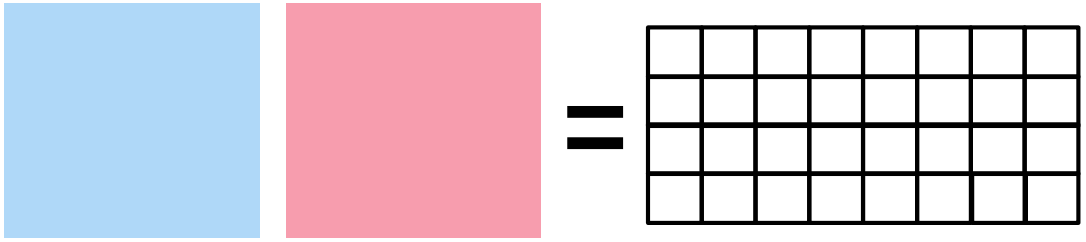
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

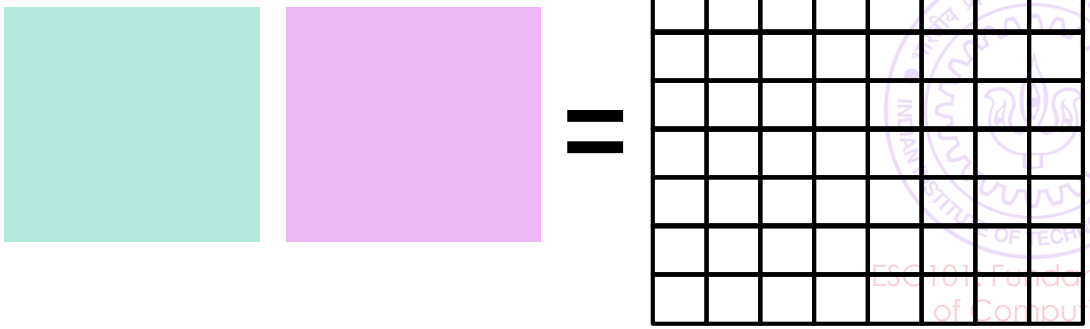
Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later



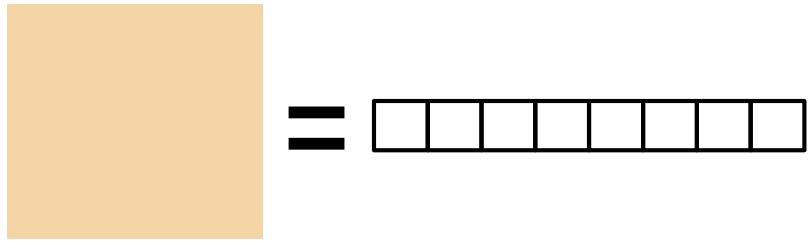
long/double takes 8 bytes = 64 bits

Max value in long is $2^{(64-1)}-1$ – verify



The sizeof various variable types

8 bits □ make a byte □□□□□□□□



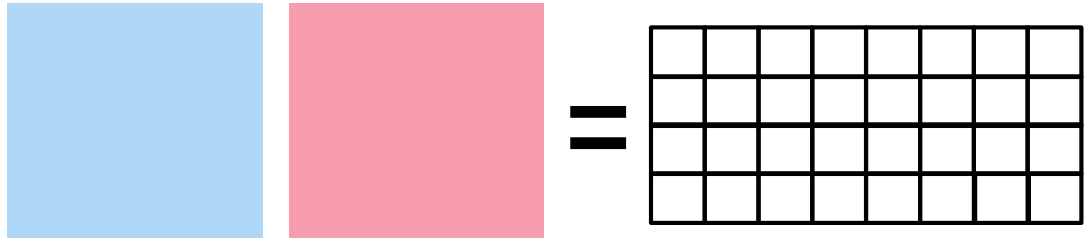
char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$

int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

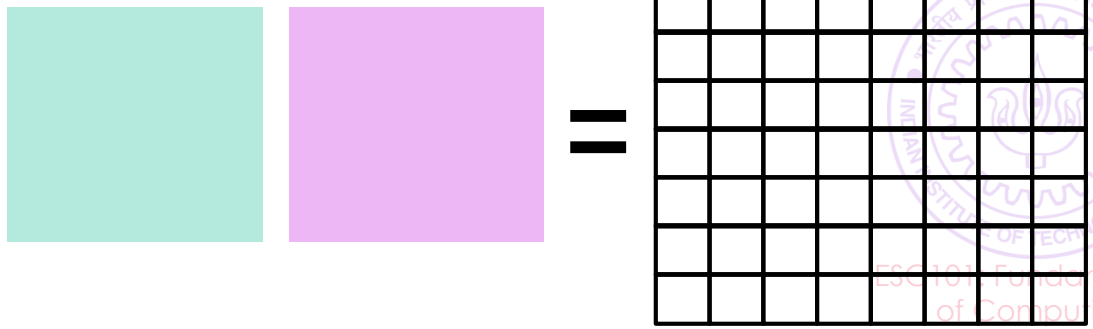
Max value of float discussed later



long/double takes 8 bytes = 64 bits

Max value in long is $2^{(64-1)}-1$ – verify

Max value of double discussed later

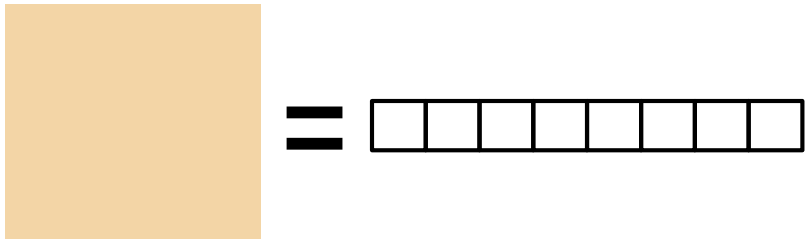


The sizeof various variable types

8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

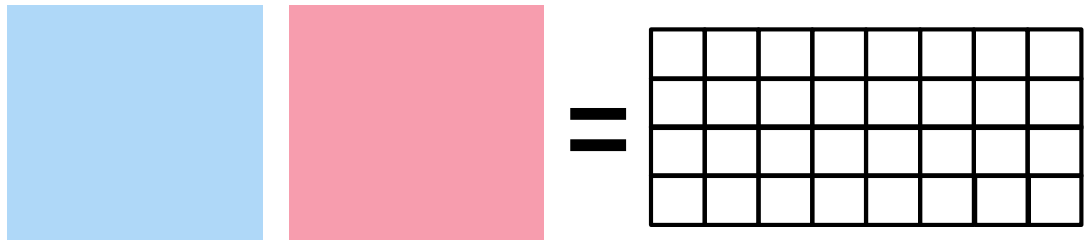
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

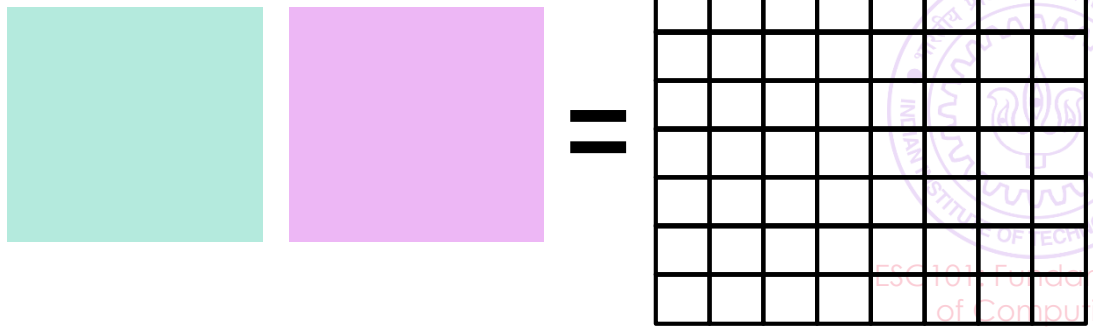
Max value of float discussed later

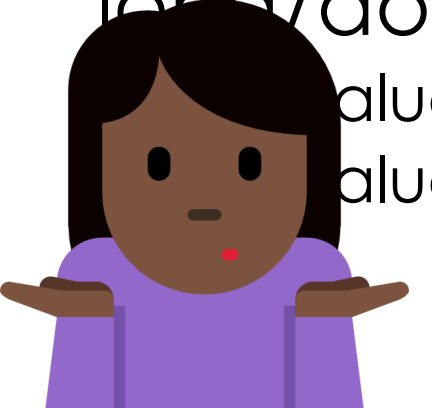


long/double takes 8 bytes = 64 bits

alue in long is $2^{(64-1)}-1$ – verify

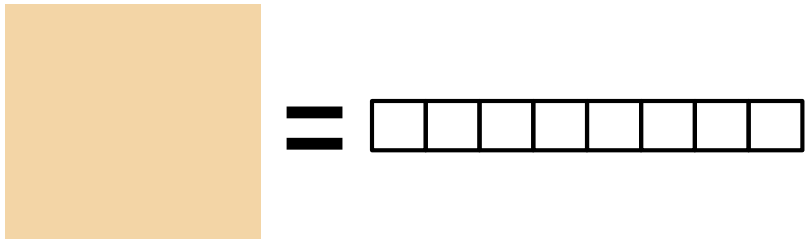
alue of double discussed later





The sizeof various variable types

8 bits □ make a byte □□□□□□□□



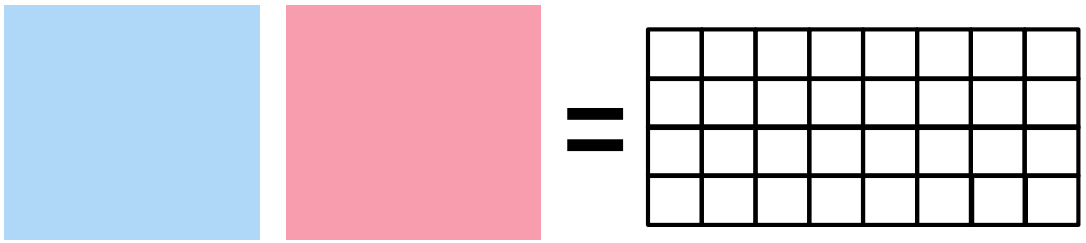
char takes 1 byte = 8 bits

Max value in a char is 127 = $2^{(8-1)}-1$

int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

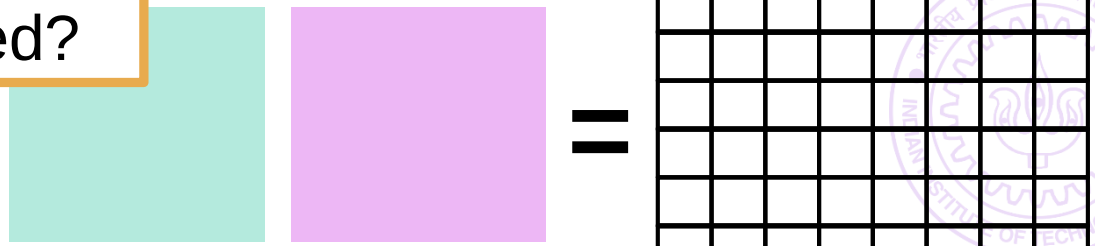


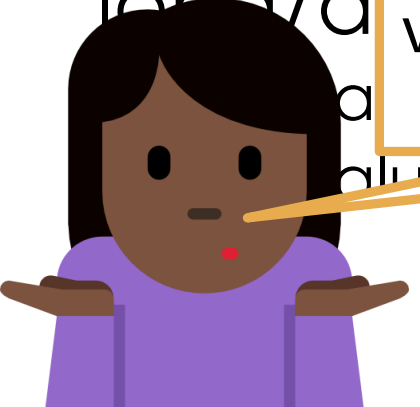
long/d… …s

a…

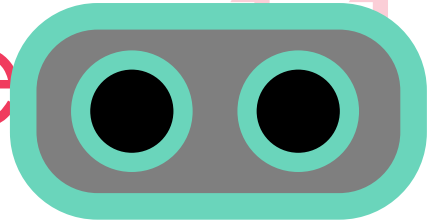
…alue of double discussed later

Why is max value for all these variables always $2^{(k-1)}-1$ and not $2^{k}-1$ when there are k bits getting used?





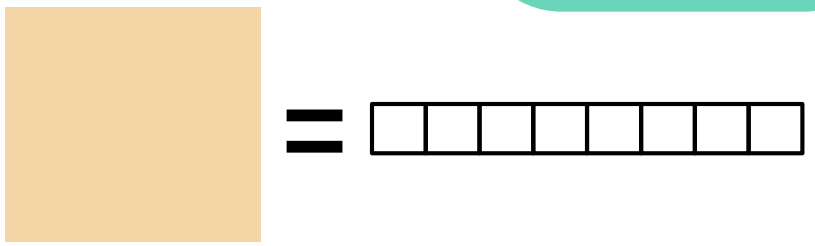
The sizeof various variable type



8 bits □ make a byte □□□□□□□□

char takes 1 byte = 8 bits

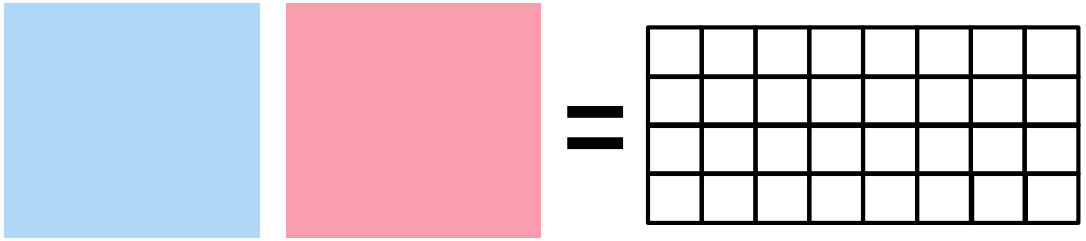
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

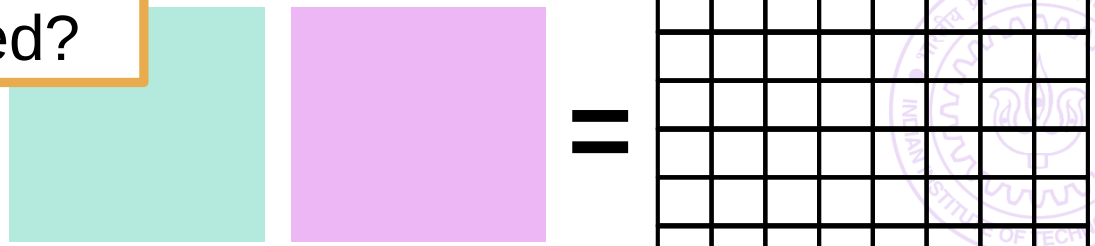


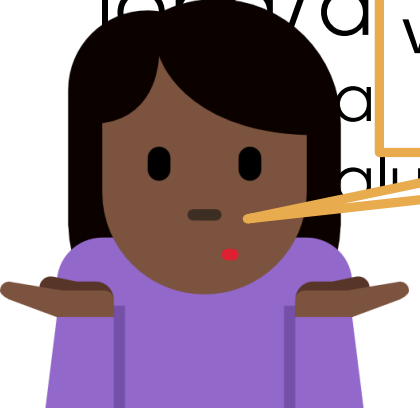
long/d… 8 bytes = 64 bits

a…

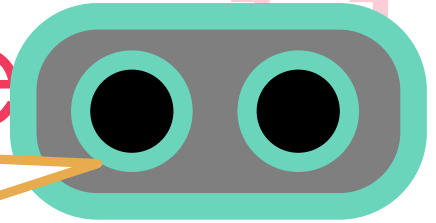
…alue of double discussed later

Why is max value for all these variables always $2^{(k-1)}-1$ and not $2^{k}-1$ when there are k bits getting used?





The sizeof various variable type

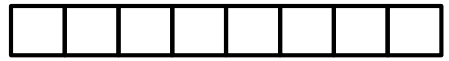


This has to do with the way I store negative numbers. Effectively, one bit gets used up in storing the sign of the number so only k-1 bits left to store the magnitude of the number

8 bits □ make

char takes 1 b

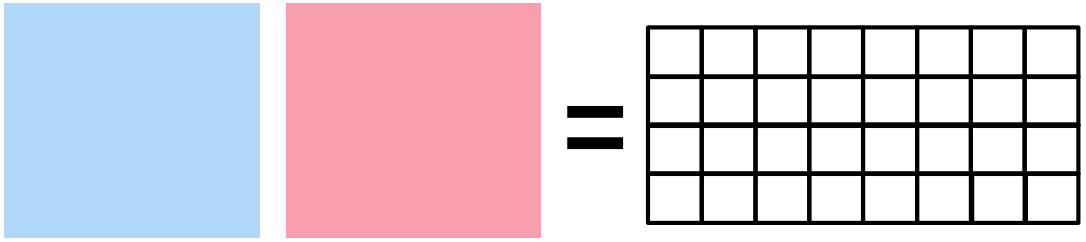
Max value in a char is 127 = $2^{(8-1)}-1$



int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

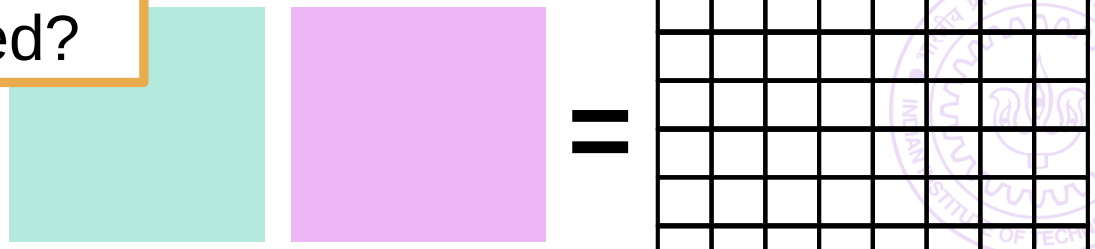


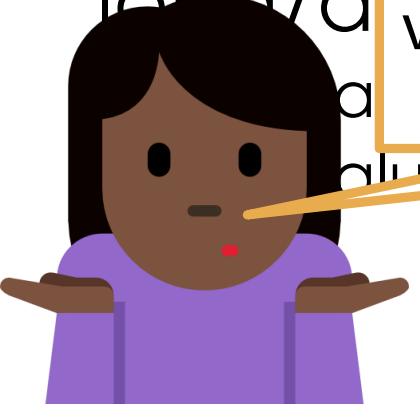
long/d s

a

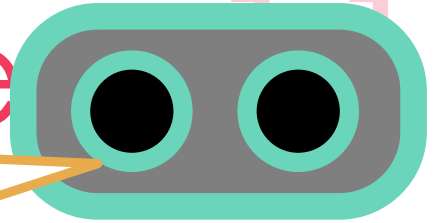
alu double discussed later

Why is max value for all these variables always $2^{(k-1)}-1$ and not $2^{k}-1$ when there are k bits getting used?





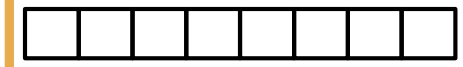
The sizeof various variable type



This has to do with the way I store negative numbers. Effectively, one bit gets used up in storing the sign of the number so only k-1 bits left to store the magnitude of the number

8 bits □ make

char takes 1 b

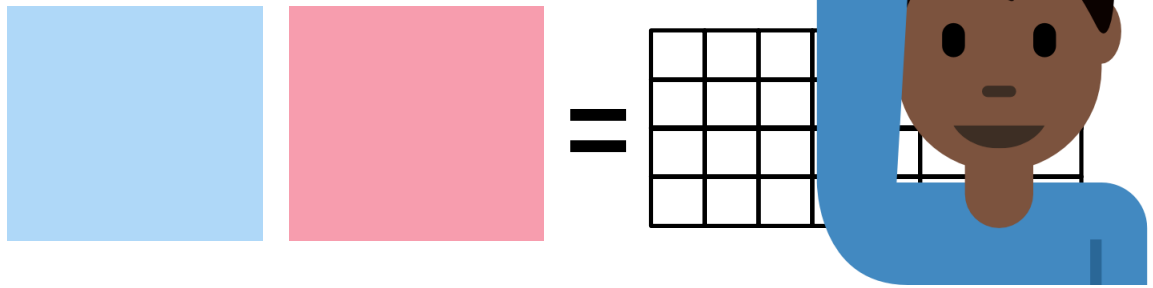


Max value in a char is 127 = $2^{(8-1)}-1$

int/float takes 4 bytes = 32 bits

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

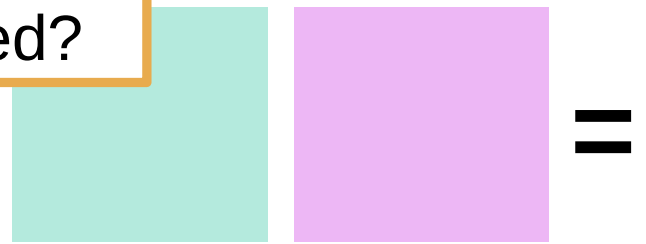


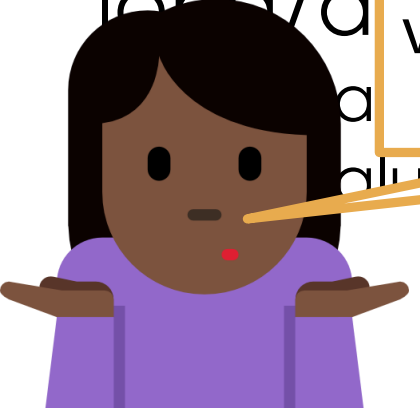
long/d s

a

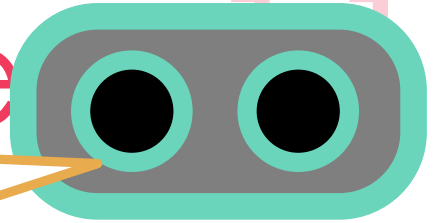
alu double discussed later

Why is max value for all these variables always $2^{(k-1)}-1$ and not $2^{k}-1$ when there are k bits getting used?





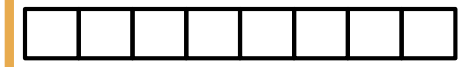
The sizeof various variable type



8 bits □ make

char takes 1 b

This has to do with the way I store negative numbers. Effectively, one bit gets used up in storing the sign of the number so only k-1 bits left to store the magnitude of the number



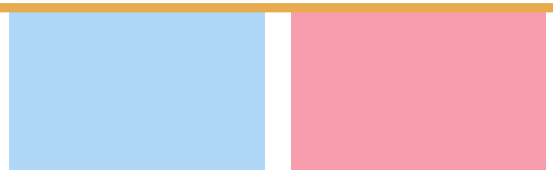
Max value in a char is 127 = $2^{(8-1)}-1$

int/float takes 4 bytes = 32 b

We will wait a few weeks to learn how negative numbers are stored

Max value in int is $2^{(32-1)}-1$ – verify

Max value of float discussed later

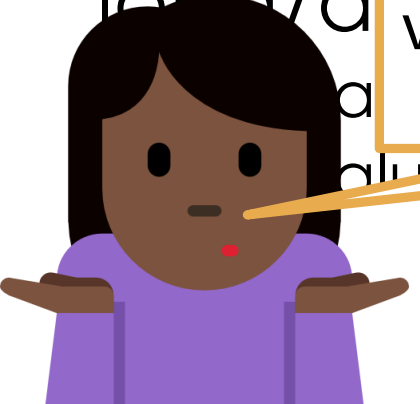


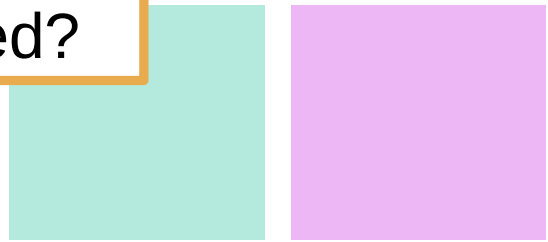
long/d

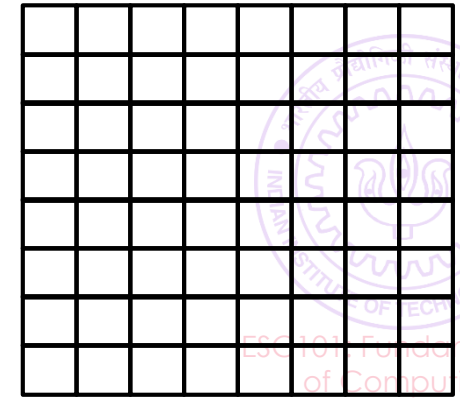
a

alue of double discussed later

Why is max value for all these variables always $2^{(k-1)}-1$ and not $2^{k}-1$ when there are k bits getting used?







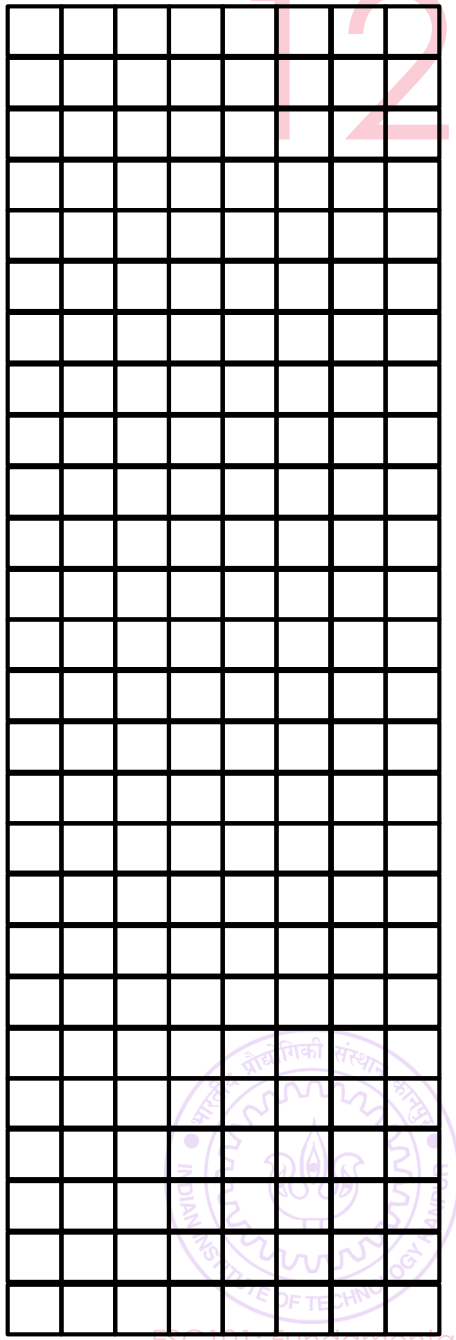
How Mr C stores variables

How Mr C stores variables

He has a very long chain of bytes

How Mr C stores variables

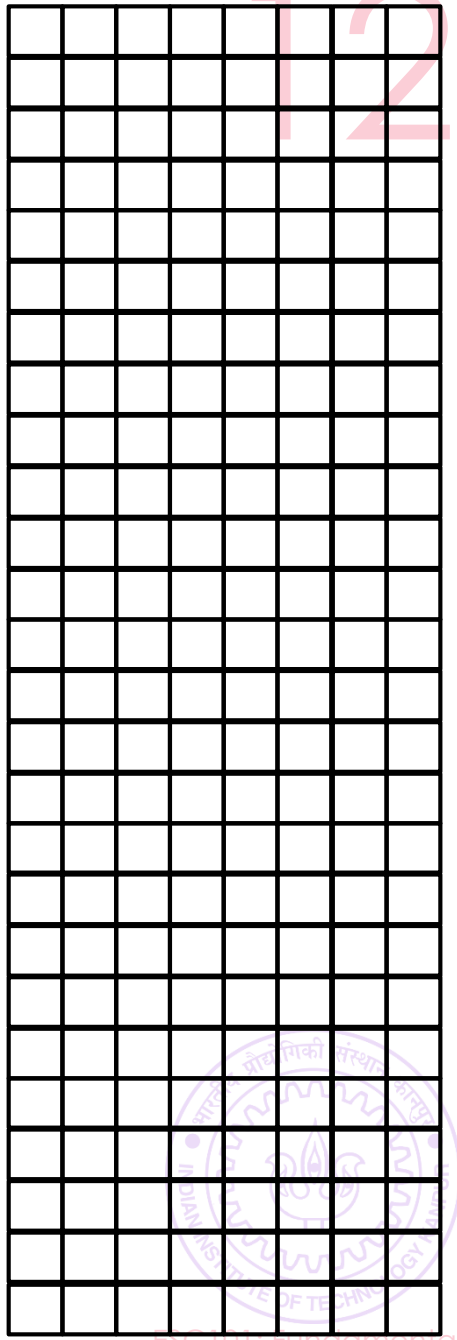
He has a very long chain of bytes



How Mr C stores variables

He has a very long chain of bytes

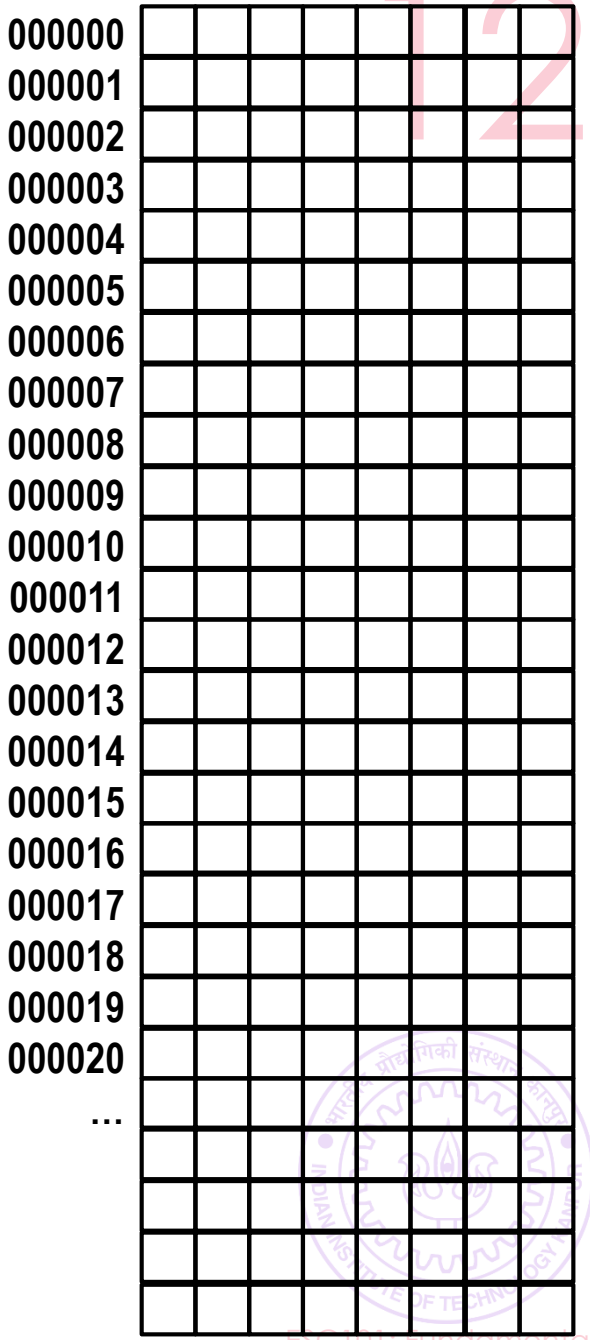
Each byte has an "address“



How Mr C stores variables

He has a very long chain of bytes

Each byte has an "address“

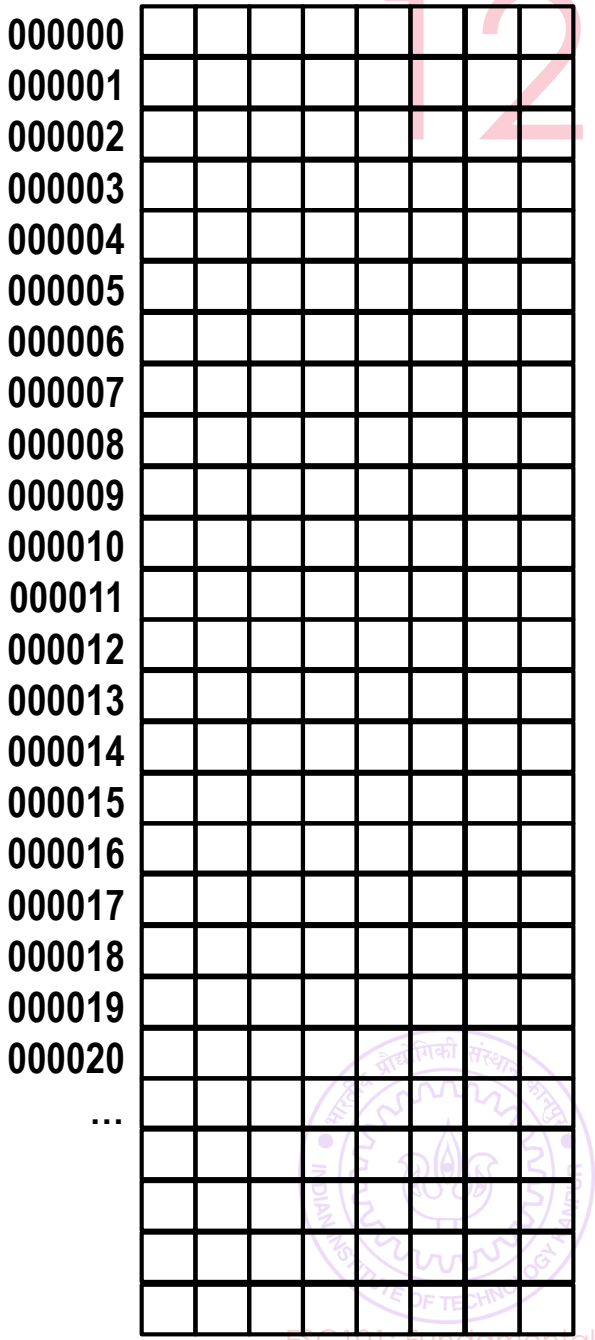


How Mr C stores variables

He has a very long chain of bytes

Each byte has an "address“

All addresses can be stored within 8 bytes



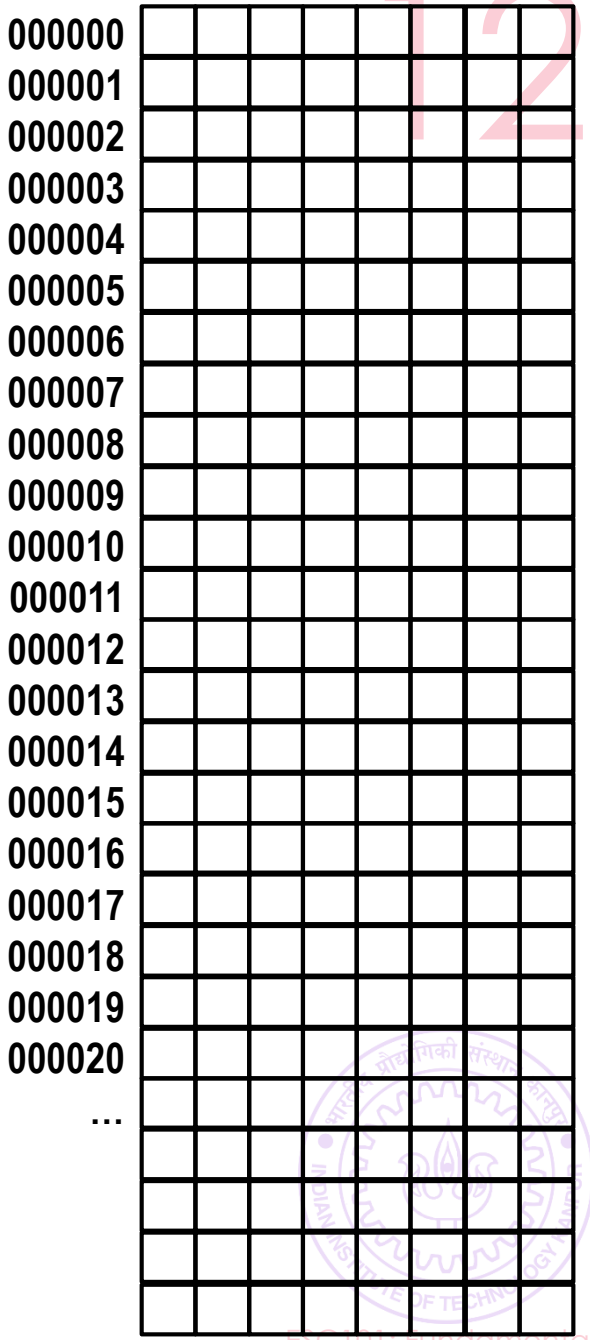
How Mr C stores variables

He has a very long chain of bytes

Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C



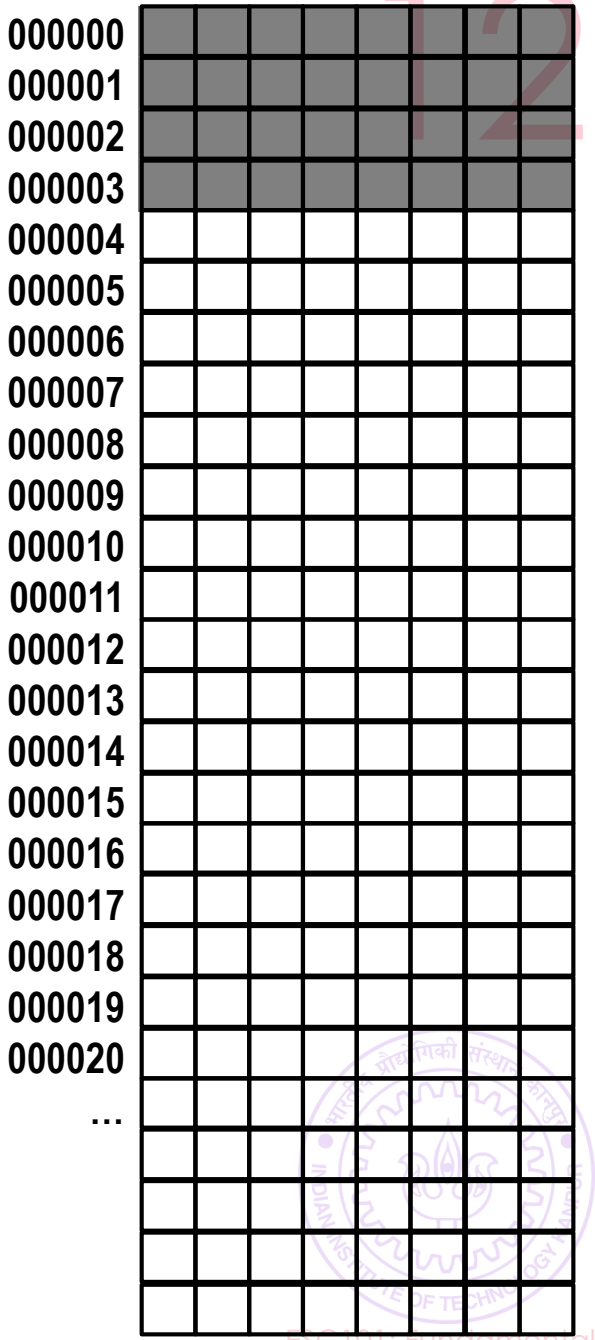
How Mr C stores variables

He has a very long chain of bytes

Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C



How Mr C stores variables

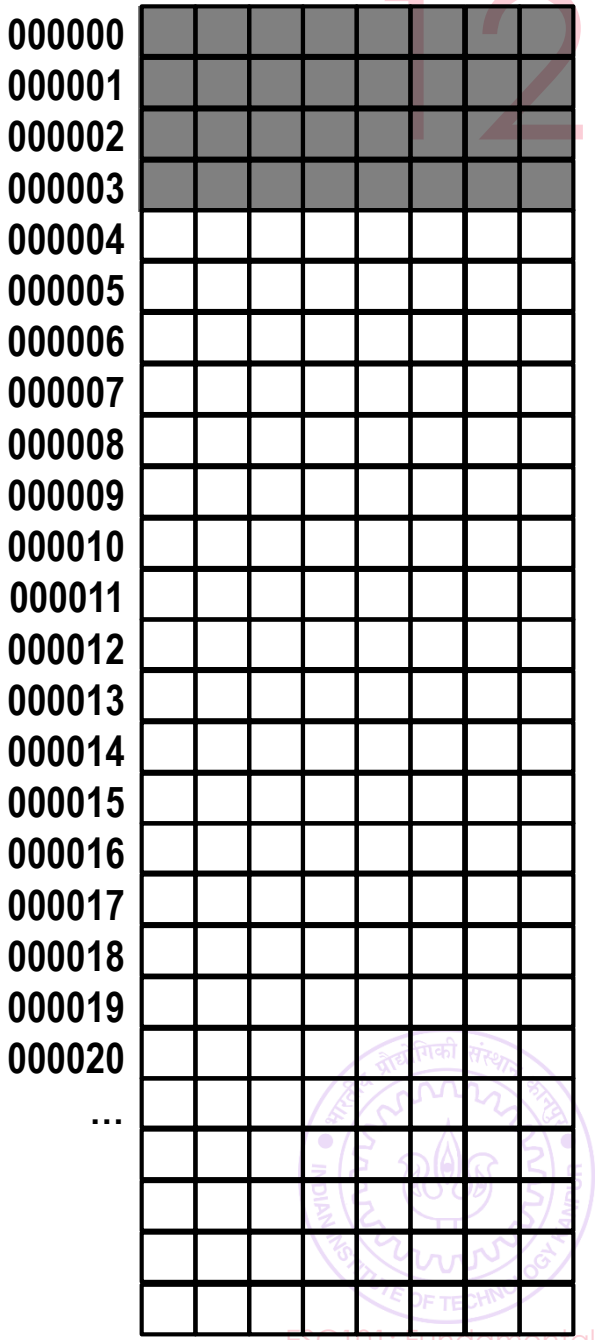
He has a very long chain of bytes

Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables



How Mr C stores variables

He has a very long chain of bytes

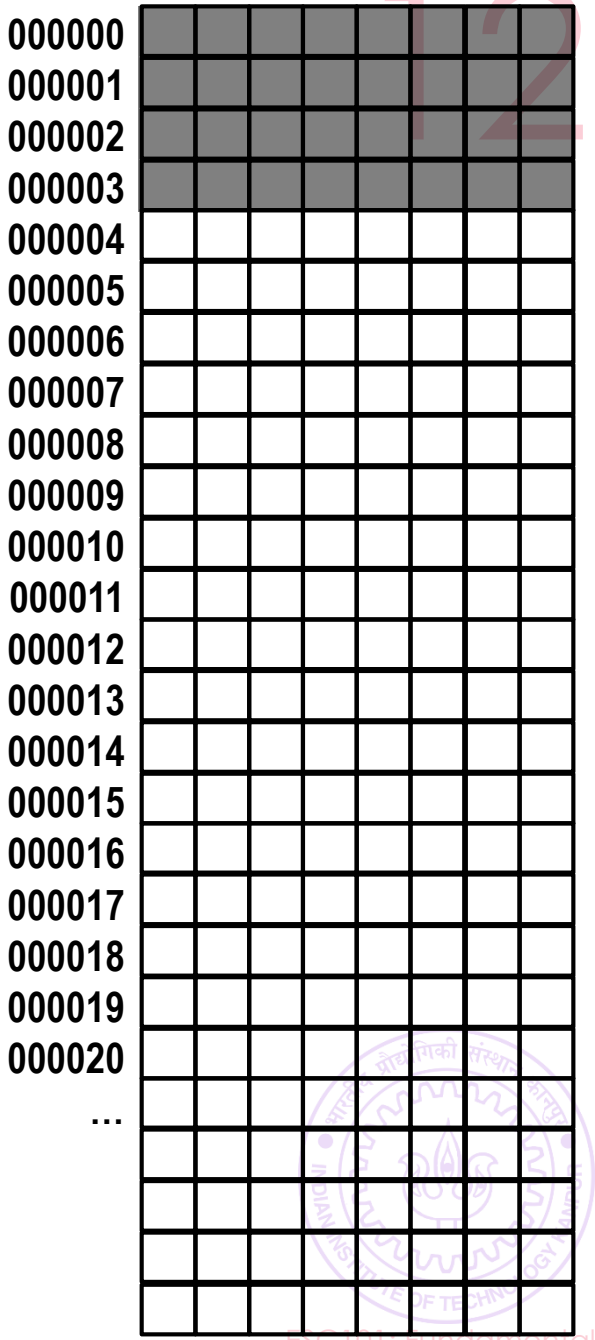
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
```



How Mr C stores variables

He has a very long chain of bytes

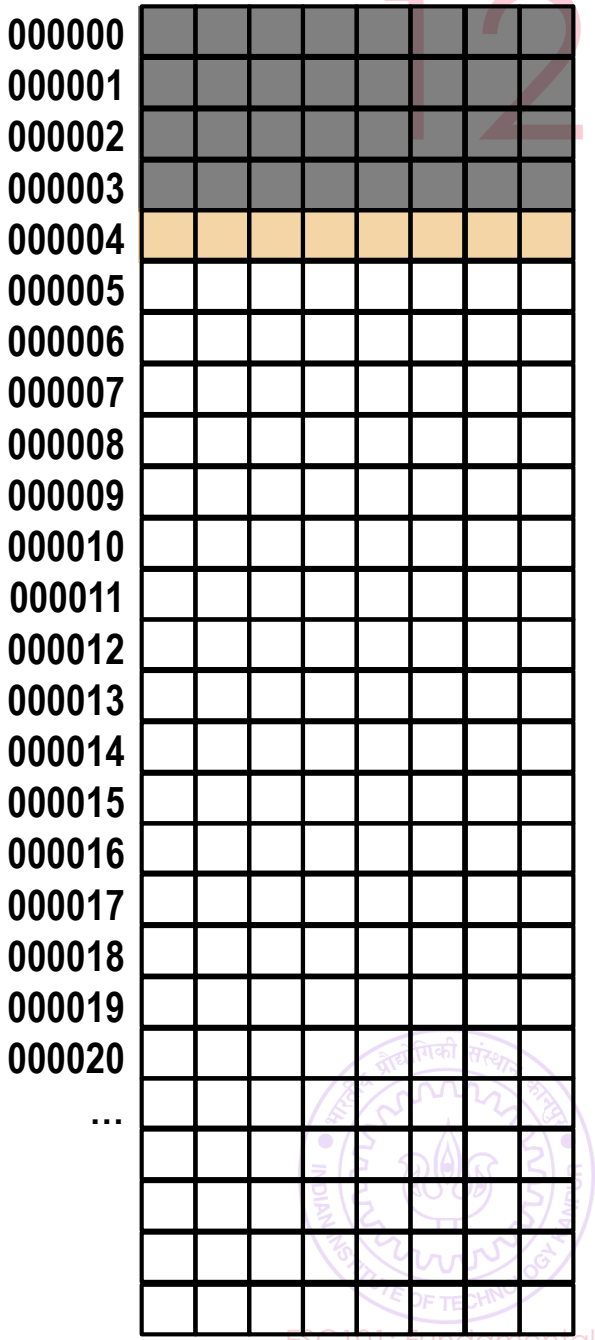
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
```



How Mr C stores variables

He has a very long chain of bytes

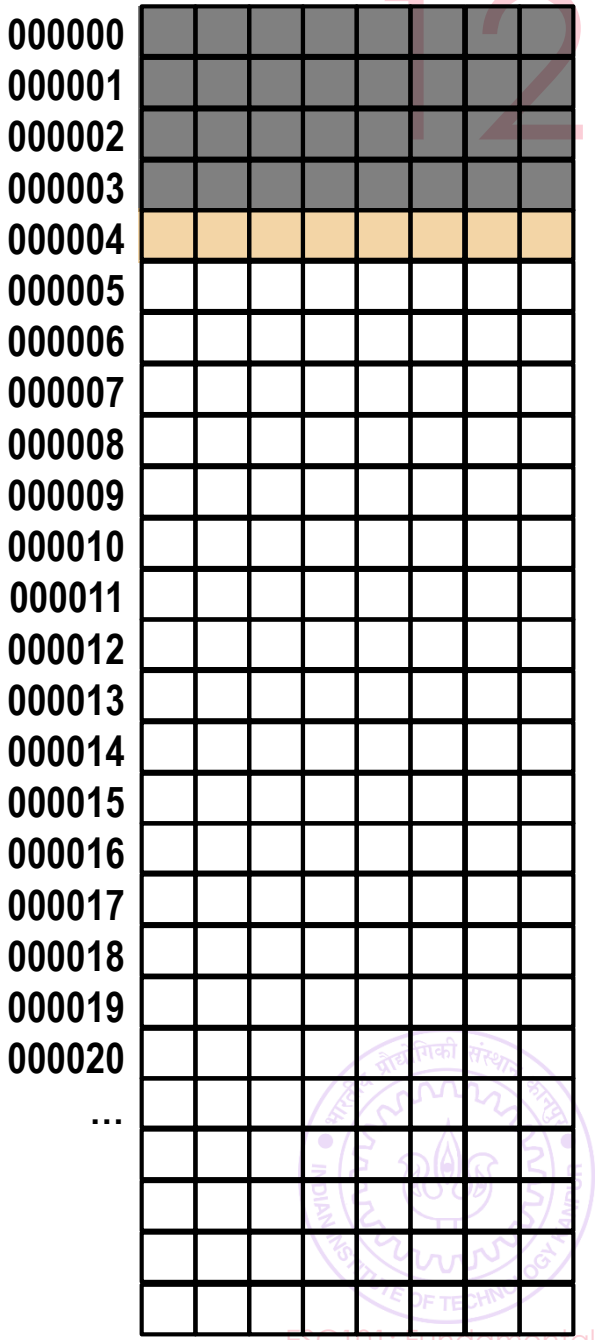
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
int a;
```



How Mr C stores variables

He has a very long chain of bytes

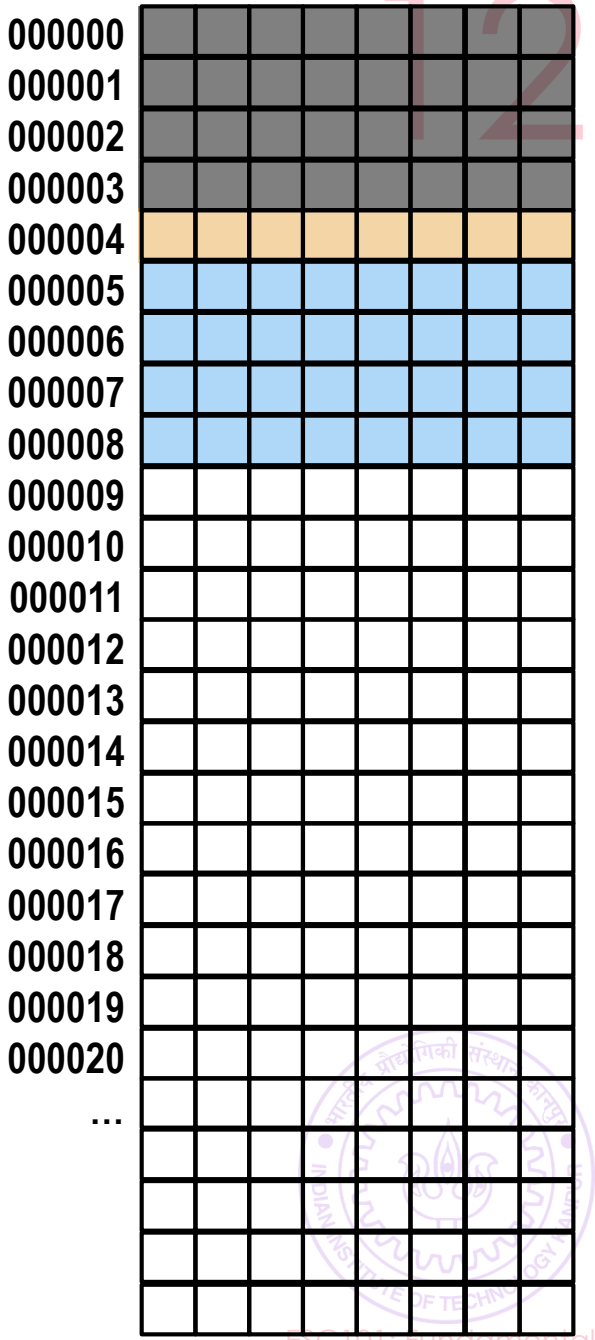
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
int a;
```



How Mr C stores variables

He has a very long chain of bytes

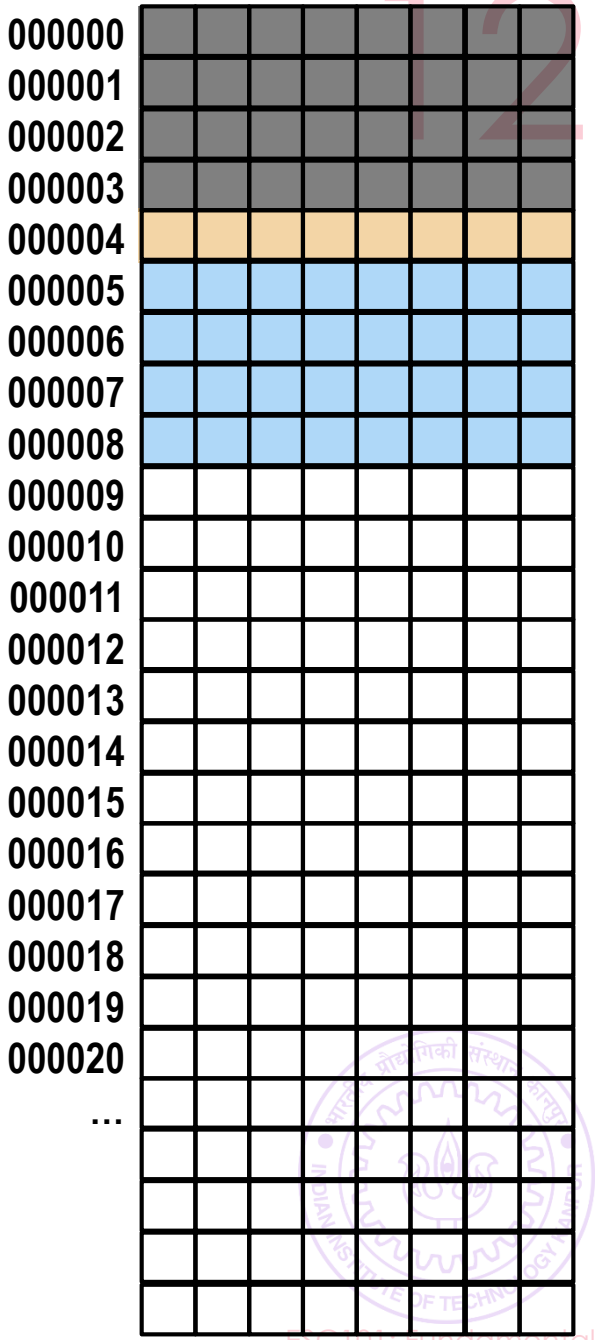
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
int a;
double d;
```



How Mr C stores variables

He has a very long chain of bytes

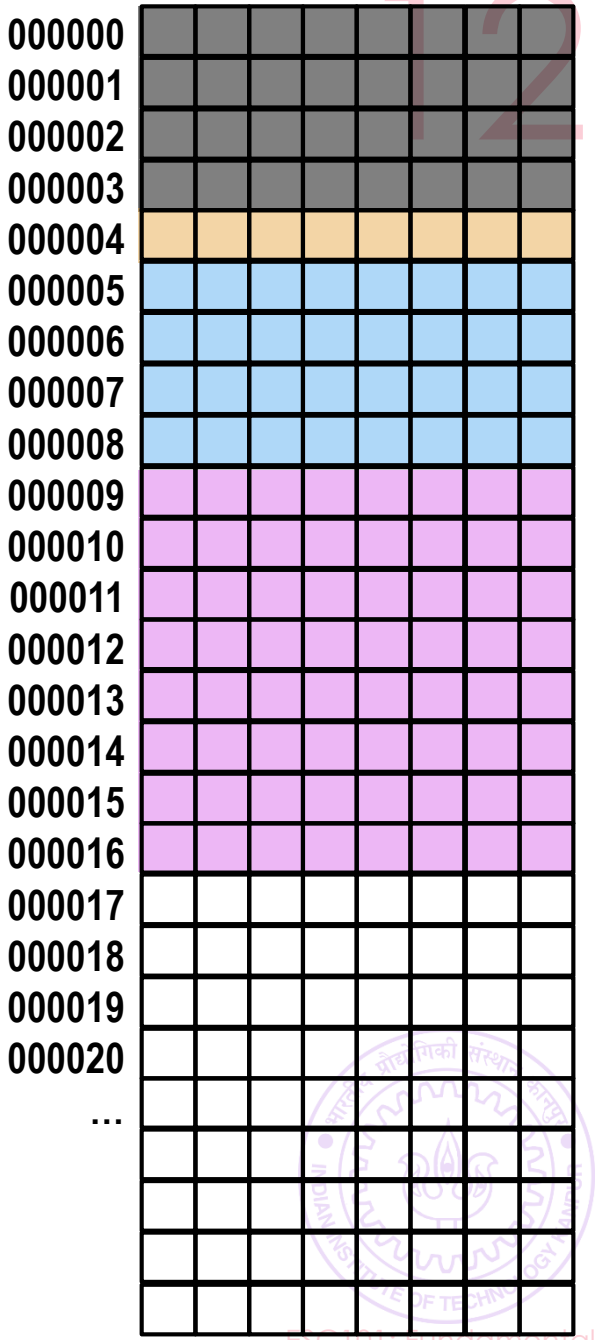
Each byte has an "address“

All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
int a;
double d;
```



How Mr C stores variables

He has a very long chain of bytes

Each byte has an "address“

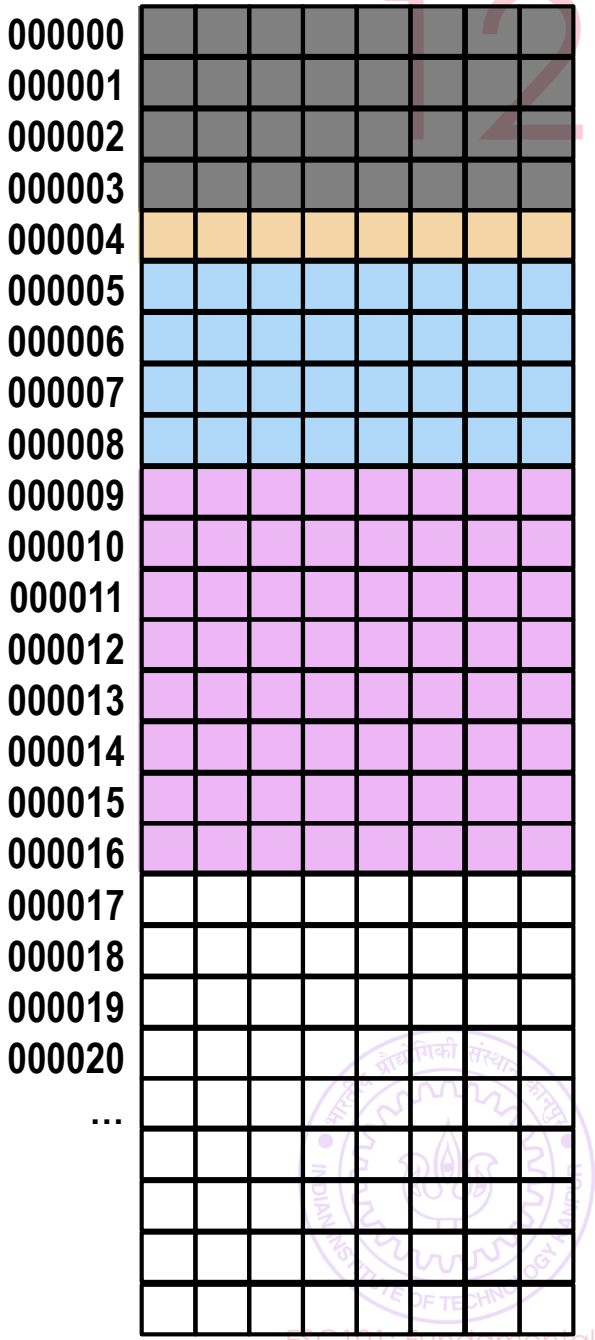
All addresses can be stored within 8 bytes

Some addresses are reserved for Mr C

Others can be used by you for variables

```
char c;
int a;
double d;
```

So c is stored at address 000004, a at 000005 and d at address 000009



Pointers

भारतीय प्रौद्योगिकी संस्थान कानपुर
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Pointers

Don’t let anyone scare you – pointers are just a way to store these addresses

Pointers

Don't let anyone scare you – pointers are just a way to store these addresses

Each pointer is a collection of 8 bytes (same size as long) that is storing one of these internal addresses

Pointers

Don't let anyone scare you – pointers are just a way to store these addresses

Each pointer is a collection of 8 bytes (same size as long) that is storing one of these internal addresses

Be careful not to confuse these internal addresses with array indices. Array indices are what **you** use to write nice code. These addresses are used by Mr C to manage stuff

Pointers

Don't let anyone scare you – pointers are just a way to store these addresses

Each pointer is a collection of 8 bytes (same size as long) that is storing one of these internal addresses

Be careful not to confuse these internal addresses with array indices. Array indices are what **you** use to write nice code. These addresses are used by Mr C to manage stuff

In some sense Mr C manages a ridiculously huge array!

Pointers

Don't let anyone scare you – pointers are just a way to store these addresses

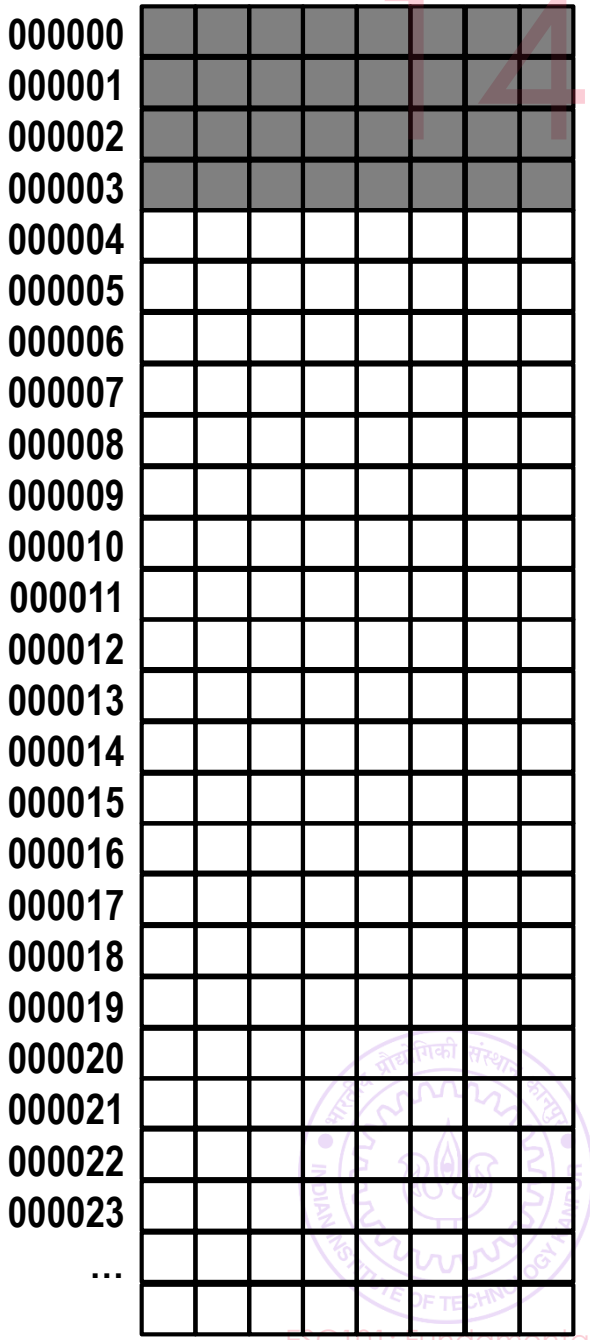
Each pointer is a collection of 8 bytes (same size as long) that is storing one of these internal addresses

Be careful not to confuse these internal addresses with array indices. Array indices are what **you** use to write nice code. These addresses are used by Mr C to manage stuff

In some sense Mr C manages a ridiculously huge array!

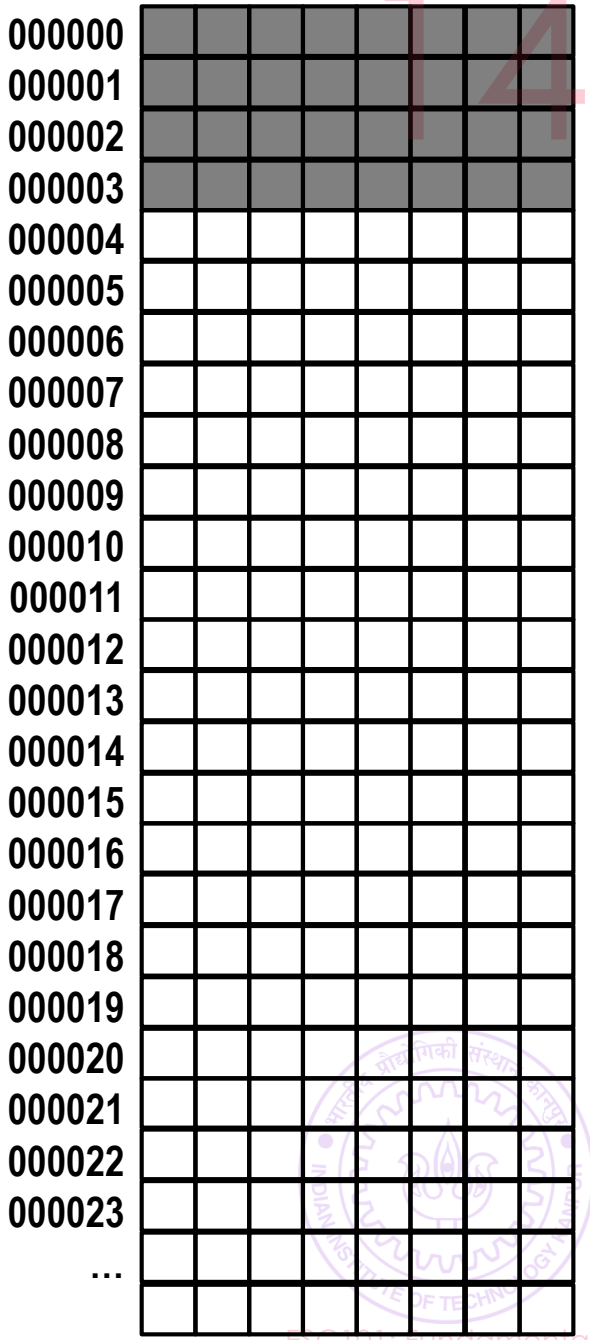
Pointers can allow us to write very beautiful code but it is a very powerful tool – misuse it and you may suffer ☺

How Mr C stores arrays



How Mr C stores arrays

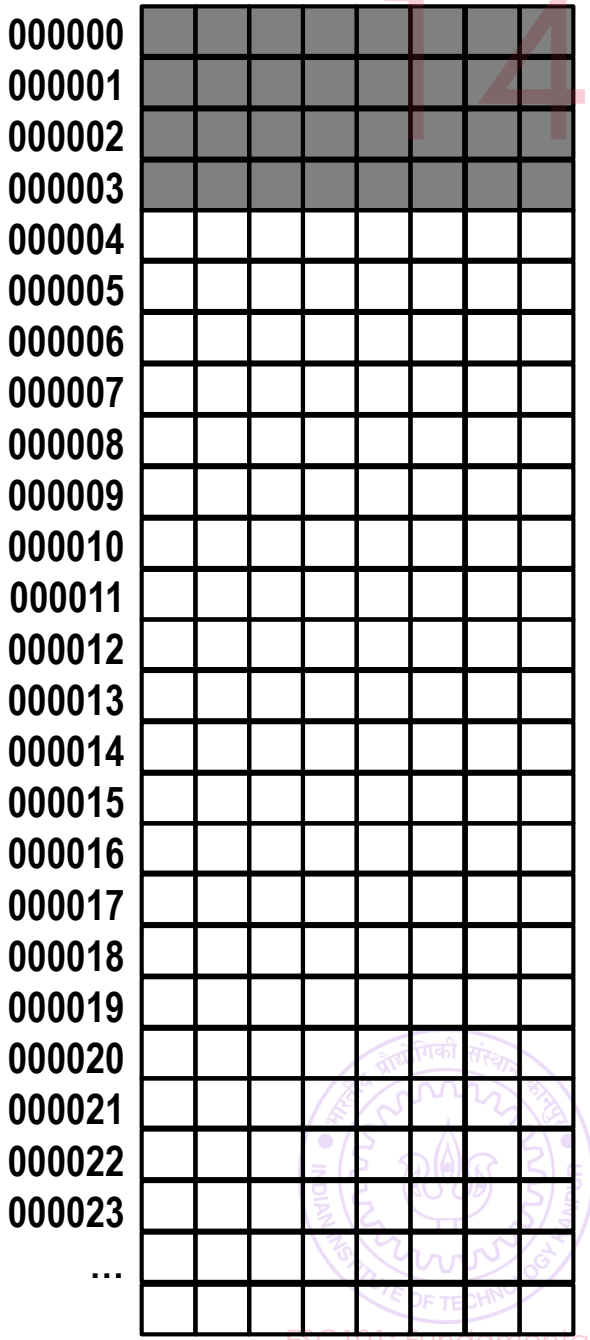
If we declare an array, a sequence of addresses get allocated



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

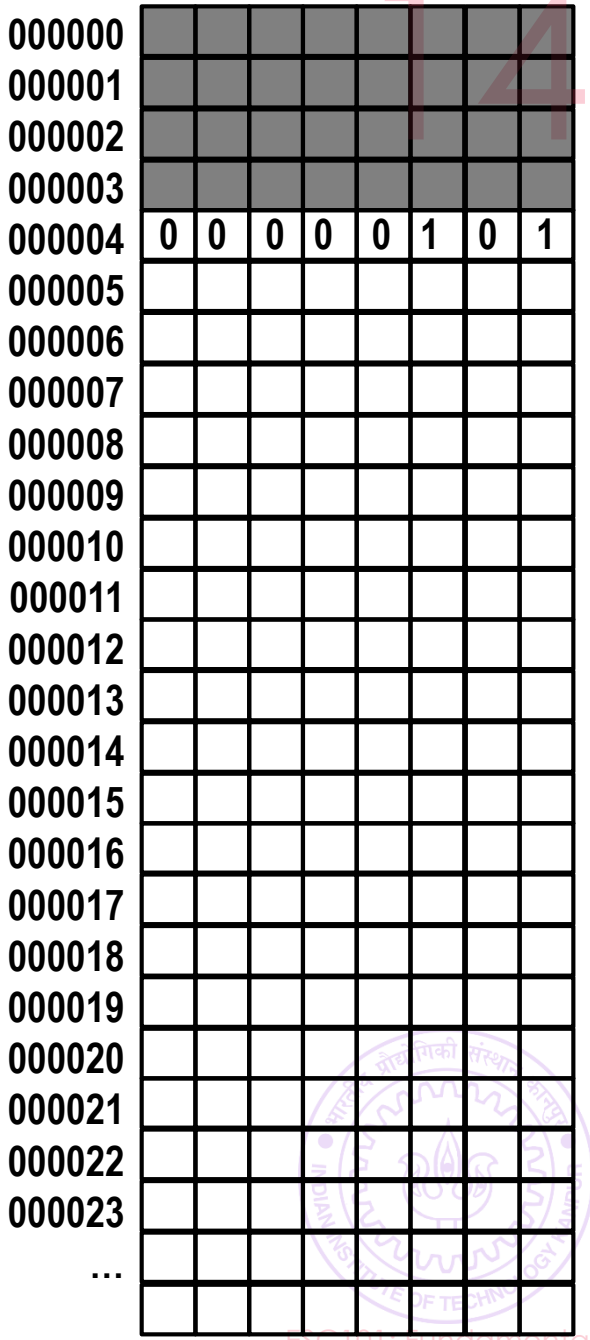
```
char c[5];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

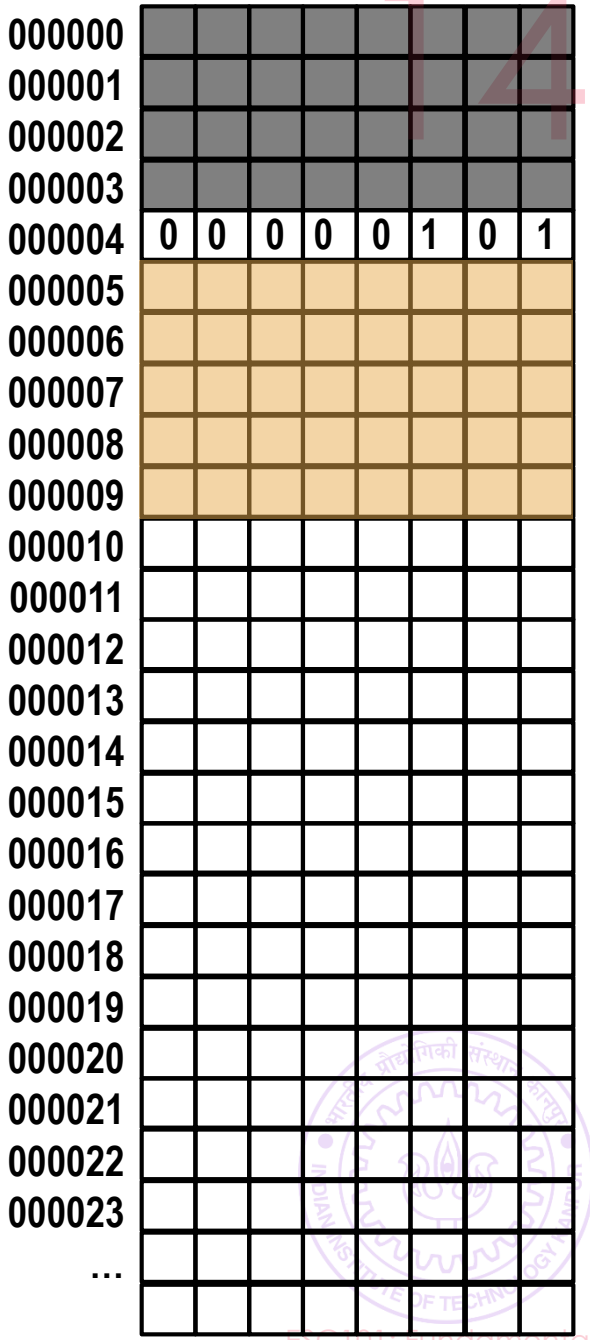
```
char c[5];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

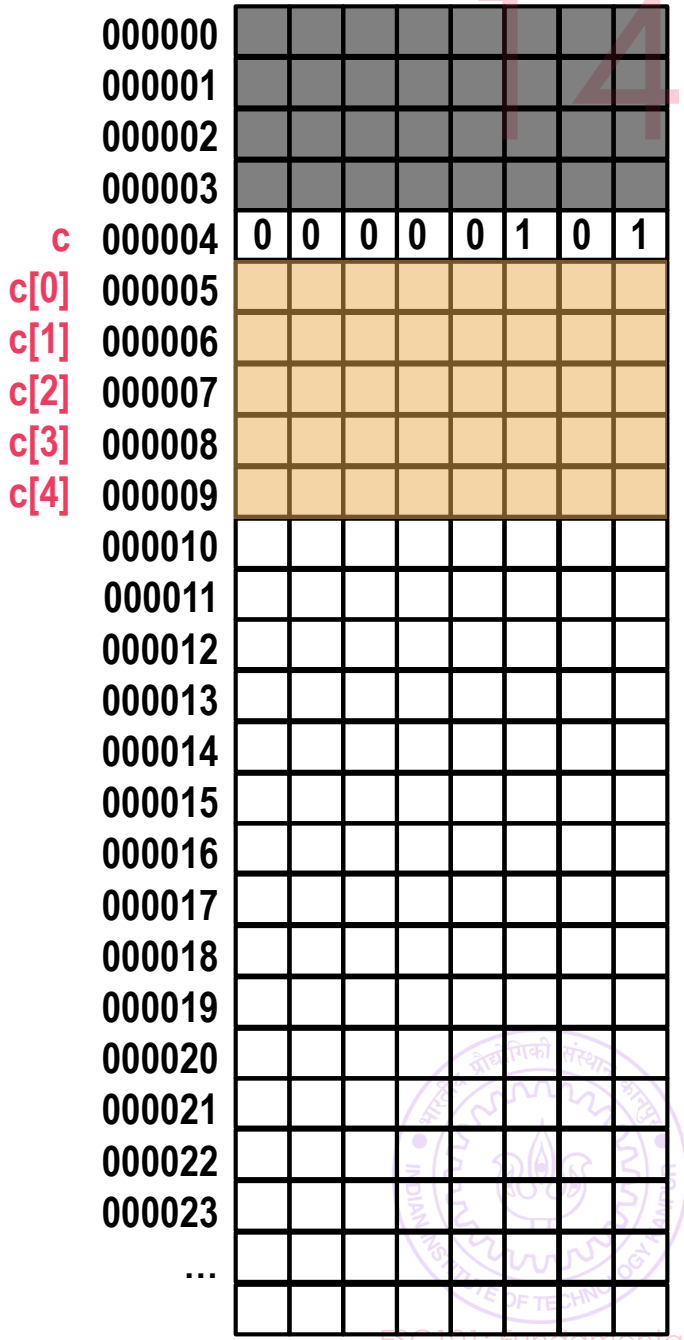
```
char c[5];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

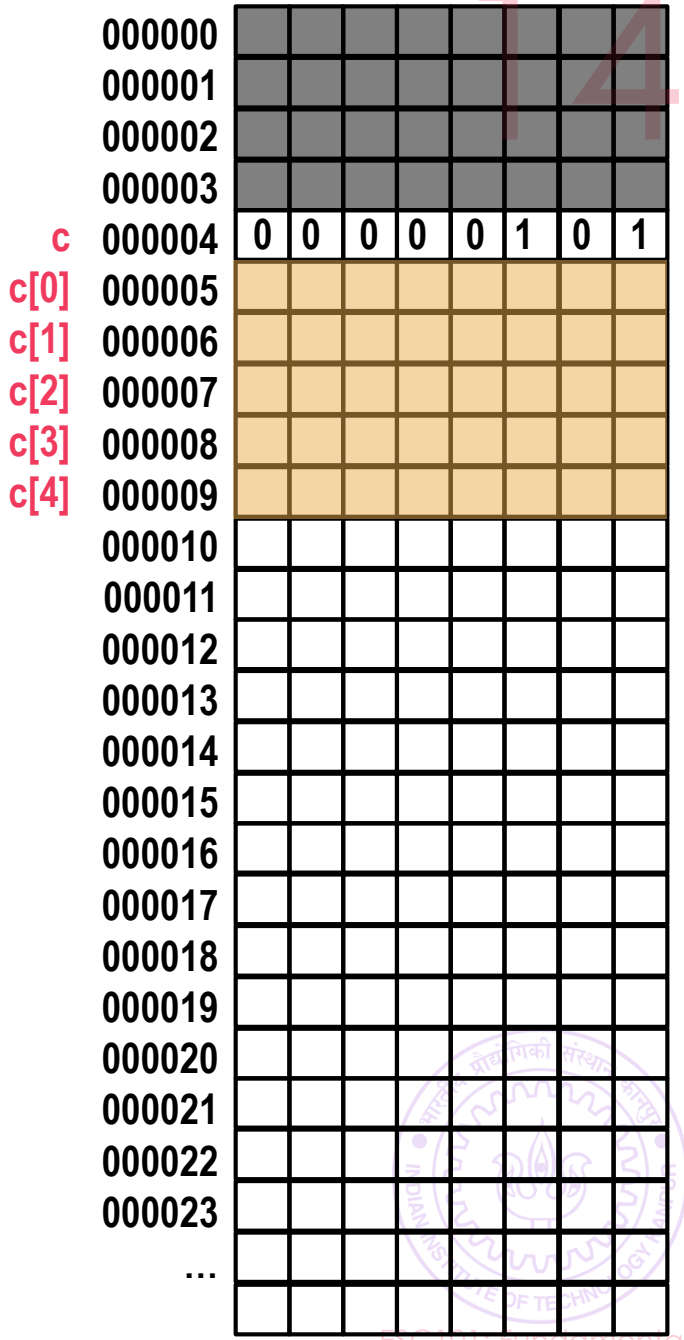
```
char c[5];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

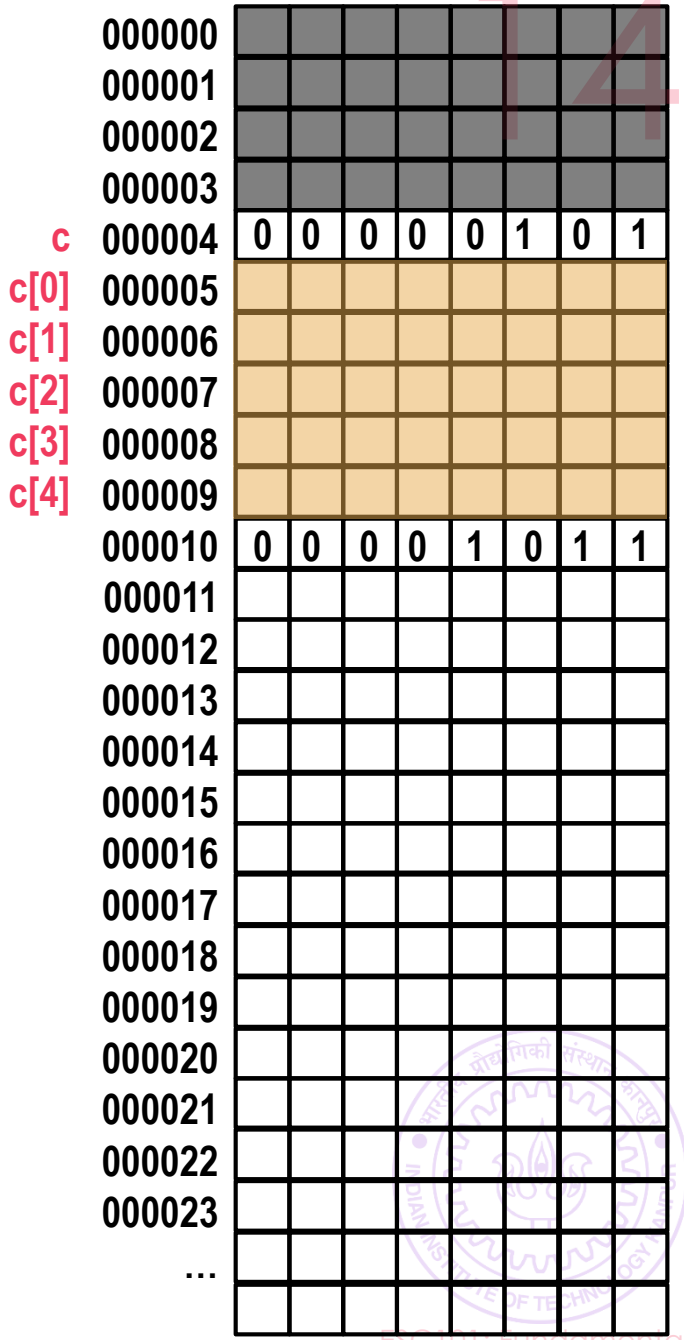
```
char c[5];
int a[3];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

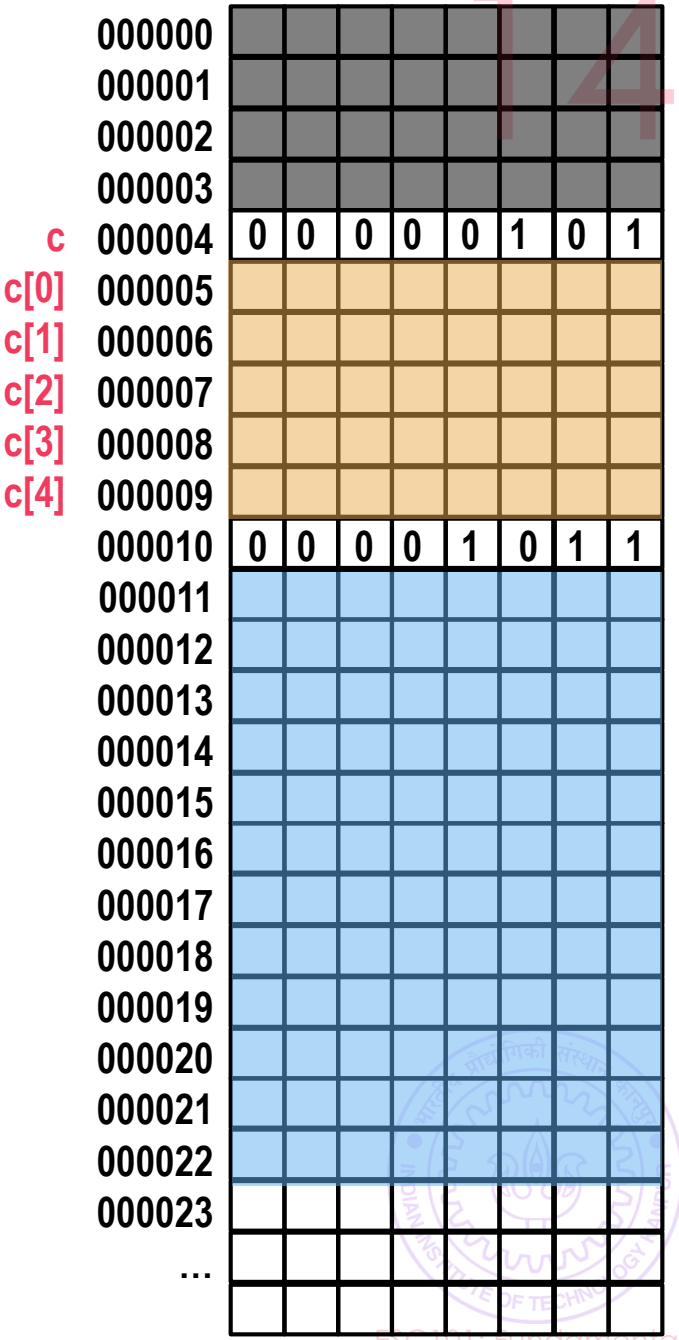
```
char c[5];
int a[3];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

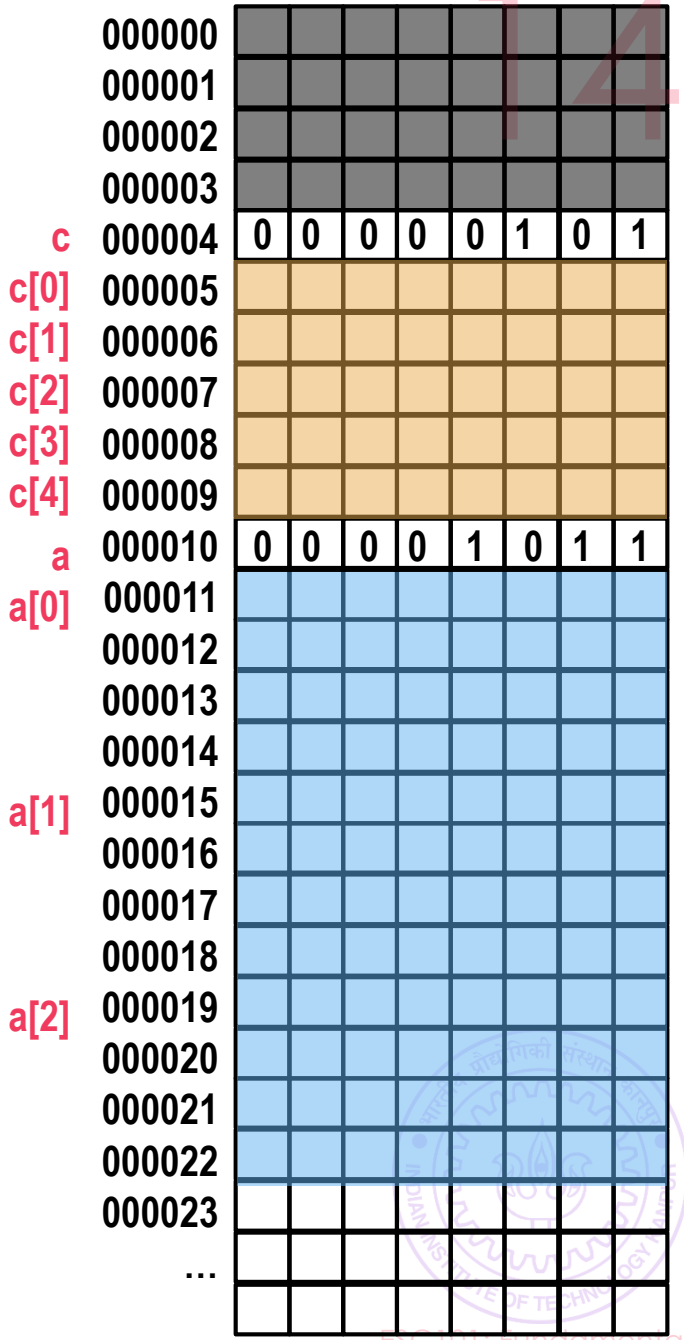
```
char c[5];
int a[3];
```



How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

```
char c[5];
int a[3];
```

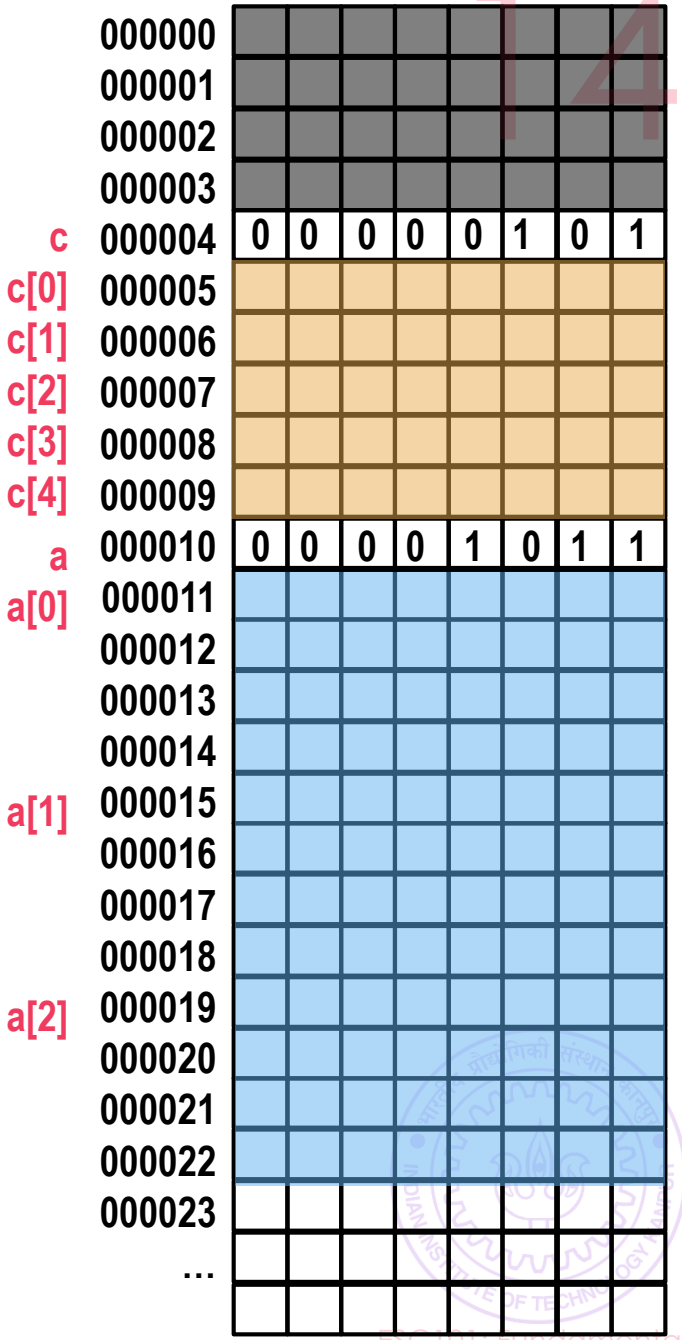


How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]



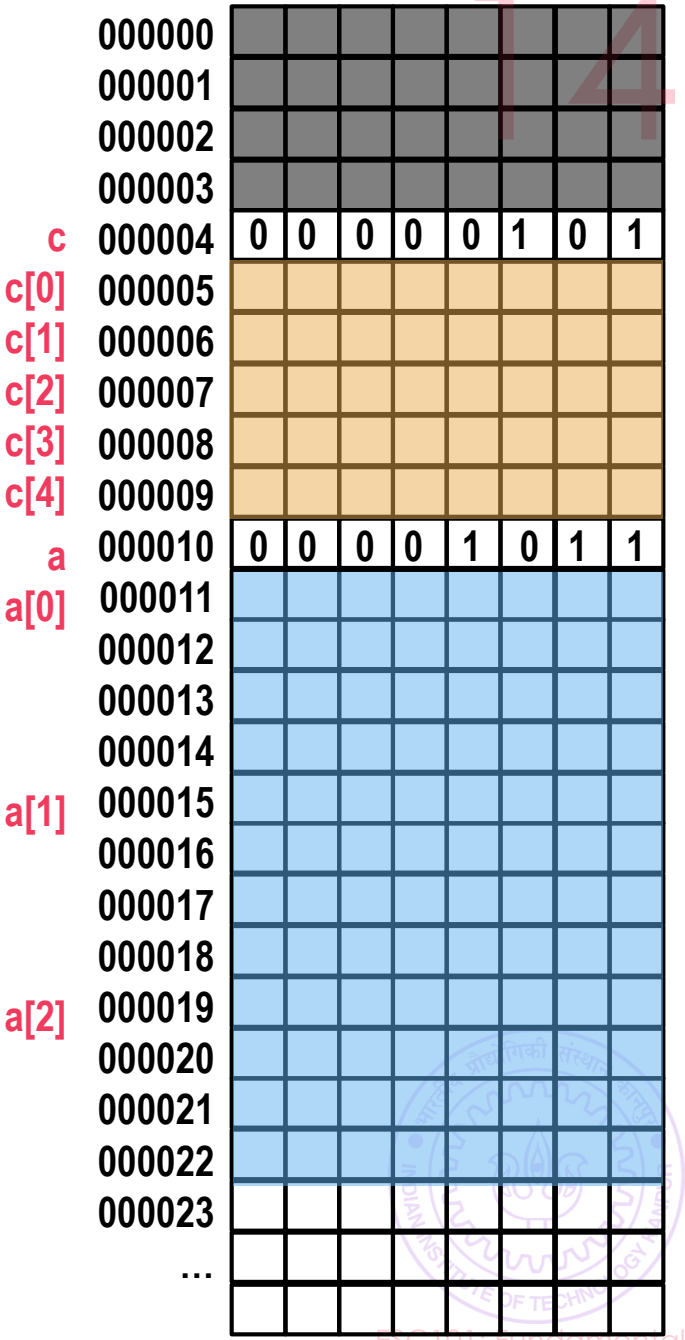
How Mr C stores arrays

If we declare an array, a sequence of addresses get allocated

```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on



How Mr C stores arrays

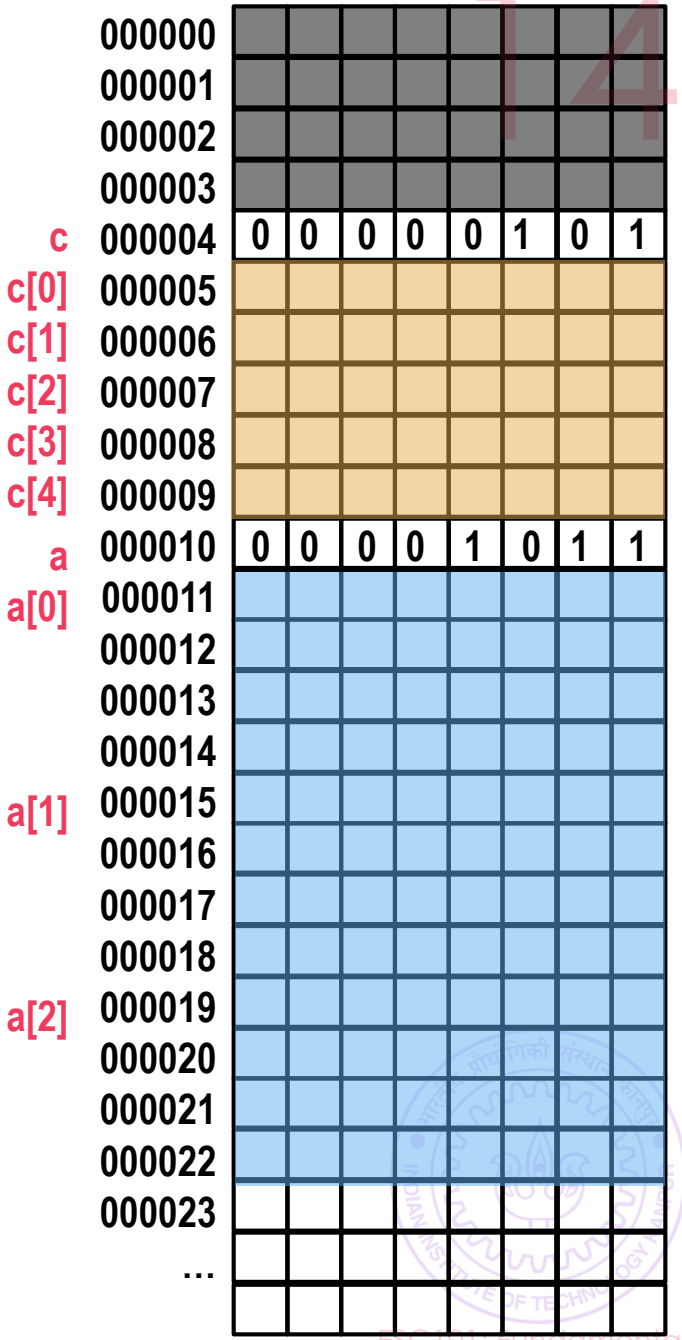
If we declare an array, a sequence of addresses get allocated

```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on

a[0] is stored at address 000011, a[1] at address 000015 (int takes 4 bytes), a[2] at address 000019, and so on



Mr C stores arrays

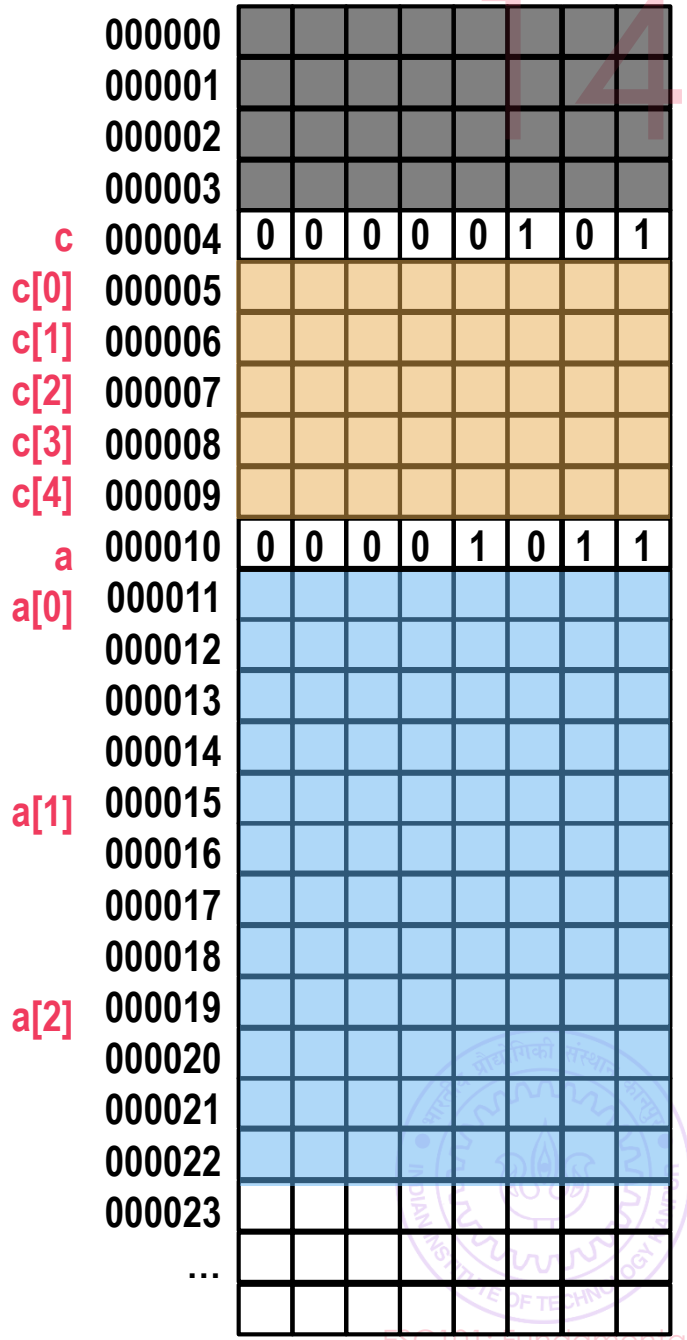
If we declare an array, a sequence of addresses get allocated

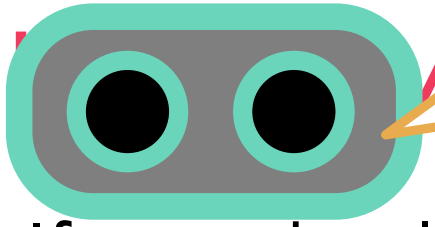
```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on

a[0] is stored at address 000011, a[1] at address 000015 (int takes 4 bytes), a[2] at address 000019, and so on





Actually, being pointers, c and a should themselves take 8 bytes to store the addresses

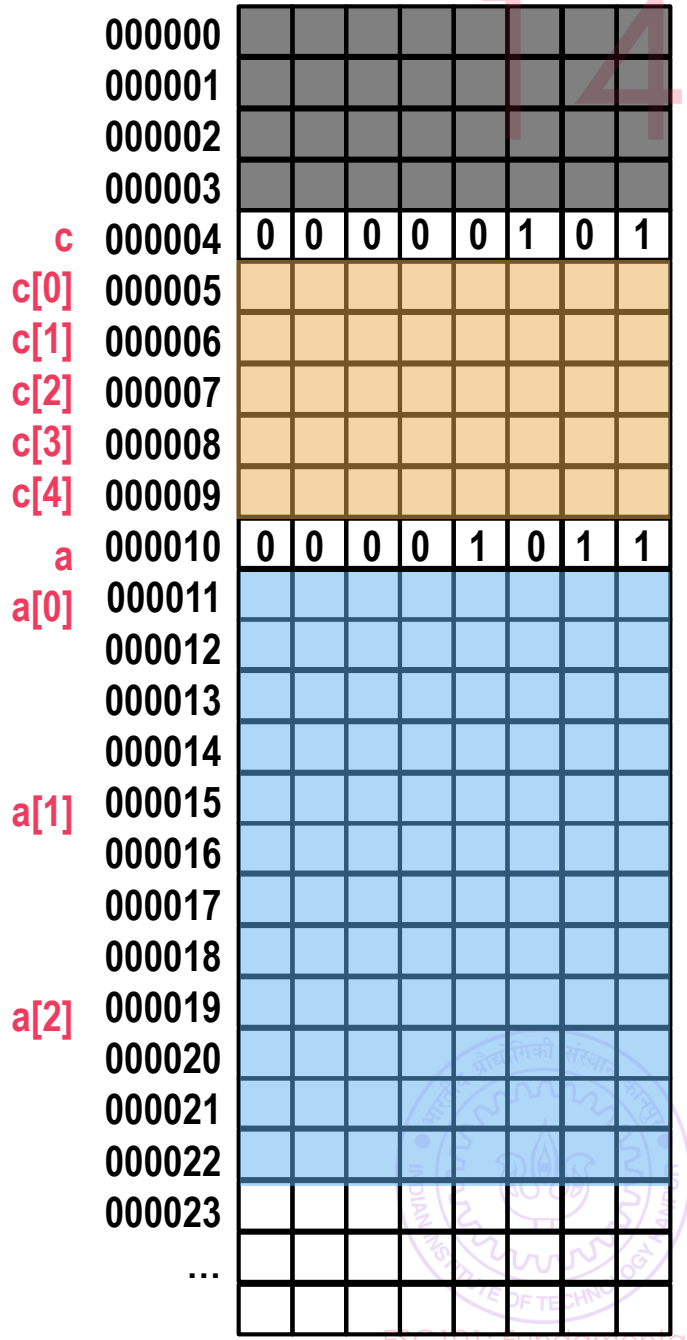
If we declare an array, a sequence of addresses get allocated

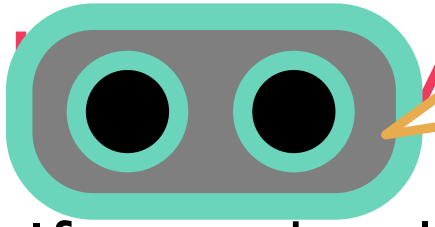
```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on

a[0] is stored at address 000011, a[1] at address 000015 (int takes 4 bytes), a[2] at address 000019, and so on





Actually, being pointers, c and a should themselves take 8 bytes to store the addresses

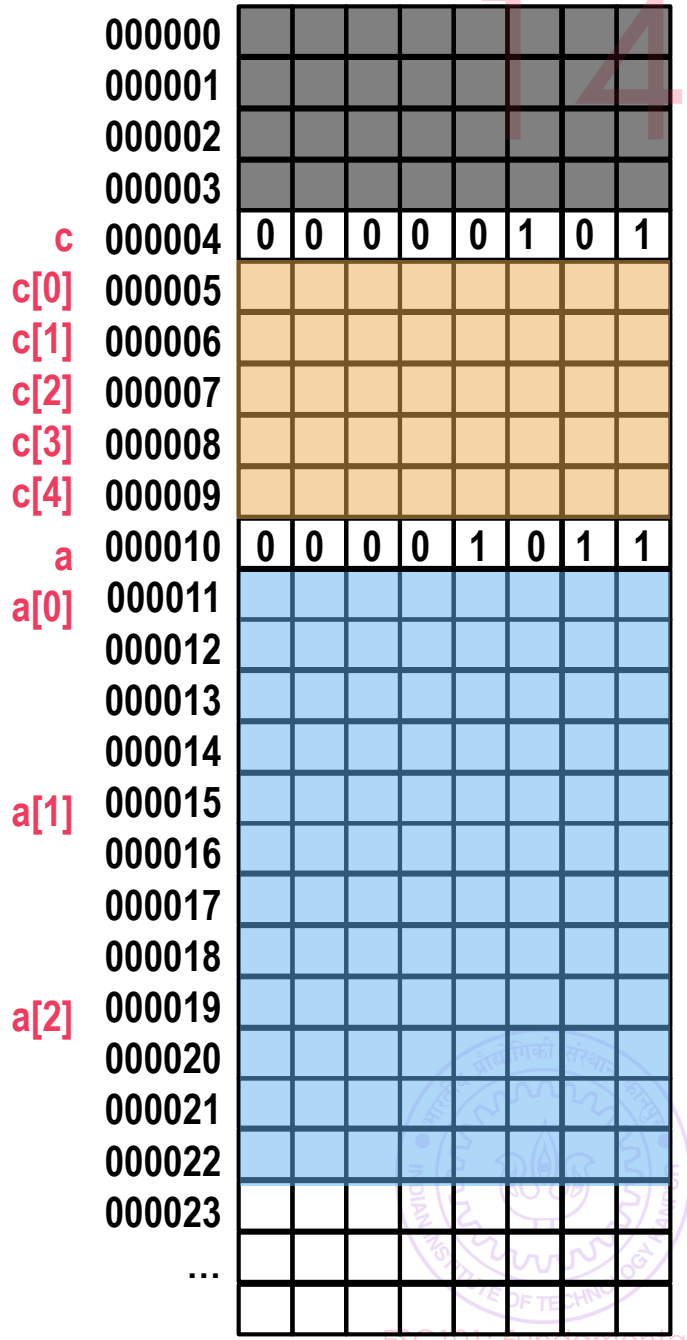
If we declare an array, a sequence of addresses get allocated

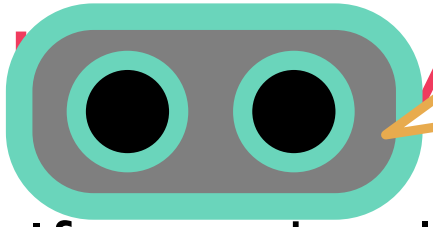
```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on

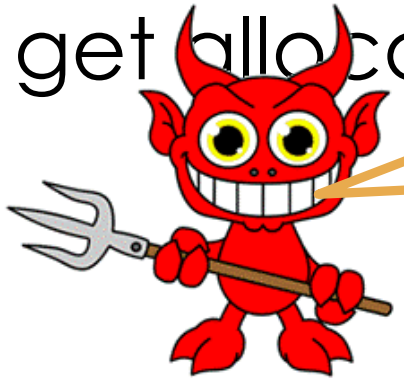
a[0] is stored at address 000011, a[1] at address 000015 (int takes 4 bytes), a[2] at address 000019, and so on





Actually, being pointers, c and a should themselves take 8 bytes to store the addresses

If we declare an array ... of addresses get allocate...



Yes, but I ran out of space space as well as ran out of patience drawing boxes

```
char c[5];
int a[3];
```

c and a are actually pointers, c stores the address of c[0], a stores address of a[0]

c[0] is stored at address 000005, c[1] at address 000006, c[2] at 000007 and so on

a[0] is stored at address 000011, a[1] at address 000015 (int takes 4 bytes), a[2] at address 000019, and so on