Embedded Systems - Prelims

UNIT – I

Prelims

Shriram K Vasudevan

What is a computer system?

Specification

compute the fibonacci sequence

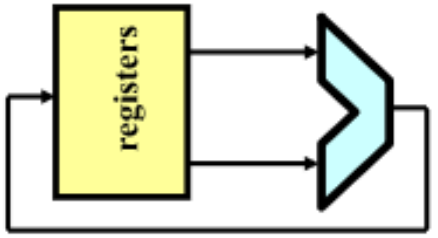
Program

```
for(i=2; i<100; i++) {
  a[i] = a[i-1]+a[i-2];}
```

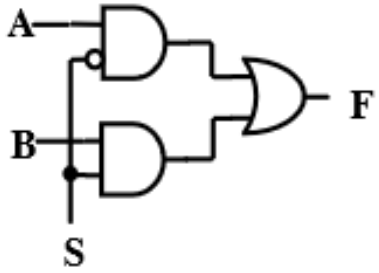
ISA (Instruction Set Architecture)

```
load r1, a[i];
add  r2, r2, r1;
```

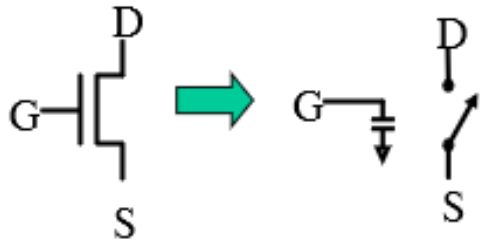
Microarchitecture



Logic



Transistors



Physics/Chemistry

Organization Vs. Architecture

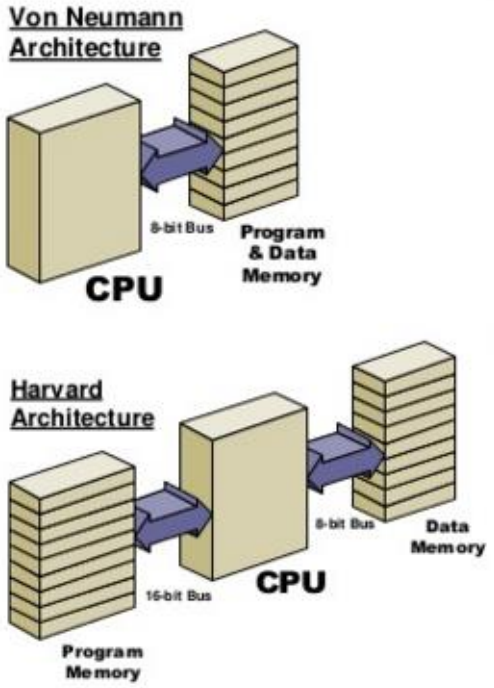
- Architecture refers to how various components in a computer are organized & the design aspect of a computer.
- Computer Organization is how features are implemented, typically hidden from the programmer. E.g. control signals, interfaces, memory technology etc.
- It is basically concerned with the way the hardware components operate and the way they are connected together to form the computer system.

What is an instruction?

- ADD A, B
- Here we tell the microprocessor to add the contents of two registers A and B. That is it! This forms the baseline!
- It is like you giving an order to your subordinate! It has to be obeyed!

Von Neumann Vs. Harvard Architecture.

Harvard Architecture vs Von Neumann Architecture

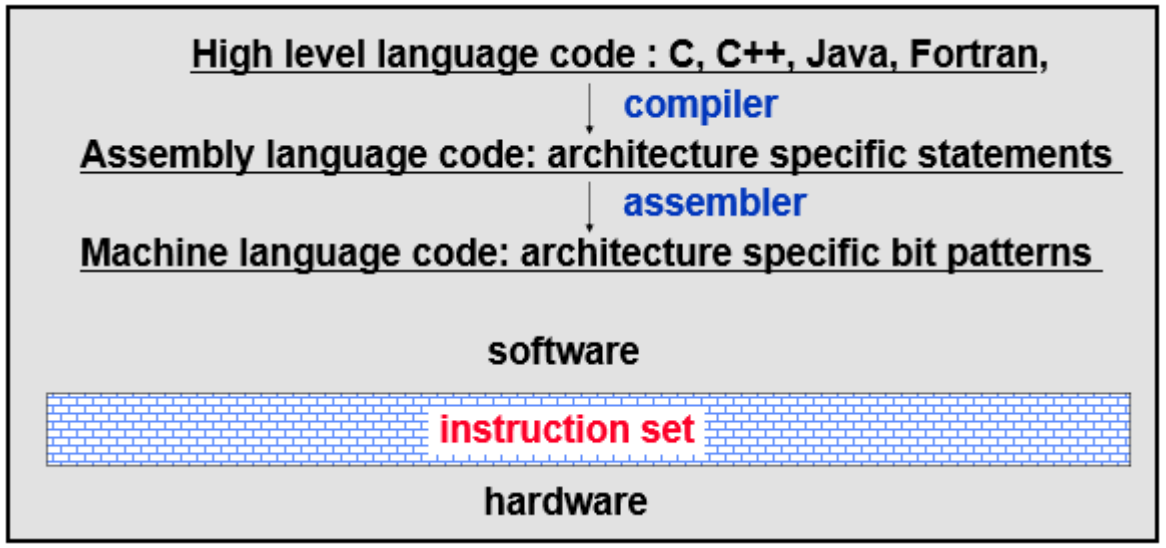


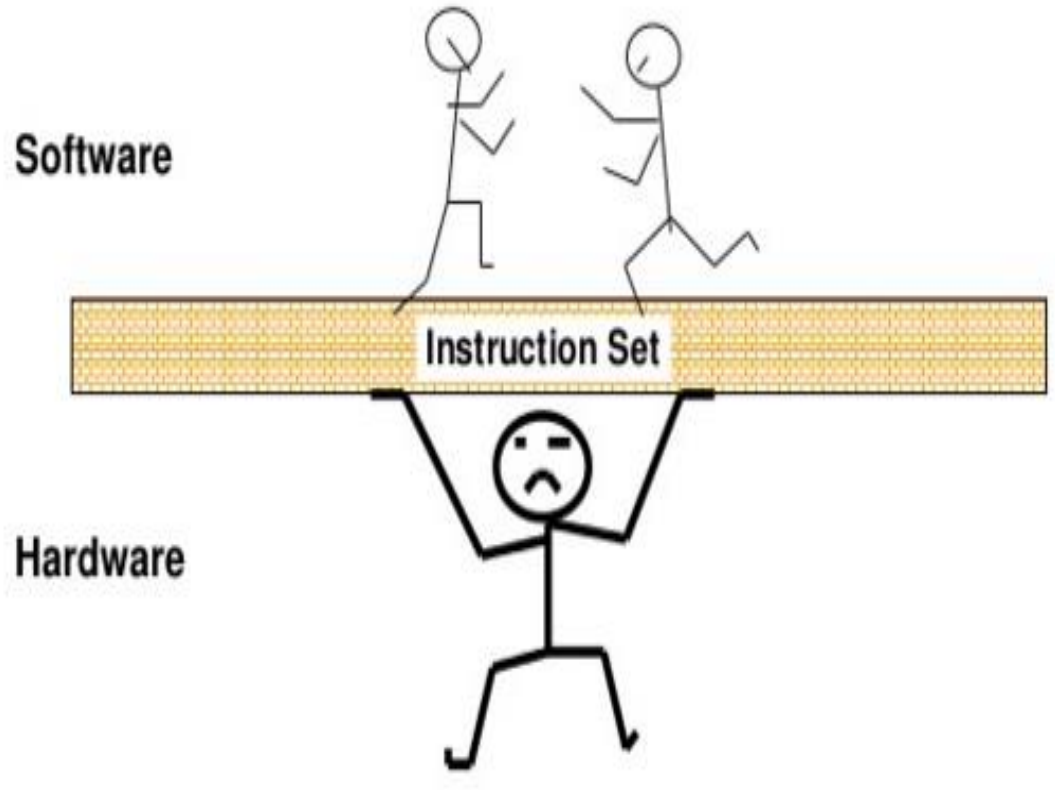
- Von Neumann Architecture:
  - Used single memory space for program and data.
  - Limits operating bandwidth
- Harvard Architecture:
  - Uses two separate memory spaces for program instructions and data
  - Improved operating bandwidth
  - Allows for different bus ~~widths~~

Instruction Set Architecture

•Serves as an interface between software and hardware.

•Provides a mechanism by which the software tells the hardware what should be done (i.e. instructs the hardware to do the right thing)





Contd.,

A good interface:

- **Lasts through many implementations (portability, compatibility)**
- **Is used in many different ways (generality)**
- **Provides convenient functionality to higher levels**
- **Permits an efficient implementation at lower levels**

What do instructions provide?

- Which operation to perform – add r0, r1, r3 // operation code
- Where to find the operand or operands – add r0, r1, r3 // source operands
- Place to store result – add r0, r1, r3 // destination
- Location of next instruction – add r0, r1, r3

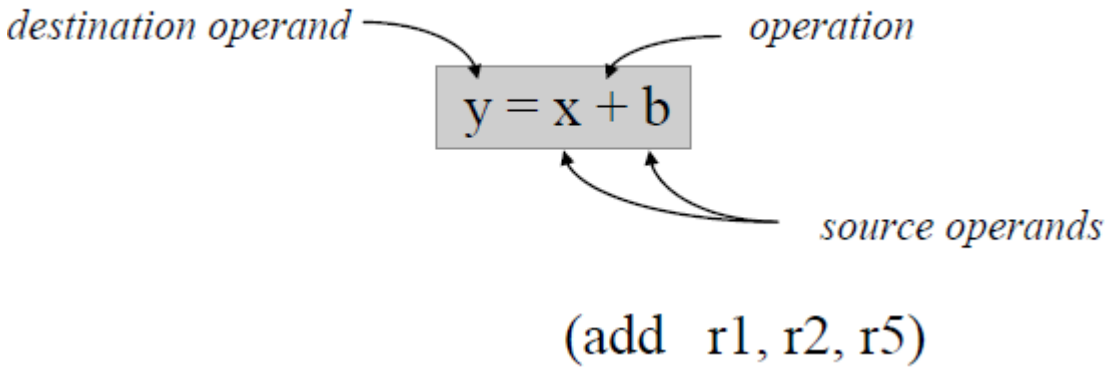
  br endloop

What is instruction?

- It is nothing but a command that can make the microprocessor perform a specific function.
- All the instructions are collectively grouped as instruction set.
- An Instruction is basically a combination of two parts one is operation code and the next is operands.
- **Operation code is also called as opcode**.
- Operands can be one or two based on the instruction.
- The opcode will be used to identify on what kind of operation needs to be performed like, addition or subtraction or data movement and so on. And operands will be used to spot the source and destination. An operand can be any of the following:
    1. **Registers (like Accumulator, B register and so on)**
    2. **Immediate value (Say 10H, 010101B)**
    3. **An address, i.e. memory location. (Say 30H)**
    4. **Any of the supported input or output ports.**

Common design issues

- Where are operands stored?
  - registers, memory, stack, accumulator
- How many explicit operands are there?
  - 0, 1, 2, or 3
- How is the operand location specified?
  - register, immediate, indirect, . . .
- What type & size of operands are supported?
  - byte, int, float, double, string, vector. . .
- What operations are supported?
  - add, sub, mul, move, compare . .



Byte Ordering

- Byte ordering, or *endianness*, is another major architectural consideration.
- If we have a two-byte integer, the integer may be stored so that the least significant byte is followed by the most significant byte or vice versa.
  - In *little endian* machines, the least significant byte is followed by the most significant byte.
  - *Big endian* machines store the most significant byte first (at the lower address).

Contd.,

Address	Example Address	Value
Base + 0	1001	..
Base + 1	1002	..
Base + 2	1003	..
Base + ..	..	..

- Ex. Represent the String

APPLE

		Address				
		Base + 0	Base + 1	Base + 2	Base + 3	Base + 4
Byte-Order	Little Endian	E	L	P	P	A
	Big Endian	A	P	P	L	E

Address ⟶	00	01	10	11
Big Endian	12	34	56	78
Little Endian	78	56	34	12

Contd.,

- Why is endianness so important?
  - Suppose you are storing int values to a file, then you send the file to a machine which uses the opposite endianness and read in the value.
  - You'll run into problems because of endianness. You'll read in reversed values that won't make sense.
- Endianness is also a big issue when sending numbers over the network.
  - Again, if you send a value from a machine of one endianness to a machine of the opposite endianness, you'll have problems.
  - This is even worse over the network, because you might not be able to determine the endianness of the machine that sent you the data.

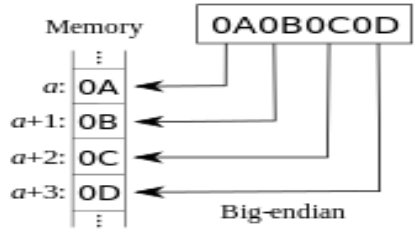
Contd.,

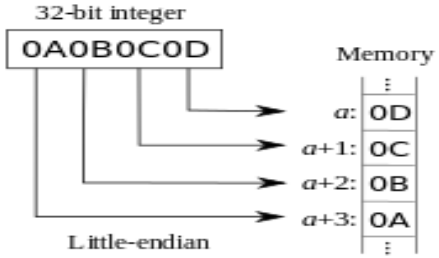
- Where does this term "endian" come from? Jonathan Swift was a satirist (he poked fun at society through his writings).
- His most famous book is "Gulliver's Travels", and he talks about how certain people prefer to eat their hard boiled eggs from the little end first (thus, little endian), while others prefer to eat from the big end (thus, big endians) and how this lead to various wars.

Contd.,

- Endianness only makes sense when you want to break a large value (such as a word) into several small ones.
  - You must decide on an order to place it in memory. However, if you have a 32 bit register storing a 32 bit value, it makes no sense to talk about endianness.
  - The register is *neither* big endian *nor* little endian. It's just a register holding a 32 bit value.
  - The rightmost bit is the least significant bit, and the leftmost bit is the most significant bit.

Contd.,





- Big endian:
  - Is more natural.
  - The sign of the number can be determined by looking at the byte at address offset 0.
  - Strings and integers are stored in the same order.
- Little endian:
  - Conversion from a 16-bit integer address to a 32-bit integer address does not require any arithmetic.

Contd.,

- Examples of **Little Endian**
  - BMP
  - RTF
  - MSPaint
- Examples of **Big Endian**
  - JPEG
  - Adobe Photoshop
  - MacPaint

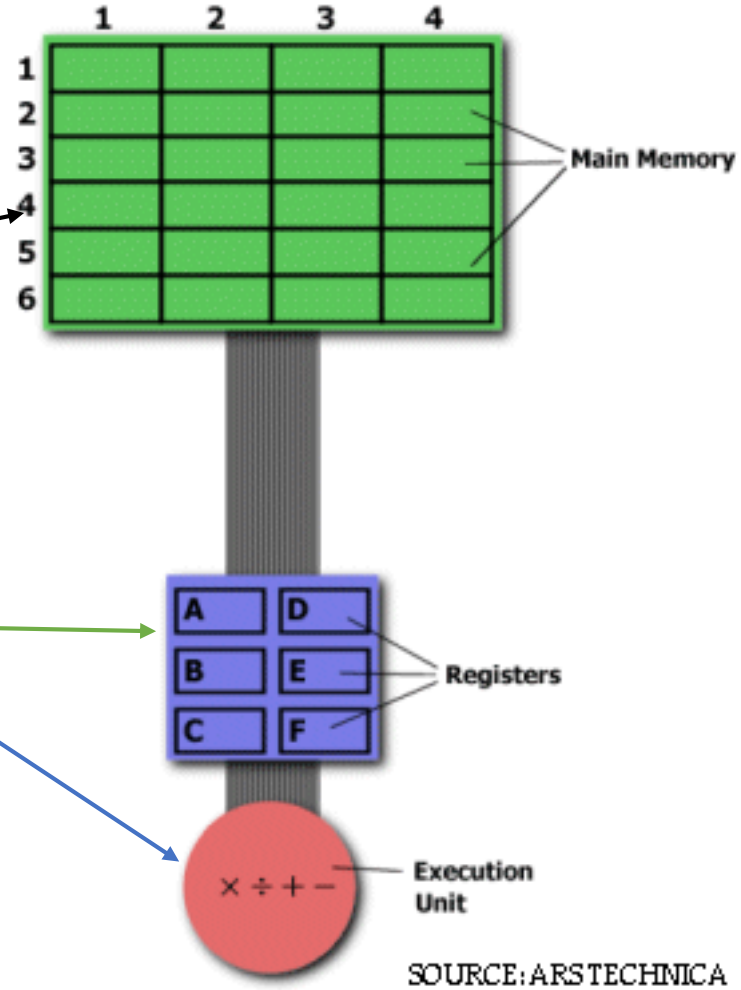
CISC vs. RISC – A Very important interview question.

- Which is better?
  - Well, a big debate which can't be answered so easily!
  - Because, both are in existence and has not vanished completely.

- Major Computer manufacturing firms Apple and Intel have always been arguing on importance of hardware and software in CPU architecture designs.
- Intel supporters want the hardware to bear more responsibility and software on the easier side. This would impact the hardware designing to be more complex but software coding would be relatively easy.
- On the other hand, Apple supporters want the hardware to be simple and easy and software to take the major role.
- ***Intel's hardware oriented approach is termed as Complex Instruction Set Computer*** while that of ***Apple is Reduced Instruction Set Computer***

Contd.,
An Example..
Multiplying two numbers in memory

- On the right is a diagram representing the storage scheme for a generic computer.
- The main memory is divided into locations numbered from (row) 1: (column) 1 to (row) 6: (column) 4.
- The execution unit is responsible for carrying out all computations.
- However, the execution unit can only operate on data that has been loaded into one of the six registers (A, B, C, D, E, or F).
- **Let's say we want to find the product of two numbers - one stored in location 2:3 and another stored in location 5:2 - and then store the product back in the location 2:3.**

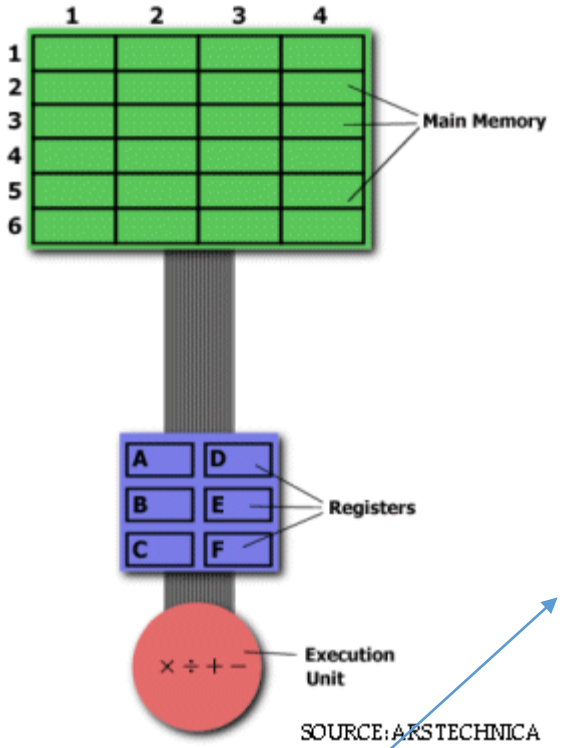


Contd.,
THE CISC APPROACH

- The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible.

- This is achieved by building processor hardware that is capable of understanding and executing a series of operations. For this particular task, a CISC processor would come prepared with a specific instruction **(we'll call it "MULT")**

- When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and **then stores the product in the appropriate register**. Thus, the entire task of multiplying two numbers can be completed with one instruction:



MULT 2:3, 5:2

- MULT is what is known as a "complex instruction."
- It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions.
- It closely resembles a command in a higher level language

Advantages:

- **Because the length of the code is relatively short, very little RAM is required to store instructions.**
- **The emphasis is put on building complex instructions directly into the hardware**

Contd.,
The RISC Approach

- RISC processors only use simple instructions that can be executed within one clock cycle.
- Thus, the "MULT" command described above could be divided into three separate commands:
  - **"LOAD," which moves data from the memory bank to a register**
  - **"PROD," which finds the product of two operands located within the registers**
  - **"STORE," which moves data from a register to the memory banks.**

```
LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, A
```

Is this less efficient?

- **More RAM is needed to store the assembly level instructions.**
- **The compiler must also perform more work to convert a high-level language statement into code of this form.**

Not actually, there are benefits!
These RISC "reduced instructions" require less transistors of hardware space than the complex instructions, leaving more room for general purpose registers.

Because all of the instructions execute in a uniform amount of time (i.e. one clock), pipelining is possible

Summarize…

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

Agenda

- Decimal systems
- Binary systems
- Octal systems
- Hexadecimal systems
- BCD
- Gray code

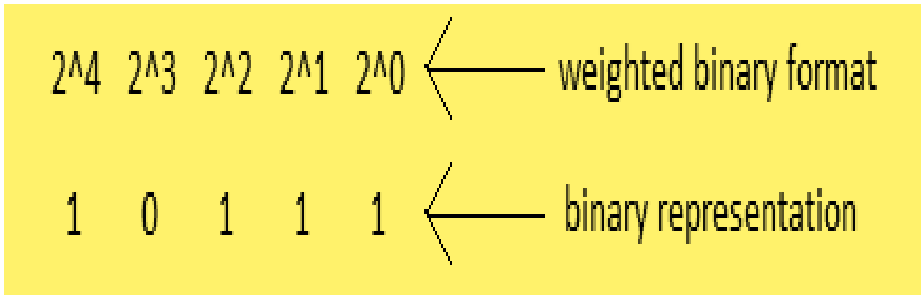
Possible conversions

NUMBER LIST	CONVERSIONS
1	Decimal to Binary
2	Binary to Decimal
3	Decimal to Octal
4	Octal to Decimal
5	Decimal to Hexadecimal
6	Hexadecimal to Decimal
7	Decimal to BCD
8	BCD to Decimal
9	Binary to Octal
10	Octal to Binary
11	Binary to Hexadecimal
12	Hexadecimal to Binary
13	Binary to BCD
14	BCD to Binary
15	Octal to Hexadecimal
16	Hexadecimal to Octal
17	Octal to BCD
18	BCD to Octal
19	Hexadecimal to BCD
20	BCD to Hexadecimal
21	BCD to Gray Code
22	**Successive multiplication (Not dealing this here)**

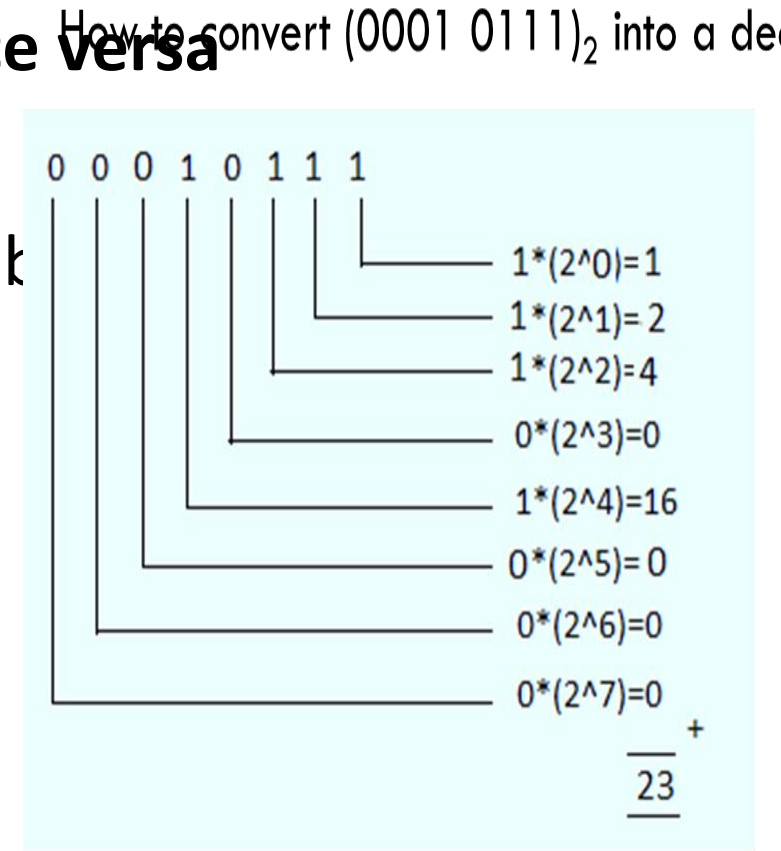
- You need not get tensed on seeing the list of 22 conversions. This is just to provide all possible combinations of conversions to the you.
- The **necessity to know code conversions** comes into picture when we work with systems which can accept inputs only in certain number systems.
- It becomes difficult to feed real time values into the system. Hence, it is immensely important to know code conversions to convert real time values into a number system which can be input into the system for further processing.

Conversions with respect to Decimal and Binary number systems

- **Decimal to Binary conversion and Vice versa**
- Let us take an example of $(23)_{10}$.
- How to convert this into a binary numb

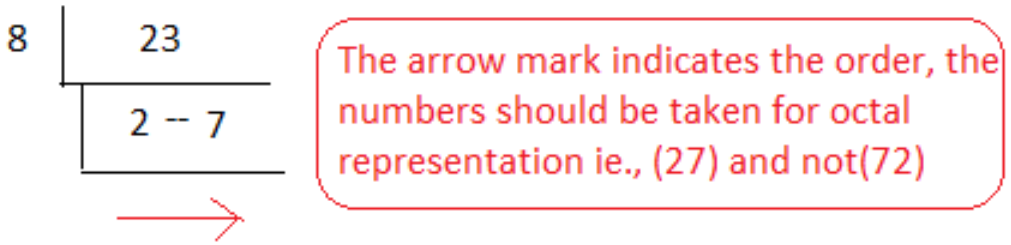


Hence, the binary representation of $(23)_{10}$ is $(0001\ 0111)_2$.



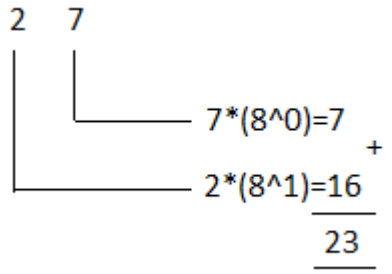
Conversions with respect to Decimal and Octal number systems

- **Decimal to Octal conversion**
- How to convert a decimal number say, $(23)_{10}$ into an octal number?
- Method: Do L-Division bv 8.



- **Octal to Decimal conversion**

How to convert (27)8 into a decimal number?

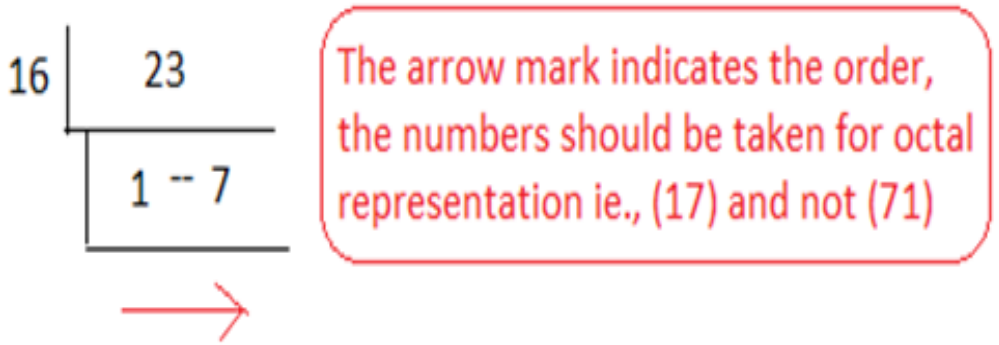


Hence, decimal representation of (27)8 is (23)10.

Conversions between Decimal and Hexadecimal number systems

Decimal to Hexadecimal conversion

How to convert $(23)_{10}$ into its corresponding hexadecimal equivalent?

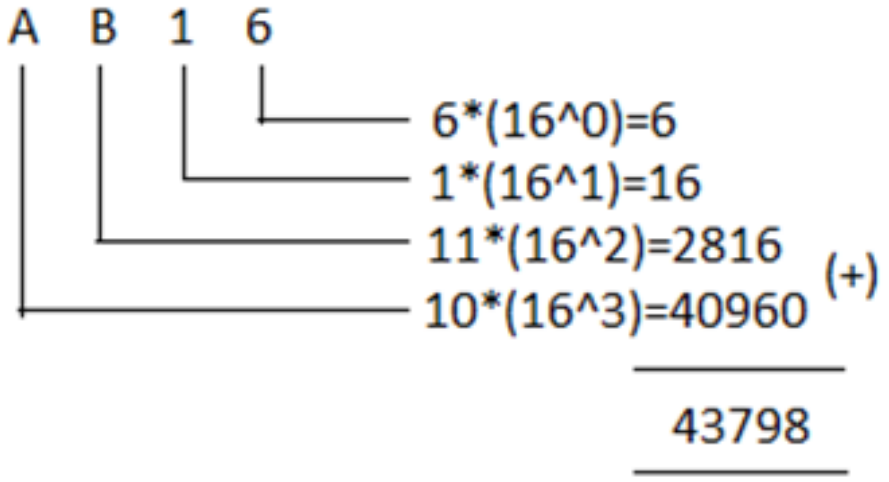


(For 23/16, quotient is 1 because 1*16=16 and remainder is 23-16=7).

Hence, hexadecimal representation of $(23)_{10}$ is $(17)_{16}$.

Hexadecimal to Decimal conversion

How to convert $(AB16)_{16}$ into its decimal equivalent?

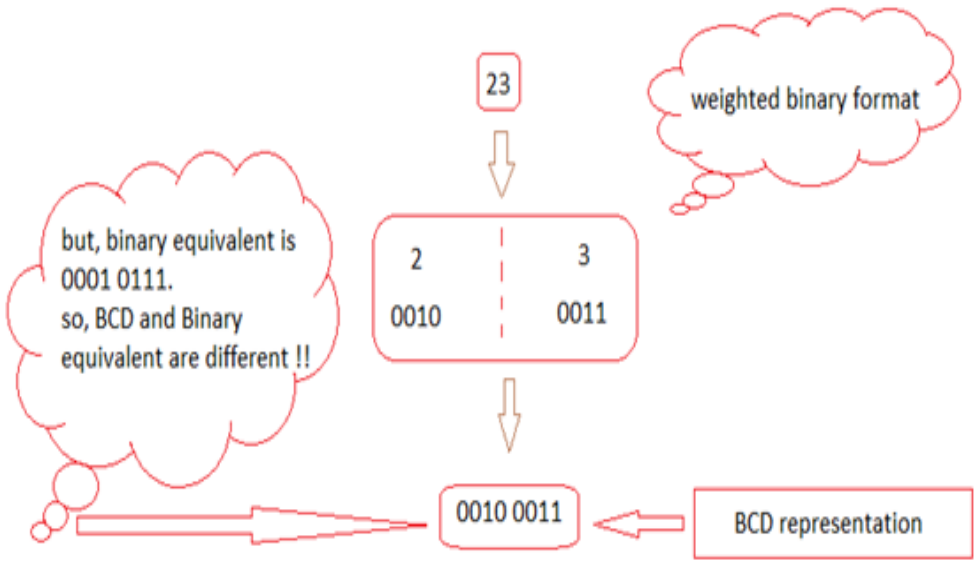


Hence, the corresponding decimal equivalent is $(43798)_{10}$.

Conversions between BCD and Decimal number systems

Decimal to BCD conversion

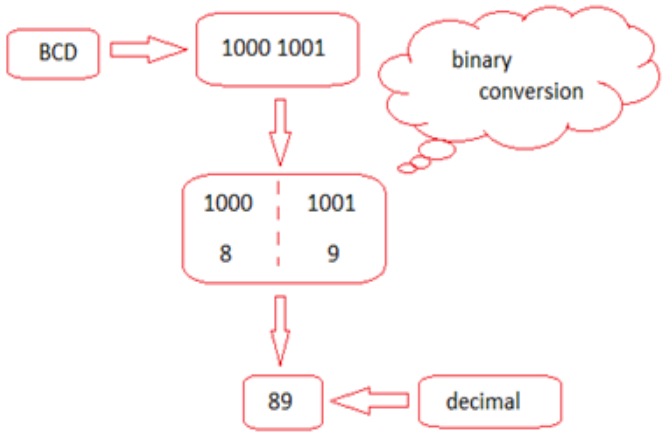
How to convert $(23)_{10}$ into its corresponding BCD equivalent?



Hence, the corresponding BCD equivalent is 0010 0011.

BCD to Decimal conversion

How to convert (1000 1001) into decimal form?



Hence, the decimal representation is $(89)_{10}$.

NOTE: When representing the individual numbers in binary form, we should represent in **4 bit only**. If 4 bits are not sufficient enough to represent the number, we can go for 8 bits.

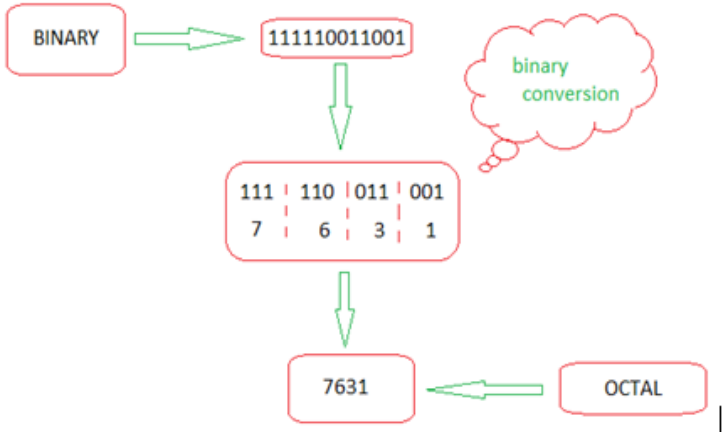
Conversions between binary and octal number systems

Binary to Octal conversion

How to convert $(111110011001)_2$ into its corresponding octal equivalent?

STEPS:

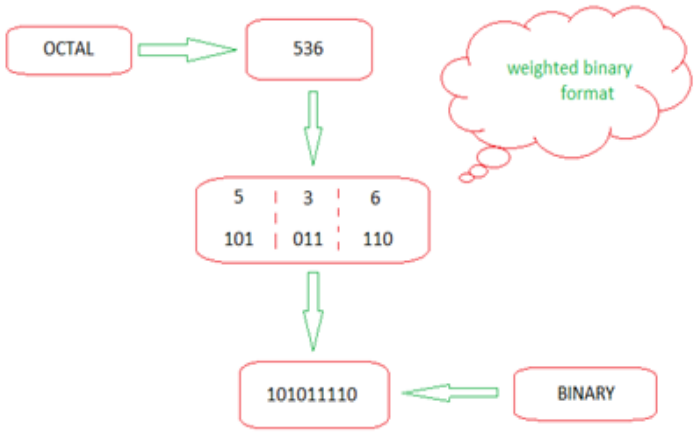
I. Split the number into groups of three from LSB. (Groups of three is done because $2^3=8$ for octal number system).
II. If at all the number near MSB remains as a single or two bit number, preceed the number with zeros.
III. Then, do the conversion.



Octal to Binary conversion

How to convert $(536)_8$ into the corresponding binary representation?

This is doneby separating the individual numbers and converting the number into binary in three bit form.



Hence, the binary representation is $(101011110)_2$.

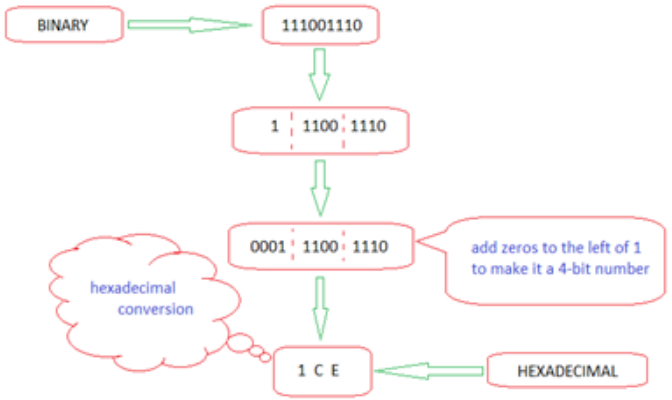
Conversions between Binary and Hexadecimal number systems

Binary to Hexadecimal conversion

How to convert $(111001110)_2$ into the corresponding hexadecimal equivalent?

STEPS:

I. Split the number into groups of four from LSB. (Groups of three is done because $2^4=16$ for hexadecimal number system).
II. If at all the number near MSB remains as a single or two bit number, preceed the number with zeros.
III. Then, do the conversion.

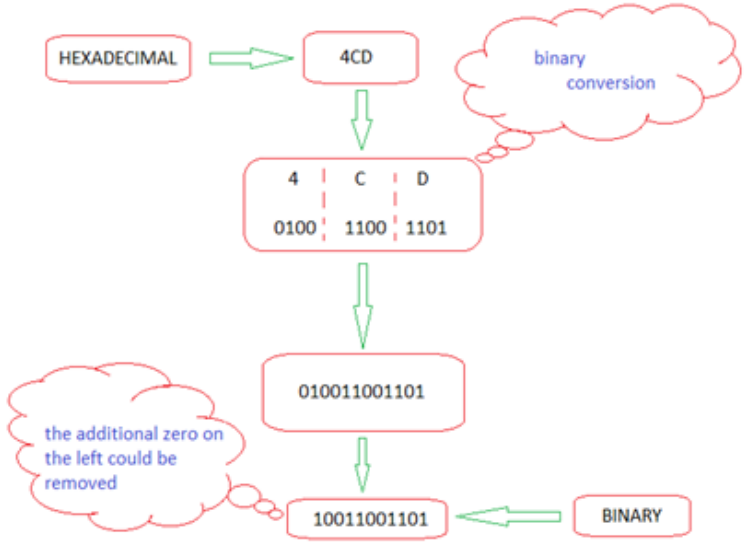


Hence, the corresponding hexadecimal equivalent is $(1CE)_{16}$.

Hexadecimal to Binary conversion

How to convert $(4CD)_{16}$ into its corresponding binary form?

This is done by separating the individual numbers and converting the number into binary in four bit form.



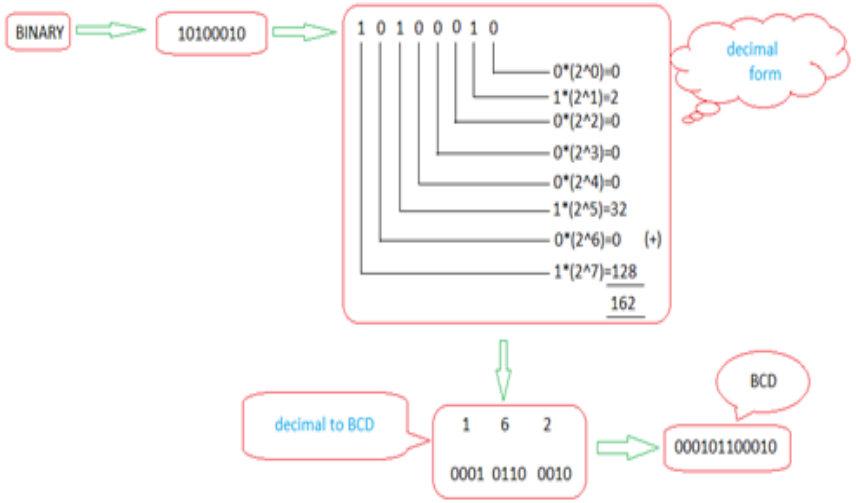
Hence, the resultant binary is $(10011001101)_2$.

Conversions between BCD and Binary number systems

Binary to BCD conversion

Binary to BCD conversion cannot be done directly. This is done by converting the binary number into its corresponding decimal form, then again converting the decimal form into BCD.

How to convert $(10100010)_2$ into BCD?

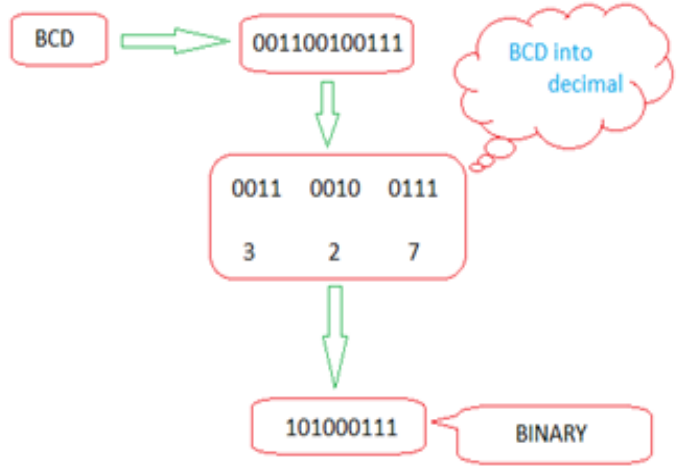


Hence, the required BCD form is 000101100010.

BCD to Binary conversion

BCD to Binary conversion is just vice versa of Binary to BCD conversion. First, convert the BCD number into equivalent decimal number and then into its corresponding binary equivalent.

How to convert 001100100111 into BCD equivalent?



Thus, the binary converted value is 101000111.

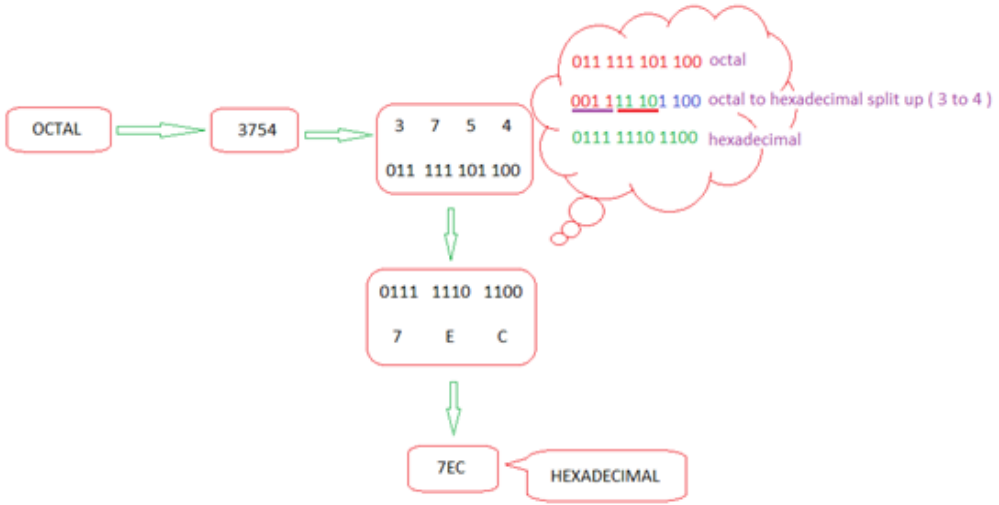
Conversions between Octal and Hexadecimal number systems

Octal to Hexadecimal conversion

How to convert $(3754)_8$ into its hexadecimal equivalent?

STEPS to be followed for the above conversion:

- The given octal number is converted into binary by converting each octal bit into a 3-bit binary number.
- This binary number is grouped into 4bits from LSB.
- Then, convert this 4-bit binary grouping into a hexadecimal number.



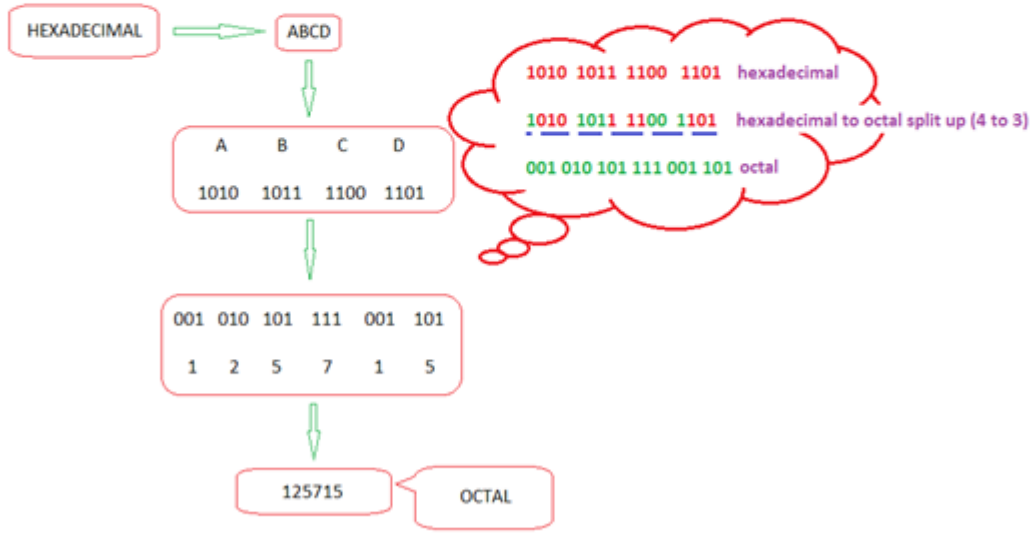
Hence, the hexadecimal equivalent is $(7EC)_{16}$.

Hexadecimal to Octal conversion

How to convert $(ABCD)_{16}$ into its corresponding octal equivalent?

STEPS to be followed for the above conversion:

- The given hexadecimal number is converted into binary by converting each hexadecimal bit into a 4-bit binary number.
- This binary number is grouped into 3bits from LSB.
- Then, convert this 3-bit binary grouping into an octal number.



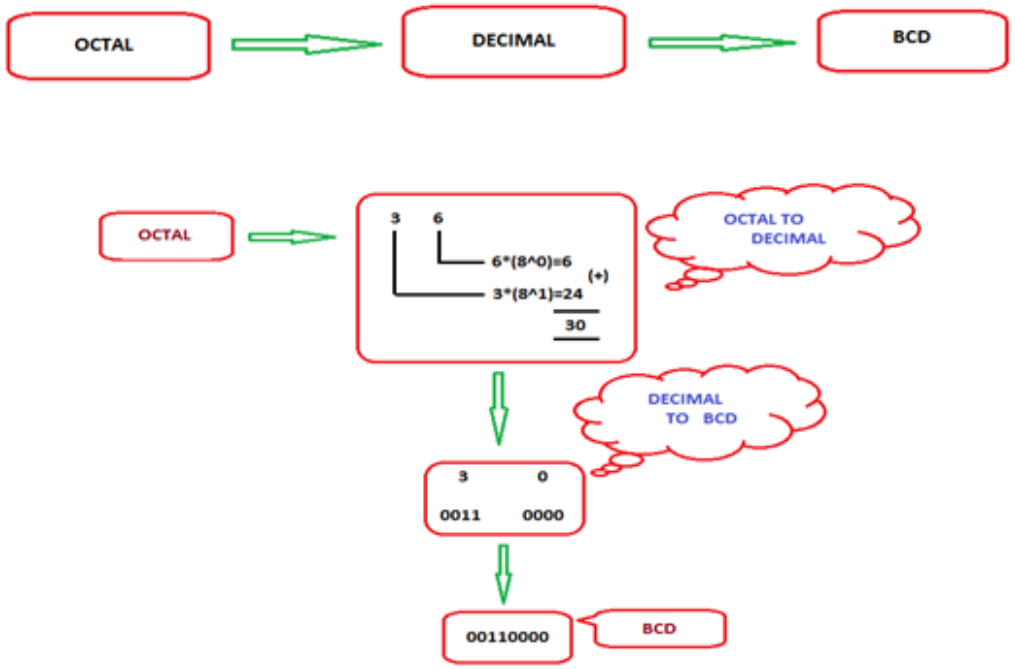
Hence, the octal equivalent is $(125715)_8$.

Conversions between BCD and octal number systems

Octal to BCD conversion

How to convert $(36)_8$ into BCD?

This can be done by the following chain:

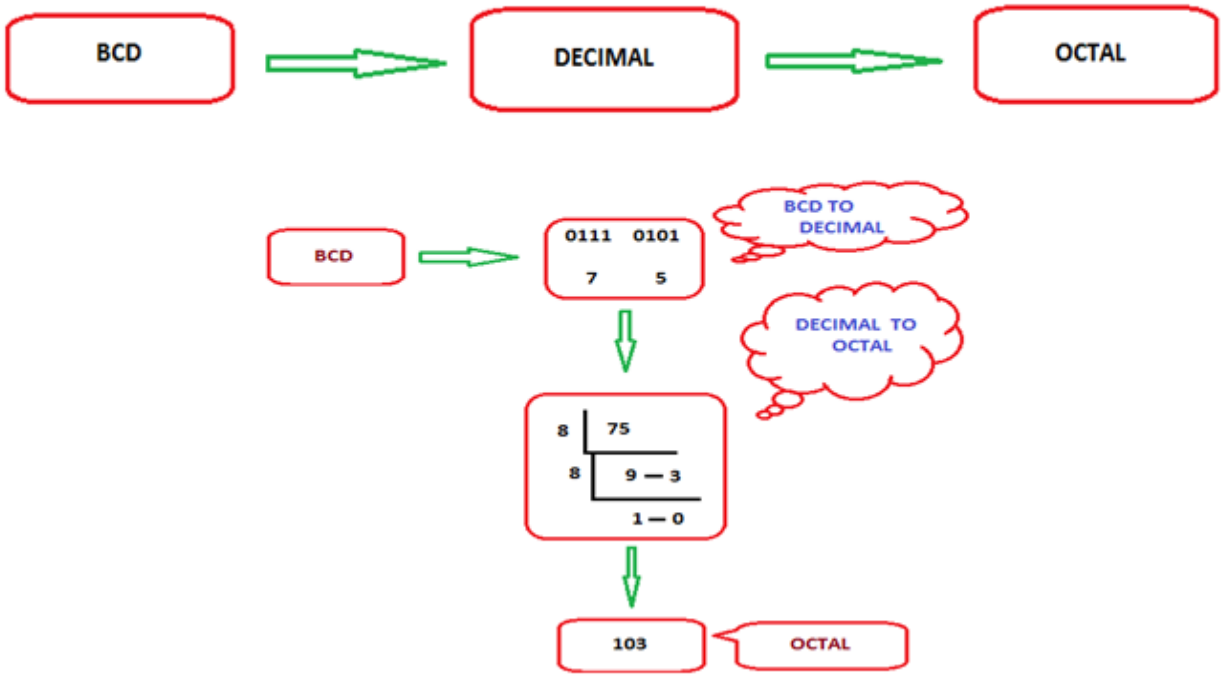


Hence, the required BCD is (0011 0000).

BCD to Octal conversion

How to convert (0111 0101) into octal number?

The following chain depicts the conversion schematic.

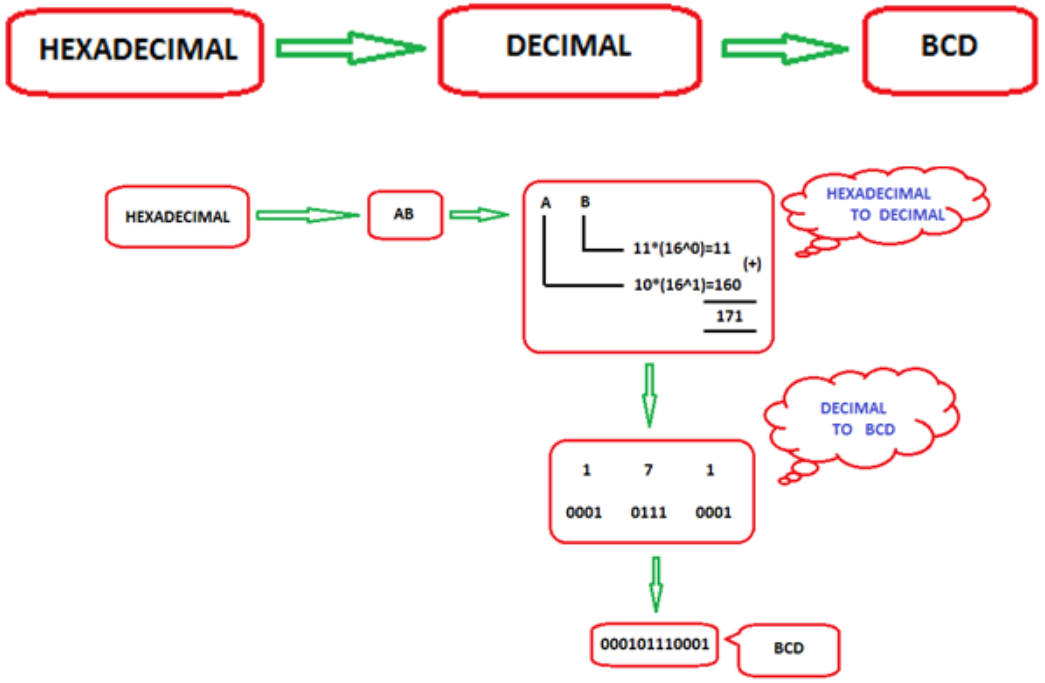


Hence, the required octal value is $(103)_8$.

Conversion between BCD and Hexadecimal number system

Hexadecimal to BCD conversion

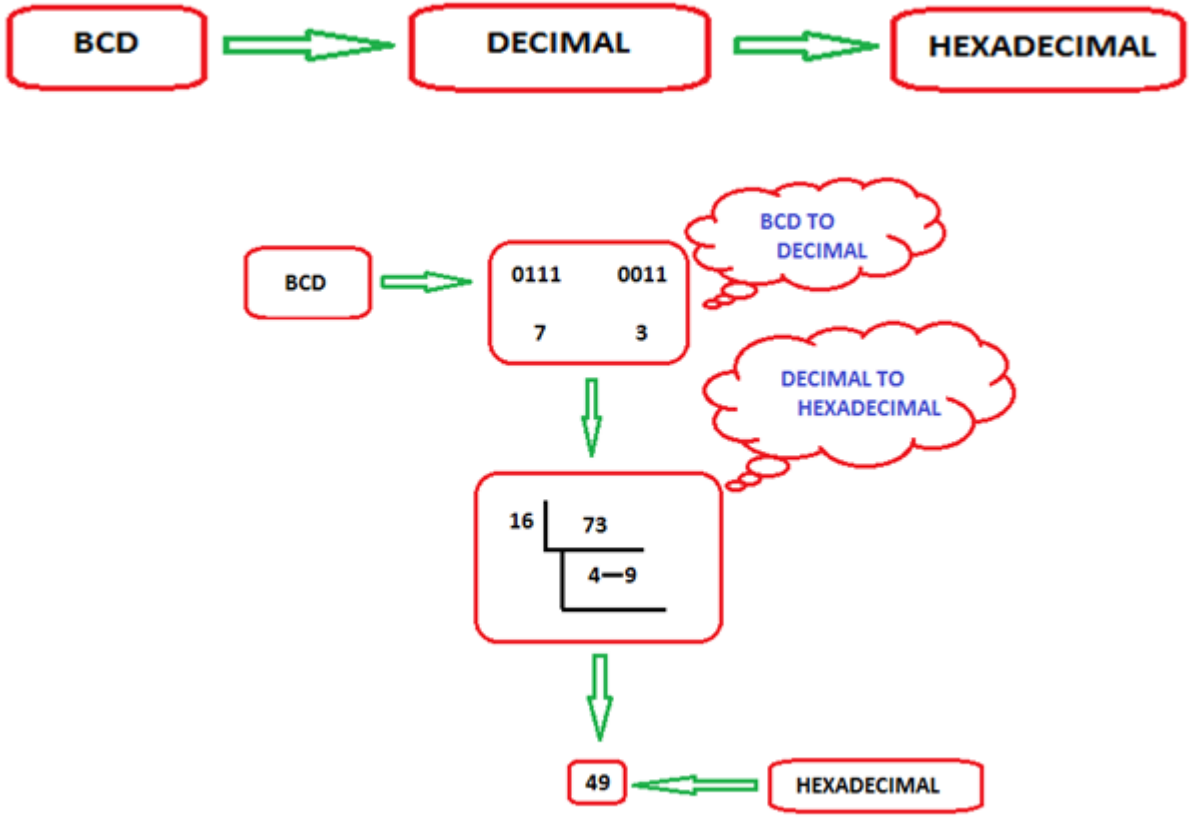
Convert $(AB)_{16}$ into BCD.



Hence, the required BCD is (0001 0111 0001).

BCD to Hexadecimal conversion

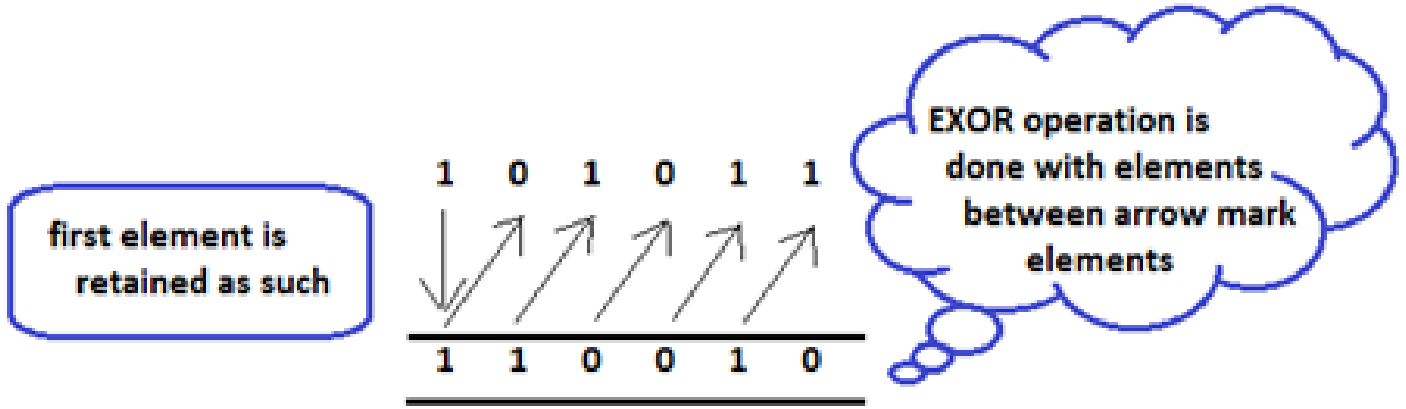
Convert (0111 0011) into hexadecimal number.



Hence, the hexadecimal equivalent is $(49)_{16}$.

BCD TO Gray code conversion

How to convert 101011 into its corresponding gray code?



TRUTH TABLE FOR EXOR GATE

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

The resulting gray code is 110010.

Points to remember

- Nothing Really, Work out ☺

Lets Go Advanced…

Shriram K Vasudevan

shriramkv@gmail.com