Batch: B3	**Roll No.: 16010122151**
Experiment / assignment / tutorial No. 10	
Grade: AA / AB / BB / BC / CC / CD /DD	

TITLE: Study of basic computer organisation and architecture concepts through Virtual lab

AIM: Understanding Virtual Lab concepts

Expected OUTCOME of Experiment:

Summarize Input output techniques and multiprocessor configurations

---

Books/ Journals/ Websites referred:

http://vlabs.iitb.ac.in/vlab/labscse.html
http://vlabs.iitb.ac.in/vlab/#
http://www.vlab.co.in/

Pre Lab/ Prior Concepts:

The main aim of this experiment is to provide remote-access to Labs in various disciplines of Science and Engineering. These Virtual Labs would cater to students at the undergraduate level, post graduate level as well as to research scholars. Also, to enthuse students to conduct experiments by arousing their curiosity. This would help them in learning basic and advanced concepts through remote experimentation. It also provides a complete Learning Management System around the Virtual Labs where the students can avail the various tools for learning, including additional web-resources, video-lectures, animated demonstrations and self-evaluation. We can share costly equipment and resources, which are otherwise available to limited number of users due to constraints on time and geographical distances

<u>Salient Features:</u>

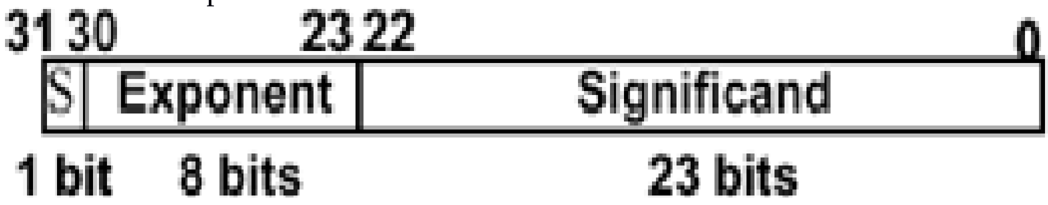
1. Virtual Labs will provide to the students the result of an experiment by one of the following methods (or possibly a combination)
   - Modeling the physical phenomenon by a set of equations and carrying out simulations to yield the result of the particular experiment. This can, at-the-best, provide an approximate version of the 'real-world' experiment.
   - Providing measured data for virtual lab experiments corresponding to the data previously obtained by measurements on an actual system.
   - Remotely triggering an experiment in an actual lab and providing the student the result of the experiment through the computer interface. This would entail carrying out the actual lab experiment remotely.
2. Virtual Labs will be made more effective and realistic by providing additional inputs to the students like accompanying audio and video streaming of an actual lab experiment and equipment.

<u>Observations</u>
Title of Study Experiment:
Floating Point Numbers Representation

Brief description of experiment under study:

- What are floating point numbers?
  As the name implies, floating point numbers are numbers that contain floating decimal points. For example, the numbers 5.5, 0.001, and -2,345.6789 are floating point numbers. Numbers that do not have decimal places are called integers. Computers recognize real numbers that contain fractions as floating point numbers. When a calculation includes a floating point number, it is called a "floating point calculation." Older computers used to have a separate floating point unit (FPU) that handled these calculations, but now the FPU is typically built into the computer's CPU.

- Floating- Point Format
  Computers store floating point numbers in a specific format. As our aim is to maximize the range of numbers that can be stored. FPUs typically represent real numbers in a binary floating-point format. In this format, a real number has three parts: a sign, a significand, and an exponent. The following explanation is in accordance with the standard IEEE format of 32-bit representation.



Objective of this experiment is to learn the fundamentals of Floating Point Representation of Numbers.

Learning's recorded:

<u>Exponent Field</u>
-> 8 - Bits Long
-> Determines the range of numbers that can be represented
-> Increasing the bits will increase the range, not precision -> To cover for –ve numbers,
exp = 127 + real exp

Sign Bit
-> 1- Bit Long
-> Determines the +ve or -ve number -> 1 = -ve number 0 = +ve number

Mantissa Field
-> 23 - Bits Long
-> Determines the precision of numbers
-> Increasing bits will increase precision, not range.

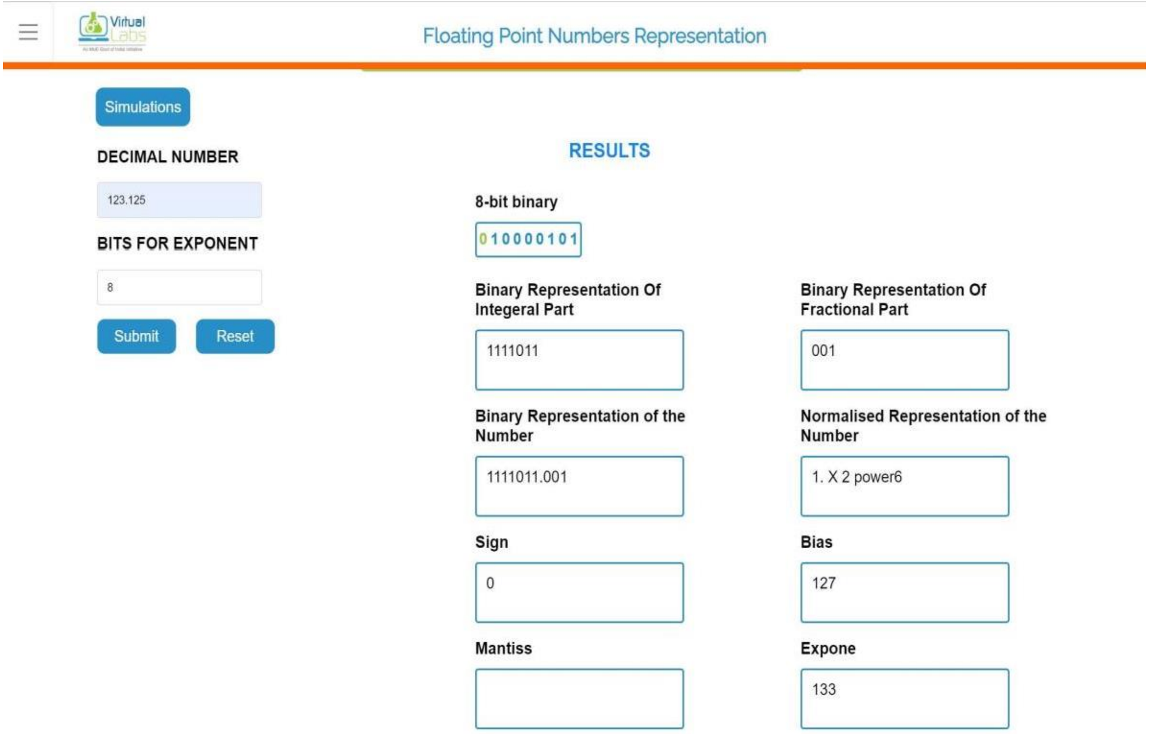
Consequence #1: Values are unevenly spaced
-> Imagine we only had 6 bits for each floating-point number (1 sign, 3 mantissas, 2 exponent)
-> Means less absolute precision for numbers with larger magnitudes
-> More bits, more the accuracy
-> That is why the numbers are denser near the zero as compared to far away from the number line.

Consequence #2: Round off errors
-> Our system can represent 6, and it can represent ¼, but not 5¾
-> So 6 - 0.25 is 6, not 5.75
-> And if 6 - 0.25 - 0.25 - 0.25 - 0.25 is evaluated left to right, the answer is still 6
-> This is not random happens exactly the same way every time.

Procedure for the Conversion:

- Convert the absolute value of the number to binary, perhaps with a fractional part after the binary point. This can be done by converting the integral and fractional parts separately.
- Normalize the number. Move the binary point so that it is one bit from the left. Adjust the exponent of two so that the value does not change. Place the mantissa into the mantissa field of the number. Omit the leading one and fill with zeros on the right.
- Add the bias to the exponent of two, and place it in the exponent field. The bias is 2k-1-1, where k is the number of bits in the exponent field. 6.
- For the eight-bit format, k-3, so the bias is 23-1-1-3. For IEEE 32-bit, k-8, so the bias is 28-1-1-127.
- Set the sign bit, 1 for negative, 0 for positive, according to the sign of the original number.

Screenshots from Simulator:



Handwritten solution:

Decimal number: 123.125

STEP 1: To Binary: 1111011.001

2 | 123
2 | 61 − 1
2 | 30 − 1
2 | 15 − 0
2 | 7 − 1
2 | 3 − 1
1 − 1

0.125 × 2 = 0.250 − 0
0.250 × 2 = 0.500 − 0
0.500 × 2 = 1.00 − 1

STEP 2: Normalization: 1.111011001×2^6
power = 6.

STEP 3:

Single Precision	Double Precision
E − 127 = 6	E − 1023 = 6
E = 127 + 6 = 133	E = 6 + 1023 = 1029
In Binary: 10000101	In Binary: 10000000101

STEP 4: Representation:

Single Precision:

Sign	Biased Exponent	Mantissa
0	10000101	111011001000..00.. (23 bits)

Double Precision:

Sign	Biased Exponent	Mantissa
0	10000000101	111011001000.00...0.. (52 bits)

Screenshots from Pre-test:

Floating Point Numbers Representation

1. Consider the decimal number 15.75. How can we convert this entire number (both the integer and fractional parts) to binary?

- ○ a: Repeated multiplication to convert the fractional part (0.75).
- ○ b: Repeated division to convert the integer part (15)
- ◉ c: Repeated division to convert the integer part (15), Repeated multiplication to convert the fractional part (0.75) and then combine both parts.
- ○ d: None of the aobve

2. In 32 bit IEEE Format Floating Point the Sign bit equals

- ◉ a: 1 when the number is negative and is 0 when the number is positive.
- ○ b: 0 when the number is negative and is 1 when the number is positive.
- ○ c: 0 when the number is negative and is 0 when the number is positive.
- ○ d: All of the above

Submit Quiz

2 out of 2

Screenshots from Post-test:

Floating Point Numbers Representation

1. Can zero be represented using normalized representation?

- ○ a: Yes
- ◉ b: No
- ○ c: Sometimes
- ○ d: Information is not sufficient

2. Which of the following is true for floating number 0.3?

- ◉ a: Only 5-bit binary Normalised representation is possible
- ○ b: 5-bit binary Denormalised representation is possible
- ○ c: It's Exponent is 00, Mantissa is 00 and Denorm is 0
- ○ d: Both b and c

Submit Quiz

2 out of 2

Knowledge gained / Inference Obtained:

The importance of Floating Point Representation was understood: Floating point representation makes numerical computation much easier.

The procedure to convert a decimal to floating point representation was also studied with the help of an example and verifying it manually as well.

Post Lab Descriptive Questions

1. **What are the applications of the virtual lab case study / tool reviewed by you?**

- Tensor Processing Units (TPUs)
  Besides the 64-bit float we explored at length, there are also 32-bit floats (single precision) and 16-bit floats (half-precision) commonly available.
  Google's Tensor Processing Units instead use a modified 16-bit format for multiplication as part of their many optimizations for deep-learning tasks.
- HDR Images
  HDR image uses floating point numbers to represent the pixels! This allows a high "dynamic" range (the exponent can be high or low) while still maintaining relative precision across all brightness scales. Perfect for keeping the data from scenes with high contrast.

Conclusion:

The various concepts of computer organisation and architecture were revised and understood efficiently using virtual lab.

Date: 10th November, 2022 **Signature of faculty in-charge**