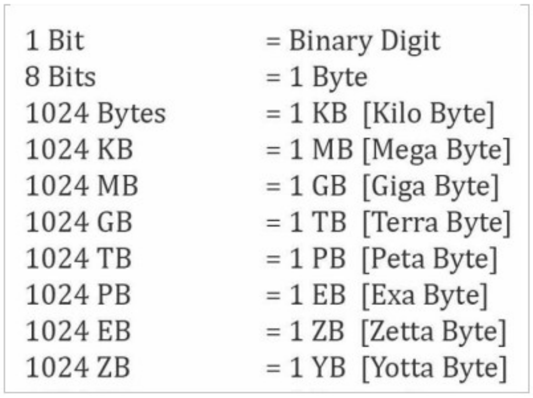
UNIVERSITY OF BIRMINGHAM

M.Sc. CS / AI & CS
Computer Systems
Floating Point Numbers, Memory, Program Execution [Solutions]

Question #1: Find out the equivalent 32-bit Floating point representation for the following?

Decimal Fraction	Binary Fraction	Floating Point Representation (32 bit)		
104.1875	1101000.0011	0	1000 0101	101 0000 0110 0000 0000 0000
-213.4375	−11010101.0111	1	1000 0110	101 0101 0111 0000 0000 0000
-0.15625	−0.00101	1	0111 1100	010 0000 0000 0000 0000 0000

Question #2: An address space is a range of valid addresses in memory that are available for a program or process. That is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data.
Considering the memory address sizes given in the table below, find out the maximum memory that could be installed in a given system. Also mention the minimum and maximum addresses in Hexadecimal.



1 Bit	= Binary Digit
8 Bits	= 1 Byte
1024 Bytes	= 1 KB [Kilo Byte]
1024 KB	= 1 MB [Mega Byte]
1024 MB	= 1 GB [Giga Byte]
1024 GB	= 1 TB [Terra Byte]
1024 TB	= 1 PB [Peta Byte]
1024 PB	= 1 EB [Exa Byte]
1024 EB	= 1 ZB [Zetta Byte]
1024 ZB	= 1 YB [Yotta Byte]

Address Size	Max Supported Memory (Address Space)	Minimum Memory Address	Maximum Memory Address
8 bits	256 bytes	0x00	0xFF
16 bits	64 KB	0x0000	0xFFFF
32 bits	4 GB	0x00000000	0xFFFFFFFF
64 bits	16 EB	0x0000000000000000	0xFFFFFFFFFFFFFFFF

UNIVERSITY OF BIRMINGHAM

Question #3: Google have been estimated as having a total storage capacity of about 15 exabytes (a couple of years ago). We didn't get as far as the exabyte during our lectures, but it's 10^{18} bytes. If Google's storage were made into a single byte-addressed memory, how many bytes would the addresses have to be?

1. The total number of bytes is about 15×10^{18}. The first step is to find roughly what power of 2 this would be. 15 is approximately 16, which is 2^4. $10^{18} = 10^{3 \times 6} = 1000^6$, and this is approximately $1024^6 = 2^{10 \times 6} = 2^{60}$. Hence 15×10^{18} is approximately $2^{4+60} = 2^{64}$. (If you're not used to these calculations with powers, check that they make sense when you think of - e.g. - 2^{60} as 60 copies of 2 all multiplied together.)
   This means that there are 2^{64} byte locations that all need different addresses. A 64-bit address would give you that number of possibilities. 64 bits is 8 bytes, so you need an 8-byte address.
   To put this into perspective: Windows comes in 32-bit and 64-bit versions, determined by the size of addresses used in different computers. 32-bit systems are limited to 4 GB of RAM, and that began to seem cramped. A 64-bit system allows your computer to be as big as Google (don't try this at home, though).

Question #4: What is the value stored in 'z' after the execution of the following MIPS assembly code, if x=5?

```
li $6, 1
li $7, 2
lw $8, &x
L1:  bgt $7, $8, L2
     mult $6, $6, $7
     addi $7, $7, 1
     j L1
L2: sw $6, &z
```

Solution:

The bgt (branch on greater than) instruction compares two registers and jumps to L2 if the content of register $7 is greater than the content of register $8. The equivalent Java code is a for loop:

```
z = 1;
for (i = 2; i  <= x; i++){
     z = z * i;
}
```

Knowing that x = 5 and the index i starts from 2, the result will be:

`z = 1 * 2 * 3 * 4 * 5 = 120.`

UNIVERSITY OF BIRMINGHAM

Question #5: A processor is operating at 30MHz. Each instruction takes a minimum of 6 cycles to execute. The processor has a six stage pipeline. If a program starts execution at time 0, what is the theoretical maximum number of instructions that will have completed their execution at the end of 1 millisecond?

Speed = 30MHz = 30,000,000 cycles/second = 30,000 cycles / millisecond
With pipelining:
Number of clock cycles taken by the first instruction = 6 cycles.
After the first instruction has completely executed, one instruction comes out per cycle.
So, number of cycles taken by each remaining instruction = 1 cycle.
Number of executed instructions =
First Instruction (1) + remaining instructions (30,000 cycles/ms – 6 cycles)
= 1 + 29,994 = 29,995 instruction/ms
In practice, there are factors that mean that this theoretical maximum is never reached.