**Level 2: Research Questions**

1. The Intel 8085 microprocessor was a first generation processor that was used in many early game systems and personal computers. Google “8085 microprocessor architecture” to answer these questions.
   1. Year Introduced: 1976
   2. Size of data bus (in bits): 8 bit
   3. Largest data number (in binary and decimal)= 2^8 - 1
   4. Size of address bus (in bits); 16 bit
   5. Largest memory address (in binary and decimal)= 2^16 -1
2. The Intel 8086 microprocessor was the processor used in the first IBM PCs running the DOS operating system. Google “8086 microprocessor architecture” to answer these questions.
   1. Year Introduced: 1976
   2. Size of data bus (in bits): 16 bit
   3. Largest data number (in decimal)= 2^16 - 1
   4. Size of address bus (in bits): 20 bit
   5. Largest memory address (in decimal)= 2^20 - 1
3. The Intel 80286 microprocessor a common processor used in IBM PCs running the Windows operating system. Google “80286 microprocessor architecture” to answer these questions.
   1. Year Introduced: 1982
   2. Size of data bus (in bits): 16 bit
   3. Largest data number (in decimal)= 2^16 - 1
   4. Size of address bus (in bits): 24 bit
   5. Largest memory address (in decimal) 2^24 - 1
4. The modern PCs run either a 32 bit or 64 bit Windows operating system. Google “32 vs 64 bit” to answer these questions.
   1. How do these systems differ in data capacity? (explain using bits)
   2. How do these systems differ in memory capacity? (explain using bits)
   3. How do these systems differ in hardware requirements?

A 64-bit processor is more capable than a 32-bit processor, because it can handle more data at once. A 64-bit processor is capable of storing more computational values, including memory addresses, which means it’s able to access over four billion times the physical memory of a 32-bit processor. That’s just as big as it sounds. Another big difference between 32-bit processors and 64-bit processors is the maximum amount of memory (RAM) that is supported. 32-bit computers support a maximum of 4 GB (232 bytes) of memory, whereas 64-bit CPUs can address a theoretical maximum of 18 EB (264 bytes).

1. Research and explain how negative (-) numbers are represented using bits and how they are stored in computer memory.

Negative numbers however, are represented by taking the one's complement (inversion, negation) of the unsigned positive number. Since positive numbers always start with a “0”, the complement will always start with a “1” to indicate a negative number.

1. Research and explain how floating point (decimal) numbers are represented using bits and how they are stored in computer memory.

“ The basic idea in decimal is that the places after the decimal point have values 1/10, 1/100, 1/1000, etc. In binary, we again use powers of 2 instead, so the places to the right of the binary point have values 1/2, 1/4, 1/8, 1/16, etc.

For example:

.1011 = 1/2+1/8+1/16 = 11/16” (Quora)

**Level 3: Sample Program**

1. Explain the result of the following Python operations:
   1. bin(11)
   2. oct(11)
   3. hex(11)

it converts the number to binary, octal, and hexadecimal

bin(11)

'0b1011'

oct(11)

'0o13'

hex(11)

'0xb'

1. Explain the following Python operations:
   1. bin(‘11’) - Why does this operation give an error?
   2. int(‘11’) - Why does this work?
   3. bin(int(‘11’)) - Why does this fix the problem?

bin('11')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'str' object cannot be interpreted as an integer

int('11')

11

bin(int('11'))

'0b1011'

1. Modify the following sample Python program to print out the digits in:

number = input("Enter a 4 digit decimal number:")

index = 0

for char in number :

index += 1

print("Digit ", index, " is : ", char)

* 1. Binary

number = input("Enter a 4 digit decimal number:")

index = 0

for char in number :

index += 1

print("Digit ", index, " is : ", bin(int(index)))

* 1. Octal

number = input("Enter a 4 digit decimal number:")

index = 0

for char in number :

index += 1

print("Digit ", index, " is : ", oct(int(index)))

* 1. Hexadecimal

number = input("Enter a 4 digit decimal number:")

index = 0

for char in number :

index += 1

print("Digit ", index, " is : ", hex(int(index)))