Number Representation & Strings

A. What is the value of the signed char 0x9E in decimal?

B. What is the value of the unsigned char 37 in binary?

C. If \( a = 0x2C \), complete the bitwise C statement so that \( b = 0x1F \).

\[
b = a \_\_\_0x\_\_\_
\]

For the following problems we are working with a floating point representation that follows the same conventions as IEEE 754 except using 7 bits split into the following fields:

- Sign (1)
- Exponent (3)
- Mantissa (3)

D. What is the magnitude of the bias of this new representation?

E. What is the decimal value encoded by 0b1110101 in this representation?

F. What value will be read after we try to store -18 in this representation? (Circle one)

-16 -NaN -∞ -18

For the following problem, assume we are working with C strings encoded in ASCII. Consider the declaration:

\[
\text{char str[]} = \text{“Hello!”};
\]

G. What will be stored in the array str?
**Pointers & Memory**

For this problem we are using a 64-bit x86-64 machine ([little endian](https://en.wikipedia.org/wiki/LittleEndian)). The current state of memory (values in hex) is shown below:

```c
char* charP = 0xD;
short* shortP = 0x1E;
```

<table>
<thead>
<tr>
<th>Word Addr</th>
<th>+0</th>
<th>+1</th>
<th>+2</th>
<th>+3</th>
<th>+4</th>
<th>+5</th>
<th>+6</th>
<th>+7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>20</td>
<td>F6</td>
<td>EF</td>
<td>EA</td>
<td>A2</td>
<td>5E</td>
<td>9F</td>
<td>1A</td>
</tr>
<tr>
<td>0x08</td>
<td>A2</td>
<td>D0</td>
<td>4F</td>
<td>C4</td>
<td>A0</td>
<td>OC</td>
<td>F7</td>
<td>27</td>
</tr>
<tr>
<td>0x10</td>
<td>B8</td>
<td>BD</td>
<td>1A</td>
<td>CA</td>
<td>35</td>
<td>95</td>
<td>CB</td>
<td>80</td>
</tr>
<tr>
<td>0x18</td>
<td>84</td>
<td>3F</td>
<td>02</td>
<td>4F</td>
<td>8E</td>
<td>F3</td>
<td>F6</td>
<td>E5</td>
</tr>
<tr>
<td>0x20</td>
<td>CD</td>
<td>4A</td>
<td>F6</td>
<td>48</td>
<td>1A</td>
<td>6F</td>
<td>7E</td>
<td>63</td>
</tr>
</tbody>
</table>

A. Using the values shown above, ill in the C type and hex value for each of the following C expressions. Leading zeros are not required for the hex values.

<table>
<thead>
<tr>
<th>C Expression</th>
<th>C Type</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>*(charP + 6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(int**)shortP - 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. For the following snippet of C code, draw out a box-and-arrow diagram for the allocated memory.

```c
int x = 351, y = 332;
int *p = &x;
int **q = &p;
q = &y;
(*q) = x;
```
C & Assembly

Answer the questions below about the following x86-64 assembly function:

```
mystery:
  jmp       .L2            # Line 1
.L2:      addq  $1, %rdi    # Line 2
      movb  %al, (%rsi)    # Line 3
      leaq  1(%rsi), %rsi # Line 4
.L4:      movzbl (%rdi), %eax # Line 5
      testb %al, %al      # Line 6
      je      .L3         # Line 7
      cmpb  %dl, %al      # Line 8
      jne     .L4         # Line 9
.L3:      movb  $0, (%rsi) # Line 10
      retq            # Line 11
```

A. What variable type would %rdi be in the corresponding C program?

B. What variable type would the third argument be in the corresponding C program?

C. This function uses a while loop. Fill in the two conditionals below, using register names as variable names (no declarations necessary).

   while (_________ && _________ )

D. Taking the variable types into account, describe at a high level what the purpose of Line 10 is (not just what it does mechanically).

E. Describe at a high level what you think this function accomplishes (not line-by-line).
Procedures & The Stack

The recursive function `count_nz` counts the number of non-zero elements in an int array. Example: if `int a[] = {-1,0,1,255}`, then `count_nz(a,4)` returns 3. The function and its x86-64 disassembly are shown below:

```c
int count_nz(int* ar, int num) {
    if (num > 0)
        return !!(*ar) + count_nz(ar + 1, num - 1);
    return 0;
}
```

```
0000000000400536 <count_nz>:
    400536: 85 f6       testl %esi,%esi
    400538: 7e 1b       jle 400555 <count_nz+0x1f>
    40053a: 53          pushq %rbx
    40053b: 8b 1f       movl (%rdi),%ebx
    40053d: 83 ee 01     subl $0x1,%esi
    400540: 48 83 c7 04  addq $0x4,%rdi
    400544: e8 ed ff ff ff  callq 400536 <count_nz>
    400549: 85 db       testl %ebx,%ebx
    40054b: 0f 95 c2     setne %dl
    40054d: 0f b6 d2     movzbl %dl,%edx
    400551: 01 d0       addl %edx,%eax
    400553: eb 06       jmp 40055b <count_nz+0x25>
    400555: b8 00 00 00 00  movl $0x0,%eax
    40055a: c3          retq
    40055b: 5b          popq %rbx
    40055c: c3          retq
```

A. How much space (in bytes) does this function take up in our final executable?

B. The compiler automatically creates labels it needs in assembly code. How many labels are used in `count_nz` (including the procedure itself)?
C. In terms of the *C function*, what value is being saved on the stack?

D. What is the return address to `count_nz` that gets stored on the stack (in hex)?

E. Assume `main` calls `count_nz(a, 5)` with an appropriately-sized array and then prints the result using `printf`. Starting with (including) `main`, answer the following *in the number of stack frames*.

<table>
<thead>
<tr>
<th>Total created:</th>
<th>Max depth:</th>
</tr>
</thead>
</table>

F. Assume `main` calls `count_nz(a, 6)` with `int a[] = {3, 5, 1, 4, 1, 0}`. We find that the return address to `main` is stored on the stack at address `0x7fffeca3f748`. What data will be stored on the stack at address `0x7fffeca3f720`?

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x7fffeca3f748</td>
<td>&lt;ret addr to main&gt;</td>
</tr>
<tr>
<td>0x7fffeca3f740</td>
<td></td>
</tr>
<tr>
<td>0x7fffeca3f738</td>
<td></td>
</tr>
<tr>
<td>0x7fffeca3f730</td>
<td></td>
</tr>
<tr>
<td>0x7fffeca3f728</td>
<td></td>
</tr>
<tr>
<td>0x7fffeca3f720</td>
<td></td>
</tr>
</tbody>
</table>

G. A similar function `count_z` that counts the number of zero elements in an array is made by making a single change to `count_nz`. What is the address of the changed assembly instruction?
Design Questions

A. What values can $S$ take in an $x86-64$ memory operand? Briefly describe why these choices are useful/important.

<table>
<thead>
<tr>
<th>Values:</th>
<th>Importance:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. Until very recently (Java 8/9), Java did not support unsigned integer data types. Name one advantage and one disadvantage to this decision to omit unsigned.

<table>
<thead>
<tr>
<th>Advantage:</th>
<th>Disadvantage:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. Condition codes are part of the processor/CPU state. Would our instruction set architecture (ISA) still work if we got rid of the condition codes? Briefly explain.

<table>
<thead>
<tr>
<th>Circle one: Yes No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Explanation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>