**Source File:** `lab38.asm`
**Input:** Standard Input
**Output:** Standard Output
**Value:** 3

For this assignment you are to write three assembly language functions. The first writes the binary (base 2) representation of a signed 32-bit integer to stdout; the second writes the octal (base 8) representation of a signed 32-bit integer to stdout; and the third writes the hexadecimal (base 16) representation of a signed 32–bit integer to stdout. Each function should be written recursively.

Client code for testing your implementation is shown in Figure 1, and a sample execution sequence is shown in Figure 2.

Do not use any "global" variables in any of your functions. Each function should contain a single `.text` section. There should be no other sections (e.g., `.bss`, `.data`, `.rodata`, etc.). Use the system stack for any local variables.

```
1  /*
2    gcc -m32 -c lab38main.c
3    nasm -f elf32 -o lab38.o lab38.asm -I/usr/local/3304/include/
4    gcc -m32 -o lab38 lab38main.o lab38.o /usr/local/3304/src/Along32.o -lm
5    ./lab38 < 01.dat
6  */
7  #include <stdio.h>
8  #include <math.h>
9  #include <limits.h>
10
11 // PrintBinary is a recursive assembly-language function that writes
12 // the binary (base 2) representation of n to stdout
13 void PrintBinary(int n);
14
15 // PrintOctal is a recursive assembly-language function that writes
16 // the octal (base 8) representation of n to stdout
17 void PrintOctal(int n);
18
19 // PrintHexadecimal is a recursive assembly-language function that writes
20 // the hexadecimal (base 16) representation of n to stdout
21 void PrintHexadecimal(int n);
22
23 int main()
24 {
25   int n, bases[] = {2,8,16};
26   char hrule[74];
27   // declare an array of pointers, each element containing the address
28   // of a function
29   void (*func[])(int) = {&PrintBinary, &PrintOctal, &PrintHexadecimal};
30
31   // Initialize hrule to contain hyphens
32   for (char *ptr = hrule; ptr < hrule + sizeof(hrule) - 1; ++ptr)
33     *ptr = '-';
34   *(hrule + sizeof(hrule) - 1) = '\0';
35
```

**Figure 1.** `/usr/local/3304/src/lab38main.c` (Part 1 of 3)

```
36    // print the table heading
37    printf("%s\n",hrule);
38    printf("   Decimal    ");
39    printf("             Binary               ");
40    printf("   Octal     ");
41    printf("Hexadecimal\n");
42    printf("%s\n",hrule);
43
44    // read an unknown # of ints from stdin; input terminates when the
45    // end-of-data marker is encountered
46    while (scanf("%d", &n) == 1)
47    {
48      printf(" %11d", n);
49
50      // Loop to call each of the assembly-language functions
51      for (int i = 0; i < sizeof(func) / sizeof(func[0]); ++i)
52      {
53        printf("  ");
54        if (bases[i] == 16)
55          printf("  ");
56
57        // For the given base, determine the power of 2. For example,
58        // if base = 2, the power of 2 is 1; if base = 8, the power
59        // of 2 is 3; and so on. To compute this, take the base_2 log
60        // of the base.
61        int powerOf2 = (int) log2(bases[i]);
62
63        // Calculate how many bits are in the internal representation
64        // of an int. Then divide by the power of 2 to determine the
65        // number of groupings that will be printed. If the division
66        // process yields a remainder, increase the # of groups by 1.
67        int width = sizeof(int) * CHAR_BIT / powerOf2;
68        width += (sizeof(int) * CHAR_BIT % powerOf2) ? 1 : 0;
69
70        // For positive ints, determine the # of digits needed to
71        // display n in base[i]. Determine this by computing the
72        // base[i] log of n. Truncate the logarithm and add 1.
73        if (n > 0)
74          width -= ((int) (log2((double) n) / log2(bases[i])) + 1);
75
76        // For non-negative n, insert the appropriate # of leading 0s
77        if (n >= 0)
78          for (int j = 0; j < width; ++j)
79            printf("0");
80
81        // Empty all non-empty output buffers before calling any of the
82        // assembly functions.
83        fflush(0);
84
```

**Figure 1.** `/usr/local/3304/src/lab38main.c` (Part 2 of 3)

```
85        // For non-zero input values, call the assembly-language function,
86        // passing n on the system stack.
87        if (n != 0)
88          (*func[i])(n);
89      }
90
91      printf("\n");
92    }
93
94    printf("%s\n",hrule);
95
96    return 0;
97  }
```

**Figure 1.** `/usr/local/3304/src/lab38main.c` (Part 3 of 3)

```
newuser@csunix ~/3304/38> cp /usr/local/3304/data/38/* .
newuser@csunix ~/3304/38> cp /usr/local/3304/src/lab38main.c .
newuser@csunix ~/3304/38> touch lab38.asm
newuser@csunix ~/3304/38> gcc -m32 -c lab38main.c
newuser@csunix ~/3304/38> nasm -f elf32 -o lab38.o lab38.asm -I/usr/local/3304/include/
newuser@csunix ~/3304/38> gcc -m32 -o lab38 lab38main.o lab38.o /usr/local/3304/src/Along32.o -lm
newuser@csunix ~/3304/38> ./lab38 < 01.dat
-------------------------------------------------------------------------
   Decimal                 Binary                   Octal     Hexadecimal
-------------------------------------------------------------------------
           0  00000000000000000000000000000000  00000000000    00000000
           1  00000000000000000000000000000001  00000000001    00000001
          -1  11111111111111111111111111111111  37777777777    FFFFFFFF
           2  00000000000000000000000000000010  00000000002    00000002
          -2  11111111111111111111111111111110  37777777776    FFFFFFFE
           3  00000000000000000000000000000011  00000000003    00000003
          -3  11111111111111111111111111111101  37777777775    FFFFFFFD
          12  00000000000000000000000000001100  00000000014    0000000C
         -12  11111111111111111111111111110100  37777777764    FFFFFFF4
         123  00000000000000000000000001111011  00000000173    0000007B
        -123  11111111111111111111111110000101  37777777605    FFFFFF85
        1234  00000000000000000000010011010010  00000002322    000004D2
       -1234  11111111111111111111101100101110  37777775456    FFFFFB2E
       12345  00000000000000000011000000111001  00000030071    00003039
      -12345  11111111111111111100111111000111  37777747707    FFFFCFC7
      123456  00000000000000011110001001000000  00000361100    0001E240
     -123456  11111111111111100001110111000000  37777416700    FFFE1DC0
     1234567  00000000000100101101011010000111  00004553207    0012D687
    -1234567  11111111111011010010100101111001  37773224571    FFED2979
    12345678  00000000101111000110000101001110  00057060516    00BC614E
   -12345678  11111111010000111001111010110010  37720717262    FF439EB2
   123456789  00000111010110111100110100010101  00726746425    075BCD15
  -123456789  11111000101001000011001011101011  37051031353    F8A432EB
  2147483647  01111111111111111111111111111111  17777777777    7FFFFFFF
 -2147483647  10000000000000000000000000000001  20000000001    80000001
 -2147483648  10000000000000000000000000000000  20000000000    80000000
-------------------------------------------------------------------------
newuser@csunix ~/3304/38> ./lab38 < 01.dat > my.out
newuser@csunix ~/3304/38> diff 01.out my.out
newuser@csunix ~/3304/38>
```

**Figure 2.** Commands to Assemble, Link, & Run Lab 38