1. Table

	Type	%rdi/edi/di	%rsi/esi/si	Instruction	CF	SF	ZF	OF
(a)	Unsigned	0xFFFE	0x4	addw %di, %si	1	0	0	0
(b)	Unsigned	0xFFFE	0x4	addl %edi, %esi	0	0	0	0
(c)	Signed two's complement	0xFFFE	0x2	addw %di, %si	1	0	1	0
(d)	Signed two's complement	0xFFFE	0x2	addl %edi, %esi	0	0	0	0
(e)	Signed two's complement	0xFFFFFFFF	0x80000000	addl %edi, %esi	1	0	0	1
(f)	Signed two's complement	0xFFFF	-0xFFFF	subl %si, %di	1	0	0	0
(g)	Signed two's complement	0xFFFFFFFE	0x7FFFFFFE	subl %esi, %edi	0	1	0	0
(h)	Unsigned	0xF	0xFF	shlq 64, %rdi	0	0	0	0

2. Table

	Address	Instructions in Hexa	---	Assembly Instructions
(a)	ab1234:	74 08		je **ab123e**
	ab1236:	48 89 d0		mov %rdx,%rax
(b)	abcdef:	7c 07		jl **abcdf8**
	abcdf1:	48 39 f7		cmp %rsi,%rdi
(c)	**0x123443**:	7d 11		jge 0x123456
	0x123445:	48 85 ab		test %rdi,%rdi
(d)	ab01f0:	7f 2f ff ff		jg **ab0123**
	ab01f4:	48 39 d6		mov %rdx,%rsi

3. Assembly Code

```
long reverse_logic(long x, long y, long z)
{
   long result;
   if (x < y){
      if (x > z){
         result = z - x;
      } else {
         result = x + z;
      }
   } else {
      if (y > z){
         result = z - y;
      } else {
         result = y + z;
      }
   }
   return result;
}
```

4. Problem 3.60

A. Which registers hold program values x, n, result, and mask?

- x - %rdx
- n - %ecx
- result - %rax
- mask - %rdx

B. What are the initial values of result and mask?

- result – 0
- mask - 1

C. What is the test condition for mask?

- If mask equals non-zero

D. How does mask get updated?

- masks gets shifted left by least significant byte of n

E. How does result get updated?

- By setting it equal doing bitwise OR with itself and the bitwise AND of x and mask

F. Fill in all the missing parts of the C code.

```
long loop(long x, long n)
{
   long result = 0;
   long mask;

   for (mask = 1; mask != 0; mask = mask << n) {
      result |= (x & mask);
   }
   return result;
}
```

5. C Code

```
long loop_while_hw5(long a, long b)
{
    long result = 1;
    while (!(b >= a)) {
        result = result + (a - b);
        b = b + 1;
    }
    return result;
}
```

6. C Code

```
void switch_hw5(long a, long b, long c, long *dest)
{
   long val;
   switch(a) {
      case 0:
         val = c - a;
         break;
      case 1:
         // c = (b << 4) + b -> b*2^4 + b -> 16 * b + b -> 17 * b
         c = 17 * b;
         /* Fall through */
      case 3:
         val = c ^ 0xFF;
         break;
      case 5:
      case 7:
         val = (b + c) >> 4;
         break;
      default:
         val = a + b;
   }
   return val;
}
```