1)

	Number Type	%rdi/edi/di	%rsi/esi/si	Instruction	CF	SG	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	1
(D.)	Signed Two's Complement	0xFFFE	0x2	addl %edi, %esi	0	0	0	0
(E.)	Signed Two's Complement	0xFFFFFFFF	0x80000000	addl %edi, %esi	0	0	0	0
(F.)	Signed Two's Complement	0xFFFF	-0xFFFF	subl %si, %di	0	0	1	0
(G.)	Signed Two's Complement	0xFFFFFFFE	0x7FFFFFFE	subl %esi, %edi	1	0	0	1
(H.)	Unsigned	0xF	0xFF	shlq 64, %rdi	1	0	1	0

2)

	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
	abcdfl:	48 39 f7		Cmp %rsi, %rdi
(c)	cccccc:	7d 11		Jge 0x123456
	dddddd:	48 85 ab		Test %rdi, %rdi
(d)	ab01f0:	7f 2f ff ff		Jg ab011f
	ab01f4	48 39 d6		Mov %rdx, %rsi

3)

```
reverse_logic:
        cmpq    %rsi, %rdi
        jge     .L3
        cmpq    %rdx, %rdi
        jle     .L4
        movq    %rdx, %rax
        subq    %rdi, %rax
```

```
        ret
.L4:
        leaq    (%rdi,%rdx), %rax
        ret
.L3:
        cmpq    %rdx, %rsi
        jle     .L7
        movq    %rdx, %rax
        subq    %rsi, %rax
        ret
.L7:
        leaq    (%rsi,%rdx), %rax
        ret
```

```
long reverse_logic(long p, long r, long b)
{
    long result;
    if (p <r) {
        if (p > b) {
            result = b- p;
        }
        Else {
            result = b + p;
        }
    }
    Else if (r > b) {
        result = b -r;
    }
    Else {
        Result = b + r;
    }
    return result;
}
```

4)

A. x = %rdi
   N = %esi
   Result = %rax
   mask = %rdx

B. result = 0
   mask = 1

C. mask != 0

D. mask = mask << n

E. long looping(long p, int r) {
   Long result =0;
   Long mask;
   For (mask = 1; mask != 0; mask <<= n) {
      Result |= (p & mask);
   }
   Return result;
}

5)

```
loop_while_hw5:
.LFB0:
        movl    $1, %eax
        jmp     .L2
.L3:
        movq    %rdi, %rdx
        subq    %rsi, %rdx
        addq    %rdx, %rax
        addq    $1, %rsi
.L2:
        cmpq    %rdi, %rsi
        jl      .L3
        rep ret
```

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

6)

a) C Code

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

(b) Assembly Code

```
switch_hw5:
.L3:
        movq    %rdx, %rax
        subq    %rdi, %rax
        ret
```

```
.L5:
        movq    %rsi, %rdx
        salq    $4, %rdx
        addq    %rsi, %rdx
.L6:
        movq    %rdx, %rax
        xorb    $-1, %al
        ret
.L7:
        leaq    (%rdx,%rsi), %rax
        sarq    $4, %rax
        ret
.L2:
        leaq    (%rdi,%rsi), %rax
        ret
```

(c) Jump Table

```
.L4:
        .quad   .L3
        .quad   .L5
        .quad   .L2
        .quad   .L6
        .quad   .L2
        .quad   .L7
        .quad   .L2
        .quad   .L7
```