Solutions to Chapter 14

Review Questions

1. b. Both bits are 1's.
3. c. Bits are different
5. b. Inclusive OR (|)
7. c. Exclusive OR (^)
9. a. afferent
11. b. Inclusive OR (|)

Exercises

13.

a. `0011 0111`
b. `1010 1011`
c. `0000 0010 0011 0111`
d. `1010 0010 0011 0100`

15.

a. `0011 1100`
b. `1010 1010`
c. `0000 0000`
d. `1010 0000`

17.

a. `1100 0011`
b. `0101 0101`
c. `1111 1001`
d. `0100 1011`

19.

a. `0201`
b. `A000`
c. `E200`
d. `0530`

21.

a. `C0AE`
b. `5245`
c. `00A5`
d. `AAC1`

23. mask: `1000 0000` or `0x80` Operator: | (inclusive or)

25. mask: `0000 1010` or `0x0A` Operator: ^ (exclusive or)

27. mask: `0001 0100` or `0x14`

29.

Binary: 0000 1000 0100 1011
Hexadecimal: `084B`

31.

a. Subtraction in modulo 2 arithmetic is the same as addition. When we add the two polynomials, we get

$x^6 + x^5 + x^4$

b.

First polynomial:	0x25
Second polynomial:	0x55
Result:	0x70

c. The result is the same because $x^6 + x^5 + x^4$ is the same as 0x70.

Problems

33. See Program 14-1.

Program 14-1 Solution to Problem 33

```
/* ======================= setOne =======================
   Set a specified bit in a short integer to 1.
      Pre   given reference parameter to number and the
            position bit to be set to one
      Post  bit set to one in calling program
            if position in invalid, number is unchanged
*/
void setOne (uint16_t *number, int  pos)
{
// Local Declarations
   uint16_t mask = 1 << pos;

// Statements
   *number |= mask;
   return;
}  // setOne
```

35. See Program 14-2.

Program 14-2 Solution to Problem 35

```
/* ======================= flipBit ======================
   Flip a specified bit in a short integer.
      Pre   given reference parameter to number and
            the position bit to be set to flipped
      Post  bit flipped in calling program
            if position in invalid, number is unchanged
*/
void flipBit (uint16_t* number, int pos)
{
// Local Definitions
   uint16_t mask = 1 << pos;

// Statements
   *number = *number ^ mask;
   return;
}  // flipBit
```

37. See Program 14-3.

Program 14-3 Solution to Problem 37

```
/* ====================== addPoly =======================
   This function adds two polynomials.
```

Program 14-3 Solution to Problem 37 (continued)

```
      Pre   given two polynomials
      Post  sum returned
*/
uint16_t addPoly (uint16_t poly1, uint16_t poly2)
{
   return (poly1 ^ poly2);
}  // addPoly
```

39. See Program 14-4.

Program 14-4 Solution to Problem 39

```
/* ==================== rightMost ====================
   Returns rightmost byte in an integer.
      Pre   given an integer
      Post  rightmost byte returned
*/
int rightMost (uint32_t num1)
{
   return (num1 & 0x0000000F) ;
}  // rightMost
```

41. See Program 14-5.

Program 14-5 Solution to Problem 41

```
/* ================= compRightmost ==================
   Returns integer with leftmost byte complemented.
      Pre   given an integer
      Post  returned with rightmost byte complemented
*/
uint32_t compRightmost (uint32_t num1)
{
   return (num1 ^ 0x0000000F);
}  // compRightmost
```

43. See Program 14-6.

Program 14-6 Solution to Problem 43

```
/* ================= changeRightmost =================
   Changes rightmost byte in an integer.
      Pre   given an integer and the new byte
      Post  integer returned with new rightmost digit
*/
uint32_t changeRightmost (uint32_t num1, int digit)
{
   return ((num1 &  0xFFFFFFF0) | digit) ;
}  // changeRightmost
```