Lab 2

Exercise 1

For input:

- 1 output 3
- 2 output 6
- 3 output 9
- 4 output 12

The function $f(x)$ returns three times x, so $f(x) = 3x$.

Exercise 2

For input:

- 4, 3 output 3
- 5, 1 output 1
- 1, 5 output 1
- 6, 10 output 6
- 10, 6 output 6
- 5, 5 output 5

The function $f(x, y)$ returns the minimum value, so either x if $x \leq y$ or y if $y \leq x$. $f(x) = \min(x, y)$.

Exercise 3

For inputs:

- 6, 15 output 3
- 6, 60 output 6

- 5, 60 output 5
- 5, 12 output 1
- 11, 12 output 1
- 0, 0 output 0
- 0, 5 infinite loop

The program implements Euclid's algorithm for finding greatest common divisor. When both inputs are set to 0's the program outputs 0, but when only one of the inputs is equal to 0 it goes into an infinite loop. To solve this we could first check whether one of the inputs is equal to zero and output the other value.

The modified code:

```
section .text
    MOV     EAX, 0          ; first argument
    MOV     EBX, 0         ; second argument
    CMP     EAX, 0
    JE      a_zero
    CMP     EBX, 0
    JE      b_zero
while:
    CMP     EAX, EBX
    JE      end
    JL      else
    SUB     EAX, EBX
    JMP     while
else:
    SUB     EBX, EAX
    JMP     while
end:
    HLT
a_zero:
    MOV     EAX, EBX
    HLT
b_zero:
```

```
    MOV     EAX, EAX
    HLT
```

Exercise 4

For input:

- 1234, 10 output 10
- 2222, 10 output 8
- 4321, 10 output 10
- 12, 10 output 3
- 12, 2 output 2
- 56, 2 output 3

This program adds all the digits of the number in the `EAX` register written in a base given in `EBX`.

Exercise 5

```
section .text
    MOV     ECX, 0
    MOV     EAX, 1
loop:
    CMP     ECX, 1
    JE      end
    JL      zero
    IMUL    ECX
    SUB     ECX, 1
    JMP     loop
end:
    HLT
zero:
```

```
    MOV     EAX, 1
    HLT
```

Exercise 6

```
section .text
    MOV     ECX, 0
    MOV     EDX, 0
    MOV     EAX, 1
loop:
    CMP     EDX, 0
    JE      end
    MOV     EBX, EDX
    IMUL    ECX
    MOV     EDX, EBX
    SUB     EDX, 1
    JMP     loop
end:
    HLT
```

Exercise 7

```
section .text
    MOV     ECX, 101
    MOV     EBX, 2
loop:
    CMP     EBX, ECX
    JE      prime
    MOV     EAX, ECX
    IDIV    EBX
    CMP     EDX, 0
    JE      not_prime
    ADD     EBX, 1
```

```
    JMP loop
prime:
    MOV     EAX, 1
    HLT
not_prime:
    MOV     EAX, 0
    HLT
```

Exercise 8

```
section .text
    MOV     ECX, 9
    MOV     EAX, 6
    MOV     EBX, 8
    IMUL    ECX
    SUB     EAX, EBX
    IMUL    ECX
    MOV     EDX, 5
    ADD     EAX, EDX
    HLT
```

Exercise 9

```
section .text
    MOV     ECX, 50
    MOV     EAX, ECX
    MOV     EBX, 0
    IDIV    3
    CMP     EDX, 0
    JE      div_3
jump_back:
    MOV     EAX, ECX
    IDIV    5
```

```
    CMP     EDX, 0
    JE      div_5
    CMP     EBX, 0
    JE      not_both
    JMP     end
not_both:
    MOV     EBX, ECX
    JMP     end
div_3:
    ADD     EBX, 3
    JMP     jump_back
div_5:
    ADD     EBX, 5
    JMP     end
end:
    MOV     EAX, EBX
    HLT
```

Exercise 10

Task 1

`ADE + 4D5 = FB3`

`9B6 - 399 = 61D`

Task 2

Converting `3D4` (or `0011 1101 0100` in binary) to negative: `-3D4` is `FC2C` in two's complement (or `1111 1100 0010 1100` in binary).

Task 3

`FFD4` has a decimal value of `-44`.

Task 4

`255 = FF`

`1024 = 400`

Task 5

`F0A1 AND B12C = B020`