

Project II – Test Scenarios (Results)

Name: Fernando Martinez

Prog. Language: Ocaml

Project 2 assignment

Assignment Instructions

Using functional programming:

- no side effects
- no mutations

create an emulator for the binary logic and arithmetic operations of a CPU

Deliverables:

- Source code
 - No mutable variables
 - Looping to be done by tail recursion
 - All functions to be done on 'binary lists'
 - Conversions to and from integer to 'binary lists' to be done by implementing the algorithms shared
- Example output
 - Show examples for each of the required functions
 - Additionally, for addition show the following:
 - 128 + 128
 - 127 + 1
 - -5 + -5
 - Additionally, for subtraction show the following:
 - -128 + 0
 - 10 - 11

Grading:

- Both deliverables included
- Code embodies "functional programming"
- Code makes good use of reusable functions
- Output is correct

Build an emulator for some simple computer arithmetic and logical operations:

- Addition, subtraction
- Bit-wise and, or, xor, not

All operations will be done on an 8-bit value

- Arithmetic operations will be on 8-bit signed integers: -128 to 127
- All logical operations will be on 8-bit unsigned values: 0x0 to 0xFF

Input / Output

Input

You will read in user input to provide arithmetic and logical operations:

- Equations for arithmetic operations will be provided using decimal numbers. E.g.:
 - 5 + 4
 - -7 + 83
 - 101 - 121
 - 45 - -6
- Entries for logical operations will be interpreted as hexadecimal. E.g.:
 - 0x48 AND 0x84
 - 48 AND 84
 - FF XOR AA

Output

The results of each equation provided as input should be printed out along with the equations

- Results for arithmetic equations should be in decimal
- Results for logical equations should be in hexadecimal
- Show edge cases and overflows

```
BinaryLogicANDArithmeticOperator.ml
12 let decimal_to_binary num =
14   let rec convert remaining powers acc =
16     let bit = if remaining >= power then 1 else 0 in
19     let new_remaining = if bit = 1 then remaining - power else remaining in
20     convert new_remaining rest (bit :: acc)
21   in
22   convert n [128: 64: 32: 16: 8: 4: 2: 1] []

Problems Output Debug Console Terminal Ports
> ocaml BinaryLogicANDArithmeticOperator.ml
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 23
Enter second decimal value: -23

>>> 23 + -23 <<<
[00010111] = 23
+[11101001] = -23
-----
[00000000] = 0

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 30
Enter second decimal value: 7

>>> 30 + 7 <<<
[00011110] = 30
+[00000111] = 7
-----
[00100101] = 37

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 30
Enter second decimal value: 23

>>> 30 - 23 <<<
[00011110] = 30
+[11101001] = -23
-----
[00000111] = 7

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)?
```

Example: -23 decimal to binary

1. 23 -> 00010111
2. NOT -> 11101000
3. ADD 1 -> 11101001

4. Check $23 + (-23) = 0$?

00010111
+11101001

00000000

ons

Example: 30 + 7

1. 30 -> 00011110
2. 7 -> 00000111
3. --- -----
4. 37 -> 00100101

ns

Example: 30 - 23

1. 23 -> 00010111
 2. NOT -> 11101000
 3. ADD 1 -> 11101001
- *****
1. 30 -> 00011110
 2. -23 -> 11101001
 3. --- -----
 4. 7 -> 00000111

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? NOT
Enter Hex value: 0xF1
Result of NOT F1 = [00001110] = 0E
```

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? and
Enter first Hex value: 0xA5
Enter second Hex value: 0xF1
```

```
>>> 0xA5 AND 0xF1 <<<
[10100101] = 0xA5
[11110001] = 0xF1
-----
[10100001] = 0xA1
```

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? OR
Enter first Hex value: 0xA5
Enter second Hex value: 0xF1
```

```
>>> 0xA5 OR 0xF1 <<<
[10100101] = 0xA5
[11110001] = 0xF1
-----
[11110101] = 0xF5
```

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? XOR
Enter first Hex value: 0xA5
Enter second Hex value: 0xF1
```

```
>>> 0xA5 XOR 0xF1 <<<
[10100101] = 0xA5
[11110001] = 0xF1
-----
[01010100] = 0x54
```

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? █
```

Project #2 – functional programming

Methods

Logical operations

AND:

Do the following bit-wise,
order doesn't matter

1. 0 AND 0 -> 0
2. 1 AND 0 -> 0; 0 AND 1 -> 0
3. 1 AND 1 -> 1

Example: 0xA5 AND 0xF1

1. 0xA5 -> 10100101
2. 0xF1 -> 11110001
3. -----
4. 0xA1 -> 10100001

OR:

Do the following bit-wise,
order doesn't matter

1. 0 OR 0 -> 0
2. 1 OR 0 -> 1; 0 OR 1 -> 1
3. 1 OR 1 -> 1

Example: 0xA5 OR 0xF1

1. 0xA5 -> 10100101
2. 0xF1 -> 11110001
3. -----
4. 0xF5 -> 11110101

NOT:

Flip all bits

1. NOT 0 -> 1
2. NOT 1 -> 0

Example: NOT 0xF1

1. 0xF1 -> 11110001
2. -----
3. 0x0E -> 00001110

XOR:

Do the following bit-wise,
order doesn't matter

1. 0 XOR 0 -> 0
2. 1 XOR 0 -> 1; 0 XOR 1 -> 1
3. 1 XOR 1 -> 0

Example: 0xA5 XOR 0xF1

1. 0xA5 -> 10100101
2. 0xF1 -> 11110001
3. -----
4. 0x56 -> 01010100

This is supposed to be **0x54**,
not **0x56**.

Verified using:

https://www.binaryconvert.com/convert_signed_char.html

```
BinaryLogicANDArithmeticOperator.ml
12 let decimal_to_binary num =
14   let rec convert remaining powers acc =
16     let bit = if remaining >= power then 1 else 0 in
19     let new_remaining = if bit = 1 then remaining - power else remaining in
20     convert new_remaining rest (bit :: acc)
21   in
22   convert n [128: 64: 32: 16: 8: 4: 2: 1] []

Problems Output Debug Console Terminal Ports

o > ocaml BinaryLogicANDArithmeticOperator.ml
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 5
Enter second decimal value: 4

>>> 5 + 4 <<<
[00000101] = 5
+[00000100] = 4
-----
[00001001] = 9

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: -7
Enter second decimal value: 83

>>> -7 + 83 <<<
[11111001] = -7
+[01010011] = 83
-----
[01001100] = 76

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 101
Enter second decimal value: 121

>>> 101 - 121 <<<
[01100101] = 101
+[10000111] = -121
-----
[11101100] = -20

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 45
Enter second decimal value: -6

>>> 45 - -6 <<<
[00100101] = 45
+[00000110] = 6
-----
[00101011] = 51

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)?
```

Project #2 – functional programming

Input / Output

Input

You will read in user input to provide arithmetic and logical operations:

- Equations for arithmetic operations will be provided using decimal numbers. E.g.:
 - 5 + 4
 - -7 + 83
 - 101 - 121
 - 45 - -6
- Entries for logical operations will be interpreted as hexadecimal. E.g.:
 - 0x48 AND 0x84
 - 48 AND 84
 - FF XOR AA

Output

The results of each equation provided as input should be printed out along with the equations

- Results for arithmetic equations should be in decimal
- Results for logical equations should be in hexadecimal
- Show edge cases and overflows

```
BinaryLogicANDArithmeticOperator.ml
1  (* utility functions *)
2  (* padding function to ensure 8 bits *)
3  let pad_to_8 lst =
4    let len = List.length lst in
5    if len < 8 then

> ocaml BinaryLogicANDArithmeticOperator.ml
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? and
Enter first Hex value: 48
Enter second Hex value: 84

>>> 0x48 AND 0x84 <<<
[01001000] = 0x48
[10000100] = 0x84
-----
[00000000] = 0x00

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? and
Enter first Hex value: 0x48
Enter second Hex value: 0x84

>>> 0x48 AND 0x84 <<<
[01001000] = 0x48
[10000100] = 0x84
-----
[00000000] = 0x00

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? xor
Enter first Hex value: FF
Enter second Hex value: AA

>>> 0xFF XOR 0xAA <<<
[11111111] = 0xFF
[10101010] = 0xAA
-----
[01010101] = 0x55

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)?
```

Project #2 – functional programming

Input / Output

Input

You will read in user input to provide arithmetic and logical operations:

- Equations for arithmetic operations will be provided using decimal numbers. E.g.:
 - 5 + 4
 - -7 + 83
 - 101 – 121
 - 45 - -6
- Entries for logical operations will be interpreted as hexadecimal. E.g.:
 - 0x48 AND 0x84
 - 48 AND 84
 - FF XOR AA

Output

The results of each equation provided as input should be printed out along with the equations

- Results for arithmetic equations should be in decimal
- Results for logical equations should be in hexadecimal
- Show edge cases and overflows

My code work for both hexadecimal forms (e.g., 0x48 | 48)

BinaryLogicANDArithmeticOperator.ml

```
25 let binary_to_decimal binary =  
27 let rec convert bits powers acc =  
28   match (bits, powers) with  
29   | [], _ | _, [] -> acc  
30   | bit :: rest_bits, power :: rest_powers ->  
31     let value = if bit = 1 then power else 0 in  
32     convert rest_bits rest_powers (acc + value)  
33 in
```

Problems Output Debug Console Terminal Ports

```
> ocaml BinaryLogicANDArithmeticOperator.ml  
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? not  
Enter Hex value: 55  
Result of NOT 55 = [10101010] = AA  
  
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add  
Enter first decimal value: 23  
Enter second decimal value: 68  
  
>>> 23 + 68 <<<  
[00010111] = 23  
+[01000100] = 68  
-----  
[01011011] = 91  
  
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? and  
Enter first Hex value: A4  
Enter second Hex value: A5  
  
>>> 0xA4 AND 0xA5 <<<  
[10100100] = 0xA4  
[10100101] = 0xA5  
-----  
[10100100] = 0xA4
```

Project #2 – functional programming

Sample
session

```
Enter the operation you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT): NOT  
Enter Hex value: 55  
Result of NOT 55 = [1; 0; 1; 0; 1; 0; 1; 0] = AA  
  
Enter the operation you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT): ADD  
Enter first decimal value: 23  
Enter second decimal value: 68  
[0; 0; 0; 1; 0; 1; 1; 1] = 23  
ADD [0; 1; 0; 0; 0; 1; 0; 0] = 68  
-----  
[0; 1; 0; 1; 1; 0; 1; 1] = 91  
  
Enter the operation you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT): AND  
Enter first Hex value: A4  
Enter second Hex value: A5  
[1; 0; 1; 0; 0; 1; 0; 0] = A4  
AND [1; 0; 1; 0; 0; 1; 0; 1] = A5  
-----  
[1; 0; 1; 0; 0; 1; 0; 0] = A4  
  
Enter the operation you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT): QUIT
```

```
BinaryLogicANDArithmeticOperator.ml
25 let binary_to_decimal binary =
27   let rec convert bits powers acc =
28     match (bits, powers) with
29     | [], _ | _, [] -> acc
30     | bit :: rest_bits, power :: rest_powers ->
```

Problems Output Debug Console Terminal Ports

o > ocaml BinaryLogicANDArithmeticOperator.ml
What operation do you want to perform (NOT,add
Enter first decimal value: 128
Enter second decimal value: 128
Error: Values must be between -128 and 127

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 127
Enter second decimal value: 1

>>> 127 + 1 <<<
[01111111] = 127
+[00000001] = 1

[10000000] = -128

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: -5
Enter second decimal value: -5

>>> -5 + -5 <<<
[11111011] = -5
+[11111011] = -5

[11110110] = -10

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: -128
Enter second decimal value: 0

>>> -128 - 0 <<<
[10000000] = -128
+[00000000] = 0

[10000000] = -128

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 10
Enter second decimal value: 11

>>> 10 - 11 <<<
[00001010] = 10
+[11110101] = -11

[11111111] = -1

Project 2 assignment

Assignment Instructions

Using functional programming:

- no side effects
- no mutations

create an emulator for the binary logic and arithmetic operations of a CPU

Deliverables:

- Source code
 - No mutable variables
 - Looping to be done by tail recursion
 - All functions to be done on 'binary lists'
 - Conversions to and from integer to 'binary lists' to be done by implementing the algorithms shared

• Example output

- Show examples for each of the required functions
- Additionally, for addition show the following:

- 128 + 128
- 127 + 1
- -5 + -5

• Additionally, for subtraction show the following:

- -128 + 0
- 10 - 11

Grading:

- Both deliverables included
- Code embodies "functional programming"
- Code makes good use of reusable functions
- Output is correct

Stress Testing Operations

```
BinaryLogicANDArithmeticOperator.ml
25 let binary_to_decimal binary =
27   let rec convert bits powers acc =
28     match (bits, powers) with
    <-->
Problems Output Debug Console Terminal Ports

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 100
Enter second decimal value: 50

>>> 100 + 50 <<<
[01100100] = 100
+[00110010] = 50
-----
[10010110] = 106

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? add
Enter first decimal value: 50
Enter second decimal value: 50

>>> 50 + 50 <<<
[00110010] = 50
+[00110010] = 50
-----
[01100100] = 100

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 10
Enter second decimal value: 11

>>> 10 - 11 <<<
[00001010] = 10
+[-11110101] = -11
-----
[11111111] = -1

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 10
Enter second decimal value: -11

>>> 10 - -11 <<<
[00001010] = 10
+{00001011} = 11
-----
[00010101] = 21

What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? sub
Enter first decimal value: 100
Enter second decimal value: 50

>>> 100 - 50 <<<
[01100100] = 100
+[-11001110] = -50
-----
[00110010] = 50
```

 $100 + 50$

50 + 50

10 - 11

10 - -11

100 - 50

Other edge cases

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? NOT
Enter Hex value: 00
Result of NOT 00 = [11111111] = FF
```

```
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? NOT
Enter Hex value: FF
Result of NOT FF = [00000000] = 00
```

NOT 0x00

NOT 0xFF

```
• > ocaml BinaryLogicANDArithmeticOperator.ml
What operation do you want to perform (NOT, OR, AND, XOR, ADD, SUB or QUIT)? AND
Enter first Hex value: 0xAA
Enter second Hex value: 0x00
```

```
>>> 0xAA AND 0x00 <<<
[10101010] = 0xAA
[00000000] = 0x00
-----
[00000000] = 0x00
```

0xAA AND 0x00