605.611 - Foundations of Computer Architecture

Assignment 01

Sabbir Ahmed

February 3, 2021

1. Convert the following from decimal to binary. Express numbers in 16 bits.

   (a) 14

$$\begin{array}{r} 7 \\ 2\overline{)14} \\ \underline{14} \\ 0 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)7} \\ \underline{6} \\ 1 \end{array} \qquad \begin{array}{r} 1 \\ 2\overline{)3} \\ \underline{2} \\ 1 \end{array} \qquad \begin{array}{r} 0 \\ 2\overline{)1} \\ \\ \\ \end{array}$$

$\Rightarrow (0000\ 0000\ 0000\ 1110)_2$

   (b) 97

$$\begin{array}{r} 48 \\ 2\overline{)97} \\ \underline{8} \\ 17 \\ \underline{16} \\ 1 \end{array} \qquad \begin{array}{r} 24 \\ 2\overline{)48} \\ \underline{4} \\ 08 \\ \underline{8} \\ 0 \end{array} \qquad \begin{array}{r} 12 \\ 2\overline{)24} \\ \underline{2} \\ 04 \\ \underline{4} \\ 0 \end{array} \qquad \begin{array}{r} 6 \\ 2\overline{)12} \\ \underline{12} \\ 0 \end{array} \qquad \begin{array}{r} 3 \\ 2\overline{)6} \\ \underline{6} \\ 0 \end{array} \qquad \begin{array}{r} 1 \\ 2\overline{)3} \\ \underline{2} \\ 1 \end{array} \qquad \begin{array}{r} 0 \\ 2\overline{)1} \\ \\ \\ \end{array}$$

$\Rightarrow (0000\ 0000\ 0110\ 0001)_2$

(c) 947

$$\begin{array}{r} 473 \\ 2\,\overline{)947} \\ \underline{8} \\ 14 \\ \underline{14} \\ 07 \\ \underline{6} \\ 1 \end{array} \quad \begin{array}{r} 236 \\ 2\,\overline{)473} \\ \underline{4} \\ 07 \\ \underline{6} \\ 13 \\ \underline{12} \\ 1 \end{array} \quad \begin{array}{r} 118 \\ 2\,\overline{)236} \\ \underline{2} \\ 03 \\ \underline{2} \\ 16 \\ \underline{16} \\ 0 \end{array} \quad \begin{array}{r} 59 \\ 2\,\overline{)118} \\ \underline{10} \\ 18 \\ \underline{18} \\ 0 \end{array} \quad \begin{array}{r} 29 \\ 2\,\overline{)59} \\ \underline{4} \\ 19 \\ \underline{18} \\ 1 \end{array}$$

$$\begin{array}{r} 14 \\ 2\,\overline{)29} \\ \underline{2} \\ 09 \\ \underline{8} \\ 1 \end{array} \quad \begin{array}{r} 7 \\ 2\,\overline{)14} \\ \underline{14} \\ 0 \end{array} \quad \begin{array}{r} 3 \\ 2\,\overline{)7} \\ \underline{6} \\ 1 \end{array} \quad \begin{array}{r} 1 \\ 2\,\overline{)3} \\ \underline{2} \\ 1 \end{array} \quad \begin{array}{r} 0 \\ 2\,\overline{)1} \end{array}$$

$\Rightarrow (0000\ 0011\ 1011\ 0011)_2$

(d) 32768 (What is special about this number that makes it easy to calculate?)

32768 is a power of 2 i.e. 2^{15}. This decimal integer is very easy to calculate since only the 15th bit is active and therefore $(32768) = (1000\ 0000\ 0000\ 0000)_2$.

(e) 32767 (What is special about this number and the last one that makes it easy to calculate?)

32767 is 1 less than 32768 which is a power of 2 i.e. $2^{15} - 1$. This decimal integer is very easy to calculate since all the 15 bits are active and therefore $(32767) = (0111\ 1111\ 1111\ 1111)_2$.

2. Convert the following numbers from binary to decimal

(a) 0000 0000 0000 1010

$$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 10$$

(b) 0000 0000 1001 1010

$$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 154$$

(c) 0000 0001 1011 1100

$$1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 444$$

3. Convert the following numbers from binary to hexadecimal

(a) 0000 0000 0000 1010

$$1010 = A$$

(b) 0000 0000 1001 1010

$$\begin{aligned} 1001 &= 9 \\ 1010 &= A \\ &= 9A \end{aligned}$$

(c) 0000 0001 1011 1100

$$\begin{aligned} 0001 &= 1 \\ 1011 &= B \\ 1100 &= C \\ &= 1BC \end{aligned}$$

4. Convert the following numbers from hex to binary.

   (a) 0x000C

$$0 \cdot 16^3 + 0 \cdot 16^2 + 0 \cdot 16^1 + 12 \cdot 16^0 = 12$$

   (b) 0x07AD

$$7 \cdot 16^3 + 10 \cdot 16^2 + 0 \cdot 16^1 + 13 \cdot 16^0 = 1965$$

   (c) 0xA49F

$$10 \cdot 16^3 + 4 \cdot 16^2 + 9 \cdot 16^1 + F \cdot 16^0 = 42143$$

5. Convert the ASCII string "Java.2" to binary. You must include the character for the period.

   Answer: Looking up the ASCII values of the string:

$$J = 74$$

$$a = 97$$

$$v = 118$$

$$. = 46$$

$$2 = 50$$

And converting the decimals to binary,

$$\Rightarrow 74 = \begin{array}{rrrrrrr}
37 & 18 & 9 & 4 & 2 & 1 & 0 \\
2\overline{)74} & 2\overline{)37} & 2\overline{)18} & 2\overline{)9} & 2\overline{)4} & 2\overline{)2} & 2\overline{)1} \\
\underline{6} & \underline{2} & \underline{18} & \underline{8} & \underline{4} & \underline{2} & \\
14 & 17 & 0 & 1 & 0 & 0 & \\
\underline{14} & \underline{16} & & & & & \\
0 & 1 & & & & &
\end{array}$$

$\Rightarrow (0100\ 1010)_2$

$\Rightarrow 97 = (0110\ 0001)_2$ (from part 1B)

$$\Rightarrow 118 = \begin{array}{rrrrrrr}
59 & 29 & 14 & 7 & 3 & 1 & 0 \\
2\overline{)118} & 2\overline{)59} & 2\overline{)29} & 2\overline{)14} & 2\overline{)7} & 2\overline{)3} & 2\overline{)1} \\
\underline{10} & \underline{4} & \underline{2} & \underline{14} & \underline{6} & \underline{2} & \\
18 & 19 & 09 & 0 & 1 & 1 & \\
\underline{18} & \underline{18} & \underline{8} & & & & \\
0 & 1 & 1 & & & &
\end{array}$$

$\Rightarrow (0111\ 0110)_2$

$$\Rightarrow 46 = \begin{array}{rrrrrr}
23 & 11 & 5 & 2 & 1 & 0 \\
2\overline{)46} & 2\overline{)23} & 2\overline{)11} & 2\overline{)5} & 2\overline{)2} & 2\overline{)1} \\
\underline{4} & \underline{2} & \underline{10} & \underline{4} & \underline{2} & \\
06 & 03 & 1 & 1 & 0 & \\
\underline{6} & \underline{2} & & & & \\
0 & 1 & & & &
\end{array}$$

$\Rightarrow (0010\ 1110)_2$

$$\Rightarrow 50 = \begin{array}{r} 25 \\ \hline 2\,)\,50 \\ 4 \\ \hline 10 \\ 10 \\ \hline 0 \end{array} \quad \begin{array}{r} 12 \\ \hline 2\,)\,25 \\ 2 \\ \hline 05 \\ 4 \\ \hline 1 \end{array} \quad \begin{array}{r} 6 \\ \hline 2\,)\,12 \\ 12 \\ \hline 0 \end{array} \quad \begin{array}{r} 3 \\ \hline 2\,)\,6 \\ 6 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ \hline 2\,)\,3 \\ 2 \\ \hline 1 \end{array} \quad \begin{array}{r} 0 \\ \hline 2\,)\,1 \end{array}$$

$$\Rightarrow (0011\ 0010)_2$$

Therefore, the ASCII string "Java.2" converted to binary is

$$(01001010\ 01100001\ 01110110\ 01100001\ 00101110\ 00110010)_2$$

6. Is the binary number 1001 1100 0111 1011 1101 1001 odd or even?

   Answer: Since the binary number ends with a 1, it indicates that $2^0 = 1$, an odd integer, was added to powers of 2. Since adding an odd integer to an even integer makes it an odd integer, this binary number is odd.

7. What is the base 10 value of the following two's complement numbers.

   (a) 01001 0111

$$\begin{aligned}(10010111)_2 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= +151\end{aligned}$$

   (b) 1101 1000

$$\begin{aligned}\sim (1011000)_2 &= (00100111)_2 \\ (00100111)_2 + 1_2 &= (00101000)_2 \\ &= 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\ &= -40\end{aligned}$$

8. Add the following numbers in 4-bit two's complement format. Specify if there is an overflow or not.

(a) 5 + 3

$$
\begin{aligned}
&(0101)_2 + (0011)_2 \\
&= \begin{array}{ccccc} & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 0 \end{array} \\
&= (1000)_2 \\
&= 8
\end{aligned}
$$

(b) 7 + 5

$$
\begin{aligned}
&(0111)_2 + (0101)_2 \\
&= \begin{array}{ccccc} & 0 & 1 & 1 & 1 \\ + & 0 & 1 & 0 & 1 \\ \hline & 1 & 1 & 0 & 0 \end{array} \\
&= (1100)_2 \\
&= 12
\end{aligned}
$$

(c) 7 + (-5)

$$
\begin{aligned}
&(0111)_2 + (1011)_2 \\
&= \begin{array}{ccccc} & 0 & 1 & 1 & 1 \\ + & 1 & 0 & 1 & 1 \\ \hline & 0 & 0 & 1 & 0 \end{array} \\
&= (0010)_2 \\
&= 2
\end{aligned}
$$

The sum yields a carry over of 1_2 and therefore there is an overflow.

(d) (-7) + (-5)

$$\begin{aligned}
&(1001)_2 + (1011)_2 \\
&= \begin{array}{r} \quad 1\ 0\ 0\ 1 \\ + \quad 1\ 0\ 1\ 1 \\ \hline 0\ 1\ 0\ 0 \end{array} \\
&= \sim (0100)_2 + (0001)_2 \\
&= (1100)_2 \\
&= -12
\end{aligned}$$

The sum yields a carry over of 1_2 and therefore there is an overflow.

(e) (-5) + (-3)

$$\begin{aligned}
&(1011)_2 + (1101)_2 \\
&= \begin{array}{r} \quad 1\ 0\ 1\ 1 \\ + \quad 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 0 \end{array} \\
&= \sim (1000)_2 + (0001)_2 \\
&= (1000)_2 \\
&= -8
\end{aligned}$$

The sum yields a carry over of 1_2 and therefore there is an overflow.

9. Using only the bitwise complement operation and addition, implement a function in the language of your choice to calculate the negative value of an int. The bitwise complement is the ~ operator in Java and C/C++. The program should be properly formatted and documented, including:

   (a) A preamble statement that includes the name of the program, the purpose of the program, the author, and the date that it was written. The author attribute in Java can be documented using the @author Javadoc attribute.

(b) Functions should include a purpose and any parameters and return values. The parameters and return values can be documented a using the @param and @return Javadoc attributes.

(c) Proper style including variable naming, indentation, placement of brackets, etc.

(d) Students unfamiliar with proper style should refer to:
https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Answer: See the attached C file "snippets.c" lines 24-38 for the documentation and implementation of this function.

10. Implement the following multiplication problems using only bit-shift operations and addition.

Answer:

(a) 7 * 4

$$\begin{aligned}
7 \cdot 4 &= (4+3) \cdot 4 \\
&= ((00100)_2 + (00011)_2) \cdot (00100)_2 \\
&= ((00100)_2 << 2) + ((00011)_2 << 2) \\
&= (10000)_2 + (01100)_2 \\
&= \begin{array}{cccccc} & 1 & 0 & 0 & 0 & 0 \\ + & 0 & 1 & 1 & 0 & 0 \\ \hline & 1 & 1 & 1 & 0 & 0 \end{array} \\
&= (11100)_2 \\
&= 28
\end{aligned}$$

(b) 8 * 3

$$\begin{aligned}
8 \cdot 3 &= 8 \cdot (8 + -(3)) \\
&= (1000)_2 \cdot ((1000)_2 + (1101)_2) \\
&= ((1000)_2 << 3) + ((1101)_2 << 3) \\
&= (1000000)_2 + (1101000)_2 \\
&= \begin{array}{cccccccc} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ + & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \hline & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} \\
&= \sim (101000)_2 + (0001)_2 \\
&= (011000)_2 \\
&= 24
\end{aligned}$$

(c) 9 * 13

$$\begin{aligned}
9 \cdot 13 &= (8 + 1) \cdot (13) \\
&= ((1101)_2 \cdot (0001)_2) + ((1101)_2 \cdot (1000)_2) \\
&= (1101)_2 + (1101)_2 << 3 \\
&= (1101)_2 + (1101000)_2 << 3 \\
&= \begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array} \\
&= (01110101)_2 \\
&= 117
\end{aligned}$$

11. Implement the division of 13/4 using bit shift operations. What happens to the remainder. What does this tell you about the efficient implementation integer division? Does this match what you know about integer division in Java or C/C++?

Answer:

$$\begin{aligned} \frac{13}{4} &= \frac{(1101)_2}{(0100)_2} \\ &= (1101)_2 >> 2 \\ &= (0011)_2 \\ &= 3 \end{aligned}$$

The remainder $(0001)_2$ is discarded, which is the expected behavior for integer division in most programming languages.

12. Can 14/3 be implemented using simply bit shift operations? Why or why not?

    Answer: No, computing 14/3 requires more operations besides simple bit shifts. Division in binary can be done by shifting bits to the right only if the divisor is a power of 2.

13. Write a program in Java or C/C++ to convert a character to upper case, regardless of whether it started as upper of lower case. This is simply OR'ing the byte for the character 0x20. The bitwise OR operation in Java or C/C++ is "|". The same rules of program style apply to the program as for Problem 9.

    Answer: See the attached C file "snippets.c" lines 40-56 for the documentation and implementation of this function.

14. Write a program in Java or C/C++ to convert a character to lower case, regardless of whether it started as upper of lower case. This is simply AND'ing the byte for the character 0xCF. The bitwise AND operation in Java or C/C++ is "&". The same rules of program style apply to the program as for Problem 9.

    Answer: See the attached C file "snippets.c" lines 58-74 for the documentation and implementation of this function.

15. Write a program in Java or C/C++ to convert a numeric value in a String to a valid integer. You must use the following algorithm, you may not use functions like

"parseInt" or any other Java or C/C++ (or any language) libraries. The same rules of program style apply to the program as for Problem 9. The algorithm for this operation is:

```
int toInt(char[] inputNumber) {
  retval = 0;
  foreach character c in the inputNumber {
    retval = retval * 10;
    retval = retval + c - '0';
  }
  return retval;
}
```

Answer: See the attached C file "snippets.c" lines 76-108 for the documentation and implementation of this function.