Department of Computing Science and Mathematics
University of Stirling

CSCU9V4 Systems I – Tutorial 2

Week beginning 12 February 2018

1. Convert the following decimal numbers into binary and hex, using two's complement notation with 8 bits:

10, -14, -29, -31

Now perform the following operations, in both binary and hexadecimal:

-14 + 10, -29 - 31

Convert the results back to decimal to check you obtained the right answers.

2. Images are often stored as sets of 24 bit values, where each 24 bit value codes the amount of red, green and blue at each picture element, each in 8 bits. Consider a digital photograph which is 1280 by 1024 picture elements in size.

a) How many bytes does it occupy?
b) How many ASCII characters could that hold (assume that each 7-bit character is held in an 8 bit byte)?
c) How many Unicode characters could it hold (state any assumptions)?
d) (Is a "picture worth a thousand words" – should that be more or less?)

3. A computer used in a control application needs to generate a byte with bits 1 and 6 set, and the others all 0. How might you generate such a byte?

Another part of the application needs to set bit 4 if and only if bit 2 is set, and to clear bit 4 if bit 2 is not set. The other bits of the byte are not to be altered. How might this be achieved using masking and bitwise operators and an `if` statement?

4. In the lectures, it was stated that multiplication could be achieved through repeated addition. Clearly, this is very inefficient (consider multiplying 1000 by 1000: you would need to perform 999 additions). Consider how one might do this more efficiently:

- We will multiply 2 positive 8 bit binary integers
- Think of them as binary strings of length (8 bits) and placing the result into a larger (say 16 bit) word (why?)
- Assume that the ALU has a "shift left" operation, which moves all the bits one place to the left, inserting a 0 on the right, and discarding the leftmost bit.

Can you outline an algorithm which will multiply these two binary integers together reasonably efficiently? (Hint: long multiplication) How many additions and shift operations might you require?

(Harder Question) Can you do something similar for division?