Career: Engineering in Computer Science October 22, 2012

Teacher: Juan Manuel Tirado Martín

Subject: Computer Structure

Lab 1 Data Representation

Adrián María Mora Carreto NIA: 100291775

Ramses Joel Salas Machado NIA: 100292149

Index

Description of the implemented methods

- **Exercise 2**

  - integerToBinary(int integerNumber)

    This method receives and integer number and returns a string representing the number in natural binary. We implemented the algorithm learned in class; where we divided the integer number by 2 and store the remainder until we reach 1.

  - addBinary(String number1, String number2)

    This method receives two binary strings and returns a string with the binary addition result. We used the xor (^) operator in order to implement the addition and a Boolean variable to detect if we have carry when we are summing. As we have to compare bit per bit we also make both string of the same length distinguish which is the greatest and add as many 0's as necessary to the other string.

  - intToComp1(int integerNumber)

    This method receives an integer number and returns a string with the 1's complement representation of the number. We also used the algorithm learned in class; we obtain the natural binary representation with integerToBinary() method and add a zero bit to the left if the integer is positive or change 1's per 0's if it is negative and add a 1 to the left of the string.

  - intToComp2(int integerNumber)

    This method receives an integer number and returns a string with the 2's complement representation of the number. We employed the method intToComp1() to get the first complement; if the integer is positive the representation is the 1's complement and if it is negative we add 1 using the addBinary() method.

- **Exercise 3**

  - ❖ floatToBinary (float number)

    This method receives a float number as a parameter and converts it to binary. Firstly, we convert the real part into natural binary using integerToBinary() method, and then the decimal part is also converted using the algorithm of multiplying by two. We stop in 24 bits of precision of the decimal value.

  - ❖ floatToIEEE754(float realNumber)

    This method receives a float number and then converts it into IEEE 754 format. First of all, realNumber is converted into natural binary using the previous method and convert the number in three steps:

    1. Sign: checks if realNumber is positive (0) or negative (1)
    2. Exponent: when the number is in binary we have to normalize it in order to represent it, so we have now two cases: if the number starts with 0 or with 1. In both cases we have to check where the first 1 is and calculate the exponent of the number.
    3. Mantissa: the mantissa is calculated using the decimal part of the number now normalized but using the createMantissa() method.

  - ❖ createMantissa(String normalized)

    This method receives a binary string and returns a string with the decimal part of the binary string concatenating 0's at the end (if needed) until it reaches 24 characters.

1. **Calculate the number of representable numbers between 8 and 9, and between 15 and 16. In what interval can we represent a larger amount of numbers? Justify your answer.**

   The numbers of representable numbers that we can represent between 8 and 9 is $2^{20}+1$. Based on the fact that for representing the number 8 in natural binary we need 4 bits, normalizing it we get 3 decimal numbers that belong to the mantissa; permuting 23(bits in the mantissa)-3 we get 2^{20}. The result of the permutation is the numbers representing between $8 \leq x < 9$ adding 1 to include the number 9 we have the result.

Now, for calculate the representable numbers between 15 and 16 we used the same fact; for representing 15 in natural binary we need 4 bits. Therefore, 2^{20} is the amount of representable numbers for $15 \leq x < 16$ adding 1 to include the number 16 we have that $2^{20}+1$ is the result for this interval of numbers.

We conclude that we can represent the same amount of numbers in these intervals.

- **Exercise 4**

  - ❖ subBinary(String number1, String number2)

    This method receives two binary strings and returns the subtraction in binary for them. We can only subtract number in binary if the minuend is greater or equal than the subtrahend (in this case the minuend is the first string and the subtrahend the second one), is this is not the case we throw an exception.

  - ❖ binaryMultiplier(String num1,String num2)

    This method receives two binary strings and returns a string with the binary multiplication of them. First of all, we check if the numbers to multiply have a decimal part or not, and count how many numbers are after the period in order to multiply correctly. After that, we delete the period and multiply both numbers as if they were integer numbers. In order to multiply them, the method goes from right to left in the second number, and sees if the number is 1 or 0. If it is 1 then it copies the number into a temporal variable and adds as many 0 to the right as positions are moved from right to left. Then, the result is the addition of all the temporal variables.

  - ❖ floatMultiplier(float num1, float num2)

    This method receives two floats numbers and returns the multiplication of both float numbers in IEEE 754 format. First of all, we convert both numbers into IEEE 754 format, multiplying them in this representation in order to have more efficiency. At the very beginning, we compare the sign of the numbers using the xor(^) operator, then we multiply the mantissas with the explicit bit using the binaryMultiplier() method and normalized it, storing the exponent of the number. Now, we add both exponents and depending on the exponent resulting of the normalization, we have to add or subtract this new exponent and add 0 to the left until it reaches 8 characters (if the string is bigger than 8 character and the ones in the left are zero we delete them). Then, we finish the method creating the mantissa using the createMantissa() method and returning the resulting number.

- **Exercise 5**
  - ❖ integerAdder(int num1, int num2)

    This method receives two integer numbers, converts them in its 2's complement representation and returns a string with its addition in binary. The numbers are converted into 2's complement using the intToComp2() method and added using addBinary(). If the both integer has the same sign and the addition results in a different one, then we have overflow and we return an empty string.

Done tests and results

We did a lot of tests trying to solve the different problems that we experience during the development of the practice. The most problematic problem for us was the exercise number 4; where we had to implement the subtraction and multiplication methods for binary numbers. In each exercise with made at least 3 different tests in order to assure that the code works for every possible case.

Conclusions

Understanding how data is represented make us capable of decipher some machine instructions. In addition, knowing how to operate with this information and dealing with it using arithmetic we are able of manipulate the instructions in order to obtain different data.

With this laboratory practice we strongly reinforce our knowledge about data representation and binary arithmetic learned in previous subjects.