Computer Science G11 at The Dragon Academy
Assignment 7
Due date: Thu Dec. 6 2018

December 4, 2018

Write all answers in a single C source, adding the pertinent comments to make your submission as easy to understand and follow as possible. Avoid making it tedious to read though! Name the source file Assignment7.c and send it by email.

Make sure that your code compiles without errors nor warnings!

All questions have the same value.

1. **Declaration, Definition, Variable, Pointer and Dereferencing.**

   1 *Declare* a variable `j` of type int. This means just *telling* the compiler that there is a variable `j` which has a specific type of `int`. Example: `char c;` We declare the label 'c' to refer to a variable of type `char`.

   2 *Define* a variable `i` of type int. This means the same as declaring it, **plus** actually reserving the space in memory by assigning it a particular value. Also called *instantiating* the variable `i`. Example: `char c = 'S';` We declare the label'c' to refer to a variable of type `char` **AND** assign it the value 'S', thereby asking to actually reserve the necessary space in memory, have c referencing that space and storing there the bits representing the character 'S'.

   3 Declare a pointer, ptr, to an integer

   4 Assign the address of `i` to the pointer ptr

   5 Assign the value 1 to the variable `i`

   6 Print out the value of the pointer ptr (Hint: use the format `"%p"` and recast the pointer as void, i.e., `(void*)ptr`, when passing it to the printf function)

   7 Print out the value of the variable `i`

   8 Print out the value the pointer ptr is pointing to

   9 Print out the content of the (memory address) `ptr`

   10 We established that `"*ptr"` is the "content of (the memory address pointed to by) `ptr`". Store the value 2 into the memory location of `i`.

   11 Print out the value of `i`

   12 Following the previous question, assign the value 3 to i using `ptr`

   13 Print out the value of `i`

   14 Assign the value of 137 to `j` using a pointer to `j` and print out `j`

2. **Correct type of a pointer**

   1 In one statement, *define* a pointer `ptr_s` to a *short int* with the value of variable i

   2 Print out the value pointed by `ptr_s` and content of variable `ptr_s`.

   3 Does the value of that short int coincide with i? In which memory address is that short int been stored?

   4 Assign to j the value 65537, then assign its address to `ptr_s`

   5 Print out the value pointed by `ptr_s` and content of variable `ptr_s`.

   6 Does the value of that short int coincide with i? In which memory address is that short int been stored?

3. **How does the computer lay out the bits of an integer in memory? *Little Endian*, *Big Endian* and *basic pointer arithmetics*.**

    1 Define an integer variable `a` with the hex value value `0x00010203` (Note: the first 2 characters, `0x`, just mean that the rest is a number in hexadecimal notation. Assign it just as it is.)

    2 Print out the value of `a` and its address

    3 Define a pointer to a short int `ptr_sa` and assign it the address of a

    4 Print out the address contained in `ptr_sa` and the value it's referencing.

    5 Print out the content of the *next memory address* following that pointed to by `ptr_sa`. (Hint: The next memory address is given by `ptr_sa`+1)

    6 What is the difference between the memory addresses of this last one and that in `ptr_sa`? Express it in bytes.

    7 What did hapen? Explain the difference between these last three print statements?

    8 Consider the last print statement and answer the following questions:

        1 What value of `a` was printed when dereferencing the address following that pointed to by `ptr_sa`

        2 Hence, when we "go up" in memory we retrieve the higher bits (more to the left) or the lower bits (more to the right) of a byte?

        3 The hex number `0x010203` occupies 3 bytes. For arguments sake, let's say *it is store in memory address* `0x7fff51b0f900`. a) What number is stored in the first byte starting at that address, b) In which address is the `01` stored?