BLG 212E
Microprocessor Systems
Homework 1

Res. Assist. Meral Kuyucu

November 29, 2021

Dear all, this assignment consists of three questions. Please find the template for "Q{QuestionNumber}.s" file and write your code into that file (e.g. your answer for Question 1 should be written in: "Q1.s"). Zip your code files into a folder titled "FirstnameLastname_StudentID.zip".

Feel free to change the values of the hard coded inputs to test your codes. However, do not change the template of the files. The part of the file that your code should reside in is indicated with comments.

When coding, please make sure to provide comments **per-line**. Be informed that code without proper explanation will be penalized.

UPDATE – IMPORTANT NOTE: You will receive full points for completing Questions 1, 2, 3 OR 1, 2, 4. The assignment will be graded out of 3 parts, and you have the option of picking one of question 3 and 4 for the third part of the assignment. If however, you choose to do all 4 questions, you will receive bonus points. ☺

If you have any questions, feel free to email me at korkmazmer@itu.edu.tr or visit me at EEB 1208. ☺

Topic - Computation of the Power Function

For this homework, you will be coding the computation of the power function in three different ways. The formula you will be implementing is: $y = x^a$. In this formula, the base (x) and the exponent (a) are hard-coded into memory and space is allocated for the result (y). You can access these variables using their symbolic names ("x", "a", and "y"). Read the inputs, perform the exponentiation, and write the output into the designated space to achieve full credit.

NOTE: The values for base and exponent will always be positive. You don't need to check!

Question 1 - Iterative Power Function

The most straightforward way to solve this problem is to compute by iteration. Implement the pseudocode given in Algorithm 1 for this part of the assignment.

Data: a = $Exponent$, x = $Base$
Result: $y = x^a$
$count \leftarrow a$;
$i \leftarrow 1$;
$x \leftarrow x$;
$y \leftarrow 1$;
while $i \leq count$ **do**
  $y \leftarrow y * x$;
  $i \leftarrow i + 1$;
end

Algorithm 1: Iterative Function: $y = Power(x, a)$

Question 2 - Recursive Power Function ($O(a)$)

Recursion can also be used to compute power. Implement a code that calls a power subroutine as many times as the size of the exponent. Each time the subroutine is called, the exponent argument is decremented until a base case is satisfied. Afterward, the problem is solved by multiplying from bottom-up. The complexity of this implementation is $O(a)$, proportional to the size of the exponent. Implement the pseudocode given in Algorithm 2.

Data: a = $Exponent$, x = $Base$
Result: $y = x^a$
$count \leftarrow a$;
$i \leftarrow 1$;
$x \leftarrow x$;
$y \leftarrow 1$;
if $a = 0$ **then**
  return 1;
else
  return $x * power(x, a - 1)$;
end

Algorithm 2: Recursive Power Function with $O(a)$ Complexity

NOTE: Make use of the **"BL"** instruction to enter a subroutine. Before entering the subroutine, use the stack to first save the return address, and

then the necessary arguments for the subroutine. In the subroutine, pop the arguments and use them. When returning from the subroutine, pop the return address and use the **"BX"** instruction to return one layer above in the program execution.

Question 3 - Recursive Power Function ($O(a)$)

The general idea behind recursion is to divide the problem up into equal sized smaller sub-problems and solve by the divide and conquer method. For this part, you're required to compute the power function by dividing the problem into two equally sized sub-problems (as opposed to decrementation as done in the previous part). Refer to Algorithm 3 for your solution.

Data: a = *Exponent*, x = *Base*
Result: $y = x^a$
if $a = 0$ **then**
  return 1;
else
  if *(a % 2 == 0)* **then**
    return $power(x, \frac{a}{2}) * power(x, \frac{a}{2})$;
  else
    return $x * power(x, \frac{a}{2}) * power(x, \frac{a}{2})$;
  end
end

Algorithm 3: Recursive Divide and Conquer Power Function with $O(a)$) Complexity

Question 4 - Recursive Power Function ($O(log(a))$)

In the previous examples, recursion did not bring us much of an advantage. The number of steps taken to solve the previous two parts is proportional to the size of the exponent. A better divide-and conquer approach can be taken.

Now modify the code from the previous part to improve execution time to $O(log(a))$. This means that the number of steps in the execution should be proportional to $log(a)$. Again, use the divide and conquer approach to recursively divide the problem into two problems of size $\frac{a}{2}$ until you reach a base case. Notice that the output of the two sub-problems is the same. Compute the output once. Square the output of the sub-problem instead of making two recursive calls. Follow through with a bottom up approach similar to Questions 2 and 3. Refer to Algorithm 4 to complete this part.

Data: a = $Exponent$, x = $Base$
Result: $y = x^a$
if $a = 0$ **then**
  return 1;
else
  $temp \leftarrow power(x, \frac{a}{2})$
  if *(a % 2 == 0)* **then**
    return $temp * temp$;
  else
    return $x * temp * temp$;
  end
end

Algorithm 4: Recursive Power Function with $O(log(a))$ Complexity