|  |  |
| --- | --- |
| **Hands-On Activity No. 5** | |
| **LOGICAL INSTRUCTIONS** | |
| **Course Code:** CPE021 | **Program:** Computer Engineering |
| **Course Title:** Computer Architecture and Organization | **Date Performed:** March 10, 2025 |
| **Section:** CPE22S2 | **Date Submitted:** March 10, 2025 |
| **Name:** Adia, James Russel E. | **Instructor:** Engr. Maria Rizette H. Sayo |
| **A. Procedure: Output(s) and Observation(s)** | |
| **Sample Problem 1:**  Type the following program in Notepad.   |  | | --- | | TITLE logic.asm  TITLE logic.asm  .model small  .stack 100h  .data  myStringdb "Proud to be TIPians","$"  .code  main proc  movax,@data  movds,ax  movbx,offset myString  LP1: mov dl,[bx]  Cmp dl, '$'  Je exit  Inc bx  ;insert code here  mov ah,02  int 21h  jmp lp1  Exit: Mov ax, 4c00h  Int 21h  Main endp  End main |   Save the program as **logic.asm**  Assemble and execute the program.    Analyze the output and record the output in Table 5.1  **Table 5.1 – Output of logic.asm**   |  | | --- | |  |   **Sample Problem 2:**   * + - 1. Modify program logic.asm.       2. Replace the line “**; insert code here**”, with "**and dl, 11011111B**" .       3. Save the program as **and.asm**.       4. Assemble and execute the program.      * + - 1. Observe and record the output in Table 5.2   **Table 5.2 – Output of and.asm**   |  | | --- | |  |  * + - 1. How is your output different from before? Why? * My output is different from before in a way that the string “Proud to be TIPians” in logic.asm is now outputted as all uppercase letters in and.asm. This is because of the added code which is used to convert any lowercase letter in the register dl to its uppercase equivalent.   **Sample Problem 3:**   1. Modify logic.asm again, this time replace the line “**;insert code here**”, with “**xor dl, 00100000B**". 2. Save the program as **xor.asm**. 3. Assemble and execute the program.      1. Observe and record the output in Table 5.3.   **Table 5.3 – Output of xor.asm**   |  | | --- | |  |   How is your output different from before? Why?   * The output here differs from the output of logic.asm since the case of each alphabetic character is toggled. Uppercase letters become lowercase, and lowercase letters become uppercase. This is because the xor instruction was used to toggle the case of each alphabetic character.   **Sample Problem 4:**   1. Modify logic.asm once again, this timeplace the line “**;insert code here**”, with "**or dl, 00100000B**". 2. Save the program as **or.asm.** 3. Assemble and execute the program.      1. Observe and record the output in Table 5.4.   **Table 5.4 – Output of or.asm**   |  | | --- | |  |  1. How is your output different from before? Why?  * The output is different from before because this time the string output is all in lowercase which is opposite of what was outputted when using ‘and’ in Sample Problem 2. This is because the or instruction was used with 00100000B | |
| **B. Supplementary Activity: Output(s) and Observation(s)** | |
| 1. Write an assembly program that will simulate the given Boolean expression using assembly programming.   AL = (AH∙ BH + AL∙ BL)’ xor (CL+(CH∙ DH)’ ∙ DL)’  Assembly Program (.asm was screenshotted on VSCode for better readability)      \* Note: The program outputs nothing since the final result was only stored in ‘result’ variable and there was not code implemented that lets it output the result of the sample values indicated \*   1. Give a sample problem where the logical instructions can be applied.  * In bitmap graphics processing, logical instructions form the foundation of pixel manipulation. For instance, when modifying an image, you can use AND operations with specific masks to isolate color components while preserving others. By applying OR operations with new color values, you can blend colors without affecting preserved bits. XOR operations are particularly useful for creating visual effects or toggling specific pixel attributes. This technique is fundamental in game development, image processing applications, and UI rendering where individual bits represent color channels or transparency values. * Link for more information on bitmap graphics processing:   <https://pixinsight.com/developer/pcl/doc/html/group__bitmap__bitwise__ops.html> | |
| **C. Conclusion & Lessons Learned** | |
| Upon completing the hands-on activity, I have developed a better understanding of various logical instructions and their applications. I was able to effectively compare different logical instructions, identifying their unique characteristics and appropriate use cases by doing the procedures in the activity. Additionally, I have gained practical experience by creating a program in assembly that incorporate these logical instructions when given a Boolean expression. In conclusion, I was able to perform the tasks required with the help of the procedures as my guide especially for the part wherein I needed to write an assembly program to perform the given Boolean expression. | |