**Assignment No.2**

|  |
| --- |
| **Title of Assignment:**  Write 64 bit ALP to overlapping and non-overlapping block of transfer with and without MOVSB instruction. |
| **Relevant Theory:**  In overlapped, non-overlapped block transfer, we can transfer byte-by-byte data from source block to destination block. We can check that memory is allocated for variables sequentially whatever the sequence is available in data section.  **Basic instructions:**   1. **MOVSB:**Copy byte at DS:[SI] to ES:[DI]. Update SI and DI   MOVSB No operands.  Algorithm:   |  |  | | --- | --- | |  | ES:[DI] = DS:[SI] | |  | if DF = 0 then   |  |  | | --- | --- | |  | SI = SI + 1 | |  | DI = DI + 1 |   else   |  |  | | --- | --- | |  | SI = SI – 1 | |  | DI = DI – 1 | |  1. **CLD :**Clear Direction flag. SI and DI will be incremented by chain instructions: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, STOSB, STOSW   No operands  DF=0   1. **STD :N**o operands Set Direction flag. SI and DI will be decremented by chain instructions: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, STOSB, STOSW.   No Operands  DF=1   1. **REP**   REP chain instruction Repeat following MOVSB, MOVSW, LODSB, LODSW, STOSB, STOSW instructions CX times.   Algorithm:  check\_cx:  if CX <> 0 then   |  |  | | --- | --- | |  | do following chain instruction | |  | CX = CX – 1 | |  | go back to check\_cx |   else   |  |  | | --- | --- | |  | exit from REP cycle |   **Non-overlapped block transfer:-**  A) Algorithm:-  1)declare the src and dest string and one integer variable count.  2)initialize count = 5  3)accept src string with number of character 5 e.g. HELLO  4) point SI to src and DI to dest  5)copy the [SI] to [DI] using register.  6) increment SI and DI  7) decrement count  8) compare count = 0 if no then go to step 5 else 9.  9) again point SI to src and initialize count = 0A means 5+5  10) display src string and give string length= count  11) stop  **Non-overlapped block transfer using MOVSB:-**  Algorithm:  1)declare the src and dest string and one integer variable count.  2)initialize count = 5  3)accept src string with number of character 5 e.g. HELLO  4) point SI to src and DI to dest  5) initialize rcx = 5  6) clear direction flag using CLD instruction.  7) copy content from SI to DI using REP MOVSB  8) again point SI to src and initialize count = 0A means 5+5  9) display src string and give string length= count  10) stop  **Overlapping block transfer:-**  A) Algorithm:-  1)Declare the src and dest string and one integer variable count.  2)initialize count = 5  3)accept src string with number of character 5 e.g. HELLO  4) point SI to src and DI to dest  5) Give displacement to SI and point SI to last location e.g. add SI,04  6) Give displacement to DI and point DI to some location e.g. add DI,02  **Here 02 is number of characters will get overlapped**  7) Copy the [SI] to [DI] using register.  8) decrement SI and DI  9) Decrement count.  10) Compare count = 0 if no then go to step 7 else 11.  11) Again point SI to src and initialize count = 0A means 5+(5-2)  12) display src string and give string length= count  13) stop  **Overlapped block transfer using MOVSB:-**  Algorithm:  1)Declare the src and dest string and one integer variable count.  2)initialize count = 5  3)accept src string with number of character 5 e.g. HELLO  4) point SI to src and DI to dest  5) Give displacement to SI and point SI to last location e.g. add SI,04  6) Give displacement to DI and point DI to some location e.g. add DI,02  **Here 02 is number of characters will get overlapped**  7) initialize rcx = 5  8) Set direction flag using STD instruction.  9) copy content from SI to DI using REP MOVSB  10) again point SI to src and initialize count = 0A means 5+5  11) display src string and give string length= count  12) stop |
| **Mathematical Modeling:**  Let srcbe a given source string with n number of characters. And dest is destination string which can stored n number of characters.  For overlapped: output set: {(src\_len- 2)+dest}, where + is concatenation and number of characters overlapped= 2.  For non-overlapped: output set= {src+dest} where + is concatenation. |
| **Testing:**  **Test Conditions:**  Check for overlapped and non-overlapped block transfer.  **Input:**  **For both: HELLO**  **Output:**   1. for overlapped: HELHELLO 2. **for Non Overlapped: HELLOHELLO** |
| **FAQs:**   1. What is flag registered? 2. Use of DF. 3. Explain CLD, STD, MOVSB instruction. |
| **Conclusion:**  Successfully completion of overlapped and non-overlapped block of transfer with and without MOVSB instruction. |