|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**PBLE**

|  |  |
| --- | --- |
| Semester | S.E-Semester IV – Computer Engineering |
| Subject | Micro Processor |
| Subject Professor In-charge | Prof. Suvarna Bhat |
| Assisting Teachers | Prof. Suvarna Bhat |

|  |
| --- |
| Student Name – Deep Salunkhe |
| Roll Number – 21102A0014 |
| Division and Batch – Division A, Batch 1 |
| **PBLE Title:** Develop assembly language code to convert the input from the keyboard to the format which can be used for processing internally |
| |  | | --- | | **Implementation**  .model small  .stack 100h  .data      buffer db 256        *; buffer to store the input string*      prompt db "Enter a string: $"      prompt\_len equ $-prompt      crlf db 0dh, 0ah, "$" *; carriage return and line feed for printing*  .code      mov ax, @data        *; initialize data segment*      mov ds, ax      lea dx, prompt       *; print prompt message*      mov ah, 09h      int 21h      mov ah, 0ah           *; read string from keyboard*      mov dx, offset buffer      int 21h      lea si, buffer        *; load address of input buffer into SI*      lea di, buffer        *; load address of output buffer into DI*      xor cx, cx            *; clear CX register*      convert\_loop:          lodsb             *; load next character from input buffer*          cmp al, 0dh       *; check if it's a carriage return*          je done           *; if it is, jump to done*          mov bl, al        *; move character to BL register*          mov bh, 0         *; clear BH register*          mov ax, 0         *; clear AX register*          mov cl, 4         *; shift 4 bits at a time*          shl bl, cl        *; shift left by 4 bits*          or ax, bx         *; combine into ASCII code*          shl ax, cl        *; shift left by 4 bits*          or ax, bx         *; combine into ASCII code*          stosw             *; store ASCII code in output buffer*          inc cx            *; increment CX register*          jmp convert\_loop  *; loop back to process next character*      done:          lea dx, crlf      *; print new line*          mov ah, 09h          int 21h          mov ah, 4ch       *; exit program*          int 21h  end  **Explanation:**  .model small  .stack 100h  The first two lines define the memory model and stack size for the program.  .data      buffer db 256        *; buffer to store the input string*      prompt db "Enter a string: $"      prompt\_len equ $-prompt      crlf db 0dh, 0ah, "$" *; carriage return and line feed for printing*  The **.data** section declares the variables that the program will use. **buffer** is a buffer to store the input string from the user. **prompt** is a string message to prompt the user to enter a string. **prompt\_len** is the length of the **prompt** string, calculated using the **$** operator. **crlf** is a string containing a carriage return, line feed, and dollar sign for printing purposes.  .code      mov ax, @data        *; initialize data segment*      mov ds, ax  The **.code** section contains the executable code for the program. The first two lines set up the data segment register **DS** to point to the **.data** section.      lea dx, prompt       *; print prompt message*      mov ah, 09h      int 21h  These three lines use the **lea** instruction to load the offset of **prompt** into the **DX** register, **mov** to set the value of **AH** to **09h** which is the interrupt code for "print string", and **int 21h** to call the DOS interrupt to print the prompt message to the console.      mov ah, 0ah           *; read string from keyboard*      mov dx, offset buffer      int 21h  These lines set the value of **AH** to **0ah**, which is the interrupt code for "read string from keyboard", use **mov** to load the offset of **buffer** into the **DX** register, and then call the DOS interrupt to read a string from the keyboard and store it in the **buffer**.  lea si, buffer        *; load address of input buffer into SI*      lea di, buffer        *; load address of output buffer into DI*      xor cx, cx            *; clear CX register*  These lines use the **lea** instruction to load the address of **buffer** into the source index **SI** and destination index **DI** registers, and **xor** to clear the counter **CX** register.      convert\_loop:          lodsb             *; load next character from input buffer*          cmp al, 0dh       *; check if it's a carriage return*          je done           *; if it is, jump to done*          mov bl, al        *; move character to BL register*          mov bh, 0         *; clear BH register*          mov ax, 0         *; clear AX register*          mov cl, 4         *; shift 4 bits at a time*          shl bl, cl        *; shift left by 4 bits*          or ax, bx         *; combine into ASCII code*          shl ax, cl        *; shift left by 4 bits*          or ax, bx         *; combine into ASCII code*          stosw             *; store ASCII code in output buffer*          inc cx            *; increment CX register*          jmp convert\_loop  *; loop back to process next character*  This is the loop where the input string is processed and converted to ASCII codes. The **lodsb** instruction loads the next character from the input buffer into the AL register, and the **cmp** instruction checks if it's a carriage return (ASCII code 13). If it is, the loop jumps to the **done** label to exit.  If the character is not a carriage return, it is moved to the BL register using **mov bl, al**. Then, the BH register is cleared with **mov bh, 0**. The AX register is also cleared with **mov ax, 0**.  The next four instructions are used to convert the character to its ASCII code. The **cl** register is loaded with the value 4, indicating that the conversion is done 4 bits at a time. The **shl** instruction shifts the value in BL left by 4 bits, effectively multiplying it by 16. This result is then combined with the original value in BL using the **or** instruction, which produces the first 8 bits of the ASCII code in the AX register. The same process is repeated again, with the result shifted left by another 4 bits and combined with the original value in BL, producing the last 8 bits of the ASCII code in the AX register.  Finally, the **stosw** instruction stores the 16-bit value in the AX register into the output buffer pointed to by the DI register. The CX register is then incremented to keep track of the number of characters processed, and the loop jumps back to **convert\_loop** to process the next character.  The **done** label is reached when the loop detects a carriage return in the input string. It prints a new line using the **crlf** string, then exits the program using the **int 21h** instruction with the **ah** register loaded with the value 4Ch, which terminates the program. | |  | |