DATA SEGMENT ; Start of the data segment

BLOCK1 DB 'MALAYALAM' ; Define a byte array containing the string to be checked

MSG1 DB "IT IS PALINDROME $" ; Define a byte array containing the message to be printed if it is a palindrome

MSG2 DB "IT IS NOT PALINDROME $" ; Define a byte array containing the message to be printed if it is not a palindrome

PAL DB 00H ; Define a byte variable to store the flag for palindrome

DATA ENDS ; End of the data segment

PRINT MACRO MSG ; Define a macro for printing a message

MOV AH,09H ; Set the value of AH register to 09H, which is the code for printing a string

LEA DX,MSG ; Set the value of DX register to the address of the message

INT 21H ; Call interrupt 21H to print the message

INT 3H ; Call interrupt 3H to terminate the program

ENDM ; End of the macro definition

EXTRA SEGMENT ; Define an extra segment

BLOCK2 DB 9 DUP(?) ; Define a byte array in the extra segment

EXTRA ENDS ; End of the extra segment

CODE SEGMENT ; Start of the code segment

ASSUME CS:CODE,DS:DATA,ES:EXTRA ; Set the values of CS, DS, and ES registers

START: ; Start of the program

MOV AX,DATA ; Move the address of the data segment to the AX register

MOV DS,AX ; Move the value of AX register to the DS register

MOV AX,EXTRA ; Move the address of the extra segment to the AX register

MOV ES,AX ; Move the value of AX register to the ES register

LEA SI,BLOCK1 ; Load the offset address of BLOCK1 to the SI register

LEA DI,BLOCK2+8 ; Load the offset address of BLOCK2+8 to the DI register

MOV CX,00009H ; Load the count of bytes to be copied to CX register

BACK: ; Start of the loop

CLD ; Clear the direction flag

LODSB ; Load a byte from the address in SI to the AL register

STD ; Set the direction flag

STOSB ; Store the byte in the AL register to the address in DI

LOOP BACK ; Decrement the count in CX and jump to BACK if CX is not zero

LEA SI,BLOCK1 ; Load the offset address of BLOCK1 to the SI register

LEA DI,BLOCK2 ; Load the offset address of BLOCK2 to the DI register

MOV CX,0009H ; Load the count of bytes to be compared to CX register

CLD ; Clear the direction flag

REPZ CMPSB ; Compare each byte in BLOCK1 and BLOCK2, set ZF if all bytes match

JNZ SKIP ; Jump to SKIP if the ZF flag is not set

PRINT MSG1 ; Call the PRINT macro with MSG1 as the argument if it is a palindrome

SKIP: PRINT MSG2 ; Call the PRINT macro with MSG2 as the argument if it is not a palindrome

CODE ENDS ; End of the code segment

END START ; End of the program

Different type of explanation

Sure, I can provide you with an explanation of this code and some tips to understand it better.

Firstly, let's break down the code into its various parts:

1. The code has three segments: DATA, EXTRA, and CODE.
2. The DATA segment contains the following variables:
   * BLOCK1: A string of characters "MALAYALAM".
   * MSG1: A string of characters "IT IS PALINDROME ".
   * MSG2: A string of characters "IT IS NOT PALINDROME ".
   * PAL: A byte variable initialized with 00H.
3. The PRINT macro is defined which takes a message string as input and displays it using DOS interrupt 21H.
4. The EXTRA segment contains a single variable: BLOCK2, which is an array of 9 bytes.
5. The CODE segment contains the main program logic, which performs the following steps:
   * Copies the string in BLOCK1 to BLOCK2 in reverse order.
   * Compares the strings in BLOCK1 and BLOCK2 to check if they are palindromes.
   * Displays the appropriate message depending on the result of the comparison.

Now, let's discuss some tips to understand this code better:

1. Start by understanding the purpose of the program. In this case, it checks if a given string (in this case, "MALAYALAM") is a palindrome or not.
2. Next, identify the different segments of the code and the variables defined in each segment.
3. Study the main program logic step by step. Understand how the string in BLOCK1 is copied to BLOCK2 in reverse order and how the strings are compared to check for palindromes.
4. Understand how the PRINT macro is used to display the appropriate message depending on the result of the comparison.
5. Finally, run the code and test it with different inputs to see how it works in practice.

By following these tips and breaking down the code into smaller, manageable chunks, you should be able to understand this code better and learn it more easily.