Lab Work-01

Course Name: Microprocessor and Microcontroller Lab

Course: CSE 316

Submitted by:

Name: Hashibur Rahman Redoy

ID: 18192103276

Inteke: 41 Section: 03

Program: BSc in CSE (BUBT)

Submitted To:

Jahirul Islam Babar

Lecturer: Dept. of CSE

BUBT

Submission Date: 2-03-2023

Introduction

odd number. It then displays the appropriate message on the screen. The code uses a "newline" macro to print a new line The following assembly code takes an input from the user and checks whether it is an even or character before each message and the "exit" label to skip over code that is not needed, depending on whether the input number is even or odd.

Code Explanation

This assembly code is a program that takes an input from the user, checks if the input number is even or odd, and then displays a message accordingly. Let's go through the code line by line:

The first section of the code defines a macro "newline" that prints a new line character on the console. This is used later in the code to print new lines between messages.

```
newline macro
lea dx,nl
mov ah,9
int 21h
endm
```

The ".model small" and ".stack 100h" directives define the model and stack size respectively for the program.

```
.model small
.stack 100h
```

The ".data" directive defines data that is initialized at compile-time.

```
.data
msg1 db 'number is even', '$'
msg2 db 'number is odd',  '$'
nl   db  0dh,0ah,         '$'
```

Here, three data items are defined. "msg1" and "msg2" are strings that contain the messages to be displayed if the input number is even or odd respectively. "nl" is a new line character that is used by the "newline" macro.

The ".code" directive starts the code segment of the program.

.code

The ".startup" directive marks the beginning of the program.

.startup

The instructions "mov ax,@data" and "mov ds,ax" are used to initialize the data segment register (DS) with the value of the segment address of the data segment, which is stored in the AX register.

mov ax,@data
mov ds,ax

The "@" symbol before the "data" keyword indicates that the assembler will automatically assign a segment address to the data segment. The "mov ax,@data" instruction loads this address into the AX register, and the "mov ds,ax" instruction then transfers it into the DS register.

By initializing the data segment register, the program ensures that it can correctly access any data stored in the data segment. This is important because the data segment is a separate memory area from the code segment, and the program needs to tell the processor where to find the data it needs to operate on.

The first instruction in the program reads an input number from the user using interrupt 21h with the AH register set to 1.

mov ah,1
int 21h

The input number is then divided by 2 to check if it is even or odd.

mov bl,2
div bl
mov al,ah

cmp al,0
jg odd

The "mov bl,2" instruction loads the value 2 into the BL register. The "div bl" instruction divides the contents of the AL register (which contains the input number) by the value in the BL register (which is 2 in this case). The quotient is stored in AL, and the remainder is stored in AH.

The "mov al,ah" instruction moves the remainder from the AH register to the AL register. The "cmp al,0" instruction compares the contents of the AL register with 0.

The "jg odd" instruction jumps to the "odd" label if the contents of the AL register are greater than 0, which means that the input number is odd.

If the input number is even, the program jumps to the "even" label and prints a new line character using the "newline" macro.

```
even:
  newline
  lea dx,msg1
  mov ah,9
  int 21h
  jmp exit
```

The "lea dx,msg1" instruction loads the address of the "msg1" string into the DX register. The "mov ah,9" instruction sets the AH register to 9, which is the code for printing a string to the console. The "int 21h" instruction calls interrupt 21h with the specified parameters to print the message.

The "jmp exit" instruction jumps to the "exit" label, which is used to skip over the "odd" label and any code that follows it.

If the input number is odd, the program jumps to the "odd" label and prints a new line character using the "newline" macro.

```
odd:
  newline
  lea dx,msg2
  mov ah,9
  int 21h
```

The "lea dx,msg2" instruction loads the address of the "msg2" string into the DX register.

The "mov ah,9" instruction sets the value of the AH register to 9, which is the DOS interrupt code for printing a string.

lea dx,msg2
mov ah,9

The "int 21h" instruction then executes the DOS interrupt, which prints the string pointed to by DX to the console.

int 21h

In summary, these instructions load the address of the "msg2" string into the DX register, set the AH register to the appropriate value for printing a string, and execute the DOS interrupt to print the string to the console. This results in the message "number is odd" being displayed to the user.

Conclusion

In conclusion, this assembly code demonstrates how to perform simple conditional branching and string output in x86 assembly language. The code initializes the data segment register to access the data stored in the data segment, then uses a conditional jump to determine whether the input number is even or odd. It then uses the DOS interrupt to output the appropriate message to the console. Overall, this code provides a basic introduction to x86 assembly language programming and serves as a foundation for more complex programs.