**Input-Output Instructions**

Table of Contents

[BIOS Routines 3](#_Toc83228430)

[DOS Routines 4](#_Toc83228431)

[INT 21H 4](#_Toc83228432)

[Inputs 5](#_Toc83228433)

[Outputs 5](#_Toc83228434)

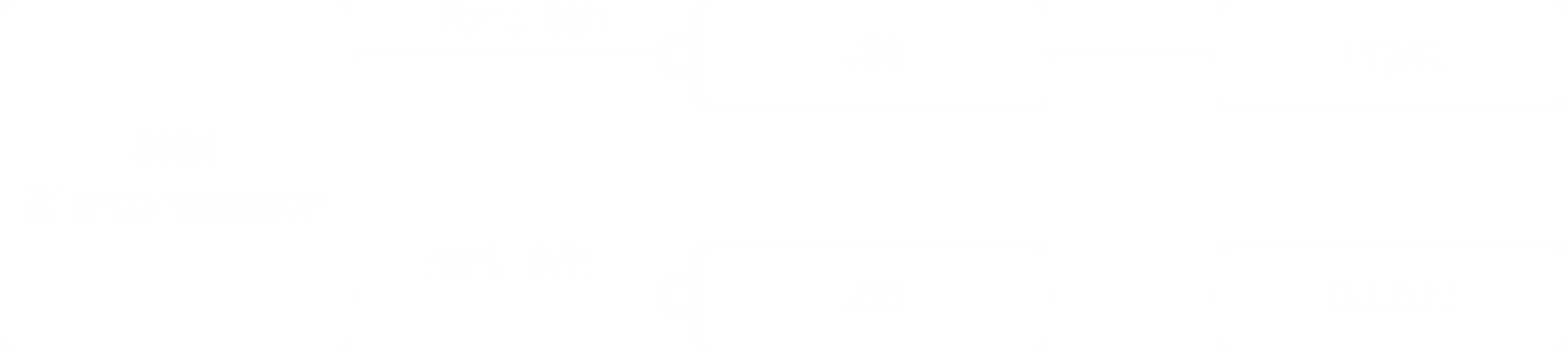
[Control Functions 6](#_Toc83228435)

To invoke an **I/O Operation** in a system, we can do one of two things:

1. Use a **BIOS routine**.
2. Use a **DOS routine**.

## BIOS Routines

Say we have a microprocessor connected to two different interfaces, one connected to an input device and one connected to an output device. The connection to each interface has a port number, say 05h and 07h respectively.



The **BIOS routine** for **input** looks like this:

IN AL, 05H *; take input into AL*MOV DL, AL *; move into DL*

ASSEMBLY

If the connected input device is a keyboard for example, we can press a **single key** and the **ASCII value** for that key will be stored in the AL register. Immediately after doing so, we moved the input data into the DL register, since we should store data in the data register. This is not just good practice, but also necessary, since the AL register can be changed unintentionally sometimes.

Similarly, the BIOS routine for **output** looks like this:

OUT 07H, DL *; output contents of DL*

ASSEMBLY

## DOS Routines

The major issue with using BIOS routines is that **port numbers are system-dependent** or manufacturer-dependent. This means that we cannot write a program using these BIOS routines and use that program on a different computer.

**DOS** solves this issue. The DOS routines are actually just using the BIOS routines in an intelligent way to make things easier for us. DOS provides an instruction to **retrieve the port number** for specific input and output devices. This instruction is INT 21H.

### INT 21H

INT 21H is an **interrupt**. The main instruction, INT, is taking an **interrupt number** as an argument. The interrupt number specifies a **specific routine**.

INT 21H can actually be used to invoke a large number of **DOS functions**. This is done by placing a **function number** in the **AH register** and then invoking INT 21H. We will primarily be interested in three functions.

1. **Function Number** , which is used for **single-key inputs**.
2. **Function Number** , which is used for **single-key outputs**.
3. **Function Number** , which is used for **character string outputs**.

For now, we will only look into the first two. The character string outputs will be seen when discussing arrays.

### Inputs

MOV AH, 1 *; single-key input*INT 21H *; ASCII code for input will be placed in AL*

ASSEMBLY

The above code is all we need to get a **single-key input**. Since INT 21H is an interrupt, the program cannot move forward until the user inputs a key. Once this is done, the **hexadecimal ASCII code** for the character inputted will be placed in the **AL register**.

### Outputs

MOV AH, 2 *; single-key output*MOV DL, 41H *; move character to output into DL*INT 21H *; character in DL will be displayed*

ASSEMBLY

To be able to give a **single-key output**, we first have to move the **hexadecimal ASCII code** for the character we want to output into the **DL register**. The corresponding character for that ASCII code will be shown as the output.

Alternatively, we can also move a character directly into the DL register.

MOV AH, 2  
MOV DL, 'A'  
INT 21H

ASSEMBLY

### Control Functions

In Assembly language, we have to do literally everything ourselves. This is going to become very obvious now.

Say we want to output two characters, one after the other. The code for that should look like this:

MOV AH, 2

MOV DL, 'A'  
INT 21H  
MOV DL, 'B'  
INT 21H

ASSEMBLY

The output for this code will place the two characters side-by-side.

Output: AB

If we want the second character to be on a **new line**, we would have to print out the **newline** character, obviously. The hexadecimal code for a newline character is 0AH.

MOV AH, 2

MOV DL, 'A'  
INT 21H  
MOV DL, 0AH  
INT 21H  
MOV DL, 'B'  
INT 21H

ASSEMBLY

Output: A

B

Notice how the second character is on the new line, but is placed one character to the right. For anyone who has written any code in high-level programming languages, this is very odd.

In any high-level programming language, along with the newline character, a **carriage return** character is also printed. This is what brings the cursor back to the front of the line and thus prints whatever comes next at the beginning of the line. But of course, since this is Assembly language, we have to do even that ourselves.

The hexadecimal code for a carriage return is 0DH.

MOV AH, 2

MOV DL, 'A'  
INT 21H  
MOV DL, 0AH  
INT 21H  
MOV DL, 0DH  
INT 21H  
MOV DL, 'B'  
INT 21H

ASSEMBLY

Output: A

B