Princess Sumaya University for Technology
Computer Engineering Department
22440: Microprocessor Lab

Experiment 2: Program Control Instructions and Interrupts

Introduction

The program control instructions direct the flow of a program and allow the flow of the program to change. These instructions include: jumps, calls, and returns.

The Jump Group

The main type of program control instructions; (**JMP**) allows the programmer to skip sections of a program and branch to any part of memory for the next instruction. A conditional jump instruction allows the programmer to make decisions based on numerical tests. The results of these numerical tests are held in the flag bits which are tested by conditional jump instruction.

A) Unconditional Jumps

There are three types of unconditional jump instructions: short, near, and far, the short jump allows a branch to within +127 to -128 bytes. The near jump allows a jump to anywhere in the current code segment. The far jump allows a jump to any location in the memory, you will note that, in the machine language, the displacement that follows a short or near jump is the distance from the next instruction to the jump location.

B) Conditional Jumps

Conditional jumps are always short jumps; this limits the range of the jump within +127 to -128 bytes from the location following the conditional jump. The conditional jump instructions test. The following flag bits: sign (S), zero (Z), carry (C), parity (P), and overflow (0). If the condition under lest is true, a branch to the label associated with the jump instruction occurs. If the condition is false, the next sequential step in the program executes. Because we use both signed and unsigned numbers, and the order of these numbers is different, there are *two* sets of magnitude comparison conditional jump instructions. When we compare signed numbers we use **JG**, **JGE**, **JLE**, **JE**, and **JNE**. When we compare unsigned numbers we use **JA**, **JB**, **JAE**, **JBE**, **JE**, and **JNE**.

C) Loops

A special conditional jump instruction (**LOOP**) decrements CX and jumps to the label when CX is not 0. The **LOOP** instruction also has conditional forms: **LOOPE**, **LOOPNE**, **LOOPZ**, and **LOOPNZ**. These instruction jump if CX is not 0 and if the condition exists.

Procedures

The procedure or subroutine is an important part of any computer system architecture. A procedure is a group of instructions that usually perform one task. It is a reusable section of the software that is stored in memory once, but used as often as necessary. This saves memory space and makes it easier to develop software. The only disadvantage of a procedure is that it takes the computer a small amount of time to link to the procedure and return from it. The **CALL** instruction links to the procedure and the **RET** instruction returns from it. The stack stores the return address whenever a procedure is called during execution of a program. The **CALL** instruction pushes the address of the' instruction following it on the stack. The **RET** instruction removes an address from the stack so the program returns to the instruction following the **CALL**.

There is a special type of **CALL** instructions in 8088 microprocessor. Whenever a software interrupt instruction executes it (1) pushes the flag onto the stack, (2) clears the T and I flag bits, (3) pushes CS onto the stack, (4) fetches the new value for CS from the vector, (5) pushes IP onto the stack, (6) fetches the new value for IP from the vector, and (7) jumps to the new location addressed by CS and I P. The **INT** instruction performs as a far **CALL** except that it not only pushes CS and IP onto the stack, but it also pushes the flag onto the slack.

DOS Interrupts (INT 21H)

INT 21H may he used to invoke a large number of **DOS** functions; a particular function is requested by placing a function number in the AH register and invoking INT 21H.

Function 01H: Keyboard Input

Waits for a character to be read at the standard input device, then echoes the character to the standard output device and returns the ASCII code in AL.

Input: AH=01H
Output: AL = character from the standard input device

Function 02H: Display Output

Outputs the character in DL to the standard output device.

Input: AH = 02H
DL = character
Output: none

Function 2 may also be used to perform control functions. If DL contains the ASCII code of a control character, INT 21H causes the control function to be performed. The principal control characters are as follows:

ASCII Code (Hex)	Function
08	Backspace
09	Tab
0A	Line Feed
0D	Carriage Return

Function 09H: Display a String

Outputs the characters in the print string to the standard output device.

Input: AH = 09H
DS:DX = pointer to the character string ending with '$'
Output: none

Function 4CH: Terminate the Program (EXIT)

Terminates the current process and transfers control to the invoking process.

Input: AH = 4CH
AL = return code
Output: none

Putting It All Together

Use the following sequences wherever you need in your programs:

1) To display a question mark

```
MOV AH, 2
MOV DL, '?'          ;The ASCII code for '?' is 3FH
INT 21H
```

2) To read a character

```
MOV AH, 1
INT 21H
MOV BL, AL
```

3) To display a character at the beginning of the next line

```
MOV AH, 2
MOV DL, 0DH          ;0D is the Hex code for carriage ret
INT 21H
MOV DL, 0AH          ;0A is the Hex code for line feed
INT 21H
MOV DL, BL
INT 21H
```

4) To display a string

```
;Declare the string in the data segment
  .DATA
  MSG DB 'HELLO!$'

;Add the following instructions in the code segment
  LEA DX, MSG
  MOV AH, 9
  INT 21H
```

Assignment

Will be announced in the lab.