Objective: Implementation of Interrupt mechanism into the instruction cycle

The prerequisite for a Multi Stream BATCH environment is a Multi-Programming environment. Multiprogramming is an environment in which more than one program exists in the memory at a time and any of them can be executed under the control of a resident OS. Historically, first multi-programmed operating systems were Multi-Stream Batch Monitors.

To enable the basic computer system with multiprogramming capability, its instruction cycle must be modified to allow that.

This tutorial is designed to demonstrate such a capability.
Follow the following sequence:

i) Load the debug program DEBUG thus:

```
C:\debug
-
```

ii) You will see the DEBUG CURSOR as a blinking hyphen "-".Tell the debug to start assembling a program starting from location 100 in the memory. To do that, use A100 command.

NOTE: Here the SEGMENT ADDRESS STARTS AT 0AD3:
It may be different in your PC.

NOTE: The symbol "↵" represents "Press the Enter Key". It is not part of your entries!

```
-A100↵
0AD3:0100   MOV   DX,0200   ↵
0AD3:0103   MOV   AH,09
0AD3:0105   INT   21        ↵
0AD3:0107   MOV   AH,07     ↵
0AD3:0109   INT   21        ↵
0AD3:010B   MOV   DL,AL     ↵
0AD3:010D   MOV   AH,02     ↵
0AD3:010F   INT   21        ↵
0AD3:0111   JMP   0107      ↵
```

NOTE: READ THIS!!

Explanation of the above code:
A100: Instructing the IMBEDDED ASSEMBLER in the DEBUGGER to start assembling a program in the memory starting at absolute memory address 100 (relative to its current segment denoted by CS Register)
MOV DX, 0200: Point to Memory location 200
MOV AH, 09: Initialise AH with Function 9 of interrupt 21. This function will start display of a string of characters already stored in location pointed to by DX register. It continues output to the display till it encounters a **$** character. So it recognises the **$** character as the "END OF STRING OUTPUT".
INT 21: This **system call** is a set of **Basic Input and Output functions**. Functions are numbered in HEX and each provide an Input or Output service. When a service is needed, the programmer puts the function number in AH register and **INVOKES** INT 21 in subsequent instruction (to call the function). Sometimes, other registers are involved to return parameters in return from the call or pass parameters to the function during the call. In this case we are invoking String Output function 9 at RUN TIME.

iii) Now, let's enter a string of characters into memory starting at location 200 and end it with the $ character. To do so, use the "e" command (stands for ENTER), of the debug (observe the syntax).

```
-e200 "I READ A CHARACTER AND DISPLAY IT. I CAN'T BE INTERRUPTED! $" ↵
-
```

iv) Let's use the "d" (stands for DUMP), command of the debug to see the content of memory block starting at location 200 up to the first $ we encounter.

```
-d200 ↵
```

And the debug will display 16 characters per line showing the HEX and ASCII representation of the memory content with starting address information at the beginning of each line. Note that the debug has entered your string as entered.

What is the out that you see after you enter the above command? It will display the message that you have written in the e200 command right?

VIII) Now give a name to this program and save it. (This example is save in a floppy). Use the "n=" command of the debug (stands for NAME a FILE), and specify a VALID Drive name followed by the Path name (if any), followed by the File Name. Follow EXACTLY the steps bellow to avoid any unpredictable result!

```
-N=C:\USERS\YOUR_ID\NO_INT.COM ↵      < RESERVES A FILE INYOUR FOLDER >
-RCX                           ↵      < TO SET THE CX (datum REG.)TO THE SIZE OF THE FILE >
CX 0000                               < THE CX ONTENT IS ONLY 0 BYTE >
:300                           ↵      < WE ENTER 300 TO SET IT FOR 300 BYTES >
-RBX                           ↵      < SEE THE VALUE OF BASE COUNTER >
BX 0000                               < IT IS 0. DON'T CHANGE IT! ! >
:                              ↵      < JUST PRESS ENTER TO LEAVE IT AT 0 >
-W                             ↵      < NOW WRITE IT ON THE FLOPPY DISK >
Writing 00300 bytes                   < IT IS CREATING THE FILE IN FLOPPY IN A: DRIVE >
```

IX) To check if the file exists in drive a, do the following:
Use the "q" command to quit the debug and enter the command prompt. Then use the "dir" command to see the file info. Thus:

```
-Q      ↵  <QUIT THE DEBUGGER>

C:\DIR  ↵  <CHECK TO SEE THE FILE IS ON YOUR FOLDER >
 Volume in drive A has no label.
 Volume Serial Number is 7CA5-6E80

 Directory of A:\
07/08/01  12:31p             768 no_int.com       <HERE IT IS !>
        1 File(s)          768 bytes
                           697,344 bytes free
```

X) OK! Now let's continue with the debugger and step through the program by inserting the break points and see the internal state of the CPU registers after execution of each instruction.

We first load our program from floppy A: and step through it. To do so, at command prompt C:\ enter the command "debug" followed by the full name of the file. Then at the "debug" prompt, enter the command "U100" to see the listing of your program. (U command stands for "UNASSEMBLE"). Thus:

```
C:\WINNT\system32> DEBUG C:\USERS\YOUR_ID\NO_INT.COM ↵   <Load our programme>
-U100 ↵   <Unassembled the program>
0AD3:0100 BA0002        MOV     DX,0200
0AD3:0103 B409          MOV     AH,09
0AD3:0105 CD21          INT     21
0AD3:0107 B407          MOV     AH,07
0AD3:0109 CD21          INT     21
0AD3:010B 88C2          MOV     DL,AL
0AD3:010D B402          MOV     AH,02
0AD3:010F CD21          INT     21
0AD3:0111 EBF4          JMP     0107
```

Use command "R" (stands for show the registers) to examine the register's before we step through the program. Then use the "G=" command followed by the start and end address of the instruction(s). Thus:

```
-R↵
AX=0000  BX=0000  CX=0300  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=0AD3  ES=0AD3  SS=0AD3  CS=0AD3  IP=0100   NV UP EI PL NZ NA PO NC
0AD3:0100 BA0002       MOV     DX,0200
-G=100 107↵
I READ A CHARACTER AND DISPLAY IT. i CAN'T BE INTERRUPTED!
AX=0924  BX=0000  CX=0300  DX=0200  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=0AD3  ES=0AD3  SS=0AD3  CS=0AD3  IP=0107   NV UP EI PL NZ NA PO NC
0AD3:0107 B407         MOV     AH,07
--
```

Observe that the system call (function 9) is executed and the string is output.

NOTE: READ THIS!!!

If we re-examine the code we will observe that from location 107 to 111 the code represents an INFINITE LOOP. It calls two functions (function 7 and 2 of interrupt 21) and then at 111 executes an un-conditional jump to location 107.
The service that function 7 provides is:

- Disable the interrupt mechanism
- Wait until a key is depressed (entered from the key board)
- Once the key is entered, put its ASCII CODE into register A and RETURN to the CALLING POINT in the caller program

The service that function 2 provides is:

- Display the character that is already in DL register, and return to the calling point.

(Observe that we have moved the AL to DL in the instruction "MOV DL, AL" at location 10B to prepare for a call to the function 2 and after we have returned from the function 7 that has returned the entered key in AL).

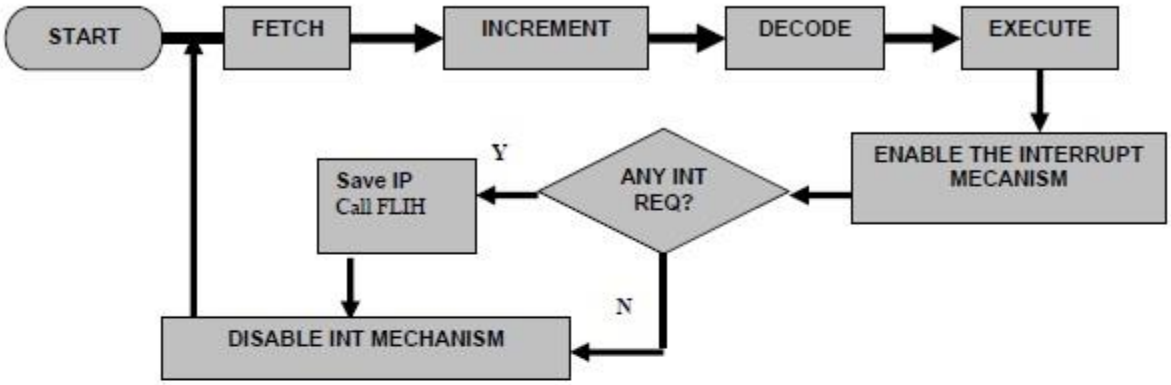
So, let's execute the loop. During the execution you will see that the program reads a character and display it. It cannot be interrupted. To interrupt the program you have to use CTRL+ ALT + DEL to terminate the program by TASK MANAGER.

```
-G=100 107
I READ A CHARACTER AND DISPLAY IT. I CAN'T BE INTERRUPTED!
AX=0924 BX=0000 CX=0300 DX=0200 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0AD3 ES=0AD3 SS=0AD3 CS=0AD3 IP=0107   NV UP EI PL NZ NA PO NC
0AD3:0107 B407       MOV     AH,07
-G=107 113
I am entering and cannot break with CTRL♥ and CTRLB●. I am trapped in loop!!!!!!
```

For now, the only way for you to terminate the program is to click the close button on the window. In order to make the program be able to close properly, change function 7 to 8 like the following code. Save the program as WITH_INT.COM

```
0AD3:0100 BA0002        MOV     DX,0200
0AD3:0103 B409          MOV     AH,09
0AD3:0105 CD21          INT     21
0AD3:0107 B407          MOV     AH,08
0AD3:0109 CD21          INT     21
0AD3:010B 88C2          MOV     DL,AL
0AD3:010D B402          MOV     AH,02
0AD3:010F CD21          INT     21
0AD3:0111 EBF4          JMP     0107
```

To be able to break the loop the INSTRUCTION CYCLE MUST BE MODIFIED to entertain an INTERRUPT REQUEST at the end of each cycle. Such an instruction cycle looks like:



Remember:

1. FETCH ==> Get the instruction addressed by IP in the memory and load it into MBR , thus:
   - 1.1 IP ==>MAR
   - 1.2 [MAR] ==> DATA BUS
   - 1.3 DATA BUS ==> MBR
2. INCREMENT ==> Increment the IP by the increment factor interpreted from the instruction's OPERATION CODE (OP CODE) to get ready for next fetch, thus:
   - 2.1 IP = IP + <INCREMENT FACTOR>
3. DECODE ==> Send the Instruction code in the MBR to IR so that IR interpret it and generate the EXECUTION CONTROL SIGNALS needed for its execution, thus:
   - 3.1 MBR ==> IR
   - 3.2 IR ==> <Produce the micro code needed for the execution control at the IR's output>
4. EXECUTE ==> Send the IR's Output to the Sequencer to starts the execution sequence, thus:
   - 4.1 IR ==> Sequencer

AND WE NEED TO ADD MORE DETAILS HERE! WE DO THAT LATER!

Now that the instruction cycle allows, a PROGRAM CAN BE WRITTEN in such a way that the external processes can communicate with it by INTERRUPTING IT TEMPORARILY. With such a machine we can implement MULITIPROGRAMMING ENVIRONMENT and start developing MULTI STREAM BATCH OPERATING SYSTEMS.

XI) **Function 8 of interrupt 21 is exactly like function 7 but it ENABLES the interrupt mechanism after reading a character and putting it in AL register. It allows the program be interrupted if CTRL+C or CTRL+B (in some machines) is depressed at any time.**

XII) To demonstrate this, Download the program named WITH_INT.COM and by now you should be able to step through it using the debug.