CSI 404 Computer Organization | Spring 2001 | Professor Chaiken
Homework 8 | Assigned: April 6 | Due: April 23**(Monday!)**, 10:10 AM

Announcement: Exam 2 will be given on April 18, 10:10-11:10. Overflow room to be announced.

Problem 1: Write a non-blocking input subroutine named `PREADN` (for parallel). Example of use:

```
          BSA     PREADN
          HEX     MY_BUFFER  /Pointer to input area
          HEX     n          /Number of characters to read
          HEX     BUSY_FLAG  /Pointer to one word that PREADN sets to
                             /a non-zero value and the interrupt
                             /service routine sets to zero when all n
                             /characters are put in the input area
          ...                /PREADN returns here
          ...                /useful work in parallel with reading
          ...
AGAIN,    LDA    BUSY_FLAG   /busy wait on BUSY_FLAG==0
          SZA
          BUN    AGAIN
          ...                /program can use input now
          ...
MY_BUFFER,                   /at least n words
BUSY_FLAG,  HEX   ---        /user program provides flag location
          ...
PREADN,   HEX     ---        /YOU write!
          ...
          BUN    PREADN I    /Return from PREADN
          ...                /Local variables belonging to PREADN
          ORG    0
ZERO,     HEX    ---         /YOU write an interrupt service routine
          ...
          ION
          BUN    ZERO I
```

a. Again, write the algorithm for your PREADN subroutine and your interrupt service routine in clear English.

b. Write the assembly code with ample comments.

c. Analyze what happens with your program if a person presses a key while the PREADN operation is not in progress.

Practice Problem A:

a. The interrupt service routine on page 207 always tests FGI with the SKI instruction and if FGI=1, always executes an INP instruction (which copies the input character to AC and resets FGI). Suppose we want to design I/O software that does not put an input character into a user buffer until the user program requests input. One idea is to write an interrupt service routine that executes the INP instruction only under the condition that the variable NToRead (number of characters that remain to be read) is non-zero. Explain why this idea won't work. Hint: Why must the interrupt service routine **always** execute an INP instruction when FGI=1?

b. Consider the interrupt service routine on page 207. Suppose the output device is so fast that it completes the output operation and sets FGO (to **1**) while the computer is still executing the interrupt service routine. Explain in detail what happens, clock step by clock step (refer to chapter 5), when the computer executes the ION and BUN ZRO I instructions and then the following few clock steps when a new interrupt is taken. Why must the system programmer write the BUN ZRO I instruction **immediately after** the ION instruction? (Hand in a solution this week too. If you had gotten a correct and complete answer on Hw. 5, you will get extra credit on Hw. 5.)