F.8 Chapter 8

8.1 (a) A device register is a register (or memory location) that is used for data transfer to/from an input/output device. It provides a means of communication between the processor and the input/output device. The processor can poll this register to find out whether it has received an input or it can send an output from/to the specific device that the device register belongs to. In memory mapped I/O device registers are dedicated memory locations for each I/O device. There may be more than one device register (dedicated memory location) for one device.

(b) A device data register is a device register (a dedicated memory location in memory-mapped I/O) that holds the data that is to be input/output.

(c) A device status register is a device register (a dedicated memory location in memory-mapped I/O) that indicates the status of the input/output. It allows for the processor to know whether or not input/output of the value in the device data register has occurred. Basically it is an important step to achieve synchronization in an asynchronous I/O system.

8.2 A ready bit is not needed if synchronous I/O is used because the processor will know exactly when the data will arrive and when it will be taken away (input and output). It will do input and/or output at regular intervals, and it will be guaranteed that during those intervals the input data is taken by the computer and the output data goes to the output device.

8.3 The processor can accept a character every clock cycle at its maximum rate. This means that a 300 MHz processor can accept a character each 1/(300M) seconds.That is this processor can accept a character every 3.333 nanoseconds which corresponds to a rate of 18 billion characters per one minute. This is the maximum rate it can accept input in one minute. If the typist would have to type 3 billion words per minute to synchronize with this maximum rate, then a word must be 18/3 = 6 characters long. (This, of course, counts the *space* between words as one of the characters in the word!)

8.4 (a) The interaction between a remote control and a television is synchronous. The television samples at specific intervals to see if a key on the remote control has been pressed. No synchronization is needed in this transaction.

(b) The interaction between the mail delivery person and you is asynchronous. Neither do you check your mail at regular intervals, nor does the mail delivery person come at the same time everyday. Instead you use the mailbox as a synchronization mechanism (much like the ”Ready bit”). Some mailboxes are located at the street, rather than at the door. They usually come equipped with a flag that the mail delivery person lifts when depositing mail, and you lower when removing mail. The flag is very much a ready bit.

(c) The interaction between a mouse and the PC is synchronous. The PC samples mouse movements at specific intervals. At each interval, the direction and speed of the mouse is read by the PC. No synchronization is needed in this interaction.

8.5 Bit [15] of the KBSR is the ready bit. This is used as the synchronization mechanism to let the processor know that input has occurred. If KBSR[15] is 0, no key has been struck and the value in KBDR is not valid. If KBSR[15] is 1, the value in KBDR is the ASCII code corresponding to the last key struck.

8.6 If KBSR[15] is 0, the data contained in the KBDR has already been read. The program would read the same character again.

8.7 Memory mapped and polling. The system is memory-mapped because KBSR and KBDR device registers have assigned addresses in the memory address space of the ISA. The system is polling because the Ready bit is tested to see if a key has been struck.

8.8 Check if the value in memory location x4000 is between 0 and 127.

```
START     LDI     R0,A
          BRn     DONE      ;Branch if value is less than 0
          LD      R1,B
          ADD     R1,R0,R1
          BRp     DONE      ;Branch if value is above 127
          OUT
DONE      HALT
A         .FILL   x4000
B         .FILL   #-127
```

8.9 If KBSR[15] is 1, the data contained in the KBDR has not been read by the processor. Thus, if the keyboard hardware does not check the KBSR before writing to the KBDR, user input could be lost.

8.10 The display device is an output device and can hence not write to the DDR.

8.11 Interrupt-driven I/O is more efficient than polling. Because, in polling, the processor needs to check a specific register (or memory location) regularly to see if anything is being input or output. This consumes unnecessary processing power because the processor checks the register periodically (stopping all other jobs) even when nothing is being input or output. (Most of the time the register will not be inputting or outputting anything unless it is a really I/O-intensive program). However, in interrupt-driven I/O, when something is input or output by a device, the device sends a signal to the processor. Only when the processor receives that signal, it stops all other jobs and does the I/O. Hence, processing power is used for I/O only when it is necessary to do so.

8.12 Assume that the KBDSR is assigned to address xF400.

```
START    LDI         R0, A
         BRz         START      ;Branch if KBDSR
                                ;contains zero
         AND         R1, R1, #0
         STI         R1, A      ;Clear the KBDSR
         BR          NEXT_TASK  ;Goto the next task
A        .FILL       xF400
```

8.13 Suppose the LC-3 datapath allows combining the two registers into one. Using separate registers, the test to see if the Ready bit is set simply involves checking bit 15 of the status register. This is performed using a branch instruction that tests if the value of the register is negative. If the KBSR and DSR are combined, the test to see if the Display device Ready bit is set involves masking out bit 14 and testing if the bit is set or cleared. Doing it this way requires more instructions than the first method.

8.14 The address control logic takes care of this. It accesses the KBDR if the address is xFE02. For the user, this access to KBDR looks like a normal load instruction.

8.15 NOTE: Please refer to the errata for the new problem statement.

(a) The keyboard interrupt is enabled, and the digit 2 is repeatedly written to the screen.

(b) The character typed is echoed twice to the screen.

(c) The digit 2 some number of times, followed by the digit typed twice or three times, followed by the digit 2 continually thereafter.

(d) The digit typed will be displayed to the screen twice or three times, depending on when the typed character interrupted the program. If the program was interrupted immediately after LD R0, B , the character typed would appear on the screen three times.

8.16 This program outputs ABCDEFGHI.