9.2 9.6 9.10 9.14 9.26

9.2

Why is a ready bit not needed if synchronous I/O is used?

If the typist could type at a constant speed, and we did have a piece of hardware that would accept typed characters at precise intervals (e.g., one character every 200 million cycles), then we would not need the ready bit. The computer would simply know, after 200 million cycles of doing other stuff, that the typist had typed exactly one more character, and the computer would read that character. In this hypothetical situation, the typist would be typing in lockstep with the processor, and no additional synchronization would be needed. We would say the computer and typist were operating synchronously. That is, the input activity was synchronous.

If IO is synchronous , CPU will know when to get a character from keyboard or send a character to monitor by hardware that accept typed characters at precise intervals. Ready bit is not needed because CPU already know when to do IO operations

9.6

What problem could occur if a program does not check the ready bit of the KBSR before reading the KBDR?

When a key on the keyboard is struck, the ASCII code for that key is loaded into KBDR[7:0], and the electronic circuits associated with the keyboard automatically set KBSR[15] to 1. When the LC-3 reads KBDR, the electronic circuits associated with the keyboard automatically clear KBSR[15], allowing another key to be struck. If KBSR[15] = 1, the ASCII code corresponding to the last key struck has not yet been read, and so the keyboard is disabled; that is, no key can be struck until the last key is read.

don't check KBSR may cause :

if KBSR[15] =1 ,means the last character hasn't been read,and if keyboard isn't disabled,unread character will be replace , it causes information missing

9.10

What problem could occur if the display hardware does not check the DSR before writing to the DDR?

DSR[15] controls the synchronization of the fast processor and the slow monitor display. When the LC-3 transfers an ASCII code to DDR[7:0] for outputting, the electronics of the monitor automatically clear DSR[15] as the processing of the contents of DDR[7:0]begins. When the monitor finishes processing the character on the screen, it (the monitor) automatically sets DSR[15]. This is a signal to the processor that it (the processor) can transfer another ASCII code to DDR for outputting. As long as DSR[15] is clear, the monitor is still processing the previous character, so the monitor is disabled as far as additional output from the processor is concerned.

if DSR isn't check ,and if the last character isn't shipped by the processor ,in other word ,data to display is not ready,and just display it will cause unexpected error.

for example

DSR[15] = 0 ,the character read yet not displayed completely would be missing because display hardware don't check DSR then skip last unfinished task.

9.14

An LC-3 Load instruction specifies the address xFE02. How do we know whether to load from the KBDR or from memory location xFE02?

In the case of memory-mapped output, the same steps are carried out, except ***instead of MAR being loaded with the address of a memory location****, MAR is* ***loaded with the address of a device register****. Instead of the address control logic enabling memory to write, the address control logic asserts the load enable signal of DDR*

Memory-mapped *output also requires the ability to read output device registers. You saw in Section 9.2.3.2 that before the DDR could be loaded, the ready*

bit had to be in state 1, indicating that the previous character had already finished being written to the screen. The LDI and BRzp instructions on lines 01 and 02 perform that test. To do this, the LDI reads the output device register DSR, and BRzp tests bit [15]. If the MAR is loaded with xFE04 (the memory-mapped address of the DSR), the ***address control logic selects DSR as the input to the MDR****, where it is subsequently loaded into R1, and the condition codes are set*

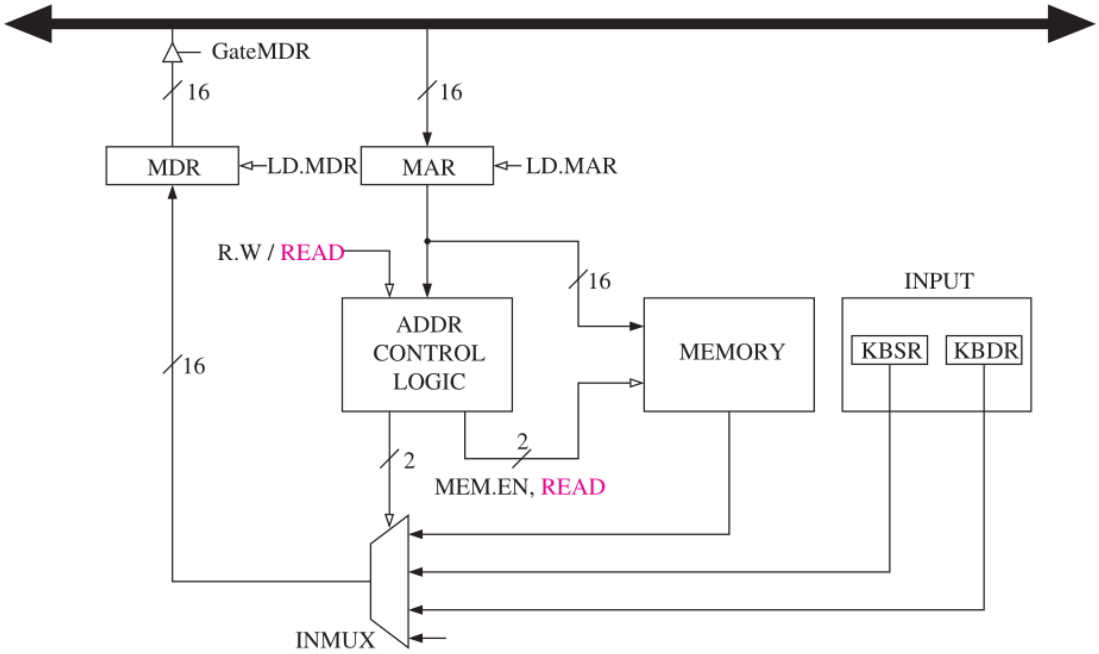

Figure 9.4 Memory-mapped input.

The **Addr control logic** will receive the address xFE02 and it will figure out that this address belongs to IO page ,so it can make use of the INMUX to let the content in KBDR be the output

9.26

The following program is supposed to print the number 5 on the screen.
It does not work. Why? Answer in no more than ten words, please.

```
.ORIG x3000
JSR A   1 R7 = &OUT
OUT
BRnzp DONE

A
AND R0,R0,#0
ADD R0,R0,#5
JSR B   2 R7 = &RET
RET     4 PC = R7 LOOP...

DONE HALT

ASCII .FILL x0030

B LD R1,ASCII
ADD R0,R0,R1
RET     3 PC = R7
.END
```

R7 value is overwritten due to not Saved before.