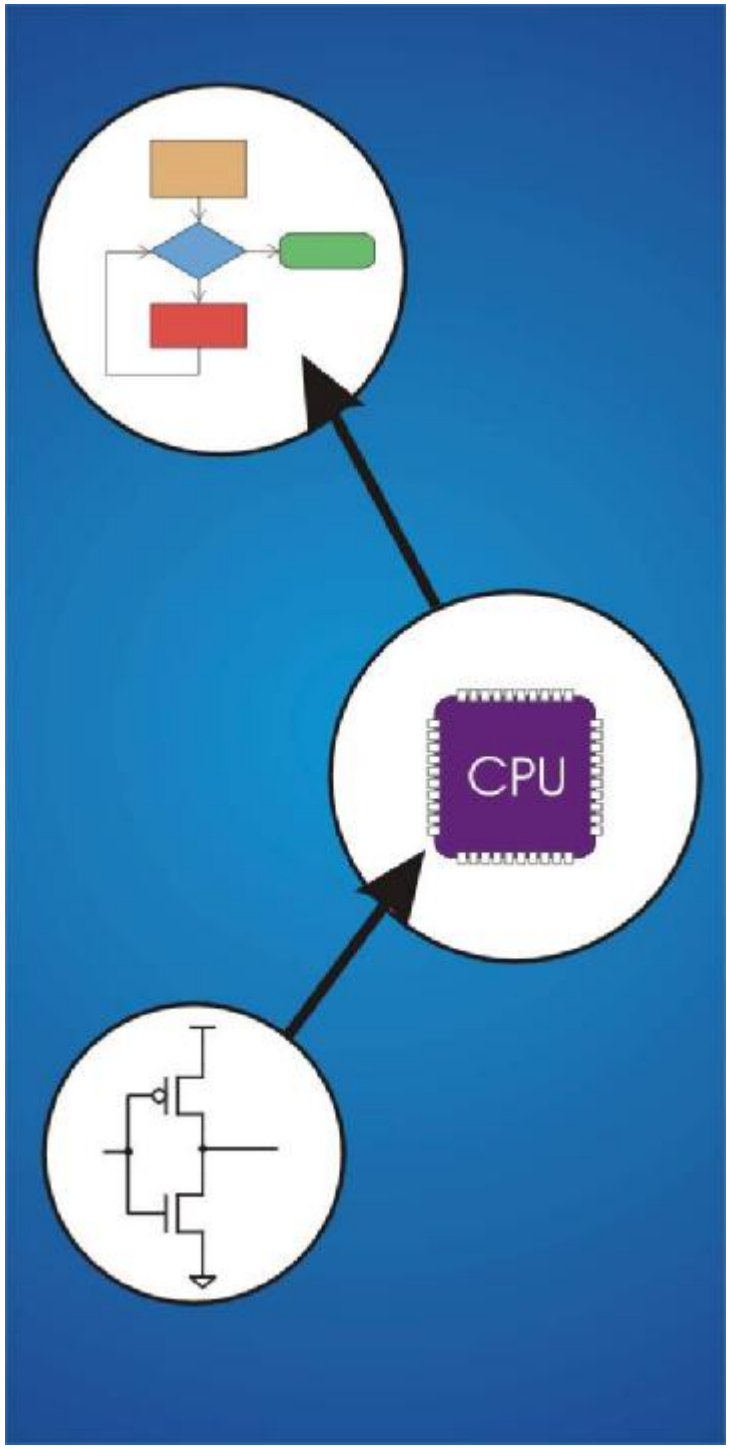


第八章
输入/输出

输入/输出: 与外界相连接

到目前为止，我们已经学习:

- 利用寄存器中的值进行计算
- 将数据从内存装载到寄存器
- 将数据从寄存器存储到内存

但是内存中的数据是从哪里来的呢？

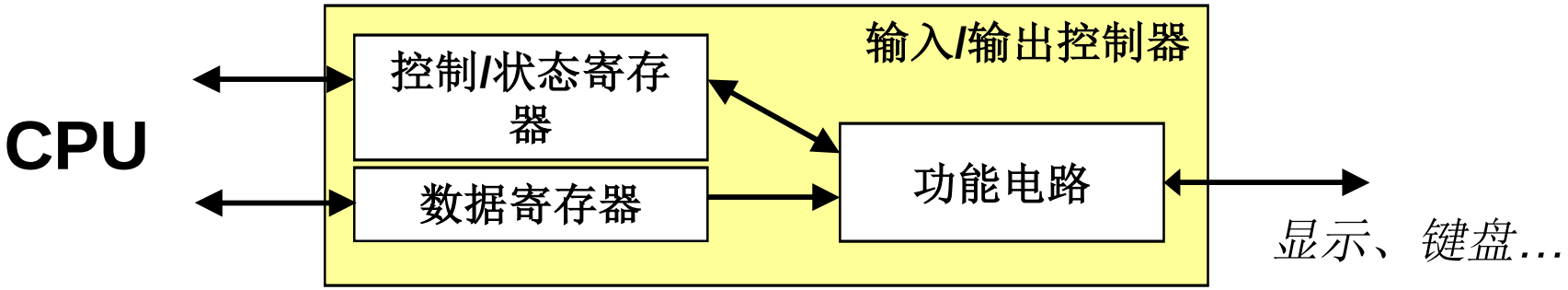
而且， 人们使用的数据是怎么从系统中传出去的呢?

输入/输出: 与外界相连接

I/O设备种类按以下特征分为几类:

- 功能: 输入, 输出, 存储设备
  - Ø输入: 键盘, 运动检测器,网络接口
  - Ø输出: 显示器, 打印机, 网络接口
  - Ø存储: 软盘, 光盘
- 速率: 数据能传输的多快?
  - Ø键盘: 100 bytes/sec
  - ØU盘: 30-40 MB/s
  - Ø网络: 1 Mb/s - 1 Gb/s

和CPU内部总线连接：输入/输出控制器



控制/状态寄存器

- CPU告诉设备做什么-- 写控制寄存器
- CPU 检查任务是否已经完成 – 读状态寄存器

数据寄存器

- CPU 通过数据寄存器从设备读/写数据

功能电路

- 执行实际操作
  - Ø输出像素到屏幕, 比特流入/出 软盘, 来自键盘的字符流

编程接口

怎么访问设备内部寄存器?两种编址方式

- 内存-映射(寄存器作为内存的一部分) vs. 专用IO指令(寄存器具有独立地址空间，可和内存重叠)

快速处理器和慢速的外部设备之间传输的时序是怎么控制的?

- 异步 I/O vs. 同步 I/O

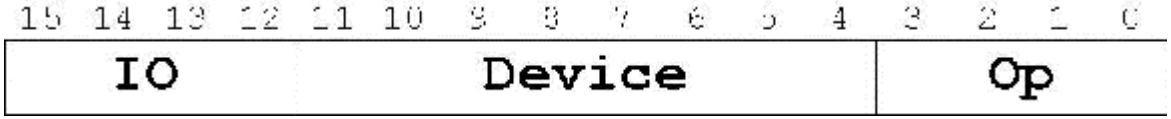
谁控制传输?

- CPU (轮询) vs. 设备 (中断)
- 举例：在家里等客人

编址方式:内存映射 vs. 专用IO指令

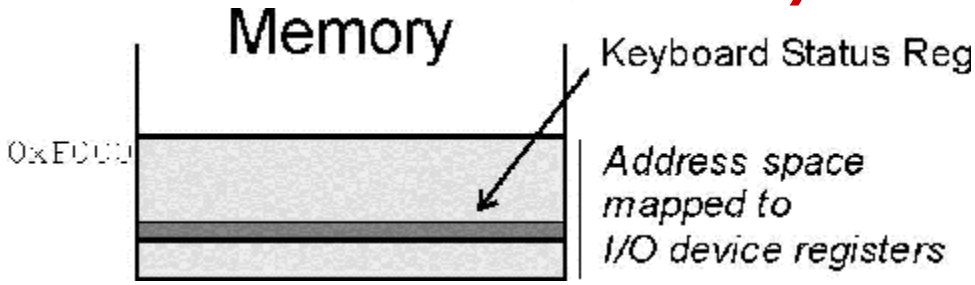
专用指令:为访问外部设备设计专用的指令 (如:X86 的 IN/OUT)

- 为访问外部设备设计专用的指令和操作码 (X86 IN/OUT)
- 在指令中编码要访问的设备寄存器地址和对应的操作码
- 空间和内存可以是重叠的



内存映射(如:ARM,从内存中保留一段空间给外部设备用)

- 给每一个设备寄存器分配内存地址
- 使用和访问内存相同的指令 (Load/Store)访问设备寄存器
  访问速度比访存慢



传输时序控制:异步 I/O 同步 I/O

I/O 事件一般发生的比CPU周期慢得多.

同步

- 数据以一种固定的、可预测的数据供给
- CPU 以某个固定的周期进行读/写操作

异步

- 数据速率不可预测
- CPU必须同步装置，以免遗失数据或者写入太快
- 方法:设置状态寄存器或者标志位,在操作前检查设备是否准备好

谁控制传输:CPU (轮询) vs. 设备 (中断)

谁决定下一个数据传送何时发生?

轮询

- CPU不断检查状态寄存器，直到新的数据到达
  或者设备已经为下一个数据做好准备
- “客人到了没?客人到了没?客人到了没?”
- 缺点：CPU和外设串行工作,利用效率低.

中断

- 当新数据到达或者设备已经为下一个数据做好准备时，设备会发送一个特殊信号到CPU
- CPU在此期间可以执行其他任务，而不是反复轮询.
- “当客人到达时请通知我.”
- Cpu和外设可并行工作,利用效率高,但需要专用中断硬件支持.

LC-3 的IO机制

内存映射的输入输出 (Table A.3)

地址	*输入/输出寄存器*	*作用*
xFE00	键盘状态寄存器(KBSR)	第十五位为1当键盘收到新的字符.
xFE02	键盘数据寄存器(KBDR)	第0到7位包含键盘上打出最后一个字符.
xFE04	显示输出状态寄存器(DSR)	第十五位为1当设备准备好,可向屏幕显示一个字符.
xFE06	显示输出数据寄存器(DDR)	写入到第7到0为的字符将显示在屏幕上.

异步装置

- 通过状态寄存器进行同步

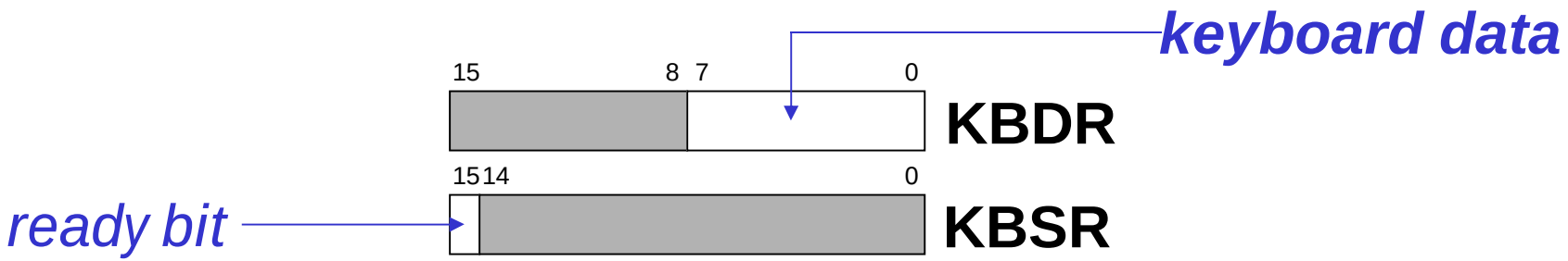
轮询 和 中断

- 中断的实现细节将会在第10章讨论

LC-3键盘输入机制

当按下一个字符时:

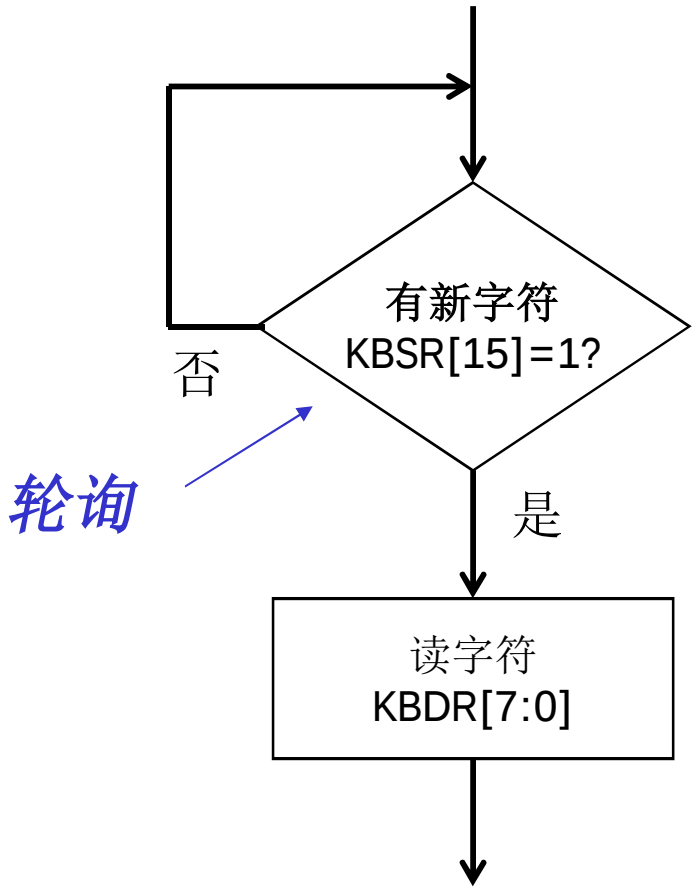
- 1)其ASCII码放置在KBDR的比特[7:0]位(比特[15:8]位总是0)
- 2)“就绪位” (KBSR[15]) 被设置为1
- 3)键盘被禁用– 任何输入的字符都会被忽略



读键盘数据寄存器(KBDR)数据:

- 1）检测KBSR[15] 是否为1，为1则读取KBDR[7：0]
- 2) KBSR[15] 设置为0
- 3) 键盘置为使能,准备接收下一个字符

基本输入程序

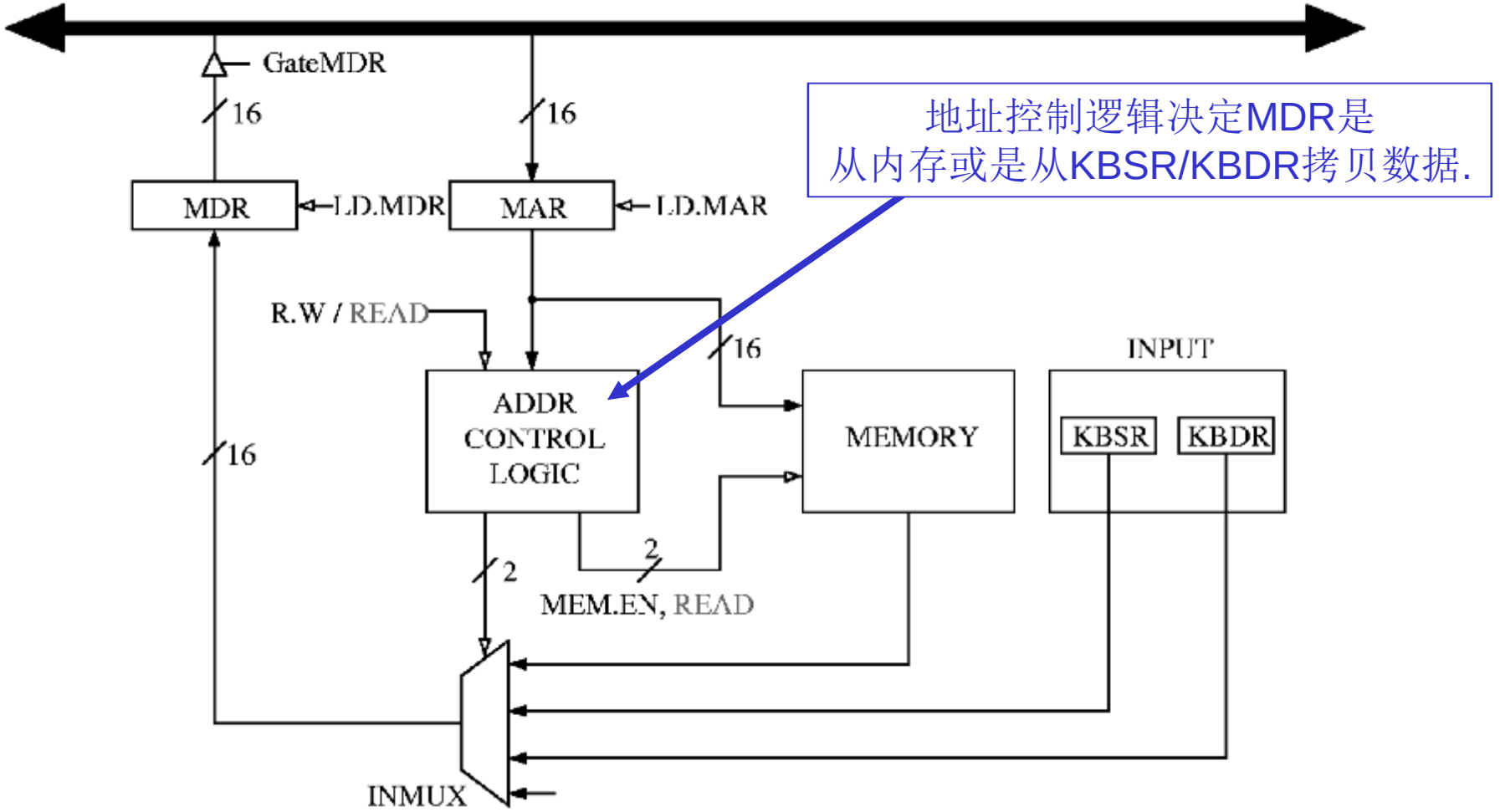



```
POLL    LDI  R0, KBSRPtr
        BRzp POLL
        LDI  R0, KBDRPtr

        ...

KBSRPtr .FILL xFE00
KBDRPtr .FILL xFE02
```

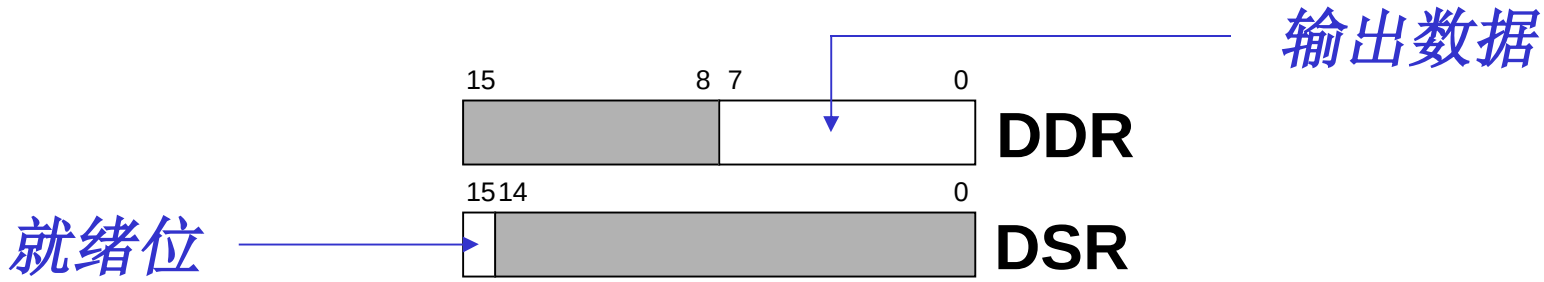
LC-3键盘内存映射的实现



LC-3显示器输出

当显示器准备输出一个字符时:

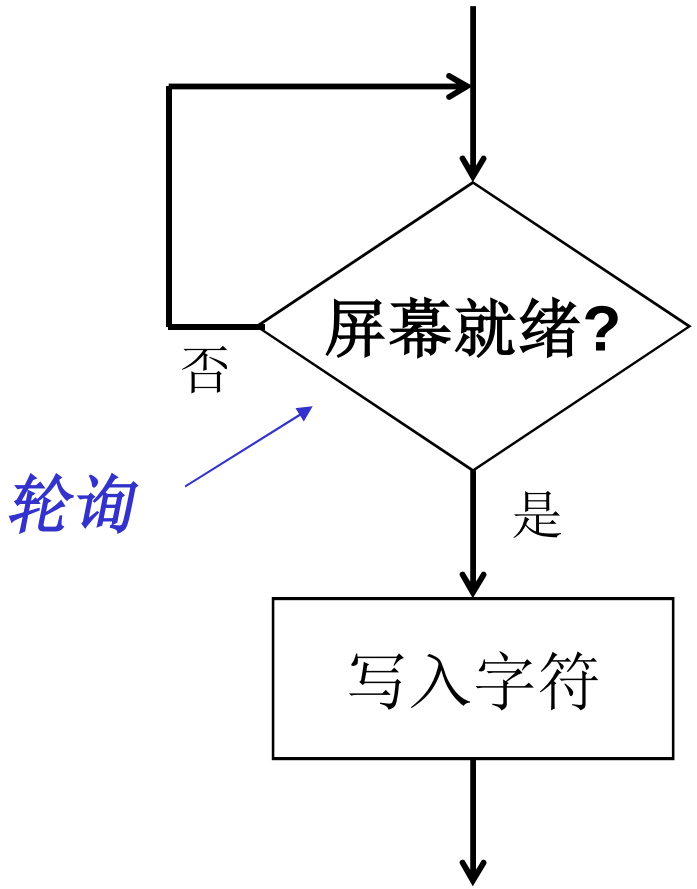
- 就绪位(DSR[15]) 置为1



当就绪时,数据可写入输出数据寄存器:

- DSR[15] 置为0
- 写在DDR[7:0] 中的字符将被输出
- 任何其他写入输出数据寄存器的字符将被忽略(DSR[15]=0期间)
- 显示完成后,就绪位(DSR[15]) 置为1
-

基本输出程序

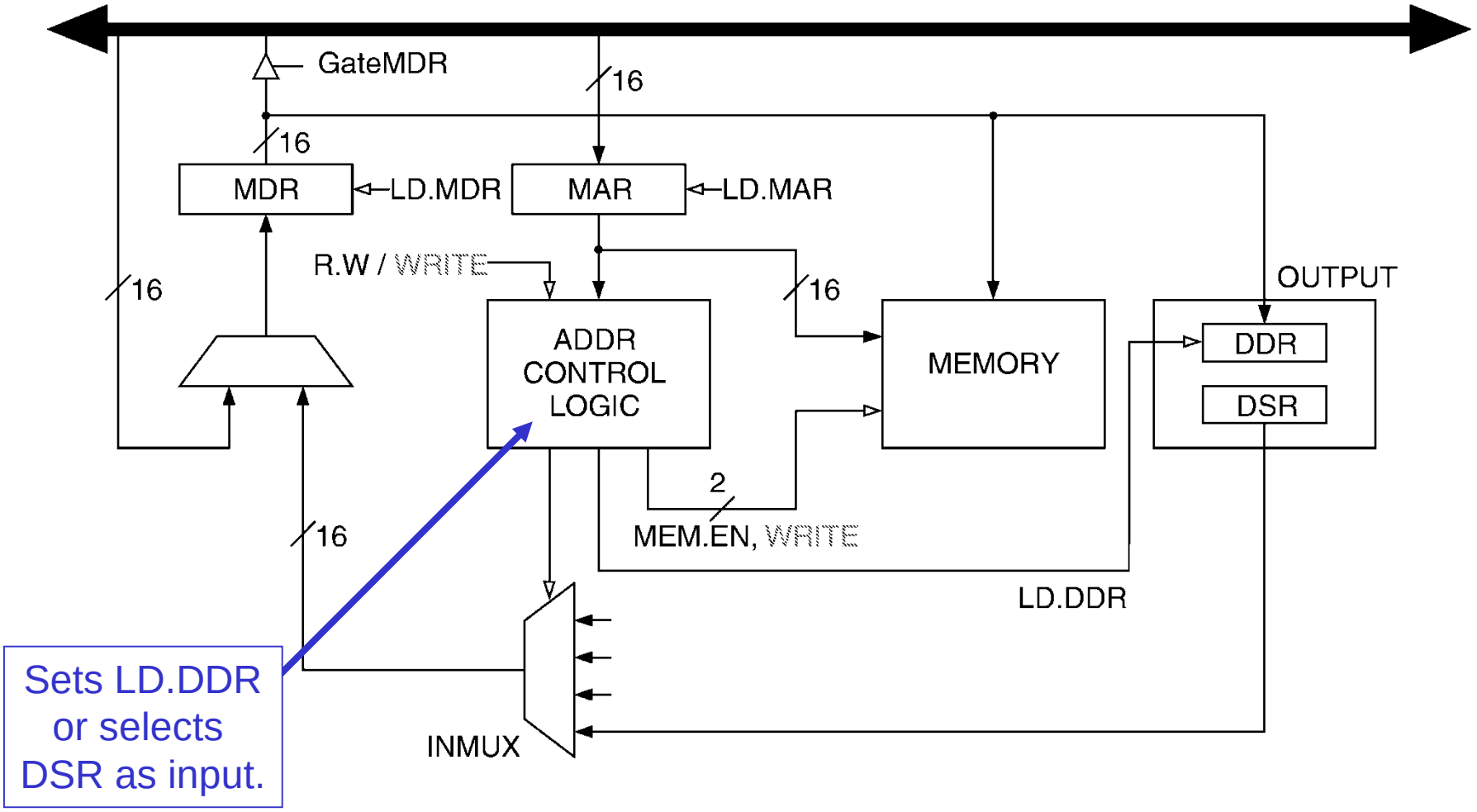



```
POLL    LDI   R1, DSRPtr
        BRzp  POLL
        STI   R0, DDRPtr

        ...

DSRPtr  .FILL xFE04
DDRPtr  .FILL xFE06
```

LC-3显示内存映射的实现



键盘回显输入程序

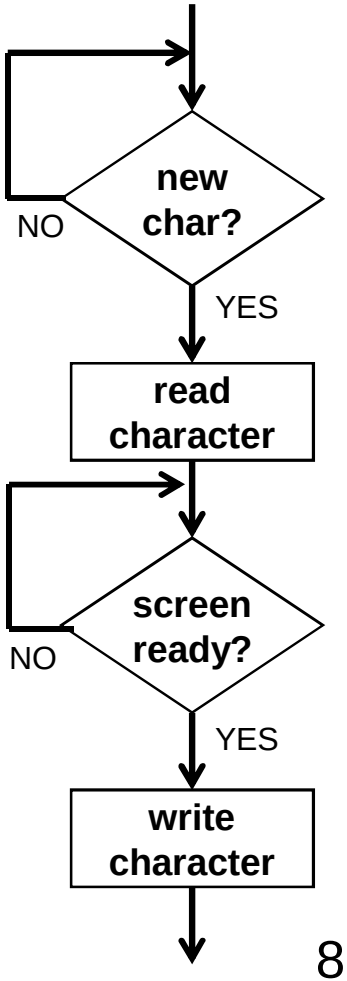
通常, 输入的字符先被显示在屏幕上.

- 用户得到输入字符的反馈, 知道可以输入下一个字符.

```
POLL1    LDI  R0, KBSRPtr
         BRzp POLL1
         LDI  R0, KBDRPtr
POLL2    LDI  R1, DSRPtr
         BRzp POLL2
         STI  R0, DDRPtr

         ...

KBSRPtr  .FILL xFE00
KBDRPtr  .FILL xFE02
DSRPtr   .FILL xFE04
DDRPtr   .FILL xFE06
```



P140. LC-3完整的键盘输入历程

```
01  START   ST      R1,SaveR1   ; Save registers needed
02          ST      R2,SaveR2   ; by this routine
03          ST      R3,SaveR3
04  ;       LD      R2,Newline
05          LDI     R3,DSR      ; Loop until monitor is ready
06  L1      BRzp    L1          ; Move cursor to new clean line
07          STI     R2,DDR
08  ;       LEA     R1,Prompt   ; Starting address of prompt string
09          LDR     R0,R1,#0    ; Write the input prompt
0A  Loop    BRz     Input       ; End of prompt string
0B          LDI     R3,DSR
0C  L2      BRzp    L2          ; Loop until monitor is ready
0D          STI     R0,DDR      ; Write next prompt character
0E          ADD     R1,R1,#1    ; Increment prompt pointer
0F          BRnzp   Loop        ; Get next prompt character
10
11  ;
12  Input   LDI     R3,KBSR
13          BRzp    Input       ; Poll until a character is typed
14          LDI     R0,KBDR     ; Load input character into R0
15  L3      LDI     R3,DSR
16          BRzp    L3          ; Loop until monitor is ready
17          STI     R0,DDR      ; Echo input character
18
19  ;
1A  L4      LDI     R3,DSR
1B          BRzp    L4          ; Loop until monitor is ready
1C          STI     R2,DDR      ; Move cursor to new clean line
1D          LD      R1,SaveR1   ; Restore registers
1E          LD      R2,SaveR2   ; to original values
1F          LD      R3,SaveR3
20          BRnzp   NEXT_TASK   ; Do the program's next task
21  ;
22  SaveR1  .BKLW   1           ; Memory for registers saved
23  SaveR2  .BKLW   1
24  SaveR3  .BKLW   1
25  DSR     .FILL   xFE04
26  DDR     .FILL   xFE06
27  KBSR    .FILL   xFE00
28  KBDR    .FILL   xFE02
29  Newline .FILL   x000A       ; ASCII code for newline
2A  Prompt  .STRINGZ ''Input a character>''
```

图8-5 LC-3的键盘输入例程

基于中断驱动的输入/输出

为什么使用中断?

- 轮询占用一定的处理器周期,
  尤其是对稀有事件– 这些周期可以用来处理更多计算.
- 例子: 处理之前的输入，同时收集新的输入. **(See Example 8.1 in text.)**

基于中断,外部设备可以:

(1) 强制当前处理器执行的程序停止;

(2) 使处理器响应设备的需求;

(3) 完成后,处理器恢复停止的程序就像没发生过.

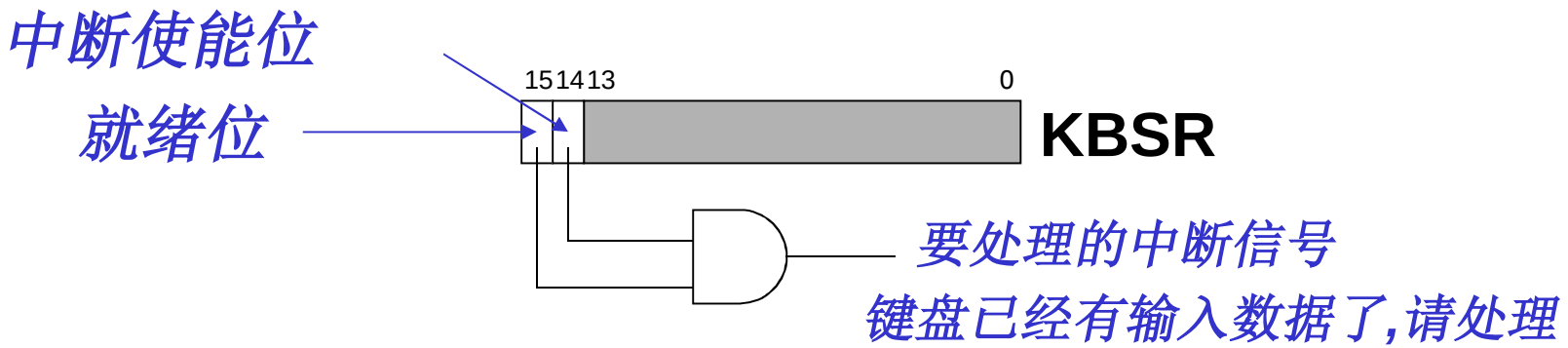
基于中断驱动的输入/输出

为了实现中断机制:

- 一种方式是I/O设备通知CPU一个感兴趣的事件发生了.
- 一种方式是CPU去测试是否信号是否置位而且
  它的优先级是否比当前正在运行的程序高.

中断信号产生

- 软件在设备寄存器中设置中断使能位 。中断使能位=0时设备不产生中断.中断使能位=1时设备可产生中断.
- 当就绪位和中断使能位都置位时，产生中断信号.



优先级

每一条指令在一个规定的紧急等级执行.

LC-3: 8 优先等级 (PL0-PL7)

- 例子:
  - Ø工资计算程序在**PL0**级执行.
  - Ø核能校正程序在 **PL6**级执行.

- 运行在 **PL6**级的设备可以中断**PL0**级的程序 ,反过来却不行.

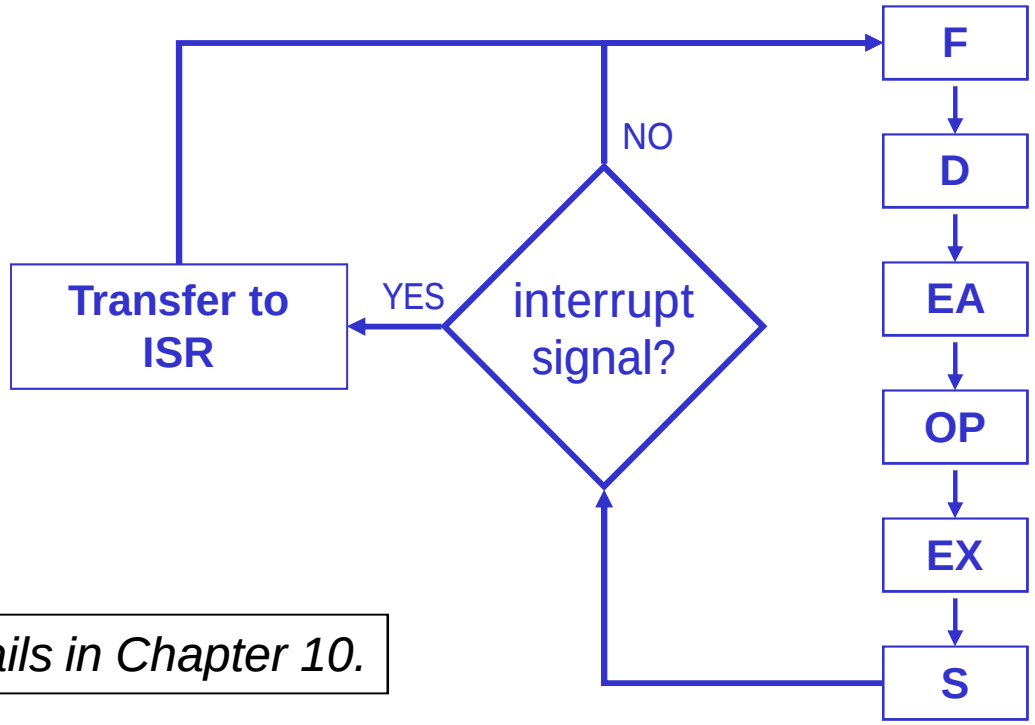
优先级编码器 优先级编码器同当前处理的程序的优先级相比较选择优先级最高的程序，如果允许的话将会产生中断信号.

中断信号的检测

CPU 在指令执行时**S(**写内存**)**阶段和**F(**取指**)**阶段之间检测是否有设备中断信号 **.(**由硬件自动检测**)**

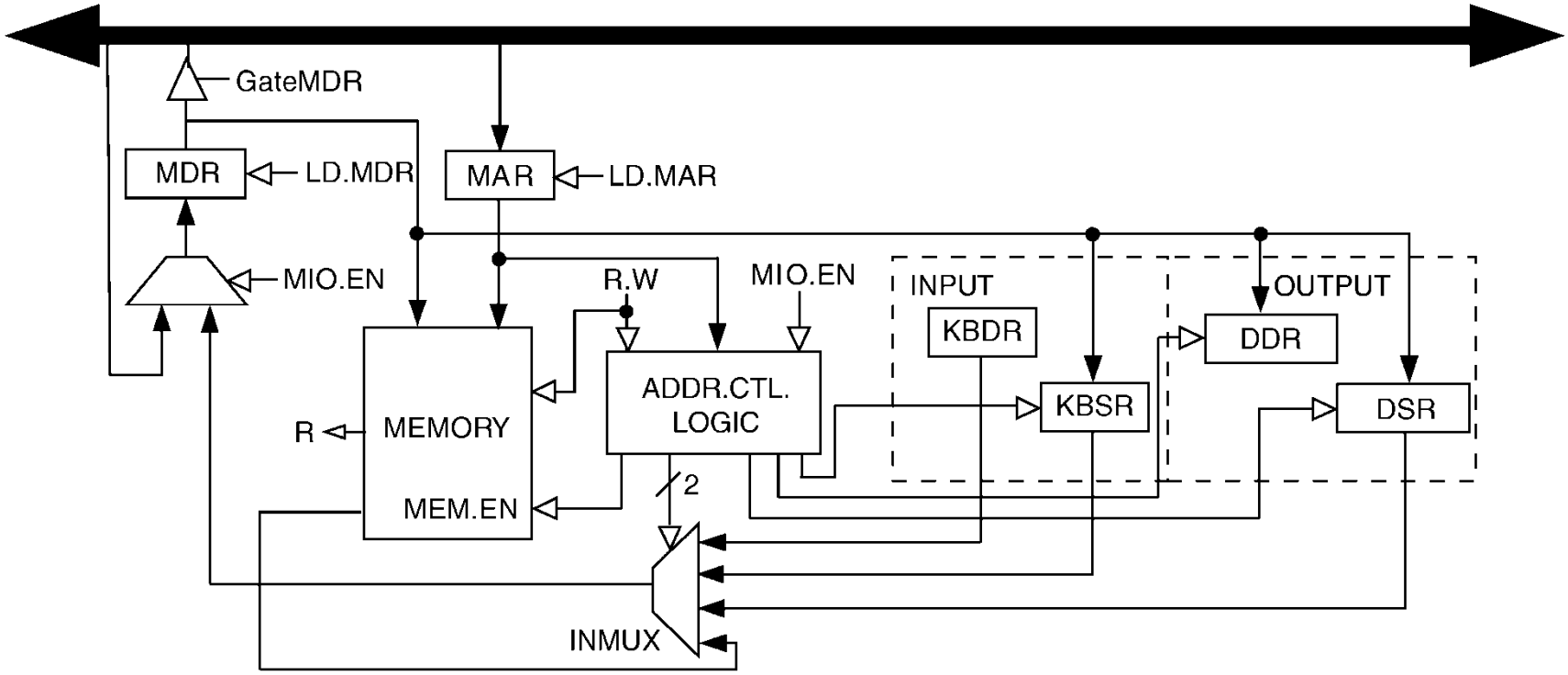
如果没有中断置位, 继续执行下一条指令.

如果有中断置位, 则把控制权交给中断服务程序.



More details in Chapter 10.

LC-3 内存映射输入输出的完整设计实现



因为有中断使能位, 所以状态寄存器必须要能写也要能读(KBSR/DSR)

课堂讨论

1 在没有测试DSR之前就写数据到屏幕会有什么样的风险?

2 在没有测试KBSR之前就从键盘读入数据会有什么样的风险?

3 假如显示器是一个同步设备会怎么样,假定我们知道写一个字符大概需要1毫秒的时间.

- 我们能避免轮询吗? 怎么样避免?
- 利弊又是什么?

4 你认为轮询对其他形式的设备来说是一个好的方式吗,比如软盘和网络接口?

5 用LDI/STI 来访问设备寄存器的优点是什么?

习题

1　使用键盘的状态和数据寄存器 (KBSR 和 KBDR,不允许使用系统调用),编写汇编语言子程序完成以下功能: 从键盘读取一个字符,如果输入是’Y’(x59)或者’y’(x79)则R0寄存器返回1，否则为0。

2 编写**LC-3**汇编语言程序片段**:**

读入一个字符，如果输入字符是空格，则输出‘**Y**‘到显示器，否则输出**’N’**到显示器。空格的**ASCII**码是**x20,’Y’** 的**ASCII**码是 **x59,’N’** 的**ASCII**码是**x4e**。要求使用键盘和显示器的状态和数据寄存器完成，不能使用系统调用。

作业

8.13

8.14

8.15

8.16