

# Introduction to Computer System Organization

---

## SZU Review

## Chapter8

## General

---

### 输入/输出设备

#### 1. 基本原理:

编址方式: 内存映射、独立编址

传输时序控制: 同步、异步

#### 轮询与中断

#### 2. LC-3 Keyboard and Display 的实现 (**Ban IN,OUT**)

(1). KBSR/KBDR 轮询的代码原理

(2). DSR/DDR 代码原理

(3). 中断的工作原理 (**Concept**)

中断通知

中断响应: 保存现场(硬件上多个状态机实现)

中断号 --> 定位入口地址

中断返回: RET

(4). 中断程序格式

### 输入与输出 (Variety)

1. 利用Register中的值进行计算

2. 将data从Memory装载到Register

3. 将data从Register存储到Memory

(I/O)

**Question:**

1. 内存中的数据是从哪里来的?
2. 人使用的数据是怎样通过计算机系统传出去的?

Answer: 外部设备(外设:e.g. 键盘, 显示器)

Rate:

(How fast can the data transform.)

keyboard: 100 bytes/sec

U 盘 : 30 - 40 MB/s (2.0 480 mbps) 5GB/s - 10GB/s (3.1 gen1/gen2)

Internet: 1 MB/s - 1 GB/s - 10 GB/s **Obvious Difference (Variety)**

慢速的外部设备如何接到快速的总线上?

solu: 和CPU 内部总线连接

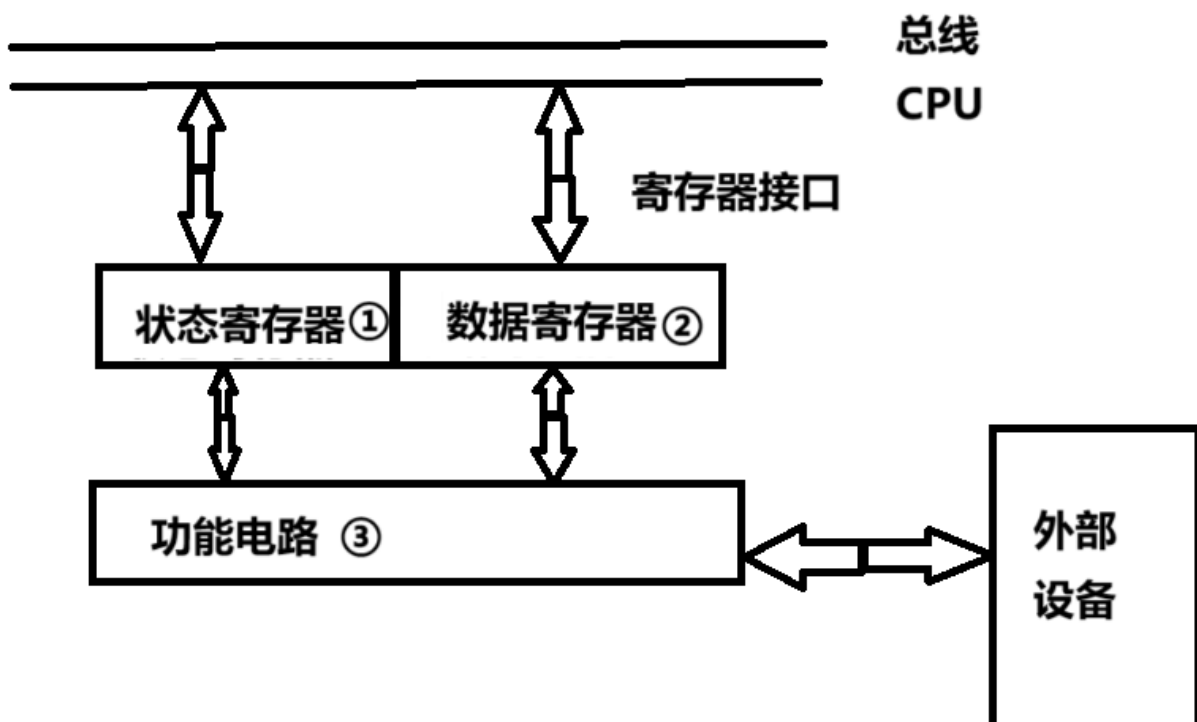
输入/输出控制器:

(与外部设备直接相连)

Function:

1. 格式转换(统一转换为寄存器接口格式的数据)
2. 缓存(不是来一个传一个,而是等达到一定数目一起传)(fit rate!)

comment:先缓存数据再等待处理器读取



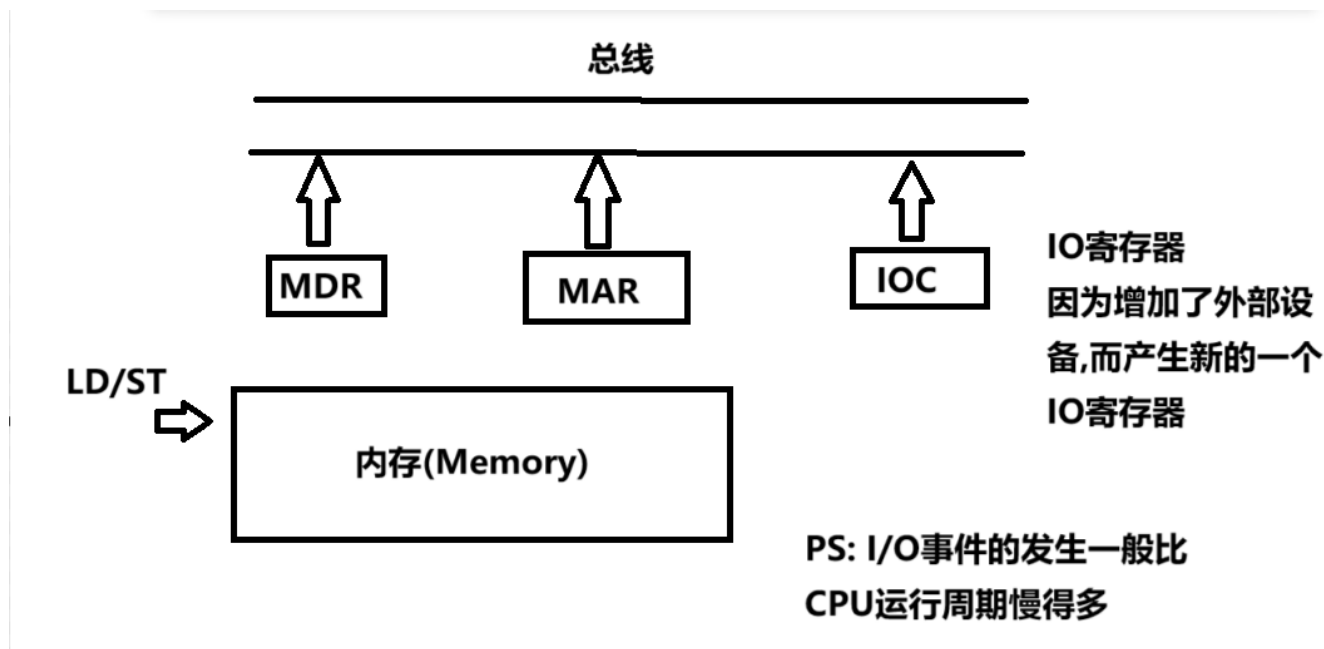
①. CPU 告诉我们设备做什么---Write (控制寄存器)

CPU check if tasks 've done --- Read (状态寄存器)

e.g. check the state of extern equipment:显示器是否空闲

②. CPU通过数据寄存器从设备 read/write 数据

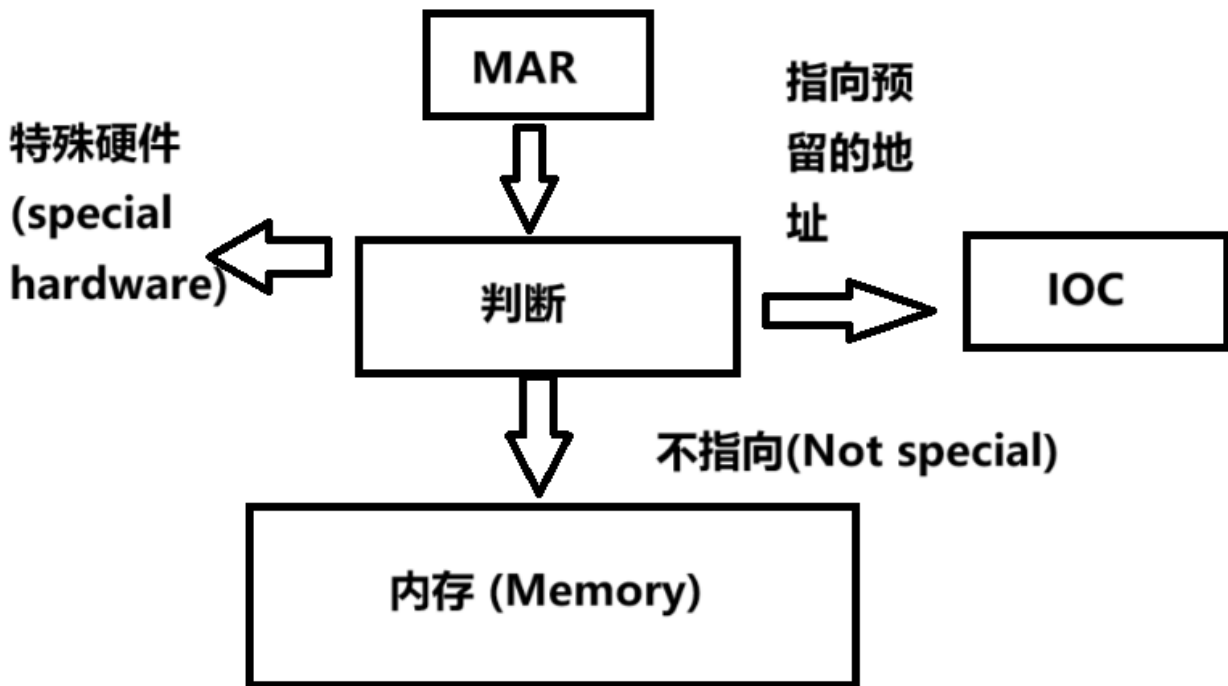
③. 执行实际操作(e.g. 输出像素带屏幕的字符流)



## 编址方式: 内存映射 VS 专用 I/O 指令

内存映射: e.g. ARM

专用I/O指令: e.g. X86 (IN / OUT) (专用IO指令是为访问外部设备而专门设计的指令)



## 内存映射:

专门预留出一块内存,用于IO,而不额外增添新的指令(内存不可有重复)

## LD/ST



内存映射:

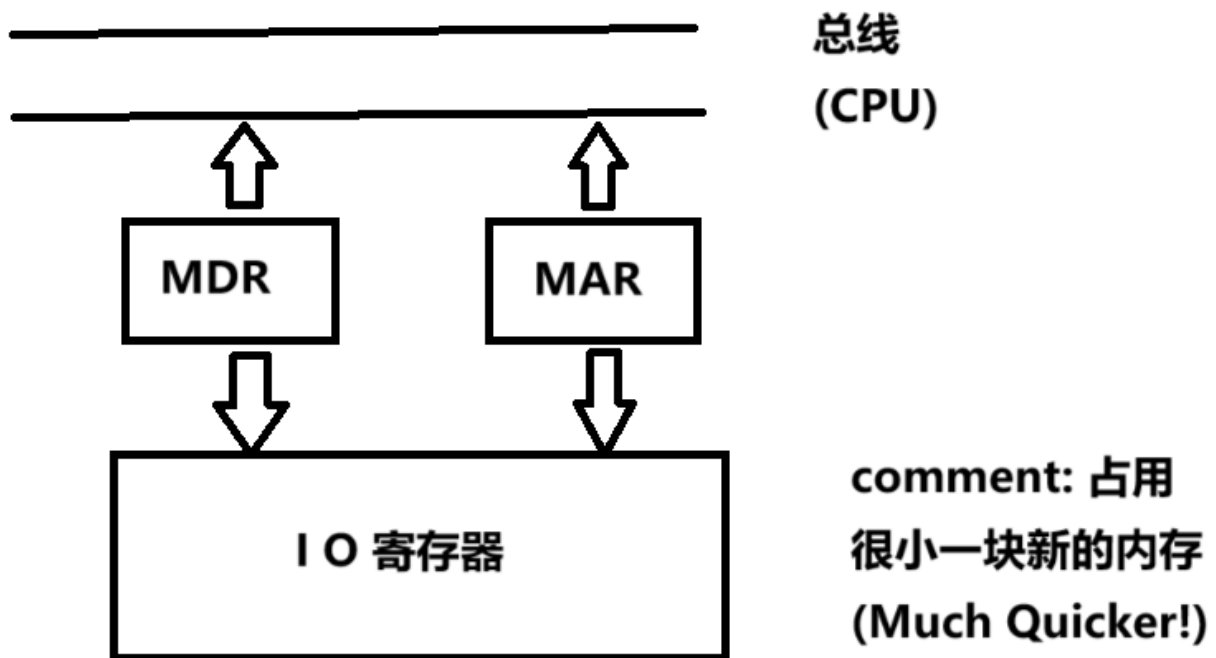
专门预留出一块内存,用于I/O,而不额外增添新的指令  
(内存不可有重复)

comment:过度依赖Status Register 实质上是实现一种"轮询"机制.  
( Efficiency Low! )

(访存性能下降)

## 专用 IN/OUT 指令:

仿照LD/ST设计新的专用IN/OUT指令.



## 同步与异步:

1. 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.
2. 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.

### Method:

设置状态寄存器(Status Register)或者标志位(signed bit),以满足/保证传输的正确性.

Status Register:

e.g.

0 Wrong Message --- 不读

1 Correct Message --- 读

**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 (轮询机制-->中断机制)**

A Scene:

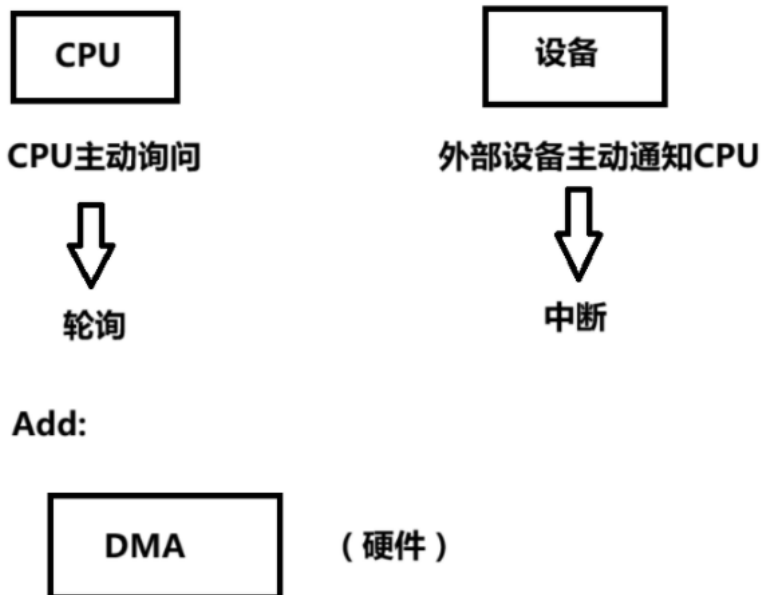
"客人到了没?客人到了没?客人到了没?" ---> "当客人到达时请通知我"

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.

Difference: 谁控制传输?

谁控制传输？

Who controls the transmission?



如何进一步提升中断的效率？

(DMA 机制: 直接内存访问)

(A new hardware)

A Scene:

请自己家的小孩去迎接客人到家,自己先做别的事情,小孩接到家了再直接招待客人.

Advantage:

甚至不用自己亲自下楼接客人.

an example for IO program

```

01  START    ST      R1,SaveR1    ; Save registers needed
02          ST      R2,SaveR2    ; by this routine
03          ST      R3,SaveR3
04  ;
05          LD      R2,Newline
06  L1       LDI     R3,DSR
07          BRzp    L1           ; Loop until monitor is ready
08          STI     R2,DDR        ; Move cursor to new clean line
09  ;
0A          LEA     R1,Prompt     ; Starting address of prompt string
0B  Loop     LDR     R0,R1,#0     ; Write the input prompt
0C          BRz     Input        ; End of prompt string
0D  L2       LDI     R3,DSR
0E          BRzp    L2           ; Loop until monitor is ready
0F          STI     R0,DDR        ; Write next prompt character
10          ADD     R1,R1,#1     ; Increment prompt pointer
11          BRnzp   Loop         ; Get next prompt character
12  ;
13  Input     LDI     R3,KBSR
14          BRzp    Input        ; Poll until a character is typed
15          LDI     R0,KBDR      ; Load input character into R0
16  L3       LDI     R3,DSR
17          BRzp    L3           ; Loop until monitor is ready
18          STI     R0,DDR        ; Echo input character
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>'

```