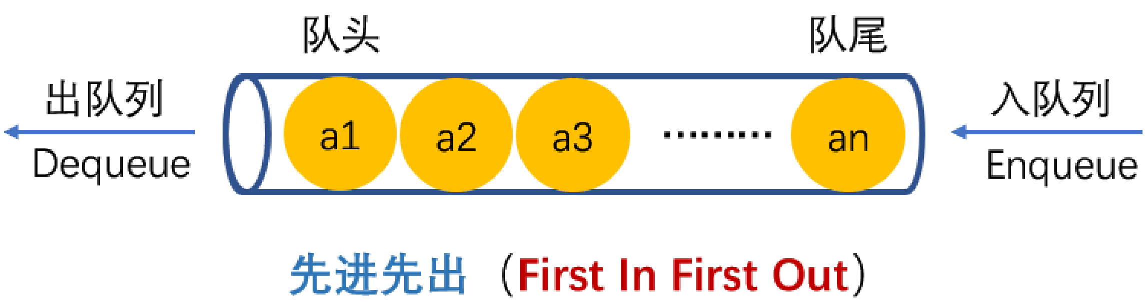
CH8

一些补充

- 比如带检测的$push$和pop，函数的参数、返回值只能是由$caller\ save$因为这些信息被主程序需要
- $STRCMP$ in $LC-3$

```
STRCMP ST R0,SaveR0
ST R1,SaveR1
ST R2,SaveR2
ST R3,SaveR3
;
AND R5,R5,#0 ; R5 <-- Match
;
NEXTCHAR LDR R2,R0,#0 ; R2 contains character from 1st string
LDR R3,R1,#0 ; R3 contains character from 2nd string
BRnp COMPARE ; String is not done, continue comparing
ADD R2,R2,#0
BRz DONE ; If both strings done, match found
COMPARE NOT R2,R2
ADD R2,R2,#1 ; R2 contains negative of character
ADD R2,R2,R3 ; Compare the 2 characters
BRnp FAIL ; Not equal, no match
ADD R0,R0,#1
ADD R1,R1,#1
BRnzp NEXTCHAR ; Move on to next pair of characters
;
FAIL ADD R5,R5,#1 ; R5 <-- No match
;
DONE LD R0,SaveR0
LD R1,SaveR1
LD R2,SaveR2
LD R3,SaveR3
RET
;
SaveR0 .BLKW 1
SaveR1 .BLKW 1
SaveR2 .BLKW 1
SaveR3 .BLKW 1
```

- The defining property of the abstract data type queue is **FIFO**



CH9

Privilege VS Priority

- **两个例子说明可能存在**$High\ Priority, Low\ Privilege$
- **Two Orthogonal Notions**

> We said privilege and priority are two orthogonal notions, meaning they have nothing to do with each other.
>
> - 书上的三个例子
> - the right to do sth vs the urgency to do sth

Processor status register (PSR)

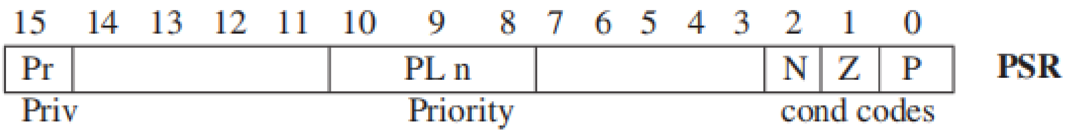


Figure 9.1 Processor status register (PSR).

- $Bit[15]$ specifies the privilege, where $PSR[15]=0$ means **supervisor privilege**, and $PSR[15]=1$ means **unprivileged**.
- $Bits[10:8]$ specify the priority level(PL) of the program. The highest priority level is 7 ($PL7$), the lowest is $PL0$.
- $Bits[2:0]$是$Condition\ Code$可能会被中断的程序破坏，因此我们需要保存当前的$Condition\ Code$
- 中断时我们需要用栈保存**PC and PSR**

P713中断的状态机图

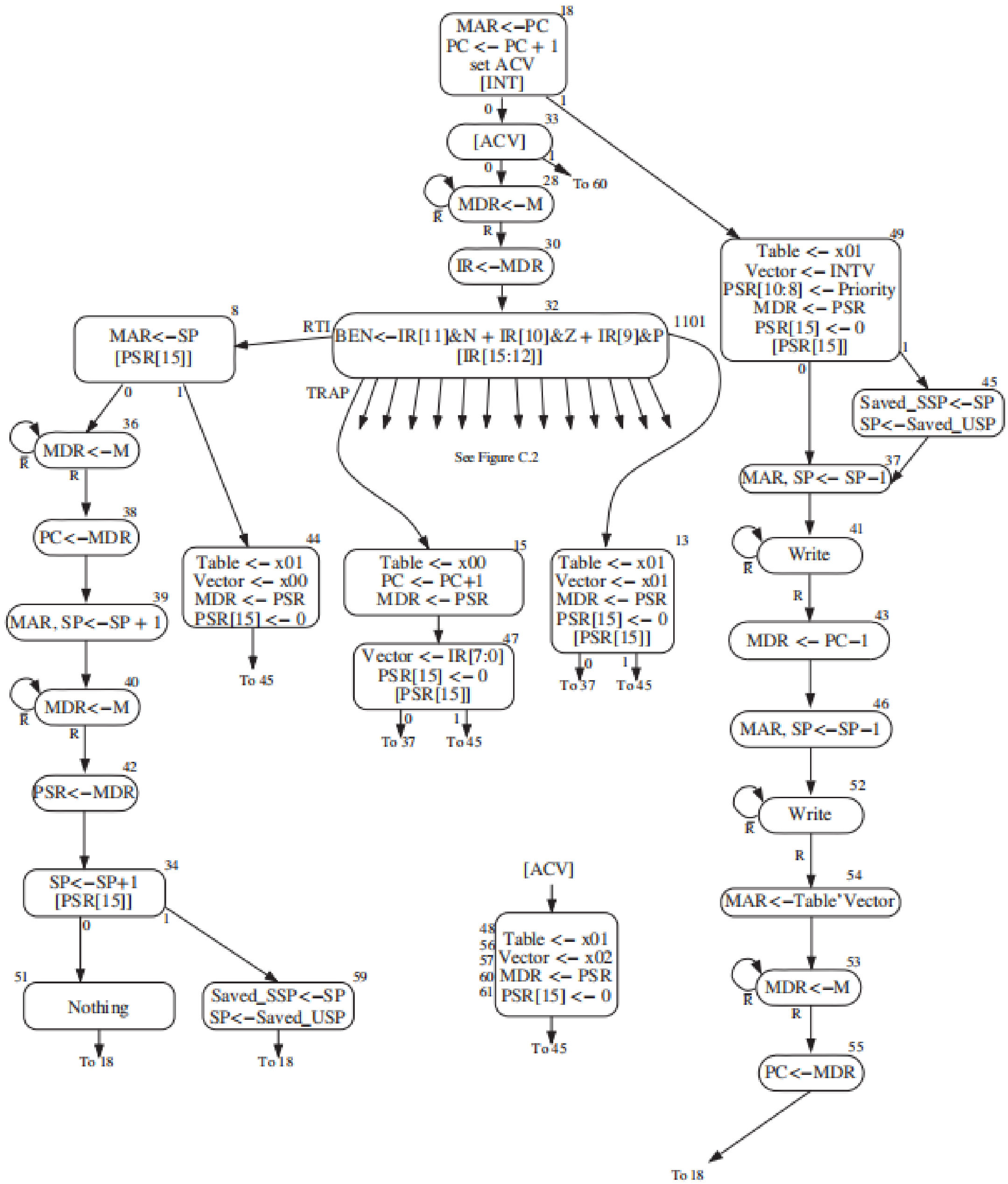


Figure C.7 LC-3 state machine showing interrupt control.

Region of Memory

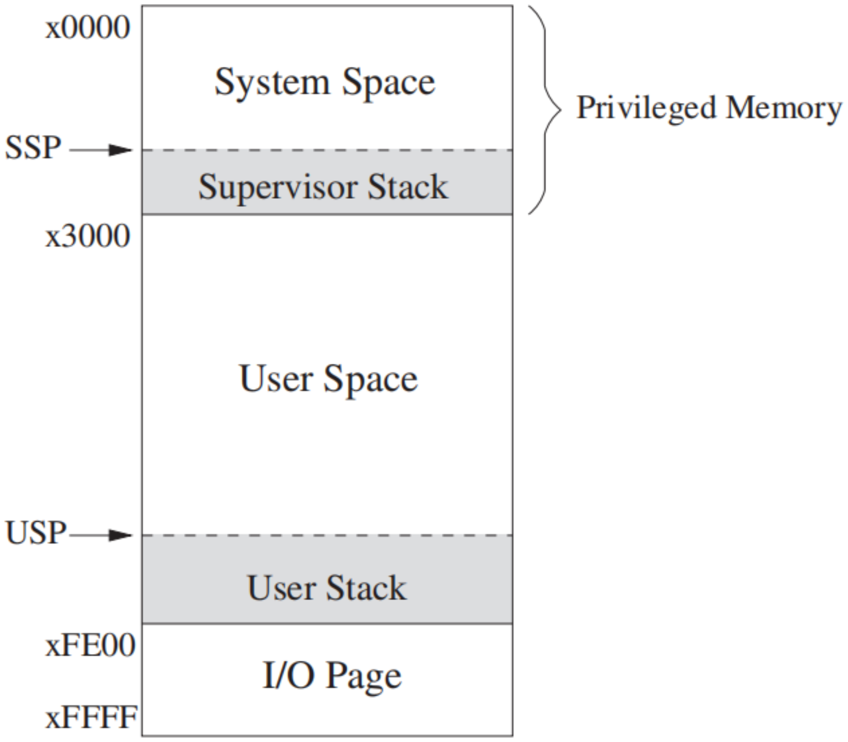
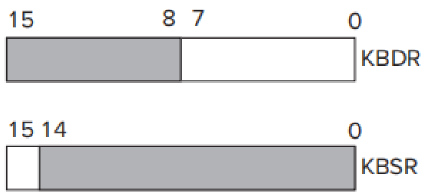


Figure 9.2 Regions of memory.

- 切换系统的权限时，我们需要用$Saved\ SSP$或者$Saved\ USP$（这两个东西是寄存器还是内存的某个固定位置）来保存当前的SSP或者USP
- IO page may actually not in memory physically!

Input

- **Basic Registers**

> 
>
> Figure 9.3 Keyboard device registers.
>
> | KBDR | KBSR |
> | --- | --- |
> | xFE02 | xFE00 |
>
> - $KBDR$的7~0位为键盘输入的$Ascii$码值
> - $KBSR$的15位为$Ready\ Bit$用来标识是否能够读取键盘输入的数据
> - $KBSR$会被$reset$当我们访问$KBDR$时

- 我们可以通过$BRzp$来不断地进行尝试读取的过程：

```
START LDI R1, A ; Test for
BRzp START ; character input
LDI R0, B
BRnzp NEXT_TASK ; Go to the next task
A .FILL xFE00 ; Address of KBSR
B .FILL xFE02 ; Address of KBDR
```

- **Implementation of Memory-Mapped Input**

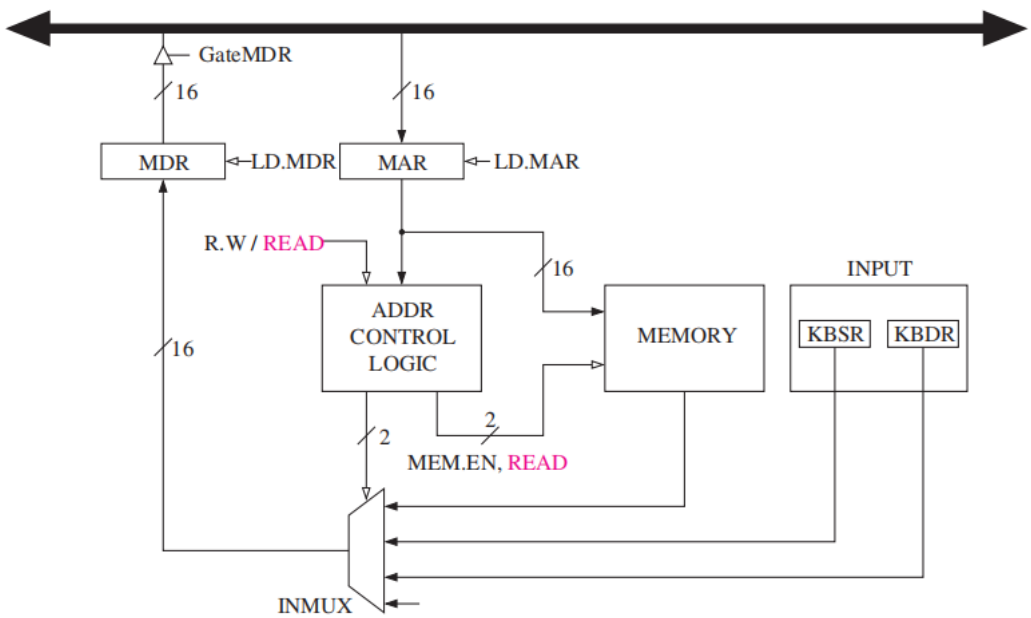


Figure 9.4 Memory-mapped input.

Output

- **Basic Registers**

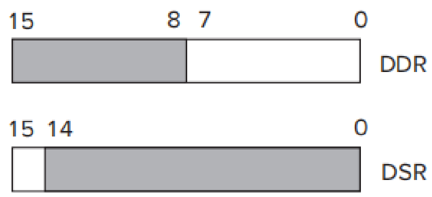


Figure 9.5 Monitor device registers.

DDR	DSR
xFE06	xFE04

  - DDR的7~0位为希望输出的$Ascii$码值
  - DSR的15位为$Ready\ Bit$用来标识显示器是否处理完了之前的数据
  - DSR会被$reset$当我们访问DDR时

- 同样地，我们能够实现不断地尝试输出的过程

```
START LDI R1, A ; Test to see if
BRzp START ; output register is ready
STI R0, B
BRnzp NEXT_TASK
A .FILL xFE04 ; Address of DSR
B .FILL xFE06 ; Address of DDR
```

- **Implementation of Memory-Mapped Output**

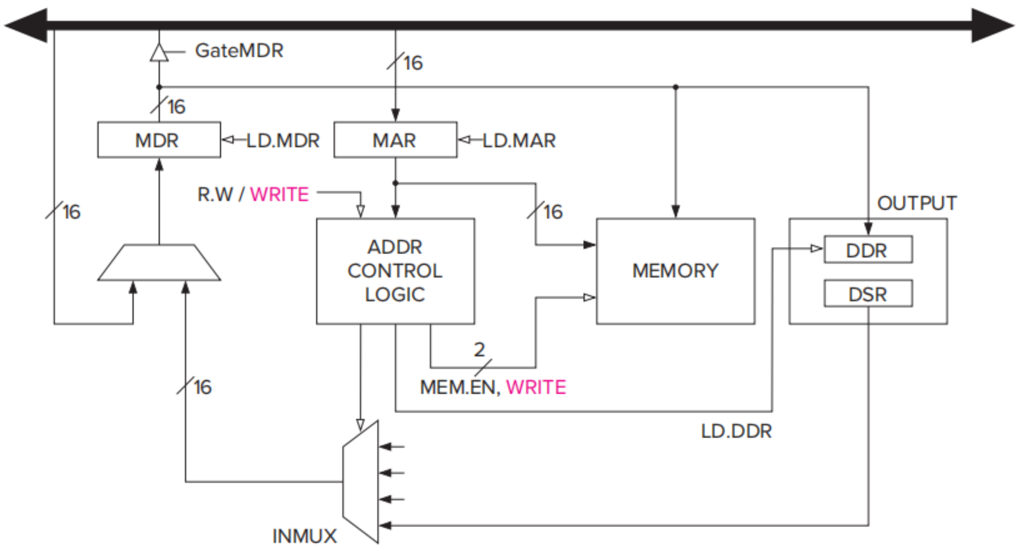


Figure 9.6 Memory-mapped output.

Some Basic Characteristics of IO

- memory-mapped IO vs special IO instructions
- asynchronous 异步 vs synchronous 同步
- interrupt-driven vs polling 轮询

(recommend to read 9.2.1)

组合I/O实现回显

```
START LDI R1, KBSR ; Test for character input
BRzp START
LDI R0, KBDR
ECHO LDI R1, DSR ; Test output register ready
BRzp ECHO
STI R0, DDR
BRnzp NEXT_TASK
KBSR .FILL xFE00 ; Address of KBSR
KBDR .FILL xFE02 ; Address of KBDR
DSR .FILL xFE04 ; Address of DSR
DDR .FILL xFE06 ; Address of DDR
```

相关的Datapath

- 我们不需要Input from DDR也不需要Output to KBDR，所以相关的线都不需要连接
- $KBSR$和DSR的重置是由物理电路完成的(P320页9.2.2.2节)，但我们仍需要设置它的第14位。

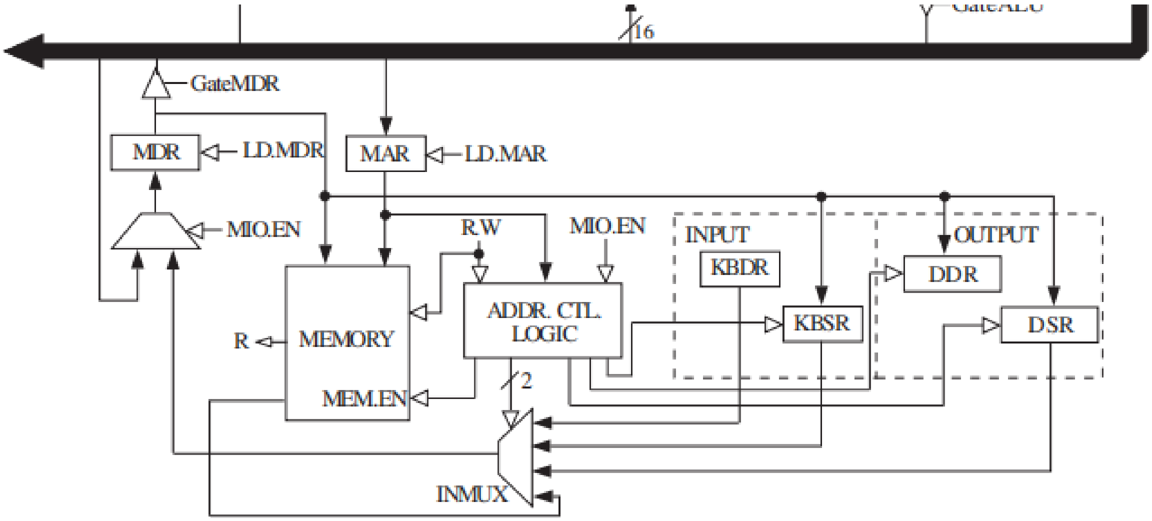


Figure C.3 The LC-3 data path.

时钟与如何重启(P136)

- 真正的时钟是由振荡器产生的
- 计算机感受到的时钟是**真正的时钟 AND MCR[15]（就是课本上的Run Latch）**
- **重启不能通过指令**，而是通过物理的按钮等进行重启

HW5

- 5.37对着Datapath再讲一遍，STI要有ALU，没有NZP，LEA没有NZP
- 7.32计算LABEL注意BLKW和STRINGZ，问答题时关键在于时间、阶段