ICS 期末讨论课

叶升宇

2023 年 02 月 19 日

chap7

- 操作数：必须包含标识数字的表示基数的符号。用 # 表示十进制，x 表示十六进制，b 表示二进制；
- 标签：由 1 到 20 个字母或数字字符组成 (每个字符是大写或小写字母，或十进制数字)，以字母开头。注意不能是保留字 (ADD, NOT, x1000, R4)；
- 伪操作：$.BLKW$ 分配 n 个空间（n 为操作数)；$.STRINGZ$ 分配 $n+1$ 个空间（n 为字符串长度，最后一位是 x0000)；
- 汇编时两遍扫描：第一遍找到所有标签并计算出对应的地址，创建符号表，第二遍翻译生成对应的机器码；
- 注意检查地址是否越界，如 LD 指令 $PCoffset9 \in [-256, 255]$，因此 LD 后面的标签距离当前 LD 指令的地址范围应该为 $[-255, 256]$（注意 PC 自增)。

chap8

- JSR、JSRR 指令，用来调用子程序，注意与 JMP 的区别 (R7 记录自增的 PC，即下一条指令的地址)；
- 调用子程序注意寄存器的恢复与保存；
- 递归 Bad Example（计算阶乘）中的问题：
  1. 覆盖 R7，无法返回主程序；
  2. 覆盖 R1 中的 n；
  3. 存在 Save1 中的 R1 也被覆盖；
- 栈：$R6$ 栈顶指针（指向栈顶），注意 $Underflow$（栈空 Pop）和 $Overflow$（栈满 $Push$）以及对应的检测；
- （循环）队列：$R3$ 队首（$FRONT$）指针（下一个是队首），$R4$ 队尾（$REAR$）指针（指向队尾）。队空 $FRONT = REAR$，队满 $FRONT + 1 = REAR$，同样注意 $Underflow$ 和 $Overflow$ 及检测。

内存地址空间

内存地址空间 16 位，对应 2^{16} 个位置，每个位置包含一个字 (16 位)。地址的编号从 0 (x0000) 到 65535(xFFFF)。地址用于标识内存位置和内存映射 I/O 设备寄存器。
内存的某些区域是预留给特殊用途的:

- 地址 x0000 到 x2FFF 包含特权内存（被称为系统空间），只有在 Supervisor 模式下 ($PSR[15] = 0$) 执行时才可以访问;
- 地址 x3000 到 xFDFF 包含用户模式 ($PSR[15] = 1$) 可用的内存;
- 地址 xFE00 到 xFFFF 包含输入和输出设备寄存器和特殊的内部处理器寄存器，这些寄存器也只有在 Supervisor 模式下执行时才可以访问 ($PSR[15] = 0$)。为了控制对这些设备寄存器的访问，它们的地址也被认为是特权内存的一部分;
- x0000-x00FF Trap Vector Table; x0100-x01FF Interrupt Vector Table; 其中 x0100-x017F 是异常处理例程，x0180-x01FF 是中断服务程序。

LC3 Overview

内存映射 I / O

输入和输出由 $load/store(LD/ST,\ LDI/STI,\ LDR/STR)$ 指令处理，使用从 xFE00 到 xFFFF 的内存地址来指定每个设备寄存器。具体参见 $P654\ Table\ A.1$

优先级

$LC-3$ 支持 8 级优先级。优先级 7 $(PL7)$ 最高，$PL0$ 最低。当前执行进程的优先级以 $PSR[10:8]$ 来指定。

Processor status register (PSR)

一个 16 位寄存器，包含当前正在执行的进程的状态信息。到目前为止，已经定义了七个 PSR 位。$PSR[15]$ 指定执行进程的特权模式；$PSR[10:8]$ 指定当前正在执行的进程的优先级；$PSR[2:0]$ 包含条件码；$PSR[2]$ 为 N，$PSR[1]$ 为 Z，$PSR[0]$ 为 P。(PSR 之所以要条件码是为了 $TRAP$ 或中断跳转之后 RTI 进行原来程序状态的恢复)

LC3 Overview

Supervisor mode

$LC-3$ 规定了两种操作模式：管理模式 (特权模式) 和用户模式 (非特权模式)。中断服务程序和 $TRAP$ 服务程序 (例如，系统调用) 在 $Supervisor$ 模式下执行。由 $PSR[15]$ 指定特权模式。$PSR[15]=0$ 表示 $Supervisor$ 模式；$PSR[15]=1$ 表示用户模式。

Privilege mode exception

RTI 指令以 $Supervisor$ 模式执行。如果处理器试图在用户模式下执行 RTI 指令，则会发生特权模式异常。

Access Control Violation (ACV) exception

当一个进程在 $User$ 模式下运行时，如果试图访问特权内存中的一个位置 (系统空间中的一个位置或一个地址在 xFE00 到 xFFFF 的设备寄存器)，就会发生 ACV 异常。

LC3 Overview

Illegal opcode exception

opcode $= 1101$ 没有被指定。如果一条指令 opcode 为 1101，则会发生非法操作码异常。

Supervisor stack

系统空间中的一个内存区域，可以通过管理器堆栈指针 (SSP) 访问。当 $PSR[15] = 0$ 时，堆栈指针 ($R6$) 是 SSP。当处理器运行在 $User$ 模式 ($PSR[15] = 1$) 时，SSP 被存储在 $Saved\_SSP$ 中。

User stack

用户空间中的内存区域，可通过用户堆栈指针 (USP) 访问。当 $PSR[15] = 1$ 时，堆栈指针 ($R6$) 是 USP。当运行在 $Supervisor$ 模式 ($PSR[15] = 0$) 时，USP 被存储在 $Saved\_USP$ 中。

Input

两个特殊寄存器：

- keyboard data register (KBDR)
- keyboard status register (KBSR)

$KBSR[15]$ 控制键盘和处理器的同步：

- 当敲击键盘上的一个键时，该键的 $ASCII$ 码被载入 $KBDR[7:0]$ 中，与键盘相关的电路自动将 $KBSR[15]$ 设置为 1；
- 当 $LC-3$ 读取 $KBDR$ 时，与键盘相关的电路自动清除 $KBSR[15]$，允许敲击另一个键。
- 如果 $KBSR[15]=1$，则最后一个击键对应的 $ASCII$ 码还没有被读取，因此键盘被禁用。

Output

两个特殊寄存器：

- Display Data Register (DDR)
- Display Status Register (DSR)

$DSR[15]$ 控制处理器和显示器的同步：

- 当 $LC-3$ 向 $DDR[7:0]$ 传输一个 $ASCII$ 码进行输出时，开始处理 $DDR[7:0]$ 时，显示器的电子设备自动清除 $DSR[15]$；
- 当显示器在屏幕上处理完字符时，它自动设置 $DSR[15]$。这是给处理器的一个信号，表明处理器可以将另一个 $ASCII$ 码传输到 DDR 进行输出；
- 只要 $DSR[15]$ 是清零的，说明显示器仍在处理前一个字符，因此就处理器的额外输出而言，此时显示器器是禁用的。

TRAP

TRAP 指令周期的 EXECUTE 阶段做三件事:

1. PSR, PC 入栈（此时 PC 已经自增，因此之后出栈的 PC 就是 TRAP 下一条指令的位置）。如果是在用户模式下，R6 指向用户栈。在 PSR 和 PC 入栈之前，需要执行 $Saved\_USP \leftarrow R6, R6 \leftarrow Saved\_SSP$。
2. $PSR[15]$ 被设置为 0，因为服务例程需要管理员权限才能执行。$PSR[10:8]$ 保持不变，因为 $TRAP$ 例程的优先级与请求它的程序的优先级相同。
3. 8 位的 trap vector 无符号扩展到 16 位，形成一个地址，对应于 TARP VECTOR TABLE 中的某个位置。该位置中的值（程序的起始位置）被加载到 PC 中。

TRAP

对应到状态机和数据通路里，完成如下操作：（参见 $P713$）

- state15: $TABLE \leftarrow x00$, $PC \leftarrow PC + 1$, $MDR \leftarrow PSR$.
  TARP VECTOR TABLE 在内存中是 x0000-x00FF，因此 $TABLE \leftarrow x00$；$PC + 1$ 是因为后面要保证后面出栈时的 PC 指向 $TRAP$ 下一条指令；注意此时的 PC 已经自增，这里又进行了一次 $+1$ 看起来好像有问题，其实是因为后面 state43，PC 会 -1；
- state47: $Vector \leftarrow IR[7:0]$, $PSR[15] \leftarrow 0$.
  Vector 存的是 trap vector，这里根据原来的 $PSR[15]$ 确认下一状态，若为 0，直接进入 state37，若为 1，进入 state45；
- state45: $Saved\_USP \leftarrow R6$, $R6 \leftarrow Saved\_SSP$.
  state45 表明原来是 User mode ，要向 Supervisor mode 转变，因此要先存用户栈的栈顶指针，然后加载系统栈的栈顶指针。

注

书上 *state45* 有误，这里已经根据 *patt 21 final exam* 修正.

TRAP

- state37: $MAR, SP \leftarrow SP - 1$; state41: Write
  这两个状态完成 PSR 入栈（根据 state15，此时 MDR 存的是 PSR ）
- state43: $MDR \leftarrow PC - 1$; state46: MAR, $SP \leftarrow SP - 1$; state52: Write
  这三个状态完成 PC 入栈。特别地
  对于 INT，PC 已经在 state18 自增，但是由于中断，当前的指令也还没执行，因此应该 PC-1；
  对于 TRAP，前面 state15 已经 $+1$，所以这里 -1 也是对的，也就是前面 state15 的 PC $+$ 1 实际上是为了和这里 PC - 1 统一)
- state54、state53、state55 即加载 PC 为相应程序的起始地址，然后执行，不赘述。

RTI

- 首先检查 $PSR[15]$ 若为 0 正常继续；若为 1，说明在用户模式下 RTI，则发生异常，进行异常处理。
- 接着，PC, PSR 出栈；
- 然后再次检查 $PSR[15]$，若为 0，说明原来就是 Supervisor mode，不用切换，否则因此执行 $Saved\_SSP \leftarrow R6$，$R6 \leftarrow Saved\_USP$，完成 Supervisor mode 转变为 User mode；
- 具体到状态机里面其实和 TRAP 是镜像的，就不多赘述，出栈的 PC 就是原先程序要接着执行的正确的地址。

Interrupt

- 大多数 I/O 设备中，有 Interrupt Enable(IE)，是设备状态寄存器的一部分。IE 位为 [14] 位；
- 如果中断使能位 (位 [14]) 被清除，则与是否设置就绪位无关，I/O 设备将不能中断处理器，因为它 (I/O 设备) 没有被授予中断处理器的权利。在这种情况下，程序将不得不轮询 I/O 设备，以确定它是否准备好了；
- 如果设置了 [14] 位，则启用中断驱动的 I/O。在这种情况下，只要有人键入一个键 (或者只要显示器完成对最后一个字符的处理)，就会设置位 [15]，然后组合生成中断信号，进行处理；
- 为了停止处理器继续执行当前运行的程序并处理中断请求，必须 INT 信号。任何具有 [14] 和 [15] 位的设备都会发出中断请求信号。中断请求信号被输入到优先级编码器，优先级编码器是一种组合逻辑结构，它从所有的请求中选择优先级最高的请求。如果该请求的 PL 高于当前执行程序的 PL，则设置 INT 信号。

Interrupt

- 发生中断时，令 PSR, PC 入栈，根据中断程序的优先级设置 $PSR[10:8]$，并且要 $PSR[15] = 0$（即工作在 Supervisor mode）；
- 同 TRAP 一样，如果原来是在用户模式下，R6 指向用户栈。在 PSR 和 PC 入栈之前，需要执行，$Saved\_USP \leftarrow R6, R6 \leftarrow Saved\_SSP$;
- 中断和 TRAP 类似，Trap Vector Table 由 x0000 到 x00FF 的内存位置组成，每个位置都包含一个 Trap 服务例程的起始地址；而中断向量表由 x0100 到 x01FF 的内存位置组成，每个位置都包含一个中断服务程序的起始地址。处理器将 $8 - bit$ 中断向量 interrupt vector(INTV) 展开，把形成的地址对应的中断向量表中位置的内容装入 PC。

Interrupt

从状态机来说：

- state49: $TABLE \leftarrow x01$, $Vector \leftarrow INTV$, $PC \leftarrow PC+1$, $MDR \leftarrow PSR$, $PSR[10:8] = Priority$, $PSR[15] -> 0$
  中断向量表在内存中是 x0000-x01FF，因此 $Vector \leftarrow x01$；注意没有 $PC+1$，因为中断时该条指令还没有执行，返回时应该继续执行，所以这里是自增后的 PC 入栈，后续 state43 又进行了一次 -1，则为当前 PC；
- 后续就和 TRAP 一样，根据原来的 $PSR[15]$ 确认下一状态，若为 0，进入 state37，若为 1，进入 state45，之后的都一样，不赘述。
- 中断返回，用的也是 RTI。

数据通路

下面结合书本附录 C 详细讲解状态机与数据通路。

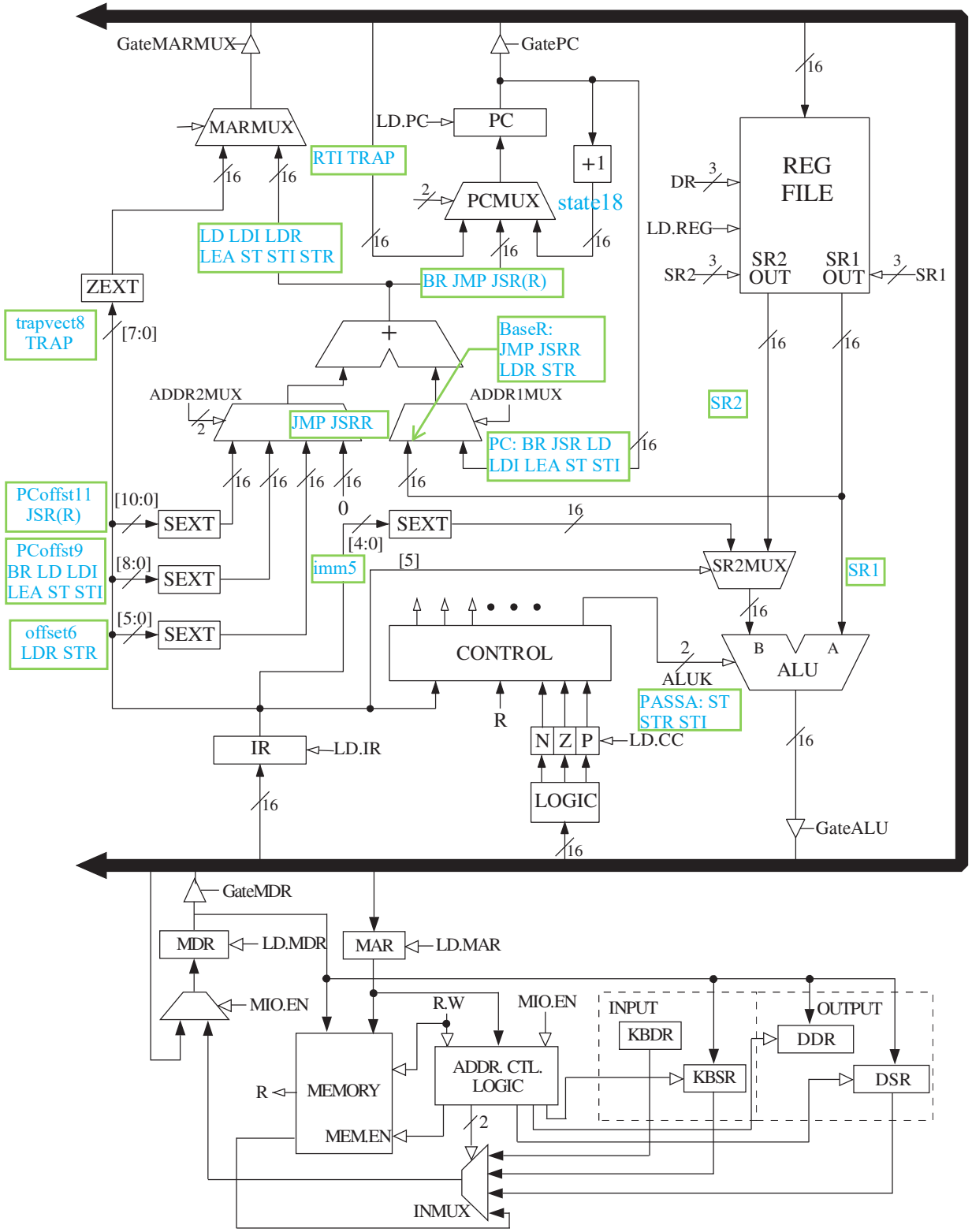

Figure C.3 The LC-3 data path.

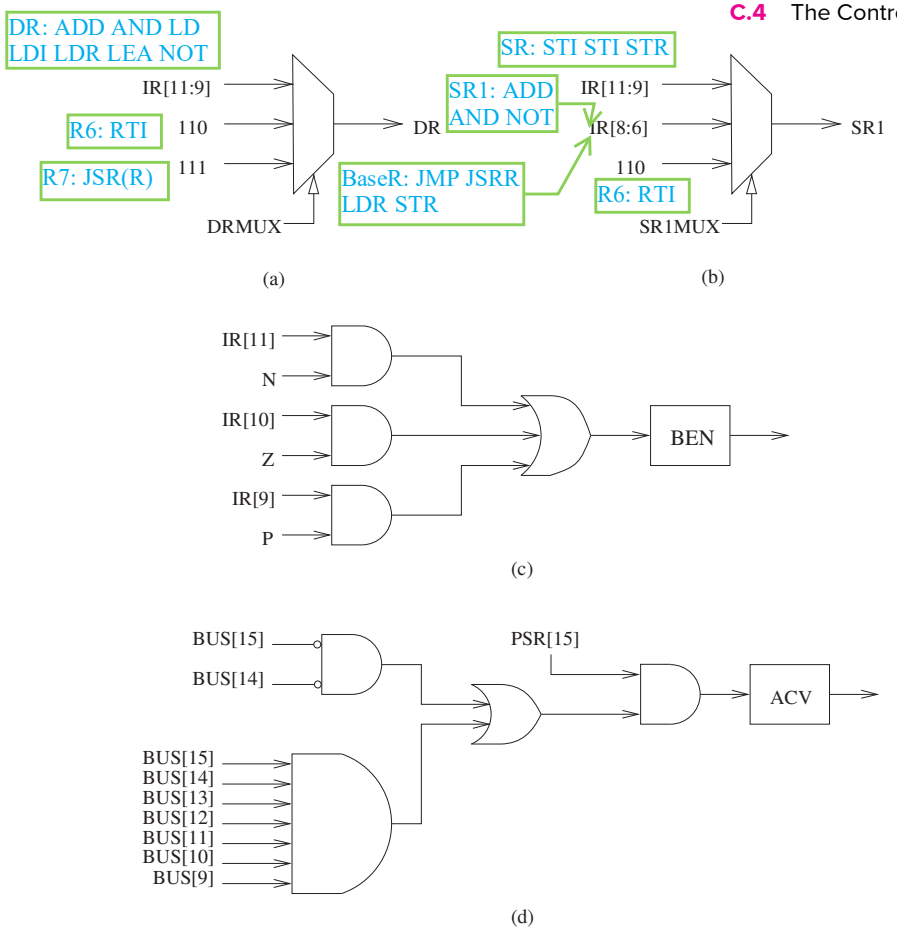

Figure C.6 Additional logic required to provide control signals.

Several signals necessary to control the data path and the microsequencer are not among those listed in Tables C.1 and C.2. They are DR, SR1, BEN, INT, ACV, and R. Figure C.6 shows the additional logic needed to generate DR, SR1, BEN, and ACV.

The INT signal is supplied by some event external to the normal instruction processing, indicating that normal instruction processing should be interrupted and this external event dealt with. The interrupt mechanism was described in Chapter 9. The corresponding flow of control within the microarchitecture is described in Section C.7.

The remaining signal, R, is a signal generated by the memory in order to allow the LC-3 to operate correctly with a memory that takes multiple clock cycles to read or store a value.

Suppose it takes memory five cycles to read a value. That is, once MAR contains the address to be read and the microinstruction asserts READ, it will take five cycles before the contents of the specified location in memory is available to be loaded into MDR. (Note that the microinstruction asserts READ by means of two control signals: MIO.EN/YES and R.W/RD; see Figure C.3.)

附录 A

- A.1 LC-3 概述：一些基本概念的介绍，PPT 4-7 已讲述重点；
- A.2 指令集：用于速查各条指令的行为；
- A.3 中断和异常处理，讲述处理细节，建议看一遍，异常处理可以不看；

附录 C

- C.1 LC-3 微结构概述：描述主要组件以及信号，可以不看；
- C.2 状态机，重点必看；
- C.3 数据通路，重点必看；
- C.4 控制结构：看一下 DRMUX 和 SR1MUX 即可，其它可不看；
- C.5 TRAP: TRAP 细节，建议看一遍；
- C.6 内存映射 I/O：简单看一下即可；
- C.7 中断和异常处理以及 RTI，异常可以不看，其它建议看一遍；
- C.8 可以说那张表是 LC-3 的核心，假如你能填了那张表，可以说基本完全理解了 LC-3。

完结撒花

祝大家期末取得好成绩！