LC3-tools

注意点：路径不能有中文字符，忽略console的warning，载入程序前randomize机器，寄存器使用前用AND初始化

HW 2

2.40 Write the decimal equivalents for these IEEE floating point numbers.

a. 0 10000000 00000000000000000000000
b. 1 10000011 00010000000000000000000
c. 0 11111111 00000000000000000000000
d. 1 10000000 10010000000000000000000

> c. 是正无穷不是1×10^{128}

3.36 A comparator circuit has two 1-bit inputs A and B and three 1-bit outputs G (greater), E (Equal), and L (less than). Refer to Figures 3.43 and 3.44 for this problem.

G is 1 if $A > B$	E is 1 if $A = B$	L is 1 if $A < B$
0 otherwise	0 otherwise	0 otherwise

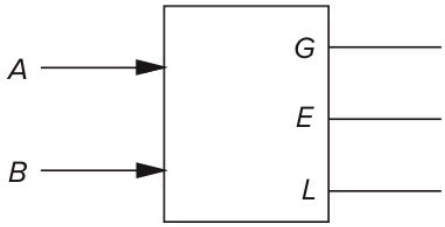
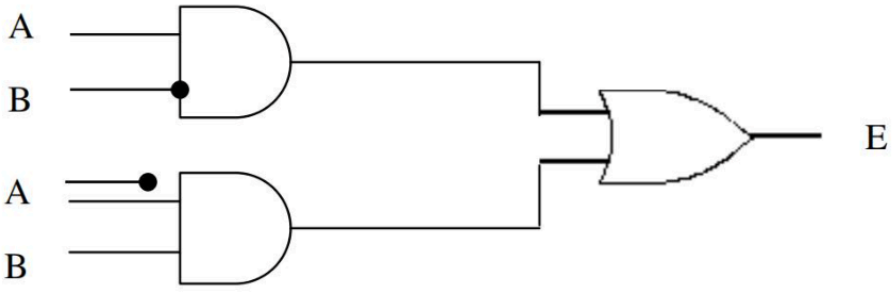


Figure 3.43 Diagram for Exercise 3.36.

> 
>

题目序号！

Ch5

Opcode.

The opcode is specified in bits [15:12] of the instruction.

Addressing Modes.

(1) **Literal (or immediate)**
operand as a part of the instruction.

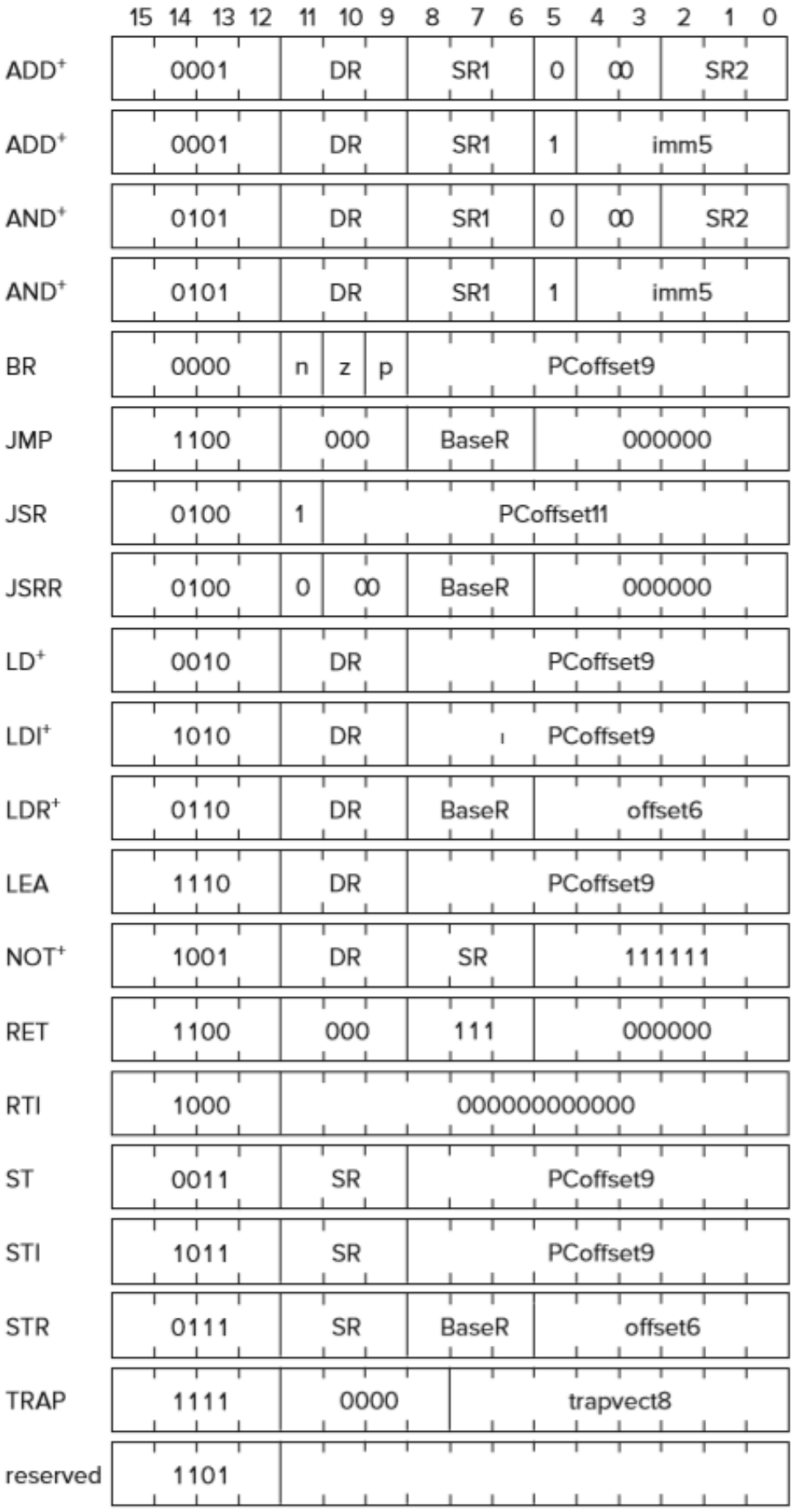
(2) **Register**

(3) Three memory addresing modes: (3.1) **PC-relative**; (3.2) **indirect**; (3.3) **Base+offset**.

Condition Codes.

Each time one of the 8 General Purpose Registers is written by an **3 operate** or a **3 load** instruction **(注意没有LEA)**, the 3 single-bit registers **N(negative) and Z(zero) and P(positive)** are individually set to 0 or 1, corresponding to whether the result written into the register is negative, zero or positive.

Instructions.



+ indicates instructions that modify condition codes

- *Operate Instructions*

- **ADD**. [15:12] = 0001. [5] indicates the mode taken.
  - Mode 0: [11:9] - destination register. [8:6] - source register 1. [4:3] = 00. [2:0] - source register 2.

  `ADD DR, SR1, SR2` *DR = SR1 + SR2*

  - Mode 1: [11:9] - destination register. [8:6] - source register 1. [4:0] - Immediates (-16~15).

  `ADD DR, SR, #imm5` *DR = SR + #imm5*
- **AND**. [15:12] = 0101. Others are same as ADD.

  `AND DR, SR1, SR2` *DR = AND(SR1, SR2)*

  `AND DR, SR, #imm5` *DR = AND(SR, #imm5)*

  > When operating, immediates will be sign-extended to 16 bits before performing the ADD or AND.
- **NOT.** [15:12] = 1001. [11:9] - destination register. [8:6] - source register. [5:0] = 111111

  `NOT DR, SR` *DR = NOT(SR)*
- **LEA, Load Effective Address 有效地址装载.**

  [15:12] = 1110. [11:9] - destination register. [8:0] - PC offset 9 (-256~255).

  `LEA DR, #imm9` *DR = PC + #imm9*

  > Note that the PC has been incremented before addition.

- *Data Movement Instructions*
  - *PC-Relative Mode*
    - **LD, Load.** [15:12] = 0010. [11:9] - destination register. [8:0] - PC offset 9.

      `LD DR, #imm9` *DR = [PC + #imm9]*
    - **ST, Store.** [15:12] = 0011. [11:9] - source register. [8:0] - PC offset 9.

      `ST SR, #imm9` *[PC + #imm9] = SR*

      > Note that the PC has been incremented before addition.
  - *Indirect Mode*
    - **LDI, Load Indirect.** [15:12] = 1010. [11:9] - destination register. [8:0] - PC offset 9.

      `LDI DR, #imm9` *DR = [[PC + #imm9]]*
    - **STI, Store Indirect.** [15:12] = 1011. [11:9] - source register. [8:0] - PC offset 9.

      `STI SR, #imm9` *[[PC + #imm9]] = SR*

      > Note that the PC has been incremented before addition.

The range of accessible address of LD and ST are limited by bits [8:0].
However the address of the operand of LDI and STI can be anywhere in the computer's memory.

    - *Base+offset Mode*
        - **LDR, Load Register.** [15:12] = 0110. [11:9] - DR. [8:6] - Base Register. [5:0] - offset 6.

          `LDR DR, BaseR, #imm6` *DR = [BaseR + #imm6]*
        - **STR, Store Register.** [15:12] = 0111. [11:9] - SR. [8:6] - Base Register. [5:0] - offset 6.

          `STR SR, BaseR, #imm6` *[BaseR + #imm6] = SR*

- *Control Instructions*
    - **BR, Branch.** [15:12] = 0000, [11:9] - n z p, [8:0] - PC offset 9.

      `BRn #imm9` or `BRz #imm9` or `BRp #imm9` or

      `BRnz #imm9` or `BRnp #imm9` or `BRzp #imm9` or `BRnzp #imm9`

      PC += #imm9 if ((n AND N) OR (z AND Z) OR (p AND P)) is true.

      > Note that the PC has been incremented before addition.
      >
      > BRnzp ([11:9] = 111) is an unconditional branch instruction.
      >
      > The range of accessible address of is limited by bits [8:0].
    - **JMP, Jump.** [15:12] = 1100, [11:9] = 000, [8:6] - BaseR, [5:0] = 000000

      `JMP BaseR` *PC=[BaseR]*
    - **TRAP.** [15:12] = 1111, [11:8] = 0000, [7:0] - trap vector 8

      `TRAP #imm8` *Call Service #imm8*

File

a sequence of ascii code, end with 0x04(EOT)
字母和数字的ascii码是连续的
举例：大小写转换(65:A; 97:a)

Tip

- *Question:* If a HALT instruction can clear the RUN latch, thereby stopping the instruction cycle, what instruction is needed to set the RUN latch, thereby reinitiating the instruction cycle? *Hint:* This is a trick question!
- c语言的if，for，while怎么用汇编写
- (重点) 解释Figure5.18 FigureC.2