5.5
```
BEGIN: MOV  AH,1     ;键盘输入字符
       INT  21H
       SUB  AL,'0'
       JB   STOP     ;<'0'结束
       CMP  AL,9     ;>9结束
       JA   STOP
       CBW
       MOV  CX,AX    ;响铃次数
       JCXZ STOP
BELL:  MOV  DL,07H
       MOV  AH,2     ;输出响铃字符
       INT  21H
       CALL DELAY 1000MS ;响铃设置延时
       LOOP BELL     ;重复N次
STOP:  RET
```

5.6
```
DSEG  SEGMENT
      COUNT  EQU 20.
      ARRAY  DW  20 DUP(?)
      COUNTP DB  0    ;正数个数
      ARRAYP DW  20 DUP(?) ;略
      COUNTN DB  0    ;负数个数
      ARRAYN DW  20 DUP(?)
```

```
POS     DB    'Positive number:', '$'
NEG     DB    'Negative number:', '$'
CRLF    DB    0DH, 0AH, '$'
DSEG    ENDS
;
CSEG    SEGMENT
    MAIN    PROC FAR
            ASSUME CS:CSEG, DS:DSEG
    START:  PUSH DS
            SUB  AX, AX   ; AX置0
            PUSH AX
            MOV  AX, DSEG
            MOV  DS, AX   ; 设置DS
    BEGIN:  MOV  CX, COUNT  ; 循环次数
            LEA  BX, ARRAY
            LEA  SI, ARRAYP
            LEA  DI, ARRAYN
    COMP:   MOV  AX, [BX]
            CMP  AX, 0
            JS   NEGA
            MOV  [SI], AX   ; 正数
            INC  COUNTP
            ADD  SI, 2
            JMP  SHORT NEXT
    NEGA:   MOV  [DI], AX   ; 负数
            INC  COUNTN
            ADD  DI, 2
```

```
NEXT:   ADD  BX,2
        LOOP BEGIN COMP
        LEA  DX, POS
        MOV  AL, COUNTP
        CALL DISPLAY      ;显示正数个数
        LEA  DX, NEG
        MOV  AL, COUNTN
        CALL DISPLAY      ;显示负数个数
        RET
MAIN    ENDP
;
DISPLAY PROC NEAR
        MOV  AH,9
        INT  21H
        AAM               ;AL中二进制数转化为BCD码
        ADD  AH,'0'       ;转换为ASCII码
        MOV  DL,AH
        MOV  AH,2
        INT  21H
        ADD  AL,'0'
        MOV  DL,AL
        MOV  AH,2
        INT  21H
        LEA  DX,CRLF      ;输出回车换行
        MOV  AH,9
        INT  21H
        RET
DISPLAY ENDP
CSEG    ENDS
        END START
```

5.11

```
DSEG SEGMENT
    BUFF DB 50 DUP(' ')
    COUNT DW 0
DSEG ENDS
CSEG SEGMENT
    ASSUME CS:CSEG, DS:DSEG
BEGIN: LEA BX, BUFF
    MOV COUNT, 0
INPUT: MOV AH, 01
    INT 21H
    MOV [BX], AL
    INC BX
    CMP AL, '$'
    JNZ INPUT      ;不是$继续输入
    LEA BX, BUFF
NEXT: MOV CL, [BX]
    INC BX
    CMP CL, '$'    ;结束符,表示没有字符了
    JZ DISP
    CMP CL, 30H    ;小于0则略过
    JB NEXT
    CMP CL, 39H    ;大于9则略过
    JA NEXT
    INC COUNT      ;计数
    JMP NEXT
DISP: ~~MOV DL,~~   ;输出结果
```

5.16

```
DSEG SEGMENT
   DATA DW 100H DUP(?)
DSEG ENDS
;
CSEG SEGMENT
   MAIN PROC FAR
        ASSUME CS:CSEG,DS:DSEG
   START: PUSH DS
        SUB AX,AX
        PUSH AX.
        MOV AX.DSEG
        MOV DS,AX   ;设置DS
  BEGIN: MOV CX,100H
        MOV SI,0
        MOV BX,0
        MOV DI,0
  NEXT: MOV AX,DATA[SI]
        CWD
        ADD BX,AX      ;累加
        ADC DI,DX
        ADD SI,2
        LOOP NEXT
        MOV DX,DI
        MOV AX,BX
        MOV CX,100H
        IDIV CX        ;计算平均数
        MOV BX,0
        MOV SI,0
```

```
COMP:  CMP   AX,DATA[SI]
       JLE   NO
       INC   BX        ; 对小于平均值计数
NO:    ADD   SI,2
       LOOP  COMP
       RET
MAIN   ENDP
CSEG   ENDS
; END  START
```

题目 1

循环部分的代码需要反复执行，当循环次数增多时，循环不变量可以被反复使用多次，而分支不变量只能在分支中执行一次，因此循环不变量可以带来的收益更大。

题目 2

循环语句可以根据缓存的大小来选择合适的数组结构或者选择合适的循环展开次数，如果循环展开过大，会造成缓存不命中的次数增加，降低效率。在访问多维数组时，也需要按照合适的访问顺序，才能确保在当前的缓存大小下的命中率更高。

题目 3

避免使用向量指令集的情况为：使用移动设备等低功率的设备进行图像处理，内存带宽有限的情况。

向量化会增加 CPU 的功耗，对于一些移动设备，高功耗会缩短设备使用时间，影响体验。

向量化需要一次处理一组数据，因此对于内存带宽要求较高，当内存带宽有限时不利于指令执行。

题目 4

FFTW 采用了自适应算法选择机制，可以根据输入数据的大小、类型和硬件特性自动选择最优的算法。用户在调用 FFT 之前需要创建一个计划，FFTW 会在计划阶段进行优化。

用户通过调用 fftw_plan_dft_1d、fftw_plan_dft_2d 等函数创建一个计划。创建计划时，FFTW 会根据输入数据的大小和类型进行一系列优化，包括选择最佳的

算法、确定内存布局和数据对齐等。FFTW 支持多线程计算，可以通过 OpenMP 或 Pthreads 等多线程库实现并行化。用户可以通过设置环境变量或 API 函数来启用多线程支持，从而在多核处理器上获得更好的性能。