第二章 答案  
Tarzan 版

题2.1 8086/8088通用寄存器的通用性表现在何处？8个通用寄存器各自有何专门用途？哪些  
      寄存器可作为存储器寻址方式的指针寄存器？

答：8086/8088通用寄存器的通用性表现在：  
    这些寄存器除了各自规定的专门用途外，他们均可以用于传送和暂存数据，可以保存  
    算术逻辑运算中的操作数和运算结果；  
    8个通用寄存器的专门用途如下：  
    AX 字乘法，字除法，字I/O  
    BX 存储器指针  
    CX 串操作或循环控制中的计数器  
    DX 字乘法，字除法，间接I/O  
    SI 存储器指针（串操作中的源指针）  
    DI 存储器指针（串操作中的目的指针）  
    BP 存储器指针（存取堆栈的指针）  
    SP 堆栈指针  
     
    其中BX，SI，DI，BP可作为存储器寻址方式的指针寄存器

题2.2  从程序员的角度看，8086/8088有多少个可访问的16位寄存器？有多少个可访问的8位  
       寄存器？

答：   从程序员的角度看，8086/8088有14个可访问的16位寄存器；有8个可访问的8位寄存器；

题2.3  寄存器AX与寄存器AH和AL的关系如何？请写出如下程序片段中每条指令执行后寄存器  
      AX的内容：  
      MOV AX,1234H  
      MOV AL,98H  
      MOV AH,76H  
      ADD AL,81H  
      SUB AL,35H  
      ADD AL,AH  
      ADC AH,AL  
      ADD AX,0D2H  
      SUB AX,0FFH

答：  MOV AX,1234H        AX=1234H  
      MOV AL,98H          AX=1298H  
      MOV AH,76H          AX=7698H  
      ADD AL,81H          AX=7619H  
      SUB AL,35H          AX=76E4H  
      ADD AL,AH           AX=765AH  
      ADC AH,AL           AX=D15AH  
      ADD AX,0D2H         AX=D22CH  
      SUB AX,0FFH         AX=D12DH

题2.4  8086/8088标志寄存器中定义了哪些标志？这些标志可分为哪两类？如何改变这些标志  
       的状态？

答：  8086/8088标志寄存器中定义了9个标志，如下：  
      CF: Carry Flag  
      ZF: Zero Flag  
      SF: Sign Flag  
      OF: Overflow Flag  
      PF: Parity Flag  
      AF: Auxiliary Carry Flag  
      DF: Direction Flag  
      IF: Interrupt-enable Flag  
      TF: Trap Flag  
       
      这些标志可分为两类，分别为：  
      1、运算结果标志；  
      2、状态控制标志；

      采用指令SAHF可把AH中的指定位送至标志寄存器低8位SF、ZF、AF、PF、CF；  
      采用CLC可清除CF，置CF到0  
      采用STC可置CF到1  
      采用CLD可置DF到0  
      采用sTD可置DF到1  
      采用CLI可置IF到0  
      采用STI可置IF到1  
      另外，在某些指令执行过程中会改变部分标志的状态；

题2.5  请说说标志CF和标志OF的差异。

答：   如果把指令中处理的数据按照无符号数看待，则处理结果达到进位是，置CF为1；  
       如果把该处理中的数据按照有符号数看待，则处理结果超过有符号数表达范围的，  
       置OF为1；两个标志同步进行，CPU并不知道该数的类型；

题2.6  8086/8088如何寻址1M字节的存储器物理地址空间？在划分段时必须满足的两个条件  
       是什么？最多可把1M字节空间划分成几个段？最少可把1M字节地址空间划分成几个段？

答：   8086/8088通过对存储器分段和使用段寄存器的方式寻址1M字节的存储器物理地址空间；  
       在划分段时必须满足的两个条件是：  
       1、逻辑段的开始地址必须是16的倍数；  
       2、逻辑段的嘴道长度是64K；  
       1M的字节空间划分为64K个逻辑段；最少可把1M字节地址划分成16个逻辑段；

题2.7  在8086/8088上运行的程序某一时刻最多可访问几个段？程序最多可具有多少个段？  
       程序至少几个段？

答：   在8086/8088上运行的程序某一时刻最多可访问4个当前段：代码段，数据段，堆栈段  
       和附加段；程序最多可具有4种类型的段，最少要有一个代码段；

题2.8  存储单元的逻辑地址如何表示？存储单元的20位物理地址如何构成？

答：   存储单元的逻辑地址由段值和偏移两部分组成：段值：偏移；  
       存储单元的20位物理地址可以表示为：  
       物理地址＝段值×16＋偏移；

题2.9  当段重叠时，一个存储单元的地址可表示成多个逻辑地址。请问物理地址12345H可表示  
       多少个不同的逻辑地址？偏移最大的逻辑地址是什么？偏移最小的逻辑地址是什么？

答：   12345H可表示1000H（4096）个不同的逻辑地址，偏移最大的逻辑地址是235:0FFF5H  
       偏移最小的逻辑地址是1234:0005H

题2.10 为什么称CS为代码段寄存器？为什么称SS为堆栈寄存器？

答：   因为在取指令的时候，规定的段寄存器就是CS，所以CS为代码段寄存器；  
       而堆栈操作时规定的寄存器是SS，所以SS为堆栈寄存器；

题2.11 请举例说明何为段前缀超越。什么场合下要使用段前缀超越？

答：   在存取一般存储器操作数时，段寄存器可以不是DS；当偏移设计BP寄存器时，段寄存器  
       也可以不必是SS；如Mov AX,[si] 默认段地址在DS中，也可以改变：Mov AX, ES:[si]  
       当数据并不在默认的DS指定段时，可以采用段前缀超越；

题2.12 8086/8088的基本寻址方式可分为哪三类？他们说明了什么？

答：   8086/8088的基本寻址方式可分为以下三类：  
       1、存储器寻址；  
       2、立即寻址；  
       3、寄存器寻址；  
       他们说明了cpu有三类合计七种方式进行基本寻址；

题2.13 存储器寻址方式分为哪几种？何为存储器的有效地址？

答：   存储器寻址方式分为以下几种：  
       1、立即寻址；  
       2、直接寻址；  
       3、寄存器寻址；  
       4、寄存器间接寻址；  
       5、寄存器相对寻址；  
       6、基址加变址寻址；  
       7、相对基址加变址寻址；

       存储器的有效地址是一个16bit的无符号数；

题2.14 什么场合下缺省的段寄存器是SS？为什么这样安排？

答：   当使用堆栈时，缺省的段寄存器是SS；  
       因为SS定义为堆栈段寄存器，配合SP堆栈指针，用来指向堆栈的栈顶；

题2.15 请说明如下指令中源操作数的寻址方式，并作相互比较:  
       MOV BX,[1234H]  
       MOV BX,1234H  
       MOV DX,BX  
       MOV DX,[BX]  
       MOV DX,[BX+1234H]  
       MOV DX,[BX+DI]  
       MOV DX,[BX+DI+1234H]

答：   MOV BX,[1234H]               ；直接寻址  
       MOV BX,1234H                 ：立即寻址  
       MOV DX,BX                    ：寄存器寻址  
       MOV DX,[BX]                  ：寄存器间接寻址  
       MOV DX,[BX+1234H]            ：寄存器相对寻址  
       MOV DX,[BX+DI]               ：基址加变址寻址  
       MOV DX,[BX+DI+1234H]         ：相对基址加变址寻址

题2.16 8086/8088提供了灵活多样的寻址方式，如何适当的选择寻址方式？

答：   每种寻址方式都有其特点，首先应该掌握不同寻址方式之间的区别，以及  
       适用的范围，结合程序中的需要进行灵活选择。

题2.17 设想一下这些寻址方式如何支持高级语言的多种数据结构？

答：   自己设想！

题2.18 为什么目标操作数不能采用立即寻址方式？

答：   立即寻址表示是一个操作数，并非一个存储空间，作为目标操作数是不合适的；

题2.19 处理器的通用寄存器是否越多越好？通用寄存器不够用怎么办？

答：   处理器的通用寄存器并非越多越好，因为如果处理器的通用寄存器数量太多，势必  
       造成处理器的成本增加，同时也增加了处理器设计的复杂度；  
       如果通用寄存器不够用，应该采用内存中的存储单元代替，不过速度上要有所牺牲；

题2.20 哪些存储器寻址方式可能导致有效地址超出64K的范围？8086/8088如何处理这种  
       情况？

答：   寄存器相对寻址，基址加变址寻址，相对基址加变址寻址这三种寻址方式有可能导致  
       有效地址超出64K的范围，8086/8088将取其64K的模进行访问；

题2.21 什么情况下根据段值和偏移确定的存储单元地址会超出1M？8086/8088如何处理这种  
       情况？

答：   当物理地址的计算超过FFFFFH时，存储单元地址会超出1M，8086/8088将取其1M的模  
       覆盖存取；

题2.22 8086/8088的指令集可分为哪6个子集？

答：   8086/8088的指令集可分为以下6个子集：  
       1、数据传输  
       2、算术运算  
       3、逻辑运算  
       4、串操作  
       5、程序控制  
       6、处理器控制

题2.23 8086/8088的指令集合中，最长的指令有几个字节？最短的指令有几个字节？

答：   8086/8088的指令集合中，最长的指令4个字节，最短的指令2个字节；  
       MOV AX,[BX+SI+1234H]

题2.24 8086/8088的算术逻辑运算指令最多一次处理多少二进制位？当欲处理的数据  
       长度超出该范围怎么办？

答：   8086/8088的算术逻辑运算指令最多一次处理16bit的二进制位；如果处理的数据  
       长度超出则分成若干部分进行逻辑运算，最后进行整合；

题2.25 如何时序数据段和代码段相同？

答：   将数据段的内容写入代码段中，并将代码段的段值赋给DS即可；

题2.26 通常情况下源操作数和目的操作数不能同时是存储器操作数。请给出把存储器操作  
       数甲送到存储器操作数乙的两种方法。

答：    
       法一：  
       MOV AX, [BX]  
       MOV [SI],AX       DS:[BX]=甲，DS:[SI]=乙  
       法二：  
       MOV AX,[BX]  
       XCHG AX,[SI]  
       法三：  
       PUSH WORD PTR [BX]  
       POP  WORD PTR [SI]

题2.27 请用一条指令实现把BX的内容加上123并把和送到寄存器AX。

答：   LEA AX, [BX+123H]

题2.28 堆栈有哪些用途？请举例说明。

答：   堆栈的用途主要有：  
       1、现场和返回地址的保护；  
       MOV AX, OFFSET ADDRESS  
       PUSH AX  
       JMP XXX  
       ...  
       RET

       2、寄存器内容的保护；  
       PUSH AX  
       PUSH BX  
       ...  
       POP BX  
       POP AX

       3、传递参数；  
       PUSH [BX]  
       CALL XXX  
       ...  
       XXX:  
       POP AX  
       ...

       4、存储局部变量；  
       PUSH DS  
       PUSH CS  
       POP DS  
       ...  
       POP DS

题2.29 在本章介绍的8086/8088指令中，哪些指令把寄存器SP作为指针使用？8086/8088指令  
       集中，哪些指令把寄存器SP作为指针使用？

答：   以下指令把寄存器SP作为指针使用：  
       1、PUSH  
       2、POP  
       3、PUSHF  
       4、POPF

       5、PUSHA  
       6、POPA  
       7、RET  
       8、CALL  
       9、RETF  
      
题2.30 请说说标志CF的用途。请至少给出使标志CF清0的三种方法。

答：   CF的用途主要有：  
       1、配合条件转移语句进行条件转移；  
       2、配合移位指令实现操作数之间的位转移；  
       3、常作为子程序的出口参数；如DOS磁盘文件管理功能调用等；

       CF清0的方法：  
       法一：  
       CLC  
       法二：  
       ADD AX,0FFFFH  
       法三：  
       CMP AX,0

题2.31 请写出如下程序片段中每条算术运算指令执行后标志CF、ZF、SF、OF、PF和AF的状态。  
       MOV AL,89H  
       ADD AL,AL  
       ADD AL,9DH  
       CMP AL,0BCH  
       SUB AL,AL  
       DEC AL  
       INC AL  
        
答：   INSTRUCTION           CF    ZF    SF    OF    PF    AF  
       MOV AL,89H            0     0     0     0     0     0  
       ADD AL,AL             1     0     0     1     1     1  
       ADD AL,9DH            0     0     1     0     1     0  
       CMP AL,0BCH           1     0     1     0     1     0  
       SUB AL,AL             0     1     0     0     1     0  
       DEC AL                0     0     1     0     1     1  
       INC AL                0     1     0     0     1     1  
        
题2.32 什么是除法溢出？如何解决16位被除数8位除数可能产生的溢出？

答：   除法溢出是指除数如果是0，或者在8位除数时商超过8位，或者在16位除时商超过16位，  
       则认为是除法溢出，引起0中断；  
        
       首先要确定8位除数不能为0，其次要确定商的最大值不能超过8位，如果超过8位，则可  
       采用16位的除法；  
        
题2.33 请写出如下程序片段中每条逻辑运算指令执行后标志ZF、SF、PF的状态：  
       MOV AL,45H  
       AND AL,0FH  
       OR AL,0C3H  
       XOR AL,AL  
        
答：   INSTRUCTION           ZF     SF     PF  
       MOV AL,45H            0      0      0  
       AND AL,0FH            0      0      1  
       OR AL,0C3H            0      1      0  
       XOR AL,AL             1      0      1  
        
题2.34 “MOV AX,0”可寄存器AX清0。另外再写出三条可使寄存器AX清0的指令。

答：   法一：  
       XOR AX,AX  
       法二：  
       AND AX,0  
       法三：  
       SUB AX,AX  
        
题2.35 请写出如下程序片段中每条移位指令执行后标志CF、ZF、SF和PF的状态。  
       MOV AL,84H  
       SAR AL,1  
       SHR AL,1  
       ROR AL,1  
       RCL AL,1  
       SHL AL,1  
       ROL AL,1  
        
答：   INSTRUCTION          CF     ZF     SF     PF  
       MOV AL,84H           0      0      0      0  
       SAR AL,1             0      0      1      0  
       SHR AL,1             0      0      0      0  
       ROR AL,1             1      0      0      0    （该命令不影响SF位）  
       RCL AL,1             1      0      0      0  
       SHL AL,1             0      0      1      0  
       ROL AL,1             1      0      1      0  
        
题2.36 8086/8088中，哪些指令把寄存器CX作为计数器使用？哪些指令把寄存器BX作为基指针寄存器使用？

答：   8086/8088中，以下指令把寄存器CX作为计数器使用：  
       1、LOOP             
       2、LOOPE  
       3、LOOPZ  
       4、LOOPNZ  
       5、LOOPNE  
       6、JCXZ  
        
       以下指令把寄存器BX作为基指针寄存器使用：  
       1、MOV  
       2、XCHG  
       3、LEA  
       4、LDS  
       5、LES  
       6、ADD  
       ...  
        
题2.37 请不用条件转移指令JG、JGE、JL和JLE等指令实现如下程序片段的功能：  
       CMP AL,BL  
       JGE OK  
       XCHG AL,BL  
   OK: ......  
    
答：   如下命令可实现同样功能：  
       PUSH CX            ;Reserve CX  
       XOR CX,CX          ;CX=0  
       MOV CH,02H         ;CH=02H  
       MOV CL,AL          ;CL=AL  
       MOV BH,0H          ;BH=0  
       SUB CX,BX          ;If CH=2, AL>=BL; If CH=1, AL<BL  
       SHR CX,8           ;CX=2 or 1  
       LOOP OK            ;If CX=2 jmp to OK; If CX=1 Exchange AL,BL  
       XCHG AL,BL  
OK:  
       POP CX             ;Revert CX     
       ......        
           
题2.38 段间转移和段内转移的本质区别是什么？8086/8088哪些指令可实现段间转移？

答：   段间转移和段内转移的本质区别是有没有对CS进行设置，如果设置了新的CS代码寄存器，  
       程序将转移到另一个段中，即实现了段间转移；否则CS和原来一致，则在同一代码段中  
       继续进行，只是IP指针进行了调整，即为段内转移；  
       8086/8088中如下指令可以实现段间转移：  
       1.JMP FAR PTR LEAEL  
       2.JMP OPRD  
        
       3.CALL  
       4.RET/RETF  
        
题2.39 8086/8088的条件转移指令的转移范围有多大？如何实现超出范围的条件转移？

答：   8086/8088的条件转移指令的转移范围只能从－126到＋129之间，如果出现超出  
       范围的条件转移，要借助无条件转移命令JMP；  
        
题2.40 相对转移和绝对转移的区别是什么？相对转移的有何优点？

答：   相对转移和绝对转移的区别是相对转移记录了目标地址与当前地址的差值，而绝对  
       转移在转移命令中直接包含了目标地址；  
       相对转移有利于程序的浮动，比如说增加了命令语句等；  
        
题2.41 请指出下列指令的错误所在：  
       MOV CX,DL  
       XCHG [SI],3  
       POP CS  
       MOV IP,AX  
       SUB [SI],[DI]  
       PUSH DH  
       OR BL,DX  
       AND AX,DS  
       MUL 16  
       AND 7FFFH,AX  
       DIV 256  
       ROL CX,BL  
       MOV ES,1234H  
       MOV CS,AX  
       SUB DL,CF  
       ADC AX,AL  
       MOV AL,300  
       JDXZ NEXT  
        
答：   MOV CX,DL             ；寄存器大小不一  
       XCHG [SI],3           ；不能与立即数进行交换  
       POP CS                ；POP指令的对象不能是CS，PUSH可以  
       MOV IP,AX             ；IP不能是源也不能是目的  
       SUB [SI],[DI]         ；如果参与的操作数有两个，只能有一个是存储器操作数  
       PUSH DH               ；PUSH和POP只能处理16位的操作数（8086/8088）  
       OR BL,DX              ；寄存器大小不一  
       AND AX,DS             ；段寄存器不可以是操作数  
       MUL 16                ；不可以使用立即数  
       AND 7FFFH,AX          ；立即数不能是目的操作数  
       DIV 256               ；不可以使用立即数  
       ROL CX,BL             ；BL不可以作为操作数  
       MOV ES,1234H          ；段寄存器为目的时，源不能是立即数，需由通用寄存器转  
       MOV CS,AX             ；代码段寄存器CS不能为目的  
       SUB DL,CF             ；CF是Flag中的一个bit，不能如此  
       ADC AX,AL             ；寄存器大小不一  
       MOV AL,300            ；300超过0FFh,Over 8bit  
       JDXZ NEXT             ；JCXZ  
        
题2.42 请指出如下指令哪些是错误的，并说明原因：  
       MOV [SP],AX  
       PUSH CS  
       JMP BX+100H  
       JMP CX  
       ADD AL,[SI+DI]  
       SUB [BP+DI-1000],AL  
       ADD BH,[BL-3]  
       ADD [BX],BX  
       MOV AX,BX+DI  
       LEA AX,[BX+DI]  
       XCHG ES:[BP],AL  
       XCHG [BP],ES  
        
答：   MOV [SP],AX            ；SP非有效寄存器间接寻址之寄存器  
       PUSH CS                ；对  
       JMP BX+100H            ；对  
       JMP CX                 ；对  
       ADD AL,[SI+DI]         ；SI和DI只能出现一个，与BX，BP一致  
       SUB [BP+DI-1000],AL    ；对  
       ADD BH,[BL-3]          ；BL只是一个8bit寄存器  
       ADD [BX],BX            ；对  
       MOV AX,BX+DI           ；对  
       LEA AX,[BX+DI]         ；对  
       XCHG ES:[BP],AL        ；对  
       XCHG [BP],ES           ；段寄存器不能是操作数    
        
题2.43 下列程序片段完成什么功能，可否有更简单的方法实现同样的功能：  
       XCHG AX,[SI]  
       XCHG AX,[DI]  
       XCHG AX,[SI]  
        
答：   程序实现[SI]和[DI]中的内容交换；AX中内容不变；  
       有，如下：  
       PUSH [SI]  
       PUSH [DI]  
       POP  [SI]  
       POP  [DI]        
      
题2.44 请比较如下指令片段：  
       LDS SI,[BX]  
        
       MOV SI,[BX]  
       MOV DS,[BX+2]  
        
       MOV DS,[BX+2]  
       MOV BX,[BX]  
        
答：   LDS SI,[BX]            ；DS=[BX+2],SI=[BX]  
        
       MOV SI,[BX]            ; DS=[BX+2],SI=[BX]  
       MOV DS,[BX+2]  
        
       MOV DS,[BX+2]          ; DS=[BX+2],BX=[BX]  
       MOV BX,[BX]

       第一组和第二组功能一致；

第三章：  
Tarzan版

题3.1  伪指令语句与指令语句的本质区别是什么？伪指令的主要作用是什么？

答：   伪指令语句与指令语句的本质区别是指令语句有其对应的机器指令，而伪指令没有；  
       伪指令的主要作用是指示汇编程序如何汇编源程序；  
        
题3.2  汇编语言中的表达式与高级语言中的表达式有何相同点和不同点？

答：   汇编语言中的表达式与高级语言中的表达式的相同点是都采用运算符、操作符以及括号  
       把常数和符合连起来；  
       不同点是汇编语言的表达式除了数值表达式外还有地址表达式；  
        
题3.3  汇编语言中数值表达式与地址表达式有何区别？

答：   汇编语言中数值表达式在汇编过程中由汇编程序计算出数值，而地址表达式中部分相对地址  
       的地方，在汇编时无法确定其确定地址；  
        
题3.4  汇编语言中的变量和标号有何异同之处？

答：   汇编语言中的变量和标号的相同之处是都代表着一个地址；  
       不同之处是变量表示的地址中存放的是数据，而标号表示的地址中存放的是代码；  
        
题3.5  请计算如下各数值表达式的值：  
       23H AND 45H OR 67H  
       1234H/16+10H  
       NOT(65535 XOR 1234H)  
       1024 MOD 7+3  
       LOW 1234 OR HIGH 5678H  
       23H SHL 4  
       "Eb" GE 4562H XOR -1  
       1234H SHR 6  
       'a' AND (NOT ('a'-'A')  
       'H' OR 00100000B  
       76543Q LT  32768 XOR 76543  
       3645H AND 0FF00H  
        
答：   23H AND 45H OR 67H                ;67H  
       1234H/16+10H                      ;133H  
       NOT(65535 XOR 1234H)              ;1234H  
       1024 MOD 7+3                      ;5  
       LOW 1234 OR HIGH 5678H            ;D6H 注意1234 不是1234H  
       23H SHL 4                         ;30H  
       "Eb" GE 4562H XOR -1              ;0  
       1234H SHR 6                       ;0048H  
       'a' AND (NOT ('a'-'A')            ;41H or 'A'  
       'H' OR 00100000B                  ;68H or 'h'  
       76543Q LT  32768 XOR 76543        ;题目最后的76543有错，按照76543Q处理：829CH  
       3645H AND 0FF00H                  ;3600H  
        
题3.6  请计算如下程序片段中各地址表达式的值，设BX＝1000H，SI＝2000H，DI＝3000H，  
       BP＝4000H  
       [BX+100H]  
       [DI][BP]  
       2000H[SI]  
       10H[BX][SI]  
       [BP-128]  
       [BX][DI-2]  
        
答：   [BX+100H]                         ;[1100H]  
       [DI][BP]                          ;[7000H]  
       2000H[SI]                         ;[4000H]  
       10H[BX][SI]                       ;[3010H]  
       [BP-128]                          ;[3F80H]  
       [BX][DI-2]                        ;[3FFEH]   
        
题3.7  设在某个程序中有如下片段，请写出每条传送指令执行后寄存器AX的内容：  
              ORG 100H  
       VARW   DW  1234H,5678H  
       VARB   DB  3,4  
       VARD   DD  12345678H  
       BUFF   DB  10 DUP(?)  
       MESS   DB  'HELLO'  
       BEGIN: MOV AX,OFFSET VARB + OFFSET MESS  
              MOV AX,TYPE BUFF + TYPE MESS + TYPE VARD  
              MOV AX,SIZE VARW + SIZE BUFF + SIZE MESS  
              MOV AX,LENGTH VARW + LENGTH VARD  
              MOV AX,LENGTH BUFF + SIZE VARW  
              MOV AX,TYPE BEGIN  
              MOV AX,OFFSET BEGIN  
               
答：          ORG 100H  
       VARW   DW  1234H,5678H  
       VARB   DB  3,4  
       VARD   DD  12345678H  
       BUFF   DB  10 DUP(?)  
       MESS   DB  'HELLO'  
       BEGIN: MOV AX,OFFSET VARB + OFFSET MESS              ;AX=0218H  
              MOV AX,TYPE BUFF + TYPE MESS + TYPE VARD      ;AX=0006H  
              MOV AX,SIZE VARW + SIZE BUFF + SIZE MESS      ;AX=000DH  
              MOV AX,LENGTH VARW + LENGTH VARD              ;AX=0002H  
              MOV AX,LENGTH BUFF + SIZE VARW                ;AX=000CH  
              MOV AX,TYPE BEGIN                             ;AX=FFFFH  
              MOV AX,OFFSET BEGIN                           ;AX=0119H  
        
题3.8  设如下两条指令中的符号ABCD是变量名，请说明这两条指令的异同。  
       MOV AX,OFFSET ABCD  
       LEA AX,ABCD  
        
答：   两条指令都是将ABCD的偏移地址放入AX寄存器中；  
       不同之处是OFFSET只能取得用数据定义伪指令的变量的有效地址，而不能取得一般操作  
       数的有效地址；  
        
题3.9  请指出如下指令的不明确之处，并使其明确：  
       MOV ES:[BP],5  
       ADD CS:[1000H],10H  
       DEC SS:[BX-8]  
       JMP CS:[SI+1000H]  
       MUL [BX+DI+2]  
       DIV [BP-4]  
        
答：   MOV ES:[BP],5                ；未指定存储单元属性 MOV WORD PTR ES:[BP],5  
       ADD CS:[1000H],10H           ；同上 ADD WORD PTR CS:[1000H],10H  
       DEC SS:[BX-8]                ；同上 DEC WORD PTR SS:[BX-8]  
       JMP CS:[SI+1000H]            ；无法确定段间还是段内转移  
                                      JMP WORD PTR CS:[SI+1000H]  
       MUL [BX+DI+2]                ；无法确定是8位乘法还是16位乘法  
                                      MUL WORD PTR [BX+DI+2]  
       DIV [BP-4]                   ；同上 DIV WORD PTR [BP-4]  
        
题3.10 设在某个程序中有如下片段，请改正其中有错误的指令语句：  
       VARW   DW  1234H,5678H  
       VARB   DB  3,4  
       VARD   DD  12345678H  
       ......  
              MOV AX,VARB  
              MOV VARD,BX  
              MOV VARD+2,ES  
              MOV CL,VARW+3  
              LES DI,VARW  
               
答：          MOV AX,VARB           ；VARB是8bit量，应该修改AX到AL or AH  
              MOV VARD,BX           ；VARD是32bit量，要分两次传  
              MOV VARD+2,ES         ；同上  
              MOV CL,VARW+3         ；同上，CL改为CX  
              LES DI,VARW           ；VARW非32位量，应改为VARD  
               
题3.11  请举例说明伪指令ASSUME的作用。

答：    ASSUME的作用是声明现在开始CS寄存器对应于哪个段，DS对应于哪个段，SS和ES  
        分别对应哪个段，可以相同也可以不同；  
        如：  
        ASSUME CS:CSEG,DS:DSEG,SS:SSEG,ES:ESEG  
        可以根据需要重新建立对应关系；  
         
题3.12  设在某个程序片段中有如下语句，请说明各符号的属性：  
        SYMB1 LABEL BYTE  
        SYMB2 EQU THIS BYTE  
        SYMB3 DW ?  
        SYMB4 EQU BYTE PTR SYMB3  
         
答：    SYMB1：BYTE  
        SYMB2：BYTE  
        SYMB3：WORD  
        SYMB4：BYTE

题3.13  为什么说汇编语言中的等价语句EQU可理解为简单的宏定义？请举例说明。

答：    EQU可以用符号定义常数，表达式，指令助记符，字符串等；  
        而宏定义是指定一个宏指令名，宏指令可表示相对应的程序片段。  
        如：  
        HELLO EQU "How are you !"  
        与：  
        HELLO MACRO  
        'How are you !'  
        ENDM  
         
        一致；  
         
题3.14  设在某个程序片段中有如下语句，请说明各符号所表示的值：  
        SYMB1 = 10  
        SYMB2 = SYMB1\*2  
        SYMB1 = SYMB1 + SYMB2 + 4  
        SYMB3 EQU SYMB1  
         
答：    SYMB1 = 22H           
        SYMB2 = 14H  
        SYMB3 = 22H       
                 
题3.15  请改写3.3.3的程序T3－1.ASM，使其只有一个段。

答：    ;程序名：T3-1.ASM  
        ;功能  ：显示信息“HELLO"

        cseg segment  
             assume cs:cseg  
             mess  db 'HELLO',0dh,0ah,'$'  
        start:  
             mov ax,cseg  
             mov ds,ax  
             mov dx,offset mess  
             mov ah,9  
             int 21h  
             mov ah,4ch  
             int 21h              
        cseg ends  
             end start   
              
题3.16  请说明指令”JMP $+2“指令的机器码中的地址差值是多少？

答：    2H

题3.17  源程序是否一定要以END语句结束？程序是否一定从代码段的偏移0开始执行？  
        如果不是，那么如何指定？  
         
答：    源程序可以不以END语句结束，不过END之后的内容汇编程序将忽略。  
        程序不一定要从代码的偏移0开始执行，一个比较简单的方法是利用END语句，  
        如END XXX，程序将从XXX标号处开始执行；  
         
题3.18  利用查表的方法实现代码转换有何特点？利用查表的方法求函数值有何特点？

答：    利用查表的方法实现代码转换的特点是：  
        1、转换代码间不需要直接的算术或逻辑关系，只需要安排好表的组织即可；  
        2、对于部分代码，其转换效率比较高，主要时间用在寻址上；  
        
        利用查表的方法求函数值的特点是：  
        1、对于大部分的数学函数值的求值，直接计算困难较大，采用查表法可祢补  
        不足；  
        2、程序比较简单；  
        3、能够得到十进制或者十六进制格式的高精度函数值。  
        4、函数值必须事先计算好；  
        5、精度无法由程序控制；  
         
题3.19  利用地址表实现多向分支有何特点？请举例说明。

答：    利用地址表实现多向分支的特点有：  
        1、对于实现5路以上的多向分支，使用地址表既方便又高效；  
        2、对于如何确定地址的位置，需要采用不同的方法实现；  
        例子看书。  
         
题3.20  请举例说明如何避免条件转移超出转移范围。

答：    如果出现条件转移超出了范围，则可以利用无条件转移指令帮助跳转；  
        如：  
        cmp ax,'A'  
        jb out\_program  
         
        如果超出范围：  
        cmp ax,'A'  
        jb out\_com  
        ...  
        out\_com:     
        jmp far ptr out\_program  
         
题3.21  请写一个程序片段统计寄存器AX中置1的个数。

答：   count db ?,?,0dh,0ah,'$'  
       ... ...  
       call countAX  
        
       cmp bl,9  
       ja sub10  
       jmp go  
sub10:  
       sub bl,10  
       mov count,31h  
go:           
       add bl,30h  
       mov count+1,bl  
       mov dx,offset count  
       mov ah,9  
       int 21h  
       mov ah,4ch  
       int 21h  
;==============================  
        ;入口：AX  
        ;出口：BL＝AX中1的个数  
        countAX proc  
        mov cx,16  
        mov bl,0  
count1:      
        shl ax,1  
        jnc ADDAX1  
        add bl,1  
ADDAX1:      
        loop count1  
        ret   
        countAX endp          
;=============================                                                   
   
题3.22  设一个32位有符号数存放在DX：AX中，请写一个求其补码的程序片段。  
                
答：    Invert proc  
        mov bx,dx  
        and bx,8000h  
        cmp bx,0  
        jz out\_1  
        not dx  
        not ax  
        add ax,1  
        adc dx,0  
        or  dx,8000h         
out\_1:  
        nop         
        ret  
        Invert endp  
         
题3.23  写一个程序片段实现如下功能：依次重复寄存器AL中的每一位，得到16位的结果存  
        放到DX寄存器中。  
         
答：    Expand proc  
        mov cx,7  
        xor dx,dx  
         
   S0:   
        shl dx,1  
        shl dx,1  
        shl al,1  
        jnc CF0  
        add dx,3h  
   CF0:  
        nop           
        loop S0  
        ret  
        Expand endp     
    
题3.24  写一个程序片段实现如下功能：依次重复四次寄存器AL中的每一位，得到32位的结果  
        存放到DX：AX寄存器中。  
         
答：    Expand proc  
        mov cx,3  
        xor dx,dx  
        xor bx,bx  
   S0:  
        shl al,1  
        jnc CF0  
        add dx,0Fh  
   CF0:  
        shl dx,1  
        shl dx,1  
        shl dx,1  
        shl dx,1          
        loop S0  
         
        mov cx,4  
   S1:  
        shl al,1  
        jnc CF0\_1  
        add bx,0Fh  
   CF0\_1:  
        shl bx,1  
        shl bx,1  
        shl bx,1  
        shl bx,1          
        loop S1      
        mov ax,bx     
        ret  
   Expand endp   
    
题3.25  写一个程序片段实现如下功能：把寄存器AL和BL中的位依次交叉，得到的16位结果  
        存放到DX寄存器中。  
         
答：    Expand proc  
        mov cx,8  
        xor dx,dx  
   S0:  
        shl dx,1  
        shl al,1  
        jnc CF0  
        add dx,1h  
   CF0:  
        shl dx,1  
        shl bl,1  
        jnc CF0\_1  
        add dx,1h  
   CF0\_1:        
        loop S0  
        ret  
        Expand endp   
         
题3.26  写一个优化的程序片段，实现把字符串中的小写子母变换为对应的大写子母。设字符串  
        以0结尾。  
         
答：    InvertC proc  
  begin:  
        mov al,mess[si]  
        cmp al,'0'  
        jz exit\_1     
        cmp al,61h  
        jb next  
        cmp al,7AH  
        ja next  
        and al,11011111b  
        mov mess[si],al  
 next:  
        inc si  
        jmp begin  
exit\_1:  
        nop  
        ret  
        InvertC endp   
         
题3.27  写一个优化的程序片段，统计字符串的长度。设字符串以0结尾。

答：    count proc  
        mov al,mess[si]  
        cmp al,'0'  
        jz exit\_2  
        inc si  
        jmp count  
   exit\_2:  
        nop  
   ret  
   count endp        ;si=数量  
    
题3.28  写一个程序片段，滤去某个字符串中的空格符号（ASCII码20H），设字符串以0结尾。

答：    DeleteSpace proc     ;设si＝0，bx＝0，Mess为字符串首地址  
        mov al,mess[si]  
        cmp al,'0'  
        jz exit\_2  
        cmp al,' '  
        jz next3  
        xchg al,mess[bx]  
        xchg al,mess[si]  
        inc bx  
   next3:  
        inc si    
        jmp DeleteSpace  
   exit\_2:  
        nop  
   ret  
   DeleteSpace endp

题3.29  请写一个把两个字符串合并的示例程序。

答：    dseg segment  
        string1 db 'Welcome to $'  
        string2 db 'Beijing !$'  
   dseg ends  
    
   cseg segment  
        assume cs:cseg,ds:dseg  
   start:  
        mov ax,dseg  
        mov ds,ax  
         
        xor bx,bx  
        xor si,si  
         
   keepfind:  
        mov al,string1[bx]  
        cmp al,'$'  
        jz combine  
        inc bx  
        jmp keepfind  
   combine:  
        mov al,string2[si]  
        mov string1[bx],al  
        cmp al,'$'  
        jz exit  
        inc si  
        inc bx  
        jmp combine  
   exit:  
      
        mov dx,offset string1  
        mov ah,09h  
        int 21h   
         
        mov ah,4ch  
        int 21h  
   cseg ends  
        end start

80X86汇编语言程序设计教程（杨季文）习题答案三下(2009-12-01 09:53:32)

|  |  |
| --- | --- |
| 标签：[mov](http://uni.sina.com.cn/c.php?t=blog&k=mov&ts=bpost&stype=tag) [ascii码](http://uni.sina.com.cn/c.php?t=blog&k=ascii%C2%EB&ts=bpost&stype=tag) [bl](http://uni.sina.com.cn/c.php?t=blog&k=bl&ts=bpost&stype=tag) [杨季文](http://uni.sina.com.cn/c.php?t=blog&k=%D1%EE%BC%BE%CE%C4&ts=bpost&stype=tag) [杂谈](http://uni.sina.com.cn/c.php?t=blog&k=%D4%D3%CC%B8&ts=bpost&stype=tag) |  |

Tarzan版

题3.30  请写一个可把某个字变量的值传唤为对应二进制数ASCII码串的示例程序。

答：    ;Name  : Show\_hex\_ascii  
        ;input : ah=Hex  
        ;output: dx='Hex'  
        show\_hex\_ascii proc  
         
               mov dh,ah  
               and dh,0f0h        ;reserve high 4bit  
               shr dh,4  
               add dh,30h         ;change 0-9 to '0-9'  
               cmp dh,39h  
               ja add\_dh\_7         
        dh\_ok:  
               mov dl,ah  
               and dl,0fh         ;reserve low 4bit  
               add dl,30h  
               cmp dl,39h  
               ja add\_dl\_7  
        dl\_ok:  
               nop  
               ret        
        add\_dl\_7:  
               add dl,7h           ;revert A-F to 'A-F'  
               jmp dl\_ok  
         
        add\_dh\_7:  
               add dh,7h  
               jmp dh\_ok  
                
        show\_hex\_ascii endp

题3.31  请写一个可把某个十进制数ASCII码串转换成对应非压缩BCD何压缩BCD的示例程序。

答：    ;非压缩BCD码  
        ;Input  ah=十进制数ASCII码  
        ;Output al=非压缩BCD码  
        TEST1 proc  
        cmp ah,'0'  
        jb exit  
        cmp ah,'9'  
        ja exit  
        sub ah,30h  
        mov al,ah  
 exit:   
        nop        
        ret  
        TEST1 endp  
         
        ;压缩BCD码  
        ;Input  ax=两个十进制数ASCII码  
        ;Output bl=压缩BCD码  
        TEST2 proc  
         
        cmp ah,'0'  
        jb exit  
        cmp ah,'9'  
        ja exit  
        sub ah,30h  
        mov bl,ah  
        shl bl,4  
         
        cmp al,'0'  
        jb exit  
        cmp al,'9'  
        ja exit1  
        sub al,30h  
        add bl,al  
exit1:  
        nop         
        ret  
        TEST2 endp

题3.32  请写一个可把某个十进制数ASCII码转换为对应的二进制的示例程序。

答：    table db '0000','0001','0010','0011','0100','0101','0110','0111'  
              db '1000','1001'  
        ......  
        ;Input  bl=一个十进制数ASCII码  
        ;Output dx:ax=二进制ASCII码  
        ;程序未检验该十进制数是否在范围以内  
        TEST3 proc  
        sub bl,30h  
        xor bh,bh  
        shl bx,1  
        shl bx,1  
        mov dh,table[bx]  
        mov dl,table[bx+1]  
        mov ah,table[bx+2]  
        mov al,table[bx+3]  
            
        ret  
        TEST3 endp   
         
题3.33  请写出一个可把某个十六进制数ASCII码转换为对应的二进制的示例程序。

答：    table db '0000','0001','0010','0011','0100','0101','0110','0111'  
              db '1000','1001','1010','1011','1100','1101','1110','1111'  
        ... ...  
        ;Input  bl=一个十六进制数ASCII码  
        ;Output dx:ax=二进制ASCII码  
        TEST3 proc  
        cmp bl,30h          
        jb exit1         ;小于30H的不在范围内  
        sub bl,30h         
        cmp bl,0Ah       ;如果在9以内，开始转换0-9  
        jb change1  
         
        sub bl,0Ah  
        cmp bl,6h        ;如果在‘9’－‘A’之间，不在范围内  
        jb exit1  
         
        sub bl,6h        ;‘A’＝0  
        cmp bl,7h        ;如果在‘A’－‘F’之间，开始转换  
        jb change2  
         
        cmp bl,21h       ;如果大于‘F’，看是否在‘F’和‘a’之间  
        jb exit1         ;如果在，则不在范围内  
        sub bl,20h       ;'a'=0  
        cmp bl,6h        ;如果大于‘f’，则不在范围内  
        ja exit1        
change2:  
        add bl,9h        ;按照table表，如果A＝0还需要加9才可以  
         
change1:         
        xor bh,bh  
        shl bx,1  
        shl bx,1  
        mov dh,table[bx]  
        mov dl,table[bx+1]  
        mov ah,table[bx+2]  
        mov al,table[bx+3]  
exit1:           
        ret  
        TEST3 endp       
         
         
题3.34  请写一个实现数据块移动的示例程序。

答：    data segment  
        data1 db 'Hello!!!.....$'  
        ....  
        data2 db 128 dup(?)  
        data ends  
         
        ....  
        xor ax,ax  
        xor bx,bx  
  mov1:      
        mov al,data1[bx]              
        cmp al,'$'  
        jz out1  
        mov data2[bx],al  
        inc bx  
        jmp mov1  
         
 out1:  
        ......  
         
题3.35  请编一个程序求从地址F000：0000H开始的64K字节内存区域的检验和，并转换为  
        十六进制的数的ASCII码串。  
   
答：    ;F000:0000H 字检验和  
        ;Output: BX=字检验和  
        TEST5 proc  
         
        mov ax,0F000H  
        mov es,ax  
         
        mov cx,0ffffh  
        xor si,si  
        xor bx,bx  
  ADD0:  
        add bx,es:[si]  
        inc si  
        inc si  
        loop add0  
            
        ret  
        TEST5 endp  
         
         
        table1  db '0','1','2','3','4','5','6','7','8','9'  
                db 'A','B','C','D','E','F'

        ;Input bx=字检验和  
        ;Output dx:ax=字检验和ASCII码  
        TEST4 proc  
        push cx  
         
        mov cx,bx  
        push cx  
        mov cl,12  
        shr bx,cl  
        pop cx  
        mov dh,table1[bx]  
        mov bx,cx  
        and bx,0F00h  
        push cx  
        mov cl,8  
        shr bx,cl  
        pop cx  
        mov dl,table1[bx]  
        mov bx,cx  
        and bx,00f0h  
        push cx  
        mov cl,4  
        shr bx,cl  
        pop cx  
        mov ah,table1[bx]  
        mov bx,cx  
        and bx,000fh  
        mov al,table1[bx]  
         
        mov bx,cx  
        pop cx  
        ret  
        TEST4 endp  
         
题3.36  设已在地址F000：0000H开始的内存区域安排了100个字节的无符号8位二进制数。  
        请编写一个程序求它们的和，并转换为对应十进制数的ASCII码串。  
         
答：    ;从 F000:0000H开始100个byte无符号数相加  
        ;output BX=Sum  
        TEST6 proc  
        push cx  
        push ax  
        push si  
        mov ax,0f000h  
        mov es,ax  
        xor bx,bx  
        xor si,si  
        xor ax,ax  
        mov cx,100  
         
 ADD2:  
        mov al,es:[si]           
        add bx,ax  
        inc si  
        loop ADD2  
         
        pop si  
        pop ax  
        pop cx  
        ret  
        TEST6 endp  
         
        ... ...  
        Dec\_ASC db '     $'     ;在数据区  
        ... ...  
         
        ;Name:Convert1  
        ;function: Hex convert to Dec  
        ;Input: BX=a word of Hex  
        ;Output: DS:Dec\_ASC  
        Convert1 proc  
        push ax  
        push cx  
        push dx  
        mov ax,bx  
        xor dx,dx  
        mov cx,2710h            ;2710H=10000  
        div cx  
        add ax,30h  
        mov dec\_asc[0],al       ;[0]=万位  
        mov ax,dx  
        xor dx,dx  
        mov cx,3E8h             ;3E8H=1000  
        div cx  
        add ax,30h  
        mov dec\_asc[1],al       ;[1]=千位  
        mov ax,dx  
        mov cl,64h              ;64H=100  
        div cl  
        add al,30h  
        mov dec\_asc[2],al       ;[2]=百位  
        mov al,ah  
        mov ah,0  
        mov cl,0ah              ;0A=10  
        div cl  
        add ax,3030h  
        mov dec\_asc[3],al       ;[3]=十位  
        mov dec\_asc[4],ah       ;[4]=个位  
        pop dx  
        pop cx  
        pop ax  
        ret  
        Convert1 endp

题3.37  设已在地址F000：0000H开始的内存区域安排了1024个16位有符号数。请编写一个程序  
        统计其中的正数、负数和零的个数，并分别转换为对应的十进制数的ASCII码串。  
         
答：    ;从 F000:0000H开始1024个Word有符号数统计  
        ;output Di=0的个数  
        ;       Bx=正数的个数  
        ;       DX＝负数的个数  
        TEST7 proc  
        push cx  
        push ax  
        push si  
        mov ax,0f000h  
        mov es,ax  
        xor bx,bx  
        xor si,si  
        xor ax,ax  
        xor di,di  
        xor dx,dx  
        mov cx,1024  
         
 Next1:  
        mov ax,es:[si]           
        cmp ax,0  
        jnz check\_P  
        inc di  
        jmp next2  
check\_p:  
        shl ax,1  
        jnc ADD\_P  
        inc dx  
        jmp next2  
ADD\_P:  
        inc bx  
next2:  
        inc si  
        inc si    
        loop Next1  
         
        pop si  
        pop ax  
        pop cx  
        ret  
        TEST7 endp  
         
        分别call convert1, 并保存到不同的地方即可；  
         
题3.38  设从地址F000：0000H开始的内存区域是缓冲区，存放了一组单字节的正数或负数，  
        以0结尾。请编写一个程序确定其中最大的正数和最小的负数。  
         
答：    ;从 F000:0000H开始以0结尾的单字节正数负数统计  
        ;output bh=最大的正数  
        ;       bl=最小的负数     
     TEST8 proc  
     xor bx,bx  
     xor si,si  
     mov ax,0F000h  
     mov es,ax  
next9:      
     mov al,es:[si]  
     cmp al,0  
     jz exit9  
     test al,80h  
     jnz Neg\_1  
     cmp al,bh  
     jb next7  
     xchg al,bh  
next7:  
     inc si  
     jmp next9      
      
Neg\_1:  
     cmp al,bl  
     jg next8      
     xchg al,bl  
next8:  
     inc si  
     jmp next9      
exit9:      
     ret  
     TEST8 endp

题3.39  设从地址F000：0000H开始的1K字节内存区域是缓冲区。请写一个可收集该区域内  
        所有子串“OK”开始地址的程序

答：    ;从 F000:0000H开始1K字节内存区域，统计子串“OK”开始地址  
        ;output  开始地址＝ADDRESS   
     TEST9 proc  
     xor bx,bx  
     xor si,si  
     mov ax,0F000h  
     mov es,ax  
     mov cx,1024  
next5:  
     mov ax,es:[si]  
     cmp ax,'OK'  
     jnz next6  
     mov ADDRESS[BX],si  
     inc bx  
     inc bx  
      
next6:  
     inc si  
     inc si  
     loop next5        
     ret  
     TEST8 endp

题3.40  请优化3.6.2节例7所示排序程序。

答：自己优化下；