华中科技大学计算机科学与技术学院2020~2021 第二学期
＂汇编语言程序设计＂考试试卷（模拟卷）

考试方式 闭卷 考试日期 2021-**-** 考试时长 150 分钟

专业班级 校交 1902 班 学 号 U201913821 姓 名 朱旭鹏

题号	一	二	三	四	五	六		总分	核对人
题分	20	20	20	10	10	20		100	
得分									

得分	评卷人

一、填空题（共 20 分，每空 1 分）

1、假设 32 位数据段中定义有如下变量：

X DW 1122H，3344H

PX DD X

注：寻址方式，偏移地址计算，基本工作原理与概念（含实方式），中断的基本处理过程，输入输出，窗口程序的框架及其特点

现需要将 X 中的第 2 个字单元中的内容（即 3344H）送入 AX 中。请写出使用以下 5 种不同的方式访问该单元的指令语句或语句组合（一条横线上只能写一条指令语句）：

①直接寻址方式：___mov ax,x+2___。

②寄存器间接寻址方式：___mov ebx,offset x+2___、___mov ax,[ebx]___。

③ 在“LEA EBX,X”之后执行“MOV AX,[EBX+2]”，则第 2 条语句的源操作数的寻址方式是__变址寻址__，若（EBX）=1000H，则该源操作数的偏移地址为__1002H__。

④基址加变址寻址方式：___mov ebx,offset x___、___mov esi,2___、___mov ax,[ebx+esi]___。

⑤通过变量 PX 间接访问：___mov ebx,px___、___mov ax,[ebx+2]___。

2、 实方式下，CPU 在执行完一条指令后，会检查是否有中断发生。若检测到了中断信号（设为 m 号中断），并且要响应该中断时，CPU 会将标志寄存器中的内容压栈，然后，将_CS___压栈，再将_IP__压栈，最后，从内存的最低端（段地址为 0）、偏移地址为__m*4__处取一个字单元内容送入 IP 中；取该单元的下一个字单元的内容送给_CS_。在此之后，CPU 根据 CS 和 IP 取指令，就可以得到中断处理程序中第一条待执行的指令。在中断处理程序执行结束时，会有 IRET 指令，该指令完成的操作是_从堆栈取出 IP_、_从堆栈取出 CS_、弹出一个字送给标志寄存器。

3、Windows 一般由_主程序_、__窗口主程序_、__窗口消息处理程序__和用户程序组成。在 VS2019 中动态调试窗口应用程序的用户程序时，常通过在用户程序中设置__断点_的方法

解答内容不得超过装订线

进入到相关代码的调试状态。

得分	评卷人

二、问答题（共 20 分）

设一个 32 位段程序中有如下程序片段:

```
.DATA
  STR1   DB    '158'
  LEN = $ - STR1
  STR2   DB    LEN   DUP(0)
  SX     DW    2019H, 3344H, 0
  BY     DB    2   DUP (6,  '7' )
  PX     DD    SX
.CODE
 MAIN   PROC   C
    ……                ;省略号代表其他代码
    SET_ADR   BY   ;在 BY 指向的缓冲区建地址表
    ……
    LEA   EAX,BY
    MOV   EAX,[EAX+0*4]
    JMP   EAX         ;转移到 T1 标号处执行
    ……
    LEA   EAX,BY
    MOV   EAX,[EAX+1*4]
    CALL   EAX      ;转移到 T2 标号处执行
    ……
T1:  MOV   EAX,EBX
    ……
T2:  CMP   EDX,ECX
    ……
 MAIN   ENDP
    ……
```

31H	STR1 004AB000H
35H	004AB001H
38H	004AB002H
00H	004AB003H
00H	004AB004H
00H	004AB005H
19H	SX 004AB006H
20H	
44H	
33H	
00H	
00H	
06H	
37H	
06H	
37H	
06H	
B0H	
4AH	
00H	

注：程序与内存之间的对应关系，偏移地址的处理，对数据段、代码段、堆栈段的操纵(修改、内容迁移、存储区重新定义)

（1）请在右表格中以字节为单位填写 DATA 段中各数据在存储器中的存放形式，并标明各变量所处的位置及偏移地址(STR1 的偏移地址为 004AB000H，对齐方式为紧凑方式)。(10 分)

（2）写出上述程序中用来建立地址表的宏指令 SET_ADR 的定义，实现将标号 T1 和 T2 的偏移地址设置到以 BY 为首址的存储区的功能。注意，对应存储区原来的内容需要先备份到堆栈中（不考虑恢复原来内容的功能）。(10 分)

```
SET_ADR macro BY
      mov esi,BY
      push dword ptr[esi]
      push dword ptr[esi+4]
      lea edi,T1
      mov [esi],edi
      lea edi,T2
      mov [esi+4],edi
Endm
```

```
SET_ADR MACRO V1
        PUSH DWORD PTR V1
        PUSH DWORD PTR V1+4
        MOV V1[0*4], OFFSET T1
        MOV V1[1*4], OFFSET T2
        ENDM
```

得分	评卷人

三、程序填空与改错题（共 20 分，每空 1 分）

1. 下面的子程序 STRCMP 的功能类似于 C 函数 STRCMP（STR1, STR2）的功能，即比较两个以 0 作为结束符的字符串 STR1 与 STR2 的大小关系。若 STR1>STR2，返回 EAX 为 1；若 STR1=STR2，返回 EAX 为 0；若 STR1<STR2，返回 EAX 为-1。请补充完善有关语句（每处 1 分，共 10 分）。

   在数据段中定义了 STR1 和 STR2，其中，以 STR1 为首地址的串中内容为 hello，其定义语句为：

```
STR1    DB    _'hello',0_
```

在主程序中，给出对子程序的调用语句：

```
        PUSH    OFFSET    STR2
        PUSH    OFFSET    STR1
         __call STRCMP_
        ……
```

```
;  入口参数，两个字符串的首地址 ;  出口参数，EAX 为 1，第 1 个串>第 2 个串 .....
STRCMP    PROC    C
          PUSH    EBP
          __MOV EBP,ESP__
          PUSH    ESI
          PUSH    EDI
          MOV     ESI, ___[EBP+08H]_____     ;  (ESI)  为后入栈的串的首地址
          MOV     EDI, __[EBP+0CH]_____
    LP: MOV     AL, [ESI]
          CMP     AL, [EDI]
          _JA_    BIGGER
          _JNZ_     SMALLER
          CMP     AL, 0
          JZ      STR_EQU
          INC     ESI
          INC     EDI()
          JMP     LP
BIGGER:
          _MOV EAX,1_____________
          JMP     EXIT
SMALLER:
          MOV     EAX, -1
```

```
        JMP     EXIT
STR_EQU:
        MOV     EAX, 0
EXIT:
        POP   EDI
        POP   ESI
        POP   EBP
        _RET_ (stdcall 才需要写 ret 8)
STRCMP   ENDP
```

2. 下列程序的功能是：用户输入一个无符号的数字字符串，然后将该串转换为一个字类型的数据（不考虑数值溢出）。请将程序中的语法错误和逻辑错误圈出来（圈出具体的错误位置）并在其右侧写出正确的形式（请重点关注带*的行，每改正一行中的错误得 1 分，共 10 分）。

注：以编程中常见的语法错误为主，逻辑及语义层面的错误是很少的。

```
.686P
.model flat, stdcall
   ExitProcess proto stdcall :dword
   includelib   kernel32.lib
   printf         proto   c :ptr sbyte, :vararg
    scanf          proto   :ptr sbyte, :vararg     ; *scanf          proto   C :ptr sbyte, :vararg
    includelib     libcmt.lib
    includelib     legacy_stdio_definitions.lib
.data
     buf          db     10   dup(0)
     outputFmt   db   "%d",0ah, 0dh     ; *outputFmt     db   "%d",0ah, 0dh, 0
     inputFmt     db   "%s", 0
.stack   200
.code
main   proc
        invoke   scanf, offset   inputFmt,   buf     ; *buf 后面加上一个 offset
        MOV   EAX, 0
        MOV   BX, 10
        MOV   ESI, offset BUF     ; *   (ESI) 是访问元素的下标,改为 0 即可
LP:
       MOV   CX, BUF[ESI]        ; *   MOV 改为 MOVSX
       CMP    CL,   0
       JZ      EXIT
       IMUL   BX                      ; *   MUL BX
       SUB    CL, '0'
```

```
        ADD    AX, CL                ; *   这里 AX 与 CL 不匹配,改成 CX 即可
        INC    [ESI]                 ; *   INC ESI 即可
        JNE    LP                    ; *   JMP LP
    EXIT:
      invoke printf,    outputFmt,    eax       ; *outputFmt 前面加上 offset
      invoke ExitProcess, 0
      main endp
      end
```

得分	评卷人

四、程序阅读理解（10 分）

阅读下面的程序，回答问题。

```
.686P
.model    flat, c
  ExitProcess    proto    stdcall :dword
  printf         proto    c :ptr sbyte, :vararg
  includelib     kernel32.lib
  includelib     libcmt.lib
  includelib     legacy_stdio_definitions.lib
.DATA
BUF         DB    'I Like Assembly Language'
LEN         EQU    $ - BUF
COUNT       DB    26 DUP('a',30H),0
outFmt      DB    "%s", 0
.STACK    1024
.CODE
main    proc
        LEA      EDI, BUF
        MOV      ECX, LEN          ; ---------------------------------- ①
NEXT:   MOV      BL, [EDI]         ; ---------------------------------- ②
        CMP      BL, 'a'
        JB       L1
        CMP      BL, 'z'           ; z 是小写字母
        JA       L1
        SUB      BL, 'a'
        MOVZX    EBX, BL
        INC      COUNT[EBX*2+1]
L1:     INC      EDI               ; ---------------------------------- ③
        DEC      ECX
        JNZ      NEXT
        PUSH     OFFSET COUNT
```

```
        PUSH    OFFSET outFmt
        CALL    STROUT
L2:     invoke ExitProcess, 0
main endp
STROUT    PROC
        MOV    ESI,[ESP+4]
        MOV    EBX,[ESP+8]
        MOV    ECX,25
 NEXT2: ADD    [EBX+ECX*2],ECX
          LOOP    NEXT2
        invoke    printf,    ESI,    EBX
        RET    8
STROUT    ENDP
        END
```

(1) 上述程序的功能是什么？（3 分）

计算 BUF 缓冲区中每个小写字母的个数，将其以字符的形式存入 COUNT 缓冲区对应字母的后面，将 COUNT 缓冲区对应字母与其个数显示出来。

(2) 如果将①处的语句，写成了 “MOV ECX，0”，程序执行结果会怎样？（1 分）

程序会执行 1 0000 0000H 次

(3) 若将语句②处的标号 NEXT 上移一行，误写到语句①处，则程序执行结果会怎样？（2 分）

程序会陷入死循环

(4) 若漏写了语句③，程序功能会发生什么变化？（2 分）

如果 BUF 缓冲区中第一个字符是小写字母，则会将其个数计数为 LEN，并且将 LEN 对应数字的字符 ASCII 码值存储到 COUNT 缓冲区对应小写字母的后面，否则不做处理，最后显示所有小写字母及其个数。

(5) 该程序执行过程中的局部反汇编信息如下（指令前面的数字是指令偏移地址），以字为单位画出刚进入子程序 STROUT 的堆栈示意图（图中给出传递过来的参数、返回地址的值，标出堆栈指针的位置）。（2 分）

```
00708268     push     offset COUNT (0778018h)
0070826D     push       offset outFmt (077804Dh)
00708272     call      STROUT (070827Eh)
00708277     push     0
00708279     call     _ExitProcess@4 (07082ACh)
```

8277H
0070H
804DH
0077H
8018H
0077H

得分	评卷人

五、分析与优化程序（共 10 分）

如下的 C 语言程序段（32 位段）实现了找一个整型数组(int a[5];)中的最小数并放入 x 中的功能，其编译后调试版本的汇编语言代码如下(注：斜体部分为 C 语句，x, i 均是 int 类型的变量)。(10 分)

```
        x = a[0];
009213F1    mov             eax,4
009213F6    imul            ecx,eax,0
009213F9    mov             edx,dword ptr a[ecx]
009213FD    mov             dword ptr [x],edx
        for (i = 1; i <= 4; i++){
00921400    mov             dword ptr [i],1
00921407    jmp             wmain+62h (0921412h)
00921409    mov             eax,dword ptr [i]
0092140C    add             eax,1
0092140F    mov             dword ptr [i],eax
00921412    cmp             dword ptr [i],4
00921416    jg              wmain+80h (0921430h)
            if (x > a[i])
00921418    mov             eax,dword ptr [i]
0092141B    mov             ecx,dword ptr [x]
0092141E    cmp             ecx,dword ptr a[eax*4]
00921422    jle             wmain+7Eh (092142Eh)       ; 机器码  7E 0A
                x = a[i];
00921424    mov             eax,dword ptr [i]
00921427    mov             ecx,dword ptr a[eax*4]
0092142B    mov             dword ptr [x],ecx
        }
0092142E    jmp             wmain+59h (0921409h)
00921430    ……
```

(1) 指出该段程序执行效率不高的原因 （2 分）。

没有将变量与寄存器绑定，每当需要修改变量的值时，都先将其值送给一个寄存器，然后修改寄存器的值，最后又送回到变量中，这样使得代码很长，执行效率低。并且存在很多

(2) 改编相应的汇编程序，以提高程序的执行效率。要求写出变量与寄存器对应关系，尽可能与调试版本一致。(6 分)

```
mov ecx,  a[0]
mov eax, 1
Next:
cmp eax,4
jg     EXIT
cmp ecx,a[eax*4]
jle   L1
mov ecx,a[eax*4]
L1:
inc eax
jmp Next
EXIT:
.....
ecx：对应变量 x；eax:对应变量 i;
```

(3) 解释 jle 指令语句的机器码（7EH, 0AH）中 0AH 代表的含义（2 分）

0AH 指的是目的地址与跳转指令的下一条指令的地址之差，即目的地址相对于跳转指令的下一条指令的偏移

得分	评卷人

六、程序设计（20 分）

设以 BUFA、BUFB 为首地址的存储区中，分别存储有 N 个和 M 个非零的字数据。现编一个完整的 32 位段程序，将在两个存储区中都出现的数据拷贝到 BUFC 中（BUFA 中的 N 个数可能相互相同，要避免重复拷贝，N 和 M 自定）。要求：

(1) 简要描述设计思想，给出寄存器分配方案。

(2) 用子程序 FIND 判断一个数是否在指定的某个存储区中出现，描述其入口参数、出口参数。

(3) 画出主程序和子程序 FIND 的流程图。主程序与子程序分别属于两个不同的模块。

(4) 程序完整（包括堆栈段、数据段、代码段定义等，库函数相关信息可参考第四题），至少给出 4 条必要的注释。