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（A 卷）

考试方式 闭卷 **考试日期** 2021-05-30 **考试时长** 150 分钟

专业班级 _____ **学　　号** _____ **姓　　名** _____

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、 操作系统在加载一个 32 位段的程序去运行时，会把将要执行的第一条机器指令的段首址信息送到 CS 中，_偏移地址_送入 EIP 中。CPU 根据 CS: EIP 取指令后，对该指令进行解析，此时 （EIP）+_该指令机器码的字节数_ →EIP，使其指向紧接其后的下一条指令。之后执行已经取回来的指令：若该指令不涉及到转移，则该指令执行完后就会直接取紧接其后的下一条指令，程序顺序执行；如果该指令是直接调用子程序 FUNC 的指令，如“_CALL _FUNC”，CPU 就会将(EIP)_ 压栈 _，然后_子程序的偏移地址_→EIP;若是 RET 指令，CPU 会 _POP 4 个字节_→EIP; 若是无条件的间接转移指令，如“ _JMP__ DWORD PTR [EBX]”，则 CPU 会 _从内存单元[EBX]取 4 个字节__→EIP ；若是传输指令“MOV AL, 5[EBX+ESI]”，则 CPU 在取源操作数时，先按照__基址加变址__寻址方式的要求，计算源操作数的偏移地址，计算的表达式为 EA=__（EBX）+（ESI）+ 5__。

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。在响应 n 号中断请求时，CPU 会将（FLAGS）压栈，然后将(CS)压栈，最后将_ IP（下一条指令的偏移地址）_压栈，并从内存的最低端（段地址为 0）、偏移地址为_4*n __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。中断处理程序中常涉及到 IO 操作，如“IN _AL__,71H”就是将端口 71H 中的一个字节输入到 CPU 的寄存器中。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弹出一个字送给 IP,、再_弹出一个字送给

解答内容不得超过装订线

CS _、弹出一个字送给标志寄存器。

3、Windows 窗口应用程序运行时，由窗口应用程序的_主程序_直接调用窗口主程序，窗口主程序做好相关初始化工作后就进入_消息__循环，窗口主程序中并不存在直接调用 __窗口__消息处理程序的指令语句。

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
buf1    db    '78'
len   = $ - buf1
buf2    db    len   dup(0)
x       dw    2021h, 1234h
y       db    2   dup (5, '6')
px      dd    x
.code
main   proc   c
   ……                    ;省略号代表其他代码
   TO_FUNC   VAL    ;根据 VAL 的大小调用子程序
   ……
main   endp
func1   proc   c
   mov   eax, 10
   ……
func2   proc   c
   sub   edx, ebx
   ……
```

37H	buf1 00456000H + 00 H
38H	+01H
00H	+02H, buf2
00H	+03H
21H	+04H, x
20H	+05H
34H	+06H
12H	+07H
05H	+08H, y
36H	+09H
05H	+0AH
36H	+0BH
04H	+0CH, px
60H	+0DH
45H	+0EH
00H	+0FH

（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(buf1 的偏移地址为 00456000H，对齐方式为紧凑方式)。（10 分）

（2）写出上述程序中的宏指令 TO_FUNC 的定义。该宏指令的功能是：根据参数 VAL 的值决定调用哪个子程序，即当（VAL）=0 时，调用 func1；当（VAL）=1 时，调用 func2。func1 和 func2 均为没有入口参数的子程序；VAL 可能是一个常量，也可能是一个字节变量。要求 call 语句在宏体中只出现一次且指令语句只能写成“call px”。(不用考虑局部标号和寄存器保护问题，但 ACM 班需要考虑在将要执行 CALL 语句时，该宏没有影响任何通用寄存器的内容)（10 分）

```
TO_FUNC   MICRO   VAL
          LOCAL   L1
          MOV     AL, VAL
          MOV     px, OFFSET FUNC1
          CMP     AL, 0
          JZ      L1
          MOV     px, OFFSET FUNC2
L1:       CALL    px
          ENDM
```

得分	评卷人

三、分析完善题（程序填空与改错，共 20 分，每处 1 分）

1. 下面的子程序 find_max 的功能是从一个整型数组中（有符号数）找出最大的数， eax 中存放最大值（每空 1 分，共 10 分）。

在数据段中有如下定义 ：

```
buf   dd   10,-20,0,30,-25
buflen = 5
```

在主程序中，子程序的调用语句：

```
        push    __buflen___
        push    ___offset buf__
        call    find_max
        add     esp,8
        ……
find_max  proc
        push    ebp
        mov     ebp, esp
        push    ebx
        push    ecx_____
        mov     ebx, [ebp+8]          ; (ebx) 数组中第 0 个元素的地址
        mov     ecx, [ebp+_0ch_]      ; (ecx) 数组元素个数
        mov     eax, [ebx]            ; (eax) 当前找到的最大数
find_loop:
        dec     ecx
        jz      exit_____
        add     ebx,_4__
        cmp     eax,  [ebx]
        jge /jg / jnle__ next
        mov     eax, [ebx]
next: _jmp find_loop____
exit:pop  ecx
        __pop ebx_____
        pop   ebp
```

```
        ret
find_max endp
```

2. 下列程序的功能是：用户输入一个数，然后将该数转换为一个二进制字符串并输出。请将程序中的语法错误和逻辑错误圈出来，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）。

```
.686p
.model flat, c
  ExitProcess proto stdcall :dword
  includelib  kernel32.lib
  includelib  libcmt.lib
  includelib   legacy_stdio_definitions.lib
  printf      proto    :ptr sbyte, :vararg
  scanf       proto    :ptr sbyte, :vararg
.data
  outputFmt  db  0ah,0dh, "%s", 0
  inputFmt   db  "%d"        ; * db "%d", 0
  x          dd  0
  buf        db  32 dup(0),'B',0
.stack  200
.code
main proc
    push   x            ;  * push offset x
    push   inputFmt     ;  * push offset inputFmt
    call   scanf        ;  输入一个数，c 语言的调用形式是 scanf(“%d”,&x);
    mov    eax, x
    mov    ecx, 16      :  * mov ecx,32
    mov    esi,   buf   ;  * lea esi, buf / mov esi, offset buf
lp: shl    eax, 1       ; (eax) 左移一个二进制位
    mov    dl, 0
    rcr    dl, 1        ; * 将标志位 cf 移到 dl的最低位  rcl dl, 1
    add    dl, 30       ; * 由数码变成对应的字符  add dl, 30h
    mov    esi, dl      ; * 将对应的字符存到缓冲区中  mov [esi],dl
    jnc    esi          ; * inc esi
    dec    ecx
    jmp    lp           ; * jnz lp
    invoke  printf, offset outputFmt, offset buf
    invoke  ExitProcess, 0
main  endp
end
```

得分	评卷人

四、分析思考题（10 分）

阅读下面的程序，回答问题。

```
.686p
.model flat, c
  ExitProcess proto stdcall :dword
  includelib  kernel32.lib
  includelib  libcmt.lib
  includelib  legacy_stdio_definitions.lib
  printf      proto  c :ptr sbyte, :vararg
.data
  msg1       db  "not "
  msg2       db    "same", 0dh,0ah,0
  addr_table dd  msg1, msg2
  string1    db  'hello',0
  string2    db  'very good',0
.stack 200
.code
main  proc  c
      push   offset string2
      push   offset string1
      call   strcmp
      add    esp, 8
      invoke printf,addr_table[eax*4]
      invoke ExitProcess, 0
main  endp
strcmp  proc
      push  ebp
      mov   ebp, esp
      mov   edi, [ebp+8]
      mov   esi, [ebp+12]
strcmp_l1:
      mov   dl,  [edi]
      cmp   dl,  [esi]
      jne   strcmp_different
      cmp   dl, 0
      jz    strcmp_same
      inc   esi
      inc   edi
      jmp   strcmp_l1
strcmp_different:
      mov   eax, 0
```

```
        jmp    strcmp_exit      ; ·····.①
strcmp_same:
        mov    eax, 1
strcmp_exit:
        pop  ebp          ; ····· .②
        ret
strcmp endp
end
```

(1) 上述程序运行后，屏幕上显示的是什么？（2 分）

not same

(2) 子程序 strcmp 的功能是什么？它的入口参数和出口参数分别是什么？（3 分）

比较 2 个以 0 结尾的字符串是否相同，若相同则返回 1（eax）=1，否则返回 0（eax=0）。入口参数是 2 个字符串的首地址（偏移地址），出口参数是 eax。

(3) 若漏写了语句①，子程序功能会发生什么变化？程序运行后，显示的结果是什么？（3 分）

漏写语句①，子程序总是返回（eax）=1，主程序显示结果总是 same

(4) 若漏写了语句②，程序运行会出什么问题？（2 分）

子程序不能正确返回到主程序，可能程序崩溃。

得分	评卷人

五、分析优化题（共 10 分）

如下的 C 语言程序段（32 位段）实现了统计一个整型数组(int buf[5];)中的正数个数并放入 count 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)。(10 分)

```
        int     i;
        int    count = 0;
0011180B    mov          dword ptr [ebp-30h],0
      for (i = 0;i <= 4;i++)
00111812    mov          dword ptr [ebp-24h],0
00111819    jmp          00111824
0011181B    mov          eax,dword ptr [ebp-24h]
0011181E    add          eax,1
00111821    mov          dword ptr [ebp-24h],eax
00111824    cmp          dword ptr [ebp-24h],4
00111828    jg           0011183F
          if (buf[i] > 0)
0011182A    mov          eax,dword ptr [ebp-24h]
0011182D    cmp          dword ptr [ebp+eax*4-18h],0
00111832    jle          0011183D
              count++;
```

```
00111834   mov          eax,dword ptr [ebp-30h]
00111837   add          eax,1
0011183A   mov          dword ptr [ebp-30h],eax
0011183D   jmp          0011181B
0011183F   …….
```

(1) 指出该段程序执行效率不高的原因 （2 分）。

主要是下面 2 条 C 语句的机器指令耗时：

```
            count++;
00111834   mov          eax,dword ptr [ebp-30h]
00111837   add          eax,1
0011183A   mov          dword ptr [ebp-30h],eax

i++;
0011181B   mov          eax,dword ptr [ebp-24h]
0011181E   add          eax,1
00111821   mov          dword ptr [ebp-24h],eax
```

(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)

```
            count++;
00111834   mov          eax,dword ptr [ebp-30h]
00111837   add          eax,1
0011183A   mov          dword ptr [ebp-30h],eax
修改为：add dword ptr [ebp-30h], 1

i++;
0011181B   mov          eax,dword ptr [ebp-24h]
0011181E   add          eax,1
00111821   mov          dword ptr [ebp-24h],eax
修改为：add dword ptr [ebp-24h], 1
```

(3) “00111832 jle 0011183D”处指令的机器码为 7EH 09H，解释 09H 代表的含义（2 分,卓越班 1 分）

09H 是需要转移的目标指令的偏移地址相对于当前指令的下一条指令的偏移地址的相对偏移量。

(4) 请用一条语句实现：将(eax)*4+10 的结果送到(ebx)，不用考虑溢出。(1 分，此题仅卓越班做)

lea ebx, [4*eax+10]

得分	评卷人

六、设计题（20 分）

设以 STR1、STR2 为首地址的存储区中，分别存放了以 0 为结束符的字符串，变量 STR0 中存放了一个字符。现编写一个完整的 32 位段程序，统计 STR0 中存放的字符在两个存储区中累计出现的次数，并存放到双字变量 SUM 中。要求：

带格式的：缩进：首行缩进： 2 字符

带格式的：缩进：首行缩进： 2 字符

(1) 简要描述设计思想，给出寄存器分配方案。

(2) 画出子程序 COUNT 的流程图。

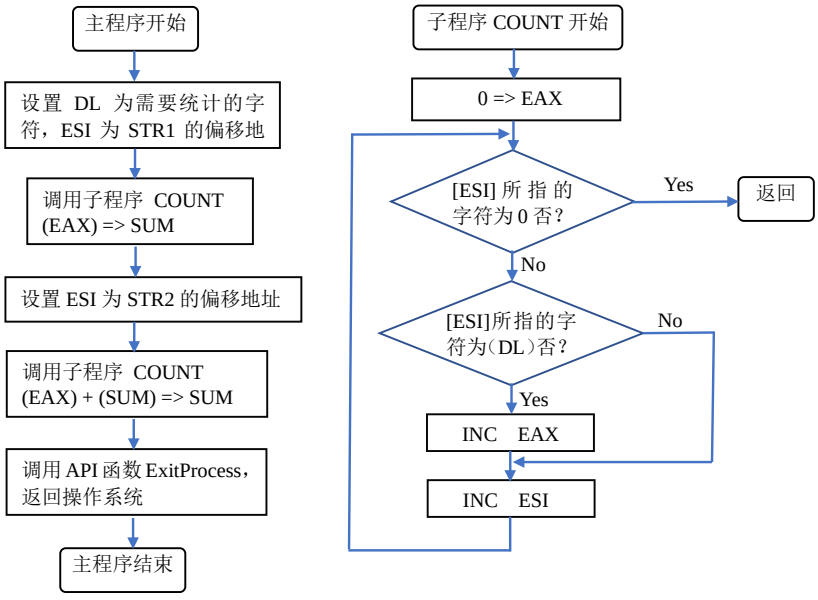
(3) 用子程序 COUNT 统计某个字符在一个串中出现的次数，描述其入口参数、出口参数（STR1、STR2，STR0 的名字均不能出现在子程序中）。

(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。

(1) 设计思想：子程序 COUNT 统计字符 DL 在一个串中出现的次数，主程序依次使用字符串 STR1 和 STR2 的首地址调用 2 次 COUNT，得到字符 DL 出现在 STR1 和 STR2 中的次数，将这 2 个次数相加保存到 SUM 变量中。

DL = 需要统计的字符
DS:ESI = 字符串首地址
EAX = 字符 DL 在字符串中出现的次数

(2) 画出主程序和子程序的流程图。



```
.686p
.model flat, stdcall
  ExitProcess proto stdcall :dword
  includelib    kernel32.lib

.data
  str1      db    "hjjhhjjhaadbjkf", 0
  str2      db    "hjhsadgjkhdsjk", 0
  str0      db    'a'
  sum       dd    0

.stack    200

.code
start       proc
              mov       dl, str0
              mov       esi, offset str1
              call       count                ;统计字符 DL 出现在 str1 中的个数  => eax
              mov       sum, eax
              mov       esi, offset str2
              call       count                ;统计字符 DL 出现在 str2 中的个数  => eax
              add        sum,  eax            ;字符 DL 出现在 str1 中的个数  +  字符 DL 出现在 str2 中
的个数  => sum
              invoke    ExitProcess, 0    ;返回 Windows 操作系统
start         endp

;统计字符在字符串（以 0 结尾）中出现的次数
;入口：   DL =  需要统计的字符
;          DS:ESI =  字符串首地址
;出口：   EAX =  字符 DL 在字符串中出现的次数
;说明：   改变 ESI，不改变 DL
count      proc
              mov     eax, 0
lc1:          cmp     byte ptr [esi], 0
              jnz       lc2
              ret                                 ;遇到 0，比较结束
lc2:          cmp     byte ptr [esi], dl
              jnz       lc3
              inc       eax
lc3:          inc       esi
              jmp           lc1
count       endp

              end start
```