******华中科技大学计算机科学与技术学院2021~2022第二学期**

解答内容不得超过装订线

**“ 计算机系统基础 ”考试试卷 (A卷)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考试方式** | **闭卷** | **考试日期** | **2022-05-21** | **考试时长** | **150 分钟** |
| **专业班级** |  | **学 号** |  | **姓 名** |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **题号** | **一** | **二** | **三** | **四** |  | **总分** | **核对人** |
| **分值** | 40 | 20 | 20 | 20 |  | 100 |  |
| **得分** |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **程序运行的基本原理问答（共40分）**

对如下 C语言程序，用VS2019（Intel CPU）编译、链接、调试运行。

#include <stdio.h>

int fmax(int x, int y)

{ int u;

if (x > y) u= x;

else u = y;

return u;

}

int main(int argc, char\* argv[])

{ int a = 100; // 0x64

int b = 0x12345678;

int r = 0;

char msg[6] = "abcde"; // 'a'的ASCII是 0x61

r = fmax(a, b);

msg[17]=’1’; // '1'的ASCII是 0x31

r = \*(int \*)(msg+1);

return 0;

}

在调试时，在监视窗口中看到 变量 a 的地址（即&a）为 0x010ffe98；变量 b 的地址(即&b) 为 0x010ffe94；变量 r 的地址为 0x010ffe90， 数组 msg 的起始地址为 0x010ffe88。

1. 填写执行到“r=fmax(a,b)；”时，内存中数据存放的结果。要求以字节为单位、用16进制数的形式填空，最左边是内存窗口显示的内存地址。（10分）

0x010ffe88 \_\_61\_ \_\_62\_ \_\_63\_ \_\_64\_ \_\_65 \_\_00\_ XX XX

0x010ffe90 \_00\_\_ \_\_00\_ \_00\_\_ \_00\_\_ \_\_78\_ \_56\_\_ \_\_34\_ \_\_12\_

0x010ffe98 \_64\_\_ \_\_00\_ \_00\_\_ \_00\_\_ XX XX XX XX

1. 数据传送指令解读 （10分，每空 2分）

语句 “int a = 100;”的反汇编指令为： **m**ov dword ptr [ebp-4], 64h

根据观察到的 a 的地址， 推断执行该语句时， (ebp)= 0x \_\_10ffe9c\_\_\_\_\_\_, 该指令的机器码为 C7 45 FC 64 00 00 00，其中 FC 对应的是指令中\_\_\_-4\_\_\_\_\_\_的编码。

根据观察到的 a 与b的地址关系，推断出 “int b = 0x12345678; ” 对应的反汇编指令为：

\_\_mov dword ptr [ebp-8], 12345678h

执行 “msg[17]=’1’; ”后，变量 a中的值为 0x\_\_\_3164\_\_\_\_ 。

执行 “r = \*(int \*)(msg+1);”后， r中的值为 0x\_\_65646362\_\_\_\_\_ 。

1. 函数调用语句解读 （10分，每空2分）

对于语句“r=fmax(a,b);”对应的反汇编代码（最左边的是机器指令的地址）如下。

|  |  |
| --- | --- |
|  | 0x010ffe28 |
|  | 0x010ffe2c |
| 0x00721632 | 0x010ffe30 |
| 0x00000064 | 0x010ffe34 |
| 0x12345678 | 0x010ffe38 |
| XXXXXXXX | 0x010ffe3c |

0072162B mov eax,dword ptr [ebp-8]

0072162E push eax

0072162F mov ecx,dword ptr [ebp-4]

00721632 push ecx

00721633 call 007212B2

00721638 add esp,8

0072163B mov dword ptr [ebp-0Ch],eax

设执行“r=fmax(a,b);”之前，(esp) = 0x010ffe3c

① 在表格的适当位置填写 刚进入函数fmax内部时，堆栈中存放的相关数据。

② 刚进入函数fmax内部时，(esp)= 0x\_010ff330\_\_\_\_\_\_\_

③ 执行 “ add esp, 8 ” 之后，(esp) = 0x\_0x10ffe3c\_\_\_\_\_\_\_。

4. 函数体的指令解读 (10分，每小题2分)

函数体对应的反汇编代码有：

push ebp

mov ebp, esp ……

if (x > y) u= x;

007215C3 mov eax,dword ptr [ebp+8]

007215C6 cmp eax,dword ptr [ebp+0Ch]

007215C9 jle 007215D3

007215CB mov eax,dword ptr [ebp+8]

007215CE mov dword ptr [ebp-4],eax

007215D1 jmp 007215D9

else u = y;

007215D3 mov eax,dword ptr [ebp+0Ch]

007215D6 mov dword ptr [ebp-4],eax

return u;

007215D9 mov eax,dword ptr [ebp-4] ……

pop ebp

ret

① 执行到 “cmp eax, dword ptr [ebp+0Ch]”指令时，(ebp)=0x\_\_010ff32c。

② 函数参数 x的地址（即&x）是 0x\_010ff334\_\_。

③ 执行 “cmp eax, dword ptr [ebp+0Ch] ” 后， CF=\_1\_\_, SF=\_\_1\_\_, ZF=\_\_0\_\_, OF= \_0\_\_\_\_\_。

④ 执行 RET 指令时，CPU 所做的工作是\_\_将栈顶双字出栈到EIP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

⑤ 函数返回结果的方法是\_\_使用EAX传递返回整数值\_\_\_。

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **程序阅读与理解 （共20分）**

**1、阅读下面的程序，回答问题 （10分）。**

.686P

.model flat, c

includelib kernel32.lib

includelib libcmt.lib

includelib legacy\_stdio\_definitions.lib

printf proto :ptr sbyte, :vararg

ExitProcess proto stdcall :dword

.data

nums sdword 1234, -1, 100,-20, -2

len = ($-nums)/4

fmt db "%d ", 0dh,0ah, 0

.code

main proc c

mov edx, 0

mov ecx, len ; ①

lea esi, nums ; ②

process:

cmp dword ptr [esi], 0

jge next ; ③

inc edx

next:

add esi, 4

dec ecx

jne process

invoke printf, offset fmt, edx

invoke ExitProcess, 0

main endp

end

1. 上述程序的功能是什么？运行后，屏幕上显示的是什么？（2分）

统计nums中，负数的个数，保存到edx中，并显示edx的值。显示3

1. 若将 ③ 处的语句改为 “jae next”,程序运行的结果是什么？(2分)

**显示0**

1. 若标号 process 写到 ②处语句前，程序运行的结果是什么？为什么？（3分）

显示0, 始终比较第一个元素

1. 若标号 ①处语句写到标号process之后，程序运行会出现什么现象？为什么？（3分）

process: mov ecx, len

cmp dword ptr [esi], 0 ……

**死循环。每次循环未修改ecx的值。**

**2. 程序填空（10分，每空 1分）**

在一个以0结束的字符串中，将所有的大写字母转换为对应的小写字母，并将转换结果、以及修改的字母个数输出。

……

.data

buf db "abcDEFgh", \_0\_\_\_

fmt db "%s %d" , 0

.code

main proc c

mov esi, \_offset buf\_ ; esi 存放待访问字符的地址

mov ecx, \_0 ; ecx 存放修改字母的个数

next:

mov al, [esi]

cmp al, 0

je exit\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cmp al, 'A'

\_jb cont\_\_\_\_\_\_\_\_\_\_

cmp al, 'Z'

\_ja cont\_\_\_\_\_\_\_\_\_\_

add al, \_\_’a’ – ‘A’

mov \_[esi]\_\_\_\_ , al

inc ecx\_\_\_\_\_\_\_\_\_\_\_

cont:

inc esi

jmp next\_\_\_\_\_\_\_\_\_\_\_

exit:

invoke printf, offset fmt, offset buf, ecx

invoke ExitProcess, 0

main endp

end

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **程序优化 （共20分）。**
2. 对C程序进行编译生成Release版程序时，编译器会做一些优化工作。请举出5个优化场景，简要说明做了什么优化，以及为什么能加快执行速度。 (10 分)

使用寄存器作为循环计数器；

使用寄存器作为数组下标，进行变址寻址；

使用寄存器传递参数；

使用带比例因子的变址寻址操作数，获取EA，代替算术运算；

使用宽字节类型的寄存器（MMX，SIMD单指令多数据流），进行数组元素的操作，减少循环次数；

使用顺序访问数组元素的方式，代替循环访问数组元素；

2、如下的 C语言程序段（32位段）实现了“统计一个字符串 buf 中小写字母的个数，并将其放入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；在反汇编时，勾选显示符号名)。(10分)

int count = 0;

00A616A0 mov dword ptr [count],0

int i;

for (i = 0; buf[i] != 0;i++)

00A616A7 mov dword ptr [i],0

00A616AE jmp fcount+59h (0A616B9h)

00A616B0 mov eax,dword ptr [i]

00A616B3 add eax,1

00A616B6 mov dword ptr [i],eax

00A616B9 mov eax,dword ptr [i]

00A616BC movsx ecx,byte ptr buf[eax]

00A616C1 test ecx,ecx

00A616C3 je fcount+8Ah (0A616EAh)

if (buf[i] >= 'a' && buf[i] <= 'z')

00A616C5 mov eax,dword ptr [i]

00A616C8 movsx ecx,byte ptr buf[eax]

00A616CD cmp ecx,61h

**00A616D0 jl fcount+88h (0A616E8h)**

00A616D2 mov eax,dword ptr [i]

00A616D5 movsx ecx,byte ptr buf[eax]

00A616DA cmp ecx,7Ah

00A616DD jg fcount+88h (0A616E8h)

count++;

00A616DF mov eax,dword ptr [count]

00A616E2 add eax,1

00A616E5 mov dword ptr [count],eax

00A616E8 jmp fcount+50h (0A616B0h)

(1) 指出该段程序执行效率不高的原因 （2分）。

使用局部变量作为循环计数器；使用局部变量作为数组下标，进行变址寻址；

在条件表达式中，对同一个变量buf[i]，进行了两次赋值

(2) “00A616D0 jl fcount+88h (0A616E8h)”处指令的机器码为7CH 16H，解释16H代表的含义（2分）

为跳转目的地址到该指令的距离：0A616E8h – (0A616D0h + 2)

(3) 改编相应的汇编语言程序，以提高程序的执行效率。(6分)

要求写出采取的优化方法，在优化程序段中，可以用标号来代替转移指令的目的地址。

count: edx

i: ecx

buf[i]: al

mov edx, 0

mov ecx, 0

next: mov al, buf[ecx]

cmp al, 0

je exit

cmp al, ‘a’

jb cont

cmp al, ‘z’

ja cont

inc edx

cont: jmp next

exit: mov count, edx

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **设计题（20分）**

在BUF1为首址的字类型存储区中，存放了M个互不相同的数据；在BUF2为首址的字类型存储区中存放了 N个数据。用汇编语言编写完整的程序，统计在两个存储区中同时出现的数据个数，存放到字类型变量COUNT中，最后在屏幕上输出COUNT的值。

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 判断某个字类型数据是否在某存储区中出现；描述其入口参数、出口参数。

(3) 画出程序的流程图。

(4) 程序完整（包括堆栈段、数据段、代码段定义等），至少给出4条必要的注释。

(5) BUF1、BUF2中的数据在定义时直接给出初值，无需输入；M、N自己给定；子程序FIND中不允许使用全局变量。

**(1)**

**esi：BUF1寻址、BUF2寻址**

**ecx：外循环、内循环**

(2) FIND入口参数：

ax: 待查找字

ebx:存储区首址

edx: 存储区长度

出口参数：

ax: 1表示找到；0表示未找到