******华中科技大学2024~2025第一学期**

**“ 计算机系统基础 ”考试试卷 (A卷)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考试方式** | **闭卷** | **考试时间** | **2024-11-26晚上** | **考试时长** | **150 分钟** |
| **院（系）** |  | | **专业班级** |  | |
| **学 号** |  | | **姓 名** |  | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **题号** | **一** | **二** | **三** | **四** | **五** |  |  | **总分** | **总分人** | **核对人** |
| **分值** | 20 | 20 | 20 | 20 | 20 |  |  | 100 |  |  |
| **得分** |  |  |  |  |  |  |  |  |  |  |

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

**一、（20分）数据的表示和访问。**

1. 有一个C语言程序如下。（10分）

typedef struct{

char id[10];

int age;

float height;

} person\_info;

void main() {

person\_info p1;

strcpy(p1.id, “12350”); #字符0的ASCII为0x30

p1.age = 15;

p1.height = 1.75f;

unsigned short x1 = \*(unsigned short\*)(p1.id + 1);

int y1 = -100; ------①

}

将上述程序编译链接生成可执行程序，用调试工具运行完语句①后，观察变量地址如下：p1的地址是0xffffd038，p1.age的地址是0xffffd044，p1.height的地址是0xffffd048，x1的地址是0xffffd032，y1的地址是0xffffd034。以字节形式观察这些变量在内存中的存放形式。以16进制的形式填写程序用到的内存单元内容（无关的字节中的内容填写XX，person\_info的大小为20字节）。

0xffffd032: \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_

0xffffd03a: \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_

0xffffd042: \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_

0xffffd04a: \_\_\_\_\_\_ \_\_\_\_\_\_ 。

提示：浮点数float的编码规则：float用32位表示，最高位（第31位）为符号位，23-30位为用移码表示的指数（移码值为0x7F），0-22位为尾数位。

2. 调试一个C程序时，看到源程序语句与对应的反汇编语句如下。根据观察到的信息填空。（10分，每空1分）

**int arr1[5];**

**short arr2[3][2];**

**void main( ) {**

**int i, j ;**

**i = 2; //** 执行该语句时，ebp的值为0xffffd068，eax的值为0x56558fdc

0x5655619d <+16>: c7 45 ec 02 00 00 00 movl $0x2, -0x14(%ebp)

变量i的地址是 0x\_\_\_\_\_\_\_\_\_\_\_\_，该变量存放在\_\_\_\_\_\_\_\_\_段中。

**arr1[i] = 34;**

0x565561a4 <+23>: 8b 55 ec mov -0x14(%ebp), %edx

0x565561a7 <+26>: c7 84 90 30 00 00 00 22 00 00 00 movl $0x22, 0x30(%eax,%edx,4)

此时，%edx存放的是\_\_\_\_\_\_\_\_\_的值，arr1[i]的地址是 0x\_\_\_\_\_\_\_\_\_\_\_\_。

**i = 1;**

0x565561b2 <+37>: movl $0x1, -0x14(%ebp)

**j = 1;**

0x565561b9 <+44>: movl $0x1, -0x10(%ebp)

**arr2[i][j] = 27;**

0x565561c0 <+51>: 8b 55 ec mov -0x14(%ebp), %edx

0x565561c3 <+54>: 8d 0c 12 lea (%edx,%edx,1), %ecx

执行这条指令后，%ecx中的值是\_\_\_\_\_\_\_\_\_\_\_\_\_，arr2的每行有两个元素，%ecx表示的含义是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

0x565561c6 <+57>: 8b 55 f0 mov -0x10(%ebp), %edx

0x565561c9 <+60>: 01 ca add %ecx, %edx

0x565561cb <+62>: 66 c7 84 50 44 00 00 00 1b 00 movw $0x1b, 0x44(%eax,%edx,2)

arr2[i][j]的地址是 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

**int \*p1 = &arr1[2];**

0x565561d5 <+72>: 8d 90 30 00 00 00 lea \_\_\_\_\_\_\_\_(%eax), %edx

0x565561db <+78>: 89 55 f4 mov %edx, -0xc(%ebp)

**\*(p1 + 1) = 20;**

0x565561de <+81>: mov -0xc(%ebp), %edx

0x565561e1 <+84>: add \_\_\_\_\_\_\_\_, %edx

0x565561e4 <+87>: movl $0x14, (%edx)

**short \*p3 = &arr2[1][1];**

0x565561e7 <+90>: 8d 80 4a 00 00 00 lea \_\_\_\_\_\_\_\_(%eax), %eax

0x565561ed <+96>: 89 45 fc mov %eax, -0x4(%ebp)

}

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

**二、（20分）C程序的转换**

1. 阅读下面的程序段，回答问题（10分，每空1分）

void main( ) {

short s = -5;

unsigned short us = 0xfffb;

int i = s; // ①

unsigned int ui = us; // ②

if (ui < 5) // ③

s = (int)(s + s) >> 1;

else

us = (unsigned int)(us + us) >> 1; // ④

}

变量s、us、i、ui，对应的地址表达式为：-0xc(%ebp)、-0xa(%ebp)、-0x8(%ebp)、-0x4(%ebp)。

1. 执行完①处语句后，i = 0x\_\_\_\_\_\_\_\_\_\_\_\_；

该语句对s变量的处理指令为：\_\_\_\_\_\_\_\_\_\_\_\_\_ -0xc(%ebp),%eax。

1. 将②处的运算指令序列补充完整：

0x565561b0 <+35>: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ -0xa(%ebp),%eax

0x565561b4 <+39>: mov %eax,-0x4(%ebp)

执行完②处语句后 ui =\_\_\_\_\_\_\_\_\_\_\_\_;

1. 对于③处语句，补充完整这条语句的处理序列：

0x565561b7 <+42>: cmpl $0x5, -0x4(%ebp)

0x565561bb <+46>: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 0x565561cb <main+62>

③ 处if-else语句，执行分支\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

1. ④处语句对应的机器指令是：

0x565561cb <+62>: \_\_\_\_\_\_\_\_\_\_\_\_ -0xa(%ebp),%eax

0x565561cf <+66>: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0x565561d1 <+68>: \_\_\_\_\_\_\_\_\_\_ %eax

0x565561d3 <+70>: mov \_\_\_\_\_\_\_\_,-0xa(%ebp)

2. 阅读如下程序及其反汇编代码，回答问题（10分）。

int max(int a, int b) {

if (a > b) return a;

else return b;

}

void main() {

int x = 100;

int y = -10;

int m = max(x, y);

}

main函数中，int m=max(x, y) 对应的反汇编代码如下：

0x565561ca <+30>: push -0x8(%ebp) --------------- ①

0x565561cd <+33>: push -0xc(%ebp)

0x565561d0 <+36>: call 0x5655618d <max>

0x565561d5 <+41>: add $0x8,%esp ---------------- ②

0x565561d8 <+44>: mov %eax, -0x4(%ebp)

max函数的反汇编代码如下：

0x5655618d <+0>: push %ebp --------------- ③

0x5655618e <+1>: mov %esp, %ebp

0x5655619a <+13>: mov 0x8(%ebp), %eax

0x5655619d <+16>: cmp 0xc(%ebp), %eax

0x565561a0 <+19>: jle 0x565561a7 <max+26>

0x565561a2 <+21>: mov 0x8(%ebp), %eax

0x565561a5 <+24>: jmp 0x565561aa <max+29>

0x565561a7 <+26>: mov 0xc(%ebp), %eax

0x565561aa <+29>: mov %ebp, %esp

0x565561ac <+31>: pop %ebp --------------- ④

|  |  |
| --- | --- |
|  | 0xffffd048 |
|  | 0xffffd04c |
|  | 0xffffd050 |
|  | 0xffffd054 |
|  | 0xffffd058 |
|  | 0xffffd05c |
|  | 0xffffd060 |
|  | 0xffffd064 |
|  | 0xffffd068 |

0x565561ad <+32>: ret

设执行①处语句前，(esp)= 0xffffd058，(ebp)= 0xffffd068。

填写执行指令 ③ 后堆栈的示意图，包括main函数的局部变量x, y的位置和内容，无关的位置填写XX。

指令④的作用是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。若缺少指令④，子程序（能/不能）\_\_\_\_\_\_\_\_正常返回。

执行指令②前，esp=\_\_\_\_\_\_\_\_\_\_\_\_\_。指令②的作用是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

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

**三、（20分）程序的运行的基本原理**

1. C程序的部分反汇编代码如下。完成填空（10分，每空1分）

**void main( ) {**

**int x = -1;**

0x565561b9 <+28>: movl $0xffffffff,-0x14(%ebp)

**int y = -2;**

0x565561c0 <+35>: movl $0xfffffffe,-0x10(%ebp)

**int z = y - x;**

0x565561c7 <+42>: mov -0x10(%ebp),%edx

0x565561ca <+45>: sub -0x14(%ebp),%edx----------------- ①

0x565561cd <+48>: mov %edx,-0xc(%ebp)

**if (y == z)**

0x565561d0 <+51>: mov -0x10(%ebp),%edx

0x565561d3 <+54>: cmp %edx,-0xc(%ebp)

0x565561d6 <+57>: jne 0x565561ee <main+81> ----------------②

**printf("equal\n");**

0x565561d8 <+59>: sub $0xc,%esp

…………..

0x565561ec <+79>: jmp 0x56556202 <main+101> --------------- ③

**else printf("not equal\n");**

0x565561ee <+81>: ……

0x56556202<+101> ……

}

内存地址0x565561c0—0x565561c6的内存单元中，存放的是汇编语句mov $0xfffffffe, 0x10(%ebp) 的\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

1. 处的指令执行后，CF=\_\_\_\_\_, OF=\_\_\_\_\_, ZF=\_\_\_\_\_, SF=\_\_\_\_\_。

程序在运行时，寄存器EIP中存放的是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。当②处的指令预取和解码后，EIP的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。由于此时ZF = \_\_\_\_\_\_\_\_，转移条件成立，②处的指令执行后，EIP的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

1. 处的jmp指令，其在实现if-else语句时的作用是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

2. 已知函数swap的C语言框架如下。回答问题（10分）

**int a = 200, b = 100;**

**void swap(int \*x, int \*y) {**

0x5655618d <+0>: push %ebp

0x5655618e <+1>: mov %esp,%ebp

0x56556190 <+3>: sub $0x10,%esp

0x56556193 <+6>: call 0x565561ee <\_\_x86.get\_pc\_thunk.ax>

0x56556198 <+11>: add $0x2e44,%eax

**int tmp;**

**if (\*x > \*y) {**

0x5655619d <+16>: mov 0x8(%ebp),%eax

0x565561a0 <+19>: mov (%eax),%edx

0x565561a2 <+21>: mov 0xc(%ebp),%eax

0x565561a5 <+24>: mov (%eax),%eax

0x565561a7 <+26>: cmp %eax,%edx

0x565561a9 <+28>: jle 0x565561c5 <swap+56>

**tmp = \*x;**

**\*x = \*y;**

**\*y = tmp;**

**}**

**}**

**void main() {**

**swap(&a, &b);**

=> 0x565561d5 <+13>: lea 0x30(%eax), %edx

0x565561db <+19>: push %edx

0x565561dc <+20>: lea 0x2c(%eax), %eax

0x565561e2 <+26>: push %eax

0x565561e3 <+27>: call 0x5655618d <swap>

(0x565561e3 <+27>: e8 a5 ff ff ff call 0x5655618d <swap>)

0x565561e8 <+32>: add $0x8, %esp

**}**

1. 根据反汇编指令， swap(&a, &b)函数调用时，如何传递参数的地址？（2分）
2. 在swap函数中，如何根据地址，获取a, b的值？（2分）
3. 反汇编指令：call 0x5655618d <swap>，对应的机器指令为：e8 a5 ff ff ff， 其中a5 ff ff ff的含义是什么？（3分）。
4. 请写出0x565561c5<swap+56> 处的机器指令，使得程序能正常返回。（3分）

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

**四、（20分）程序分析与优化**

1、根据反汇编代码，填补C语言语句缺失的部分（7分，每空1分）

**void main() {**

**char str1[13] = \_\_\_\_\_\_\_\_\_\_\_\_\_\_;**

0x565561d5 <+40>: movl $0x64636261, -0x26(%ebp) #字符a的ASCII为0x61

0x565561dc <+47>: movl $0x68676665, -0x22(%ebp)

0x565561e3 <+54>: movl $0x6c6b6a69, -0x1e(%ebp)

0x565561ea <+61>: movb $0x0, -0x1a(%ebp)

**char str2[13];**

**for (int i = \_\_\_\_\_\_\_\_; \_\_\_\_\_\_\_\_; \_\_\_\_\_\_\_\_) {**

0x565561ee <+65>: movl $0x0, -0x2c(%ebp)

0x565561f5 <+72>: jmp 0x56556215 <main+104>

**\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_;**

0x565561f7 <+74>: lea -0x26(%ebp), %ecx

0x565561fa <+77>: mov -0x2c(%ebp), %edx

0x565561fd <+80>: add %ecx, %edx

0x565561ff <+82>: movzbl (%edx), %edx

0x56556202 <+85>: mov %edx, %ecx

0x56556204 <+87>: and $0xf, %ecx

0x56556207 <+90>: lea -0x19(%ebp), %ebx # -0x19(%ebp)代表str2数组首字符

0x5655620a <+93>: mov -0x2c(%ebp), %edx

0x5655620d <+96>: add %ebx, %edx

0x5655620f <+98>: mov %cl, (%edx)

**}**

0x56556211 <+100>: addl $0x1, -0x2c(%ebp)

0x56556215 <+104>: cmpl $0xb, -0x2c(%ebp)

0x56556219 <+108>: jle 0x565561f7 <main+74>

**\_\_\_\_\_\_\_\_;**

0x5655621b <+110>: movb $0x0, -0xd(%ebp)

**}**

2. 上述程序的功能是什么？（2分）

3. 上述反汇编程序，效率较低的原因有哪些？（3分）

4、常见的程序优化方法有哪些？（3分）

5. 请用汇编语言程序片段给出功能等价的优化方案。（5分）

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

1. **（20分）程序的链接和异常控制流**

1、有两个源文件: main.c和test.c，它们的内容如下，将这两个文件分别编译后再链接成一个执行程序。回答下列问题。（10分）

/\*main.c\*/

int mul( );

int a[5]={1,2,3,4,5};

int main( ) {

int val;

val=mul();

printf("sum\_val=%d\n", val);

return 0;

}

/\*test.c\*/

extern int a[];

static int b[5]={6,7,8,9,10};

int mul( ) {

int i, sum\_val=0;

for (i=0; i<5; i++)

{ sum\_val = sum\_val + a[i] \* b[i]; }

return sum\_val;

}

（1）可重定位目标文件中有哪些节？每个节中包含有哪些信息？至少写出4个。（4分）

1. 对于编译生成的可重定位目标文件main.o和test.o，各自生成了哪些符号？并说明其类型（全局、外部、本地），以及这些符号出现在各自模块的哪个节中。（4分）
2. main.c中哪些语句中的那些符号需要重定位？（2分）
3. 回答与中断和异常相关的问题（10分）
4. 可屏蔽中断和不可屏蔽中断有什么区别？（2分）
5. 举3个异常的例子。（3分）
6. 用户单击了一下鼠标，请描述计算机的处理过程。（3分）
7. 中断向量表在中断响应过程中起到什么作用？（2分）