四川大学期末考试试题（闭卷）

# （2018--2019 学年第 1 学期） B卷

课程号： **311046040** 课程名称：系统级编程任课教师：适用专业年级：软件工程 **2016**级学号：姓名：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 考生承诺  我已认真阅读并知晓《四川大学考场规则》和《四川大学本科学生考试违纪作弊处分规定（修订）》，郑重承诺：   1. 已按要求将考试禁止携带的文具用品或与考试有关的物品放置在指定地点； 2. 不带手机进入考场； 3. 考试期间遵守以上两项规定，若有违规行为，同意按照有关条款接受处理。   考生签名： | | | | | | |  |
| 题号 | 一**(40%)** | 二**(10%)** | 三**(5%)** | 四**(15%)** | 五**(12%)** | 六**(8%)** | 七**(10%)** |
| 得分 |  |  |  |  |  |  |  |
| 卷面总分 |  | | 阅卷时间 |  | | |  |

注意事项：1. 请务必将本人所在学院、姓名、学号、任课教师姓名等信息准确填写在试题纸和添卷纸上；

1. 请将答案全部填写在本试题纸上；
2. 考试结束，请将试题纸、添卷纸和草稿纸一并交给监考老师。



|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

一、 单项选择题（本大题共20 小题，每小题2 分，共40 分）提示：在每小题列出的四个备选项中只有一个是符合题目要求的，请将其代码填写在下表中。错选、多选或未选均无分

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 3 4 5 6 7 8 9 10 | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  |  |  |  |  |  |  |  |  |  |

1. Which of the following numerical operations is most likely to lead to loss of precision?

a. Integer multiplication

* 1. Integer addition
  2. Floating-point addition
  3. Floating-point multiplication

答案：c

解析：整数的算术操作除了溢出的情况外，是不会出现精度损失的。浮点数，其本身就是个复杂的类型，采用的是近似存储，至于为什么是加法的精度损失更高，这要对计算机浮点数的表示方法和浮点数的计算都有了解，才能弄明白。加法运算，每次都会按位数少的小数来算，而乘法每次都需要重新计算位数，所以损失大，至于为什么那样计算的，就是上述的两个问题，表示和运算方式。

1. Which is the value of expression 0x23^0x64?
   1. 0x66
   2. 0x0
   3. 0x67
   4. 0x20

答案：算出来是0x47,0x23=0010 0011；0x64=0110 0100；异或的结果是0100 0111=0x47

1. Given the array int md[3][4], which code segment will cause compiler error?

a. for (int i = 0; i < 3; i++)

for (int j = 0; j < 4; j++)

\*(md + i\*4 + j) = 0;

* 1. for (int i = 0; i < 3; i++) for (int j = 0; j < 4; j++)

\*(\*(md+ i) + j) = 0;

* 1. for (int i = 0; i < 3; i++) for (int j = 0; j < 4; j++) md[i][j] = 0;
  2. for (int i = 0; i < 3; i++) for (int j = 0; j < 4; j++)

\*((int\*)(md) + i\*4 + j) = 0;

答案：a，md表示首地址，可以对其进行加减，但是前面加个\*号，其依旧是一个地址，将一个地址赋值成整数0，编译器会报类型不符合的错误，b,d选项都进行了类型转变，b将其变为指针，d将其变为整型的指针，所以编译能通过。c是一般访问数组的方式。

1. Which statement has NO effect on stack?
   1. ADD
   2. JMP
   3. PUSH
   4. RET

答案：a，b

解析：

A选项，add只是改变值，不改变栈帧

B选项，跳转指令，跳过某些代码，不影响栈帧

C选项，push入栈，改变。

D选项，ret指令，函数结束时指令，影响栈

1. Given the following code segment, which statement is right? Char str[10]=”Hello”; int x = size of(str); int y = strlen(str);

a. x is 5 and y is 5;

* 1. x is 10 and y is 10;
  2. x is 10 and y is 5;
  3. x is 5 and y is 10;

答案：c，sizeof是占空间的大小，10个字节，strlen是字符串的长度，5个字符。

1. Consider the following segment of a C program. int i = 99; int a[100]; i = a[i + 1];

Which of the following is true of the segment?

* 1. I will have the value of the last element of the array a at the end of any execution of the segment.
  2. When executed, the program will be prematurely terminated by the operating system because of an illegal memory access.
  3. I will have the value 99 at the end of any execution of the segment.
  4. Execution will fail because a has the wrong size.

答案：c，数组越界，栈帧99号上面一个元素是i的值，故a[100]的值为99，考点栈帧。

1. Why is the mantissa of a floating point number shifted left as far as possible?
   1. to retain as much precision as possible
   2. to avoid overflow
   3. to avoid underflow
   4. to align bit positions, simplifying addition

答案：a

解析：尾数尽可能的左移，使得右边的空间变大，左边要表示的数就越小，右边的数就多了，那么精度就会增加，所以选a。

1. How does x86 assembly store the return value when a function is finished?
   1. The ret instruction stores it in a special retval register.
   2. By convention, it is always in eax
   3. It is stored on the stack just above the ebp of the callee
   4. It is stored on the stack just above all the arguments to the function

答案：b，看一下汇编代码就清楚了，经常的做法是放入eax（累加器）

1. Immediately after the CPU executes an instruction that is neither a branch nor a jump instruction, the program counter

a. remains unchanged

* 1. is incremented by one
  2. is incremented to point to the following instruction
  3. has a value that cannot be determined without further information

答案：c，加一取下一跳指令

1. Which of the following features apply to standard heap allocation in C?
   1. The size of heap objects must be known at compile time.
   2. Heap memory must be explicitly allocated.

iii) Heap memory is deal located when a function returns.

b. II only.

* 1. I and II only.
  2. I and III.
  3. I only.

答案：b，堆的分配必须显示的使用malloc来分配。对于i，动态数组，数组的大小是在运行时确定的，并分配到堆上，所以是错的。对于iii，应该是调用free和malloc的过程中处理。

1. Why is it wrong to return the address of a local variable?
   1. It allows illegal access to the variable from arbitrary functions.
   2. The local variable may be in a machine register.
   3. It is faster to return the value of the variable.
   4. The variable address is invalid after the return.

答案：d，局部变量位于栈上，函数返回时，该栈帧会被销毁，地址无效。

1. In C, which of the following is the best way to detect when a pointer is freed twice?
   1. Set pointers to NULL after freeing them.
   2. Flag all blocks as free or not, and check the flag when calling free ().
   3. Keep a log of addresses that have been freed and scan the log before calling free ().
   4. Modify free () to set the freed data to zero.

答案：a，释放内存后赋值Null，再次释放会报错。

1. A memory pool is a large block of memory from which small objects are allocated piecemeal by breaking them off from the pool as required. Under which of the following conditions would such a scheme result in greatly improved performance?

i) All objects allocated from the pool are freed at around the same time.

ii) All objects allocated from the pool are of similar sizes.

iii) A garbage collector takes care of freeing memory. a. I only.

* 1. II only.
  2. III only.
  3. I and II only.

答案：b，所有的内存块分配相似大小，不用进行算法的比较，性能当然高，但是局限性也多，内部碎片严重，还有某个需求比较大，无法满足。

1. Two [code](https://cn.bing.com/dict/search?q=code&FORM=BDVSP6&mkt=zh-cn) fragments are i) and ii) as following:
   1. M= 100000; N=100000; int data[M][N]; For (j = 0 ; j < N; j++) {

For (i = 0; i < M; i++) {

Sum += data[ i ] [ j ];

}

}

* 1. M= 100000; N=100000; int data[M][N]; For (i = 0; i < M; i++) {

For (j = 0 ; j < N; j++) {

Sum += data[ i ] [ j ];

}

}

Which one is more efficient (better performance) if you program with Mat lab which is column-major order for matrix?

* 1. Both are very efficient.
  2. i)
  3. ii)
  4. Both are not efficient.

答案：c，行访问更快。

1. A garbage collector
   1. Frees memory blocks that cannot be reached by dereferencing pointers.
   2. Frees all memory blocks that will not be accessed in the future.
   3. Frees memory blocks marked as "delete able".
   4. Removes old versions of local variables from the stack.

答案：a，垃圾回收就是回收那些不可达的内存块，不可达就是没有指针在指向他。

1. What is tsc?
   1. a timer mechanism of OS
   2. a timer mechanism of x86 platform, which is the short name of time stamp counter

c. a system call of OS

d. a timer mechanism of c library

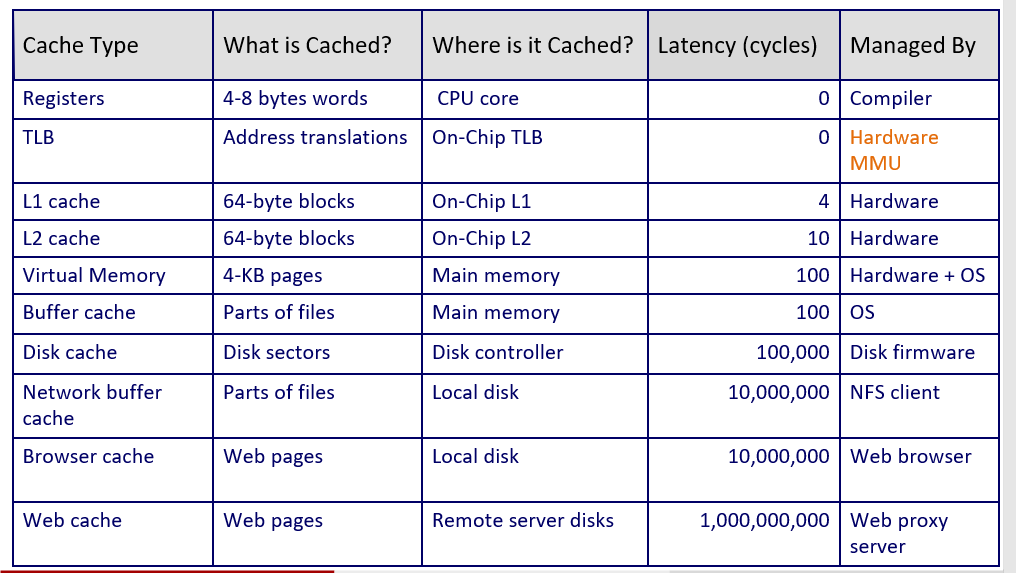
答案：b，tsc时间戳计数器。

1. Which of the following is not optimization technique?
   1. code motion
   2. loop unrolling
   3. constant folding
   4. memory aliasing

答案：d，存储器别名和函数副作用是妨碍优化的因素。

1. Which of the following manages the transfer of data between the cache and main memory?
   1. Registry.
   2. Operating System.
   3. Hardware.
   4. Compiler.

答案：b



1. What can Linker do?

i) Symbol Resolution

ii) Resolution

iii) load or map the Executable object file from the disk to memory

a. i) and ii) only.

* 1. i) and iii) only.
  2. i), ii) and iii).
  3. iii) only.

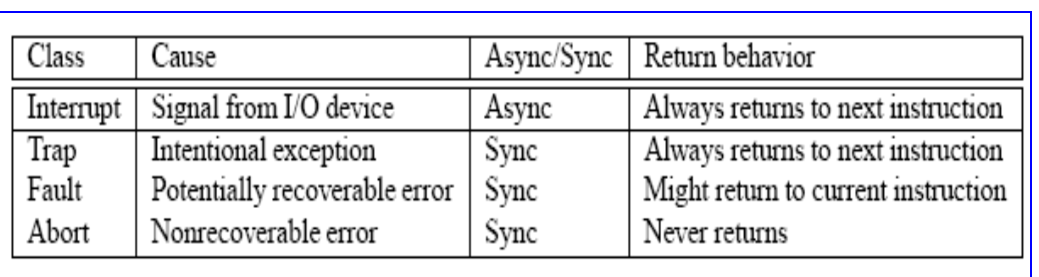
答案：a，iii是加载器做的事，链接器负责符号解析和重定位。

1. In IA32 or X86, which exception returns back to the point where exception happens?

a. interrupt

* 1. trap
  2. fault
  3. Abort

答案：c，返回当前的指令，指出错误的位置。



|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 二、 bit operation（本大题共2 小题，各5 分，共10 分）

Now complete the following functions according to the following rules.

Each "Expr" is an expression using ONLY the following:

1. Integer constants 0 through 0xFFFFFFFF inclusive.
2. Function arguments and local variables (no global variables).
3. Some of the problems restrict the set of allowed operators.

You are expressly forbidden to:

1. Use any control constructs such as if, do, while, for, switch, etc.
2. Define or use any macros.
3. Define any additional functions in this file.
4. Call any functions.
5. Use any other operations, such as &&, ||, -, ?, or [] :
6. Use any form of casting.

You may assume that your machine:

1. Uses 2s complement, 32-bit representations of integers.
2. Performs right shifts arithmetically.
3. Has unpredictable behavior when shifting an integer by more than the word size.

/\* least Bit Pos - return a mask that marks the position of the

* least significant 1 bit. If x == 0, return 0
* example: least Bit Pos (96) = 0x20
* legal ops:! ~ & ^ | + << >>
* max ops: 6
* rating: 4

\*/ int least Bit Pos(int x) {

return x&((~x)+1);

}

/\*

* logical Shift - shift x to the right by n, using a logical shift
* Can assume that 1 <= n <= 31
* Examples: logical Shift(0x87654321,4) = 0x08765432
* Legal ops: ~ & ^ | + << >>
* Max ops: 16
* Note: data type int indicates it is mathematical shift

\*/ int logical Shift(int x, int n) {

return (x >> n) & 0x7FFFFFFF;//逻辑右移符号位无论正负，均填0

}

|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 三、memory allocation（本大题共1 小题，共5 分）

Are there any errors in the following program? If there are, please correct them on the right side.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void get memory(char \*p)

{

//p=(char \*) malloc(100);

strcpy(p,”hello world”);

}

int main( )

{

char \*str=NULL;

str = (char \*) malloc(100);

getmemory(str); printf(“%s/n”,str); free(str); return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void getmemory(char \*\*p)

{

\*p=(char \*) malloc(100);

strcpy(\*p,"hello world");

}

int main( )

{

char \*str=NULL;

getmemory(&str);

printf("%s\n",str);

free(str);

return 0;

}

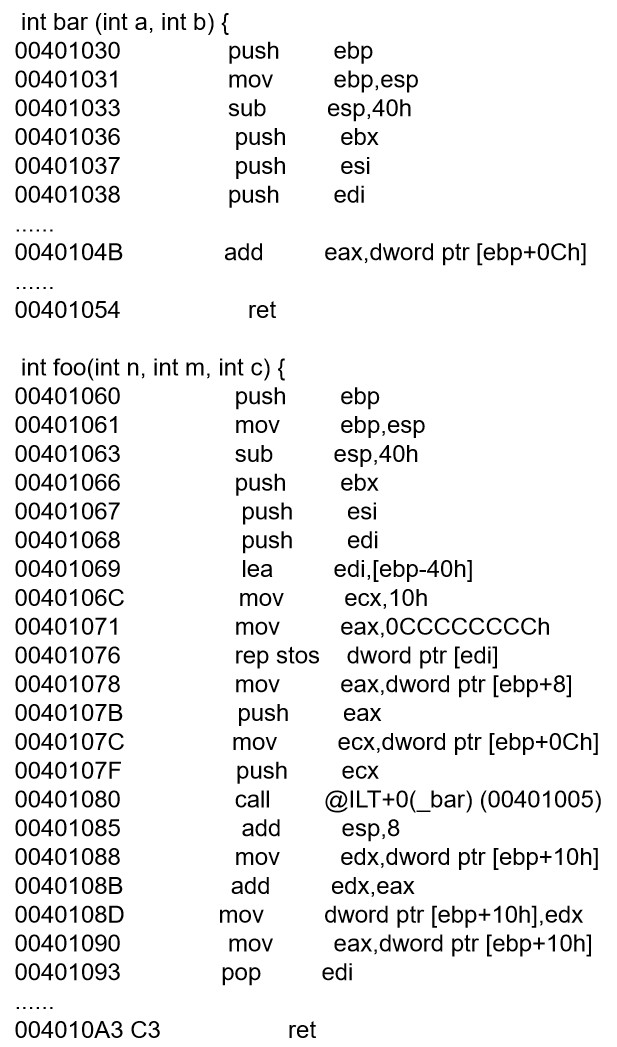
解析：

头文件少了string.h，在函数中分配空间，随着栈帧的变化，会被擦除，那么str始终指向的null，不会得到正确结果，修改如上，在外部分配str的空间，在函数内对它进行修改，便能得到正确的结果。

|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 四、stack discipline（本题共15 分）

Stack discipline. Consider the following C code and its corresponding 32-bit x86 machine code. Please complete the stack diagram on the following page.

 int bar (int a, int b) {

return a + b;

}

int foo(int n, int m, int c) {

c += bar(m, n);

return c;

}

1. (12 points) Draw a detailed picture of the stack, starting with the caller invoking foo (3, 4, 5), and ending immediately before execution of the ret instruction in bar. Return address in function that called foo : 0x004010E3

The diagram starts with the address arguments for foo().

|  |  |  |
| --- | --- | --- |
| **Address no.** | **value** | **description** |
| 0xffffd850 | 5 | 参数c |
| 0xffffd84c | 4 | 参数m |
| 0xffffd848 | 3 | 参数n |
| 0xffffd844 | 0x004010E3 | 返回地址 |
| 0xffffd840 | unknown | old ebp |
| **Address no.** | **value** | **description** |
| 0xffffd83c |  |  |
| 0xffffd838 |  |  |
| 0xffffd834 |  |  |
| 0xffffd830 |  |  |
| 800 |  | 40hcc |
| 7fc | ebx | push ebx |
| 7f8 | esi | push esi |
| 7f4 | edi | push edi |
| 7f0 | 3 | 参数b |
| 7ec | 4 | 参数a |
| 7e8 | 00401085 | 返回值 |
| 7e4 | ffffd840 | old ebp |
| 7a4 |  | 40h cc |
| 7a0 | ebx | push ebx |
| 79c | esi | push esi |
| 798 | edi | push edi |

1. (1.5 points) What is the final value of ebp, immediately before execution of the ret instruction in bar?

ebp=0x\_\_\_\_\_\_\_\_ffffd840\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (1.5 points) What is the final value of esp, immediately before execution of the ret instruction in bar?

esp=0x\_\_\_\_ffffd7e8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 五、performance optimization（本大题共2 小题，共12 分）

1. (6 points) We are considering an enhancement to the processor of a web server. Then new CPU is 30 times faster on search queries than the old processor. The old processor is busy with search queries 80% of the time, what is the speedup gained by integrating the enhanced CPU?

加速比的计算公式：1/(1-P+P/S)=1/(0.2+0.8/30)=4.35

1. (6 points) Read the following two program fragments. Do they have same behavior in any circumstances? Explain your answer.

void twiddle1 (int \*xp, int \*yp) {

\*xp += \*yp;

\*xp += \*yp;

}

void twiddle2 (int \*xp, int \*yp) { \*xp +=2 \*yp;

}

解析：可能或得到不同的结果，当xp和yp都指向同一块区域时，便会得到不同的结果，假设都指向整数2，前一个结果是8，后一个的结果是6.

|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 六、cache（本大题共1 小题，共8 分）

Read the code as following:

int i

int sum=0; int data[100000];

for (i = 0; i < 100000; i += stride) sum += data[i];

Assume that size of int is 4-byte and the cache line (block) length is 16-byte.

If the stride is 1,2,4,8, what is their hit ratio and miss ratio? Fill the blanks in the table and give your explain.

|  |  |  |
| --- | --- | --- |
| Stride | Hit ratio | Miss ratio |
| 1 | 0.75 | 0.25 |
| 2 | 0.5 | 0.5 |
| 4 | 0 | 1 |
| 8 | 0 | 1 |

i=1，每4次就未命中一次，总共访问1000次，故未命中率等于250/1000=0.25，命中率为0.75

i=2，每2次就未命中一次，总共访问500次，故未命中率等于250/500=0.5，命中率为0.5

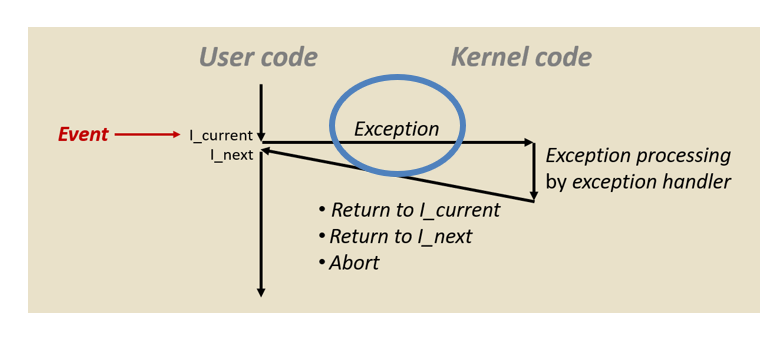
i=4，每次均未命中，所以未命中率为1，命中率为0

i=8，每次均未命中，所以未命中率为1，命中率为0

|  |  |
| --- | --- |
| 评阅教师 | 得分 |
|  |  |

## 七、Exception（本大题共1 小题，共10 分）

According to our textbook, what is the relationship between Exception and the OS? Please give your point on this, and drawing a figure is suggested.



详情参考ppt week15