Andrew ID (print clearly!):	
Full Name:	

# 15-213/18-213, Fall 2012

# **Midterm Exam**

Tuesday, October 16, 2012

#### **Instructions:**

- Make sure that your exam is not missing any sheets, then write your Andrew ID and full name on the front.
- This exam is closed book, closed notes (except for 1 double-sided note sheet). You may not use any electronic devices and the latest and the latest electronic devices and the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices are the latest electronic devices and the latest electronic devices are the latest electronic devices electroni
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maintain Ste of powcoder.com
- The problems are of varying difficulty. The point value of each problem is indicated. Good luck!

# Add WeChat powcoder 2 (07): 3 (06): 4 (08): 5 (08): 6 (04): 7 (08): 8 (10): 9 (10): TOTAL (71):

# Problem 1. (10 points):

Multiple choice. Write your answer for each question in the following table:

1	2	3	4	5	6	7	8	9	10

1. What is the output of the following code?

Assume that int is 32 bits, short is 16 bits, and the representation is two's complement.

```
unsigned int x = 0xDEADBEEF;
unsigned short y = 0xFFFF;
signed int z = -1;
if (x > (signed short) y)
    printf("Hello");
if (x > z)
    printf("World");
```

- (a) Prints nothing.
- (b) Prints "Hello"
- (c) Pri Assignment Project Exam Help
- (d) Prints "HelloWorld"
- 2. 1) mov (%eaxhttps4)/pewcoder.com

Which of the above accomplishes the following: eax = 5 \* eax?

- (a) Neither 1 nor Add WeChat powcoder
- (b) Only 1.
- (c) Only 2.
- (d) Both 1 and 2.
- 3. The x86-64 instruction test is best described as which of the following:
  - (a) Same as sub.
  - (b) Same as sub, but doesn't keep the result (only sets flags).
  - (c) Same as and.
  - (d) Same as and, but doesn't keep the result (only sets flags).

4. In the following code, what order of loops exhibits the best locality?

```
// int a[X][Y][Z] is declared earlier
int i, j, k, sum = 0;
for (i = 0; i < Y; i++)
    for (j = 0; j < Z; j++)
        for (k = 0; k < X; k++)
        sum += a[k][i][j];</pre>
```

- (a) i on the outside, j in the middle, k on the inside (as is).
- (b) j on the outside, k in the middle, i on the inside.
- (c) k on the outside, i in the middle, j on the inside.
- (d) The order does not matter.
- 5. Which expression will evaluate to  $0 \times 1$  if x is a multiple of 32 and  $0 \times 0$  otherwise? Assume that x is an unsigned int.
  - (a) ! (x & 0x1f)
  - (b) !(x & 0x3f)
  - (c) Assignment Project Exam Help
  - (d)  $(x \mid 0x31)$
  - (e)  $!(x ^0x1f)$
- 6. On a 32-bit Linux lysten posat /s/the poetwacoder.com
  - (a) 2 bytes
  - (b) 4 bytes

# Add WeChat powcoder

- (c) 6 bytes
- (d) 8 bytes
- (e) 16 bytes
- 7. Consider the C declaration

```
int array[10] = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\};
```

Suppose that the compiler has placed the variable array in the %ecx register. How do you move the value at array[3] into the %eax register? Assume that %ebx is 3.

- (a) leal 12 (%ecx), %eax
- (b) leal (%ecx, %ebx, 4), %eax
- (c) movl (%ecx, %ebx, 4), %eax
- (d) mov1 8(%ecx, %ebx, 2), %eax
- (e) leal 4(%ecx, %ebx, 1), %eax

- 8. Why does the technique called "blocking" help with cache utilization when transposing a matrix?
  - (a) Inductive locality
  - (b) Spatial locality
  - (c) Monadic locality
  - (d) Temporal locality
  - (e) Internet locality
- 9. What is NOT true about 64-bit Linux systems?
  - (a) %rax is used for function return values
  - (b) There are more registers than there are in 32-bit systems
  - (c) All function arguments are passed on the stack
  - (d) %rbp can be used like any other register; there is no base pointer
  - (e) %eax and %ebx can be used like in a 32-bit system.
- 10. On a 64-bit system, if %rsp has the value 0x7ffff0000 immediately before a retq instruction, what is the value of %rsp immediately after the retq?
  - (a) OASSIGNMent Project Exam Help

  - (c) 0x7fffff0004
  - (d) 0x7ffffff https://powcoder.com
    (e) The return address

Add WeChat powcoder

# Problem 2. (7 points):

*Integer encoding.* Assume we are running code on two machines using two's complement arithmetic for signed integers. Machine 1 has 4-bit integers and Machine 2 has 6-bit integers. Fill in the empty boxes in the table below. The following definitions are used in the table:

```
int x = -5;
unsigned ux = x;
```

Expression	4-bit decimal	4-bit binary	6-bit decimal	6-bit binary
-8	-8		-8	
-TMin				
x >> 1				
$(-x^{}(-1)) >> 2$				

# Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

# Problem 3. (6 points):

*Floating point encoding.* In this problem, you will work with floating point numbers based on the IEEE floating point format. We consider two different 6-bit formats:

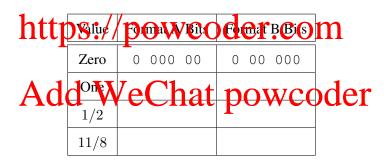
#### Format A:

- There is one sign bit s.
- There are k=3 exponent bits. The bias is  $2^{k-1}-1=3$ .
- There are n=2 fraction bits.

#### Format B:

- There is one sign bit s.
- There are k=2 exponent bits. The bias is  $2^{k-1}-1=1$ .
- There are n=3 fraction bits.

For formats Ans 5 bease with 6 which bild or presentation for defolor in the bild of the bild of the formation of the bild of the bild



## Problem 4. (8 points):

}

Loops. Consider the following x86 assembly code:

```
(gdb) disassemble transform
  0 \times 080483 d0 <+0>:
                         push
                                 %ebp
  0x080483d1 <+1>:
                         mov
                                 %esp, %ebp
  0x080483d3 <+3>:
                                 0x8(%ebp), %edx
                         mov
                                 $0x0, %eax
  0x080483d6 < +6>:
                         mov
                                 %edx, %edx
  0 \times 080483 db < +11>:
                        test
                        je
  0 \times 080483 dd < +13>:
                                0x80483ec <transform+28>
  0x080483df <+15>:
                        test $0x1,%dl
                               0x80483e8 <transform+24>
  0x080483e2 < +18>:
                        jе
  0x080483e4 <+20>:
                       lea
                               0x1(%eax, %eax, 1), %eax
  0x080483e8 < +24>:
                        shr
                                %edx
  0x080483ea <+26>:
                                0x80483df <transform+15>
                         jne
  0x080483ec < +28>:
                         pop
                                 %ebp
  0x080483ed <+29>:
                         ret
```

Given this assembly code, reconstruct the C transform function.

• Recall that %dl is the low-order byte of edx.

ASSIGNMENT Project Exam Help
• Recall that if a shift amount is not specified in the shr instruction, a default shift amount of 1 is used.

# Problem 5. (8 points):

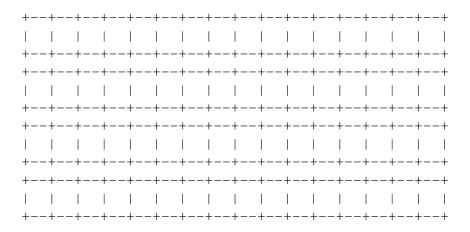
Struct alignment. Consider the following C struct declaration:

```
typedef struct {
  char a;
  long b;
  float c;
  char d[3];
  int *e;
  short *f;
} foo;
```

1. Show how foo would be allocated in memory on an x86-64 Linux system. Label the bytes with the names of the various fields and **clearly mark the end of the struct**. Use an X to denote space that is allocated in the struct as padding.



2. Rearrange the elements of foo to conserve the most space in memory. Label the bytes with the names of the various fields and **clearly mark the end of the struct**. Use an X to denote space that is allocated in the struct as padding.

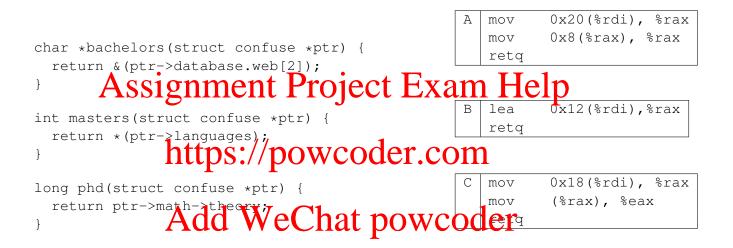


## Problem 6. (4 points):

Struct Access. Now for something totally different... Dr. Grave O'Danger is now head of the computer science department and has decided to make it impossible for you to graduate.

```
struct confuse {
  char systems;
  long theory;
  struct applications {
    char web[3];
  } database;
  int *languages;
  struct confuse *math;
};
```

Below are three C functions and three x86-64 Linux code blocks.



In the following table, next to the name of each x86-64 code block, write the name of the C function that it implements.

Code Block	<b>Function Name</b>
А	
В	
С	

# Problem 7. (8 points):

Switch statements. The following problem tests your understanding of switch statements that use jump tables.

Consider a switch statement with the following implementation. The code uses this <code>jmpq</code> instruction to index into the jump table:

```
0x40047b jmpq *0x400598(,%rdi,8)
```

#### Using GDB we extract the jump table:

0x400598:	0x000000000400488	0x0000000000400488
0x4005a8:	0x000000000040048b	0x0000000000400493
0x4005b8:	0x000000000040049a	0x000000000400482
0x4005c8:	0x000000000040049a	0x000000000400498

Here is the assembly code for the switch statement:

```
#on entry
                   ent Project Exam Help
0x400474
                   0x40049a
0x400477:
             ja
0x400479 :
                   %edi,%edi
             mov
                   *%/powcoder.com
0x40047b:
0x400482 :
0x400487 :
             retq
                      hat powcoder
0x400488 :
             sub-
0x40048b :
0x400492 :
             retq
0x400493 :
                   $0x2, %edx
             mov
0x400498 :
                   %edx,%esi
             and
0x40049a :
             lea
                   0x4(%rsi), %eax
0x40049d:
            retq
```

#### Fill in the C code implementing this switch statement:

```
int main(int a, int b, int c){
    int result = 4;
   switch(a){
      case 0:
      case 1:
      case __:
          break;
      case __:
      Assignment Project Exam Help
          break;
      case 3: https://powcoder.com
      case 7: Add' WeChat powcoder
      default:
   }
   return result;
}
```

# Problem 8. (10 points):

*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;
}
08048374 <bar>:
                55
8048374:
                                         push
                                                %ebp
8048375:
                89 e5
                                                %esp, %ebp
                                         mov
8048377:
                8b 45 0c
                                         mov
                                                0xc(%ebp), %eax
804837a:
                03 45 08
                                                0x8(%ebp), %eax
                                         add
 804837d:
                5d
                                         pop
                                                %ebp
               ignment Project
 804837e:
0804837f <foo>:
 804837f:
                                                %ebp
                https://powcoder.comebp
8048380:
                83 ec 08
8048382:
                                         sub
                                                $0x8, %esp
                8b 45 08
                                         mov
                                                0x8(%ebp), %eax
8048385:
8048388:
                                                 %eax.1x4(%esp)
                                                 exe(sebp), %eax
804838c:
804838f:
                89 04 24
                                                %eax, (%esp)
                                         mov
8048392:
                e8 dd ff ff ff
                                         call
                                                8048374 <bar>
                03 45 10
8048397:
                                         add
                                                0x10(%ebp), %eax
 804839a:
                С9
                                         leave
 804839b:
                с3
                                         ret
```

**A.** 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.

- The stack diagram should begin with the three arguments for foo that the caller has placed on the stack. To help you get started, we have given you the first one.
- Use the actual values for function arguments, rather than variable names. For example, use 3 or 4 instead of n or m.
- Always label %ebp and give its value when it is pushed to the stack, e.g., %ebp: 0xffff1400.
- You may not need to fill in all of the boxes in the diagram.

%esp=0x\_\_\_\_\_

	ebp when foo is called: 0xffffd858 ress in function that called foo: 0x08	30483c9
Stack addresss	The diagram starts with the arguments for foo()	
0xffffd850	5	
0xffffd84	ssignment Project Ex	am Help
0xffffd848		
0xffffd844	https://powcoder.c	om
0xffffd840	A 1 1 XX7 (C1)	1
0xffffd83c	Add WeChat power	coder
0xffffd838	 	
0xffffd834	 	
0xffffd830		
	+	<del>-</del>
<b>B.</b> What is the f	inal value of %ebp, immediately <b>before</b> execution of th	e ret instruction in bar?
%ebp=0x		
<b>C.</b> What is the f	inal value of %esp, immediately <b>before</b> execution of the	e ret instruction in bar?

# Problem 9. (10 points):

*Caches*. Consider a computer with an **8-bit address space** and a **direct-mapped 64-byte** data cache with **8-byte cache blocks**.

- A. The boxes below represent the bit-format of an address. In each box, indicate which field that bit represents (it is possible that a field does not exist) by labeling them as follows:
  - B: Block Offset
  - S: Set Index
  - T: Cache Tag

9

10

67

07

0110 0111

0000 0111

7	6	5	4	3	2	1	0

B. The table below shows a trace of load addresses accessed in the data cache. Assume the cache is initially empty. For each row in the table, please complete the two rightmost columns, indicating (i) the *set number* (in decimal notation) for that particular load, and (ii) whether that loads *hits* (H) or misses (M) in the cache (circle either "H" or "M" accordingly)

misses (M) in the cache (circle either "H" or "M" accordingly). Load Hex **Binary** Set Number? Hit or Miss? (in Decimal) (Circle one) No. Address-Address 0100 0011 43 1 2 b2 1011 0010 Η powcoder 010000 40 3 4 f9 1111 1001 Η M 5 1011 0010 Η b2 M 93 1001 0011 6 Η M 7 d01101 0000 Η M 8 **b**0 1011 0000 Η M

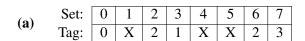
Η

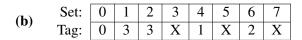
Η

M

M

C. For the trace of load addresses shown in Part B, below is a list of possible final states for the cache, showing the hex value of the tag for each cache block in each set. Assume that initially all cache blocks are invalid (represented by X).





(c)	Set:	0	1	2	3	4	5	6	7
	Tag:	0	X	3	X	1	X	2	3

Assignment Project Exam Help

Set: 0 1 2 3 5 6 **(g)** 3 Tag: X 2 1 X 2 X https://powcoder.com