Introduction to Computer Systems
2013 Spring Midterm Examination

Name_____ Student No._____ Score_____

Problem 1: (18 points)

[1] [2] [3]

[4] [5] [6]

[7] [8] [9]

Problem 2: (10 points)

[1] [2]

[3] [4]

[5] [6]

[7] [8]

[9] [10]

Problem 3: (12 points)

[1] [2]

[3] [4]

[5] [6]

Problem 4: (20 points)

1 [1] [2]

[3]

2 [1] [2]

[3] [4]

[5] [6]

3 [1]

Problem 5: (20 points)

1 [1] [2] [3]

[4] [5] [6]

[7] [8] [9]

[10] [11] [12]

2

3

Problem 6: (20 points)

1 **[1]** **[2]**

[3] **[4]**

[5] **[6]**

[7] **[8]**

[9]

2 **[1]** **[2]**

[3] **[4]**

[5] **[6]**

[7] **[8]**

3

Problem 1: (18 points)

1. Consider the following C program

```
short s = -3;
unsigned short us = s;
int i = -27;
unsigned int k = i | 21;
unsigned int a = ~1;
int result = (-1 <= a);
```

Assume we are running code on a **6-bit** machine using two's complement arithmetic for signed integers. Also assume that right shifts of signed values are performed **arithmetically**. A "short" integer is encoded using **3 bits**. Please fill in the blanks of table below. (2'*9=18')

Expression	Binary Representation
-3	[1]
k	[2]
~i	[3]
(us>>2) + (s>>1)	[4]
k ^ i	[5]
(short)-1 && (us>>2)	[6]
result	[7]
TMIN	[8]
i = i - k	CF = [9]

Problem 2: (10points)

Suppose a **32-bit little endian** machine has the following memory and register status. (NOTE: **Instructions are independent)**.

Memory status

Address	Value
0x102	0xabcdef12
0x104	0x12345678
0x108	0x10011001

Register status

Register	Value
%eax	0x00000104
%ebx	0x00000002
%ecx	0x00000101
%edx	0xa1b2c3d4

Fill in the blanks using **4 byte size** and **hex**. (1' * 10 = 10')

Operation	Destination	Value
decl (%eax)	[1]	[2]
imull $32, (%eax, %ebx, 2)	[3]	[4]
leal (%eax, %ebx, 4), %ecx	[5]	[6]
addl %eax, 4(%eax)	[7]	[8]
movb %dl, %al	[9]	[10]

Problem 3: (12points)

Suppose the following code is executed on a **32-bit little-endian** machine, where "int" is 4 bytes, "short" is 2 bytes, "char" is 1 byte and "pointer" is 4 bytes. The address of global variable **"struct data d"** is **0x804a060**. Please calculate the value of the following expressions according to definition of struct (2'*6 = 12')

```
struct data {
    char c[3];
    short s;
    int i;
    union {
        char c;
        int i;
    } u;
    char *p;
};
struct data d;
```

Representation	Value
sizeof(struct data)	[1]
&(d.c[2])	[2]
&(d.s)	[3]
&(d.i)	[4]
&(d.u.c)	[5]
&(d.u.i)	[6]

Problem 4: (20points)

Suppose the following code is executed on a **32-bit little-endian** machine.

```
int nested_array[4][4];
int main(void) {
   int i, j;
   for (i = 0; i != 4; i++) {
      /* access as a SHORT array */
      short *sp = (short *) nested_array[i];
      for (j = 0; j != 8; j++) {
         *sp = (i << 8) + j;
         sp++;
      }
   }
}
```

```
 80483dc: 55                    push   %ebp
 80483dd: 89 e5                 mov    %esp,%ebp
 80483df: 83 ec 10              sub    $0x10,%esp
 80483e2: c7 45 f4 00 00 00 00  movl   $0x0,-0xc(%ebp)
 80483e9: eb 3e                 jmp    8048429 <main+0x4d>
 80483eb: 8b 45 f4              mov    -0xc(%ebp),%eax
 80483ee: c1 e0 04              shl    ___[1]___,%eax
 80483f1: 05 40 a0 04 08        add    $0x804a040,%eax
 80483f6: 89 45 fc              mov    %eax,-0x4(%ebp)
 80483f9: c7 45 f8 00 00 00 00  movl   $0x0,-0x8(%ebp)
__[2]__: eb 1d                  jmp    804841f <main+0x43>
 8048402: 8b 45 f4              mov    -0xc(%ebp),%eax
 8048405: 89 c2                 mov    %eax,%edx
 8048407: c1 e2 08              shl    $0x8,%edx
 804840a: 8b 45 f8              mov    -0x8(%ebp),%eax
 804840d: 01 d0                 add    %edx,%eax
 804840f: 89 c2                 mov    %eax,%edx
 8048411: 8b 45 fc              mov    -0x4(%ebp),%eax
 8048414: 66 89 10              mov    ___[3]___,(%eax)
 8048417: 83 45 fc 02           addl   ___[4]___,-0x4(%ebp)
 804841b: 83 45 f8 01           addl   $0x1,-0x8(%ebp)
 804841f: 83 7d f8 08           cmpl   ___[5]___,-0x8(%ebp)
 8048423: 75 dd                 jne    8048402 <main+0x26>
 8048425: 83 45 f4 01           addl   $0x1,-0xc(%ebp)
 8048429: 83 7d f4 04           cmpl   ___[6]___,-0xc(%ebp)
 804842d: 75 bc                 jne    80483eb <main+0xf>
 804842f: c9                    leave
 8048430: c3                    ret
```

Suppose the global variable **nested_array** starts from **0x804a040**. Please answer the following questions:

1. **BEFORE** executing the instruction **"sub $0x10,%esp" (0x80483df)**, the value stored in **%ebp** is **0xbfda8210**. Please calculate the stack address of the following local variables: (2'*3 = 6')
   address of i: ___[1]___
   address of j: ___[2]___
   address of sp: ___[3]___
2. Please fill in the blanks within C and assembly code. (2' * 6 = 12')
3. What is the value of **nested_array[3][3] BEFORE** the **main** function returns? Please write it in hexadecimal. (2')
   nested_array[3][3]: 0x___[1]___

Problem 5: (20points)

Reference:

Char	'0'	'9'	'A'	'Z'	'a'	'z'
ASCII	48(0x30)	57(0x39)	65(0x41)	90(0x5A)	97(0x61)	122(7A)

Code:

```
int encrypt(int ch) {
  int ret, digit;
  if (__[1]__) {
    digit = ch - '0';
    ret = __[2]__;
  } else {
    switch (ch) {
      case 'A':
        ret = ch + 5; break;
      case __[3]__: case __[4]__:
        ret = ch ^ 1; break;
      case 'E':  case 'F':
        ret = ch - 1; break;
      default:
        ret = 'A'; break;
    }
  }
  return ret;
}
```

```
.section .rodata
.align 4
L6:
   .long L5
   .long L7
   .long L7
   .long __[5]__
   .long __[6]__
   .long __[7]__
```

```
<_encrypt>:
   pushl %ebp
   movl  %esp, %ebp
   subl  $16, %esp
   cmpl  $48, 8(%ebp)
   jl    L2
   cmpl  $57, 8(%ebp)
   jg    L2
   movl  8(%ebp), %eax
   subl  $48, %eax
   movl  %eax, -8(%ebp)
   movl  $57, %eax
   subl  -8(%ebp), %eax
   movl  %eax, -4(%ebp)
   jmp   __[8]__
L2:
   movl  8(%ebp), %eax
   subl  __[9]__, %eax
   cmpl  __[10]__, %eax
   ja    __[11]__
   jmp   __[12]__
```

```
L5:
   movl  8(%ebp), %eax
   addl  $5, %eax
   movl  %eax, -4(%ebp)
   jmp   L3
L7:
   movl  8(%ebp), %eax
   xorl  $1, %eax
   movl  %eax, -4(%ebp)
   jmp   L3
L8:
   movl  8(%ebp), %eax
   subl  $1, %eax
   movl  %eax, -4(%ebp)
   jmp   L3
L4:
   movl  $65, -4(%ebp)
   nop
L3:
   movl  -4(%ebp), %eax
   leave
   ret
```

Suppose the C and assembly code are executed on a 32-bit little endian machine. Read the code and answer the following questions.

1. Please fill in the blanks within C and assembly code. (1'*12)

2. If we use the above function to encrypt the string '**BE1DA7'**, what is the encrypted string? (4')

3. We find this function a bit slow due to many jump instructions in the code. Can you optimize the C code to eliminate them as many as possible? (**Hint**: you can use array and refer to the idea of 'Jump Table'.) (4')

Problem 6: (20points)

Suppose the following C code and assembly code are executed on a 32-bit little endian machine. Read the code and answer the following question:

```
int main(void) {
   fib(4);
   return 0;
}
```

```
int fib(int n) {
   if(n<3) return n;
   return fib(n-1) + fib(n-2);
}
```

```
080483a0 <main>:
 80483a0: 55                    push   %ebp
 80483a1: 89 e5                 mov    %esp,%ebp
 80483a3: 83 e4 f0              and    $0xfffffff0,%esp
 80483a6: 83 ec 10              sub    $0x10,%esp
 80483a9: c7 04 24 04 00 00 00  movl   __[1]__,(%esp)
 80483b0: e8 07 00 00 00        call   __[2]__
 80483b5: b8 00 00 00 00        mov    $0x0,__[3]__
 80483ba: c9                    leave
 80483bb: c3                    ret

080483bc <fib>:
 80483bc: 55                    push   %ebp
 80483bd: 89 e5                 mov    %esp,%ebp
 80483bf: 53                    push   __[4]__
 80483c0: 83 ec 14              sub    $0x14,%esp
 80483c3: 83 7d 08 02           cmpl   $0x2,0x8(%ebp)
 80483c7: 7f 05                 jg     80483ce <fib+0x12>
 80483c9: 8b 45 08              mov    __[5]__, %eax
 80483cc: eb 21                 jmp    80483ef <fib+0x33>
 80483ce: 8b 45 08              mov    0x8(%ebp),%eax
 80483d1: 83 e8 01              sub    $0x1,%eax
 80483d4: 89 04 24              mov    __[6]__,(%esp)
 80483d7: e8 e0 ff ff ff        call   80483bc <fib>
 80483dc: 89 c3                 mov    %eax,%ebx
 80483de: 8b 45 08              mov    0x8(%ebp),%eax
 80483e1: 83 e8 02              sub    $0x2,%eax
 80483e4: 89 04 24              mov    %eax,(%esp)
 80483e7: e8 d0 ff ff ff        call   80483bc <fib>
 80483ec: 8d 04 03              lea    __[7]__,%eax
 80483ef: 83 c4 14              add    __[8]__,%esp
 80483f2: 5b                    pop    __[9]__
 80483f3: 5d                    pop    %ebp
 80483f4: c3                    ret
```

1. Fill in the blanks in the Assembly Code. (1' * 9 = 9').

2. BEFORE executing the instruction **"push %ebp" (0x80483a0)**, the values of registers are **%esp = 0xbfdaff34**, **%ebp = 0xbfdaf4a4**. Please fill the following blanks. (1'*8=8')

   BEFORE executing the instruction **"call __[2]__" (0x80483b0)**

register/address	value
%esp	[1]
%ebp	[2]
0xbfdaff20	[3]

   AFTER the **FIRST** time executing the instruction **"call 80483bc<fib>" (0x80483d7)**

register/address	value
%esp	[1]
%ebp	[2]
0xbfdaff00	[3]

   BEFORE the **FIRST** time executing the instruction **"mov %eax, %ebx" (0x80483dc)**

register/address	value
%eax	[7]
%ebp	[8]

3. The function **fib ()** implements Fibonacci Sequence Number. In **main()** function, how many stack frames of function **fib()** will be created by executing **fib(4)**? (3')