上 海 交 通 大 学 试 卷（A 卷）

（ 2019 至 2020 学年 第 1 学期 ）

班级号_____ 学号_____ 姓名 _____

课程名称 计算机系统基础（汇编） 成绩 _____

Problem 1

[1] [2]

[3] [4]

[5] [6]

Problem 2

[1] [2]

[3] [4]

[5] [6]

[7] [8]

[9] [10]

Problem 3

1. [1] [2]

   [3] [4]

2. FP=

   sign= exp= frac=

Problem 4

1. [1] [2]

   [3] [4]

   [5] [6]

   [7] [8]

我承诺，我将严格遵守考试纪律。

题号	1	2	3	4	5	6			
得分									
批阅人(流水阅卷教师签名处)									

2.

3.

Problem 5

1 [1] [2]

[3] [4]

[5] [6]

[7] [8]

2. [1] [2]

[3] [4]

Problem 6

1 [1]

[2]

2

3

Problem 1 (12 points)

1. Consider the following C program

```
int a = 0x3b;
int b = a - 61;
unsigned short c = a & 0xe6;
short d = !a | 0x8;
int e = (int)d + 6;
```

Assume the program will run on an **8-bit** machine and use two's complement arithmetic for signed integers. A 'short' integer is encoded in **4 bits**, while a normal '**int**' is encoded in **8 bits**. Please fill in the blanks below. (2'*6=12')

Expression	Binary Representation
a	0011 1011
b	[1]
c	[2]
d	[3]
e	[4]
b && 0x3f	[5]
a ^ 0xf	[6]

Problem 2 (20 points)

Suppose a **64-bit little-endian** machine has the following memory and register status.

Memory status

Address	Low							High
0x2000	0x19	0x20	0x00	0x00	0x00	0x00	0x00	0x00
0x2008	0x30	0x20	0x10	0x00	0x00	0x00	0x00	0x00
0x2010	0xff	0xff	0xff	0xff	0x00	0x00	0x00	0x00
0x2018	0x00	0xab	0xcd	0xef	0x00	0x00	0x00	0x00
0x2020	0x01	0x23	0x45	0x67	0x89	0xab	0xcd	0xef
0x2028	0xff	0xff	0xff	0xff	0xff	0xff	0xff	0xff

Register status

Register	Hex Value
%rax	0x00000000 00000001
%rbx	0x00000001 000000e0
%rcx	0x1fffffff fffffffe
%rdx	0x00000000 00000010
%rsi	0x00000000 00000001
%rsp	0x00000000 00002020

The following instructions are executed **sequentially**.

	Operation
1	`addq $2019, %rax`
2	`movq %rax, 0x2020(,%rcx,8)`
3	`leaq 0x5(%rbx,%rbx,2), %rsi`
4	`sarq $0x4, %rbx`
5	`pushq %rcx`

After executing five instructions above, please fill in the blanks below. For **'Hex Value'**, write in **8-byte hex value**. For example, the value on the address from `0x2000` to `0x2007` are `0x19 0x20 0x00 0x00 0x00 0x00 0x00 0x00`, the hex value should be `0x2019. (2'*10=20')`

Address	Hex Value
0x2000 ~ 0x2007	0x2019
0x2008 ~ 0x200f	[1]
0x2010 ~ 0x2017	[2]
0x2018 ~ 0x201f	[3]
0x2020 ~ 0x2027	[4]
0x2028 ~ 0x202f	[5]

Register	Hex Value
%rax	[6]
%rbx	[7]
%rcx	[8]
%rdx	0x10
%rsi	[9]
%rsp	[10]

Problem 3 (12 points)

The following figure is a 16-bit floating point representation based on the IEEE floating point format. Assume we use the IEEE round-to-even mode to do the approximation.

sign (1bit)	exp (6bits)	frac (9bits)

1. Fill the blanks with proper values. (2'*4=8')

   - Normalized: $(-1)^{sign} \times (1.fraction) \times 2^{exp-bias}$, where bias=___[1]___;
   - **+Infinite(+∞)** (in **hexadecimal** form):___[2]___;
   - Largest Positive Normalized Value (in **hexadecimal** form):___[3]___;
   - Largest Negative Denormalized Value (in **hexadecimal** form):___[4]___;

2. Consider the number $\left(\frac{2019}{256}\right)_{10}$. Please convert it into the floating point format (hexadecimal) **we designed above**. You need also provide **sign**, **exp**, and **frac** part separately in **hexadecimal** form. (4')

Problem 4 (24 points)

Please answer the following questions according to the definition of heterogeneous data structures in **x86-64**. (**NOTE**: the size of data types is shown in Figure 3.1 in ICS book.)

```
struct s {
    char *name;
    int flags;
    union u {
        void *ptr;
        short h;
    } u;
    char c;
} s;
```

1. Fill in the following blocks. (2'*8=16')

Representation	Value
sizeof(s.name)	[1]
sizeof(s.u)	[2]

Please represent address with **Hex**

&s	0x550000001040
&(s.name)	[3]
&(s.flags)	[4]
&(s.u)	[5]
&(s.u.ptr)	[6]
&(s.u.h)	[7]
&(s.c)	[8]

2. How many bytes are **WASTED** in **struct s**? Explain your solution. (4’)
3. Rearrange the above fields in **`struct s`** to conserve the most space in the memory. How many bytes are **WASTED** in **`rearranged struct s`**? Explain your solution. (4’)

Problem 5 (24 points)

One of TA wrote a simple C program and the assembly code is provided. Suppose both of them are executed on a **64-bit little-endian** machine. Please read the code and answer the following questions.

```
int transform(int n, short *xp, int *yp){
    int ret = 0;
    switch (n) {
        case 32:
            *(char *)yp = *((char *)xp + 1);
            // No break here!!!
        __[1]__:   /* Case Label */
            __[2]__;
            break;
        default:
            ret = *xp;
            break;
        __[3]__:   /* Case Label */
            *yp = *xp + 1;
            break;
        case 33: case 37:
            *((short *)yp + 1) = *xp;
            break;
    }
    return ret;
}
```

```
transform:
    pushq  %rbp
    movq   %rsp, %rbp
    movl   %edi, -20(%rbp)
    movq   %rsi, -32(%rbp)
    movq   %rdx, -40(%rbp)
    movl   $0, -4(%rbp)
    movl   -20(%rbp), %eax
    subl   $32, %eax
    cmpl   $__[4]__, %eax
    ja  .L2
    movl   %eax, %eax
    movq   __[5]__(,%rax,8), %rax
    jmp*%rax
.L3:
    movq   -32(%rbp), %rax
    movzbl 1(%rax), %edx
    movq   -40(%rbp), %rax
    movb   %dl, (%rax)
.L6:
    movq   -40(%rbp), %rax
    movl   (%rax), %eax
    movl   %eax, -4(%rbp)
    jmp.L8
```

```
.L7:
    movq   -32(%rbp), %rax
    movswl (%rax), %eax
    leal   1(%rax), %edx
    movq   -40(%rbp), %rax
    movl   %edx, (%rax)
    jmp.L8
.L5:
    movq   -40(%rbp), %rax
    leaq   2(%rax), %rdx
    movq   -32(%rbp), %rax
    movzwl (%rax), %eax
    movw   %ax, (%rdx)
    jmp __[6]__
.L2:
    movq   -32(%rbp), %rax
    movswl (%rax), %eax
    movl   %eax, -4(%rbp)
.L8:
    movl   -4(%rbp), %eax
    popq   %rbp
    ret

.section  .rodata
    .align 8
.L4:
    .quad __[7]__
    .quad .L5
    .quad .L2
    .quad .L6
    .quad .L7
    .quad __[8]__
```

1. Please fill in the blanks within C code and assembly code. (2'*8=16')

2. For inputs given below, please write down the value of **`*yp`** after the execution of **`transform`** and the function's return value (in **hexadecimal** form). (2'*4=8')

Function Inputs			After Execution	
n	*xp	*yp	*yp	return value
31	0x8877	0x12345678	0x12345678	0xffff8877
32	0x8877	0x12345678	[1]	[2]
37	0x8877	0x12345678	[3]	[4]

Problem 6 (8 points)

Given the C code and the assembly code of the function **P**, answer the questions below.

```
// x in %rdi, y in %rsi
long P(long x, long y) {
    long u = Q(y);
    long v = Q(x);
    return u + v;
}
```

```
1  P:
2    pushq    %rbp          /* Comment 1 */
3    pushq    %rbx
4    subq  $8, %rsp
5    movq  %rdi, %rbp
6    movq  %rsi, %rdi       /* Comment 2 */
7    call  Q
8    movq  %rax, %rbx       /* Comment 3 */
9    movq  %rbp, %rdi
10   call  Q
11   addq  %rbx, %rax
12   addq  $8, %rsp
13   popq  %rbx
14   popq  %rbp
15   ret
```

1. Fill in `Comment 1,2,3` to describe the purpose of the corresponding instructions. (4')

   Comment 1: Save callee saved register as it's modified in function P.

   Comment 2: [1]

   Comment 3: [2]

2. Can `%rbp` in **line 5** and **line 9** be replaced with `%r12`? Why? (2')

3. Can `%rbp` in **line 5** and **line 9** be replaced with `%r11`? Why? (2')