上 海 交 通 大 学 试 卷（ A 卷）

（ 2020 至 2021 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.

2. [1] [2]

   [3] [4]

   [5] [6]

   [7]

3.

str	output
“62”	0x2
“26”	
“84”	
“4791”	

4.

Problem 6

1. [1] [2] [3]

2.

3 [4] [5]

[6] [7]

[8] [9]

4 4.a

[10]

[11]

4.b

Problem 1 (18 points)

1. Consider the following C program

```
int a = 0x62;
int b = a << 2;
unsigned int c = a - 0x63;
short d = !a | 0;
int e = a & 9;
```

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. (3'*6=18')

Expression	Binary Representation
a	0110 0010
b	[1]
c	[2]
d	[3]
e	[4]
a ^ a	[5]
~a	[6]

Problem 2 (10 points)

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

Memory status

Address	Low							High
0x4000	0xaf	0xbe	0x00	0x00	0x00	0x00	0x00	0x00
0x4008	0x56	0x34	0x12	0x00	0x00	0x00	0x00	0x00
0x4010	0xfd	0xff	0xff	0xff	0xff	0xff	0xff	0xff
0x4018	0x12	0x23	0x34	0x45	0x56	0x67	0x78	0x89
0x4020	0xf0	0xff	0xff	0xff	0x00	0x00	0x00	0x00
0x4028	0xf0	0xe0	0xd0	0xc0	0xb0	0xa0	0x90	0x80

Register status

Register	Hex Value
%rax	0x00000000 00002020
%rbx	0x8fffffff 000000ab
%rcx	0xffffffff fffffffc
%rdx	0x00000000 00004030
%rsi	0x00000000 00000004
%rsp	0x00000000 00004028

The following instructions are executed **sequentially**.

	Operation
1	`subq  $2020, %rax`
2	`movq  %rax, (%rdx,%rcx,8)`
3	`imulq %rcx, %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 `0x4000` to `0x4007` are `0xaf 0xbe 0x00 0x00 0x00 0x00 0x00 0x00`, the hex value should be `0xbeaf`. (1'*10=10')

Address	Hex Value
0x4000 ~ 0x4007	0xbeaf
0x4008 ~ 0x400f	[1]
0x4010 ~ 0x4017	[2]
0x4018 ~ 0x401f	[3]
0x4020 ~ 0x4027	[4]
0x4028 ~ 0x402f	[5]

Register	Hex Value
%rax	[6]
%rbx	[7]
%rcx	[8]
%rdx	0x4030
%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 (8bits)	frac (7bits)

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 Negative Normalized Value (in **hexadecimal** form):___[3]___;
- Smallest Positive Denormalized Value (in **hexadecimal** form):___[4]___;

2. Consider the number $(20.77)_{10}$. 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 (20 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.)

```
union u {
    union u *ptr;
    struct s {
        char data[2];
        int *loc;
        short x;
    } s;
} u;
```

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

Representation	**Value**
sizeof(u)	**[1]**
sizeof(u.s)	**[2]**
sizeof(u.ptr)	**[3]**

Please represent address with **Hex**

&u	**0x550000001000**
&(u.ptr)	**[4]**
&(u.s)	**[5]**
&(u.s.data)	**[6]**
&(u.s.loc)	**[7]**
&(u.s.x)	**[8]**

2. How many bytes are **WASTED** in **struct s**? Explain your solution. (2')

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. (2')

Problem 5 (25 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.

```
// ASCII(0~9):0x30~0x39
int aaa(char *str){
    int result = str[0];
    int* ptr = str;
    switch(__[1]__){
        case '2':
            result = __[2]__;
            break;
        case '7':
            result = (*ptr) >> 1;
        case '5':
            result = __[3]__;
            break;
        case __[4]__:
            result = 9;
        default:
            result = result * 2 - 1;
    }
    return result;
}
int main(){
    char *str = "62";
    printf("0x%x\n", aaa(str));
    return 0;
}
```

```
1:    .section .text
2:    <aaa>:
3:        pushq   %rbp
4:        movq    %rsp, %rbp
5:        movsbl  (%rdi), %eax
6:        movl    %eax, -4(%rbp)//result
7:        movsbl  1(%rdi), %eax
8:        subl    ___[immediate]___, %eax
9:        cmpl    $6, %eax
10:       ja      .L3
11:       movq    ___[5]___, %rax
12:       jmp     *%rax
13:   .L4:
14:       cmpl    $0x35, -4(%rbp)
15:       jg      .L10
16:       movl    -4(%rbp), %eax
17:       jmp     .L11
18:   .L10:
19:       movl    $2, %eax
20:   .L11:
21:       movl    %eax, -4(%rbp)
22:       jmp     .L2
23:   .L7:
24:       movl    (%rdi), %eax
25:       sarl    %eax
26:       movl    %eax, -4(%rbp)
27:   .L8:
28:       movl    -4(%rbp), %eax
29:       notl    %eax
30:       movl    %eax, -4(%rbp)
31:       jmp     .L2
32:   .L9:
33:       movl    $9, -4(%rbp)
34:       ___[6]___
35:   .L3:
36:       movl    -4(%rbp), %eax
37:       addl    %eax, %eax
38:       subl    $1, %eax
39:       movl    %eax, -4(%rbp)
40:   .L2:
41:       movl    -4(%rbp), %eax
42:       popq    %rbp
43:       ret
44:
45:   .section .rodata
46:       .align 8
47:   .L5:
48:       .quad .L4
49:       .quad .L3
50:       .quad .L9
51:       .quad .L8
52:       .quad .L3
53:       .quad ___[7]___
```

1. Given that the minimum case number is '2', what immediate number should be filled in Line 8 of the assembly code?(2')

2. Please fill in rest of the blanks within C code and assembly code. **If you think nothing is required to write, please write NONE**. (2'*7=14')

3. If we change the value of 'str' in 'main', what is the output of this program?(1*3=3')

str	output
"62"	0x2
"26"	[1]
"84"	[2]
"4791"	[3]

4. Please explain the advantage and limitation of using "Jump Table" to implement 'switch' statement, and provide a simple code which is not suitable to be translated into a "Jump Table" (6').

Problem 6 (15 points)

One of the TAs wrote a strange function **bar**. Answer the questions below.

```c
void bar(long a, long b, long c,
         long d, long e, char f[],char g) {
  if (a <= 0) {
    printf("&g is: %p, g is: %d\n", &g, g);
    return;
  }
  bar(a-1, b-2, 0, 0, 0, 0, g);
  bar(b-1, a-1, 0, 0, 0, 0, g + 1);
}
```

```c
int main() {
  bar(2, 3, 0, 0, 0, 0, 0);
}
```

```
 <bar>:
1    pushq    %r14
2    pushq    %rbx
3    subq     $8, %rsp
4    movb     32(%rsp), %al
5    testq    %rdi, %rdi
6    jle      .L1
7    movq     %rsi, %r14
8    movq     %rdi, %rbx
9    subq     $1, %rdi
10   subq     $2, %rsi
11   movb     %al, (%rsp)
12   callq    bar
13   subq     $1, %r14
14   subq     $1, %rbx
15   movb     32(%rsp), %al
16   addb     $1, %al
17   movb     %al, (%rsp)
18   movq     %r14, %rdi
19   movq     %rbx, %rsi
20   callq    bar
21   addq     $8, %rsp
22   popq     %rbx
23   popq     %r14
24   retq
 .L1:
25   movsbl   %al, %edx
26   leaq     32(%rsp), %rsi
27   movl     $.L.str, %edi
28   xorl     %eax, %eax
29   callq    printf
// ...... code for return
 <main>:
34   subq     $8, %rsp
35   movl     $0, (%rsp)
36   movl     $2, %edi
37   movl     $3, %esi
38   callq    bar
// ...... code for return
```

NOTE: For **ALL** the following questions, assume **BEFORE** the execution of instruction at **line 34** "`subq $8, %rsp`", the value of the register `%rsp` is `0x7fffffffde68`.

1. Here are **3 options** to describe the purpose of instructions:

   A. Passing arguments to functions

   B. Save callee-saved registers

   C. Save caller-saved registers

   Choose an option for each of the instructions below: (2'*3 = 6')

   Line 1 "**pushq %r14**": [1]

   Line 7 "**movq %rsi, %r14**": [2]

   Line 17 "**movq %al, (%rsp)**": [3]

2. Can we remove **line 15** "**movb 32(%rsp), %al**"? Why? (3')

3. The program outputs multiple lines, each of which contains the address of **g**("**&g is %p:**") and the value of g("**g is %d:** "). Fill the table to show the output of the program. (0.5'*6 = 3')

"&g is: %p"	"g is: %d"
[4]	0
0x7fffffffde20	[5]
[6]	1
0x7fffffffde00	[7]
[8]	[9]

4. Jack observes that before execution of instruction at **line 20** "**callq bar**", the current stack frame is **NOT** needed any more. Then he figures out that the "**callq bar**" at **line 20** can be removed and "**retq**" at **line 24** can be replaced by a simple "**jmp**" instruction while the output value for "**g is:%d**" will remain **UNCHANGED**. (3')

   a. The table below details how he modifies the assembly (Note: **nop** is an instruction that does nothing). Fill in the table.

Line before modification	Line after modification
24: retq	24: jmp bar
20: callq bar	20: nop
[10]	[11]

   b. Show the output of the program after the modification of assembly.