1. MIPS32 编程，并在 SPIM 上调试、运行通过。

实现以下功能——

```
/* Recursive popcount */
int pcount_r(unsigned int x) {
   if (x == 0)
      return 0;
   else
      return (x & 1)
               + pcount_r(x >> 1);
}
```

要求：参照我们上课 ppt 中的"阶乘"完成，即采用与其相同的的传参/返回值方式，并建立栈帧、保存相关寄存器（可以不打开 branch delay slot）。

附上源代码+详细注释，以及运行顺利输出的截图，包括输入数字的打印以及 popcount 的计数输出。请详细些，因为主要直接依据你提供的上述内容来判分。

2. 补全下列 MIPS32 汇编代码：

lw $t6, 65536($sp)经过MIPS汇编器处理后，产生的代码如下，请补全。

lui $1,_____

addu $1, $1, _____

lw $t6, 0(_____)

li $6, 0x563478 经过MIPS汇编器处理后，产生的代码如下，请补全

lui $1, _____

_____ $6, $1, _____

（下面两题帮助练习下 x86-32 反汇编）

3. 有如下的 C 代码与对应的汇编（linux x86-32）。strcpy(char* dst, char* src)的功能是将地址 src 指向的字符串（包括结尾的'\0'）拷贝给 dst 指向的地址。

```
/* copy string x to buf */
void foo(char *x) {
  int buf[1];
  strcpy((char *)buf, x);
}
void callfoo() {
  foo("abcdefghi");
}
```

Character	Hex value	Character	Hex value
'a'	0x61	'f'	0x66
'b'	0x62	'g'	0x67
'c'	0x63	'h'	0x68
'd'	0x64	'i'	0x69
'e'	0x65	'\0'	0x00

```
080484f4 <foo>:
080484f4: 55                pushl %ebp
080484f5: 89 e5             movl %esp,%ebp
080484f7: 83 ec 18          subl $0x18,%esp
080484fa: 8b 45 08          movl 0x8(%ebp),%eax
080484fd: 83 c4 f8          addl $0xfffffff8,%esp
08048500: 50                pushl %eax
08048501: 8d 45 fc          leal 0xfffffffc(%ebp),%eax
08048504: 50                pushl %eax
08048505: e8 ba fe ff ff    call 80483c4 <strcpy>
0804850a: 89 ec             movl %ebp,%esp
0804850c: 5d                popl %ebp
0804850d: c3                ret
08048510 <callfoo>:
08048510: 55                pushl %ebp
08048511: 89 e5             movl %esp,%ebp
08048513: 83 ec 08          subl $0x8,%esp
08048516: 83 c4 f4          addl $0xfffffff4,%esp
08048519: 68 9c 85 04 08    pushl $0x804859c # push string address
0804851e: e8 d1 ff ff ff    call 80484f4 <foo>
08048523: 89 ec             movl %ebp,%esp
08048525: 5d                popl %ebp
08048526: c3                ret
```

（1） 就在 strcpy 调用完成返回到 foo 后（但在执行 0804850a 这一地址的指令之前），请列出下列以下内存中的存储内容（以 16 进制形式）

buf[0] = 0x_____

buf[1] = 0x_____

buf[2] = 0x_____

（2） 就在 ret 指令(地址 0x0804850d)被执行之前，%ebp 的内容是？

（3） 在 ret 指令(地址 0x0804850d)被执行之后，%eip 的内容是？

4. 有如下的递归程序以及编译出来的汇编程序（linux x86-32架构）。

```
int silly(int n, int *p)
{
int val, val2;
if (n > 0)
    val2 = silly(n << 1, &val);
else
    val = val2 = 0;
*p = val + val2 + n;
return val + val2;
}
```

```
silly:
  pushl %ebp
  movl %esp,%ebp
  subl $20,%esp
  pushl %ebx
  movl 8(%ebp),%ebx
  testl %ebx,%ebx
  jle .L3
  addl $-8,%esp
  leal -4(%ebp),%eax
  pushl %eax
  leal (%ebx,%ebx),%eax
  pushl %eax
  call silly
  jmp .L4
.L3:
  xorl %eax,%eax
  movl %eax,-4(%ebp)
.L4:
  movl -4(%ebp),%edx
  addl %eax,%edx
  movl 12(%ebp),%eax
  addl %edx,%ebx
  movl %ebx,(%eax)
  movl -24(%ebp),%ebx
  movl %edx,%eax
  movl %ebp,%esp
  popl %ebp
  ret
```

1）变量val是否存储在栈上？如是，其存储位置相对于%ebp的offset是多少，并给出其存储于栈上的理由。

2）val2是否存储在栈上？如是，其存储位置相对于%ebp的offset是多少，并给出其存储于栈上的理由。

3）什么变量或者参数存储于-24(%ebp)这个位置，为何？

4）如果有的话，什么变量或者参数存储于-8(%ebp)这个位置，为何？