1. (1) chitao:

```
cmpl   $10, %edi
je     .L3
subq   $8, %rsp
addl   $1, %edi
call   chitao
leal   2(%rax, %rax), %eax
jmp    .L2
.L3:
movl   $1, %eax
ret
.L2:
addq   $8, %rsp
ret
```

1(2) 加入无关语句 避免当n未计算好时跳转错误，影响效率

2.

变量	十进制 offset
a	a[0], a[1]直接记为之后数，栈上无a
a[2]	同上.
x	保存在 %rdi，后移到 %ebx
buf	保存在 %rdi 中，后移到 %rcx
buf[3]	-29
%rbx 保存值	-8

3. ① $15213
② swap(&zip1, &zip2)
③ 91125
④ %rdi

4. 1. 返回地址为GET函数的返回地址，值为 1.
2. #(A), eax 存放输入参数，当前栈信息的内存地址.
#(B), ecx存放 GET函数的 Rtn address
#(C), 计算栈顶的上一位地址（用于退栈时复原）
#(E) 将 GET函数的 Rtn address 压栈，令SET返回地址为GET的返回地址.
#(D) movl 72(%eax), %esp.

5(1) 先把当前协程（%rsi）的寄存器状态保存，再调用另一个协程（%rdi）的寄存器状态.

(2) 这几个寄存器是 被调用者负责保存的。

6. M=13, N=7

7.
```
int i;
int x=0;
for (i=0; i<n; i++) {
    if (a[i] > x+1) x= 2*a[i];
    else x++;
}
return x;
```

8. 先在 function1 中开辟新的空间（subq $32, %rsp）
再把这个指针（%rsp）放入第一个参数（%esi）
即 call return_struct 时有两个参数（%rsi, %rdi，分别对应栈顶地址和 i）

在函数 return_struct中，直接将struct的数据放入对应栈位置上，返回值为栈指针（即传入的第一个参数）

简单而言，就是函数开辟一段栈空间，传入指针，函数用这个指针直接操作栈，在原函数中便可得到结构体的值.

9.

+36	i	(d)	
+32	2i	(c)	
+28	i	(b)	
+24	i	(a)	
+20	\		
+16	i	(e)	
+12	i	(d)	
+8	2i	(c)	
+4	i	(b)	
%rsp	i	(a)	
	Rtn Address		
旧地址			

+24
+20
+16
+12
+8
%rsp
新地址

(1) 调用call input_struct 前，栈如左图所示，并压入 Rtn Address

此时，在 input_struct 中访问 age 和 eee 是按照新地址访问，

故 C 传入 struct 参数时，约定原%rsp 指向栈顶，然后传入%rsp 进行操作。

(2) 函数在同一个文件中（如 function2），编译器直接计算结果，不再进入函数。

若不在同文件中，此时 input_struct 默认 %rsp+8 为 struct 首地址，

故直接调用 24(%rsp)，8(%rsp) 取值。

(3) 加入 static 后，说明 input_struct 仅供同一文件的函数调用，故不需要考虑不在同一文件的情况，直接计算值便可。

10. A. f[1]

B. b.i → f[3]

C. i → e / h → b.j

D. i → g → d → a[1]