练习1.3

首先计算字符串的长度
然后对字符串进行统一的初始化操作

练习1.4

(1)

```
call xxx
pop eax
push eax
ret
```

为什么不能用mov实现，因为eip是特殊寄存器，mov指令不能操作其数值

(2)

```
push AABBCCDDh
ret

jmp AABB CCDDh
```

(3)

程序会转到此时ESP所指地址空间继续执行

(4)

小于等于4字节时，放入eax
5~8字节，edx：eax
大于8字节 用栈来临时存储变量来返回

练习1.6

跟着后面的讲解一步步走即可，还算比较容易理解

练习1.7

(1)(2)

自己在看一遍画个栈图即可

(3)

c编译器特殊规则
对于cedel方式 _funcname
对于stdcall方式 _funcname@number (number) 是参数的字节大小
对于fastcall方式 @funcname@number

(4)

比较好做也比较好理解
https://ayesawyer.github.io/2019/02/27/%E8%AF%BE%E5%90%8E%E7%BB%83%E4%B9%A0%E7%AC%AC%E4%B8%80%E7
这里附上一些指令的作用
rep: ecx大于0 重复指令操作
repe: ecx大于0 zf=1 重复指令操作
repne: ecx大于0 zf=0 重复指令操作
stosd 将eax的值放入edi所指向的内存中，edi+=4
scasd 将eax的值和edi所指向的内存的值比较，比较完edi+=4

(5)

因为不熟悉windbg调试，暂且搁置
http://blog.sina.com.cn/s/blog_e3154f150102wv7l.html

(6)(7)(8)

因为找不到H的样例 暂且搁置

(9)

```
_BYTE *__cdecl sub_1000CEA0(const char *a1, unsigned __int8 a2)
{
  unsigned int v2; // ecx
  const char *v3; // edi
  bool v4; // zf
  _BYTE *v5; // edi
  _BYTE *result; // eax

  v2 = strlen(a1) + 1;
  v3 = &a1[v2 - 1];
  do
  {
    if ( !v2 )
      break;
    v4 = *v3-- == a2;
    --v2;
  }
  while ( !v4 );
  v5 = v3 + 1;
  if ( *v5 == a2 )
    result = v5;
  else
    result = 0;
  return result;
}
```

比较好懂

(10)

cpl由cpu控制，用户代码无法修改

(11)

没有windbg调试过，暂且搁置

(12)

```
.a
oot@ubuntu:/home/wsxk/Desktop/BeaEngine/lib/Linux.gnu.Debug# ./tcc
ush rbp
ov rbp, rsp
ub rsp, 0000000000000970h
ov rax, qword ptr fs:[00000028h]
ov qword ptr [rbp-08h], rax
or eax, eax
ov dword ptr [rbp-00000968h], 00000000h
ea rax, qword ptr [rbp-00000960h]
ov edx, 00000952h
ov esi, 00000000h
ov rdi, rax
all 004005B0h
ov eax, 004006F6h
ov qword ptr [rbp-00000960h], rax
ov eax, dword ptr [rbp-000000C0h]
est eax, eax
ne 004007A7h
mp dword ptr [rbp-00000968h], 63h
nle 004007A7h
ea rax, qword ptr [rbp-00000960h]
ov rdi, rax
all 004556E2h
ov dword ptr [rbp-00000964h], eax
ov eax, dword ptr [rbp-000000C0h]
mp eax, FFFFFFFFh
e 0040073Fh
ea rax, qword ptr [rbp-00000960h]
dd rax, 14h
ov rdi, rax
```