P164-168

- 运行时栈
  - 先进后出的内存管理原则，是 c 语言过程调用机制的一个关键特性。
  - 栈帧：如上次笔记的图所示，当过程需要的存储空间超出寄存器的大小，就会在栈上分配空间，这个部分叫做栈帧 stack fram。对当前正在执行的
  - 很多函数不会调用其他函数，所有局部变量也都可以保存在寄存器中时，就不需要栈帧。
- 转移控制
  - 指令：call / callq 和 ret /retq （注释：+q 是表示是 x86-64 版本汇编代码）
- 数据传递：
  - x86-64 通过寄存器，可以传递的参数最多 6 个整数值，对于更多的参数而言，就要通过栈帧来传递，如图，第 7,8 个参数是通过栈传递，汇编代码从内存取出。

```
void proc(long  a1, long  *a1p,
          int   a2, int   *a2p,
          short a3, short *a3p,
          char  a4, char  *a4p)
{
    *a1p += a1;
    *a2p += a2;
    *a3p += a3;
    *a4p += a4;
}
```

a）C代码

```
    void proc(a1, a1p, a2, a2p, a3, a3p, a4, a4p)
    Arguments passed as follows:
      a1  in %rdi          (64 bits)
      a1p in %rsi          (64 bits)
      a2  in %edx          (32 bits)
      a2p in %rcx          (64 bits)
      a3  in %r8w          (16 bits)
      a3p in %r9           (64 bits)
      a4  at %rsp+8        ( 8 bits)
      a4p at %rsp+16       (64 bits)
1   proc:
2     movq    16(%rsp), %rax    Fetch a4p   (64 bits)
3     addq    %rdi, (%rsi)      *a1p += a1  (64 bits)
4     addl    %edx, (%rcx)      *a2p += a2  (32 bits)
5     addw    %r8w, (%r9)       *a3p += a3  (16 bits)
6     movl    8(%rsp), %edx     Fetch a4    ( 8 bits)
7     addb    %dl, (%rax)       *a4p += a4  ( 8 bits)
8     ret                       Return
```

b）生成的汇编代码