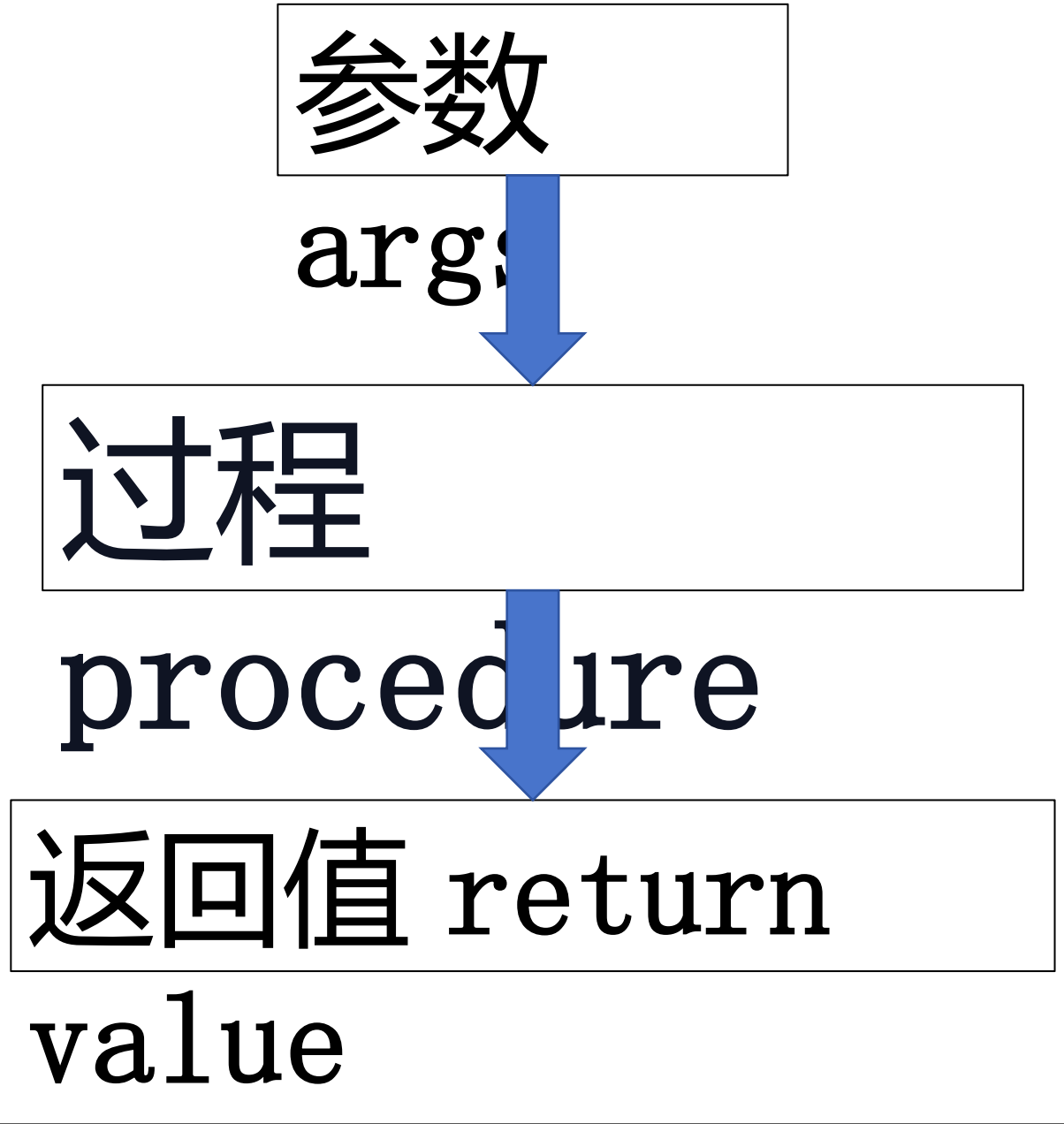
PROCEDURES

ICS, 2024/10/09, ZZJ

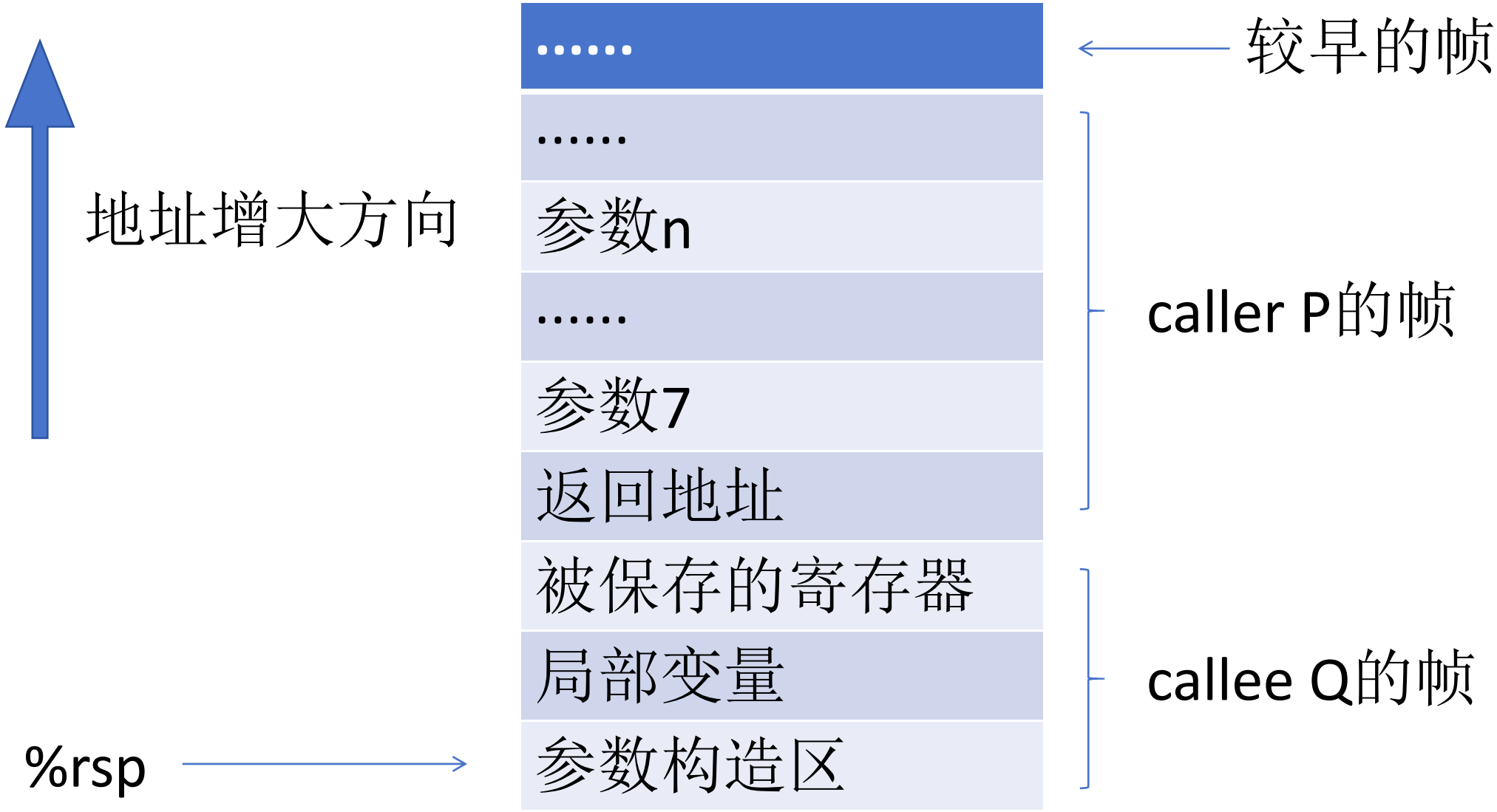

参数
args
过程
procedure
返回值 return
value

3.7 PROCEDURES

- 机制支持：
  - 转移控制
  - 传递数据
  - 分配和释放内存

都通过stack来实现

3.7.1 THE RUN-TIME STACK



3.7.2 转移控制

指令	描述
call Lable	过程调用（直接调用）
call *Operand	过程调用（间接调用）
ret	从过程调用中返回

间接调用举例

```
call *%rax
call *$0x114514
call *0x19(%rax,
```


%rip
0x400563
%rsp
0x7FFFFFFFE940
.....
callq 0x400540
%rip
0x400540
%rsp
0x7FFFFFFFE938
.....
0x400568
retq
%rip
0x400568
%rsp
0x7FFFFFFFE940
.....

3.7.3 数据传送

- 使用寄存器传送整数和指针

操作数大小	1	2	3	4	5	6
64bit	%rdi	%rsi	%rdx	%rcx	%r8	%r9
32bit	%edi	%esi	%edx	%ecx	%r8d	%r9d
16bit	%di	%si	%dx	%cx	%r8w	%r9w
8bit	%dil	%sil	%dl	%cl	%r8b	%r9b

3.7.3 数据传送

- 如果一个函数有大于6个整型参数，超出6个的部分就要通过栈来传递。
- **通过栈传递参数时，所有的数据大小都向8的倍数对齐。**

```
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;
}
```

```
proc:
   movq 16(%rsp), %rax
   addq %rdi, (%rsi)
   addl %edx, (%rcx)
   addw %r8w, (%r9)
   movl 8(%rsp), %edx
   addb %dl, (%rax)
   ret
```

3.7.4 栈上的局部存储

- 局部数据必须放在内存中的情况：
  - 寄存器不足够存放所有的本地数据
  - 对一个局部变量使用地址运算符'&'，必须要把它存放在内存上以产生一个地址
  - 某些局部变量是数组或结构
  - ……

3.7.5 寄存器中的局部存储空间

- 根据**惯例**：
- 被调用者（callee）保存寄存器
  - %rbx, %rbp, %r12~%r15
- 调用者（caller）保存寄存器
  - 其他除了%rsp以外的所有寄存器

3.7.6 递归过程

```
long rfact(long n)
{
  long result;
  if(n <= 1)
    result = 1;
  else
    result = n *
  rfact(n - 1);
  return result;
}
```

```
rfact:
   pushq %rbx
   movq %rdi, %rbx
   movl $1, %eax
   cmpq $1, %rdi
   jle .L35
   leaq -1(%rdi),
%rdi
   call rfact
   imulq %rbx, %rax
.L35:
   popq %rbx
   ret
```

THANKS!