请求分配大小	实际分配大小	分析与说明
96 字节	128B	96B+metaData（32B）=128B，被IS_SMALL认定为小内存，向上取整分配128B*1内存
128 字节	256B	128B+metaData（32B）=160B，被IS_SMALL认定为小内存，向上取整分配128B*2内存
256 字节	384B	256B+metaData（32B）=288B，被IS_SMALL认定为小内存，向上取整分配128B*3=384内存
4064 字节	4096B	4064B+32B=4096B为大内存超过819B，因此分配4096B*1
4096 字节	8192B	4096B+32B=4128B为大内存超过819B，向上取整因此分配4096B*2=8192B

96B分配流程

元数据固定32B，总申请数据为128B，判定为小内存，走uk_salloc(a, num_pages)分支

用GDB调试结果

```
uk_malloc_ifpages (a=0x40010000, size=96) at /home/hzfu/code/OS-2026/app-helloworld/workdir/unikraft/lib/ukalloc/alloc.c:169
169         __sz realsize = METADATA_IFPAGES_SIZE_POW2 + size;
(gdb) s
171         UK_ASSERT(a);
(gdb) s
173         if (!size || realsize < size) return __NULL;
(gdb) s
178         if (IS_SMALL(realsize)) {
(gdb) s
179           num_pages = size_to_s_num_pages(realsize);
```

问题一：

最小分配单元: Unikraft 两种内存分配策略的最小单元是多少？它是如何定义的?

palloc最小单元：4KB

```
#define PAGE_SIZE                0x1000UL
```

salloc最小单元：128 B

```
50    #define __S_PAGE_SHIFT 7
51    #define __S_PAGE_SIZE (1ULL << __S_PAGE_SHIFT)
```

问题二：

分配器选择: `uk_malloc()` 函数在何种条件下会选择 `palloc`，又在何种条件下会选择`salloc`？

在alloc.c函数中显示，他会用is_small来判断是否是大内存分配，如果满足is_small条件就会进入salloc，反之进入palloc。

```
if (IS_SMALL(realsize)) {
    num_pages = size_to_s_num_pages(realsize);
    intptr = (__uptr)uk_salloc(a, num_pages);
    uk_pr_err("alloc size => %llu, num_pages => %ll
              realsize, num_pages, intptr);
} else {
    num_pages = size_to_num_pages(realsize);
```

而is_small的具体定义在IS_SMALL宏定义中，判断申请内存（malloc函数填写的内存与32元数据之和）是否小于4096B/5，满足则为true。

```
#define IS_SMALL(size) ((size) < (__PAGE_SIZE / 5))
```

问题三：

大内存分配问题: 当前 `palloc` 在处理大内存（例如，一次性分配多个页面）的分配与回收时，存在一个已知的设计问题。请定位该问题，并尝试在 GDB 中通过 `set` 命令修改相关变量，模拟正确的 `free` 过程，并截图记录结果。

palloc free的过程中，完全依赖于调用者传入的 num_pages 参数来计算 order 和要释放的内存块大小。

并且free() 调用链没有传递 small 参数，导致其成为垃圾值，进而使 uk_get_metadata 走错分支，最终给 bbuddy_pfree 传递了错误的 num_pages。

```
512         size_t order = (size_t)num_pages_to_order(num_pages);
```

因此使用gdb来用set改动small的值。

0. gdb运行到main开始的地方。

1. 在使用small之前打好断点。

```
1  b uk_free_ifpages
```

```
Breakpoint 2, uk_free_ifpages (a=0x40010000, ptr=0x44101aa
0, small=0x4011ad2c <uk_free_ifpages>) at /home/hzfu/code/
OS-2026/app-helloworld/workdir/unikraft/lib/ukalloc/alloc.
c:209
209         UK_ASSERT(a);
```

2. 继续运行

```
Breakpoint 2, uk_free_ifpages (a=0x400100
00, ptr=0x44101aa0, small=0x4011ad2c <uk_
free_ifpages>) at /home/hzfu/code/OS-2026
/app-helloworld/workdir/unikraft/lib/ukal
loc/alloc.c:209
209         UK_ASSERT(a);
(gdb)
```

3. 查看small的值

```
(gdb)  p small
$1 = (const void *) 0x4011ad2c <uk_free_i
fpages>
```

这里可以发现small是一个未定义的值

4. 改正错误，用set将small修改为0

```
(gdb) set var small = 0
```

5. 运行成功，输出正确信息。a b

```
SUCCESS: Memory freed correctly after GDB intervention!
[    5.125128] ERR:  [libukallocbbuddy] <bbuddy.c @
 320> Free list 8 is empty.
[    5.125870] ERR:  [libukallocbbuddy] <bbuddy.c @
 322> Free list 9 (Order 9):

   - SUCCESS: Memory freed correctly after GDB inter
vention!

================================================
Analysis finished. The system will now halt.
```