bootmem

- 辅助函数
  - bootmem_debug_setup
    - 设置全局bootmem_debug=1
  - bootmap_bytes
    - 根据传入参数pages向上取8的整数对齐long
    - 也就是8的整数且在64位环境中8字节对齐
  - bootmem_bootmap_pages
    - 1 调用bootmap_bytes计算出所需字节数量
    - 2 计算出所占用的物理页面数量
  - link_bootmem
    - 1 将传入的bdata数据按照起始页帧号排序加入bdata_list链表
  - align_idx
    - 1 获取第一个参数bdata对应的node_min_pfn作为base，然后向上step取整，减去base获取偏移
  - find_next_zero_bit
    - 在第一个参数以第三个参数为偏移查找比特位为0且不大于第二个参数的位置
- 核心函数
  - init_bootmem_core
    - 1 根据入参初始化bdata
    - 2 调用link_bootmem添加进全局链表
    - 3 调用bootmap_bytes 计算所占bitmap字节，并且将对应的map初始化为1
  - init_bootmem_node
    - 1 对 init_bootmem_core 封装
  - init_bootmem
    - 1 初始化全局变量max_low_pfn 、min_low_pfn
    - 2 调用init_bootmem_core创建并且初始化bdatat
  - free_bootmem_late
    - 1 单页释放physaddr起始地址size大小的物理页最终调用free_pages加入到伙伴系统里
  - free_all_bootmem_core
    - 释放整个bdata，但是是以32个页面为单位释放
  - reset_node_managed_pages
    - 遍历pgdat->node_zone,reset zone->managed_pages=0
  - reset_all_zones_managed_pages
    - 遍历所有的pgdat的zone初始化zone_managed=0
    - 初始化全局变量reset_managed_pages_done为0
  - free_all_bootmem
    - 遍历所有的pgdata，调用free_all_bootmem_core释放所有的bdata
  - __free
    - 1 根据入参 标记对应地址的bitmap为0，也就是bootmem释放函数
  - __reserve
    - 1 根据入参设置对应地址的bitmap为1，也就是bootmem的分配函数
    - 2 如果指定的地址被分配，且设置了BOOTMEM_EXCLUSIVE标志则直接释放对应的地址并且返回ebusy
  - mark_bootmem_node
    - 1 根据传入的bdata以及reserve标志，调用对应的_reserve去分配对应的bootmem或者调用_free去释放对应的bootmem
  - mark_bootmem
    - 1 遍历Bdata_list 包含所有的bdata，调用mark_bootmem_node去分配或者释放mem
  - alloc_bootmem_bdata
    - 1 根据入参goal求取start起始地址
    - 2 获取start对应bdata管理物理内存的偏移
    - 3 根据传入的limit计算出max，然后获取max对应于bdatta起始地址的偏移
    - 4 调用 find_next_zero_bit查询大于sidx的起始地址作为sidx
    - 5 对齐sidx以及初始化eidx位sidx+size
    - 6 在bdata查找连续的如果没有连续的物理页回到第4步重新查找sidx
    - 7 根据新的end_off初始化bdata的last_end_off和hint_idx
    - 8 调用__reserve 分配对应的物理mem，转为位virt返回给调用者
- 封装函数
  - free_bootmem_node
    - 根据入参的物理地址以及pgdat使用reserve=0标志去调用mark_bootmem_node去释放对应的mem
  - free_bootmem
    - 根据传入的物理地址调用mark_bootmem去遍历所有的bdata释放对应的mem
  - reserve_bootmem_node
    - 分配对应的物理地址的mem
    - mark_bootmem_node(pgdat->bdata, start, end, 1, flags)
  - reserve_bootmem
    - 根据对应的物理地址以及size大小分配mem
    - mark_bootmem(start, end, 1, flags)
  - alloc_bootmem_core
    - 遍历所有的bdata,根据传入的goal查找
    - 调用alloc_bootmem_bdata查找
  - ___alloc_bootmem_nopanic
    - 调用alloc_bootmem_core查找，查找不到设置goal为0继续查找
  - __alloc_bootmem_nopanic
    - 设置limit为0 调用___alloc_bootmem_nopanic查找
  - ___alloc_bootmem
    - 调用___alloc_bootmem_nopanic查找，找到返回mem，找不到直接panic
  - __alloc_bootmem
    - limit为0，调用___alloc_bootmem
  - ___alloc_bootmem_node_nopanic
    - 设置limit为0，调用alloc_bootmem_bdata、alloc_bootmem_core，找不到设置goal为0 再一次调用
  - __alloc_bootmem_node_nopanic
    - 直接调用___alloc_bootmem_node_nopanic
  - ___alloc_bootmem_node
    - 调用___alloc_bootmem_node_nopanic，找不到panic
  - __alloc_bootmem_node
    - 先判断slab是否可用，不可用___alloc_bootmem_node
  - __alloc_bootmem_node_high
    - __alloc_bootmem_node
  - __alloc_bootmem_low
    - ___alloc_bootmem(size, align, goal, ARCH_LOW_ADDRESS_LIMIT)
  - __alloc_bootmem_low_nopanic
    - ___alloc_bootmem_nopanic(size, align, goal,ARCH_LOW_ADDRESS_LIMIT)
  - __alloc_bootmem_low_node
    - slab可用优先使用，不可用调用___alloc_bootmem_node(pgdat, size, align, goal, ARCH_LOW_ADDRESS_LIMIT);

试用版 XMind:ZEN 试用版