```
[    0.000000] Zone ranges:
[    0.000000]   DMA32    [mem 0x0000000080200000-0x00000000ffffffff]
[    0.000000]   Normal   [mem 0x0000000100000000-0x000000047fffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x000000047fffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x000000047fffffff]
[    0.000000] On node 0, zone DMA32: 512 pages in unavailable ranges
```

函数 free_area_init

```
unsigned long start_pfn, end_pfn;
int i, nid, zone;
bool descending;

/* Record where the zone boundaries are */
memset(arch_zone_lowest_possible_pfn, 0,
              sizeof(arch_zone_lowest_possible_pfn));
memset(arch_zone_highest_possible_pfn, 0,
              sizeof(arch_zone_highest_possible_pfn));

start_pfn = PHYS_PFN(memblock_start_of_DRAM());
descending = arch_has_descending_max_zone_pfns();

for (i = 0; i < MAX_NR_ZONES; i++) {
    if (descending)
        zone = MAX_NR_ZONES - i - 1;
    else
        zone = i;

    if (zone == ZONE_MOVABLE)
        continue;

    end_pfn = max(max_zone_pfn[zone], start_pfn);
    arch_zone_lowest_possible_pfn[zone] = start_pfn;
    arch_zone_highest_possible_pfn[zone] = end_pfn;

    start_pfn = end_pfn;
}

/* Find the PFNs that ZONE_MOVABLE begins at in each node */
memset(zone_movable_pfn, 0, sizeof(zone_movable_pfn));
find_zone_movable_pfns_for_nodes();

/* Print out the zone ranges */
pr_info("Zone ranges:\n");
```

```
ffffffff80410f60 <free_area_init>:
ffffffff80410f60:>711d                > addi> sp,sp,-96
ffffffff80410f62:>ec86                > sd> ra,88(sp)
ffffffff80410f64:>e8a2                > sd> s0,80(sp)
ffffffff80410f66:>e4a6                > sd> s1,72(sp)
ffffffff80410f68:>1080                > addi> s0,sp,96
ffffffff80410f6a:>e0ca                > sd> s2,64(sp)
ffffffff80410f6c:>fc4e                > sd> s3,56(sp)
ffffffff80410f6e:>f852                > sd> s4,48(sp)
ffffffff80410f70:>f456                > sd> s5,40(sp)
ffffffff80410f72:>f05a                > sd> s6,32(sp)
ffffffff80410f74:>43823783          > ld> a5,1080(tp) # 438 <PECOFF_FILE_ALIGNMENT+0x238>
ffffffff80410f78:>faf43c23          > sd> a5,-72(s0)
ffffffff80410f7c:>4781                > li> a5,0
ffffffff80410f7e:>001f8797          > auipc>a5,0x1f8
```

对应上面的for循环， 其他函数都是已经标示出来的

```
ffffffff80410fe2:>57a80813          > addi> a6,a6,1402 # ffffffff80609558 <required_kernelcore_percent>
ffffffff80410fe6:>458d                > li> a1,3
ffffffff80410fe8:>87ba                > mv> a5,a4
ffffffff80410fea:>c119                > beqz> a0,ffffffff80410ff0 <free_area_init+0x90> # if
ffffffff80410fec:>40e307bb          > subw> a5,t1,a4
ffffffff80410ff0:>03178263          > beq>a5,a7,ffffffff80411014 <free_area_init+0xb4> # if
ffffffff80410ff4:>078e                > slli> a5,a5,0x3
ffffffff80410ff6:>00f486b3          > add>a3,s1,a5
ffffffff80410ffa:>fa843603          > ld> a2,-88(s0)
ffffffff80410ffe:>6294                > ld> a3,0(a3)
ffffffff80411000:>00c6f363          > bgeu> a3,a2,ffffffff80411006 <free_area_init+0xa6> # max
ffffffff80411004:>86b2                > mv> a3,a2
ffffffff80411006:>97c2                > add>a5,a5,a6
ffffffff80411008:>fad43823          > sd> a3,-80(s0)
ffffffff8041100c:>e3b0                > sd> a2,64(a5)
ffffffff8041100e:>f794                > sd> a3,40(a5)
ffffffff80411010:>fad43423          > sd> a3,-88(s0)
ffffffff80411014:>2705                > addiw>a4,a4,1
ffffffff80411016:>fcb719e3          > bne>a4,a1,ffffffff80410fe8 <free_area_init+0x88> # for
```

再后面就是printk 了， 没有成功输出