1.从 RAM 启动

（1）首先需要保证 boot 为 RAM 启动

（2）修改 ld 文件，使得编译器编译的代码位置为 RAM，例如下图，我们直接修改了 flash 的起始地址。

```
MEMORY
{
    FLASH (rx) : ORIGIN = 0x20000000, LENGTH = 10K
    RAM (xrw) : ORIGIN = 0x20002800, LENGTH = 10K
}
```

（3）由于上电时内核的 PC 指针处于 0，由于指令中会有利用当前 PC 取址的操作（auipc），所以需要在启动时就把 PC 偏移到正确位置。Ram 启动下，内核上电的前八个字节指令不是通过存储器访问得到，而是从内部的寄存器得到，寄存器的值时由硬件从 ram 搬运过来。

所以只要保证 ram 中代码的前两条能够将 PC 偏移过去即可，修改启动文件的前两条指令：

```
    .section    .init,"ax",@progbits
    .global _start
    .align  1
_start:
    lui  t0,0x20000
    jr  8(t0)

    j   handle_reset
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00000013
    .word 0x00100073
    .section    .vector,"ax",@progbits
    .align  1
_vector_base:
```

t0=0x20000000

跳转至 t0+0x8处，由于前两条空间被占用，handle_reset的位置被偏移至20000008，该值可由list文件查看。

（4）RAM 启动的代码必须从 ram 的开头位置开始。

2.从 flash 启动

（1）内核启动始终从 0 地址取指，硬件做了 0 到 0x08000000 的映射，所以 ld 文件中使用虚拟地址 0 是没有问题的，且上电也不需要强制偏移 pc 指针。

```
MEMORY
{
    FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 288K
    RAM (xrw) : ORIGIN = 0x20000000, LENGTH = 32K
}
```

（2）如果需要 ld 文件中为实际物理地址，和 ram 中启动类似，由于内核启动的 PC 值为 0，

需要进行类似的偏移。

a.修改 ld 文件 flash 起始地址为 0x08000000

b.启动文件最前面增加两行代码

lui t0,0x08000

jr 8(t0)

3.用户 IAP 相关

（1）使用 IAP 是建议 IAP 程序放到 flash 前部 4K，芯片启用读保护后，前 4K 默认写保护，对 IAP 的意外丢失是有帮助的

（2）期望从 IAP 成功跳转到用户程序，需要注意一下问题：

a.由于用户程序会从它自己的 handler_reset 开始执行，会操作一些 CSR 寄存器。所以跳转过来后要保证处于机器模式。可通过以下两个方式保证：

1）强制 IAP 程序一直为机器模式

IAP 的启动文件 mstatus 的 MPP 设置为 0b11，即或上 0x1800;

2）不修改 mstatus，IAP 从中断函数中跳转至用户程序，中断下内核进入机器模式，此时 IAP 的这个中断函数并不会执行到中断返回“mret”。

b.跳转的指令和之前的类似，只不过这里没有 8 字节大小机器码限制。只要实现跳转即可。例如需要跳转到 0x5004 的位置

```
li   t0,   0x5004      lui    t0,   0x5      lui    t0,   0x5
jr   t0                jr     4(t0)          addi   t0,   t0,   4
                                              jr     t0
```

这三种均是一个意思。

c.IAP 打开的外设、中断等，执行跳转前建议关闭。防止对后面的用户程序有干扰。

4.程序下载至 flash，但程序在 flash 和 RAM 中互相跳转

该情况下需要修改 ld 文件以及在启动文件中需要上电时将部分 ram 运行的代码从 flash 拷贝至 ram 中。此情况可以参考 CH582 例程中的 hight_code 定义

```
.highcode :
{
    . = ALIGN(4);
    PROVIDE(_highcode_vma_start = .);
    *(.vector);
    KEEP(*(SORT_NONE(.vector_handler)))
    *(.highcode);
    *(.highcode.*);
    . = ALIGN(4);
    PROVIDE(_highcode_vma_end = .);
} >RAM AT>FLASH
```

ld 中 highcode 段定义

```
/* Load highcode code  section from flash to RAM */
2:
    la a0, _highcode_lma
    la a1, _highcode_vma_start
    la a2, _highcode_vma_end
    bgeu a1, a2, 2f
1:
    lw t0, (a0)
    sw t0, (a1)
    addi a0, a0, 4
    addi a1, a1, 4
    bltu a1, a2, 1b
```

启动文件中搬运过程