IPS::LCM::SRAM0	0xD0FE8000	IPS::LCM::SRAM0
IPS::LCM::SRAM1	0xD0FF0000	IPS::LCM::SRAM1
IPS::LCM::SRAM2	0xD0FF8000	IPS::LCM::SRAM2
AP::SQU_SRAM0::SRAM	0xD1000000	AP::SQU_SRAM0::SRAM
AP::SQU_SRAM1::SRAM	0xD1008000	AP::SQU_SRAM1::SRAM
AP::SQU_SRAM2::SRAM	0xD1010000	AP::SQU_SRAM2::SRAM

Gemstone2内建了

LCM SRAM0 - 32K

LCM SRAM1 - 32K

LCM SRAM2 - 32K

SQU_SRAM0 - 32K

SQU_SRAM1 - 32K

SQU_SRAM2 - 32K

in Gemstone2 System Address Map (Table 5)

Target	From	To	Size
Reserved 1	0xD0800000	0xD0EFFFFF	7M
Reserved 2	0xD0F00000	0xD0FE7FFF	928K

在Programmer Guide上被标为reserved，但实际上被利用了！

MiniLoader运行在R4上。

BootROM把miniloader载入0xd0f08800，而NTIM_GRA2.bin载入0xd0f08000.

然后跳转到0xd0f08800执行miniloader code。

in miniloader/source/main/main_entry.c

```
    if (false == recovery_boot())
    {
        if (STATUS_OK != load_code_images(rom_type, &AP_loaded_code_launch_addr, &R4_loaded_code_launch_addr))
        {
            minPrintf("\nCode Image loading failed\n");
            assert(0);
        }
        else
        {
            checkpoint('l');  // l Loaded
            cpu_disable_interrupts();
        }

    } else
```

load_code_images()会把u-boot bin载入`AP_loaded_code_launch_addr所返回的地址(physical address)，threadx bin`载入R4_loaded_code_launch_addr `所返回的地址。`

`根据NTIM.txt，AP_loaded_code_launch_addr = 0x08000000 (128M)，`R4_loaded_code_launch_addr = 0x06000000(96M).

```
    if (valid_code_images == true)
    {
        // x Execute loaded images, DOES NOT RETURN!!
        launch_code_images(rom_type, board_type, AP_loaded_code_launch_addr, R4_loaded_code_launch_addr);
        checkpoint('x');
    } else
```

launch_code_images()分别让R4（也是现在miniloader正在上面运行的core）跳转到`R4_loaded_code_launch_addr（96M边界）去执行，让AP core 0运行载入在128M边界处的u-boot code。`

```
1.  void launch_code_images( rom_types_e rom_type, board_types_e board_type, uint32_t
     AP_loaded_code_launch_addr, uint32_t R4_loaded_code_launch_addr )
2.  {
3.      cpu_disable_interrupts();
        ①
4.
5.      if (0 != AP_loaded_code_launch_addr)
6.              load_ap_reset_vector(AP_loaded_code_launch_addr);          ②
7.
8.      force_syncronization();
9.
10.     minPrintf("\nLaunch AP Core0 @ 0x%08x\n",AP_loaded_code_launch_addr);
11.
12.         launch_ap_core0(rom_type, board_type);                                ③
13.
14.         launch_lp_core( R4_loaded_code_launch_addr );                    ④
15. }
```

①

这里disable R4 core上的interrupt.

②

static void load_ap_reset_vector(uint32_t AP_loaded_code_launch_addr)

{

    uint32_t   *code_ptr = 0x0;

// Setup reset vector on AP complex to launch to the loaded code.

    *code_ptr++ = 0xea000006;   // Reset Vector - branch to 0x20

    *code_ptr++ = 0xeafffffe;   // branch to pc-0x8

    *code_ptr++ = 0xeafffffe;   // branch to pc-0x8

    *code_ptr++ = 0xeafffffe;   // branch to pc-0x8

    *code_ptr++ = 0xeafffffe;   // branch to pc-0x8

    *code_ptr++ = 0xeafffffe;   // branch to pc-0x8

```
    *code_ptr++ = 0xeaffffffe;       // branch to pc-0x8

    *code_ptr++ = 0xeaffffffe;       // branch to pc-0x8

    *code_ptr++ = 0xe1a00000;        // nop

    *code_ptr++ = 0xe1a00000;        // nop

    *code_ptr++ = 0xe1a00000;        // nop

    *code_ptr++ = 0xe1a00000;        // nop

    *code_ptr++ = 0xe1a00000;        // nop


// Swap the next two lines commenting to either continue or loop in place till debug attach

    *code_ptr++ = 0xe1a00000;        // nop

    //*code_ptr++ = 0xeaffffffa;       // branch to pc-0x20


    *code_ptr++ = 0xe59f0000;        // ldr r0, [pc,#8]

    *code_ptr++ = 0xe12fff10;        // bx r0

    *code_ptr++ = AP_loaded_code_launch_addr;  // Load address of AP code loaded from FLASH.

}
```

直接填写zero address的exception vector table.因为AP core的运行是通过reset core开始的，即AP core会从位于zero address的vector table的第一项（Reset vector）开始执行。

③

```
static void launch_ap_core0( rom_types_e rom_type, board_types_e board_type )

{

    CIU_REGS_t *ciu_regs = (CIU_REGS_t*) get_CIU_base();


    ciu_regs->A53_CORE_0_CFG_CTL &= ~(AARCH32_MODE | VINITHI_SET_LOW_VEC); //
```

AArch32 mode, VINITHI=low vectors

```
    if (is_Gr2() && is_RevA())
    {
        // Rev A has polarity of DISTPWRUP backwards, so fix the unused cores
        PMU_disable (board_type, ePMU_DEVICE_CORE3);   // Turn off AP CPU core 3
        PMU_disable (board_type, ePMU_DEVICE_CORE2);   // Turn off AP CPU core 2
        PMU_disable (board_type, ePMU_DEVICE_CORE1);   // Turn off AP CPU core 1
    }
    else if (is_Ge2())
    {
        PMU_disable (board_type, ePMU_DEVICE_CORE1);   // Turn off AP CPU core 1
    }

    PMU_enable (board_type, ePMU_DEVICE_CORE0);   // Turn on AP CPU core 0
}
```

Granite2 has 4 AP cores, Gemstone2 has 2 AP cores.

最后Turn on AP core 0，相当于给core 0一个reset。

④

R4本身已经在运行，所以直接跳转到threadx所在的R4_loaded_code_launch_addr即可。