内存分析：

龙芯1b系列搭载了64MB DDR2内存

由于龙芯使用byte寻址，所以可用内存地址空间为kseg0 和kseg1。（见龙芯1b开发手册） 可用地址空间为0x8000 0000 – 0x8400 0000和0xa000 0000 – 0xa400 0000。注意**这两个内存空间是映射到同样的物理内存的**。也就是这个内存地址是**虚拟地址**，但是k0区段是有缓存的虚拟地址区段，而k1区段是没有缓存的区段。故在调用k0区段时，读写速度都将显著高于k1区段。

注意：在使用GCC编译C程序时，寄存器也会优先使用k0区段作为各个函数的栈。而k1地址区段在使用DMA时可能会被调用，注意不要和k0冲突。

我们在调用asm库做一小段演示代码：

int main**()** **{**

//notice here by using the asm we can cross the assmebly into the c code

//here the address is not in the stack of the main function

asm volatile**(**

"li $t1, 10;"

//t2 is the address for test

//t2 = 0x8000 0004

"lui $t2, 0x8000;"

"ori $t2, 0x0004;"

"sw $t1, 0($t2);"

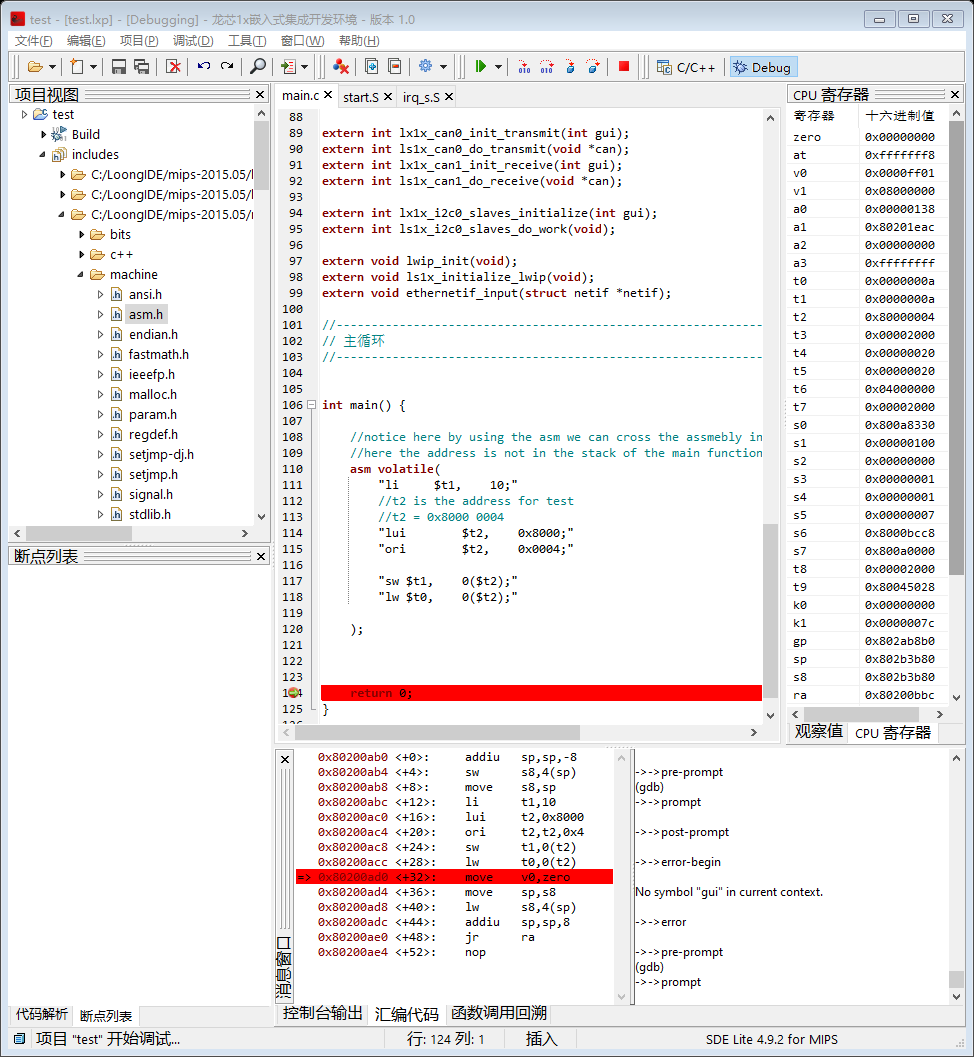
"lw $t0, 0($t2);"

**);**

**return** 0**;**

**}**

可以看到，t0寄存器已经成功通过内存 0x8000 0004 获得t1寄存器中的内容，保持ddr2的功能。当然仔细观察会发现其实汇编代码也都保存在内存中。



为了演示cash与uncash的内存的区别，我们通过k0与k1映射与同一段物理内存可以设置以下代码进行演示：

从这里我们可以看出来，在clean\_dcache这个函数执行之前，从0xa000 0000地址并无法读取到我们所输入的内容。但是在clean\_dcache这个函数执行过之后我们就可以正常的读取内存中的内容。故可以看出k0地址空间中是拥有cashe的，所以在cashe被清理之前并不会向物理内存中写入值。

int main**()** **{**

//notice here by using the asm we can cross the assmebly into the c code

//here the address is not in the stack of the main function

asm volatile**(**

"lui $t0, 0xdead;"

"ori $t0, 0xbeef;"

//t2 is the address for test

//t2 = 0x8000 0004

"lui $t1, 0x8001;"

"sw $t0, 0($t1);"

"lui $t1, 0xa001;"

"lw $t3, 0($t1);"

**);**

clean\_dcache**(**0x80010000**,**4**);**

//go to "add $t2, 4;"

asm volatile**(**

"lui $t1, 0x8001;"

"lw $t2, 0($t1);"

"lui $t1, 0xa001;"

"lw $t3, 0($t1);"

**);**

**return** 0**;**

**}**

同样的，为了能够更明显的表现uncached 内存的特性，我们使用以下的示例来表现。注意：uncached内存在被写入时，数据会被立即写入到物理内存中，故速度会相对较慢，但是在部分操作中仍然十分重要。

int main**()** **{**

//notice here by using the asm we can cross the assmebly into the c code

//here the address is not in the stack of the main function

asm volatile**(**

"lui $t0, 0xdead;"

"ori $t0, 0xbeef;"

//t2 is the address for test

//t2 = 0x8000 0004

"lui $t1, 0xa001;"

"sw $t0, 0($t1);"

"lui $t1, 0x8001;"

"lw $t3, 0($t1);"

**);**

int a **=** 0**;**

**return** 0**;**

**}**