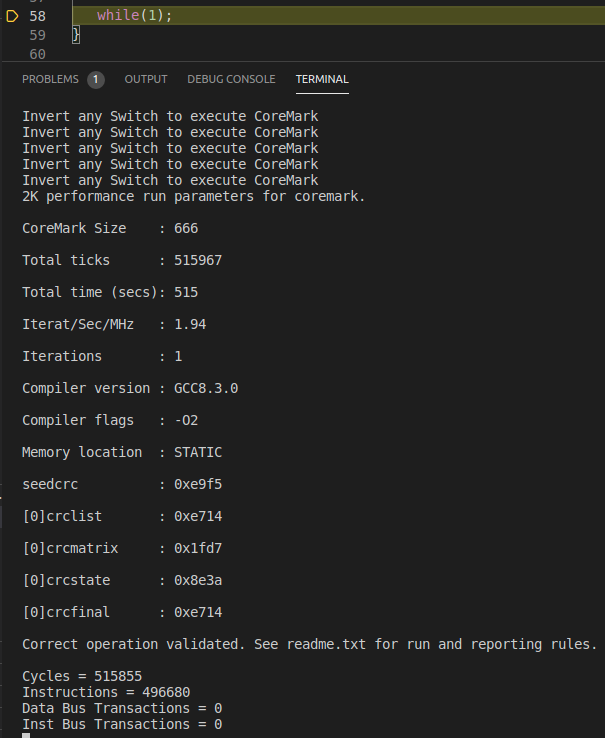
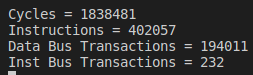
**练习1：**在文件*platformio.ini*中，注释掉第18行并取消注释第19行，以便程序使用以下路径中的链接器脚本：*Lab5/RealBenchmarks/CoreMark\_HwCounters/ld/link\_DCCM-ICCM.ld*。分析新的链接器脚本，该脚本使用DCCM存储大部分数据，使用ICCM存储指令。执行CoreMark基准测试，将结果与演示实验提供的结果进行比较。在本例中，由于我们的默认RVfpga系统不包括ICCM，请使用以下路径中的比特流： *Lab5/RealBenchmarks/Bitstreams/rvfpganexys\_DCCM-ICCM.bit*。



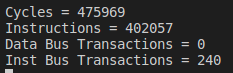
在练习1中，CM/MHz（即Iterat/Sec/MHz）的值为1.94。仅使用DCCM时，CM/MHz为1.88。与仅使用DCCM时相比，性能略微提升，这是由于周期次数稍有减少。SweRV EH1处理器包含一个指令缓存，因此，使用ICCM只会产生很小的差异，性能提升的幅度并不大。最后，可以观察到数据和指令总线事务数均为0，因为数据和指令分别通过DCCM和ICCM访问。

**练习2：**使用Dhrystone基准替代CoreMark基准，执行相同的分析。可访问以下路径获取包含Dhrystone基准的PlatformIO项目： *Lab5/RealBenchmarks/Dhrystone\_HwCounters*。根据所有基准的要求，我们已使用<https://github.com/chipsalliance/Cores-SweRV>中提供的源代码对该Dhrystone基准进行了修改，使其能够适用于特定系统（在本例中为RVfpga系统）。文件*Test.c*与CoreMark中的文件类似，但会调用函数main\_dhry()，该函数包含Dhrystone基准本身。

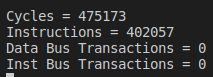
* 无编译器优化，无DCCM，无ICCM



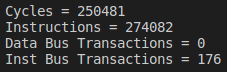
* DCCM



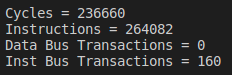
* DCCM和ICCM



* 编译器优化（-O2）和DCCM

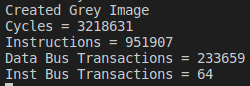


* 编译器优化（-O3）和DCCM

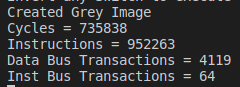


**练习3：**使用图像处理应用程序替代CoreMark，执行相同的分析。可访问以下路径获取包含图像处理应用程序的PlatformIO项目：*/Lab5/RealBenchmarks/ImageProcessing\_HwCounters。*我们在之前实验中使用该程序将RGB图像转换为灰度图像。文件*Test.c*与CoreMark中的文件类似，但会调用函数ImageTransformation()，该函数相应的图像转换基准。默认RVfpga系统的DCCM没有足够的空间来存储图像，因此需使用具有128 KiB DCCM的RVfpga系统（比特流），该文件路径为： *Lab5/RealBenchmarks/Bitstreams/rvfpganexys\_DCCM-128.bit*。

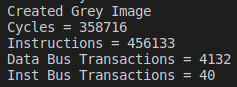
* 无编译器优化，无DCCM，无ICCM



* DCCM



* 编译器优化（-O2）和DCCM



* 编译器优化（-O3）和DCCM

