How can I measure clock cycles per instruction (CPI) using TRACE32?

2025-08-21 - Comments (0) - Tracing

There are mainly two methods to obtain **CPI** (clock cycles per instruction) or **MIPS** (Millions of Instructions Per Second) in TRACE32.

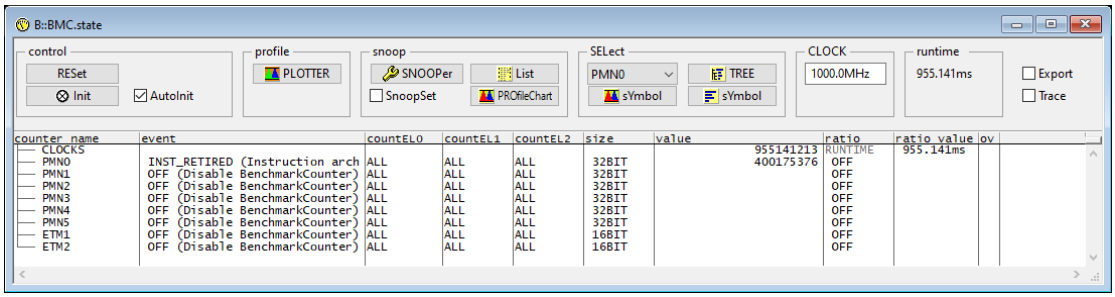
BenchMark Counters

The first method is **BMC (BenchMark Counters)**, which in the case of Arm processors correspond to the PMU (Performance Monitoring Unit). BMC is supported by the TRACE32 `BMC.*` command group. The results are directly obtained from the core counters, which makes BMC the recommended primary method for measuring CPI or MIPS.

In addition, BMC often provides counters for cache hits and misses, which may also be useful in the context of CPI analysis. To check whether BMC is supported for your processor architecture, refer to the *Processor Architecture Manual* for your target in main.pdf. Generic BMC commands are documented in the General Commands Reference Guide B.

Example for Arm Cortex-A72

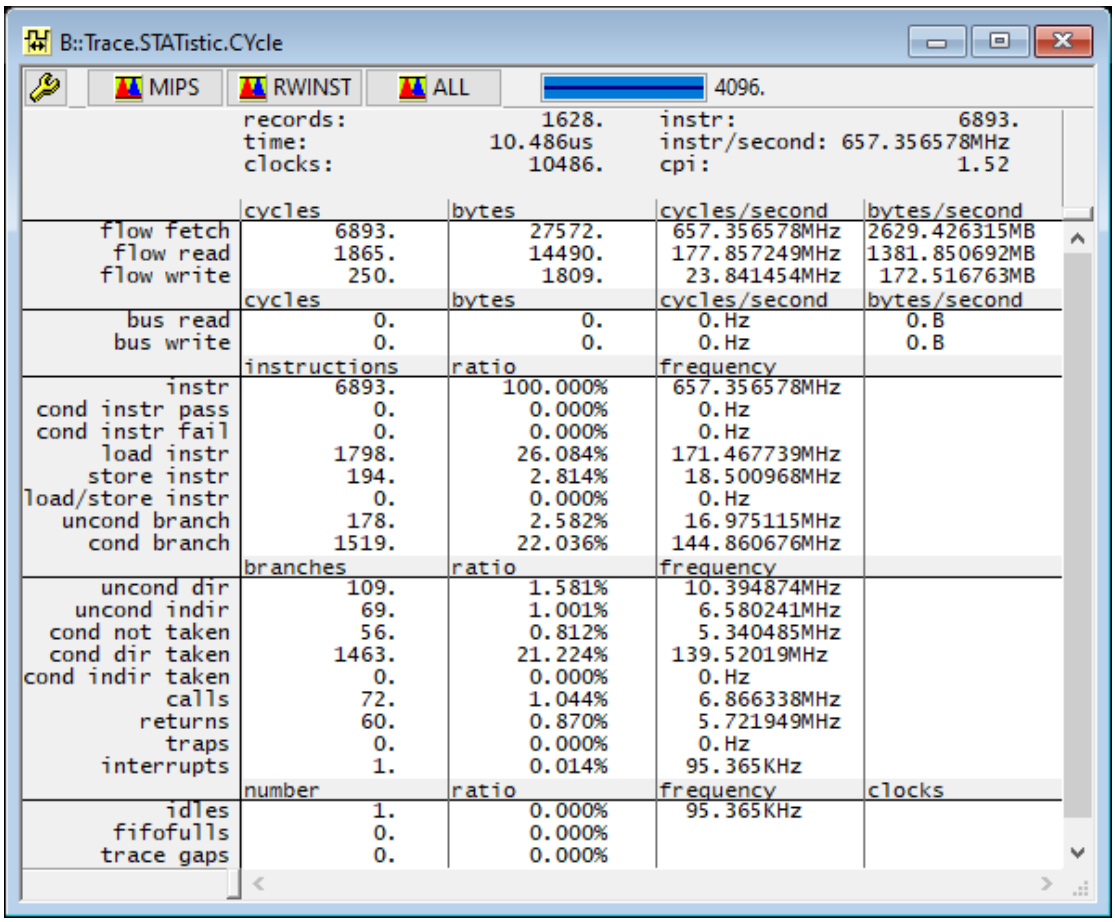
```
BMC.state
BMC.AutoInit ON
BMC.CLOCK 1000MHz
BMC.PMN0.EVENT INST_RETIRED
BMC.PMN0.RATIO X/TIME
BMC.Init
Go
WAIT 1.s
Break
```



Using Trace-Based Measurments

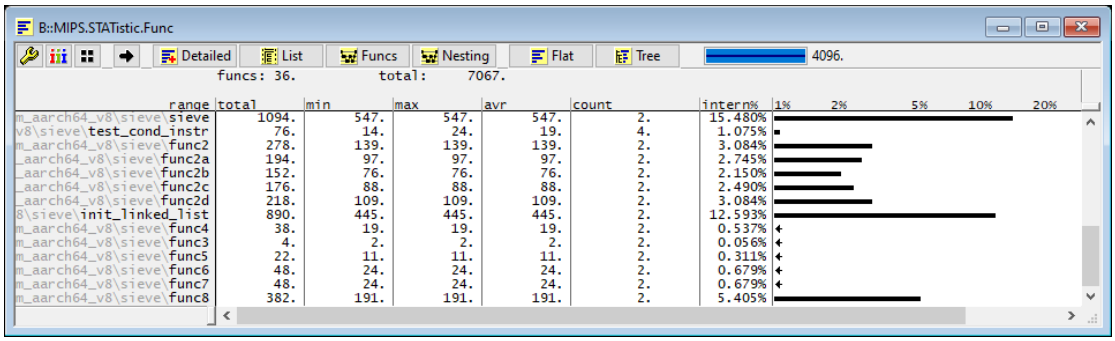
The second method is to use **trace-based measurements** to obtain **CPI** or **MIPS** values. This approach is mainly useful for metrics not covered by BMC counters. CPI can be statistically analyzed from trace data using commands such as **Trace.STATistic.CYcle**, which provides average clocks per instruction calculated from traced instructions and cycles.

```
Trace.STATistic.CYcle
```



The `MIPS.*` commands can additionally be used to analyze the **MIPS.** The system can be analyzed under different aspects: workload per task, workload per high-level language line, workload per specified functional group etc. The following command displays for instance a numerical analysis per function.

`MIPS.STATistic.Func`

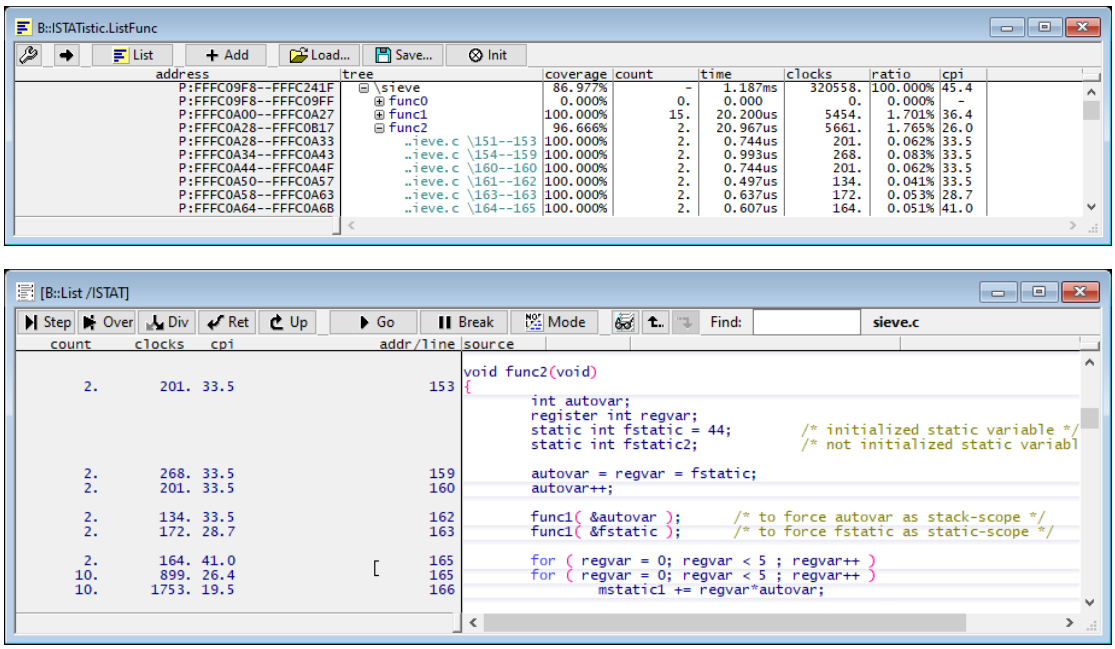


Refer for more information about the **`MIPS.*`** command group to the General Commands Reference Guide M.

Moreover, you can use the `ISTATistic.*` command group to obtain detailed information for individual instructions when a program trace is available. However, it should be noted that for “high-end” cores this method may not provide cycle-accurate results for every single instruction.

Example:

```
ISTATistic.ADD
ISTATistic.ListFunc
List /ISTAT
```



Refer for more information about the `ISTATistic.*` command group to the General Commands Reference Guide I.

Note

The trace results must not include errors, such as FIFOFULLs or FLOWERRORs, in order to provide reliable results. Refer for more information to HARDERRORs, FIFOFULLs and FLOWERRORs in the trace.