Supporting RISC-V Performance Counters

through Performance analysis tools for Linux (Perf)

Joao Mario Domingos
INESC-ID
IST, Universidade de Lisboa
joao.mario@tecnico.ulisboa.pt

Pedro Tomás
INESC-ID
IST, Universidade de Lisboa
pedro.tomas@inesc-id.pt

Leonel Sousa
INESC-ID
IST, Universidade de Lisboa
las@inesc-id.pt

CARRV’21 Workshop (June 17, 2021)

Performance Monitoring:
why it matters for system designers and software developers

There are multiple ways improve software performance:

- Execution time profiling (count cycles)
- Hardware simulators (software-based, mixed)
- Really complex do it all tools (Intel Advisor, Arm Coresight)
- **Perf** (the simplest, most powerful linux integrated tool for performance monitoring)

Monitoring performance, for whom it is?

- Software Developers
- System Designers
- System Administrators
- ...

Want to get faster?
Profile and tune your system

RISC-V HPM:

the RISC-V hardware performance monitor

- Barebones hardware performance monitor specification since v1.7.
- The counter enable mask (v1.9) allows for control over who (privilege level) can access each counter.

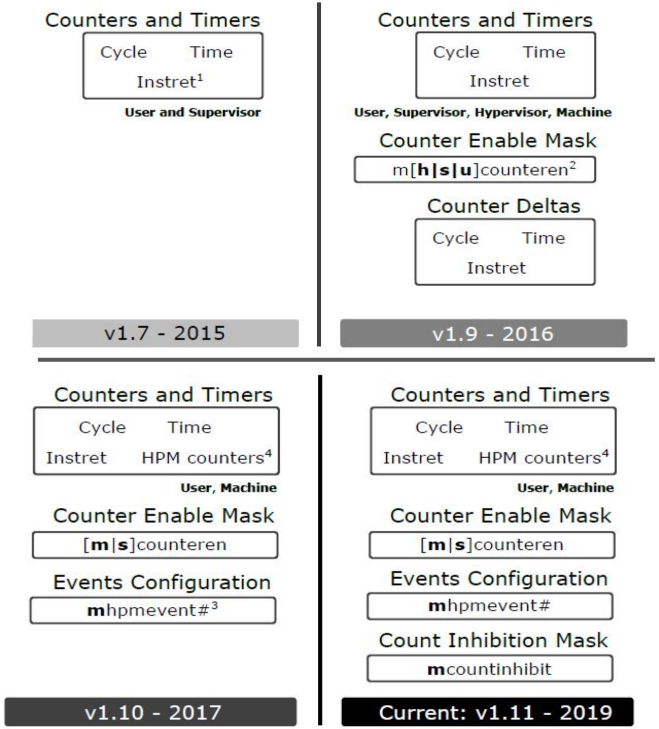
Event Configuration:

- v1.10 brought 29 new counters with configurable events (HPMcounters)
- Configuration is achieved with one event configuration register for each HPMcounter (up to 2^{64} events per counter)

Atomic Sampling:

- Count inhibition (v1.11) allows for counters to be stopped and resumed through a CSR, allowing for precise event sampling.

- RISC-V specification hints about future support for overflow interrupts



Counters and Timers
Cycle Time
Instret[1]
User and Supervisor
v1.7 - 2015

Counters and Timers
Cycle Time
Instret
User, Supervisor, Hypervisor, Machine
Counter Enable Mask
m[**h**|**s**|**u**]counteren[2]
Counter Deltas
Cycle Time
Instret
v1.9 - 2016

Counters and Timers
Cycle Time
Instret HPM counters[4]
User, Machine
Counter Enable Mask
[**m**|**s**]counteren
Events Configuration
mhpmevent#[3]
v1.10 - 2017

Counters and Timers
Cycle Time
Instret HPM counters[4]
User, Machine
Counter Enable Mask
[**m**|**s**]counteren
Events Configuration
mhpmevent#
Count Inhibition Mask
mcountinhibit
Current: v1.11 - 2019

RISC-V HPM + Linux Perf

Using Linux performance monitoring tools to support RISC-V hardware performance monitor

RISC-V HPM + Linux Perf:

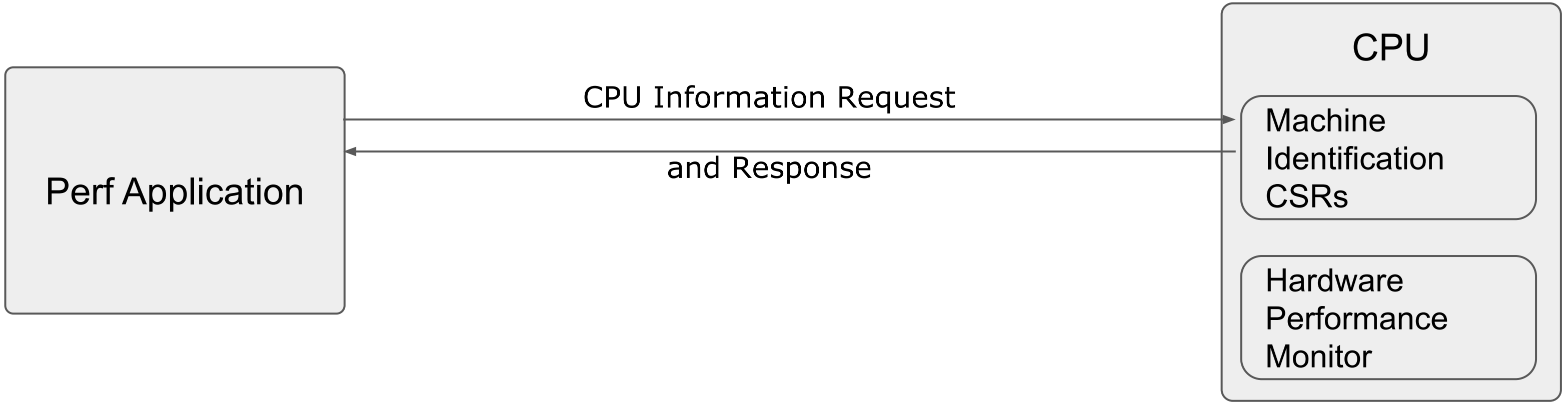
Software Overview

Perf Application
CPU
Machine Identification CSRs
Hardware Performance Monitor

- CPU events identification
- Event attribution and counting (through the driver)
- Tools for improved performance monitoring
- Display results in an uniform way
- Perf is just a frontend for the kernel driver

RISC-V HPM + Linux Perf:

Software Overview

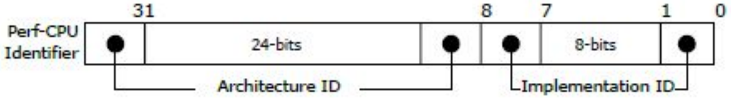


- **CPU events identification** **Contribution**
  - CPU PMU identification through the architecture ID and implementation ID
  - Each CPU unique ID selects the appropriate set of events for that processor

RISC-V HPM + Linux Perf:

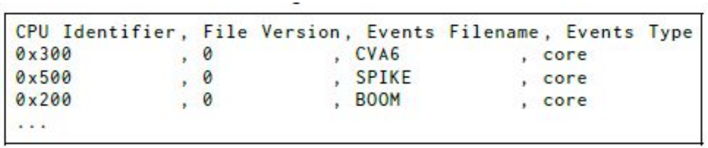
CPU specific event selection with Perf pmu_events

Contributions

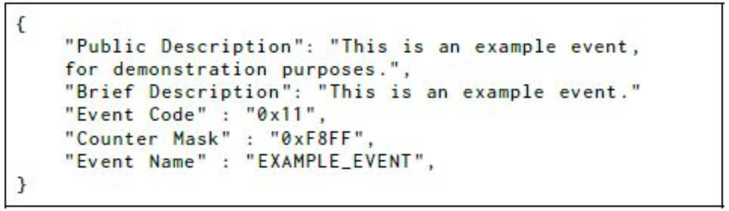


CPU PMU Unique ID (composed by architecture and implementation IDs)

Used to match the CPU PMU with a set of event description code



```
CPU Identifier, File Version, Events Filename, Events Type
0x300         , 0           , CVA6           , core
0x500         , 0           , SPIKE          , core
0x200         , 0           , BOOM           , core
...
```

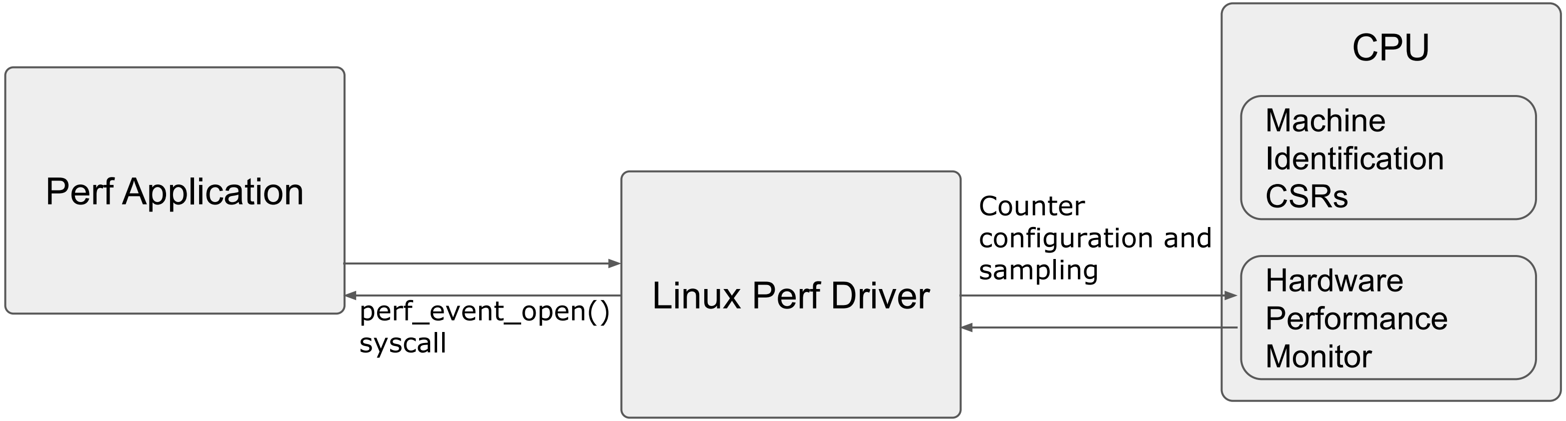


```
{
    "Public Description": "This is an example event,
    for demonstration purposes.",
    "Brief Description": "This is an example event."
    "Event Code" : "0x11",
    "Counter Mask" : "0xF8FF",
    "Event Name" : "EXAMPLE_EVENT",
}
```

Events are simply described in a JSON format. Each event is identified by the name, and provides an event code and a counter selection mask.

RISC-V HPM + Linux Perf:

Software Overview

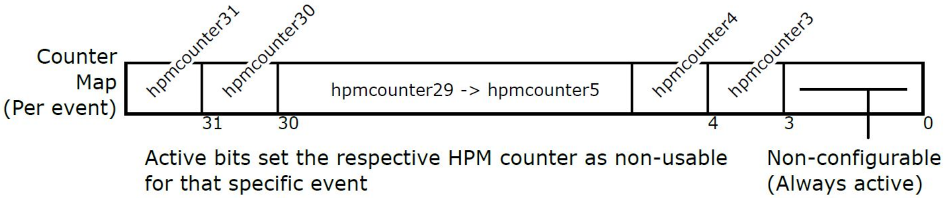


- **Event attribution and counting**
  - Perf Initiates performance counting through a system call
  - The Linux Perf driver setups events and samples counters

RISC-V HPM + Linux Perf:

Counting events with the perf kernel driver

Contributions



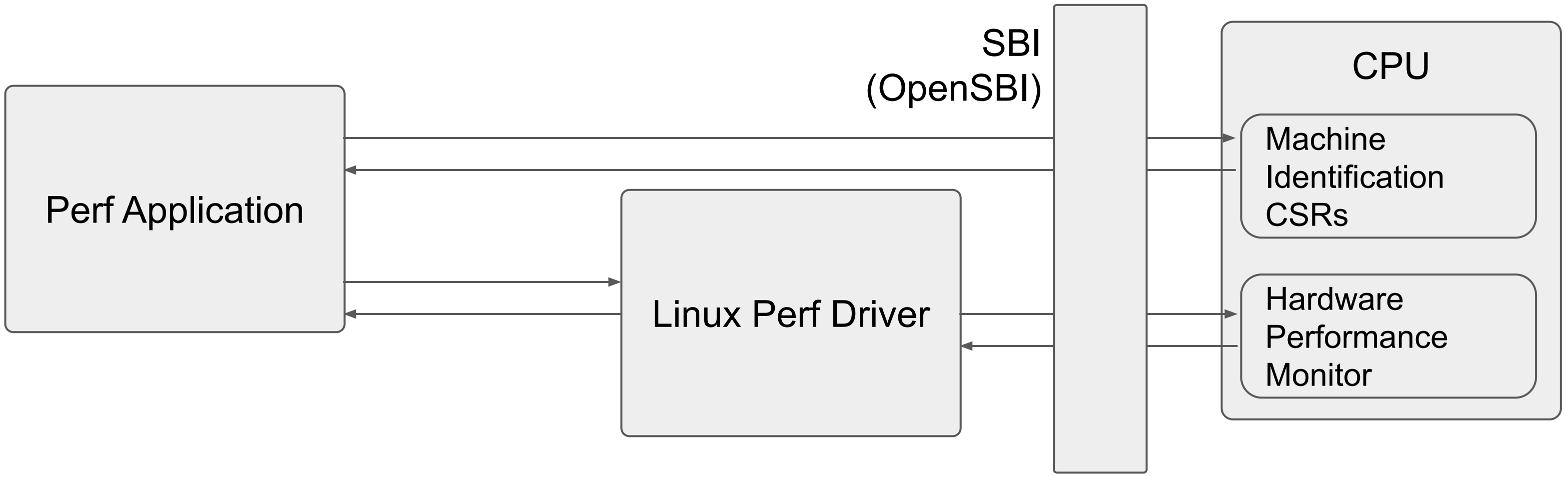
Counter Map (Per event)
hpmcounter31
hpmcounter30
hpmcounter29 -> hpmcounter5
hpmcounter4
hpmcounter3
31
30
4
3
0
Active bits set the respective HPM counter as non-usable for that specific event
Non-configurable (Always active)

New perf event configuration allows for each event to have a counter mask, providing identification of available counters.

- Improved event configuration through the RISC-V *mhpmevent#* registers.
- Improved support for raw events

RISC-V HPM + Linux Perf:

Software Overview

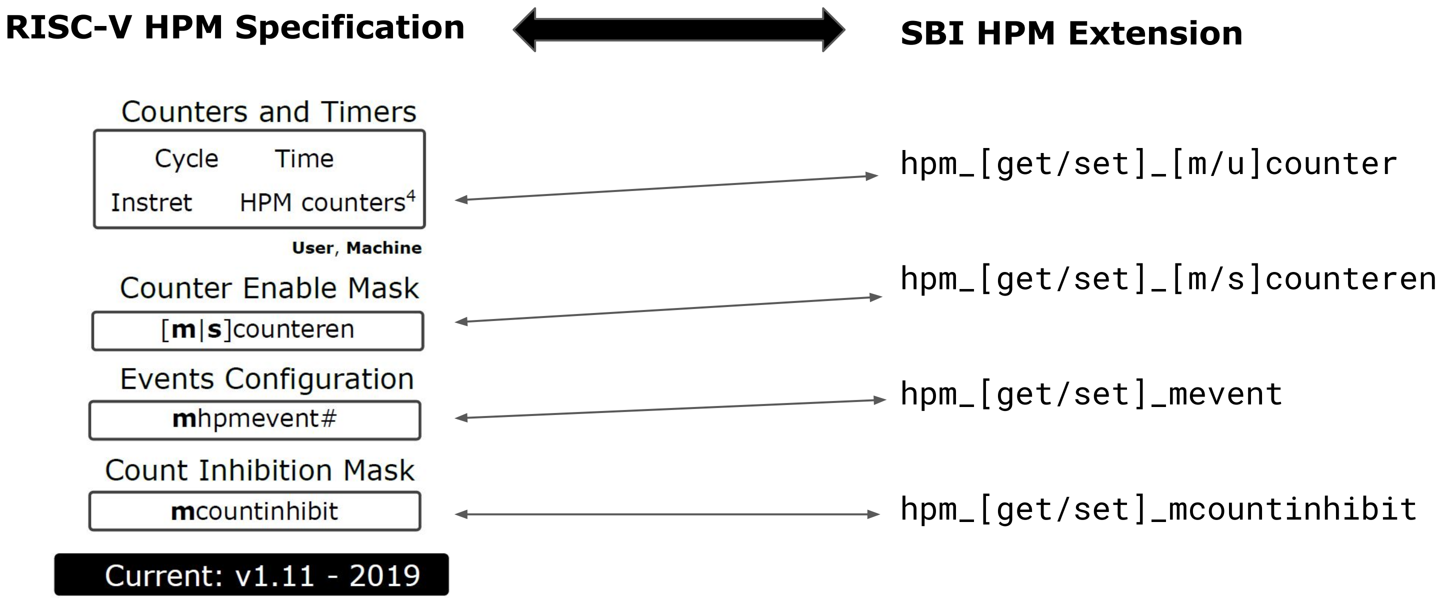


- **Privilege escalation to machine-mode** **Contribution**
  - SBI/OpenSBI extension for HPM counters interaction (configuration, read, write)
  - Added Perf and Perf Driver SBI calls for HPM extension and machine identification CSRs access

RISC-V HPM + Linux Perf:

Privileged access through OpenSBI

Contributions



RISC-V HPM Specification

SBI HPM Extension

Counters and Timers

Cycle Time

Instret HPM counters[4]

User, Machine

hpm_[get/set]_[m/u]counter

Counter Enable Mask

[**m**|**s**]counteren

hpm_[get/set]_[m/s]counteren

Events Configuration

mhpmevent#

hpm_[get/set]_mevent

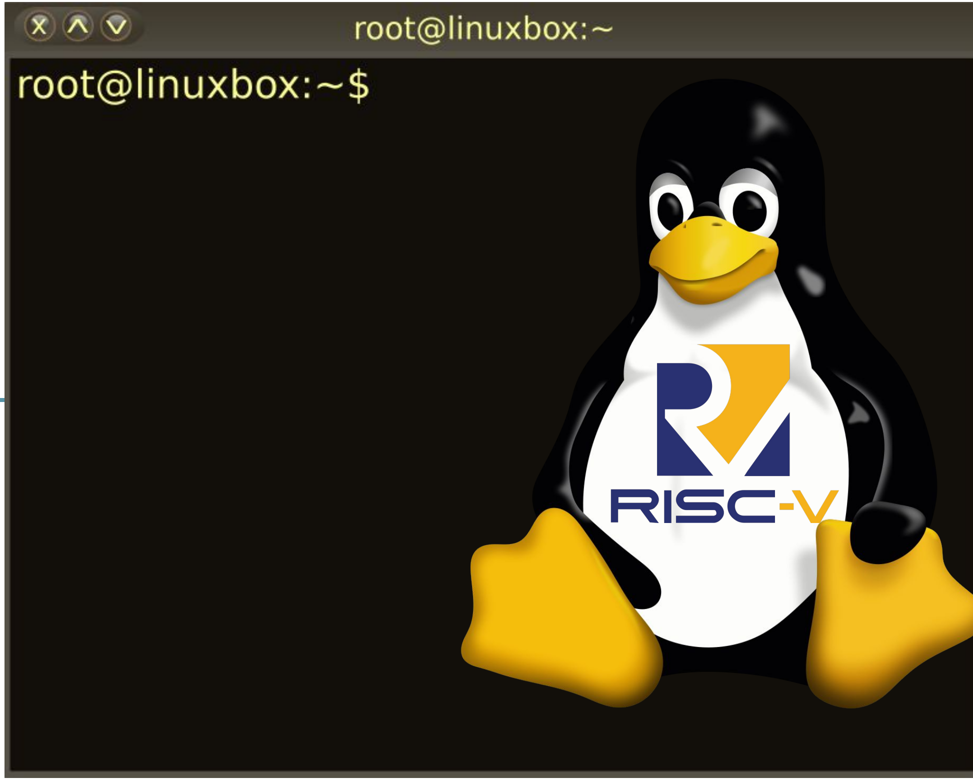
Count Inhibition Mask

mcountinhibit

hpm_[get/set]_mcountinhibit

Current: v1.11 - 2019

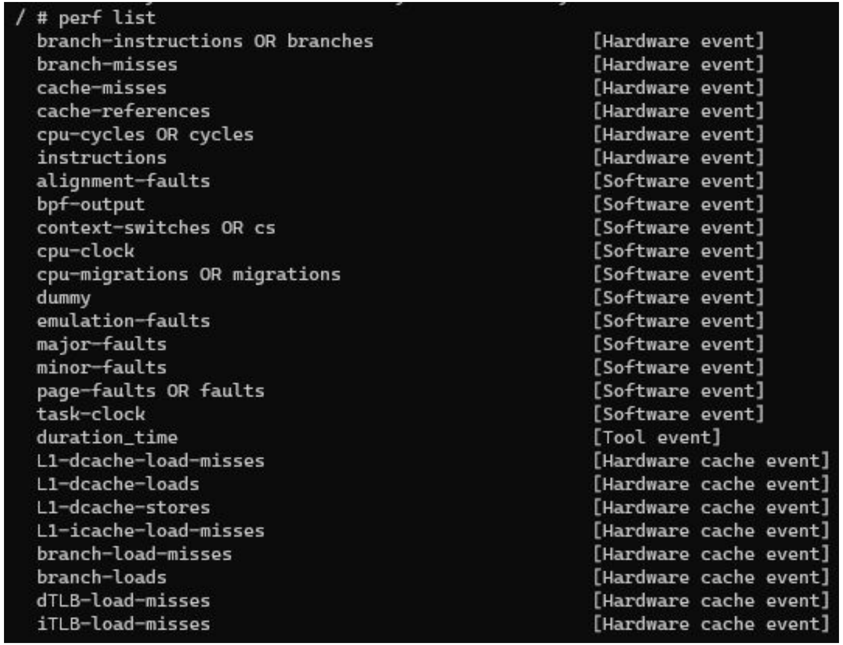
Initial Results



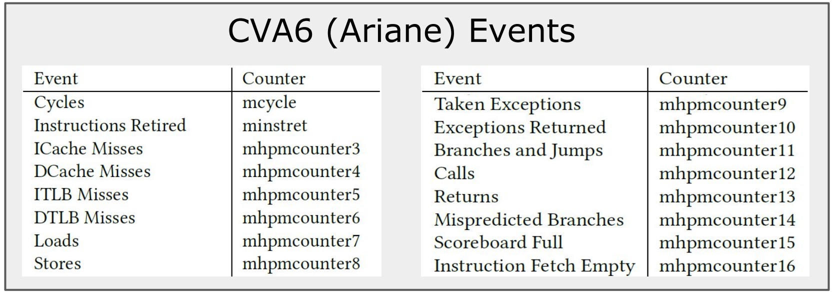
```
root@linuxbox:~$
```

Working Perf:

the *perf* ***list*** command



```
/ # perf list
  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  alignment-faults                                   [Software event]
  bpf-output                                         [Software event]
  context-switches OR cs                             [Software event]
  cpu-clock                                          [Software event]
  cpu-migrations OR migrations                       [Software event]
  dummy                                              [Software event]
  emulation-faults                                   [Software event]
  major-faults                                       [Software event]
  minor-faults                                       [Software event]
  page-faults OR faults                              [Software event]
  task-clock                                         [Software event]
  duration_time                                      [Tool event]
  L1-dcache-load-misses                              [Hardware cache event]
  L1-dcache-loads                                    [Hardware cache event]
  L1-dcache-stores                                   [Hardware cache event]
  L1-icache-load-misses                              [Hardware cache event]
  branch-load-misses                                 [Hardware cache event]
  branch-loads                                       [Hardware cache event]
  dTLB-load-misses                                   [Hardware cache event]
  iTLB-load-misses                                   [Hardware cache event]
```

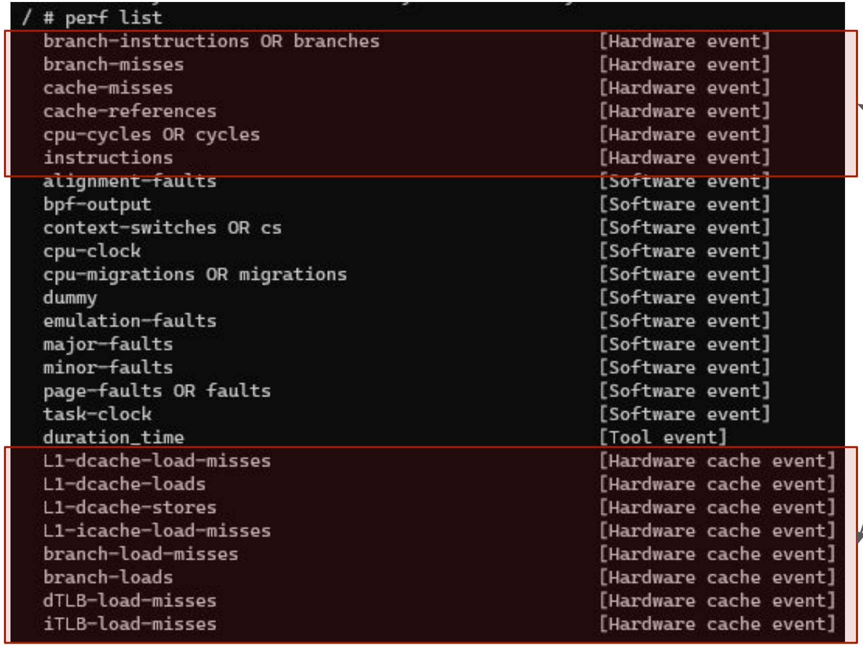


CVA6 (Ariane) Events

Event	Counter	Event	Counter
Cycles	mcycle	Taken Exceptions	mhpmcounter9
Instructions Retired	minstret	Exceptions Returned	mhpmcounter10
ICache Misses	mhpmcounter3	Branches and Jumps	mhpmcounter11
DCache Misses	mhpmcounter4	Calls	mhpmcounter12
ITLB Misses	mhpmcounter5	Returns	mhpmcounter13
DTLB Misses	mhpmcounter6	Mispredicted Branches	mhpmcounter14
Loads	mhpmcounter7	Scoreboard Full	mhpmcounter15
Stores	mhpmcounter8	Instruction Fetch Empty	mhpmcounter16

Working Perf:

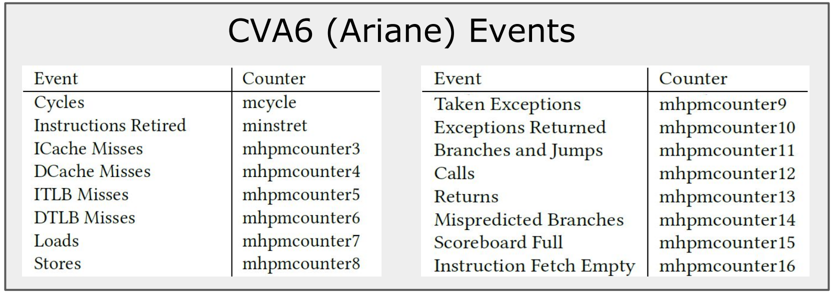
the *perf* ***list*** command



```
/ # perf list
  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  alignment-faults                                   [Software event]
  bpf-output                                         [Software event]
  context-switches OR cs                             [Software event]
  cpu-clock                                          [Software event]
  cpu-migrations OR migrations                       [Software event]
  dummy                                              [Software event]
  emulation-faults                                   [Software event]
  major-faults                                       [Software event]
  minor-faults                                       [Software event]
  page-faults OR faults                              [Software event]
  task-clock                                         [Software event]
  duration_time                                      [Tool event]
  L1-dcache-load-misses                              [Hardware cache event]
  L1-dcache-loads                                    [Hardware cache event]
  L1-dcache-stores                                   [Hardware cache event]
  L1-icache-load-misses                              [Hardware cache event]
  branch-load-misses                                 [Hardware cache event]
  branch-loads                                       [Hardware cache event]
  dTLB-load-misses                                   [Hardware cache event]
  iTLB-load-misses                                   [Hardware cache event]
```

Supports Perf default Hardware and Hardware Cache events

Not all Ariane Events are supported as Perf default events, we need **raw events**



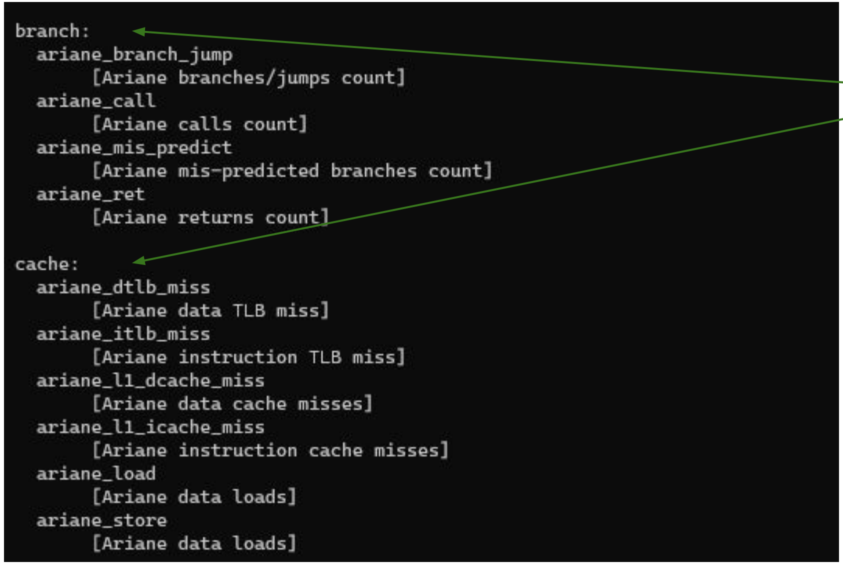
CVA6 (Ariane) Events

Event	Counter
Cycles	mcycle
Instructions Retired	minstret
ICache Misses	mhpmcounter3
DCache Misses	mhpmcounter4
ITLB Misses	mhpmcounter5
DTLB Misses	mhpmcounter6
Loads	mhpmcounter7
Stores	mhpmcounter8

Event	Counter
Taken Exceptions	mhpmcounter9
Exceptions Returned	mhpmcounter10
Branches and Jumps	mhpmcounter11
Calls	mhpmcounter12
Returns	mhpmcounter13
Mispredicted Branches	mhpmcounter14
Scoreboard Full	mhpmcounter15
Instruction Fetch Empty	mhpmcounter16

Working Perf:

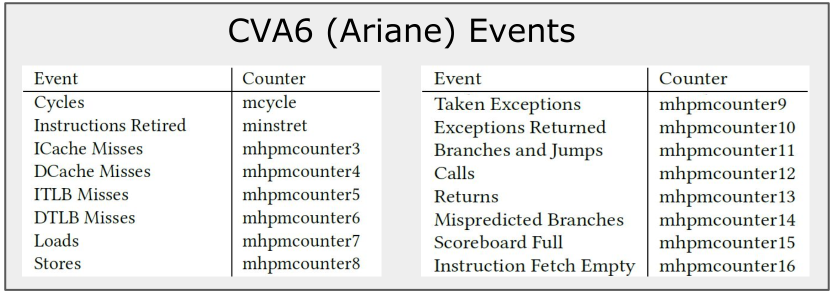
the *perf* ***list*** command



```
branch:
  ariane_branch_jump
       [Ariane branches/jumps count]
  ariane_call
       [Ariane calls count]
  ariane_mis_predict
       [Ariane mis-predicted branches count]
  ariane_ret
       [Ariane returns count]

cache:
  ariane_dtlb_miss
       [Ariane data TLB miss]
  ariane_itlb_miss
       [Ariane instruction TLB miss]
  ariane_l1_dcache_miss
       [Ariane data cache misses]
  ariane_l1_icache_miss
       [Ariane instruction cache misses]
  ariane_load
       [Ariane data loads]
  ariane_store
       [Ariane data loads]
```

Events Grouping Supported

Support for all CVA6 events

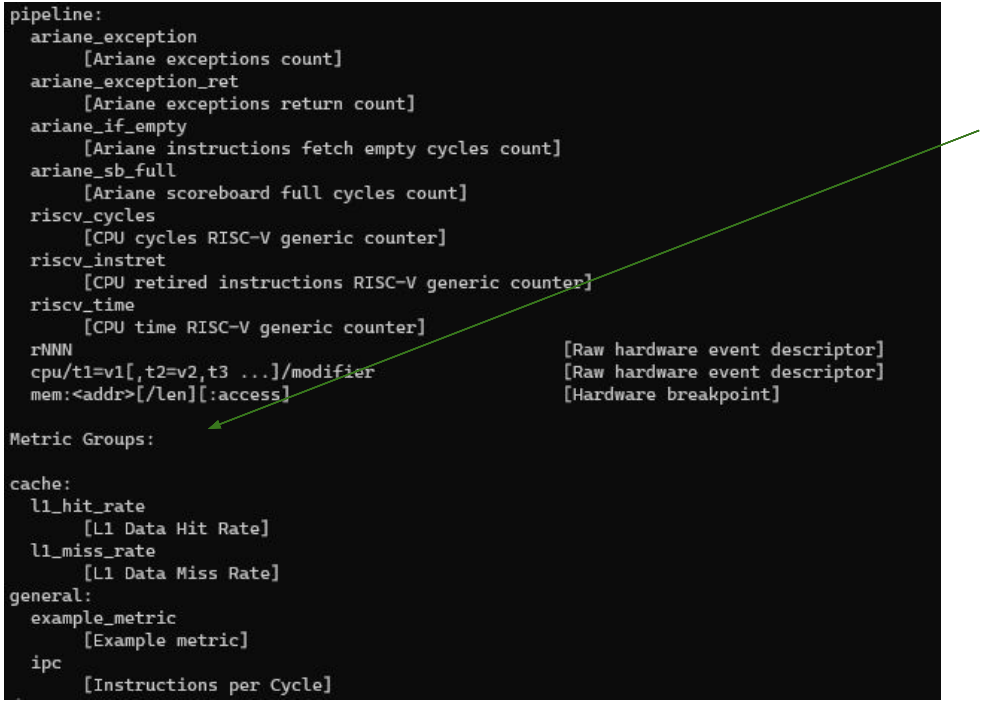


CVA6 (Ariane) Events

Event	Counter	Event	Counter
Cycles	mcycle	Taken Exceptions	mhpmcounter9
Instructions Retired	minstret	Exceptions Returned	mhpmcounter10
ICache Misses	mhpmcounter3	Branches and Jumps	mhpmcounter11
DCache Misses	mhpmcounter4	Calls	mhpmcounter12
ITLB Misses	mhpmcounter5	Returns	mhpmcounter13
DTLB Misses	mhpmcounter6	Mispredicted Branches	mhpmcounter14
Loads	mhpmcounter7	Scoreboard Full	mhpmcounter15
Stores	mhpmcounter8	Instruction Fetch Empty	mhpmcounter16

Working Perf:

the *perf* ***list*** command



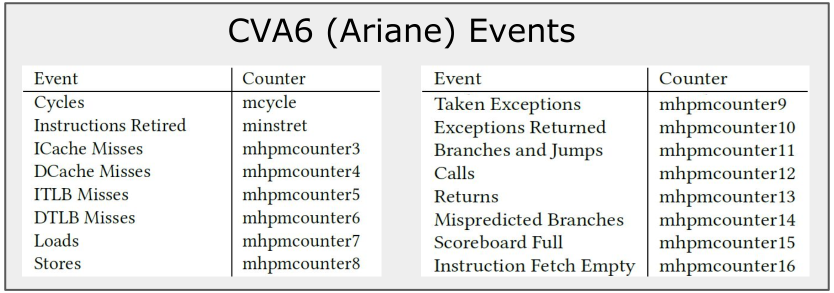
```
pipeline:
  ariane_exception
       [Ariane exceptions count]
  ariane_exception_ret
       [Ariane exceptions return count]
  ariane_if_empty
       [Ariane instructions fetch empty cycles count]
  ariane_sb_full
       [Ariane scoreboard full cycles count]
  riscv_cycles
       [CPU cycles RISC-V generic counter]
  riscv_instret
       [CPU retired instructions RISC-V generic counter]
  riscv_time
       [CPU time RISC-V generic counter]
  rNNN                                                  [Raw hardware event descriptor]
  cpu/t1=v1[,t2=v2,t3 ...]/modifier                     [Raw hardware event descriptor]
  mem:<addr>[/len][:access]                             [Hardware breakpoint]

Metric Groups:

cache:
  l1_hit_rate
       [L1 Data Hit Rate]
  l1_miss_rate
       [L1 Data Miss Rate]
general:
  example_metric
       [Example metric]
  ipc
       [Instructions per Cycle]
```

Support for all CVA6 events

Support for metrics

(e.g. l1_hit_rate ⇔ ariane_l1d_cache_misses/ariane_loads)

Metrics can be easily customized by changing a simple equation in a JSON file

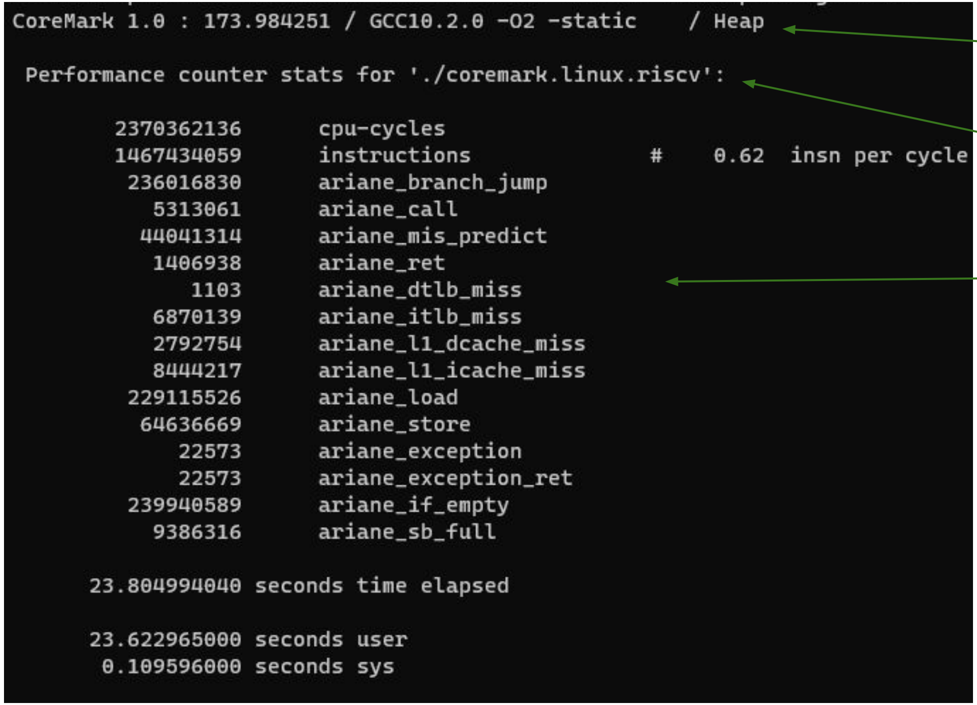


CVA6 (Ariane) Events

Event	Counter	Event	Counter
Cycles	mcycle	Taken Exceptions	mhpmcounter9
Instructions Retired	minstret	Exceptions Returned	mhpmcounter10
ICache Misses	mhpmcounter3	Branches and Jumps	mhpmcounter11
DCache Misses	mhpmcounter4	Calls	mhpmcounter12
ITLB Misses	mhpmcounter5	Returns	mhpmcounter13
DTLB Misses	mhpmcounter6	Mispredicted Branches	mhpmcounter14
Loads	mhpmcounter7	Scoreboard Full	mhpmcounter15
Stores	mhpmcounter8	Instruction Fetch Empty	mhpmcounter16

Working Perf:

the *perf* stat command



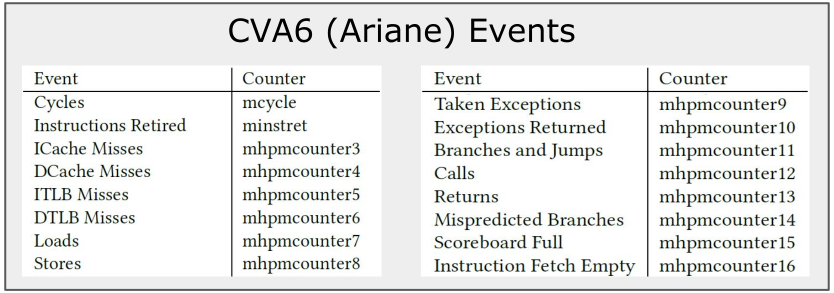
```
CoreMark 1.0 : 173.984251 / GCC10.2.0 -O2 -static    / Heap

 Performance counter stats for './coremark.linux.riscv':

        2370362136      cpu-cycles
        1467434059      instructions              #    0.62  insn per cycle
         236016830      ariane_branch_jump
           5313061      ariane_call
          44041314      ariane_mis_predict
           1406938      ariane_ret
              1103      ariane_dtlb_miss
           6870139      ariane_itlb_miss
           2792754      ariane_l1_dcache_miss
           8444217      ariane_l1_icache_miss
         229115526      ariane_load
          64636669      ariane_store
             22573      ariane_exception
             22573      ariane_exception_ret
         239940589      ariane_if_empty
           9386316      ariane_sb_full

      23.804994040 seconds time elapsed

      23.622965000 seconds user
       0.109596000 seconds sys
```

CoreMark Result: 1.74 points/MHz @ 100MHz

Perf stat outputs event counts

There is support for event multiplexing,

but CVA6 does not have configurable events



CVA6 (Ariane) Events

Event	Counter	Event	Counter
Cycles	mcycle	Taken Exceptions	mhpmcounter9
Instructions Retired	minstret	Exceptions Returned	mhpmcounter10
ICache Misses	mhpmcounter3	Branches and Jumps	mhpmcounter11
DCache Misses	mhpmcounter4	Calls	mhpmcounter12
ITLB Misses	mhpmcounter5	Returns	mhpmcounter13
DTLB Misses	mhpmcounter6	Mispredicted Branches	mhpmcounter14
Loads	mhpmcounter7	Scoreboard Full	mhpmcounter15
Stores	mhpmcounter8	Instruction Fetch Empty	mhpmcounter16

Summary

- Perf already had slim support for the RISC-V Hardware Performance Monitor
- We introduce support for HPM events and a way to couple each event to a selected set of counters
- Events are now discovered by perf based on the CPU identification
- Events are provided through a set of JSON and CSV files, simplifying the work of vendors
- A new SBI/OpenSBI extension allows for machine-mode access from Linux (supervisor-mode) to the HPM registers

Next Steps...

- Other initiatives propose different levels of perf support for the RISC-V HPM specification, this is an opportunity to improve
- Further testing and improvements are on the way, this is an early work
- We seek to test in ASIC until the end of the year

BeagleV™

HiFive Unmatched

epi European Processor Initiative

EPAC

This work is part of EPI, where we aim to implement the proposed work

The work will be open-sourced upon completion and approval