COMPUTER ORGANISATION
Assignment 1

Name: Aman Aslam Sayyad
MIS: 112103127
Div: 2
Batch: T2

1. Title: Study and experimentation using perf tool to observe different statistics of a program.

1. Installing perf

```
(base)  aman2003@Aman-Sayyad  ~/SEM5/CO  main  perf --version
perf version 6.2.16
```

2. Perf commands:

a. perf list: Gives a list of supported events

```
List of pre-defined events (to be used in -e or -M):

  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]
  stalled-cycles-backend OR idle-cycles-backend      [Hardware event]
  stalled-cycles-frontend OR idle-cycles-frontend    [Hardware event]
```

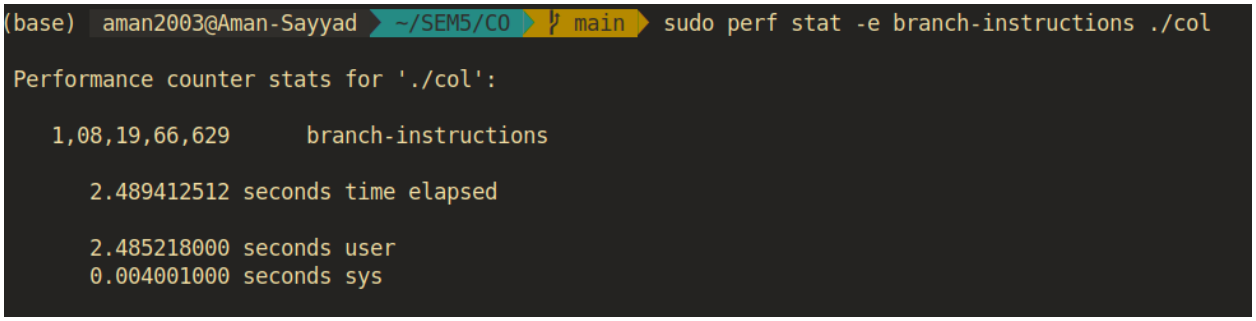
```
L1-dcache-load-misses                                     [Hardware cache event]
L1-dcache-loads                                           [Hardware cache event]
L1-dcache-prefetches                                      [Hardware cache event]
L1-icache-load-misses                                     [Hardware cache event]
L1-icache-loads                                           [Hardware cache event]
branch-load-misses                                        [Hardware cache event]
branch-loads                                              [Hardware cache event]
dTLB-load-misses                                          [Hardware cache event]
dTLB-loads                                                [Hardware cache event]
iTLB-load-misses                                          [Hardware cache event]
iTLB-loads                                                [Hardware cache event]
```

b. Perf --help: It provides a list of options and subcommands for the `perf` tool, allowing you to access help documentation for various `perf` functionalities.

```
(base)  ✘ aman2003@Aman-Sayyad  ~/SEM5/CO  main  perf --help

 usage: perf [--version] [--help] [OPTIONS] COMMAND [ARGS]

 The most commonly used perf commands are:
   annotate        Read perf.data (created by perf record) and display annotated code
   archive         Create archive with object files with build-ids found in perf.data file
   bench           General framework for benchmark suites
   buildid-cache   Manage build-id cache.
   buildid-list    List the buildids in a perf.data file
   c2c             Shared Data C2C/HITM Analyzer.
   config          Get and set variables in a configuration file.
   daemon          Run record sessions on background
   data            Data file related processing
   diff            Read perf.data files and display the differential profile
   evlist          List the event names in a perf.data file
   ftrace          simple wrapper for kernel's ftrace functionality
   inject          Filter to augment the events stream with additional information
   iostat          Show I/O performance metrics
   kallsyms        Searches running kernel for symbols
   kvm             Tool to trace/measure kvm guest os
   list            List all symbolic event types
   mem             Profile memory accesses
   record          Run a command and record its profile into perf.data
   report          Read perf.data (created by perf record) and display the profile
   script          Read perf.data (created by perf record) and display trace output
   stat            Run a command and gather performance counter statistics
   test            Runs sanity tests.
   top             System profiling tool.
   version         display the version of perf binary
   probe           Define new dynamic tracepoints

 See 'perf help COMMAND' for more information on a specific command.
```

3. Perf Events:

a. Branch instructions: This event counts the total number of branch instructions executed by the CPU. Branch instructions are instructions that can change the order of program execution, such as conditional branches and loops. Counting branch instructions can provide insights into the program's control flow and how often it makes decisions or jumps in its execution path.



b. Branch misses:This event counts the number of times the CPU's branch predictor incorrectly predicts the outcome of branch instructions. Modern CPUs use branch prediction to speculate about which branch of a conditional statement will be taken. When the prediction is incorrect, it results in a "branch miss." High branch miss rates can indicate inefficiencies in the branch prediction mechanism and may lead to performance degradation.

c. Cache misses:This event counts the number of times the CPU attempts to access data from its cache memory but fails to find the data there, requiring it to fetch the data from a slower memory location, such as RAM. Cache misses are significant because accessing data from cache is much faster than accessing it from main memory. A high cache miss rate suggests that the program's memory access patterns may not be cache-friendly, leading to slower execution.

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e cache-misses ./col

 Performance counter stats for './col':

         45,08,944      cache-misses

       2.367023322 seconds time elapsed

       2.362795000 seconds user
       0.004004000 seconds sys
```

d. Cache references: This event counts the total number of memory access operations made by the CPU to its cache memory. It includes both successful accesses (cache hits) and unsuccessful accesses (cache misses). Cache references provide an overall measure of how frequently the CPU interacts with its cache. A high cache-reference count may indicate a high degree of memory access in the program.

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e instructions ./col

 Performance counter stats for './col':

   48,25,65,82,701      instructions

       2.342411444 seconds time elapsed

       2.332788000 seconds user
       0.008002000 seconds sys
```

e. CPU cycle:This event measures the total number of clock cycles the CPU takes to execute a program or a specific portion of code. CPU cycles represent the basic unit of time in CPU operations. This metric is useful for understanding the raw computational effort required for a task, as the number of cycles is proportional to the execution time on a CPU.

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e cpu-cycles ./col

 Performance counter stats for './col':

    9,56,49,20,252      cpu-cycles

       2.416310424 seconds time elapsed

       2.407883000 seconds user
       0.007999000 seconds sys
```

f. Instructions:This event counts the total number of machine-level instructions executed by the CPU. Machine instructions are the fundamental operations that the CPU performs to execute a program. Counting instructions helps assess the complexity and computational load of a program. It is often used as a baseline metric for program performance analysis.

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e cache-references ./col

 Performance counter stats for './col':

      14,81,96,240      cache-references

       2.359936060 seconds time elapsed

       2.347622000 seconds user
       0.011998000 seconds sys
```

2. Title: Write a program to multiply two different matrices of size 1024 x 1024.

1. **Column-wise Access:**

a. Branch-instructions:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e branch-instructions ./row

 Performance counter stats for './row':

    1,08,15,19,585      branch-instructions

       2.763779713 seconds time elapsed

       2.759213000 seconds user
       0.004010000 seconds sys
```

b. Branch-misses:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO  main sudo perf stat -e branch-misses ./row

Performance counter stats for './row':

        10,97,622      branch-misses

      2.739373468 seconds time elapsed

      2.735018000 seconds user
      0.004004000 seconds sys
```

c. cache-misses:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO  main sudo perf stat -e cache-misses ./row

Performance counter stats for './row':

     79,99,72,821      cache-misses

      2.711478446 seconds time elapsed

      2.707216000 seconds user
      0.004004000 seconds sys
```

d. Cache-references:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO  main sudo perf stat -e cache-references  ./row

Performance counter stats for './row':

   2,18,08,19,518      cache-references

      2.668550628 seconds time elapsed

      2.660945000 seconds user
      0.004001000 seconds sys
```

e. Cpu-cycle:

```
(base) ✘ aman2003@Aman-Sayyad ~/SEM5/CO  main  sudo perf stat -e cpu-cycles  ./row

 Performance counter stats for './row':

   10,97,04,82,294      cpu-cycles

       2.783969816 seconds time elapsed

       2.775590000 seconds user
       0.007998000 seconds sys
```

f. Instructions:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO  main  sudo perf stat -e instructions ./row

 Performance counter stats for './row':

   48,25,73,09,449      instructions

       2.730515399 seconds time elapsed

       2.726214000 seconds user
       0.004003000 seconds sys
```

2. **Row-wise access:**

a. Branch-instructions:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO  main  sudo perf stat -e branch-instructions ./col

 Performance counter stats for './col':

    1,08,19,66,629      branch-instructions

       2.489412512 seconds time elapsed

       2.485218000 seconds user
       0.004001000 seconds sys
```

b. Branch-misses:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e branch-misses ./col

Performance counter stats for './col':

        10,89,563      branch-misses

      2.356356661 seconds time elapsed

      2.352171000 seconds user
      0.004000000 seconds sys
```

c. cache-misses:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e cache-misses ./col

Performance counter stats for './col':

        45,08,944      cache-misses

      2.367023322 seconds time elapsed

      2.362795000 seconds user
      0.004004000 seconds sys
```

d. Cache-references:

```
(base) aman2003@Aman-Sayyad ~/SEM5/CO main sudo perf stat -e instructions ./col

Performance counter stats for './col':

  48,25,65,82,701      instructions

      2.342411444 seconds time elapsed

      2.332788000 seconds user
      0.008002000 seconds sys
```

e. Cpu-cycles:

```
(base)  aman2003@Aman-Sayyad  ~/SEM5/CO  main  sudo perf stat -e cpu-cycles ./col

Performance counter stats for './col':

   9,56,49,20,252      cpu-cycles

      2.416310424 seconds time elapsed

      2.407883000 seconds user
      0.007999000 seconds sys
```

f. Instructions:

```
(base)  aman2003@Aman-Sayyad  ~/SEM5/CO  main  sudo perf stat -e instructions ./col

Performance counter stats for './col':

  48,25,65,82,701      instructions

      2.342411444 seconds time elapsed

      2.332788000 seconds user
      0.008002000 seconds sys
```

Differences:

- While comparing I came to know the output of cache misses of column-wise is more than row-wise access in matrix multiplication
- cpu cycles of column-wise were more than row-wise because cache misses of column-wise access were more than row-wise access.
- Other events will not have any significant differences because the number of instructions will be the same for both programs, so branch instructions of both programs are the same.

Conclusion:

From above experiments with perf I came to know what software and hardware events are. More specifically I came to know different types of hardware events like branch instructions, branch misses, cache misses, cache references, cpu cycles etc.