Elapsed Time: 2.362s
- **Clockticks:** 5,072,400,000
- **Instructions Retired:** 10,722,600,000
- **CPI Rate:** 0.473
- **MUX Reliability:** 0.989
- **Retiring:** 54.3% of Pipeline Slots
  - **Light Operations:** 49.7% of Pipeline Slots
    - **FP Arithmetic:** 1.0% of uOps
      - **FP x87:** 0.0% of uOps
      - **FP Scalar:** 1.0% of uOps
      - **FP Vector:** 0.0% of uOps
    - **Other:** 99.0% of uOps
  - **Heavy Operations:** 4.6% of Pipeline Slots
    - **Microcode Sequencer:** 0.9% of Pipeline Slots
      - **Assists:** 0.0% of Pipeline Slots
- **Front-End Bound:** 20.5% of Pipeline Slots

> Issue: A significant portion of Pipeline Slots is remaining empty due to issues in the Front-End.
>
> Tips: Make sure the code working size is not too large, the code layout does not require too many memory accesses per cycle to get enough instructions for filling four pipeline slots, or check for microcode assists.

- **Front-End Latency:** 8.5% of Pipeline Slots
  - **ICache Misses:** 2.1% of Clockticks
  - **ITLB Overhead:** 0.2% of Clockticks
  - **Branch Resteers:** 5.8% of Clockticks
    - **Mispredicts Resteers:** 5.3% of Clockticks
    - **Clears Resteers:** 0.0% of Clockticks
    - **Unknown Branches:** 0.5% of Clockticks
  - **DSB Switches:** 2.1% of Clockticks
  - **Length Changing Prefixes:** 0.0% of Clockticks
  - **MS Switches:** 0.0% of Clockticks
- **Front-End Bandwidth:** 12.0% of Pipeline Slots

> This metric represents a fraction of slots during which CPU was stalled due to front-end bandwidth issues, such as inefficiencies in the instruction decoders or code restrictions for caching in the DSB (decoded uOps cache). In such cases, the front-end typically delivers a non-optimal amount of uOps to the back-end.

- **Front-End Bandwidth MITE:** 26.6% of Clockticks

> This metric represents a fraction of cycles during which CPU was stalled due to the MITE fetch pipeline issues, such as inefficiencies in the instruction decoders.

Front-End Bandwidth DSB: 6.4% of Clockticks
(Info) DSB Coverage: 42.5%

> Issue: A significant fraction of uOps was not delivered by the DSB (known as Decoded ICache or uOp Cache). This may happen if a hot code region is too large to fit into the DSB.
>
> Tips: Consider changing the code layout (for example, via profile-guided optimization) to help your hot regions fit into the DSB.
>
> See the "Optimization for Decoded ICache" section in the Intel 64 and IA-32 Architectures Optimization Reference Manual.

Bad Speculation: 15.4% of Pipeline Slots

> A significant proportion of pipeline slots containing useful work are being cancelled. This can be caused by mispredicting branches or by machine clears. Note that this metric value may be highlighted due to Branch Resteers issue.

Branch Mispredict: 15.4% of Pipeline Slots

> Issue:: A significant proportion of branches are mispredicted, leading to excessive wasted work or Backend stalls due to the machine need to recover its state from a speculative path.
>
> Tips:
>
> 1. Identify heavily mispredicted branches and consider making your algorithm more predictable or reducing the number of branches. You can add more work to 'if' statements and move them higher in the code flow for earlier execution. If using 'switch' or 'case' statements, put the most commonly executed cases first. Avoid using virtual function pointers for heavily executed calls.
>
> 2. Use profile-guided optimization in the compiler.

> See the Intel 64 and IA-32 Architectures Optimization Reference Manual for general strategies to address branch misprediction issues.

Machine Clears: 0.0% of Pipeline Slots
Back-End Bound: 9.8% of Pipeline Slots
Memory Bound: 2.2% of Pipeline Slots
L1 Bound: 6.4% of Clockticks
DTLB Overhead: 0.1% of Clockticks
Load STLB Hit: 0.0% of Clockticks
Load STLB Miss: 0.1% of Clockticks
Loads Blocked by Store Forwarding: 2.1% of Clockticks
Lock Latency: 0.0% of Clockticks
Split Loads: 0.0% of Clockticks
4K Aliasing: 0.9% of Clockticks
FB Full: 0.0% of Clockticks
L2 Bound: 0.0% of Clockticks
L3 Bound: 0.0% of Clockticks
Contested Accesses: 0.0% of Clockticks
Data Sharing: 0.0% of Clockticks
L3 Latency: 1.1% of Clockticks
SQ Full: 0.0% of Clockticks
DRAM Bound: 0.0% of Clockticks
Memory Bandwidth: 0.0% of Clockticks
Memory Latency: 2.1% of Clockticks
Store Bound: 0.0% of Clockticks
Store Latency: 4.9% of Clockticks
False Sharing: 0.0% of Clockticks
Split Stores: 0.2% of Clockticks
DTLB Store Overhead: 2.9% of Clockticks
Store STLB Hit: 2.9% of Clockticks
Store STLB Hit: 0.1% of Clockticks
Core Bound: 7.6% of Pipeline Slots
Divider: 0.0% of Clockticks
Port Utilization: 22.0% of Clockticks
Cycles of 0 Ports Utilized: 9.0% of Clockticks
Serializing Operations: 0.0% of Clockticks
Mixing Vectors: 0.0% of uOps
Cycles of 1 Port Utilized: 5.3% of Clockticks
Cycles of 2 Ports Utilized: 9.0% of Clockticks
Cycles of 3+ Ports Utilized: 27.1% of Clockticks
ALU Operation Utilization: 34.9% of Clockticks
Port 0: 29.8% of Clockticks
Port 1: 33.0% of Clockticks
Port 5: 33.0% of Clockticks
Port 6: 43.6% of Clockticks
Load Operation Utilization: 34.6% of Clockticks
Port 2: 41.5% of Clockticks
Port 3: 42.6% of Clockticks
Store Operation Utilization: 31.9% of Clockticks
Port 4: 31.9% of Clockticks
Port 7: 17.0% of Clockticks
Vector Capacity Usage (FPU): 25.0%
Average CPU Frequency: 2.181 GHz
Total Thread Count: 1

Paused Time: 0s

Effective Physical Core Utilization: 24.6% (0.985 out of 4)

> The metric value is low, which may signal a poor physical CPU cores utilization caused by:
>
> - load imbalance
> - threading runtime overhead
> - contended synchronization
> - thread/process underutilization
> - incorrect affinity that utilizes logical cores instead of physical cores
>
> Explore sub-metrics to estimate the efficiency of MPI and OpenMP parallelism or run the Locks and Waits analysis to identify parallel bottlenecks for other parallel runtimes.

Effective Logical Core Utilization: 12.3% (0.985 out of 8)

> The metric value is low, which may signal a poor logical CPU cores utilization. Consider improving physical core utilization as the first step and then look at opportunities to utilize logical cores, which in some cases can improve processor throughput and overall performance of multi-threaded applications.

Collection and Platform Info:

Application Command Line: ./codecs/hm/encoder/TAppEncoderStatic "-c" "./configs/hm/encoder_intra_main.cfg" "-i" "./sequences/CLASS_C/RaceHorses_416x240_30.yuv" "-wdt" "416" "-hgt" "240" "-b" "./bin/hm/encoder_intra_main.cfg/CLASS_C/RaceHorses_416x240_30_QP_22_hm.bin" "-o" "./rec_yuv/hm/encoder_intra_main.cfg/CLASS_C/RaceHorses_416x240_30_QP_22_hm.yuv" "-fr" "30" "-fs" "0" "-f" "2" "-q" "22"

User Name: root

Operating System: 5.4.0-65-generic DISTRIB_ID=Ubuntu DISTRIB_RELEASE=18.04 DISTRIB_CODENAME=bionic DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"

Computer Name: eimon

Result Size: 15.2 MB

Collection start time: 04:24:43 10/02/2021 UTC

Collection stop time: 04:24:45 10/02/2021 UTC

Collector Type: Event-based sampling driver

CPU:

- **Name:** Intel(R) Processor code named Kabylake ULX
- **Frequency:** 1.992 GHz
- **Logical CPU Count:** 8
- **Cache Allocation Technology:**
  - **Level 2 capability:** not detected
  - **Level 3 capability:** not detected