**READ ME SECTION**

# **Screenshot of Execution Outputs**

| **LEN** | **DEBUG** | **RELEASE** |
| --- | --- | --- |
| **25** |  |  |
| **215** |  |  |
| **220** |  |  |
| **226** |  |  |
| **230** |  |  |

# **Comparative Table of Results**

**Table I. Execution Time Comparison of ASUM Kernels in Debug Mode**

| **LEN** | **C** | **x86\_64** | **XMM** | **YMM** |
| --- | --- | --- | --- | --- |
| **25** | **0.000136** | **0.000066** | **0.000054** | **0.000090** |
| **215** | **0.062622** | **0.027978** | **0.015027** | **0.012432** |
| **220** | **2.144089** | **1.027747** | **0.567383** | **0.401646** |
| **226** | **139.446764** | **64.704342** | **38.476788** | **30.243802** |
| **230** | **2224.607902** | **1031.185912** | **616.090608** | **479.020490** |

**Table II. Speed Up Comparison of Assembly ASUM Kernels in Debug Mode**

| **LEN** | **C ➜ x86\_64** | **X86\_64 ➜ XMM** | **XMM ➜ YMM** |
| --- | --- | --- | --- |
| **25** | **206%** | **122%** | **60%** |
| **215** | **224%** | **186%** | **121%** |
| **220** | **209%** | **181%** | **141%** |
| **226** | **216%** | **168%** | **127%** |
| **230** | **216%** | **167%** | **129%** |

**\*\*\* Add analysis for debug**

**Table III. Execution Time Comparison of ASUM Kernels in Release Mode**

| **LEN** | **C** | **x86\_64** | **XMM** | **YMM** |
| --- | --- | --- | --- | --- |
| **25** | **0.000011** | **0.000070** | **0.000070** | **0.000038** |
| **215** | **0.000015** | **0.028412** | **0.016946** | **0.010207** |
| **220** | **0.000011** | **1.031012** | **0.519604** | **0.444464** |
| **226** | **0.000017** | **64.513735** | **38.840670** | **30.690565** |
| **230** | **0.000012** | **1033.780762** | **619.624279** | **473.207442** |

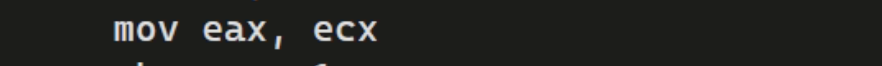
**Table IV. Speed Up Comparison of Assembly ASUM Kernels in Release Mode**

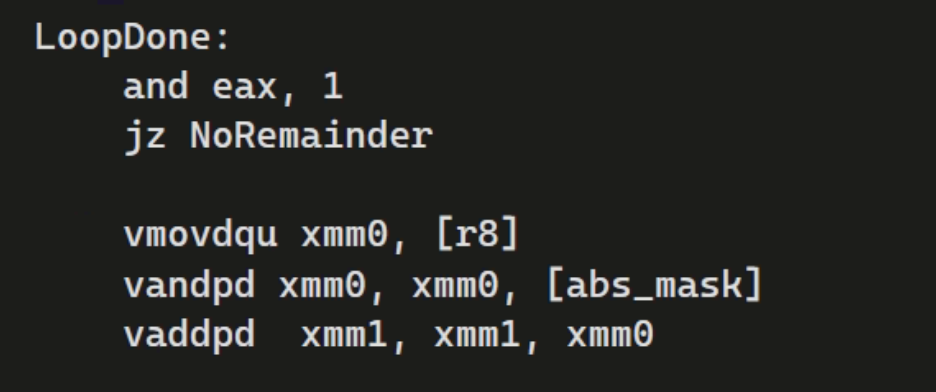
| **LEN** | **C ➜ x86\_64** | **X86\_64 ➜ XMM** | **XMM ➜ YMM** |
| --- | --- | --- | --- |
| **25** | **16%** | **100%** | **184%** |
| **215** | **0%** | **168%** | **166%** |
| **220** | **0%** | **198%** | **117%** |
| **226** | **0%** | **166%** | **127%** |
| **230** | **0%** | **167%** | **131%** |

**\*\*\* Add Analysis for Release**

# **Boundary Check**

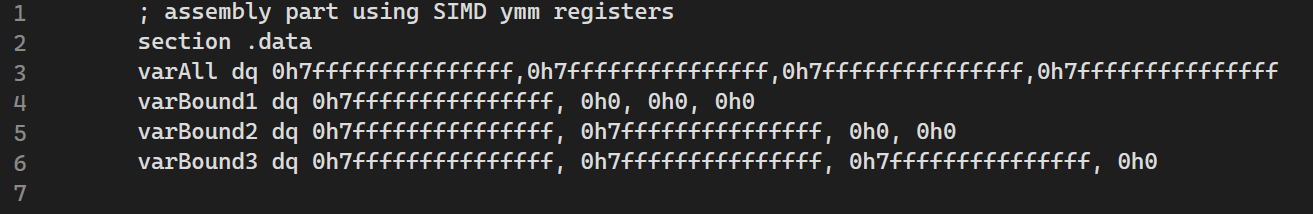
**XMM Boundary Checking**

****

****

**YMM Boundary Checking**

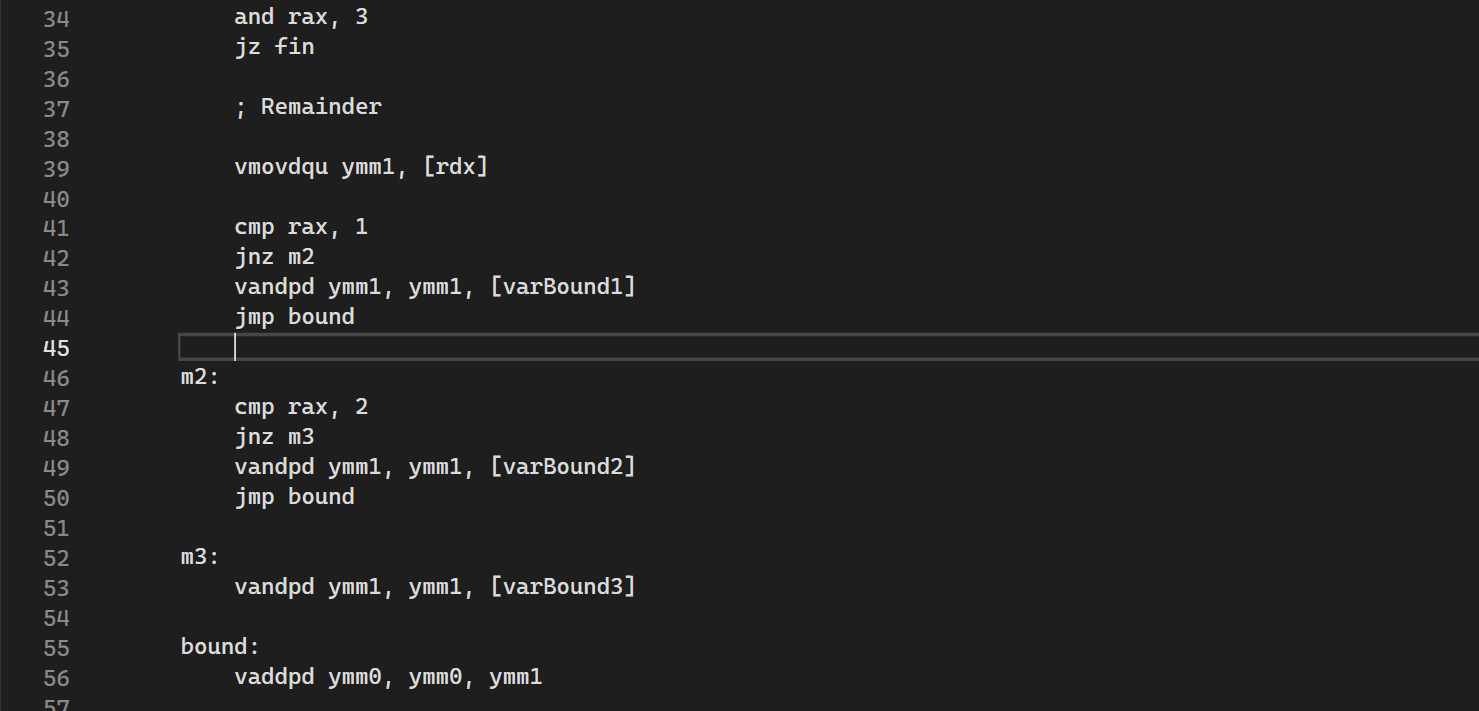
**Variables for boundary:**

****

**Setup of RAX as checker:**

****

**Boundary Checking after final loop:**

****

# **Others**

**JR - C**

* **No problems encountered.**
* **In debug mode, C has a significantly slower run time.**
* **In Release mode, C has a very fast execution time that leaves every kernel in the dust, regardless of array size. It exceeded my expectations.**
* **Thumbs up 5 stars.**

**Geena - XMM**

* **No problems encountered.**
* **Using AND for masking (absolute value)**
* **Check if odd or even using AND instead of doing modulo**

**Hans-x86-64**

* **No problems encountered**
* **C was faster compared to x86\_64 initially, even in debug mode.**
* **After applying some modifications, x86\_64 became faster marginally.**

**Lory-YMM**

* **No problems encountered**
* **Using AND and m256 instead of mov YMM, m256 yielded a marginally faster execution time**
* **When adding a buffer.asm that was supposed to test the speed of accessing the ASM file and returning back to C, did little effect in the execution time. We were expecting that the handshakes between C and ASM has a larger time, but buffer was not able to reflect that.**
* **Experimented with using r8 and rbx, instead of rdx for perusing the contents of the vector. Initial runs yielded a very fast execution time when using rbx on my machine, but after multiple executions, it suddenly went worse that the original execution. Eventually, rbx became unusable in Release mode. It might be an unexpected outcome initially from my machine, which eventually normalized when the rbx became unusable.**
* **I was expecting that YMM will be faster than XMM in execution time, but based on our runs, XMM sometimes become neck and neck with YMM’s execution time. It is possible that YMM can still optimized.**
* **For asum value of YMM and XMM, it starts to differ around datasets with a length of 2^20.**