



**Dedicated at the Lotus Feet of our Founder Chancellor,  
Bhagawan Sri Sathya Sai Baba**

# **Top-Down Microarchitectural Analysis of LLaMA 2 on Intel Skylake Processor**

By  
Roshan Prad  
Regd.No-23364  
II MTech CS  
Guide : Dr R Raghunath Sarma

# Motivation

1. Performance in Large Language environment is critical in resource constrained devices.
2. Understanding the software and hardware stacks and Identification of Bottlenecks in LLM layers is crucial for Increasing the performance of CPU

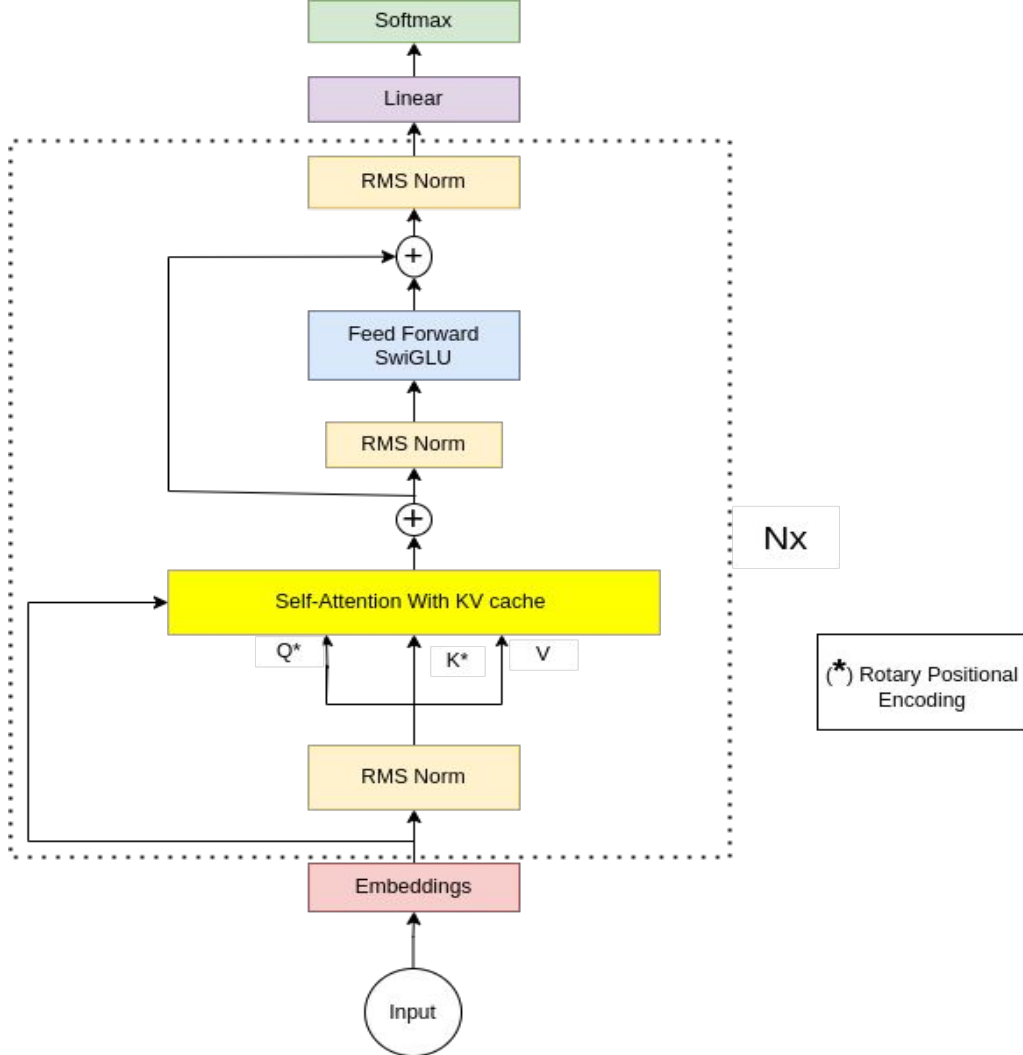
# **Problem Statement**

- 1. Study of LLaMA 2 Model Architectures**
- 2. Exploring Microarchitectural Opportunities to Improve the Performance of LLaMA 2 on the Skylake Processor**

# Features of LLaMa 2

- **Training Data:** Trained on 2 trillion tokens, a 40% increase over its predecessor.
- **Context Length:** Supports a context length of 4,096 tokens, doubling that of Llama 1.
- **Model Sizes:** Available in 7B, 13B, and 70B parameter sizes.
- **Capabilities:** Excels in tasks such as text generation, language translation, and coding assistance etc.
- **Release Date:** Released on July 18, 2023, in collaboration with Microsoft, Meta, and the open-source community.

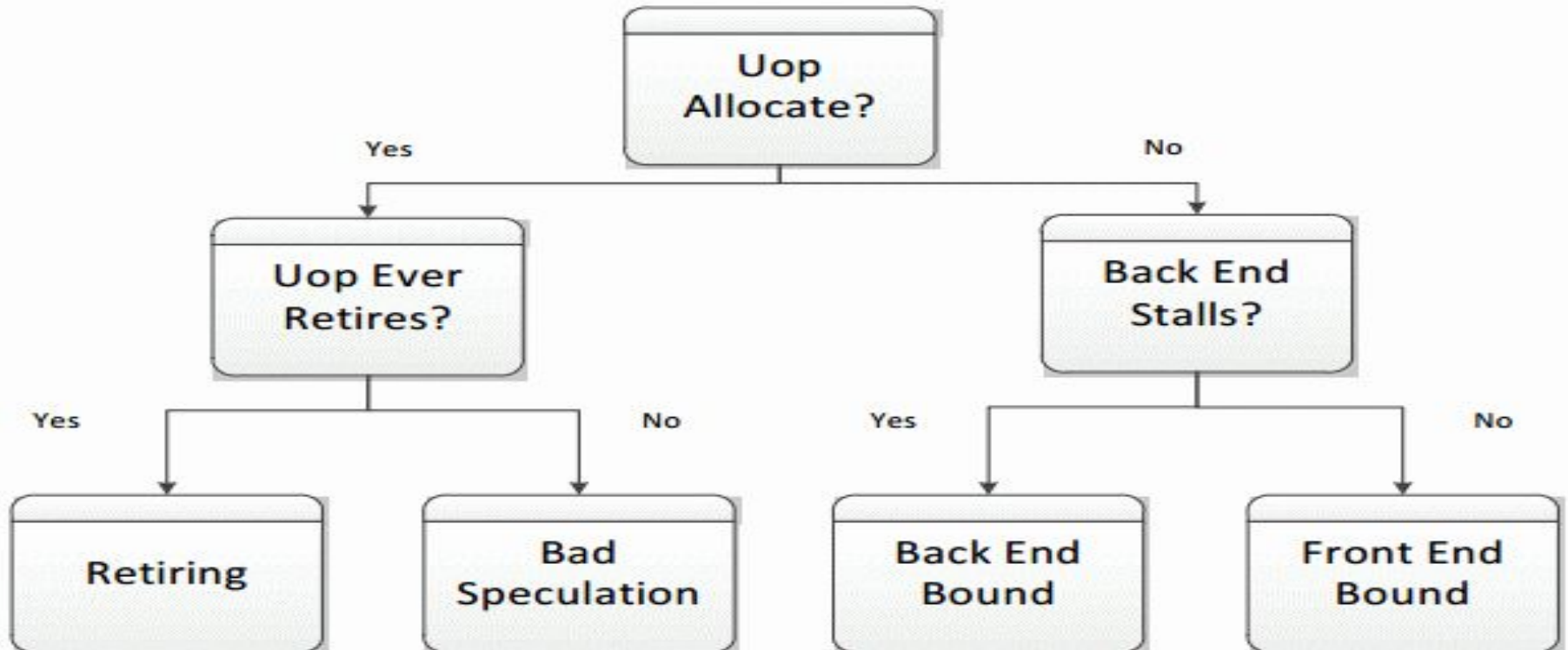
# Architecture of LLaMa 2



# Methodology and Profilers

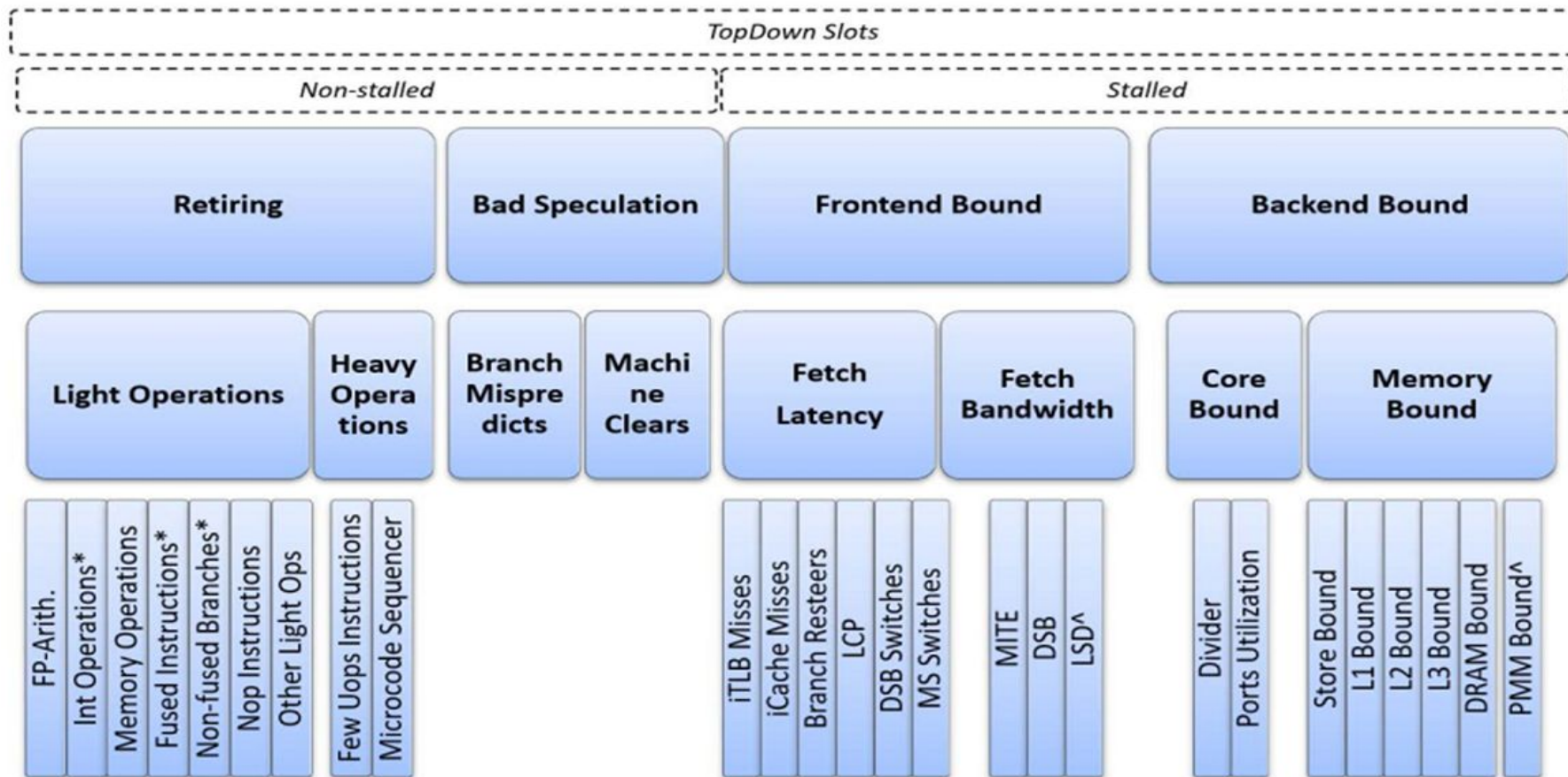
1. Top- Down Microarchitecture analysis (TMA) methodology.
2. Intel's Vtune profiler.

# Top-Down Microarchitecture Analysis





# TMA Hierarchy of Performance Bottlenecks



# Intel VTune Profiler

1. A performance profiling tool from Intel for Analyzing code on various Architectures(CPU, GPU).
2. CPU Metrics: CPU utilization, cache misses, branch misprediction, Port utilization etc.
3. Identify Hotspots, Analyze threading efficiency and memory usage

## Key Features:

- Detailed Performance analysis.
- Support for multiple hardware platform.
- Rich set of performance metrics

# Preliminary Setup

# LLM and Dataset details

Framework	Llama-cpp
Model	Llama 2 7B Inference
Dataset	llama-2-7b-chat.Q2_k.gguf
No. of Parameters	7 Billion
Size of Context window	4096

# Processor Features

Processor	Intel(R) Core i5-8350U CPU@1.70GHZ X86-64
Cache Sizes	L1 I : 128 KB L1 D : 128KB L2 : 1MB L3 : 8MB
OS	Pop! OS 22.04 Linux 6.6.6-76060606-generic kernel
Intel Vtune Profiler	2023.2.0

# **Related Work and Results**

# VTune Results - Hotspot Analysis

**87.4%** of CPU time is spend on execution of Vector Dot Product Kernel

## Top Hotspots

Function	Module	CPU Time <sup>?</sup>	% of CPU Time <sup>?</sup>
<a href="#">ggml_vec_dot_q3_K_q8_K</a>	libggml-cpu.so	448.984s	75.9%
<a href="#">ggml_vec_dot_q2_K_q8_K</a>	libggml-cpu.so	55.778s	9.4%
<a href="#">func@0x90d4</a>	libggml-cpu.so	37.439s	6.3%
<a href="#">func@0x2c0a0</a>	libggml-cpu.so	15.978s	2.7%
<a href="#">ggml_vec_dot_q6_K_q8_K</a>	libggml-cpu.so	13.599s	2.3%
[Others]	N/A*	19.837s	3.4%


# VTune Microarchitecture Analysis

Elapsed Time <sup>?</sup>: 218.552s > 

Clockticks: 650,573,000,000

Instructions Retired: 1,749,674,000,000

CPI Rate <sup>?</sup>: 0.372

➤ Retiring <sup>?</sup>: 71.8%  of Pipeline Slots

➤ Front-End Bound <sup>?</sup>: 1.6% of Pipeline Slots

➤ Bad Speculation <sup>?</sup>: 0.3% of Pipeline Slots

➤ Back-End Bound <sup>?</sup>: 26.3%  of Pipeline Slots




Average CPU Frequency <sup>?</sup>: 3.0 GHz

Total Thread Count: 1

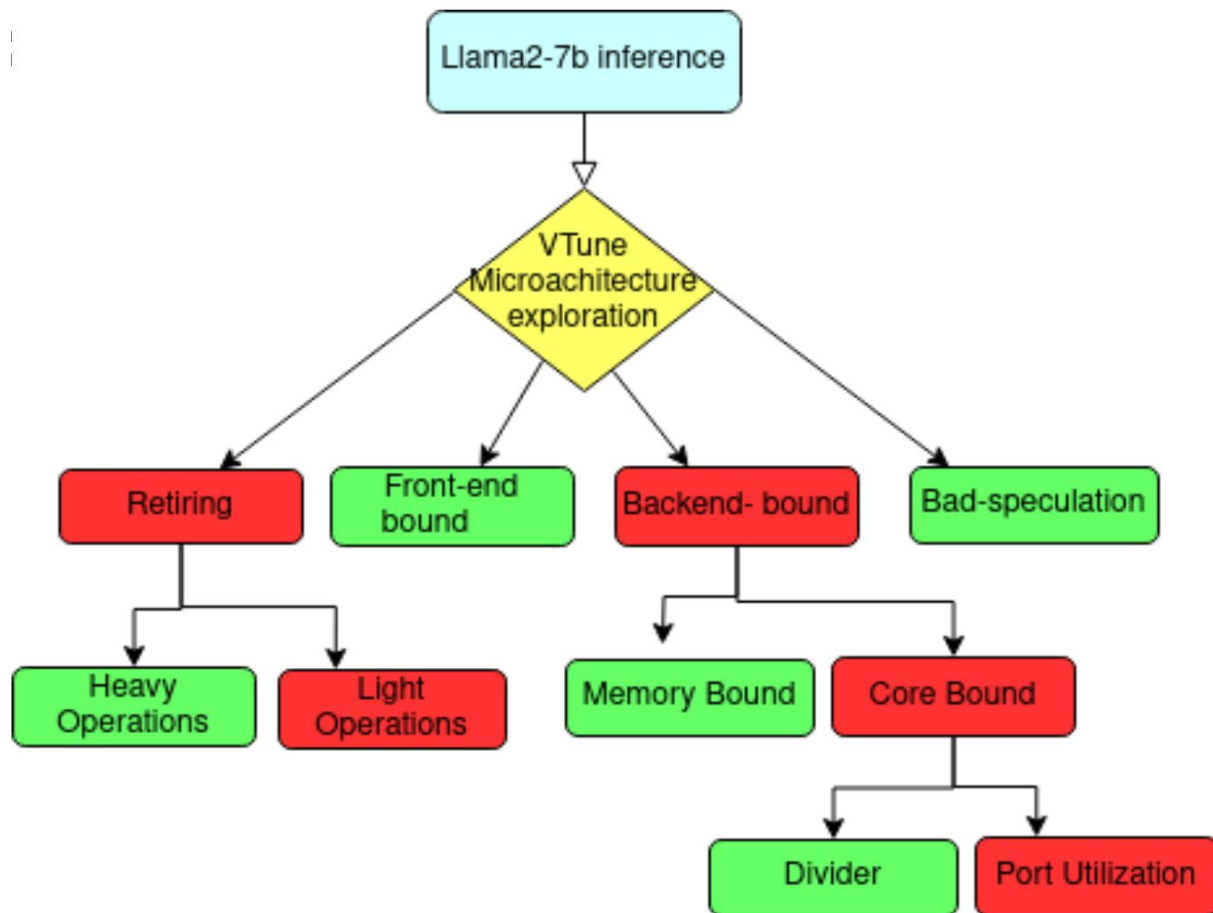
Paused Time <sup>?</sup>: 0s



## Elapsed Time <sup>?</sup>: 218.552s

Clockticks:	650,573,000,000	
Instructions Retired:	1,749,674,000,000	
CPI Rate <sup>?</sup> :	0.372	
Retiring <sup>?</sup> :	71.8% 	of Pipeline Slots
Front-End Bound <sup>?</sup> :	1.6%	of Pipeline Slots
Bad Speculation <sup>?</sup> :	0.3%	of Pipeline Slots
Back-End Bound <sup>?</sup> :	26.3% 	of Pipeline Slots
Memory Bound <sup>?</sup> :	15.6%	of Pipeline Slots
Core Bound <sup>?</sup> :	10.7% 	of Pipeline Slots
Divider <sup>?</sup> :	0.3%	of Clockticks
Port Utilization <sup>?</sup> :	10.2%	of Clockticks
Cycles of 0 Ports Utilized <sup>?</sup> :	0.8%	of Clockticks
Cycles of 1 Port Utilized <sup>?</sup> :	3.9%	of Clockticks
Cycles of 2 Ports Utilized <sup>?</sup> :	8.2%	of Clockticks
Cycles of 3+ Ports Utilized <sup>?</sup> :	71.3%	of Clockticks
ALU Operation Utilization <sup>?</sup> :	58.7%	of Clockticks
Port 0 <sup>?</sup> :	66.8%	of Clockticks
Port 1 <sup>?</sup> :	66.9%	of Clockticks
Port 5 <sup>?</sup> :	67.6%	of Clockticks
Port 6 <sup>?</sup> :	33.5%	of Clockticks
Load Operation Utilization <sup>?</sup> :	35.1%	of Clockticks
Port 2 <sup>?</sup> :	36.3%	of Clockticks
Port 3 <sup>?</sup> :	37.1%	of Clockticks
Store Operation Utilization <sup>?</sup> :	8.3%	of Clockticks
Port 4 <sup>?</sup> :	8.3%	of Clockticks
Port 7 <sup>?</sup> :	5.0%	of Clockticks
Vector Capacity Usage (FPU) <sup>?</sup> :	73.6%	

# VTune Results Summary



# Port Utilization

# CPU Executing 3+ $\mu$ ops per Cycle (71.3%)

## Potential Bottlenecks:

- **Raw Dependencies:** Instruction dependencies may limit parallel execution.
- **Execution Port Contention:** Limited execution ports may cause stalls.
- **ALU & Load Imbalance:** Uneven distribution of ALU and memory operations can reduce efficiency.

## Optimization Strategies:

- Improve **instruction scheduling** to reduce dependencies.
- Balance **execution port usage** for better throughput.
- Optimize **memory access patterns** to reduce stalls from cache/memory latency.

# Hot Loop Patterns

Address ▲	Source Line	Assembly		
0x42c40		Block 3:	0x42cb5	shr eax, 0x2
0x42c40		movzx eax, word ptr [rcx+0xc]	0x42cb8	shr esi, 0x4
0x42c44		vmovdqu ymm13, ymmword ptr [r9+0x4]	0x42cbb	or edx, r14d
0x42c4a		vmovss xmm0, dword ptr [r13+rax*4]	0x42cbe	and eax, 0x30303030
0x42c51		mov rax, qword ptr [rcx]	0x42cc3	and esi, 0xf0f0f0f
0x42c54		vmulss xmm3, xmm0, dword ptr [r9]	0x42cc9	vmovd r8d, xmm3
0x42c59		mov edx, eax	0x42cce	vmovss dword ptr [rsp+0x128], xmm3
0x42c5b		mov qword ptr [r11], rax	0x42cd7	or eax, esi
0x42c5e		mov rsi, rax	0x42cd9	vmovd xmm3, edx
0x42c61		mov eax, dword ptr [rcx+0x8]	0x42cdd	vpinsrd xmm2, xmm3, eax, 0x1
0x42c64		mov r14d, edx	0x42ce3	vpinsrd xmm1, xmm1, edi, 0x1
0x42c67		mov edi, eax	0x42ce9	vmovdqu ymm0, ymmword ptr [rcx-0x60]
0x42c69		shl edi, 0x4	0x42cee	vpunpcklqdq xmm1, xmm1, xmm2
0x42c6c		and r14d, 0xf0f0f0f	0x42cf2	vpaddb xmm1, xmm1, ymmword ptr [rsp+0x110]
0x42c73		and edi, 0x30303030	0x42cfb	vmovdqu ymm2, ymmword ptr [rcx-0x40]
0x42c79		or edi, r14d	0x42d00	vpandn ymm10, ymm0, ymmword ptr [rsp+0xe0]
0x42c7c		shr rsi, 0x20	0x42d09	vpandn ymm9, ymm0, ymmword ptr [rsp+0xc0]
0x42c80		vmovd xmm1, edi	0x42d12	vpandn ymm8, ymm0, ymmword ptr [rsp+0xa0]
0x42c84		mov r14d, esi	0x42d1b	vpandn ymm7, ymm0, ymmword ptr [rsp+0x80]
0x42c87		lea edi, ptr [rax*4]	0x42d24	vpmovsxbw ymm1, xmm1
0x42c8e		and r14d, 0xf0f0f0f	0x42d29	vmovdqu ymm12, ymmword ptr [r9+0x24]
0x42c95		and edi, 0x30303030	0x42d2f	vmovdqu ymm11, ymmword ptr [r9+0x44]
0x42c9b		or edi, r14d	0x42d35	vmovdqa xmm3, xmm1
0x42c9e		shr edx, 0x4	0x42d39	vpsrlw ymm6, ymm2, 0x2
0x42ca1		mov r14d, eax	0x42d3e	vpsrlw ymm5, ymm2, 0x4
0x42ca4		mov dword ptr [r11+0x8], eax	0x42d43	vpsrlw ymm4, ymm2, 0x6
0x42ca8		and edx, 0xf0f0f0f	0x42d48	vpsllw ymm10, ymm10, 0x2
0x42cae		and r14d, 0x30303030	0x42d4e	vpsrlw ymm9, ymm9, 0x1

0x42d54	vpsrlw ymm8, ymm8, 0x2	0x42ec0	vpsrlw ymm7, ymm7, ymm12
0x42d5a	vpsrlw ymm7, ymm7, 0x3	0x42ec2	vpshufb ymm13, ymm1, ymmword ptr [rip+0x1d4b5]
0x42d5f	vpand ymm2, ymm2, ymm14	0x42ecb	vpshufb ymm12, ymm1, ymmword ptr [rip+0x1d4cc]
0x42d64	vinerti128 ymm3, ymm3, xmm3, 0x1	0x42ed4	vpsubw ymm7, ymm7, ymm11
0x42d6a	vpaddubsw ymm10, ymm10, ymm13	0x42ed9	vpsubw ymm5, ymm5, ymm0
0x42d6f	vpaddubsw ymm2, ymm2, ymm13	0x42edd	vpshufb ymm11, ymm1, ymmword ptr [rip+0x1d4da]
0x42d74	vpsllw ymm9, ymm9, 0x2	0x42ee6	vpadd ymm0, ymm2, ymm3
0x42d7a	vpsllw ymm8, ymm8, 0x2	0x42eea	vpaddwd ymm4, ymm13, ymm4
0x42d80	vpsllw ymm7, ymm7, 0x2	0x42eee	vpaddwd ymm7, ymm12, ymm7
0x42d85	vpand ymm6, ymm6, ymm14	0x42ef2	vpsubw ymm6, ymm6, ymm8
0x42d8a	vpand ymm5, ymm5, ymm14	0x42ef7	vpshufb ymm1, ymm1, ymmword ptr [rip+0x1d4e0]
0x42d8f	vpand ymm4, ymm4, ymm14	0x42f00	vpadd ymm10, ymm10, ymm9
0x42d94	vpaddubsw ymm7, ymm7, ymmword ptr [r9+0x64]	0x42f05	vpaddwd ymm1, ymm1, ymm5
0x42d9a	vpaddubsw ymm4, ymm4, ymmword ptr [r9+0x64]	0x42f09	vpaddwd ymm6, ymm11, ymm6
0x42da0	vpaddubsw ymm9, ymm9, ymm12	0x42f0d	vpadd ymm0, ymm0, ymm10
0x42da5	vpaddubsw ymm8, ymm8, ymm11	0x42f12	vpadd ymm4, ymm4, ymm7
0x42daa	vpaddubsw ymm6, ymm6, ymm12	0x42f16	vpadd ymm6, ymm6, ymm1
0x42daf	vpaddubsw ymm5, ymm5, ymm11	0x42f1a	vpadd ymm0, ymm0, ymm4
0x42db4	vpsubw ymm2, ymm2, ymm10	0x42f1e	vmovd xmm3, r8d
0x42db9	vpshufb ymm11, ymm3, ymmword ptr [rip+0x1d5be]	0x42f23	vpadd ymm0, ymm0, ymm6
0x42dc2	vpaddwd ymm10, ymm11, ymm2	0x42f27	vbroadcastss ymm1, xmm3
0x42dc6	vpsubw ymm6, ymm6, ymm9	0x42f2c	inc r10d
0x42dcb	vpshufb ymm2, ymm3, ymmword ptr [rip+0x1d5cc]	0x42f2f	vcvt dq2ps ymm0, ymm0
0x42dd4	vpaddwd ymm9, ymm2, ymm6	0x42f33	add r9, 0x124
0x42dd8	vpsubw ymm4, ymm4, ymm7	0x42f3a	vfmadd231ps ymm15, ymm1, ymm0
0x42ddc	vpshufb ymm2, ymm3, ymmword ptr [rip+0x1d5db]	0x42f3f	add rcx, 0x6e
0x42de5	vpshufb ymm3, ymm3, ymmword ptr [rip+0x1d5f2]	0x42f43	cmp r12d, r10d
0x42dee	vandn ymm11, ymm0, ymmword ptr [rsi+0x40]	0x42f46	<a href="#">jnle 0x42c40 &lt;Block 3&gt;</a>

# Pipeline View for Hot Loop



# Instructions details

Instruction	Operands	No. of uOps	Execution Units	Port Number
MOVZX	R16, m	1	LOAD	P23
VPAND	V, v, M	2	ALU	P015 p23
VPMADDWD	Ymm, ymm, ymm	1	ALU	P01
VMOVDQU	Ymm, ymword ptr[rcx+0x20]	2	Load	P23

# Ports and Execution unit details

Skylake							
Port-0	Port-1	Port-2	Port-3	Port-4	Port-5	Port-6	Port-7
Execution Units	Execution Units	Execution Units	Execution Units	Execution Units	Execution Units	Execution Units	Execution Units
INT ALU	INT ALU	AGU	AGU	STORE	INT ALU	INT ALU	AGU
INT DIV	INT MUL	LOAD	LOAD		VEC	BRANCH	
IVEC MUL	IVEC ALU				SHUFFLE		
FP FMA	IVEC ALU				IVEC ALU		
AES	FP FMA				LEA		
VEC STR	Bit Scan						
FP DIV							
BRANCH							

# Grantt Chart for hotloop instructions

		cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10	cc11	cc12	cc13	cc14	cc15	cc16	cc17
1																		
2	movzx eax, word ptr [rcx+0xc]	issue	P2 1	P2 1	P2 1	P2 1	P2 1	retire										
3	vmovdqu ymm13, ymmword ptr [r9+0x4]	issue	P3	P3	P3	X	X	retire										
4	vmovss xmm0, dword ptr [r13+rax*4]	issue	SH	P2	P2	P2	X	retire										
5	mov rax, qword ptr [rcx]	issue	SH	P3	P3	X	X	retire										
6	vmulss xmm3, xmm0, dword ptr [r9]		issue	RAW	RAW	RAW	P1	P1	P1	P1	retire							
7	mov edx, eax		issue	P0	X	X	X	X	X	X	retire							
8	mov qword ptr [r11], rax		issue	RAW	RAW	P7 4	P7 4	X	X	X	retire							
9	mov rsi, rax		issue	RAW	RAW	P5	X	X	X	X	retire							
10	mov eax, dword ptr [rcx+0x8]			issue	P2	P2	X	X	X	X	X	retire						
11	mov r14d, edx			issue	P0	X	X	X	X	X	X	retire						
12	mov edi, eax			issue	P1	X	X	X	X	X	X	retire						
13	shl edi, 0x4			issue	RAW	P6	X	X	X	X	X	retire						
14	and r14d, 0xf0f0f0f				issue	P0	X	X	X	X	X	X	retire					
15	and edi, 0x30303030				issue	RAW	P0	X	X	X	X	X	retire					
16	or edi, r14d				issue	RAW	P5	X	X	X	X	X	retire					
17	shr rsi, 0x20				issue	RAW	P6	X	X	X	X	X	retire					
18	vmovd xmm1, edi					issue	RAW	P5	P5	X	X		X	retire				
19	mov r14d, esi					issue	SH	P0	X	X	X	X	X	retire				
20	lea edi, ptr [rax*4]					issue	SH	P1	RAW		X			retire				
21	and r14d, 0xf0f0f0f					issue	RAW	RAW	P1	X	X	X	X	retire				
22	and edi, 0x30303030						issue	RAW	RAW	RAW	P0	X	X	X	retire			
23	or edi, r14d						issue	RAW	RAW	RAW	RAW	P5	X	X	retire			
24	shr edx, 0x4						issue	P6	X	X	X	X	X	X	retire			
25	mov r14d, eax						issue	RAW	RAW	P5	X	X	X	X	retire			
26	mov dword ptr [r11+0x8], eax							issue	RAW	RAW	RAW	P2 4	P2 4	X	X	X	retire	
27	and edx, 0xf0f0f0f							issue	P0	X	X	X	X	X	X	X	retire	
28	and r14d, 0x30303030							issue	RAW	RAW	P1	X	X	X	X	retire		
29	shr eax, 0x2							issue	RAW	RAW	RAW	RAW	P6	X	X	X	retire	
30	shr esi, 0x4								issue	P0	X	X	X	X	X	X	X	retire
31	or edx, r14d								issue	P6	X	X	X	X	X	X	X	retire
32	and eax, 0x30303030								issue	RAW	RAW	RAW	RAW	P0	X	X	X	retire
33	and esi, 0xf0f0f0f								issue	RAW	P5	X	X	X	X	X	X	retire
34	vmovd r8d, xmm3									issue	P0	P0	X	X	X	X	X	retire
35	vmovss dword ptr [rsp+0x8], xmm3									issue	RAW	RAW	P7 4	P7 4	P7 4	X	X	retire

# Grantt Chart for optimized pipeline view (7.5% IPC gain)

		cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10	cc11	cc12	cc13	cc14	cc15	cc16	cc17	cc18
1																			
2	movzx eax, word ptr [rcx+0xc]	issue	P2 1	P2 1	P2 1	P2 1	P2 1	retire											
3	vmovdqu ymm13, ymmword ptr [r9+0x4]	issue	P3	P3	P3	X	X	retire											
4	vmovss xmm0, dword ptr [r13+rax*4]	issue	P8	P8	P8	X	X	retire											
5	mov rax, qword ptr [rcx]	issue	SH	P3	P3	X	X	retire											
6	vmulss xmm3, xmm0, dword ptr [r9]		issue	RAW	RAW	P1	P1	P1	P1	retire									
7	mov edx, eax		issue	P0	X	X	X	X	X	X	retire								
8	mov qword ptr [r11], rax		issue	RAW	RAW	P7 4	P7 4	X	X	X	retire								
9	mov rsi, rax		issue	RAW	RAW	P5	X	X	X	X	retire								
10	mov eax, dword ptr [rcx+0x8]			issue	P2	P2	X	X	X	X	X	retire							
11	mov r14d, edx			issue	P0	X	X	X	X	X	X	retire							
12	mov edi, eax			issue	P1	X	X	X	X	X	X	retire							
13	shl edi, 0x4			issue	RAW	P6	X	X	X	X	X	retire							
14	and r14d, 0xf0f0f0f				issue	P0	X	X	X	X	X	X	retire						
15	and edi, 0x30303030				issue	RAW	P0	X	X	X	X	X	retire						
16	or edi, r14d				issue	RAW	P5	X	X	X	X	X	retire						
17	shr rsi, 0x20				issue	RAW	P6	X	X	X	X	X	retire						
18	vmovd xmm1, edi					issue	RAW	P5	P5	X	X		X	retire					
19	mov r14d, esi					issue	P8	X	X	X	X	X	X	retire					
20	lea edi, ptr [rax*4]					issue	P1	X	X	X	X			retire					
21	and r14d, 0xf0f0f0f					issue	RAW	RAW	P1	X	X	X	X	retire					
22	and edi, 0x30303030						issue	RAW	RAW	RAW	P0	X	X	X	retire				
23	or edi, r14d						issue	RAW	RAW	RAW	RAW	P5	X	X	retire				
24	shr edx, 0x4						issue	P6	X	X	X	X	X	X	retire				
25	mov r14d, eax						issue	RAW	RAW	P5	X	X	X	X	retire				
26	mov dword ptr [r11+0x8], eax							issue	RAW	RAW	P2 4	P2 4	X	X	X	retire			
27	and edx, 0xf0f0f0f							issue	P0	X	X	X	X	X	X	retire			
28	and r14d, 0x30303030							issue	RAW	RAW	P1	X	X	X	X	retire			
29	shr eax, 0x2							issue	RAW	RAW	RAW	RAW	P6	X	X	retire			
30	shr esi, 0x4								issue	P0	X	X	X	X	X	X	retire		
31	or edx, r14d								issue	P6	X	X	X	X	X	X	retire		
32	and eax, 0x30303030									RAW	RAW	RAW	RAW	P0	X	X	retire		
33	and esi, 0xf0f0f0f								issue	RAW	P5	X	X	X	X	X	retire		
34	vmovd r8d, xmm3									issue	P0	P0	X	X	X	X	X	retire	
35	vmovss dword ptr [rsp+0x8], xmm3									issue	RAW	RAW	P7 4	P7 4	P7 4	X	X	retire	

# Overall IPC Gain

## IPC of hot loop

1.Total number of Instructions = 145

2.Numer of Clock Cycles = 58

3. IPC =  $145 / 58 = 2.5$

4. % of RAW stalls = 40

5. % of SH stalls = 9

1. IPC gain of 7.5 %



## IPC of hot loop

1.Total number of Instructions = 145

2.Numer of Clock Cycles = 54

3. IPC =  $145 / 54 = 2.69$

4. % of RAW stalls = 29

5. % of SH stalls = 3

# Conclusions

- **Primary Bottleneck:** Insufficient port utilization is the main bottleneck for the vector dot product kernel, which is crucial for the attention mechanism.
- **Optimization Strategy:** Introducing an extra unit can improve inference performance, as one iteration of the smallest loop contributes to 7% of the total execution time. Optimizing this can reduce inference time while increasing IPC.

# Future Work

- **Future Work on Multithreading:** Since multithreading and hyperthreading were disabled in the experiments, future work will enable these features, repeat the top-down microarchitecture analysis, and validate the findings.
- **Memory-Bound Performance Analysis:** While the memory-bound issue was not explicitly flagged, it had a significant impact. A detailed analysis, specifically focusing on DRAM-bound limitations, will be conducted.
- **Application to RISC-V Architectures:** The same methodology can be applied to RISC-V-based architectures like the SiFive 650, which are well-suited for AI and machine learning workloads.

# Acknowledgement

## Guide:

1. Dr. R Raghunatha Sarma, Associate Professor , mathematics and Computer science SSSIHL

## Mentors:

1. Mr. Dibyam Pradhan, Principal CPU Architect at ARM .
2. Mr Naveen M, Senior Member of Technical Staff, AMD.
3. Mr Mangala Prasad Sahu Hardware Developer at IBM
4. Mr Sai Aravind Graduate Engineer at ARM



# References

## Research Papers & Books

1. S. Eyerman et al., *"Top-Down Performance Analysis Methodology,"* IEEE Micro, 2019.
2. V. Heirman et al., *"Analyzing the Performance Bottlenecks of Deep Learning Workloads on CPUs,"* ACM Transactions on Architecture, 2022.
3. M. Pham et al., *"Efficient Transformer Inference: A Microarchitectural Perspective,"* NeurIPS, 2021.
4. A. Vaswani et al., *"Attention Is All You Need,"* NeurIPS, 2017.

## Tools & Documentation

5. **Intel VTune Profiler** – <https://www.intel.com/content/www/us/en/developer/tools/vtune-profiler/>
6. **Linux perf** – <https://perf.wiki.kernel.org/>
7. **Llama.cpp Documentation** – <https://github.com/ggerganov/llama.cpp>

## Processor & Architecture References

8. **Intel Skylake Microarchitecture** – Intel Developer Manuals
9. **SiFive 650 RISC-V Processor** – SiFive Documentation

## LLaMA Model References

10. **Meta AI Research**, *"LLaMA 2: Open-Source Large Language Models,"*  
<https://ai.meta.com/resources/models-and-libraries/llama-downloads/>
11. **Meta AI Research**, *"LLaMA: Open and Efficient Foundation Language Models,"*  
<https://research.facebook.com/publications/llama-open-and-efficient-foundation-language-models/>

**Thank You**

# Questions?