James Root
Project 1 Report
ECE 6100
17 February 2023

Report Purpose

This Project was created to design a VIPT (Virtually Indexed, Physically Tagged) cache with a TLB and HWIVPT. From this cache system, our goal is to find the most optimal configuration of the cache that minimizes both data storage and average access time (AAT) of an instruction.

Simulation Optimization

The cache parameters are as follows:

- **C:** the total size of the cache in 2^c bytes	(9 <= c <= 15)
- **B:** size of each L1 cache in 2^b	(4 <= b <=7)
- **S:** associativity of the cache	(s >= c – p)
- **P:** Page of size 2^P	(9 <= p <= min(14, c)
- **T:** TLB of 2^t	(0 <= t <= c – b – s)
- **M:** Memory with 2^m pages	(0 <= M <= min(32-P, 20))

To find an exhaustive list of AAT and cache size values, all values of C, B, S, P, T, and M should be iterated over. Despite this, there are some optimizations made to each of the restrictive ranges for variables. They are as follows:

- **C** can iterate beginning at a value of 12, based on initial simulations with C < 12 providing much less optimal AAT vs cache size.
- **P** will be set to begin at 12, as it is a function of c.
- **M** can be set at the highest possible value, min(32-P, 20), as the greater number of memory will reduce the expensive

This allows for our simulation space to operate over the following values:

- **C:** (12 <= c <= 15)
- **B:** (4 <= b <=7)
- **S:** (s >= 0)
- **P:** (12 <= p <= min(14, c)
- **T:** (0 <= t <= c – b - s)
- **M:** min(32-P, 20))

Simulation Formulas

There are 2 desired output parameters of out experiment to compare. The first is AAT, which is given as the output from running the executable (given parameters). This is calculated by the following formula(s):

$$AAT = HT + MR * MP$$

$$HT = (\text{L1 Array Lookup cost}) + (\text{TLB Hit Ratio}) * (\text{tag compare time}) + (\text{TLB Miss Ratio})$$
$$* ((\text{HWIVPT Penalty}) + (\text{tag compare time}) * (\text{HWIVPT Hit Ratio})$$

MR = L1 Miss Rate, MP = DRAM memory lookup

See project1_description.pdf *for more details*

As for calculating the total storage size, we have the following derivation:

$$\begin{aligned}
\text{Cache Size} &= \text{Size\_L1} + \text{Size\_TLB} + \text{Size\_HWIVPT}\\
&= (\text{Size\_L1\_TagStore} + \text{Size\_L1\_DataStore})\\
&\qquad + (\text{Num\_TLB\_entries} * (\text{size\_TLB\_entry}))\\
&\qquad + (\text{Num\_HWIVPT\_entries} * (\text{size\_HWIVPT\_entry}))\\
&= (2^{(C-B)} * (P+M-(C-S)+2) + 2^{C})\\
&\qquad + (2^{T} * (\text{PPN\_Size} + \text{VPN\_Size} + \text{valid}))\\
&\qquad + (2^{M} * (\text{PPN\_Size} + \text{VPN\_Size} + \text{valid}))
\end{aligned}$$

$$\begin{aligned}
\textit{Cache Size} &= (2^{(C-B)} * (P+M-(C-S)+2) + 2^{C})\\
&\qquad + (2^{T} * (P+M + 64-P + 1)\\
&\qquad + (2^{M} * (P+M + 64-P + 1)
\end{aligned}$$

This calculation is completed in an excel file with the given parameter values.

Experiment Overview

The testing of the VIPT cache will occur over the optimized range of parameter values given in **Simulation Optimizations**. There are 6 different trace files that will be tested for each possible parameter from the optimized list. Using different traces allows for a greater diversity of cache performance to be seen and better represents how a cache would operate in a system with multiple programs running. We use the following trace files to simulate the cache:

1. gcc.trace
2. leela.trace
3. linpack.trace
4. matmul_naive.trace
5. matmul_tiled.trace
6. mcf.trace

These configurations are simulated over a bash script and merged into a csv file, which with the help of Excel creates the following visualizations and conclusions.

Analyzing Cache Size v AAT for each trace

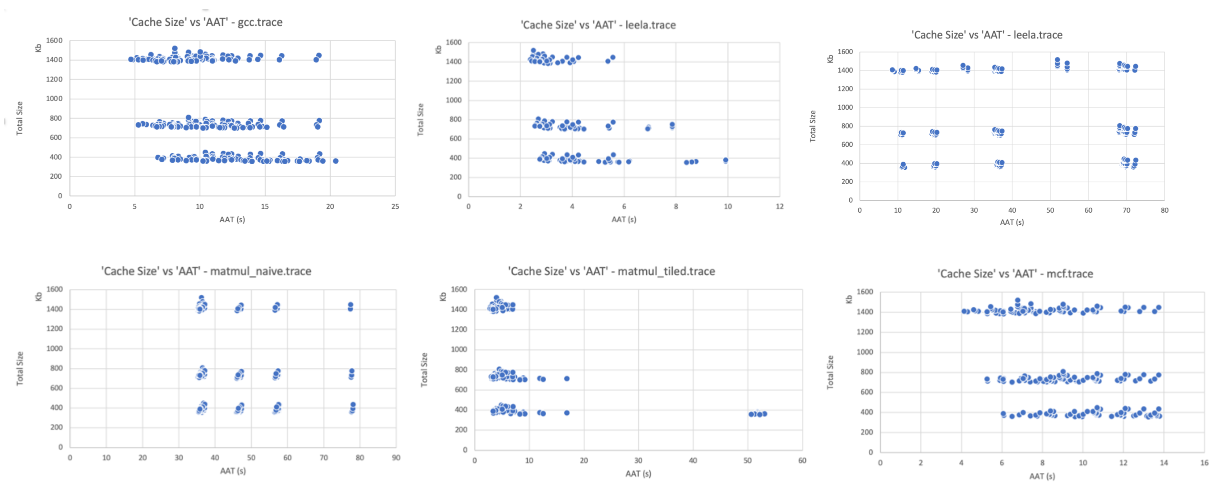

Figure 1: Plots of Cache Size (KB) vs Average Access time (s) for 6 different functions. *(Note: second appearance of leela.trace should be linpack.trace)*

There is a noticeable trend of 3 vertical groups of Total Size values, dependent on the possible M values of 18, 19, and 20. M is the determining factor for Total Size, as it produces so much byte storage in the HWIVPT (much more than any other unit provides). Generally, there is minimal difference in the AAT for the minimal setup of each vertical group. Only mcf and gcc have some difference, with gcc seeing a ~18% increase in AAT from P=20 to P=18 and mcf seeing a ~30% increase in AAT from P=20 to P=18. Meanwhile, the Total size reduces 70% for each trace from M=20 to M = 18. Seeing a reduction in M from 20 to 18, we reduce the number of comparators and the size of the HWIVPT by a factor of 4, causing less power consumption and of the system. We conclude *P = 14, M = 18, C >= 14.*

Now, we can look at the exact values of AAT for when P=14, M=18, C>=14. For each trace, below are the configurations that yield the shortest AAT:

Trace	Lowest AAT (s)	C	B	S	P	T	M
gcc.trace	4.1446	**15**	**7**	**1**	**14**	**7**	**18**
leela.trace	2.77685	**15**	**7**	**1**	**14**	**7**	**18**
		15	**7**	**1**	**14**	6	**18**
linpack.trace	35.34922	**15**	**7**	**1**	**14**	**7**	**18**
		15	**7**	**1**	**14**	6	**18**
matmul_naive.trace	8.55691	**15**	**7**	**1**	**14**	**7**	**18**
		15	**7**	**1**	**14**	6	**18**
		15	**7**	**1**	**14**	5	**18**
matmul_tiled.trace	2.39274	**15**	8	**1**	**14**	8	**18**

mcf.trace	4.70361	**15**	**7**	**1**	**14**	**7**	**18**

For each of the lowest times per trace (with the exception of matmul_tiled.trace, where the following config is the 2nd lowest AAT), we have a common config. This config is as follows: (**C=15, B=7, S=1, P=14, T=7, M=18**) and is determined to be the most optimal configuration for our cache simulator.