

# Projektbericht

Projektname: Cache-Simulator (kreativ)

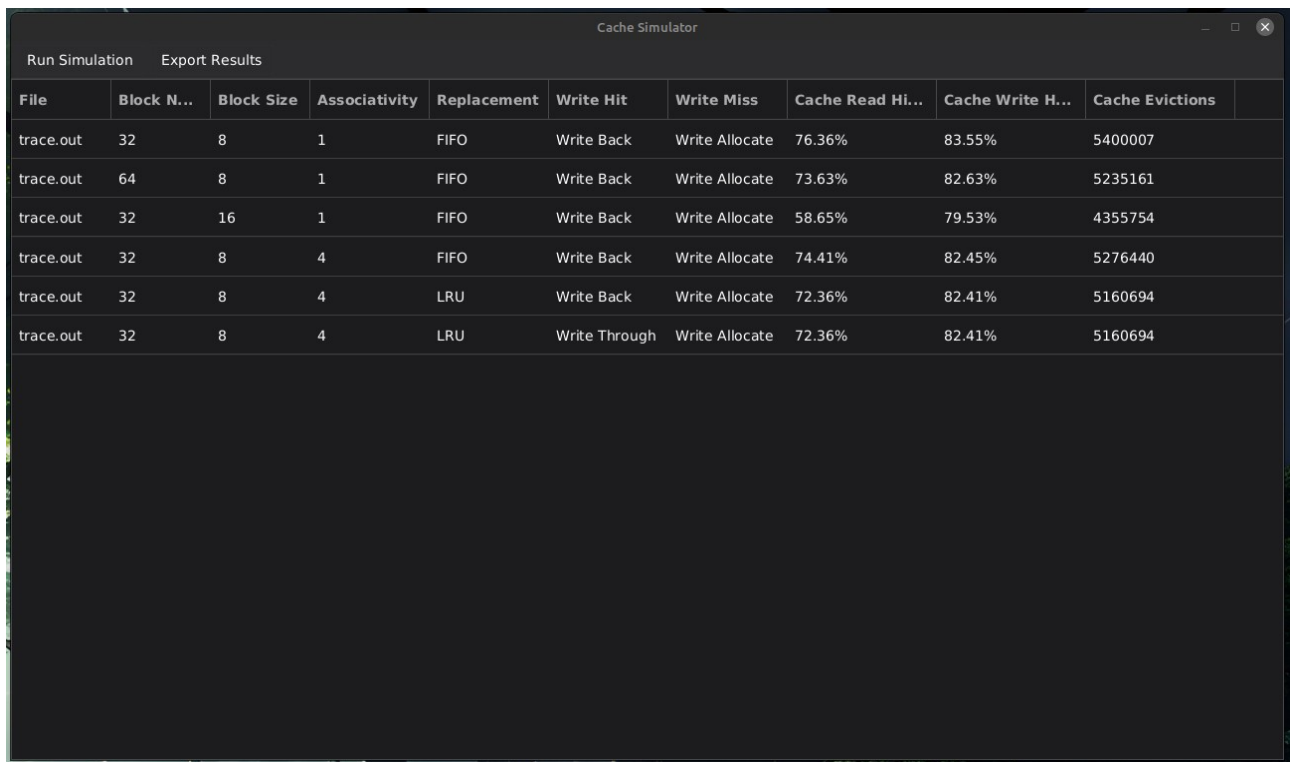
Projektcode: <https://github.com/SpinneKosinus1/Cache-Simulator>

## Überblick

Bei dem Projekt handelt es sich um einen Cache Simulator zur Berechnung der Cache Misses und der Cache Verdrängungen. Dabei sollten wir die Eingaben “Schreibrichtlinie”, “Ersetzungsrichtlinie”, “Anzahl der Cacheblöcke”, “Cacheblockgröße” und die Assoziativität zugelassen sein. Das Projekt ermöglicht die Untersuchung und Eingabe über ein entwickeltes UI zur Vereinfachung der Bedienung und der Auswertung. Die nun folgende Bewertung und Analyse der Ergebnisse. Ebenfalls ist es bei der Auswertung wichtig zu wissen, dass das Programm alle Zahlen nach der zweiten Nachkommastelle entfernt und es dabei somit zu kleineren Ungenauigkeiten bzw. Abweichungen kommen kann.

## Auswertung

Als erstes werden wir hier die Ergebnisse auswerten. Das ganze wird exemplarisch anhand der trace.out Datei gemacht. Ich werde häufiger das untere Screenshot erwähnen. Der erste Durchlauf entspricht in der Software die Standardeinstellung und wird als Referenz hergehalten. Dabei folgt die Bewertung der Ergebnisse später.



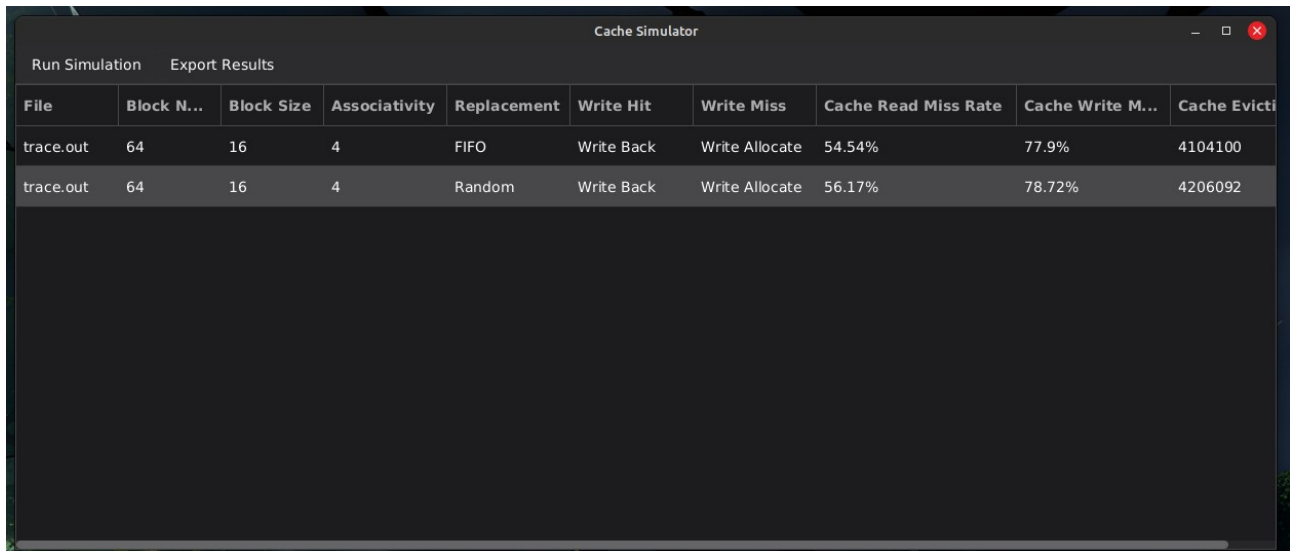
The screenshot shows the 'Cache Simulator' application window. It has a menu bar with 'Run Simulation' and 'Export Results'. Below the menu is a table with 11 columns: File, Block N..., Block Size, Associativity, Replacement, Write Hit, Write Miss, Cache Read Hl..., Cache Write H..., and Cache Evictions. The table contains six rows of data for different simulation configurations. The first row is the baseline configuration. The second row shows an increase in the number of cache blocks from 32 to 64, resulting in fewer cache misses. The third row shows an increase in block size from 8 to 16. The fourth and fifth rows show an increase in associativity from 1 to 4. The sixth row shows the effect of the replacement policy, comparing FIFO and LRU with write-through and write-back strategies.

File	Block N...	Block Size	Associativity	Replacement	Write Hit	Write Miss	Cache Read Hl...	Cache Write H...	Cache Evictions
trace.out	32	8	1	FIFO	Write Back	Write Allocate	76.36%	83.55%	5400007
trace.out	64	8	1	FIFO	Write Back	Write Allocate	73.63%	82.63%	5235161
trace.out	32	16	1	FIFO	Write Back	Write Allocate	58.65%	79.53%	4355754
trace.out	32	8	4	FIFO	Write Back	Write Allocate	74.41%	82.45%	5276440
trace.out	32	8	4	LRU	Write Back	Write Allocate	72.36%	82.41%	5160694
trace.out	32	8	4	LRU	Write Through	Write Allocate	72.36%	82.41%	5160694

Figure 1

Beim zweiten Durchlauf (zweite Zeile) wurde die Anzahl der Cache Blöcke erhöht. Dabei kann man sehen, dass sich die Cache Misses allgemein verringert haben. Folgend wurde im dritten

Durchlauf die Blockgröße erhöht, was ebenfalls ein ähnlichen Effekt aufwies. Der vierte Durchlauf hatte dann eine Erhöhung der Assoziativität auf sich, womit man ebenfalls die Werte verbessern konnte. Beim fünften Durchlauf hat sich durch die Umstellung auf „LRU“ die Werte ebenfalls verbessert, wobei die Einstellung „Random“ eine Verschlechterung ergab (siehe unten). Nur bei der letzten Einstellung, dem „Write Hit“, gab es keine Veränderungen für das Ergebnis. Die Einstellung bzgl. „Write Miss“ war nicht Teil der Aufgabenstellung und wird somit nicht weiter betrachtet.



Cache Simulator									
Run Simulation		Export Results							
File	Block N...	Block Size	Associativity	Replacement	Write Hit	Write Miss	Cache Read Miss Rate	Cache Write M...	Cache Evicti
trace.out	64	16	4	FIFO	Write Back	Write Allocate	54.54%	77.9%	4104100
trace.out	64	16	4	Random	Write Back	Write Allocate	56.17%	78.72%	4206092

Figure 2

Nach der kurzen Auflistung der Ergebnisse, werde ich diese Ihnen im nächsten Teil nun näher darauf eingehen.

## Bewertung bzw. Analyse

Nun werden die Ergebnisse im Kontext der Standardwerte bewertet. Am Ende gibt es dann abschließend ein Fazit für das Projekt.

Das durch die Erhöhung der Kapazität des Cachespeichers (Also die Anzahl der Cache Blöcke und die Größe der Blöcke) auch die Cache Misses sinkt ist logisch, da dadurch mehr Daten von dem Arbeitsspeicher in den Cache Platzfinden kann und sich somit die Chance erhöht, dass der Prozessor die benötigten Daten dort finden kann.

Cache Simulator										
Run Simulation		Export Results								
File	Block N...	Block Size	Associativity	Replacement	Write Hit	Write Miss	Cache Read Mi...	Cache Write M...	Cache Evictions	
trace.out	32	8	1	FIFO	Write Back	Write Allocate	76.36%	83.55%	5400007	
crafty_me...	32	8	1	FIFO	Write Back	Write Allocate	25.38%	11.91%	9053025	
trace.out	128	32	1	FIFO	Write Back	Write Allocate	42.41%	67.06%	3279081	
crafty_me...	128	32	1	FIFO	Write Back	Write Allocate	5.03%	0.95%	1717414	

Figure 3

Das durch die Erhöhung der Assoziativität die Werte für die trace.out Datei ein wenig besser werden (siehe Figure 1), scheint Ebenfalls so richtig, wobei es hier ein paar Einschränkungen gibt. Erstens nimmt der Effekt relativ zügig ab und zweitens ist z. B. die swim.trace Datei dabei (ohne Erhöhung der Cache Größe) kaum betroffen. Im Endeffekt ist die Wirksamkeit somit stark Programmabhängig. Wobei die Aussage im Kontext von dem Ersetzungsverfahren "FIFO" gilt. (Vergessen Sie die Kürzung nicht)

Cache Simulator										
Run Simulation		Export Results								
File	Block N...	Block Size	Associativity	Replacement	Write Hit	Write Miss	Cache Read Miss Rate	Cache Write M...	Cache Evictions	
swim.trace	128	8	1	FIFO	Write Back	Write Allocate	60.03%	100.0%	1086600	
swim.trace	128	8	2	FIFO	Write Back	Write Allocate	60.03%	100.0%	1086594	
swim.trace	128	8	4	FIFO	Write Back	Write Allocate	60.03%	100.0%	1086594	
swim.trace	128	64	1	FIFO	Write Back	No Write All...	7.51%	100.0%	81490	

Figure 4

Denn die Umstellung von FIFO zu LRU zeigt, dass die Effektivität von der Assoziativität von einer Umstellung Ebenfalls häufig eine Verbesserung nach sich zieht (außer hier bei swim.trace).

Cache Simulator										
Run Simulation		Export Results								
File	Block N...	Block Size	Associativity	Replacement	Write Hit	Write Miss	Cache Read Mi...	Cache Write M...	Cache Evictions	
swim.trace	64	16	4	FIFO	Write Back	Write Allocate	30.02%	50.03%	543544	
art.trace	64	16	4	FIFO	Write Back	Write Allocate	32.14%	13.29%	586716	
trace.out	64	16	4	FIFO	Write Back	Write Allocate	54.54%	77.9%	4104100	
swim.trace	64	16	4	LRU	Write Back	Write Allocate	30.02%	50.03%	543544	
art.trace	64	16	4	LRU	Write Back	Write Allocate	30.0%	13.29%	549602	
trace.out	64	16	4	LRU	Write Back	Write Allocate	53.73%	77.82%	4057213	
crafty_me...	32	8	4	FIFO	Write Back	Write Allocate	23.49%	10.94%	8371765	
crafty_me...	32	8	4	LRU	Write Back	Write Allocate	21.21%	9.56%	7543209	

Figure 5

Random dagegen kann ich kaum Bewerten, da es bei Durchläufen durch die Zufallsart immer unterschiedlich ausgeht. Allerdings ist es häufiger mal sogar schlechter als die anderen beiden Verfahren.

Das „Write Hit“ Verfahren hat selbst allerdings keinen Einfluss auf die drei Ergebnisse, denn das Verfahren ist vor allem im Hintergrund wichtig, wie der Simulator die Daten Schreiben soll. Somit ist das ganze wichtig für die Implementierung, allerdings beeinflusst das ganze nicht das Ergebnis.

## Fazit

Durch die obige Analyse bin ich der Auffassung, dass der Simulator im Prinzip funktioniert und seine Aufgabe erfolgreich Abschießen kann. Es ist hierbei allerdings wichtig, dass ich die Analyse anhand von kleinen Standardwerten vorgenommen habe. Es kann also sein, dass bei größeren Werten der Einfluss nochmals erhöht oder sinken kann.