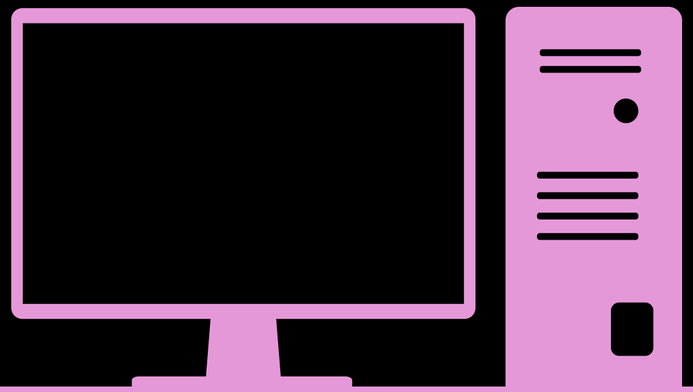
Grundlagenpraktikum:
Rechnerarchitektur

Systemdesign
Vertiefung

Cache Simulation und Analyse

Gruppe 165
Lie Aditya Bryan
Jovan Rio Tjandra
Aaron Rafael Thamin

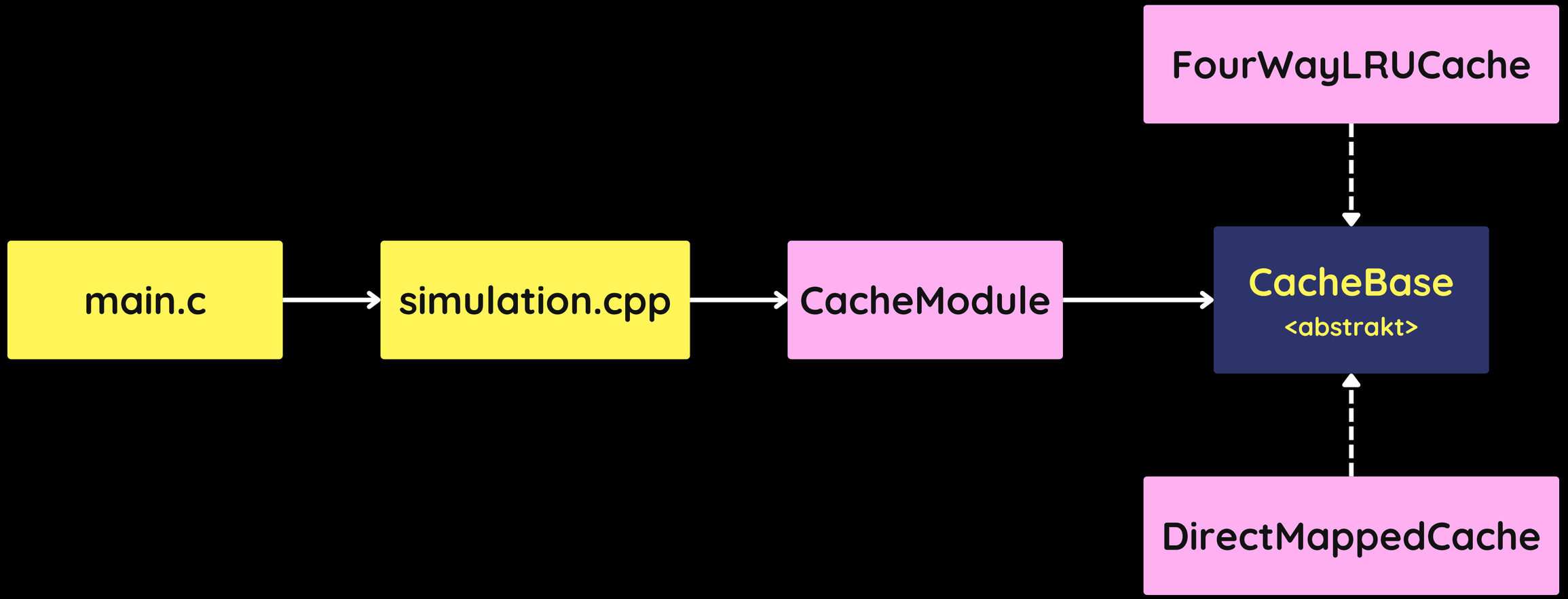


Direkt-abgebildeter und 4-fach-assoziativer Cache

Was sind die Unterschiede?
Welcher ist besser?

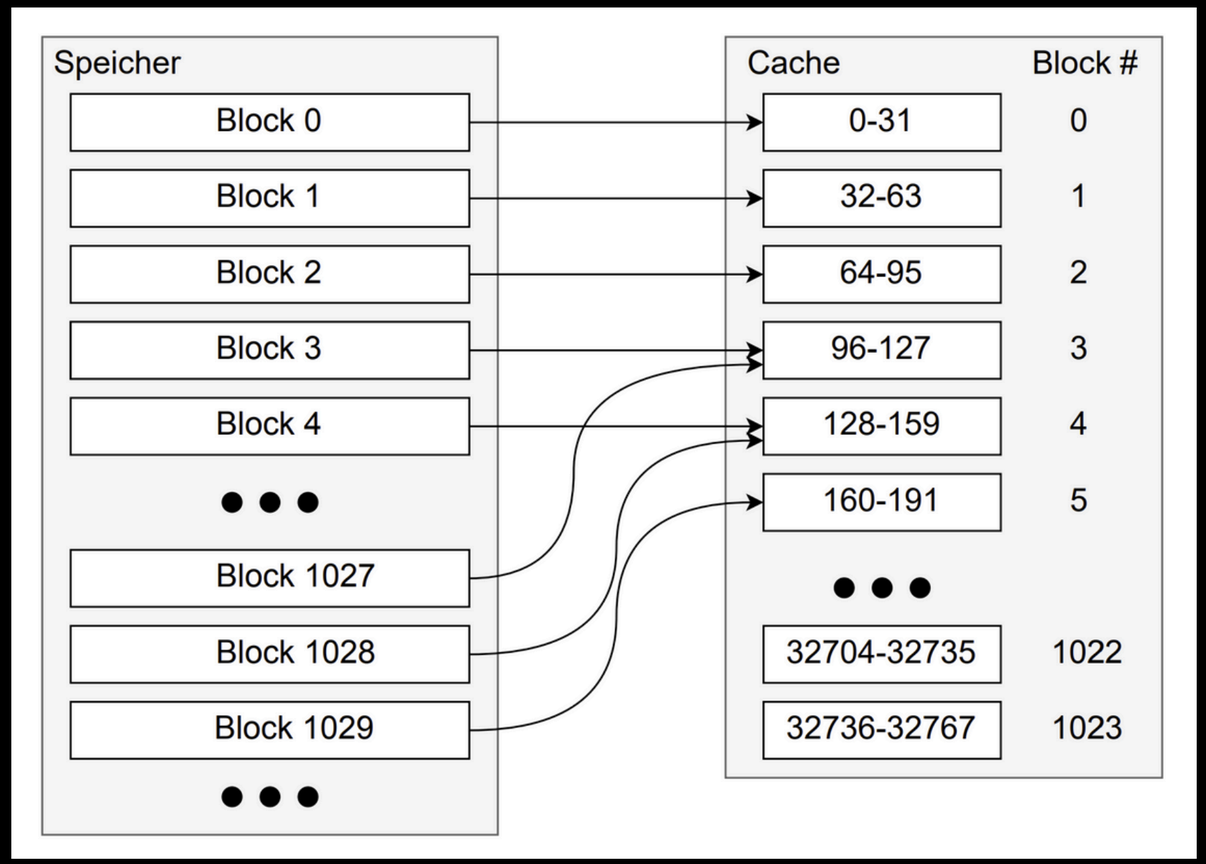
Inhaltsverzeichnis

Unsere Cacheimplementation



Cacheimplementation

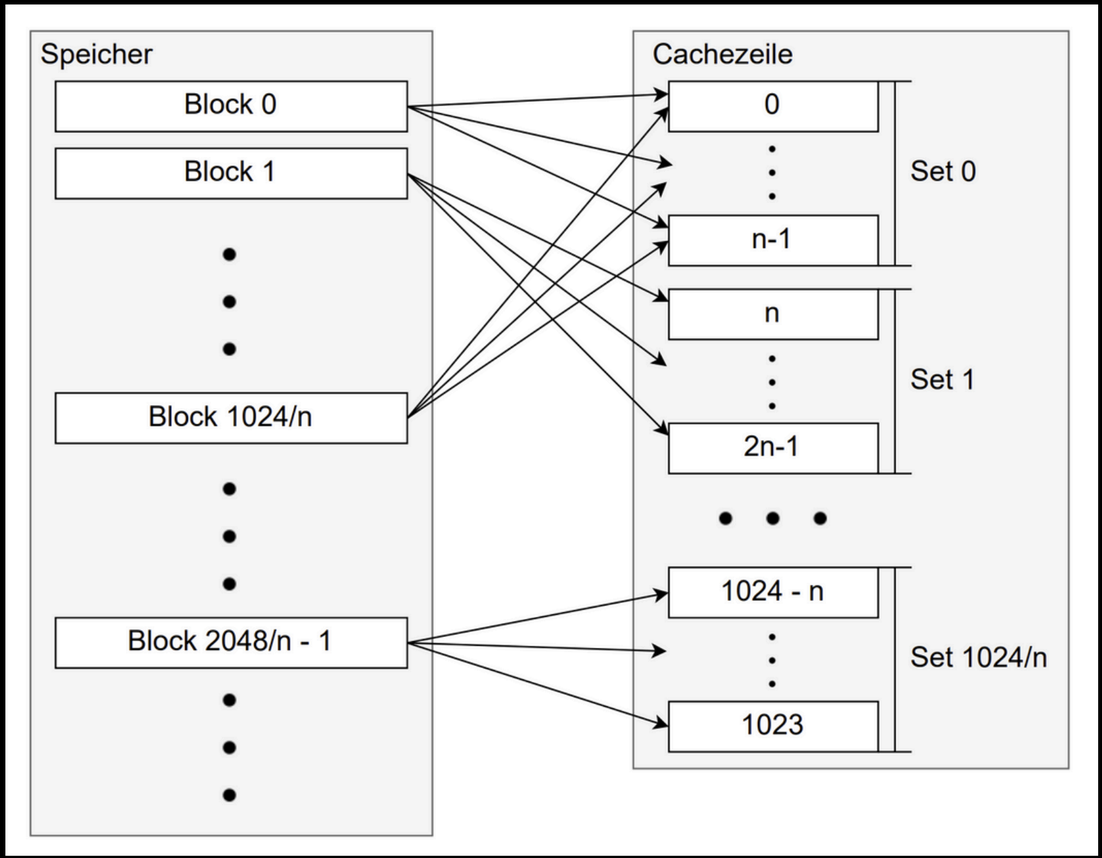
Direkt-abgebildet



Ein Block kann genau in einer Stelle im Cache abgelegt werden

Cacheimplementation

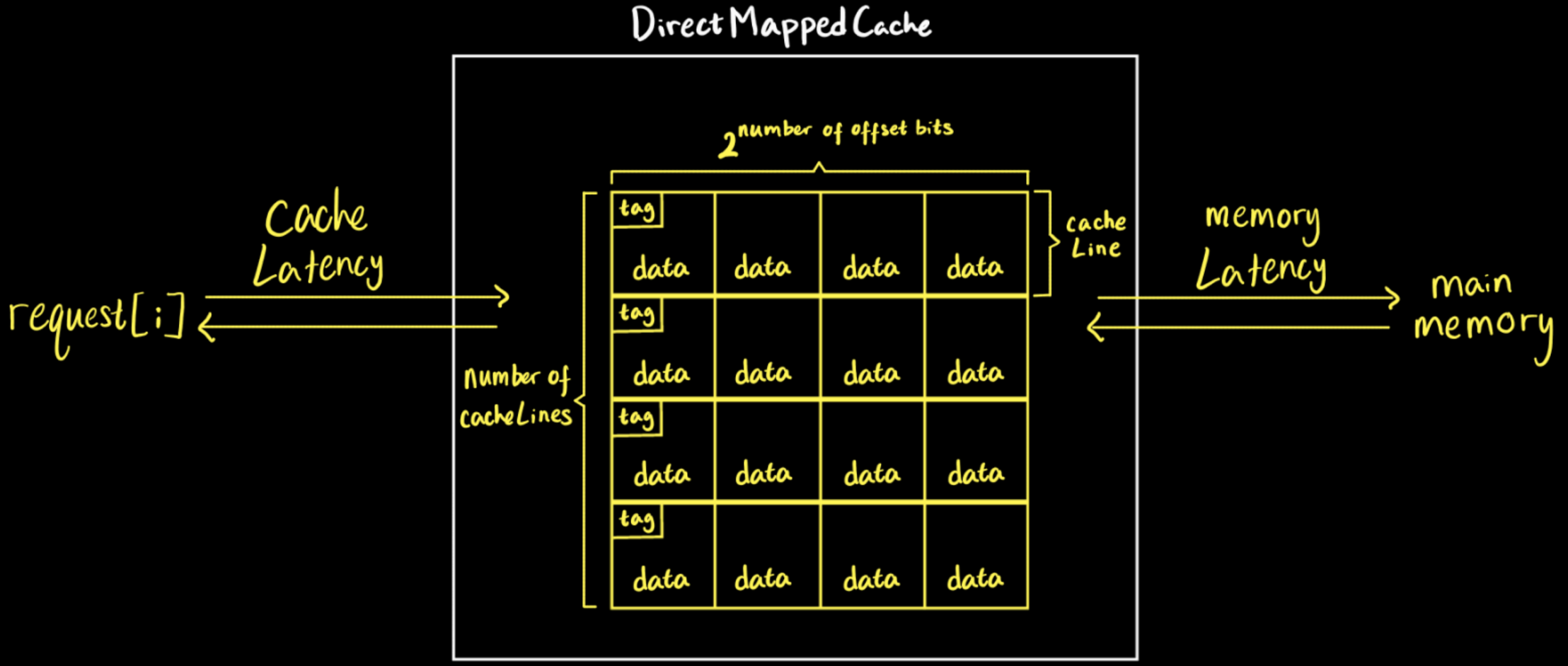
4-fach-assoziativ



Jede Adresse kann auf einen Teil der Cachezeilen abgebildet werden

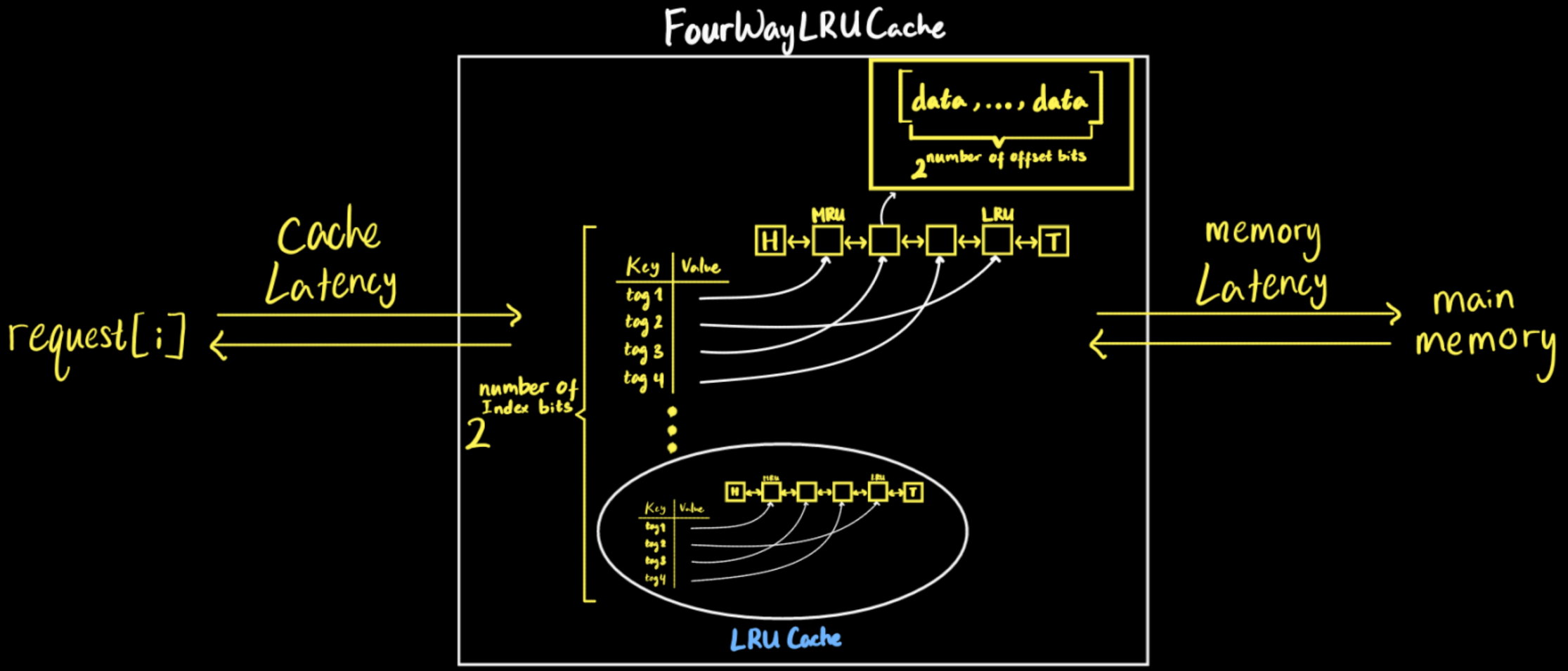
Unsere Cacheimplementation

Direkt-abgebildet



Unsere Cacheimplementation

4-fach-assoziativ



Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

$$A \times B = C$$

$$\begin{bmatrix} 3 & 7 & 4 & 12 \\ 6 & 18 & 8 & 1 \\ 5 & 23 & 3 & 41 \\ 29 & 17 & 5 & 1 \end{bmatrix} \times \begin{bmatrix} 19 & 13 & 49 & 22 \\ 4 & 21 & 37 & 34 \\ 50 & 0 & 8 & 14 \\ 26 & 7 & 13 & 0 \end{bmatrix} = \begin{bmatrix} 597 & 270 & 594 & 360 \\ 612 & 463 & 1037 & 856 \\ 1403 & 835 & 1653 & 934 \\ 895 & 741 & 2103 & 1286 \end{bmatrix}$$

$$\begin{bmatrix} 0x0 & 0x4 & 0x8 & 0xC \\ 0x10 & 0x14 & 0x18 & 0x1C \\ 0x20 & 0x24 & 0x28 & 0x2C \\ 0x30 & 0x34 & 0x38 & 0x3C \end{bmatrix} \quad \begin{bmatrix} 0x74 & 0x78 & 0x7C & 0x80 \\ 0x84 & 0x88 & 0x8C & 0x90 \\ 0x94 & 0x98 & 0x9C & 0xA0 \\ 0xA4 & 0xA8 & 0xAC & 0xB0 \end{bmatrix} \quad \begin{bmatrix} 0xC0 & 0xC4 & 0xC8 & 0xCC \\ 0xD0 & 0xD4 & 0xD8 & 0xDC \\ 0xE0 & 0xE4 & 0xE8 & 0xEC \\ 0xF0 & 0xF4 & 0xF8 & 0xFC \end{bmatrix}$$

Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

Initialisierung des Hauptspeichers mit dem Inhalt von Matrix A

```
W, 0x0, 3
W, 0x4, 7
W, 0x8, 4
W, 0xC, 12
W, 0x10, 6
W, 0x14, 18
W, 0x18, 8
W, 0x1C, 1
W, 0x20, 5
W, 0x24, 23
W, 0x28, 3
W, 0x2C, 41
W, 0x30, 29
W, 0x34, 17
W, 0x38, 5
W, 0x3C, 1
```

$$\begin{bmatrix} 3 & 7 & 4 & 12 \\ 6 & 18 & 8 & 1 \\ 5 & 23 & 3 & 41 \\ 29 & 17 & 5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0x0 & 0x4 & 0x8 & 0xC \\ 0x10 & 0x14 & 0x18 & 0x1C \\ 0x20 & 0x24 & 0x28 & 0x2C \\ 0x30 & 0x34 & 0x38 & 0x3C \end{bmatrix}$$

Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

Initialisierung des Hauptspeichers mit dem Inhalt von Matrix B

```
W, 0x74, 19
W, 0x78, 13
W, 0x7C, 49
W, 0x80, 22
W, 0x84, 4
W, 0x88, 21
W, 0x8C, 37
W, 0x90, 34
W, 0x94, 50
W, 0x98, 0
W, 0x9C, 8
W, 0xA0, 14
W, 0xA4, 26
W, 0xA8, 7
W, 0xAC, 13
W, 0xB0, 0
```

$$\begin{bmatrix} 19 & 13 & 49 & 22 \\ 4 & 21 & 37 & 34 \\ 50 & 0 & 8 & 14 \\ 26 & 7 & 13 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0x74 & 0x78 & 0x7C & 0x80 \\ 0x84 & 0x88 & 0x8C & 0x90 \\ 0x94 & 0x98 & 0x9C & 0xA0 \\ 0xA4 & 0xA8 & 0xAC & 0xB0 \end{bmatrix}$$

Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

Initialisierung der Lösungsmatrix C mit 0 als Platzhalter,
Lesen und Schreiben der Ergebnisse in C

```
W, 0xC0, 0
W, 0xC4, 0
W, 0xC8, 0
W, 0xCC, 0
W, 0xD0, 0
W, 0xD4, 0
W, 0xD8, 0
W, 0xDC, 0
W, 0xE0, 0
W, 0xE4, 0
W, 0xE8, 0
W, 0xEC, 0
W, 0xF0, 0
W, 0xF4, 0
W, 0xF8, 0
W, 0xFC, 0
```

```
R, 0x0,
R, 0x74,
R, 0xC0,
W, 0xC0, 57
R, 0x4,
R, 0x84,
R, 0xC0,
W, 0xC0, 85
R, 0x8,
R, 0x94,
R, 0xC0,
W, 0xC0, 285
R, 0xC,
R, 0xA4,
R, 0xC0,
W, 0xC0, 597
# usw.
```

$$\begin{bmatrix} 597 & 270 & 594 & 360 \\ 612 & 463 & 1037 & 856 \\ 1403 & 835 & 1653 & 934 \\ 895 & 741 & 2103 & 1286 \end{bmatrix}$$

$$\begin{bmatrix} 0xC0 & 0xC4 & 0xC8 & 0xCC \\ 0xD0 & 0xD4 & 0xD8 & 0xDC \\ 0xE0 & 0xE4 & 0xE8 & 0xEC \\ 0xF0 & 0xF4 & 0xF8 & 0xFC \end{bmatrix}$$

Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

Überprüfung der Simulation

Test-Case: Die Adressen zwischen der .csv-Datei und der manuell geschriebenen Datei in simulation.cpp stimmen nicht

```
uint32_t addressA[MATRIX_SIZE][MATRIX_SIZE] =
    {     0x4
        {0x0, 0x4, 0x8, 0xC},
        {0x10, 0x14, 0x18, 0x1C},
        {0x20, 0x24, 0x28, 0x2C},
        {0x30, 0x34, 0x38, 0x3C}
    }
```

```
Error in Matrix A: Wrote to the wrong address
Expected: 0 but got: 4
Error in Matrix A: Wrote to the wrong address
Expected: 0 but got: 4
Error in Matrix A: Wrote to the wrong address
Expected: 0 but got: 4
Error in Matrix A: Wrote to the wrong address
Expected: 0 but got: 4
```

Simulation mit Hilfe von SystemC

4 x 4 Matrixmultiplikation

Überprüfung der Simulation

Test-Case: Die Werte zwischen der .csv-Datei und der manuell geschriebenen Datei in simulation.cpp stimmen nicht

```
uint32_t B[MATRIX_SIZE][MATRIX_SIZE] =
    {                100
        {19, 13, 49, 22},
        {4, 21, 37, 34},
        {50, 0, 8, 14},
        {26, 7, 13, 0}
    }
```

```
Error in Matrix B: Something went wrong while storing data
Expected: 49 but got: 100
Error in Matrix B: Something went wrong while storing data
Expected: 49 but got: 100
Error in Matrix B: Something went wrong while storing data
Expected: 49 but got: 100
Error in Matrix B: Something went wrong while storing data
Expected: 49 but got: 100
Error in Matrix B: Something went wrong while storing data
Expected: 49 but got: 100
```

Überblick

Eingaben:

- Cycles: 3.000
- Cachelines: 4
- Cacheline size: 4 Bytes
- Cache latency: 2 Cycles
- Memory latency: 3 Cycles

```
Cycle 1393
Request address: 172 data: 0 WE: 0
Wait for cache latency!

Cycle 1394
Request address: 172 data: 0 WE: 0
Wait for cache latency!

Cycle 1395
Request address: 172 data: 0 WE: 0

Current data in cache
Offset 0: 1
Offset 1: 0
Offset 2: 0
Offset 3: 0

Replace data in cache from main memory
Address 172: 13
Address 173: 0
Address 174: 0
Address 175: 0
Wait for memory latency!
```

```
Cycle 1396
Request address: 172 data: 0 WE: 0
Wait for memory latency!

Cycle 1397
Request address: 172 data: 0 WE: 0
Wait for memory latency!

Cycle 1398
Request address: 172 data: 0 WE: 0
Data is ready: 13

Cycle 1399
Request address: 248 data: 0 WE: 0
Wait for cache latency!
```

4x4 Matrix Multiplikation

Anzahl der Cache-Misses - 11th Gen Intel(R) Core(TM) i7-1195G7

```
// Perform matrix multiplication (SIZE = 4)
for (int i = 0; i < SIZE; ++i) {
    for (int j = 0; j < SIZE; ++j) {
        for (int k = 0; k < SIZE; ++k) {
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}
```

Cache-Misses = ~68%

```
Performance counter stats for './main':

          16,015      cache-references:u
          10,929      cache-misses:u                   #   68.242 % of all cache refs

      0.016146630 seconds time elapsed

      0.004253000 seconds user
      0.000000000 seconds sys
```

Endergebnis

Direkt-abgebildet

Eingaben:

- Cycles: 10.000
- Cachelines: 8
- Cacheline size: 8 Bytes
- Cache latency: 2 Cycles
- Memory latency: 4 Cycles

```
Cycles: 1468
Misses: 139
Hits: 165
Primitive Gate Count: 2248
```

Cache-Misses

Realität = ~68%

Unsere Simulation: ~46%

Endergebnis

4-fach-assoziativ

Eingaben:

- Cycles: 10.000
- Cachelines: 8
- Cacheline size: 8 Bytes
- Cache latency: 2 Cycles
- Memory latency: 4 Cycles

```
Cycles: 1196
Misses: 71
Hits: 233
Primitive Gate Count: 2920
```

Cache-Misses

Realität = ~68%

Unsere Simulation: ~23%

Endergebnis

Anzahl von Cachezeilen erhöhen

Eingaben:

- Cycles: 10.000
- Cachelines: ~~8~~ 16
- Cacheline size: 8 Bytes
- Cache latency: 2 Cycles
- Memory latency: 4 Cycles

Direkt-abgebildet

Vorher

```
Cycles: 1468
Misses: 139
Hits: 165
Primitive Gate Count: 2248
```

Nachher

```
Cycles: 1164
Misses: 63
Hits: 241
Primitive Gate Count: 4464
```

Endergebnis

Unterschiedliche Latenz

Eingaben:

- Cycles: 10.000
- Cachelines: 8
- Cacheline size: 8 Bytes
- Cache latency: ~~2~~ 5 Cycles
- Memory latency: ~~4~~ 40 Cycles

Direkt-abgebildet

Vorher

```
Cycles: 1468
Misses: 139
Hits: 165
Primitive Gate Count: 2248
```

Nachher

```
Cycles: 7384
Misses: 139
Hits: 165
Primitive Gate Count: 2248
```

Endergebnis

Zu wenig Zyklen

Eingaben:

- Cycles: 10~~000~~ 100
- Cachelines: 8
- Cacheline size: 8 Bytes
- Cache latency: 2 Cycles
- Memory latency: 4 Cycles

Direkt-abgebildet

Vorher

```
Cycles: 1468
Misses: 139
Hits: 165
Primitive Gate Count: 2248
```

Nachher

```
Cycles: 18446744073709551614
Misses: 11
Hits: 9
Primitive Gate Count: 2248
```

Schaltkreisanalyse

Benötigte Gatteranzahl:

1-bit Speicher = 4 Gatter

Der ganze Speicher = Anzahl Cachezeilen x Cachezeilengröße x 1-bit Speicher x 8

Control Logic = 5 x Anzahl Cachezeilen

Tag-Comparator = 2 x Anzahl Tag-Bits x Anzahl Cachezeilen

Summe = ganzer Speicher + Control Logic + Tag-Comparator

Zusätzlich für 4-fach:

2-bit Speicher für Zähler = 2 x 1-Bit Speicher x Anzahl Cachezeilen

Comparator = 2-bit Zähler x 2

Update logic = 2-bit Zähler x 7

LRU = 2-bit Zähler + Comparator + Update Logic

Summe für 4-fach = Summe + LRU

Zusammenfassung

Direkt-assoziativer und 4-fach-assoziativer Cache
Welcher ist besser?

Direkt-abgebildeter Cache ist schneller als 4-fach-assoziativer Cache

4-fach-assoziativer Cache hat weniger Misses als direkt-abgebildeter Cache

4-fach-assoziativer Cache wird heutzutage als direkt-abgebildeter bevorzugt

Verbesserung

MainMemory und die andere Cacheimplementationen könnten auch die SC_MODULE implementieren.

Beispiele in der .csv-Datei und die Tests in simulation.cpp für Matrixmultiplikation sind unflexibel.

Die Adresse in unserem Beispiel sind nicht weit genug verteilt, dass unsere Simulation viel weniger Misses als in der Realität hat.

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?