CT Praktikum: Cache

1 Einleitung

In diesem Praktikum lernen Sie die Funktionsweise eines **Direct-Mapped-Cache** kennen. Abbildung 1 zeigt die schematische Darstellung eines solchen Cache. Anhand eines einfachen Programmes mit verschachtelter Schleife untersuchen Sie die Speicherzugriffe auf den Cache und optimieren diese durch Verändern der Cache-Parameter. Dazu wird der Cache auf dem CT-Board simuliert.

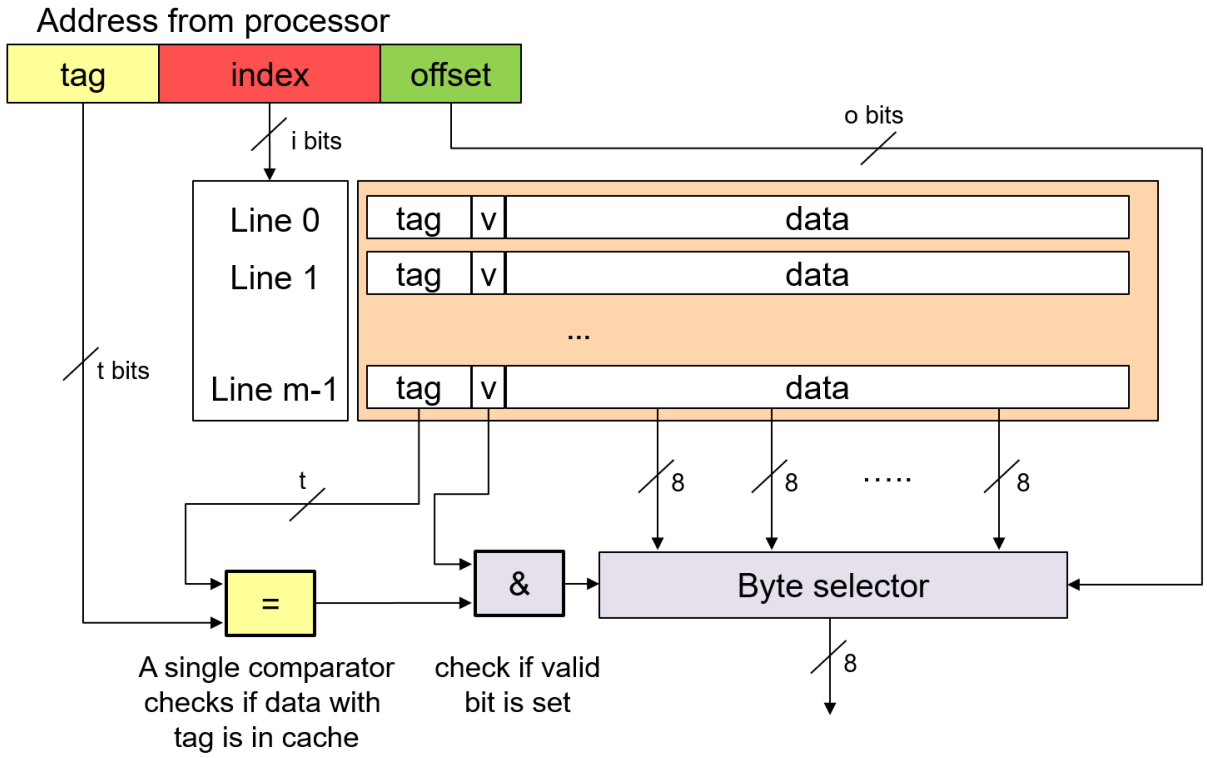

Abbildung 1: Schematische Darstellung eines Direct-Mapped-Cache

2 Lernziele

- Sie verstehen, wie ein Direct-Mapped-Cache aufgebaut ist.
- Sie können die Grössen Tag, Index und Offset erklären.
- Sie können die Hit-Rate eines Cache optimieren.

3 Aufgaben

3.1 Speicherzugriffe bei Arrays

Für unsere Analyse verwenden wir den untenstehenden Beispielcode zur Summation von zwei Arrays:

```
uint8_t a[ARRAY_ROWS][ARRAY_COLUMNS];
uint8_t b[ARRAY_ROWS][ARRAY_COLUMNS];
uint8_t c[ARRAY_ROWS][ARRAY_COLUMNS];

/* Loop through columns */
for (int j = 0; j < ARRAY_COLUMNS; j++) {

    /* Loop through rows */
    for (int i = 0; i < ARRAY_ROWS; i++) {
        a[i][j] = b[i][j] + c[i][j]
    }
}
```

Für diese Aufgabe verwenden wir folgende Konfigurationen der Arrays:

Anzahl Zeilen `[ARRAY_ROWS]`	5
Anzahl Spalten `[ARRAY_COLUMNS]`	10
Grösse eines Elements `[uint8_t]`	1 Byte
Startadresse des ersten Arrays, (a)	0x0000'0000

3.1.1 Speicheradressen

Die Arrays werden, wie im Code oben, hintereinander definiert. Tragen Sie die Adressen der Elemente in der untenstehenden Tabelle ein.

Element	Adresse des Elements im Speicher [HEX]
a[0][0]	
a[0][1]	
a[0][2]	
a[0][9]	
a[1][0]	
a[1][1]	
a[4][9]	
b[0][0]	
b[4][9]	
c[0][0]	
c[4][9]	

3.1.2 Dimensionen RAM / Cache

Für die Simulation verwenden wir folgende Parameter für den Cache:

Anzahl Bits für die Adressierung	t + i + o = 11
Anzahl Bits für den Offset	o = 2
Anzahl Bits für den Index	i = 2
Anzahl Bits für den Tag	t = 7

Basierend auf Abbildung 1: Bestimmen Sie folgende Dimensionen des RAM und des Cache:

Grösse des RAM in Bytes (Memory Size)	
Anzahl Zeilen im Cache	
Grösse einer Cache-Zeile in Bytes (nur Nutzdaten)	
Grösse des Cache in Bytes (Cache Size, nur Nutzdaten)	

Vervollständigen Sie die folgenden Aussagen für einen Direct-Mapped-Cache:

_____ bestimmt die Anzahl Zeilen im Cache.

_____ bestimmt die Anzahl Bytes in einer Cache Zeile.

Beim Direct-mapped-Cache wird jeder Memory Block, abhängig von seiner Adresse, genau einer Cache Zeile zugeordnet.

_____ einer Memory Adresse bestimmt, in welcher Cache Zeile diese Memory Adresse gespeichert wird.

_____ zeigt, welcher Memory Block sich gerade in der Cache Zeile befindet.

3.2 Simulation

In dieser Aufgabe werden die Speicherzugriffe der vorherigen Aufgabe auf dem CT-Board simuliert.

Die Funktionen des Cache sind im Header-File **`cache.h`** zu finden. Welche Funktionen gibt es und was machen sie? Analysieren Sie auch das File **`cache.c`**

Kontrollieren Sie im Header-File **`config.h`**, dass die Konfigurationen des Cache und der Arrays. mit der vorherigen Aufgabe übereinstimmen.

Schauen Sie sich nun die Funktion **`run_simulation(void)`** im File **`main.c`** an. Wodurch unterscheidet sich der Code der Simulation vom Beispielcode der vorherigen Aufgabe?

Warum unterscheidet sich der Code der Simulation mit dem des Beispiels?

Hinweis: Sie können die Simulation mit den Tasten T0 und T1 steuern (T1 -> Single Step / T0 gedrückt halten -> fortlaufend).

Die Simulation ist erfolgreich durchgelaufen, wenn das LCD blau leuchtet die Anzahl der Hits und Misses angezeigt werden

Kompilieren Sie nun das Programm und laden Sie es auf das CT-Board. Lassen Sie zuerst das Programm schrittweise durch Drücken der Taste T1 laufen. Stimmen die Adressen mit der vorherigen Aufgabe 3.1.1 überein?

- Achten Sie auf die Hits und Misses
- Wie viele Hits / Misses gab es?
- Berechnen Sie die Hit- und Miss-Rate

Anzahl Hits	
Anzahl Misses	
Hit-Rate	
Miss-Rate	

Erklären Sie die tiefe Hit-Rate durch Ausfüllen der Tabelle beim Abarbeiten der inneren for-Schleife.

Zugriff auf Element	Memory address	hit/miss	Cache Zeile	Inhalt Cache Zeile nach Zugriff und Cache update
b[0][0]	0x32	miss	Zeile 0	b[0][0], b[0][1], b[0][2], b[0][3]
c[0][0]				
a[0][0]				
b[1][0]				
c[1][0]				
a[1][0]				
b[2][0]				

3.3 Schleifen optimieren

Optimieren Sie nun die Funktion **`run_simulation(void)`**, so dass Sie eine höhere Hit-Rate erzielen.

Was haben Sie geändert?

Welche Werte erhalten Sie nun und wann tritt der erste ‘Hit’ auf?

Erster Hit bei Adresse	
Anzahl Hits	
Anzahl Misses	
Hit-Rate	
Miss-Rate	

Zur Veranschaulichung verwenden Sie nun den Direct-Mapped Cache Simulator im Internet unter https://www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html. Konfigurieren Sie diesen mit den gleichen Parametern für Cache Size, Memory Size und Offset Bits wie unter 3.1.2 angegeben. Geben Sie nun die ersten Memory Load-Zugriffe bis zum ersten ‘Hit’ ein, wie gerade auf dem CT-Board getestet. Verwenden Sie direkt die Hex-Zahlen, die Sie bei Single-Step auf dem CT-Board sehen. In welcher Speicheradresse und Cache Zeile tritt der erste ‘Hit’ auf? Warum nicht früher?

3.4 Cache optimieren

Optimieren Sie die Cache-Parameter im File **`config.h`** für die vorherigen Aufgaben. Ändern Sie schrittweise die Parameter für Offset und Index, lassen Sie jedoch den Wert für ADDRESS_SIZE auf 11 stehen. (Achtung: OFFSET + INDEX < ADDRESS_SIZE)

Mit welchen Werten erhalten Sie die besten Resultate? Versuchen Sie, den Cache möglichst klein zu halten.

Anzahl Bits für den Offset	
Anzahl Bits für den Index	
Anzahl Hits	
Anzahl Misses	

Was stellen Sie fest? Wie gross ist nun die Cache Zeile gegenüber den drei Arrays?

3.5 Grosses Array

Stellen Sie nun wieder die gleichen Parameter wie in Aufgabe 3.1.2 für den Cache ein. Ändern Sie diesmal aber die Array Grösse auf die folgenden Werte:

Anzahl Zeilen	50
Anzahl Spalten	10

Welche Werte stellen Sie fest?

Anzahl Hits	
Anzahl Misses	
Hit-Rate	
Miss-Rate	

Vergleichen Sie die Performance mit Aufgabe 3.2.1. Machen Sie schliesslich Ihre Änderungen in **`run_simulation(void)`** im Abschnitt 3.2.1 wieder rückgängig und schauen Sie sich die Hit-Rate an.

3.6 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Bewertungskriterien	**Gewichtung**
Speicherzugriffe bei Arrays	1/4
Simulation	1/4
Cache optimieren	1/4
Grosses Array	1/4