CT Praktikum: Cache

1 Funktion

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 sollen die Speicherzugriffe auf den Cache untersucht und durch Verändern der Cache-Parameter optimiert werden.

2 Lernziele

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

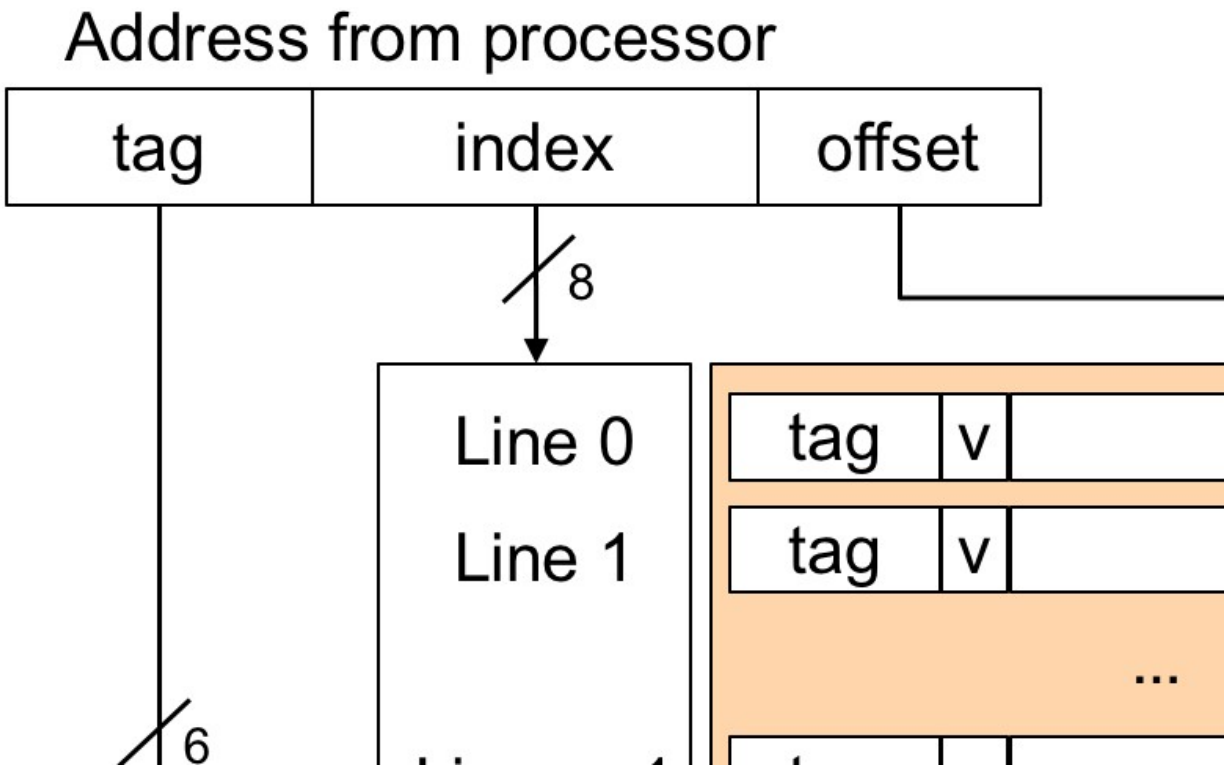

Abbildung 1: Schematische Darstellung eines Direct-Mapped-Cache

3 Aufgaben

Ein Direct-Mapped Cache soll am einfachen Beispiel einer Summation von zwei Arrays untersucht werden. Dazu wird der Cache auf dem CT-Board simuliert.

3.1 Speicherzugriffe bei Arrays

Analysieren Sie folgenden Beispielcode:

```
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 werden folgende Konfigurationen der Arrays verwendet:

Anzahl Zeilen	5 **3 Bits**
Anzahl Spalten	10 **4 Bits**
Grösse eines Elements	1 Byte
Startadresse des ersten Arrays (a)	0x0000'0000

3.1.1 Adressen?

Die Arrays werden wie im Code oben hintereinander deklariert. Welches sind die Adressen der ersten Elemente als HEX-Zahlen?

Element	Adresse des ersten Elements
a[0][0]	0x0000'0000
b[0][0]	**0x0000'007C = 2^(7 LSB for positioning.) + a[0][0]**
c[0][0]	**0x0000'00F8 = 2^(7 LSB for positioning.) + b[0][0]**

3.1.2 Dimensionen RAM / Cache

Für die Simulation werden folgende Parameter für den Cache verwendet:

Anzahl Bits für die Adressierung	11
Anzahl Bits für den Offset	2
Anzahl Bits für den Index	2
Anzahl Bits für den Tag	7

Bestimmen Sie folgende Dimensionen des RAM und Cache:

Grösse des RAM in Bytes (Memory Size)	
Anzahl Zeilen im Caches	2^2 = 4
Grösse einer Cache-Zeile in Bytes (nur Nutzdaten)	2^2 = 4bytes
Grösse des Cache in Bytes (Cache Size)	4 * 4 = 16bytes

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. Arbeiten Sie sich in die entsprechenden Funktionen ein. Analysieren Sie dann im File `cache.c` die Funktion `access_cache(uint32_t address)`.

Lesen Sie zusätzlich das Header-File `config.h`, um einen Überblick über die Konfiguration des Cache und der Arrays zu erhalten. Kontrollieren Sie ob die Konfiguration mit der vorherigen Aufgabe übereinstimmt.

Schauen Sie sich nun die Funktion `run_simulation(void)` im File `main.c` und vergleichen Sie die Simulation mit dem Beispielcode der vorherigen Aufgabe.

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

---

um ein Array der grösse 5 x10 zu erstellen werden mehr offset bits benötigt. für den Index von 5 braucht es 3 bit und für das Offset von 10 braucht es 4 Bits.

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

Im Bsp wird zu begin des Codes platz für die Array reserviert. Imfalle des Codes warden die Arrays erst später iniziiert und auch nicht in der selben reihenfolge wie im Bsp.

Hinweis: Sie können die Simulation mit den Tasten T0 und T1 steuern (T1 -> Single Step / T0 -> 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	0
Anzahl Misses	150
Hit-Rate	0%
Miss-Rate	100%

3.2.1 Schleifen optimieren

Optimieren Sie nun die Funktion `run_simulation(void)`, so dass Sie eine höhere Hitrate erzielen.

Was haben Sie geändert?

OFFSET geändert zu 4 und INDEX geändert zu 3 um die Richtigen Linen zu lessen.

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

Erster Hit bei Adresse	0x0000003C
Anzahl Hits	112
Anzahl Misses	38
Hit-Rate	56/75
Miss-Rate	19/75

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 Zeile tritt der erste ‘Hit’ auf? Warum nicht früher?

3.3 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?

Anzahl Bits für den Offset	4
Anzahl Bits für den Index	3

Was stellen Sie fest? Wie gross ist nun der Cache gegenüber dem gesamten Memory?

3.4 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	0
Anzahl Misses	1500
Hit-Rate	0%
Miss-Rate	100%

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.

3.5 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