Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Übung01: Informationstechnik (IT)

Harald Bucher

Institutsleitung
Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

Institut für Technik der Informationsverarbeitung (ITIV)

Teil 1: Organisatorisches, Einführung & Besprechung Aufgabenblatt 1

KIT – Universität des Landes Baden-Württemberg und
nationales Forschungszentrum in der Helmholtz-Gemeinschaft

www.kit.edu

Inhalt: Übung01 Teil 1

1. Organisatorisches
2. Einführung
3. Aufgabenblatt 1

IT-Veranstaltungsplan

SW	Datum Mi.	9:45-11:15	Datum Do.	14:00-15:30	Tutorium
1.	15. Apr	VL	16. Apr		
2.	22. Apr	VL	23. Apr		
3.	29. Apr	VL	30. Apr		
4.	06. Mai	VL	07. Mai	ÜB	
5.	13. Mai	VL	14. Mai	Feiertag	X
6.	20. Mai	VL	21. Mai	ÜB	X
7.	27. Mai	Pfingsten	28. Mai	Pfingsten	X
8.	03. Jun	VL	04. Jun	Feiertag	X
9.	10. Jun	VL	11. Jun	ÜB	X
10.	17. Jun	VL	18. Jun	ÜB	X
11.	24. Jun	VL	25. Jun	ÜB	X
12.	01. Jul	VL	02. Jul	ÜB	X
13.	08. Jul	VL	09. Jul	ÜB	X
14.	15. Jul	VL	16. Jul	Probeklausur	X

SW = Semesterwoche; VL = Vorlesung; ÜB = Übung

Termine - Tutorium am Rechenzentrum

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
08:00 09:30		Pool K			
09:45 11:15				Pool I Pool K	
11:30 13:00		Pool I			
14:00 15:30			Pool I		
15:45 17:15			Pool I Pool K		
17:30 19:00	Pool I Pool K		Pool I Pool K	Pool I	

SCC-Pool: Start der virtuellen Maschine

- Im Bootloader den Eintrag ***„ITIV Praktikum“*** auswählen
- Start der virtuellen Maschine durch Eingabe von
  - Benutzername: itiv
  - Passwort: itSS15
- Login nach Aufforderung auf der Konsole über **u-Account** zum Einbinden des Home-Laufwerks.

- !!! Datenspeicherung **nur** auf **Home-Netzlaufwerk** oder **USB-Stick** !!! Sämtliche Daten auf der lokalen Platte gehen nach Reboot der Maschine verloren!!!

Informationstechnik Übung

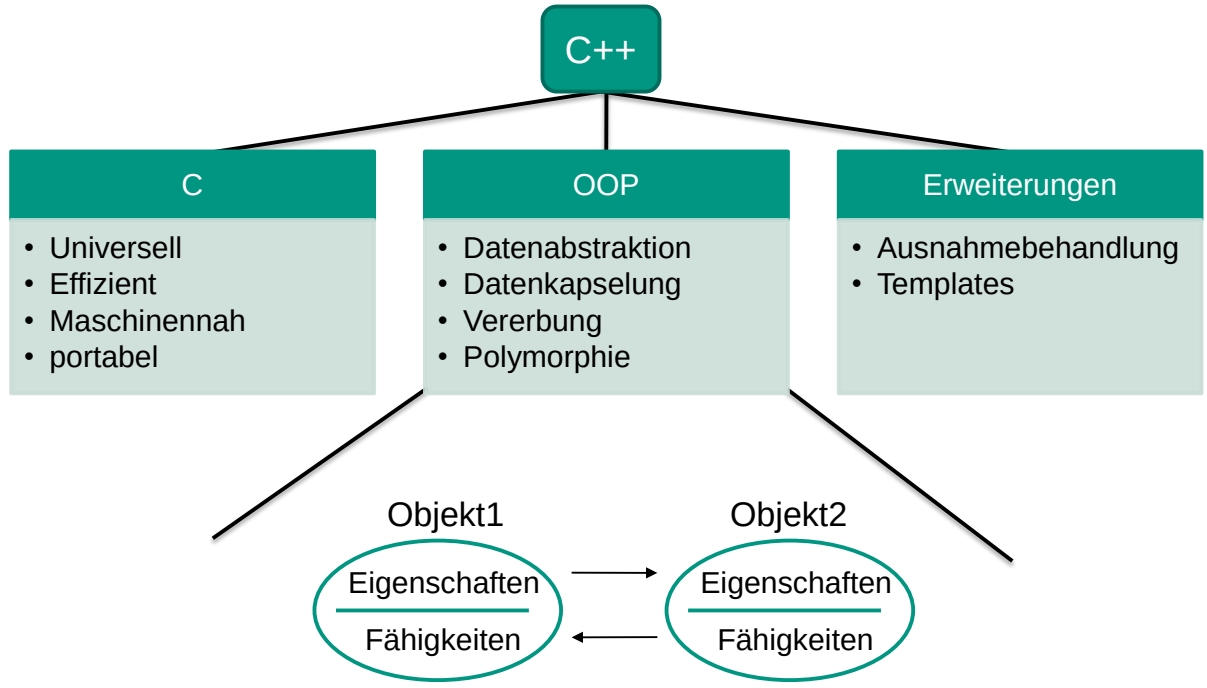
Exemplarische Einführung der höheren Programmiersprache C++

- C und C++ sind die derzeit am weitest verbreiteten Programmiersprachen bezogen auf die Zahl der Anwendungen
- C ist hardwarenah (Treiberprogrammierung)
- C++ umfangreiche Erweiterung von C
- C++ exemplarisch für objektorientierte Programmierung (kommt erst später in der Vorlesung)
- C++ ist eine der mächtigsten Programmiersprachen
- C++ auch Basis für die Hardwarebeschreibungssprache SystemC

Lernziele der Übung

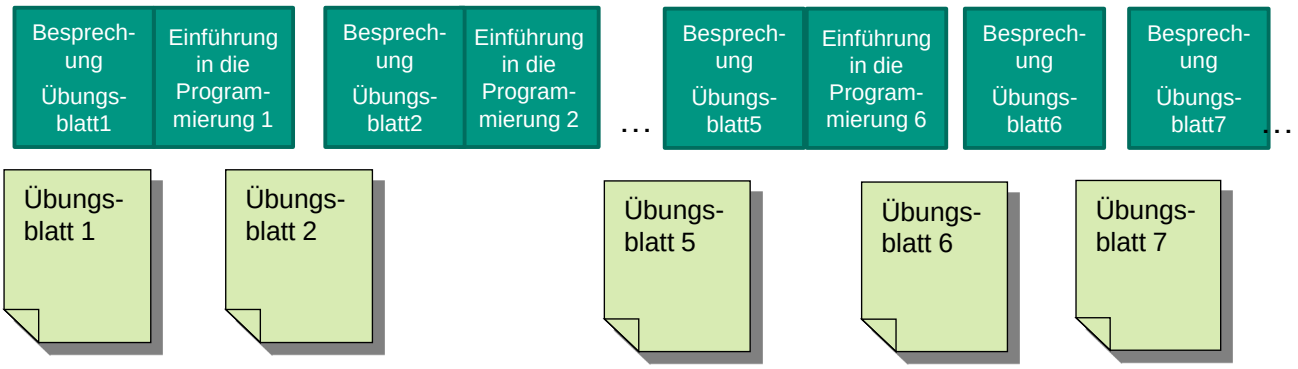
- Der Lernende soll am Ende:
  - Die Grundzügen der Programmiersprache C++ anwenden können
  - Konkrete Problemstellungen mit Hilfe der Informationstechnik lösen können
  - Seine Programme nach den Prinzipien der Objektorientierung strukturiert aufbauen können
  - Algorithmen in unterschiedlichen Darstellungsformen beschreiben können und in lauffähige Programme umsetzen können
  - Qualitätsmerkmale von Algorithmen und Programmen kennen und anwenden und hierzu Tests und Testprogramme erstellen
  - Mit den Grundfunktionen einer Entwicklungsumgebung umgehen können

Warum C++?



Teilung der Übung

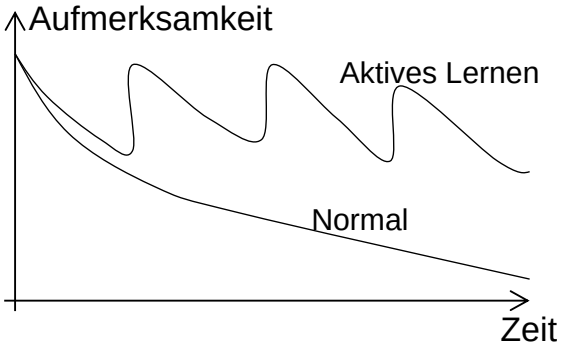
- Übung besteht im Allgemeinen aus 2 Teilen
  - 1. Teil: Besprechung der Übungsaufgaben
  - 2. Teil: Einführung in die Programmierung



Aktives Lernen

- Aktive / Direkte Mitarbeit der Studenten
  - Kurze Aufgaben während der Übung zum direkten Lösen
  - Max. 2 Minuten
  - In Gruppenarbeit
  - Direkte Lösungsanfrage an Gruppen
  - Themenwechsel

- Ziele
  - Schnelleres Lernen
  - Weniger Nacharbeit
  - Direkteres Verstehen
  - Erhöhte Aufmerksamkeit



Aufg. 1.01: Verständnisfragen Lsg. (1)

a) Der Cache kompensiert die Latenz zwischen Prozessor und Hauptspeicher.

b) Definieren Sie den Begriff "Write through" in Bezug auf Caches und nennen Sie einen Vor- und einen Nachteil.

Jeder Schreibzugriff wird auch im Arbeitsspeicher durchgeführt. Jeder Schreibzugriff bedingt einen Zugriff auf den Systembus.

Nachteile:

- durch Belastung des Systembusses, wird Zugriff anderer Komponenten auf den Arbeitsspeicher erheblich behindert.
- Längere Ausführungszeit

Vorteile:

- Datenkonsistenz ist gesichert, d.h. es ist gewährleistet, dass Datum im Cache und Arbeitsspeicher stets den gleichen Wert hat
- Erheblich einfachere Steuerung

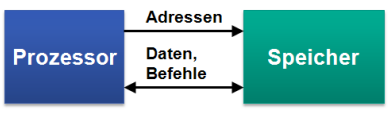
Aufg. 1.01: Verständnisfragen Lsg. (2)

c) Techniken zur Beschleunigung der Befehlsausführung sind Pipelining, Superpipelining, Superskalar, Multi-Core.

d) Bei der Interpretation eines Befehls im Speicher wird unterschieden, ob das niedrigstwerte Byte an der höchsten Adresse („big endian“, z.B. Apple), oder niedrigsten Adresse („little endian“, z.B. Intel x86) gespeichert wird.

e) Was ist der Hauptunterschied zwischen der von-Neumann- und der Harvard-Architektur? Hinweis: Sie können Ihre Lösung auch graphisch darstellen.

- Von Neumann-Architektur:



ein Speicher sowohl für Programm-Daten als auch Daten-Daten

- Harvard-Architektur:

getrennte Speicher für Programm-Daten und Daten-Daten / gleichzeitiger Zugriff möglich

Aufg. 1.01: Verständnisfragen Lsg. (3)

f) Eine Rechnerarchitektur kann grundlegend in interne und externe Architektur unterteilt werden.

g) Eine von-Neumann-Rechnerarchitektur besteht aus den Hauptkomponenten Rechenwerk, Steuerwerk, Speicherwerk und Ein-/Ausgabewerk.

h) Erläutern Sie folgende Adressierungsmodi einer CPU:

Register Mode: Der Operand wird aus einem Register bezogen.

Direct Mode: Die Speicheradresse des Operanden ist in der Instruktion gespeichert.

Immediate Mode: Der Operand ist Teil der Instruktion.

Register Indirect Mode: Ein Register enthält die Speicheradresse, an der der Operand gespeichert ist (Pointer).

i) Das Taktsignal einer CPU gibt maßgeblich die Arbeitsgeschwindigkeit vor und dient zur Synchronisation.

Aufg. 1.01: Verständnisfragen Lsg. (3)

j) Erläutern Sie die beiden Begriffe Pipelining und Superskalar:

- Pipelining: Zeitliche Überlappung der Befehlsausführungsphasen zur Reduktion der Latenzzeiten resultiert in einem höheren Durchsatz / höheren Taktrate.
- Superskalar: Wie Pipelining und zusätzlich meherere Ausführungseinheiten und deren dynamische Befehlszuweisung zur parallelen Ausführung mehrerer Befehle – im Idealfall Vervielfachung der Leistung.

k) Nennen und erläutern Sie die 5 Phasen einer typischen 5-stufigen Pipeline in der korrekten Reihenfolge.

1) Instruction Fetch (IF): Befehl holen; Program counter (PC) erhöhen.
2) Instruction Decode (ID): Instruktion dekodieren und Werte aus Registern lesen.
3) Execute (EX): Ausführung der Operation/Adressberechnung in der ALU.
4) Memory Access (MEM): Hauptspeicherzugriff; nur für Load-, Store- und Branch-Operationen.
5) Write Back (WB): Ergebnis in Registersatz zurückschreiben.

Aufg. 1.01: Verständnisfragen Lsg. (3)

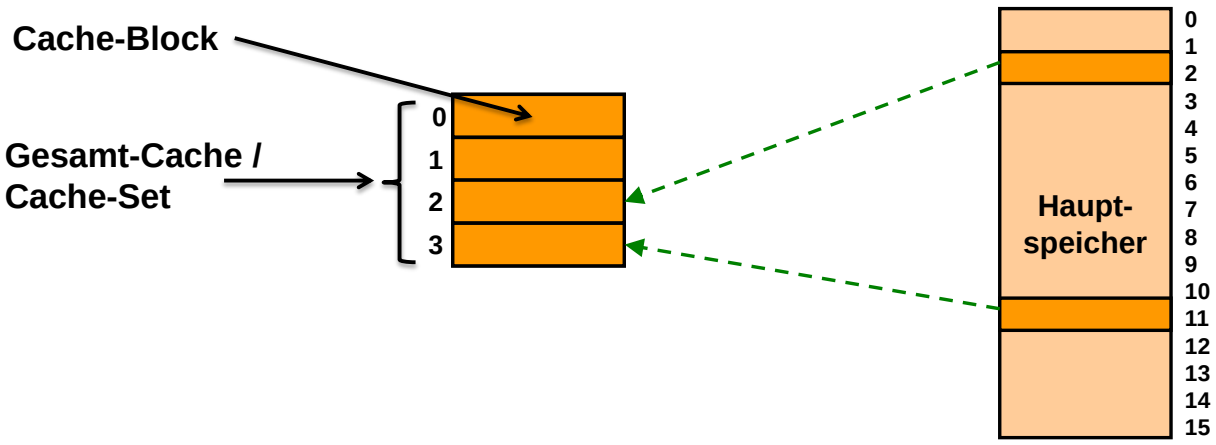
l) Erklären Sie den Unterschied zwischen einem Interrupt und Polling.

- Interrupt: Kurzfristige Unterbrechung des Programms durch Interrupt Request (IRQ), in Hardware gesteuert durch Interrupt Controller. Abarbeiten von zeitkritischen Aufgaben in Interrupt Service Routine (ISR). Laufzeiteffizient, aber zusätzliche Hardware nötig

- Polling: programmierte zyklische Abfrage eines Status
  Einfach, keine zusätzliche Hardware nötig, belegt aber ständig CPU-Leistung

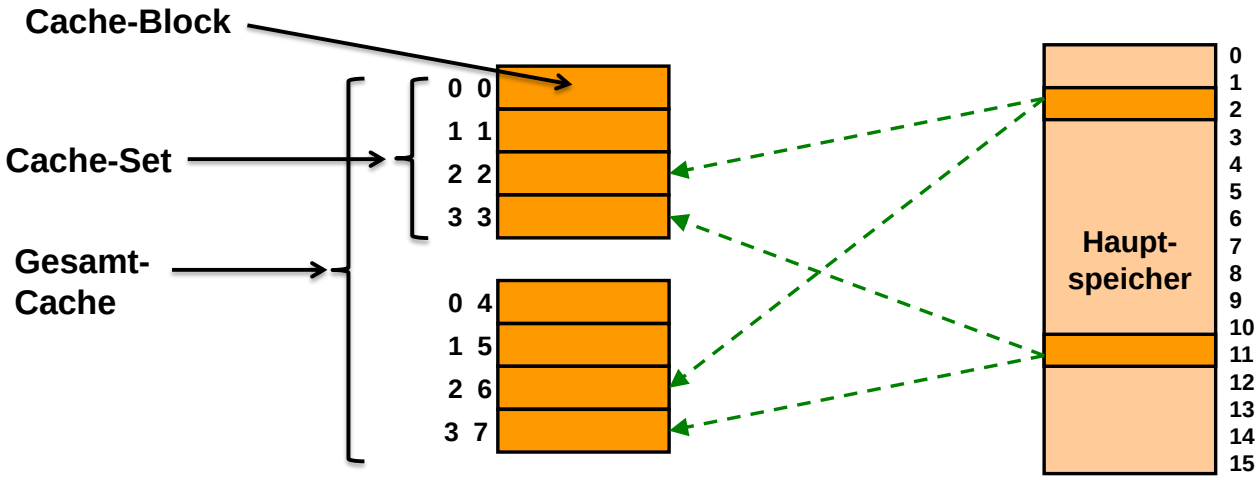
Aufg. 1.02 a): Cacheorganisation Lsg. (1) Direct Mapped

- Ein Block kann nur an einer Adresse mod m (m Cache-Größe, hier: 4) gespeichert werden



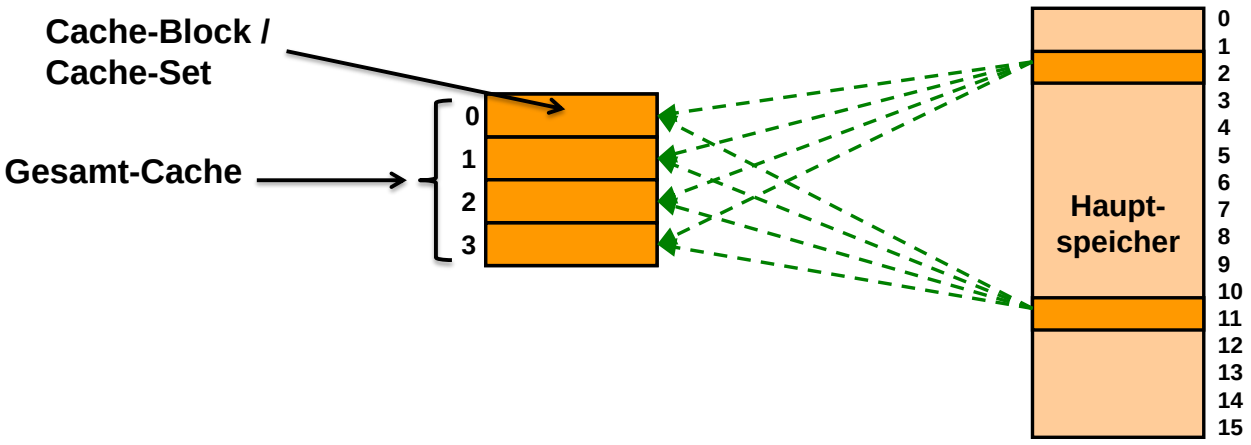
Aufg. 1.02 a): Cacheorganisation Lsg. (2) N-Wege Assoziativ

- Ein Block kann in einem von *N* Cache-Blöcken gespeichert werden, bei denen die Hauptspeicheradresse *mod n* gleich der Cache-Adresse *mod n* ist (*n = Anzahl der Cache-Blöcke in einem Set)*
- Physikalischer Aufbau von N-unabhängigen Caches
- Beispiel: 2-Wege assoziativ



Aufg. 1.02 a): Cacheorganisation Lsg. (3) voll-assoziativ

- Jeder Block kann an allen Adressen gespeichert werden
- Voll-assoziativ ist N-Weg assoziativ mit N = m = Anzahl der Cache-Blöcke



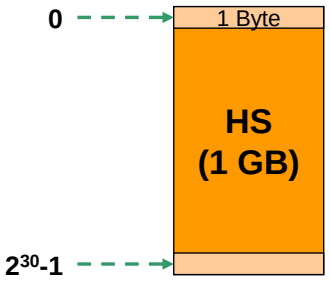
Aufg. 1.02 a): Cacheorganisation Lsg. (4) Vor- und Nachteile

- Voll-assoziativ ist N-Weg assoziativ mit N = m = Anzahl der Cache-Blöcke

N-Weg assoziativ hat den Nachteil, dass *N-1* mehr Komparatoren als direkt-abbildende Caches benötigt werden, was die Hardware komplexer macht und die benötigte Chipfläche erhöht. Vorteil ist aber die höhere Flexibilität beim Zuweisen von Daten aus dem Hauptspeicher in den Cache, da die Daten N verschiedenen Cache-Sets zugewiesen werden können. Dies führt zu einer höheren Cache-Hit-Rate als bei direkt-abbildenden Caches, da bei letzteren jede Adresse im Hauptspeicher genau einem Block im Cache zugewiesen ist.

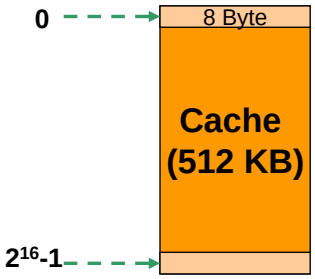
Aufg. 1.02 b): Anzahl Cache-Blöcke und –Sets Lsg.

- Direkt-Abbildend (direct mapped: DM)
  - Anzahl Cache-Blöcke = 512 KB / 8 B = 65536 = 2^{16}
  - Anzahl Cache-Sets = n = 1



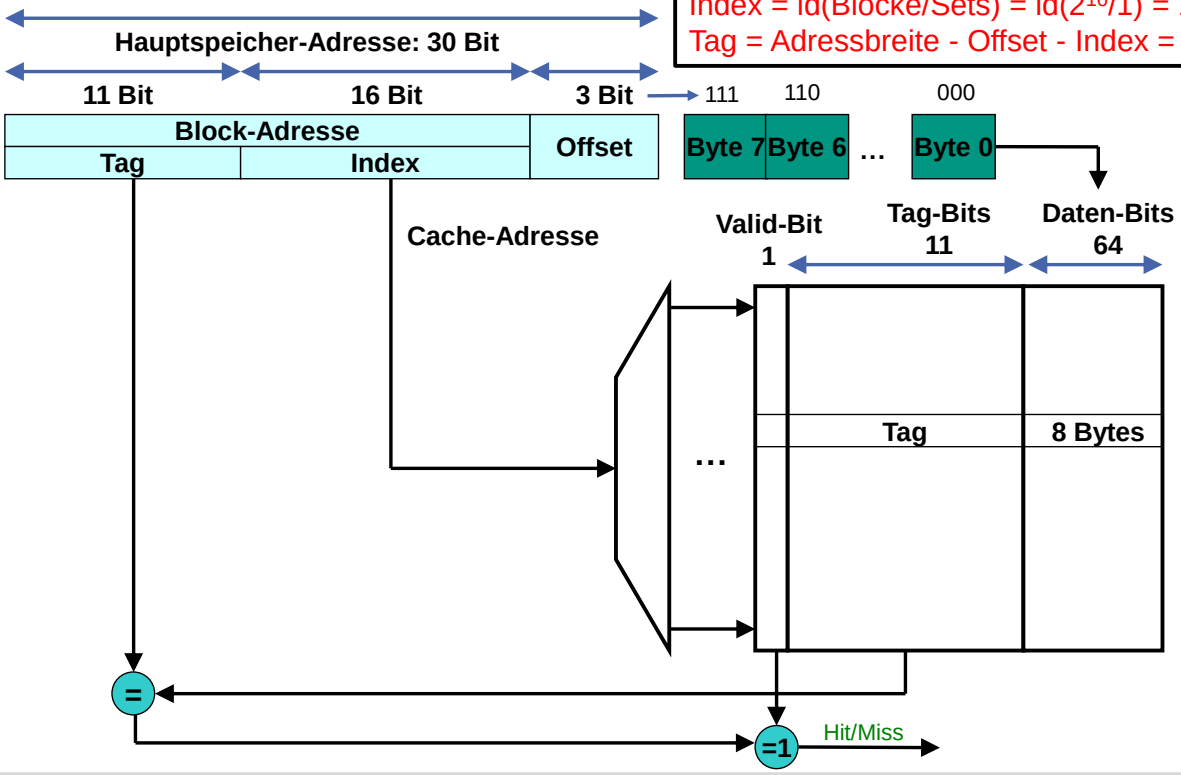
- 4- Weg assoziativ (set associative: SA)
  - Anzahl Cache-Blöcke = 512 KB / 8 B = 65536 = 2^{16}
  - Anzahl Cache-Sets = n = 4

- Voll-assoziativ (fully associative: FA)
  - Anzahl Cache-Blöcke = 512 KB / 8 B = 65536 = 2^{16}
  - Anzahl Cache-Sets = n = 65536 = 2^{16}



Aufg. 1.02 c): Speicherorganisation für den Cachetyp
Lsg. (1) Direct Mapped

Offset = ld(Blockgröße) = ld(8) = 3
Index = ld(Blöcke/Sets) = ld(2^{16}/1) = 16
Tag = Adressbreite - Offset - Index = 11



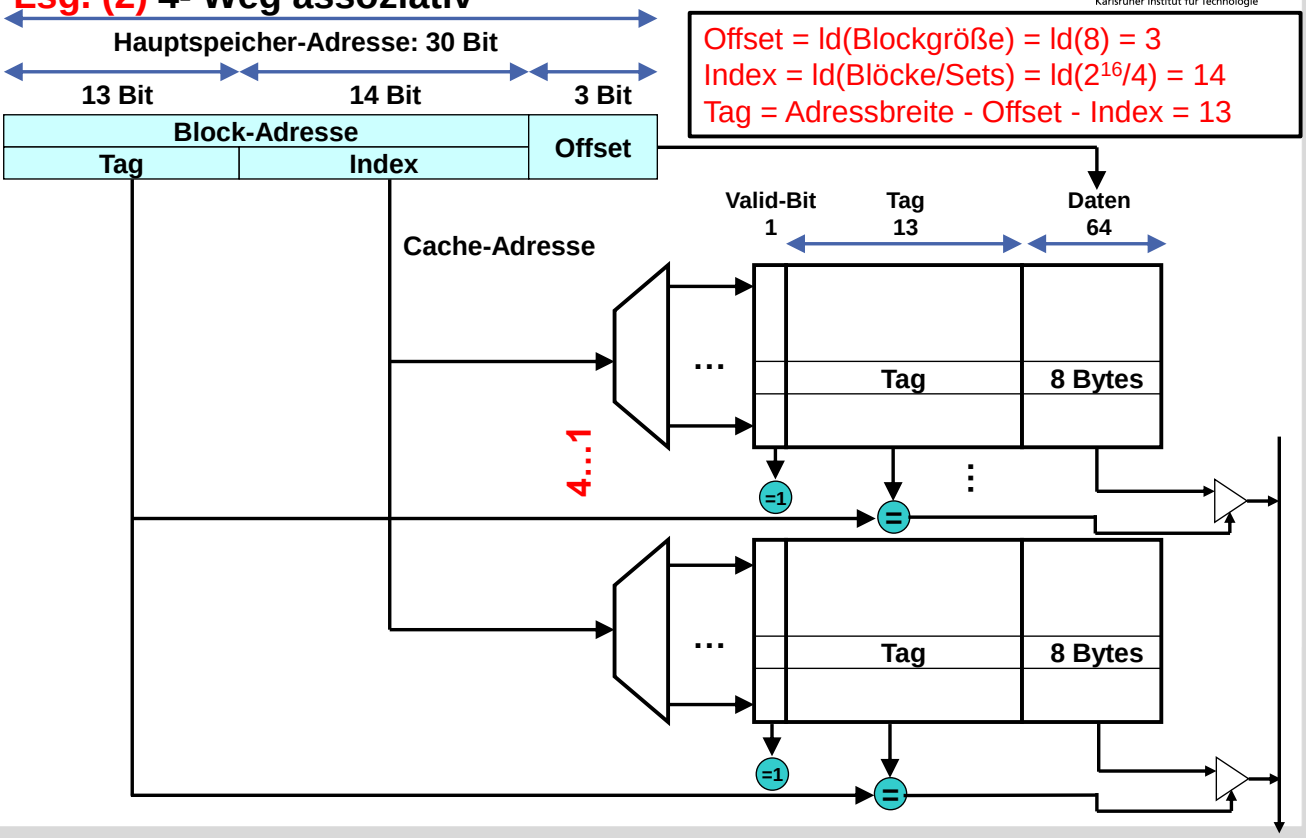
Cacheorganisation

- Beispiel: Direct Mapped
  - Cache ist in Form einer Tabelle mit mehreren Datenworten pro Zeile (Block) organisiert
  - Hauptspeicheradresse von insgesamt 30 Bit Breite setzt sich zusammen aus:

    Offset (3 Bits): adressiert ein 8-Bit Datenwort innerhalb eines Blockes, wobei jeder Block z.B. 2^3 = 8 Datenworte enthalten kann.
    Index(16 Bit): adressiert einen Block innerhalb des Caches, die Gesamtzahl der Blöcke beträgt 2^{16} = 65536.
    Tag (11 Bit): dient als Referenz im Cache, um die Gültigkeit des aktuellen Eintrages festzustellen, ist dies nicht der Fall, werden die alten Werte im Cache bei Bedarf in den Hauptspeicher übertragen und die aktuell referenzierten Daten aus dem Hauptspeicher in den Cache geschrieben.
  - Das Valid-Bit gibt an, ob in dem betreffenden Block die Daten gültig sind, d.h. ob der entsprechende Block in Benutzung ist, oder ob der Block frei ist und einfach überschrieben werden kann
  - Die gesamte Größe des Caches errechnet sich aus der Anzahl der Zeilen, Anzahl der Datenworte pro Zeile und der Größe der Datenworte: 65536 * (8 * 1) = 512 KBytes, hinzu kommen die Verwaltungsbits

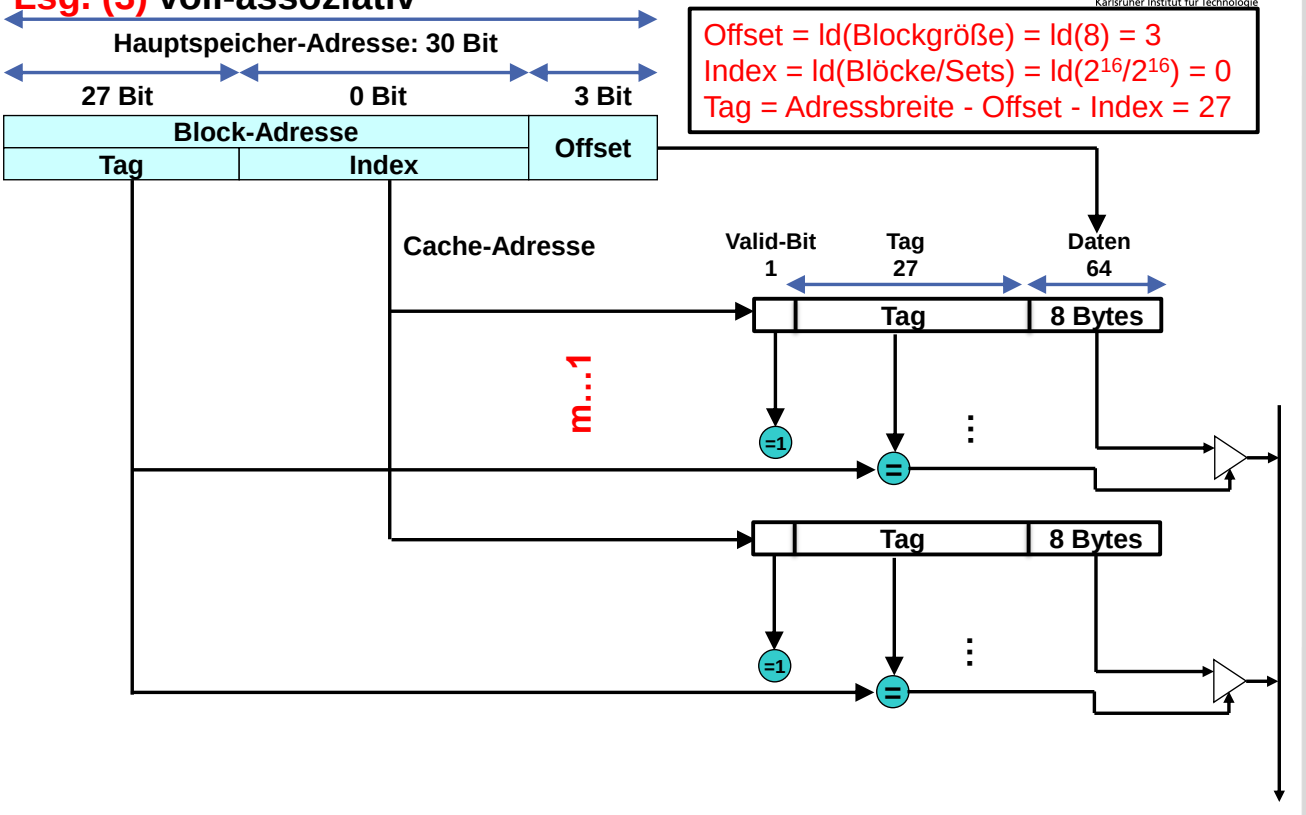
Aufg. 1.02 c): Speicherorganisation für den Cachetyp

Lsg. (2) 4- Weg assoziativ



Aufg. 1.02 : Speicherorganisation für den Cachetyp

Lsg. (3) voll-assoziativ



Aufg. 1.02 d): notwendige Speicherkapazität für den Cache Lsg.

- Speichergröße = Datenspeicher + Speicher für Tags + Speicher für Valid-Bits
- Direkt-Abbildend (direct mapped: DM)
  - Speichergröße = 512 KB + (12 Bit * 2^{16}) / (8 Bit/Byte) = 512 KB + 96 KB = 608 KB
- 4- Weg assoziativ (set associative: SA)
  - Speichergröße = 512 KB + (14 Bit * 2^{16}) / (8 Bit/Byte) = 512 KB + 112 KB = 624 KB
- Voll-assoziativ (fully associative: FA)
  - Speichergröße = 512 KB + (28 Bit * 2^{16}) / (8 Bit/Byte) = 512 KB + 224 KB = 736 KB
- Der benötigte Speicher ist offensichtlich nicht der ausschlaggebende Faktor bei der Auswahl! Was sonst?
- Trade-off: Anzahl der benötigten Komparatoren (Logik-Platz)

  vs.

  Höhere Flexibilität beim Mapping (== höhere Cache-Hit-Rate)

Referenz

- Vorlesung: Kapitel 1 – Rechnerarchitekturen

Vielen Dank für Ihre Aufmerksamkeit

Harald Bucher
Karlsruher Institut für Technology (KIT) – ITIV
harald.bucher@kit.edu