Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Übung 01: Informationstechnik (IT)

Marc Weber

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

Teil 1: Organisatorisches, Einführung & Besprechung Aufgabenblatt 1

Inhalt: Übung 01 – Teil 1

1. Organisatorisches
2. Einführung
3. Besprechung Aufgabenblatt 1
4. Exkurs: µC-Programmierung

IT-Veranstaltungsübersicht SS2016

- Vorlesung
  - 2 SWS
- Übung
  - 1 SWS
- Ort und Zeit
  - Mittwoch: 09:45 bis 11:15 Uhr, Gebäude 10.21, Benz Hörsaal
  - Donnerstag: 14:00 bis 15:30 Uhr, Gebäude 30.46, Neue Chemie
- Praktikum
  - 2 SWS
  - Genauere Informationen während der Einführungsveranstaltung
    - Montag, 23.05.2016, 15:45 – 17:15 Uhr, Gebäude 50.35 Am Fasanengarten
- Tutorium
  - Freiwillig, aber anmeldungspflichtig
  - Räume des ITIV und des SCC

IT-Veranstaltungsplan SS2016

KW	SW	Datum Mittwoch	9:45 – 11:15 Uhr (Benz)	Datum Donnerstag	14:00 – 15:30 Uhr (Neue Chemie)	Tutorium	Praktikum
16.	1.	20. Apr	VL0	21. Apr	VL1		
17.	2.	27. Apr	VL2	28. Apr	ÜB1a		
18.	3.	04. Mai	VL3	05. Mai	Feiertag	T1	
19.	4.	11. Mai	ÜB1b	12. Mai	ÜB2	T2	
20.	5.	18. Mai	Pfingsten	19. Mai	Pfingsten		
21.	6.	25. Mai	VL4	26. Mai	Feiertag	T3	P0 (EV)
22.	7.	01. Jun	VL5	02. Jun	*frei*	T4-EI1	
23.	8.	08. Jun	VL6	09. Jun	ÜB3	T5-EI2	P1 (PM)
24.	9.	15. Jun	VL7	16. Jun	ÜB4		P2
25.	10.	22. Jun	VL8	23. Jun	*frei*		P3
26.	11.	29. Jun	VL9	30. Jun	ÜB5		P4
27.	12.	06. Jul	VL10	07. Jul	ÜB6		P5
28.	13.	13. Jul	VL11	14. Jul	ÜB7		P6
29.	14.	20. Jul	VL12 (Puffer)	21. Jul	ÜB8 (Probeklausur)		P7 (PF)

KW= Kalenderwoche; SW = Semesterwoche; VL = Vorlesung; ÜB = Übung; T= Tutorium; P= Praktikum; EI = EI-Team Kurs 4.

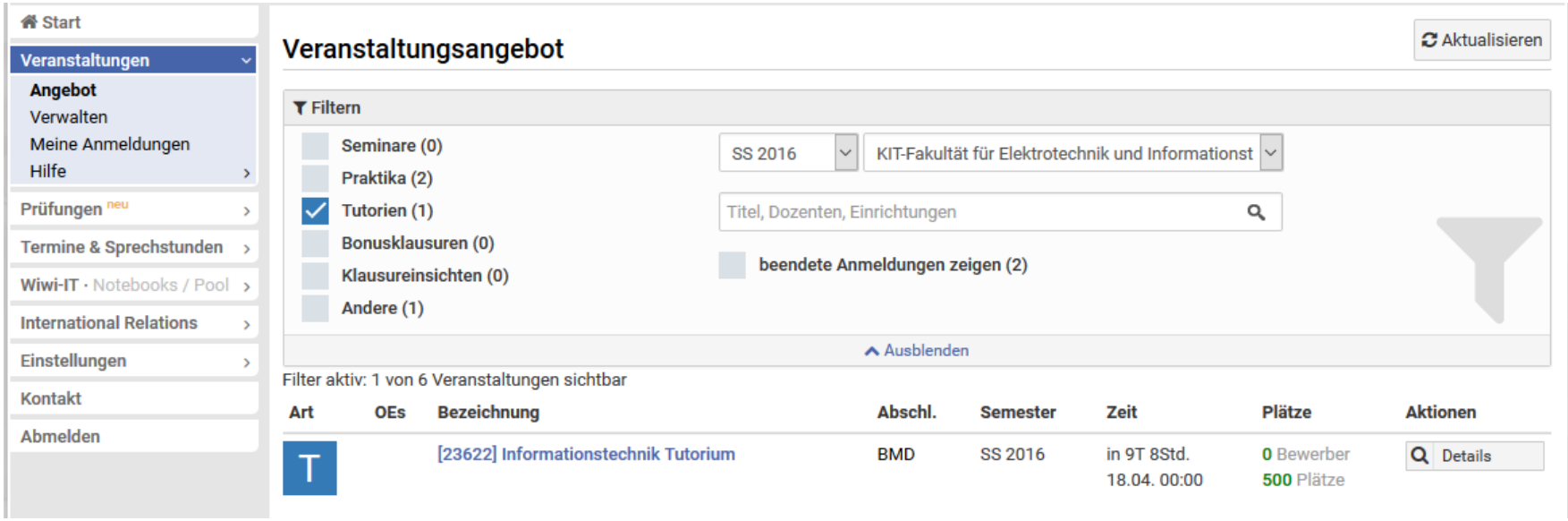
Unterlagen zu Informationstechnik

- Nutzung der Lernplattform ILIAS
  - https://ilias.studium.kit.edu
  - Registrierung notwendig (falls noch nicht erfolgt)
  - Kurs: [23622] Informationstechnik (SS 2016)
    - Passwort: it@SS16 (Groß-/Kleinschreibung beachten!)

- Inhalt
  - Vorlesungsfolien, Übungsblätter, Übungsfolien, Tutoriumsaufgaben, …

Tutorium

- Einwahl zu den Terminen des Tutoriums über Wiwi Portal
  - Registrierung notwendig unter https://portal.wiwi.kit.edu/
  - Direktlink zum Kurs: https://portal.wiwi.kit.edu/ys/934
  - Bitte jeden Termin „bewerten“ (1-5 Sterne) → Zuteilung der Studenten
  - Anmeldeende: 28.04.2016, 16:45 Uhr!



Termine – Tutorium am ITIV/Rechenzentrum SS2016

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
08:00 09:30		Pool C			Pool C
09:45 11:15	ITIV-115	Pool C			Pool C
11:30 13:00		ITIV-115			
14:00 15:30	Pool L		Pool I, ITIV-115		
15:45 17:15	Pool L		Pool I		Pool A, B, K
17:30 19:00					Pool A, B, K

Termine – Praktikum am Rechenzentrum SS2016

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

Start der virtuellen Maschine in SCC-Pools

- Im Bootloader den Eintrag „**BW-Lehrpool**“ auswählen
- Login nach Aufforderung auf der Konsole über **u-Account**
- Im „**bwLehrpool VMChooser**“ die virtuelle Maschine „**ITIV-Praktikum**“ per Doppelklick auswählen
- Windows startet mit Administratorrechten
- Datenspeicherung nur auf Home-Netzlaufwerk oder USB-Stick!
  - Sämtliche Daten auf der lokalen Platte gehen durch Reboot der virtuellen Maschine verloren!

- Für die Verwendung der virtuellen Maschine auf eigenen PCs
  - Benutzername: ITIV_IT
  - Passwort: it@SS16

Anmeldung zur Klausur

- Bachelor: Verwendung der Selbstbedienungsfunktion des Studienbüros https://campus.studium.kit.edu/
- Andere oder nicht online freigeschaltete Studiengänge: Direkte Anmeldung durch Abgabe des blauen Zettels oder Entsprechendem bei einem Betreuer (spätestens eine Woche vor der Prüfung)

- Neu im SS2016: Ein erfolgreiches Bestehen des Praktikums ist Zulassungsvoraussetzung für die Prüfung!
  - Projektplan, lauffähige Software (Simulation), Dokumentation, …

- Anmeldebeginn: 01.08.2016
- Anmeldeende: 30.08.2016
- Abmeldeende: 04.09.2016

Klausur (schriftlich)

- Inhalt: Vorlesung, Übung, Tutorium und Praktikum
- Termin: Dienstag, 06.09.2016, 11:00 bis 13:00 Uhr
- Ort: Benz, Daimler, Fritz-Haller HS, Gaede, Gerthsen, HS a. F., Neue Chemie, Tulla HS
- Dauer: 2 Stunden
- Mitzubringen: Fricard oder Lichtbildausweis und Schreibzeug
  - Nicht Bachelorstudenten zusätzlich: Studienzeitbescheinigung
- Hilfsmittel: 1x A4 Blatt (2x A4 Seiten, handschriftlich, lesbar ohne Hilfsmittel)

- Hörsaalverteilung (nach Nachnamen, bei Doppelnamen ist der erste Nachname maßgebend)
  - Details werden vor der Prüfung bekannt gegeben

- Organisatorisches

Informationstechnik Übung

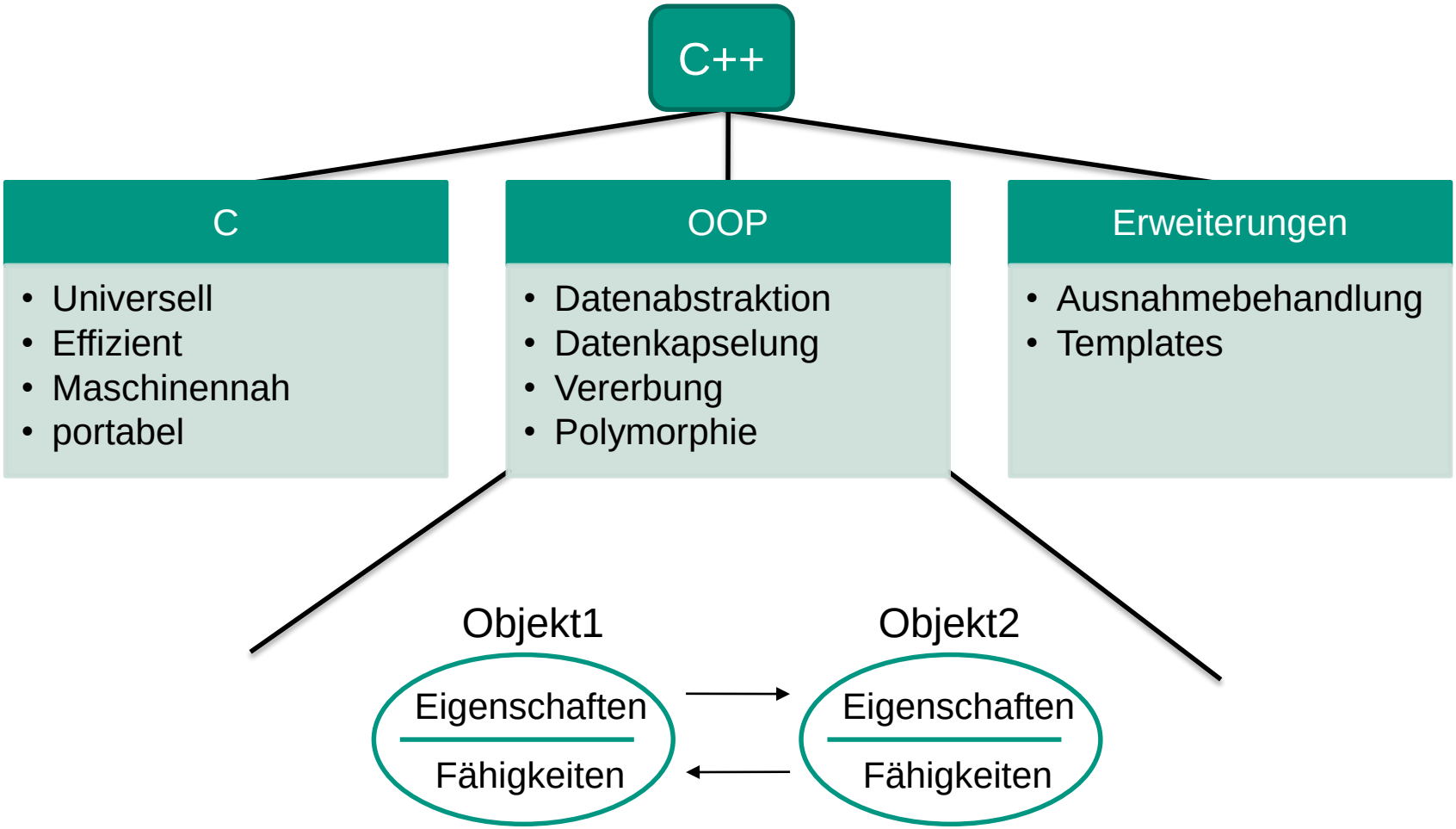
Exemplarische Einführung der höheren Programmiersprache C++

- 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

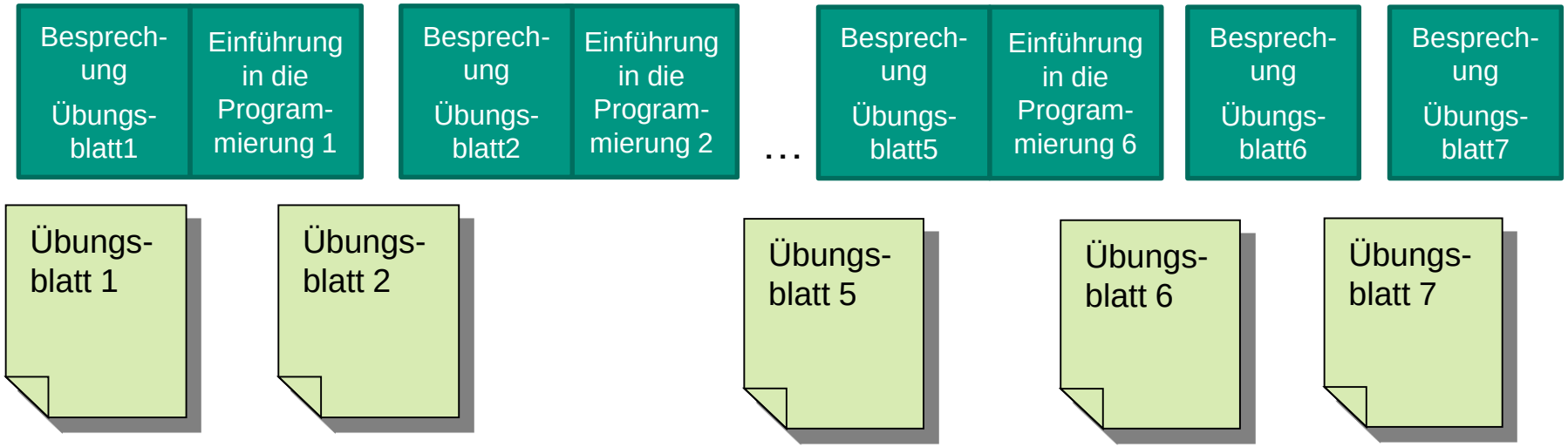
- Der Lernende soll am Ende:
  - Die Grundzügen der Programmiersprache C++ anwenden können
  - Konkrete Problemstellungen mit Hilfe der Informationstechnik lösen können
  - Seine Programme nach den Prinzipien der Objektorientierung strukturiert aufbauen können
  - Algorithmen in unterschiedlichen Darstellungsformen 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

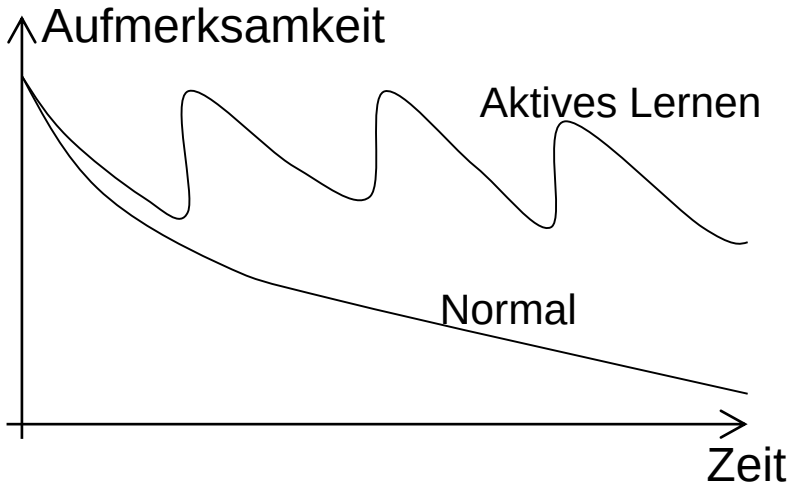
- Übung besteht im Allgemeinen aus 2 Teilen
  - 1. Teil: Besprechung der Übungsaufgaben
  - 2. Teil: Einführung in die Programmierung



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



Inhaltsverzeichnis Übung

1. Variablen & Gültigkeitsbereich, Operatoren, Arrays, Kontrollstrukturen
2. Funktionen, Zeiger & Header
3. Objektorientierung & Dynamische Speicherverwaltung
4. Vererbung & Polymorphie, Strings
5. Dateiverarbeitung & STL
6. Sortieralgorithmen
7. Q&A
8. Probeklausur

- Einführung

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:

1. Durch Belastung des Systembusses, wird der Zugriff anderer Komponenten auf den Arbeitsspeicher erheblich behindert
2. Längere Ausführungszeit

Vorteile:

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

Aufg. 1.01: Verständnisfragen Lsg. (2)

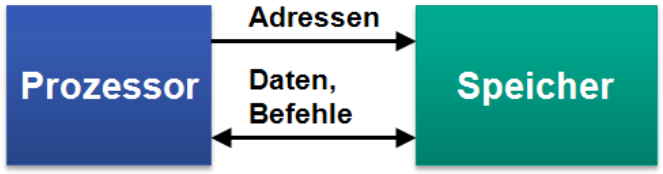
c) Techniken zur Beschleunigung der Befehlsausführung sind Pipelining, Superpipelining, Superskalar, Multi-Core.

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.

Aufg. 1.01: Verständnisfragen Lsg. (3)

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. (4)

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:

a. Register Mode: Der Operand wird aus einem Register bezogen.
b. Direct Mode: Die Speicheradresse des Operanden ist in der Instruktion gespeichert.
c. Immediate Mode: Der Operand ist Teil der Instruktion.
d. Register Indirect Mode: Ein Register enthält die Speicheradresse, an der der Operand gespeichert ist (Pointer).

Aufg. 1.01: Verständnisfragen Lsg. (5)

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

j) Erläutern Sie die beiden Begriffe Pipelining und Superskalar:

a) Pipelining: Zeitliche Überlappung der Befehlsausführungsphasen zur Reduktion der Latenzzeiten resultiert in einem höheren Durchsatz.

b) Superskalar: Wie Pipelining und zusätzlich meherere Ausführungseinheiten und deren dynamische Befehlszuweisung zur parallelen Ausführung mehrerer Befehle – im Idealfall Vervielfachung der Leistung.

Aufg. 1.01: Verständnisfragen Lsg. (6)

k) Nennen und erläutern Sie die 5 Phasen einer typischen 5-stufigen DLX 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. (7)

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

a) 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.

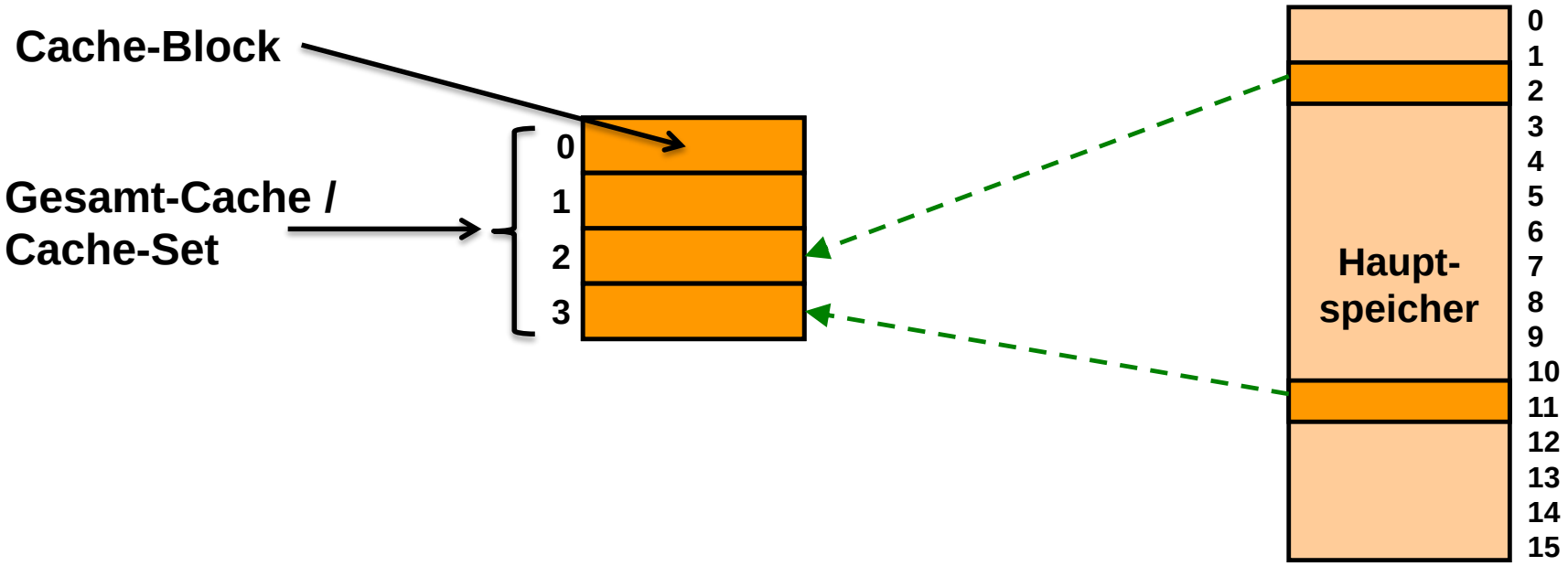
b) Polling: Programmierte zyklische Abfrage eines Status.
→ Einfach, keine zusätzliche Hardware nötig, belegt aber ständig CPU-Leistung.

- Aufg. 1.01: Verständnisfragen

Aufg. 1.02: Cacheorganisation Lsg. (1)

a) Nennen Sie die drei Möglichkeiten der Cacheorganisation. Welche Vor- und Nachteile haben diese?

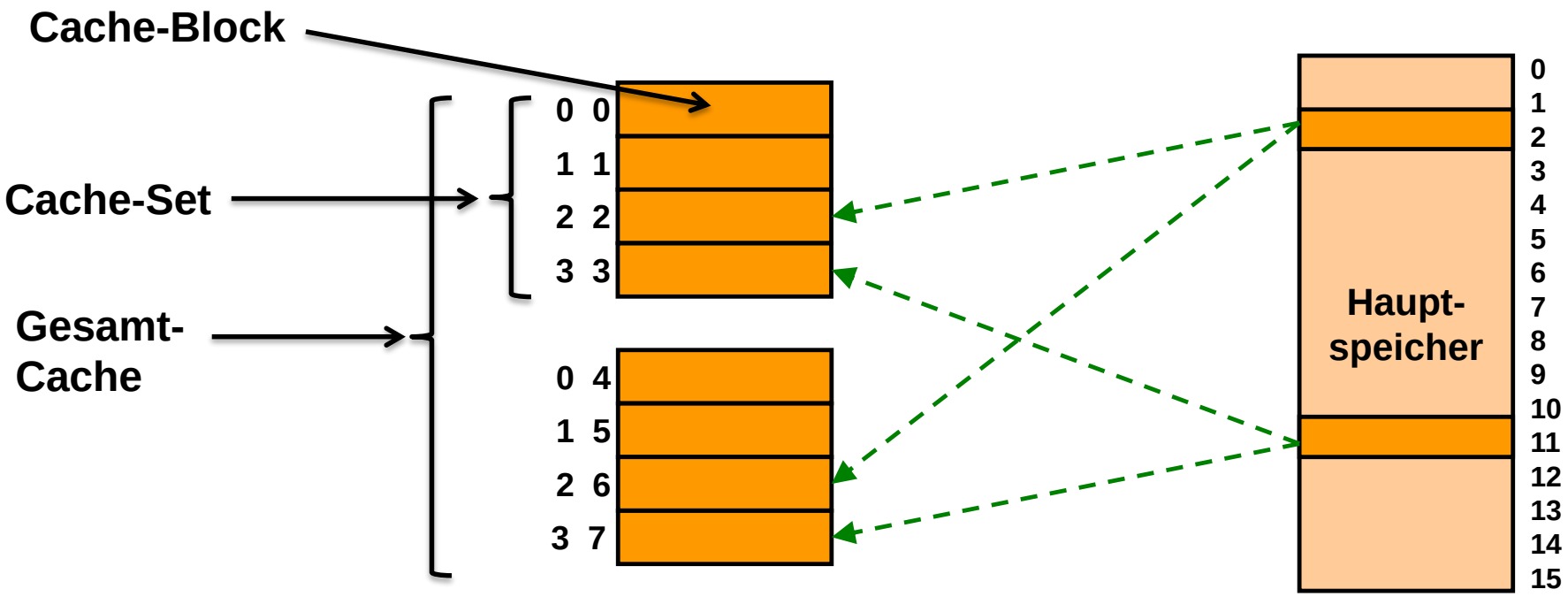
Direct Mapped: Ein Block kann nur an einer Adresse mod m (m = Cache-Größe, hier: 4) gespeichert werden.



Aufg. 1.02: Cacheorganisation Lsg. (2)

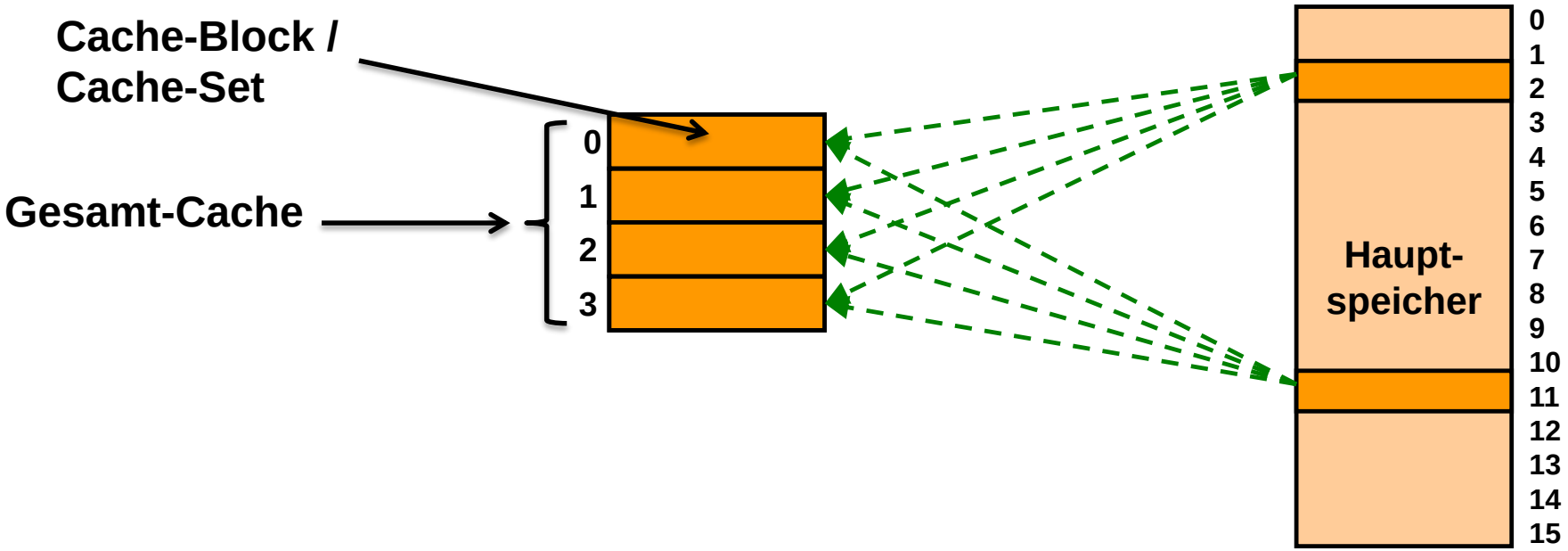
N-Wege assoziativ: Ein Block kann in einem von *N* Cache-Blöcken gespeichert werden, bei denen die Hauptspeicheradresse mod n gleich der Cache-Adresse mod n ist (n = Anzahl der Cache-Blöcke in einem Set). Physikalischer Aufbau von *N*-unabhängigen Caches.

Beispiel: 2-Wege assoziativ



Aufg. 1.02: Cacheorganisation Lsg. (3)

Voll-assoziativ: Jeder Block kann an allen Adressen gespeichert werden. Voll-assoziativ ist *N*-Wege assoziativ mit $N = m$ = Anzahl der Cache-Blöcke.

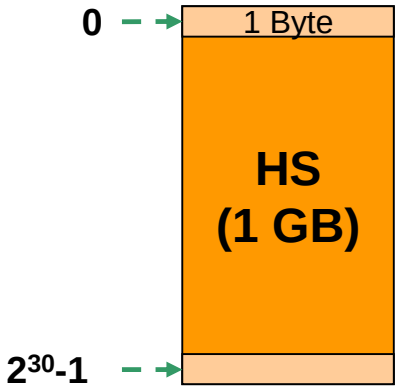


Aufg. 1.02: Cacheorganisation Lsg. (4)

Vor- und Nachteile: *N*-Wege 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.

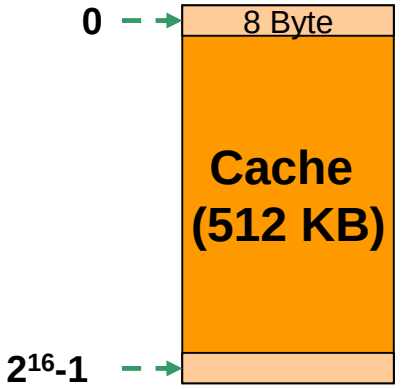
Aufg. 1.02: Cacheorganisation Lsg. (5)

b) Berechnen Sie für jeden Typ die Gesamtanzahl der Cache-Blöcke und die Anzahl der Cache-Sets. *Hinweis*: Beim Cachetyp, bei dem eine Variable *N* festgelegt werden soll, nehmen Sie den Wert 4 an.



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

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



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

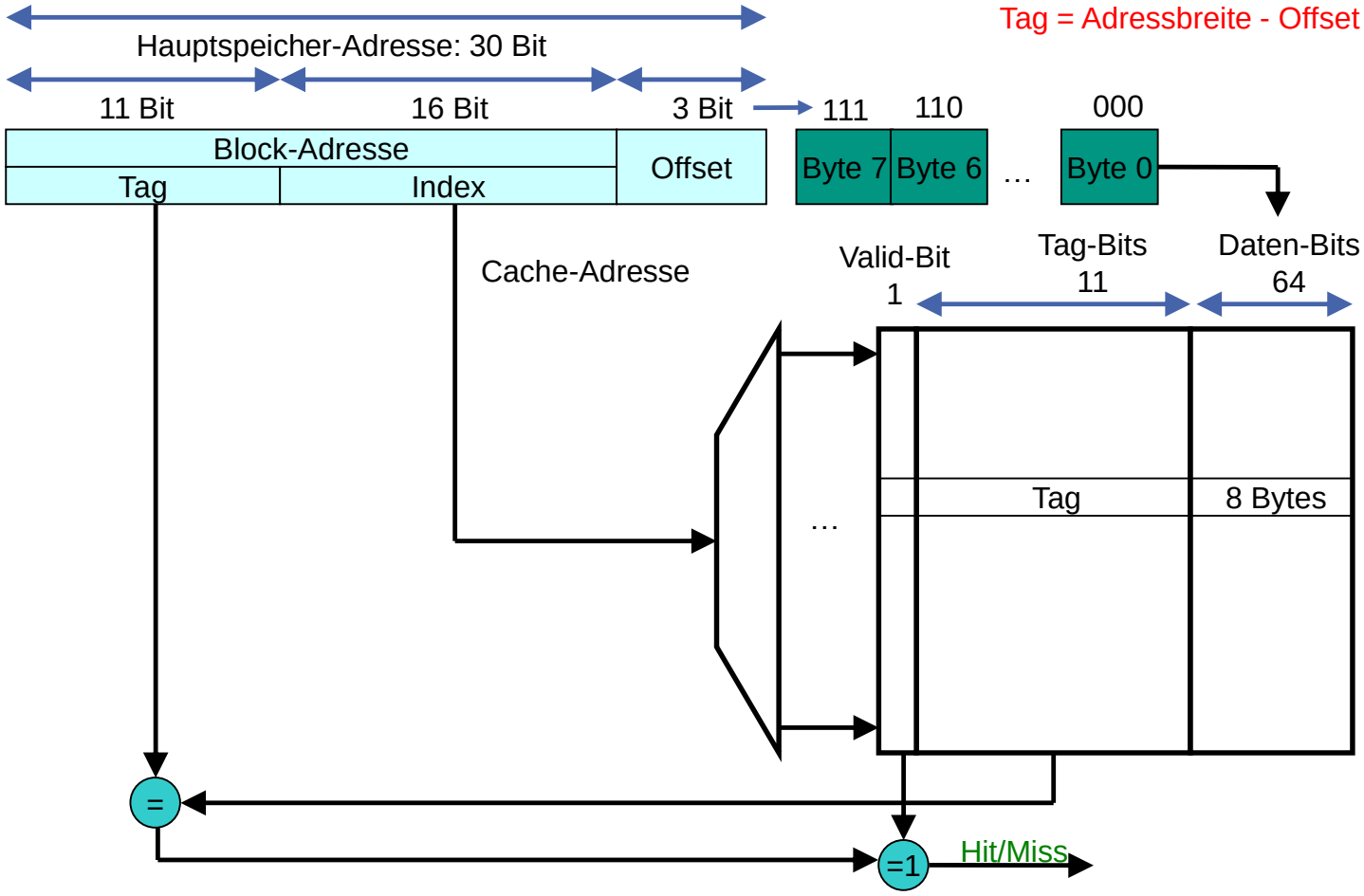
Aufg. 1.02: Cacheorganisation Lsg. (6)

c) Erläutern Sie zu jedem Typ die Speicherorganisation des Caches bezüglich im Cache abzulegende Daten und Adressierung
Hinweis: Verwenden Sie als Hilfsmittel die Abbildungen zur Cacheorganisation aus der Vorlesung.

Aufg. 1.02: Cacheorganisation Lsg. (7)

1. Direct Mapped

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



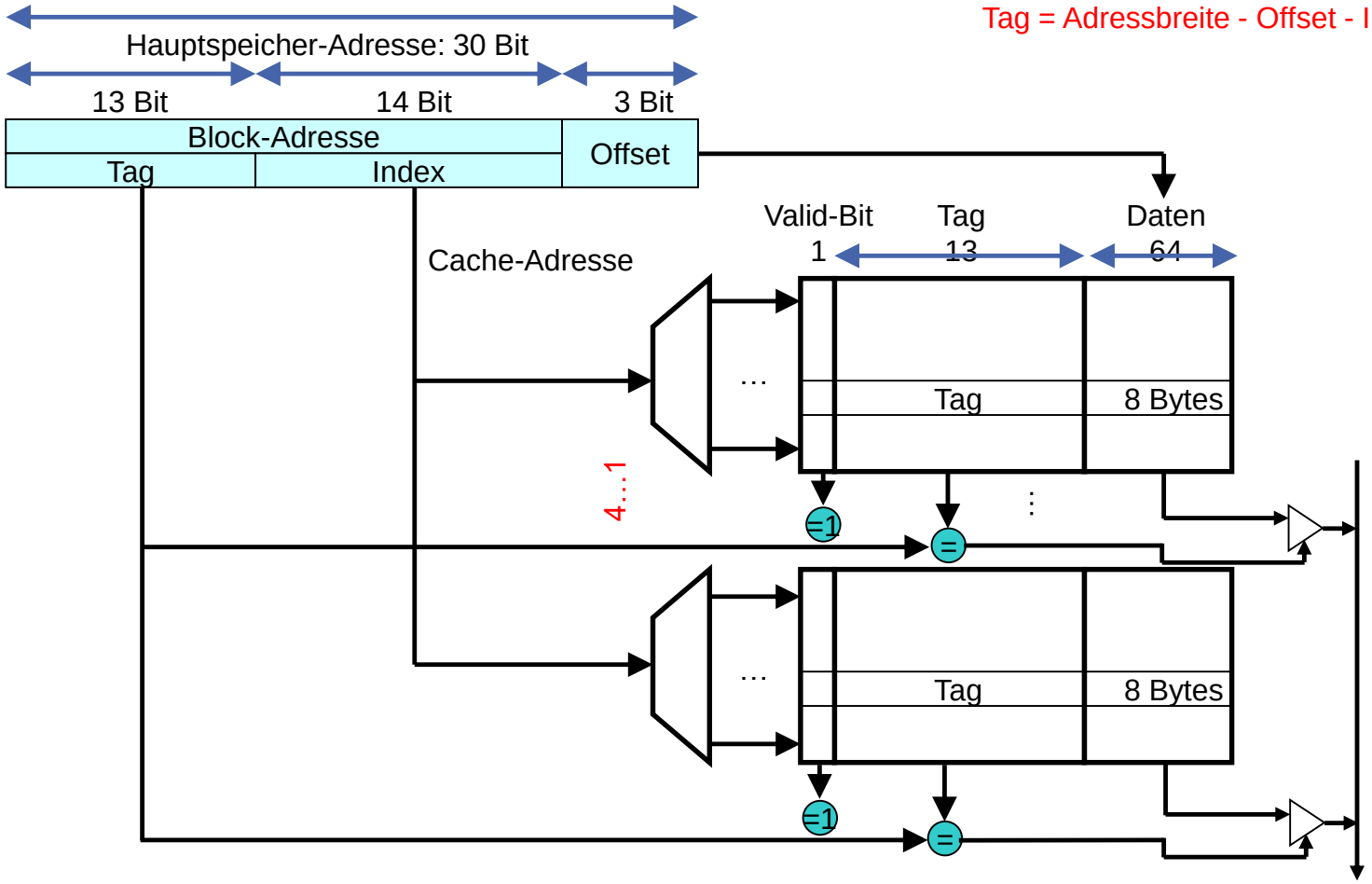
Cacheorganisation

- 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: Cacheorganisation Lsg. (8)

2. 4-Wege assoziativ

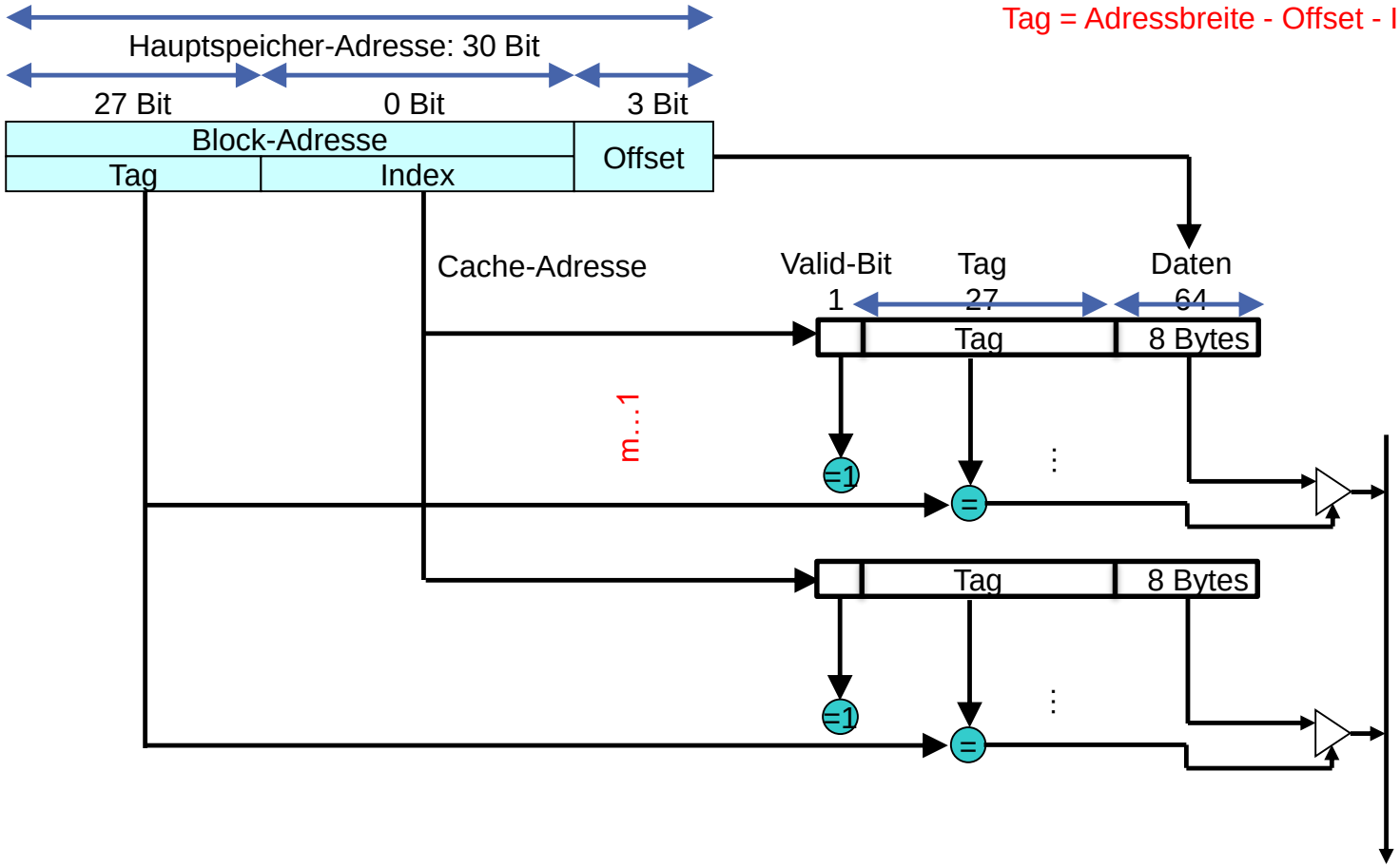
Offset = ld(Blockgröße) = ld(8) = 3
Index = ld(Blöcke/Sets) = ld(2^{16}/4) = 14
Tag = Adressbreite - Offset - Index = 13



Aufg. 1.02: Cacheorganisation Lsg. (9)

3. Voll-assoziativ

Offset = ld(Blockgröße) = ld(8) = 3
Index = ld(Blöcke/Sets) = ld($2^{16}/2^{16}$) = 0
Tag = Adressbreite - Offset - Index = 27



Aufg. 1.02: Cacheorganisation Lsg. (10)

d) Berechnen Sie für jeden Typ die in Wirklichkeit notwendige Speicherkapazität für den Cache Baustein

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-Wege 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

Aufg. 1.02: Cacheorganisation Lsg. (11)

Der benötigte Speicher ist offensichtlich nicht der ausschlaggebende Faktor bei der Auswahl! Was sonst?

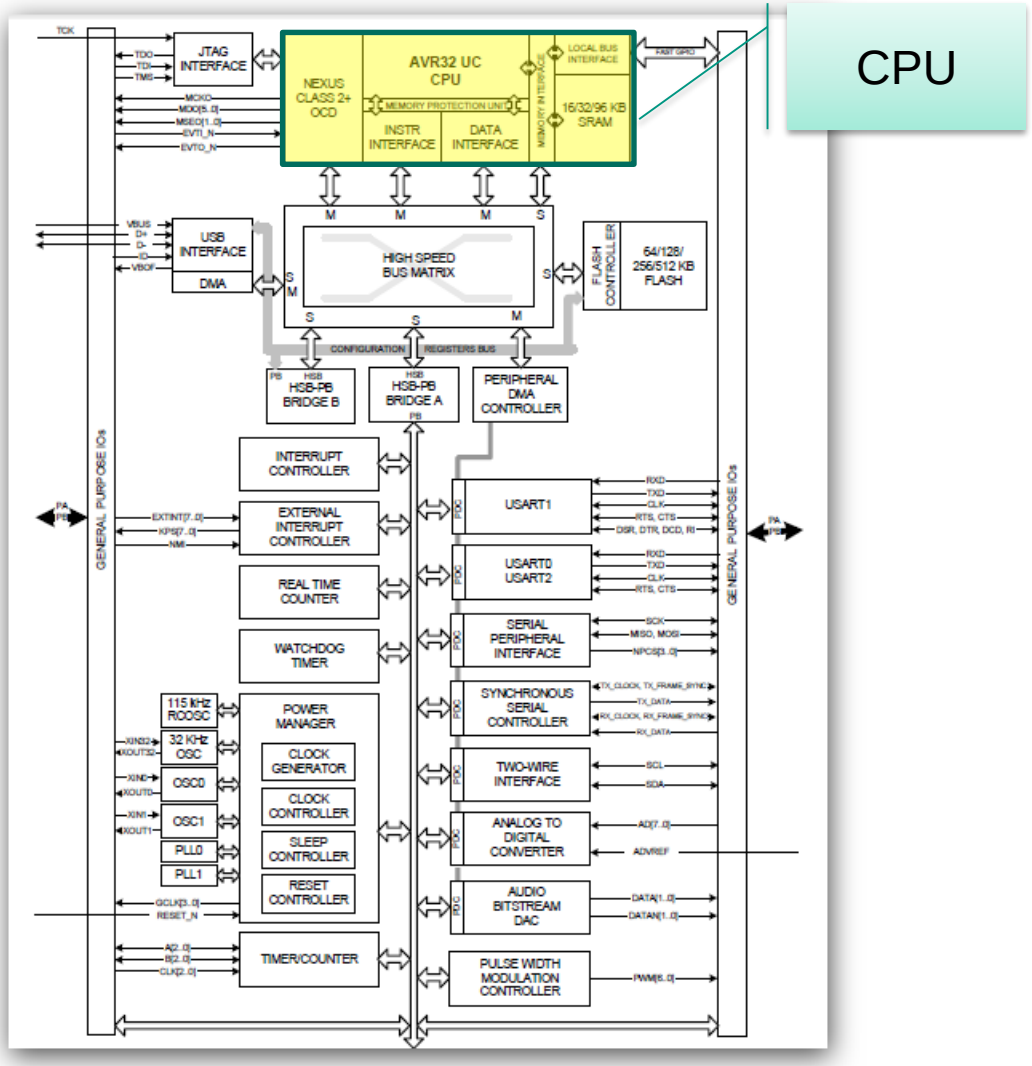
Trade-off: Anzahl der benötigten Komparatoren (Logik-Platz)

vs.

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

- Aufg. 1.02: Cacheorganisation

Exkurs µC Programmierung: Atmel AVR32



Atmel 32UC3B256 Key Features

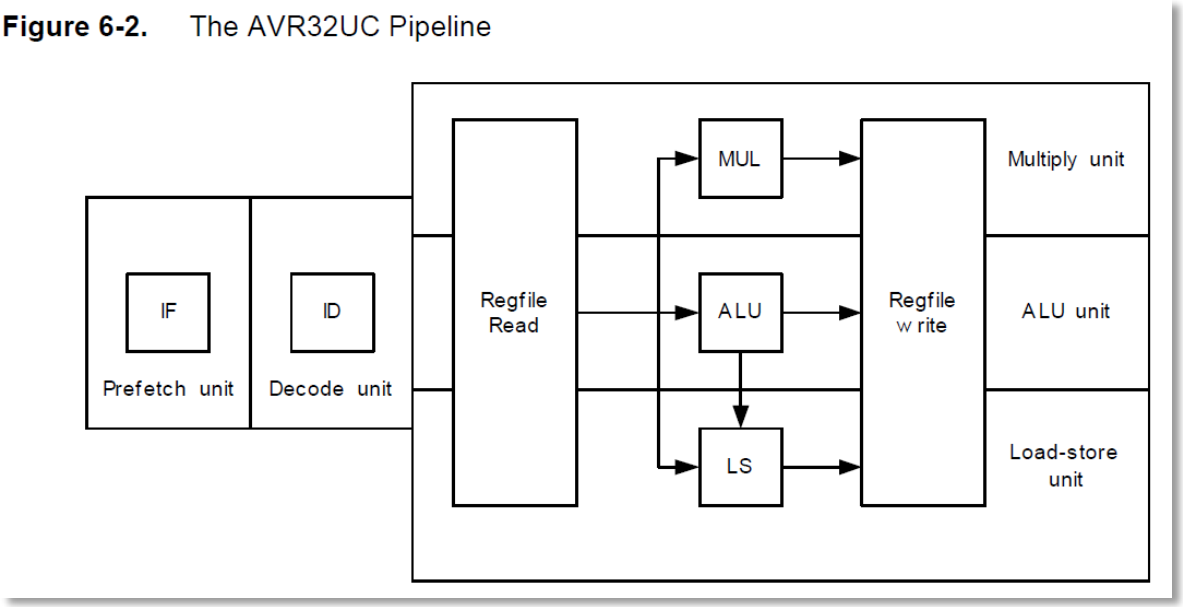
- Low Power 32 Bit RISC Architektur
- 7 DMA Kanäle
- Embedded Memory
  - 256kB Flash
  - 32kB SRAM
- Interrupt Controller
- Power Management
- USB Support
- 3-kanaliger 16 Bit Timer/Counter (TC)
- 7-kanaliger 20 Bit Pulse Width Modulation (PWM) Controller
- 3 Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
- Master/Slave Serial Peripheral Interface (SPI)
- 8-kanaliger 10 Bit Analog-Digital-Wandler
- 16 Bit Stereo Audio Bitstream Digital-Analog-Wandler
- JTAG Debug Schnittstelle
- …

Device		Embedded SRAM	Embedded Flash	USB Data	HSB-PB Bridge A	HSB-PB Bridge B
Start Address		0x0000_0000	0x8000_0000	0xD000_0000	0xFFFF_0000	0xFFFE_0000
Size	AT32UC3B0512 AT32UC3B1512	96 Kbytes	512 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0256 AT32UC3B1256	32 Kbytes	256 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B0128 AT32UC3B1128	32 Kbytes	128 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes
	AT32UC3B064 AT32UC3B164	16 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes

Atmel AVR32 – CPU Key Features

- 32 Bit Load/Store RISC Architektur
- 15 General Purpose 32 Bit Register
- 3-stufige Pipeline
- Byte, Half-Word, Word und Double Word Speicherzugriff
- Fast Interrupts mit mehreren Interrupt Prioritäts-Ebenen
- DSP Erweiterungen (Sättigungsarithmetik und HW-Multiplizierer)

Figure 6-2. The AVR32UC Pipeline

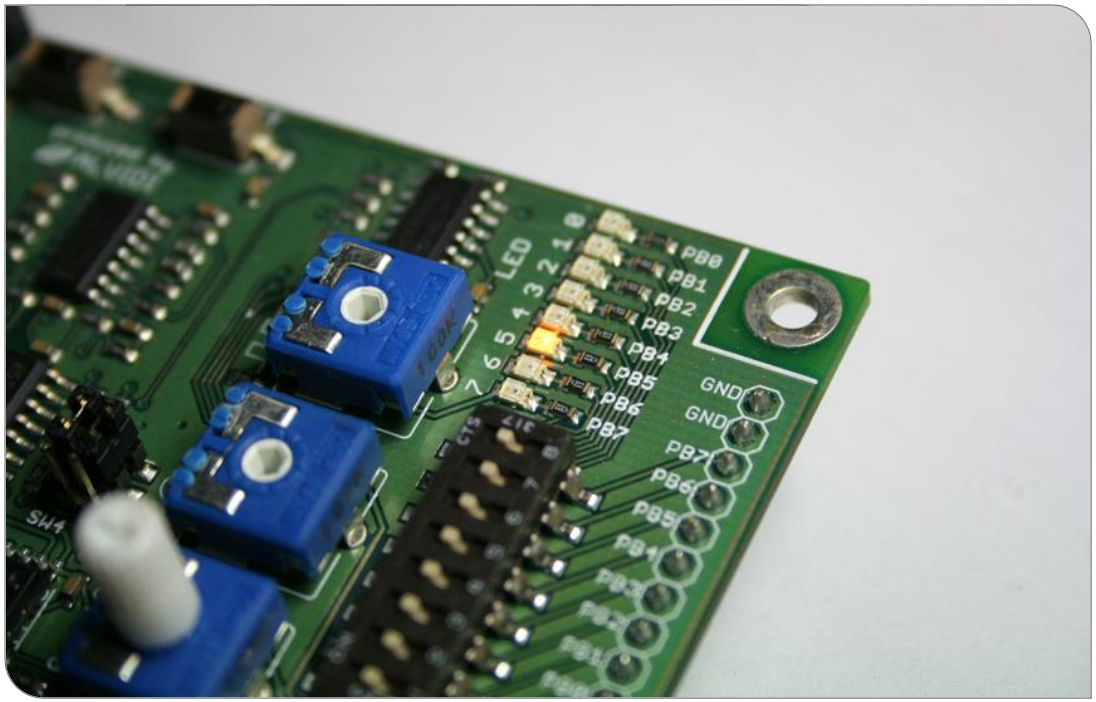


Atmel AVR32 – Memory Map

Adresse	Gerät	Name
0xFFFE0000	USB	USB 2.0 Interface - USB
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE1400	HFLASHC	Flash Controller - HFLASHC
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFF0800	INTC	Interrupt controller - INTC
0xFFFF0C00	PM	Power Manager - PM
0xFFFF0D00	RTC	Real Time Counter - RTC
0xFFFF0D30	WDT	Watchdog Timer - WDT
0xFFFF0D80	EIM	External Interrupt Controller - EIM
0xFFFF1000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF1400	USART0	Universal Synchronous/Asynchronous Receiver/Transmitter - USART0
0xFFFF1800	USART1	Universal Synchronous/Asynchronous Receiver/Transmitter - USART1
0xFFFF1C00	USART2	Universal Synchronous/Asynchronous Receiver/Transmitter - USART2
0xFFFF2400	SPI0	Serial Peripheral Interface - SPI0
0xFFFF2C00	TWI	Two-wire Interface - TWI
0xFFFF3000	PWM	Pulse Width Modulation Controller - PWM
0xFFFF3400	SSC	Synchronous Serial Controller - SSC
0xFFFF3800	TC	Timer/Counter - TC
0xFFFF3C00	ADC	Analog to Digital Converter - ADC
0xFFFF4000	ABDAC	Audio Bitstream DAC - ABDAC

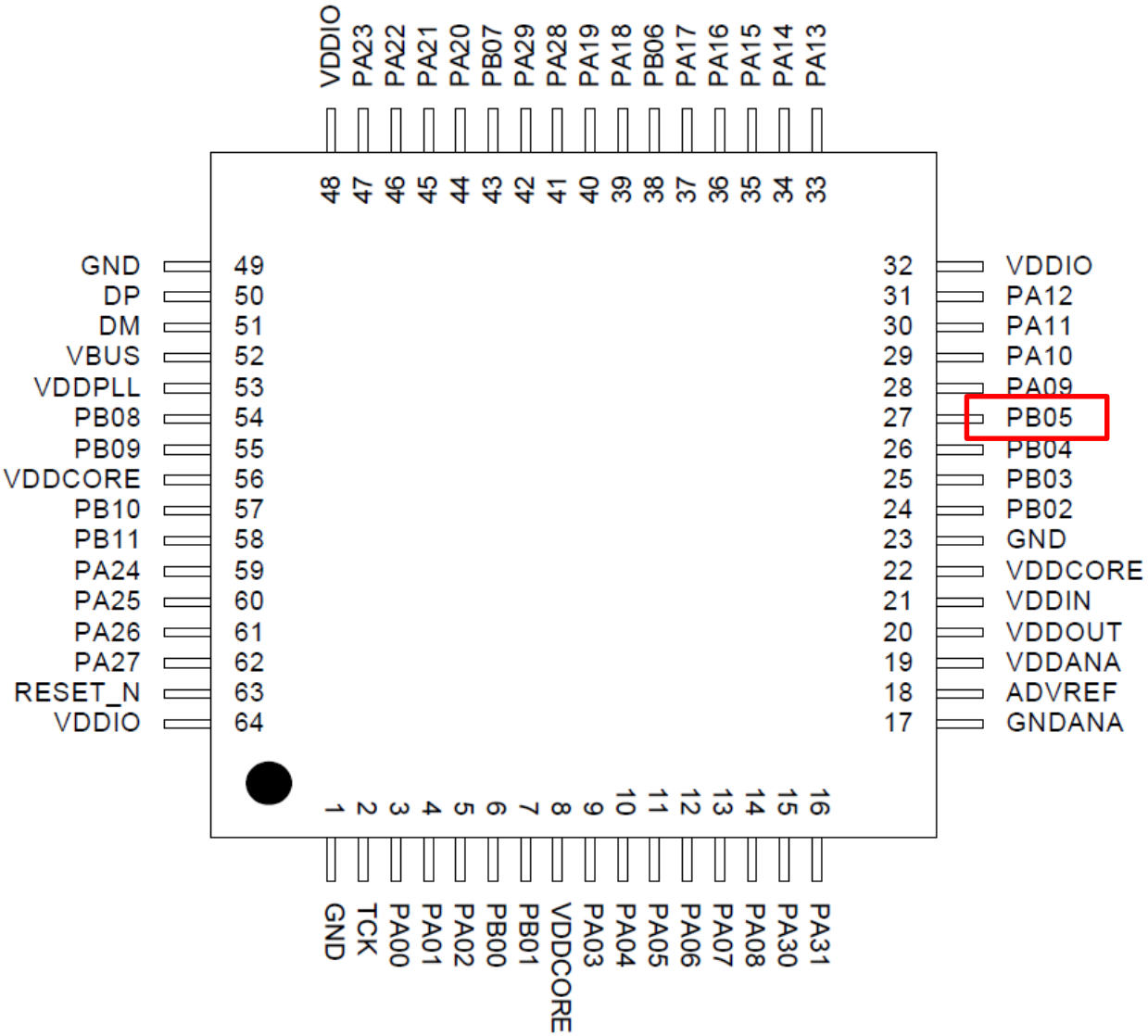
- General Purpose Input/Output Controller - GPIO
  - Basisadresse: 0xFFFF1000
  - Größe (in Bytes): **0x400** = 0xFFFF1400 – 0xFFFF1000 = 1024

Blinklicht



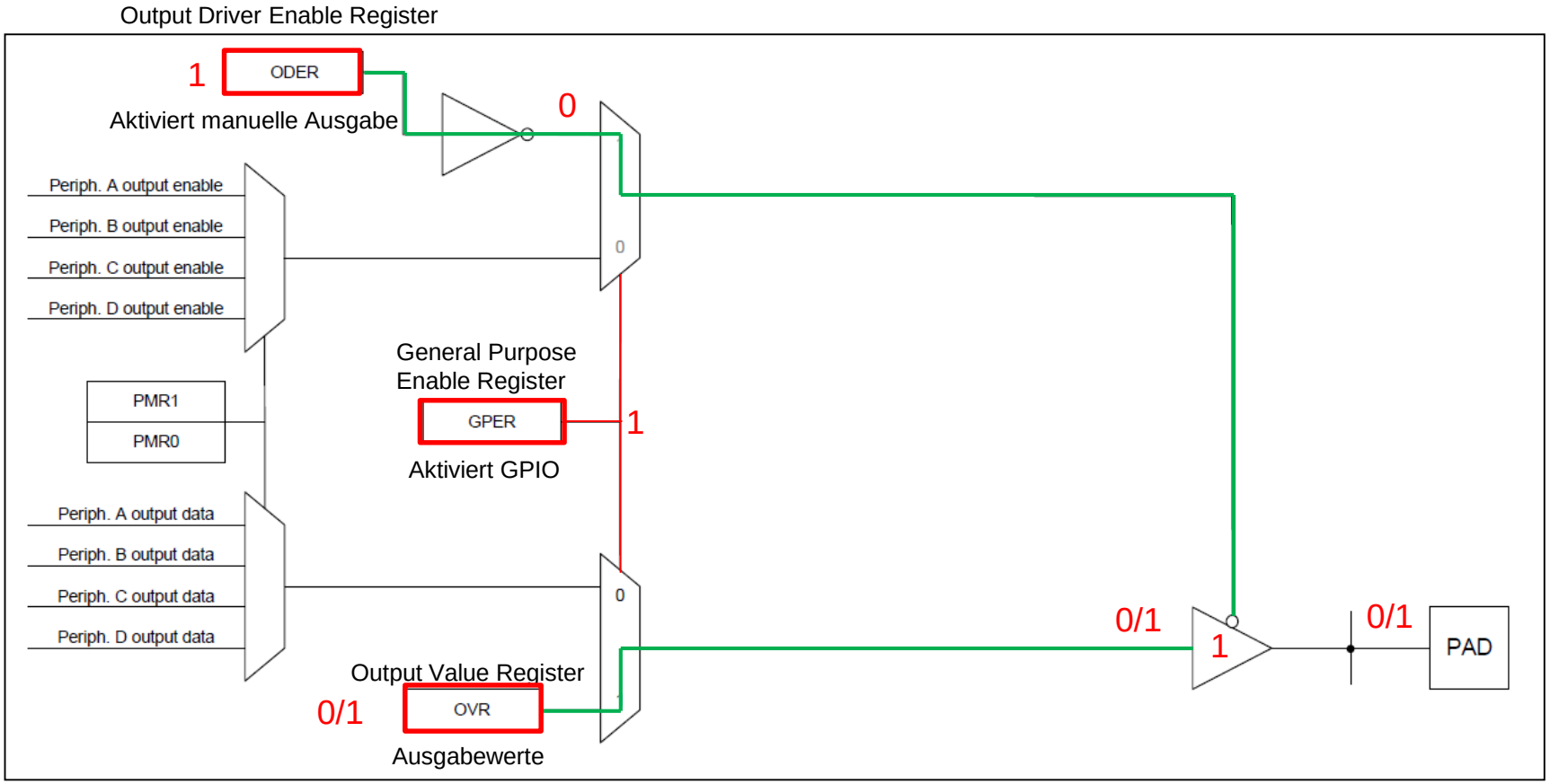
- Vorstellung
  - Blinken der LED an Pin PB05
  - Registeradressierung

ATMEL 32UC3B0512



Handbuch S. 6

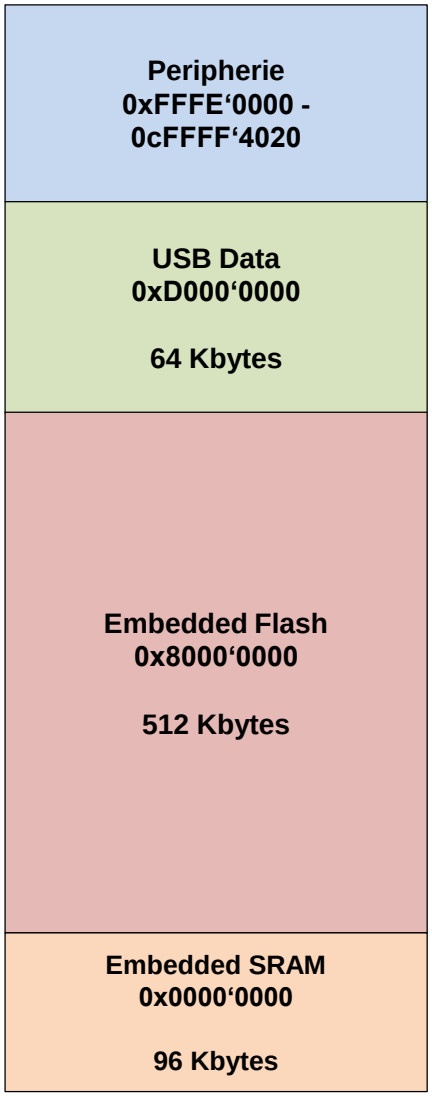
Schaltplan General Purpose Input Output



Handbuch S. 173

Blinklicht-Register

- GPIO Pin aktivieren
- Daten aus Handbuch
- Vorrechnung



Blinklicht-Register

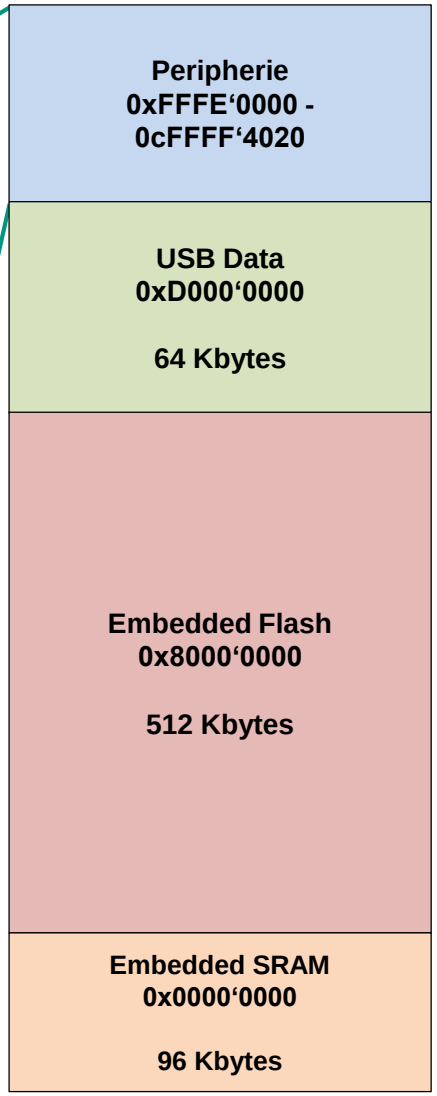
- GPIO aktvieren

7.3 Peripheral Address Map

Table 7-2. Peripheral Address Mapping

Address		Peripheral Name
0xFFFE0000	USB	USB 2.0 Interface - USB
0xFFFE1000	HMATRIX	HSB Matrix - HMATRIX
0xFFFE1400	HFLASHC	Flash Controller - HFLASHC
0xFFFF0000	PDCA	Peripheral DMA Controller - PDCA
0xFFFF0800	INTC	Interrupt controller - INTC
0xFFFF0C00	PM	Power Manager - PM
0xFFFF0D00	RTC	Real Time Counter - RTC
0xFFFF0D30	WDT	Watchdog Timer - WDT
0xFFFF0D80	EIM	External Interrupt Controller - EIM
0xFFFF1000	GPIO	General Purpose Input/Output Controller - GPIO
0xFFFF1400	USART0	Universal Synchronous/Asynchronous

0x400 = 1024 Adressen



Handbuch S.32

Blinklicht-Register – Port

- Pin PB05 → GPIO Pin 37

	25	PB03	GPIO 35	EIC - EXTINT[7]	TC - B1	USART1 - RXD	
	26	PB04	GPIO 36	USART1 - CTS	SPI0 - NPCS[3]	TC - CLK2	
	27	PB05	GPIO 37	USART1 - RTS	SPI0 - NPCS[2]	PWM - PWM[5]	
	38	PB06	GPIO 38	SSC - RX_CLOCK	USART1 - DCD	EIC - SCAN[4]	ABDAC - DATA[0]
	43	PB07	GPIO 39	SSC - RX_DATA	USART1 - DSR	EIC - SCAN[5]	ABDAC - DATAN[0]

Handbuch S.8

GPIO port = floor((GPIO number) / 32), example: floor((36)/32) = 1

Handbuch S.176

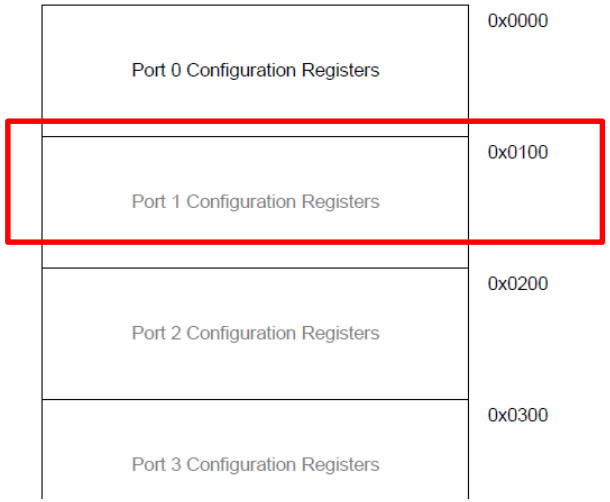
Ganzzahlig Abrunden

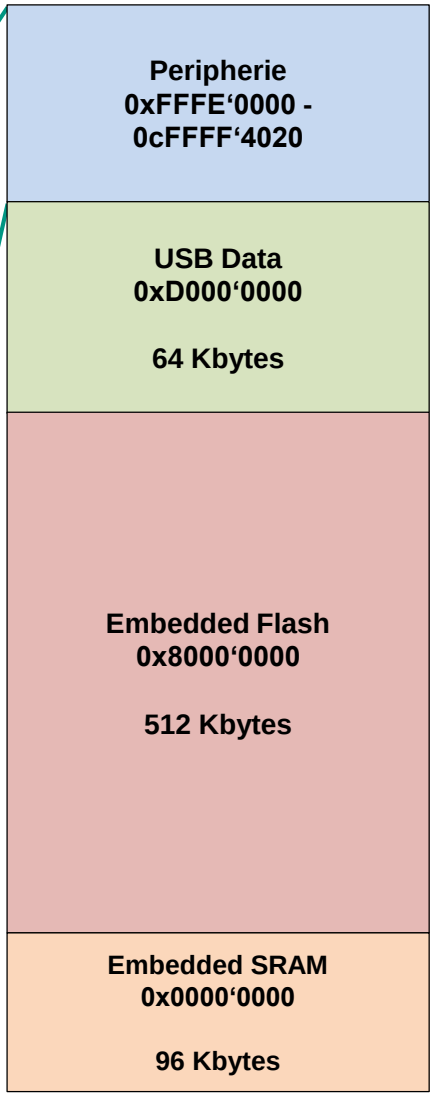
- GPIO Pin 37 → GPIO Port 1

Blinklicht-Register – Port 1

- GPIO → 0xFFFF1000
- Zugriff auf Port 1 (PB05)

Figure 17-6. Overall Mermory Map





Handbuch S.176

Blinklicht-Register – Aktivierung

- GPIO → **0xFFFF1000**
- Zugriff Port 1 → **0x00000100**
- Aktivierung von GPIO

Table 17-1. GPIO Register Memory Map

Offset	Register	Function	Name	Access	Reset value
0x00	GPIO Enable Register	Read/Write	GPER	Read/Write	(1)
0x04	GPIO Enable Register	Set	GPERS	Write-Only	
0x08	GPIO Enable Register	Clear	GPERC	Write-Only	
0x0C	GPIO Enable Register	Toggle	GPERT	Write-Only	
0x10	Peripheral Mux Register 0	Read/Write	PMR0	Read/Write	(1)
0x14	Peripheral Mux Register 0	Set	PMR0S	Write-Only	
0x18	Peripheral Mux Register 0	Clear	PMR0C	Write-Only	
0x1C	Peripheral Mux Register 0	Toggle	PMR0T	Write-Only	
0x20	Peripheral Mux Register 1	Read/Write	PMR1	Read/Write	(1)
0x24	Peripheral Mux Register 1	Set	PMR1S	Write-Only	
0x28	Peripheral Mux Register 1	Clear	PMR1C	Write-Only	
0x2C	Peripheral Mux Register 1	Toggle	PMR1T	Write-Only	
0x40	Output Driver Enable Register	Read/Write	ODER	Read/Write	(1)
0x44	Output Driver Enable Register	Set	ODERS	Write-Only	
0x48	Output Driver Enable Register	Clear	ODERC	Write-Only	
0x4C	Output Driver Enable Register	Toggle	ODERT	Write-Only	
0x50	Output Value Register	Read/Write	OVR	Read/Write	(1)
0x54	Output Value Register	Set	OVRS	Write-Only	
0x58	Output Value Register	Clear	OVRC	Write-Only	
0x5c	Output Value Register	Toggle	OVRT	Write-Only	

GPIO	**0xFFFF1000**
PORT	**0x00000100**
GPER	**0x00000000**
Summe	**0xFFFF1100** GPER

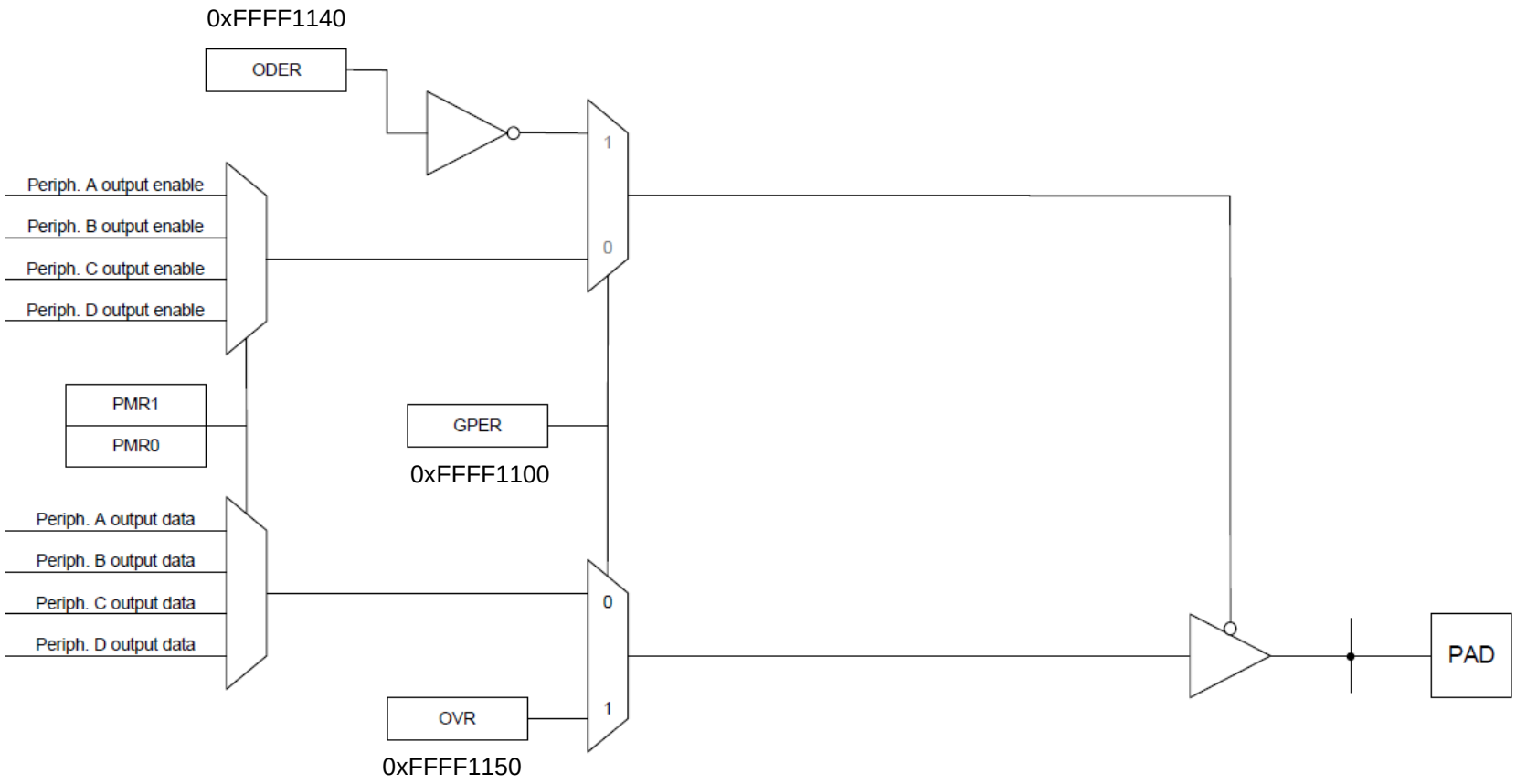
Analoge Berechnung:

0xFFFF1140 ODER

0xFFFF1150 OVR

Handbuch S.177

Schaltplan GPIO mit Registeradressen



Aufbau Enable-Register

17.6.2 Enable Register

Name: GPER

Access Type: Read, Write, Set, Clear, Toggle

Offset: 0x00, 0x04, 0x08, 0x0C

Reset Value: -

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Pin Enable**
  0: A peripheral function controls the corresponding pin.
  1: The GPIO controls the corresponding pin.

Für PB05: Binär 0b00100000
Hexadezimal 0x20

Vielen Dank für Ihre Aufmerksamkeit

Marc Weber
Karlsruher Institut für Technology (KIT) – ITIV
marc.weber3@kit.edu