Rechnerarchitektur

Labor 1

Organisatorische Aspekte

- 2 Stunden/Woche
- Labor: **90% Anwesenheit** (maximal 2 UNMOTIVIERTE Abwesenheiten)
- Anwesenheit im Labor zu erhalten -> aktive Teilnahme erforderlich

Email: dicu.madalina@yahoo.com

Prüfungsform

- Während des Semesters werden 3 Teste abgelegt

  T1: **10%** der Endnote

  T2: **10%** der Endnote

  T3: **20%** der Endnote
- Minimale Leistungsstandard: **L>=5** (um an der schriftlichen Prüfung teilzunehmen)
- BONUS (Möglichkeit): max. 10% der Endnote (für Studenten, die die minimale Leistungsstandard erfüllen)

Zahlensystem

(Nummerierungssystem)

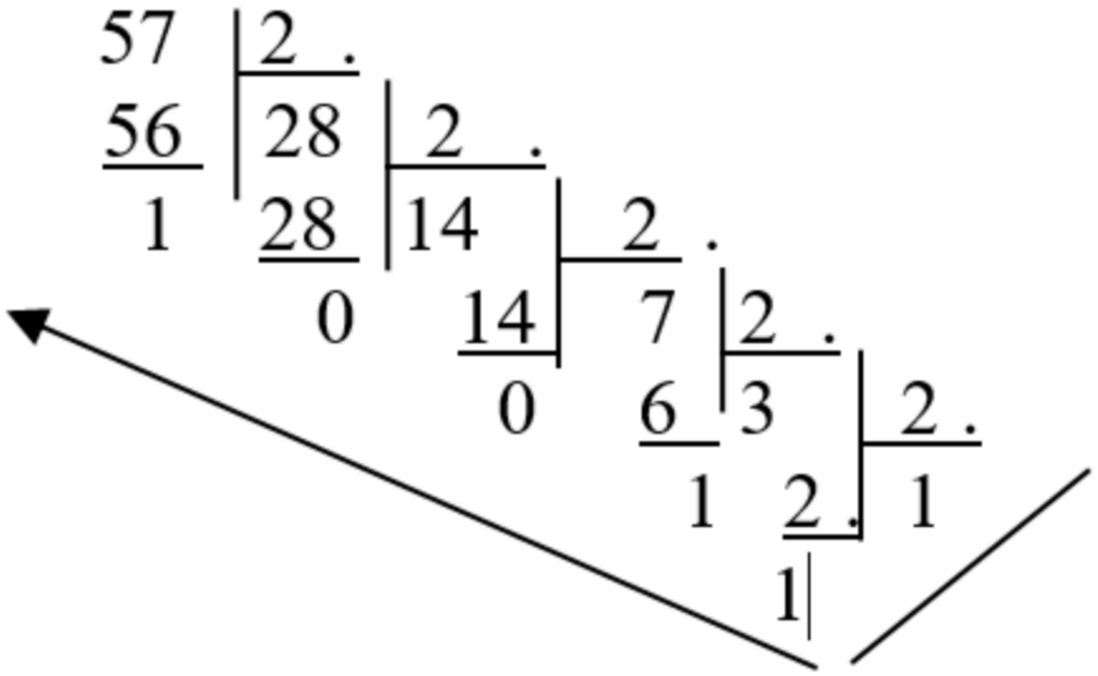
- System zur Darstellung von Zahlen nach bestimmten Regeln (mit Hilfe von Symbole = Zahlen)
- jedes Zahlensystem hat unterschiedliche Anzahl von Zeichen, der gleich mit der Basis (b) ist:
    - **Basis 2** (Binärsystem): 0, 1
    - **Basis 10** (Dezimalsystem/Zehnersystem): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - **Basis 16** (Hexadezimalsystem): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15)

Umrechnung vom Dezimalsystem (Basis 10) in ein anderes Zahlensystem

- Es wird eine ganzzahlige schriftliche Division durchgeführt und den Rest betrachten.
- **Algorithmus:** Man dividiert einfach die Dezimalzahl durch die numerische Basis des gewünschten Zahlensystems und notiert den Rest als letzte Stelle der umgerechneten Zahl. Nun dividiert man den Ganzzahlquotienten der vorhergehenden Division wieder durch die numerische Basis und notiert den Rest als vorletzte Stelle. Dies wird solange wiederholt, bis der Ganzzahlquotient gleich o ist. Am Ende werden die erhaltenen Reste in umgekehrter Reihenfolge genommen. Dies ist der Wert in der gewünschten Basis.

Beispiele

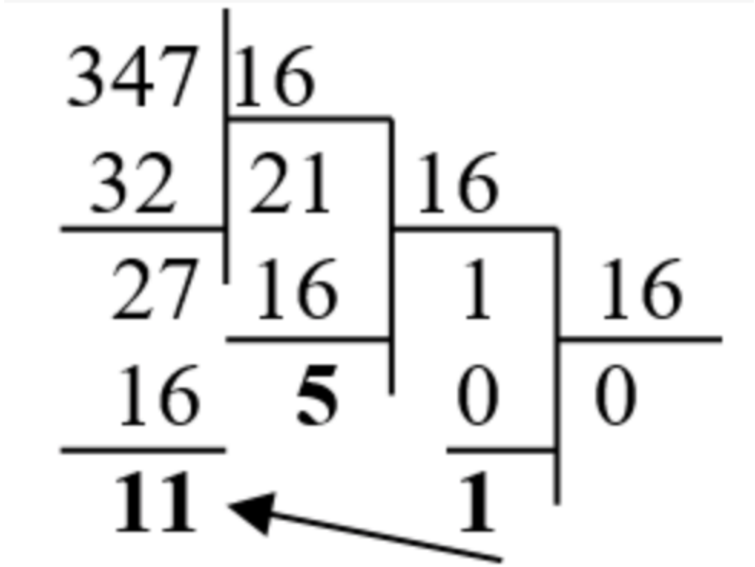
Dezimalzahl 57 in Binärzahl umwandeln:



- $57_{(D)} = 111001_{(B)}$

Beispiele

Dezimalzahl 347 in Hexadezimalzahl umwandeln:



(=**‘B’**)

- $347_{(D)} = 15B_{(H)}$

Beispiele

Dezimalzahl 61604 in Hexadezimalzahl umwandeln:

					Ganzzahlquotient	Rest
61604	:	16	=	3850 * 16 + 4	3850	4
3850	:	16	=	240 * 16 + 10	240	10 (A)
240	:	16	=	15 * 16 + 0	15	0
15	:	16	=	0 * 16 + 15	0	15 (F)

- $61604_{(D)} = F0A4_{(H)}$

Umwandlung zwischen Binärsystem (2) und Hexadezimalsystem (16)

Dezimal	Hexadezimal	Binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Beispiel

Dezimalzahl 347 in Binärzahl und Hexadezimalzahl umwandeln:

- $347_{(D)} = 15B_{(H)} = 0001\ 0101\ 1011_{(B)}$

Umrechnung in das Dezimalsystem

- Betrachten wir nun eine Zahl z mit n Stellen im Zahlensystem b (auch numerische Basis b genannt):

$$z = (z_{n-1} z_{n-2} \dots z_0)_2$$

- Diese Zahl z lässt sich folgendermaßen in eine Dezimalzahl umrechnen:

$$(z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z_0 \cdot b^0)_{10}$$

Beispiele

Hexadezimalzahl $3A8_H$ in Dezimalsystem umwandeln:

$$3A8_H = 3 \cdot 16^2 + 10 \cdot 16^1 + 8 = 3 \cdot 256 + 160 + 8 = 936_{10}$$

Hexadezimalzahl $86C_H$ in Dezimalsystem umwandeln:

$$86C_H = 8 \cdot 16^2 + 6 \cdot 16^1 + 12 = 2156_{10}$$

Binärzahl 1101101_2 in Dezimalsystem umwandeln:

$$1101101_2 = 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2 + 1 = 109_{10}$$

Das Bit. Das Vorzeichenbit. Der Komplementär Code. Darstellungsregel für ganze Zahlen mit Vorzeichen.

Das Bit

- binäre Arithmetik ist besser in der Automatisierung in Vergleich zu jeder Arithmetik in anderen Zahlensystemen -> deshalb verwenden wir Arithmetik in der Basis 2 in Computern
- weiter verwenden wir den Begriff Bit alternativ als Synonym für Binärziffer
- Das Bit = kleinste Informationseinheit
- hat zwei mögliche Werte: 0 und 1
- verschiedene Interpretationen, je nach Kontext: 0 oder 1, wahr oder falsch, positiv oder negativ, etc.

Oktett (Byte)

- bezeichnet als eine geordnete Zusammenstellung von 8 Bit (8-Bit-Sequenz), die von 0 bis 7 nummeriert ist

7 Das “HIGH” Bit	6	5	4	3	2	1	0 Das “LOW” Bit

- Byte = 8-Bit-Einheit
- Wort = 16-Bit-Einheit
- Doppelwort = 32-Bit-Einheit

Das Vorzeichenbit. Der Komplementär Code.

- Interpretation einer bestimmten Bit-Anordnung als Ganzzahl mit einem Vorzeichen (positive/negative Zahlen) -> wird ein einzelnes Bit verwendet, um das Vorzeichen einer Zahl darzustellen -> das HIGH-Bit (Bit 7) des HIGH-Bytes der Ortes, an dem die Nummer dargestellt wird
- = das erste Bit von links gibt dem Vorzeichen an

  - HIGH-Bit = **0** -> **positive** Zahl
  - HIGH-Bit = **1** -> **negative** Zahl

Wie werden die Ganzzahlen in der Vorzeichenkonvention repräsentiert ?

- **Direkter Code:** Darstellung des positiven Wert (absoluten Wert) der Zahl auf den $n-1$ Bits und auf den höchstwertigen Bit das Vorzeichen zu setzen.
    - **Problem:** -7 + 7≠ 0
- **Einerkomplelent:** Falls die Zahl positiv ist, bleibt wie es ist, falls die Zahl negativ ist, werden alle Bits der Darstellung invertiert.
    - **Problem:** -7 + 7≠ 0
- **Zweierkomplement:** Wird in der Praxis benutzt!
    - Alle Bits werden negiert und am Ende wird ein 1 an dem Ergebnis addiert

Komplementierung der Zahl 18_{10}

Anfang	00010010
Invertierung aller Bits	11101101
	11101101+
1 addieren	00000001
Komplement	11101110

$18_{10} = 12_{16} = 00010010_2$

Alternative Regeln zur Komplementieren der Zahlen

- Man beginnt von rechts nach links
- wir schreiben alle Nullen und der erste Einz ab
- alle andere Bits werden invertiert

Regeln zur Komplementieren der Zahlen

- Wir subtrahieren die binäre Zahl von der Wertebereichgrenz
- Wertebereichgrenz ist bezeichnet als $100..0_2$, wobei wobei diese Zahl n o-Ziffern hat

	100000000-
Anfang	00010010
Komplement	11101110

Regeln zur Komplementieren der Zahlen

- Wir subtrahieren die hexadezimale Zahl von der Wertebereichgrenz
- Wertebereichgrenz ist bezeichnet als $100..0_{16}$, wobei wobei diese Zahl n o-Ziffern hat

	100-
Anfang	12
Komplement	EE

Darstellung der ganzen Zahlen

- Eine ganze Zahl zwischen -2^{n-1} und 2^{n-1} wird auf n Bits folgendermaßen dargestellt:
- positive Zahl ->durch den binären Wert dargestellt
- negative Zahl -> durch den Zweierkomplement der binären Wert dargestellt

- **Bemerkung:** -2^{n-1} kann nicht auf $n-1$ Bit dargestellt werden (Platz für das Vorzeichenbit ist vorhanden). Diese Zahl wird au n Bits als $100\ldots0$ dargestellt. Das Komplement der Zahl $100\ldots0$ ist sich selbst -> der Wert -2^{n-1}.

Warum komplementärer Code?

- Implementierungen der Operationen über ganze Zahlen müssen effizient sein und unabhängig von der Darstellungskonvention
- Darstellung in komplementärer Code erfüllt diese Anforderungen am besten.
    - **Addition** wird ausgeführt, unabhängig davon, ob wir mit oder ohne Vorzeichen arbeiten. Ist eine einfache Addition auf n Bits, wobei der letzter Transport ignoriert wird.
    - **Subtraktion** wird bezeichnet als Addition zwischen Minuend und dem Komplement der Subtrahend
- Darstellungsbereich ist viel größer

Darstellungsbereiche

- Weil wir über Berechnungen reden, die von einer Maschine durchgeführt werden, gibt es eine Reihen von Einschränkungen bezüglich der Bitstellenzahlen (und somit auch für den Zahlenbereiche)
- Wichtig ist die Dimension der Darstellung (=Maximale Anzahl von Bits) einer ganzen Zahl. Die Zahlen die außerhalb einer bestimmten Bereichs können nicht dargestellt werden -> bei der Rechnung, wenn das Ergebnis außerhalb des Bereichs ist, wird die falsche Antwort angegeben.

- die Werte für n (Anzahl von Bits) für aktuelle Computer sind: 8, 16, 32
- Interpretaion ohne Verzeihen: füllen wir mit 0, die verbleibenden HIGH-Bits
- Interpretaion mit Verzeihen: füllen wir mit dem Vorzeichenbit, die verbleibenden HIGH-Bits

Beispiel

Darstellung von $(10011)_2$ als Oktett

- Interpretaion ohne Verzeihen: $(00010011)_2$
- Interpretaion mit Verzeihen: $(11110011)_2$

Anzahl Bytes	Interpretation ohne Vorzeichen	Interpretation mit Vorzeichen
1	$[0, 2^8-1] = [0 , 255]$	$[-2^7 , 2^7-1] = [-128 , 127]$
2	$[0, 2^{16}-1] = [0 , 65535]$	$[-2^{15} , 2^{15}-1] = [-32\ 768 , 32\ 767]$
3	$[0, 2^{32}-1] = [0 , 4\ 294\ 967\ 295]$	$[-2^{31} , 2^{31}-1] = [-2\ 147\ 483\ 648 , 2\ 147\ 483\ 647]$
4	$[0, 2^{64}-1] = [0 , 18\ 446\ 824\ 753389\ 551\ 615]$	$[-2^{63} , 2^{63}-1] = [-9\ 223\ 412\ 376\ 694\ 775\ 808 ,9\ 223\ 412\ 376\ 694\ 775\ 807]$