Reinhard Nitzsche
[redacted]
[redacted]
E-Mail: [redacted]

Unterrichtsentwurf

Schulpraktisches Seminar:	7. Steglitz (S)
Fachseminar:	Informatik
Fachseminarleiter:	Herr Witten
Unterrichtsfach:	Informatik
Thema der Stunde:	**Rechnerinterne Addition**
Ort:	OSZ Handel I Abteilung V Wrangelstraße 98 10997 Berlin
Bildungsgang:	Gymnasiale Oberstufe
Klasse:	11[redacted]
Raum:	[redacted]
Datum:	[redacted] 2002
Zeit:	11.40 bis 12.25 Uhr (1. Stunde des 3. Blocks)

1 Entscheidungsvoraussetzungen

1.1 Angaben zur Klasse und zum Lehrer

Die Teilungsgruppe besteht aus 13 Schülern (7 Schüler und 6 Schülerinnen).[1]

Alle Schüler sind in diesem Schuhljahr neu an das OSZ Handel I gekommen. Die Teilnahme am Informatikunterricht ist für sie als Teil der Berufsfeldorientierung „Wirtschaft“ verbindlich.

4 Schüler der Teilungsgruppe nehmen zusätzlich am so genannten „Profilkurs Informatik“ teil, der mit einer Informatik-AG vergleichbar ist.

[1] Aus Gründen der besseren Lesbarkeit verwende ich im Folgenden die männliche Form gleichermaßen für Schülerinnen und Schüler.

Altersverteilung:

Geburtsjahr	1983	1984	1985	1986
Anzahl	1	3	6	3

Zwei Schüler haben zuvor ein Gymnasium besucht, die anderen haben auf Realschulen die Berechtigung zum Besuch der gymnasialen Oberstufe erworben.

Das Leistungsniveau und die Motivation der Lerngruppe schätze ich als durchschnittlich ein. Im Informatikunterricht sind Leistungen und Motivation etwas niedriger als in Wirtschaftslehre.

Die Schüler pflegen untereinander einen freundlichen Umgang.

Ich bin seit Beginn meines Referendariats im Mai 2003 am OSZ Handel und unterrichte die Teilungsgruppe seit Schuljahresbeginn. Ebenfalls seit Schuljahresbeginn unterrichte ich zwei der fünf Stunden Wirtschaftslehre in der gesamten Klasse.

1.2 Stellung der Stunde im Unterricht

In den bisherigen Stunden fand eine Einführung in das Internet statt:

- Funktionsweise von E-Mail, Netiquette
- Suchen und Finden im Internet
- Funktionsweise von HTTP
- Datenübertragung

In der vorangegangenen Stunde habe ich in das heutige Unterrichtsthema Stellenwertsysteme eingeführt und hier exemplarisch das 3er-System behandelt. Die Umrechung vom Dezimalsystem in das 3er-System und umgekehrt, wurde ausschließlich anhand des Stellenwertes geübt. Das Dualsystem wurde bisher nur gestreift.

Nach den Herbstferien sollen noch rechnerinterne Textdarstellungen behandelt werden, bevor eine Unterrichtsreihe zum Thema Kryptographie beginnt.

2 Inhalts- und Zielentscheidungen

2.1 Sachanalyse

2.1.1 Datenspeicherung

Die heutigen Rechner basieren auf dem von-Neumann-Universalrechner-Konzept. Es sieht unter anderem einen gemeinsamen Speicher für Programmcode und Daten vor. Sein Inhalt ist

binär kodiert. Den einzelnen Speicherzellen kann man nicht ansehen, ob es sich um Programmcode oder Daten handelt. Auch kann der Typ der Daten am Speicherinhalt nicht erkannt werden.

Die Interpretation des Speicherinhaltes obliegt ausschließlich dem Programm. Die Speicherzellen werden zu größeren Einheiten zusammengefasst. Die kleinste Einheit, auf die über einen Speicherbefehl zugegriffen werden kann, heißt Wort und umfasst bei heutigen PCs 16 oder 32 Bit. Daneben können intern von der CPU meist auch Daten mit weniger Bits verarbeitet werden, zum Laden und Speichern müssen jedoch ganze Wörter verwendet werden.

Höhere Programmiersprachen beherrschen auch die Verarbeitung von Datenstrukturen, die im Speicher mehr als ein Wort belegen.

2.1.2 Beschränkungen durch Hardware

Anders als beim Rechnen mit Papier und Bleistift sind Zahlen im Rechner Beschränkungen hinsichtlich ihres Wertebereiches unterworfen. Dies liegt daran, dass

- die CPU nur Daten bis zu einer bestimmten Länge manipulieren können und
- den Zahlen im Speicher bzw. im Prozessor ein Speicherplatz fester Länge zugewiesen wird.

Für spezielle Anwendungen gibt es Möglichkeiten, diese Bechränkungen aufzuheben. Auf Grund des damit verbundenen Rechenaufwandes stehen diese Möglichkeiten für die meisten Einsätze nicht zur Verfügung.

2.1.3 Zahldarstellungen in Digitalrechnern

Zu unterscheiden ist zwischen der *Ziffernkodierung* und der *Zahlkodierung*.

Bei der Ziffernkodierung wird jede Dezimal*ziffer* für sich kodiert und in einem Halbbyte gespeichert (z. B. BCD, Binary Coded Decimal: binäre Kodierung der Ziffern). Da in einem Halbbyte 16 Werte kodiert werden können, aber nur 10 Ziffern benötigt werden, ist mit der Ziffernkodierung ein unnötig großer Speicherverbrauch verbunden, der jedoch für zusätzliche Prüfmöglichkeiten verwendet werden kann. Anders als Zahlkodierungen haben ziffernkodierte Zahlen einen „natürlichen“ Wertebereich, der in Dezimalstellen angegeben werden kann, ein Umstand, der z. B. in finanzmathematischen Applikationen von Interesse ist. Die Ziffernkodierung wird in der Regel nicht auf Prozessorebene implementiert.

Bei der Zahlkodierung wird die zu kodierende Zahl als Ganzes kodiert und über mehrere (Halb-)Bytes hinweg gespeichert.

Schließlich muss noch berücksichtigt werden, ob das höchstwertige Bit (MSB, Most Significant Bit) am „linken“ Rand (also neben dem letzten Bit des vorangegangenen Speicherwortes) oder am „rechten“ Rand (also neben dem ersten Bit des folgenden Speicherwortes) steht.

2.1.3.1 Vorzeichenlose Ganzzahlen Vorzeichenlose Zahlen sind einfach zu kodieren. Sie müssen lediglich dual kodiert werden und mit Nullen am linken Rand bis zum Erreichen des Endes des zugewiesenen Speicherbereichs aufgefüllt werden.

Bei Ganzzahlen kommen keine Rundungsfehler vor, alle Ergebnisse sind genau.

2.1.3.2 Vorzeichenbehaftete Ganzzahlen Für vorzeichenbehaftete Ganzzahlen hat sich die Darstellung des Zweierkomplements durchgesetzt. Das Zweierkomplemnt wird gebildet, indem zunächst der Betrag dual kodiert, dann bitweise negiert und schließlich noch 1 dazu addiert wird:

-4 im Zweierkomplement im Halbbyte: $4_{10} = \texttt{0100}_2$, negiert: `1011`, Addition von 1 führt zu `1100`.

Wird hiervon wiederum das Zweierkomplement gebildet, erhält man wiederum die Ausgangszahl.

Wichtige Eigenschaften des Zweierkomplements (Beispiele $n = 4$ Bits, Halbbyte):

1. Das Zweierkomplement des Zweierkomplements einer Zahl ist die Zahl selbst.
2. Die Darstellung nicht negativer Zahlen ist gegenüber vorzeichenloser Zahlen unverändert.
3. Das erste Bit gibt das Vorzeichen der Zahl an (`1`: negativ).
4. Das höchstwertige Bit hat den Wert -2^n (-16). So kann das Zweierkomplement auch mit diesem Wissen gebildet werden.
5. Die kleinste Zahl ist die `1000` (-8), `1111` repräsentiert -1.
6. Die größte Zahl ist die `0111` (7).
7. Der Betrag der kleinsten Zahl ist um 1 größer als die größte Zahl.

2.1.3.3 Gleitkommazahlen Für Berechnungen mit reellen Zahlen oder sehr großen Ganzzahlen wird ein anderes Zahlenformat benötigt. An die Genauigkeit werden hier meist keine so hohen Anforderungen gestellt wie bei Ganzzahlen. Durchgesetzt hat sich der IEEE-Standard, der eine gute Kombination aus Wertebereich und Genauigkeit liefert:

Standard		IEEE 754 (32 Bit)	IEEE 854 (64 Bit)
		Bit	Bit
Vorzeichen	S	31 (1 Bit)	63 (1 Bit)
Exponent	E	23–30 (8 Bit)	52–62 (11 Bit)
Mantisse	F	0–22 (23 Bit)	0–51 (52 Bit)

Der repräsentierte Wert ist dann $(-1)^S \cdot (1+F) \cdot 2^E$, wobei die Mantisse als Nachkommanteil der normalisierten wissenschaftlichen Schreibweise interpretiert wird.

2.1.4 Rechnen mit Ganzzahlen

Mit Zahlen im Zweierkomplement kann genauso gearbeitet werden wie mit vorzeichenlosen Zahlen, lediglich die Überläufe erfordern eine abweichende Behandlung.

2.1.4.1 Addition Bei der Addition von vorzeichenlosen Binärzahlen kann nach dem aus der Grundschule bekannten Algorithmus vorgegangen werden:

	0	0	1	1	(3_{10})
	0	0	0	1	(1_{10})
0	*0*	*0*	*1*	*0*	(Überträge)
	0	1	0	0	Summe (4_{10})

Jedoch kann es passieren, dass das Ergebnis auf Grund der Beschränkung der Stellen, fehlerhaft ist:

	1	1	0	1	(13_{10})
	1	1	0	1	(13_{10})
1	*1*	*0*	*1*	*0*	(Überträge)
	1	0	1	0	Summe (10_{10})

Das Ergebnis der Rechnung $13+13$ ist in diesem Fall 10. Diesen Fehler kann man daran erkennen, dass ein Übertrag aus dem höchstwertigen Bit heraus erfolgte. Man spricht von einem *Überlauf*, der vom Programm erkannt und bearbeitet werden sollte.

Bei vorzeichenbehafteten Zahlen wird die Sache ein wenig komplizierter:

	1	1	0	1	(-3_{10})
	1	1	0	1	(-3_{10})
1	*1*	*0*	*1*	*0*	(Überträge)
	1	0	1	0	Summe (-6_{10})

Die Rechnung ist dieselbe wie beim zweiten Beispiel $(13+13)$, jedoch ist das Ergebnis diesmal bei Interpretation als Zweierkomplement korrekt.

Ein Überlauf kann hier in zwei Fällen vorkommen:

A	B	$A+B$
≥ 0	≥ 0	< 0
< 0	< 0	≥ 0

Diese Bedingungen sind sehr einsichtig, können aber hardwaremäßig nicht mit vertretbarem Aufwand überprüft werden. Daher wird in der Regel geprüft, ob der Übertrag aus dem und in

das höchstwertige Bit ungleich sind. Sind sie es, so hat ein Überlauf stattgefunden.

Zu beachten ist, dass die Überprüfung auf Überlauf abhängig von der Interpretation der Bitfolgen ist.

2.1.4.2 Subtraktion Die Subtraktion wird durch Addition des negierten Subtrahenden realisiert.

2.1.4.3 Multiplikation, Division Die „Punktrechenarten“ werden verblüffend ähnlich wie die bekannten Grundschulalgorithmen berechnet.

2.2 Stoffauswahl

Die Interpretationsnotwendigkeit von Speicherinhalten wird in einer der folgenden Stunden behandelt werden, wenn die Schüler andere Interpretationsmöglichkeiten kennen lernen. Die Ziffernkodierung findet rechnerintern kaum Verwendung, so dass auf ihre Darstellung völlig verzichtet wird. Das LSB/MSB-Problem wird als rein technisches Problem ausgeblendet.

Die Repräsentation von Gleitkommazahlen ist komplex und auf Grund ihres mathematischen Hintergrundes problematisch. Außerdem sind Algorithemn für Gleitkommazahlen schwierig zu verstehen, so dass auf Gleitkommazahlen komplett verzichtet wird.

Bei den Ganzzahlen ist die Unterscheidung vorzeichenlos/vorzeichenbehaftet auf Grund der Datentypen in Delphi beim Beginn der Programmierung in jedem Fall zu machen, so dass aus diesem Grund die Behandlung beider Repräsentationen im Verlauf der Unterrichtsreihe angezeigt ist. Es ist zweckmäßig, zunächst mit den einfacheren vorzeichenlosen Zahlen zu arbeiten, bevor in der folgenden Stunde dann im Rahmen der Subtraktion auf das Zweierkomplement eingegangen wird.

Die Rechenoperationen sollen auf die Addition und die auch auf Hardwareebene daraus abgeleitete Subtraktion beschränkt werden, sie ermöglichen bereits ausreichend tiefe Einblicke in die rechnerinterne Datendarstellung und -bearbeitung.

2.3 Lernziele

2.3.1 Fachliche Lernziele

Nach der Teilnahme an dieser Unterrichtsstunde können die Schülerinnen und Schüler

1. Dezimalzahlen in ihre binäre Halbbyte-Darstellung überführen,
2. die Arithmetik des Rechners als Folge einfachster Entscheidungen beschreiben,

3. darstellen, dass die Begrenzung der Speicherplätze Ursache für Überläufe ist,
4. die Situationen nennen, in denen es zu Überläufen kommt sowie
5. erklären, warum Überläufe eine Problem sind.

2.3.2 Instrumentelle Lernziele

Die Schüler sollen in dieser Stunde lernen, die abstrakte rechnerinterne Zahlenrepräsentation für einen gewählten Zweck zu veranschaulichen.

3 Methodische Entscheidungen

3.1 Übersicht zur Verlaufsplanung

Phase, Dauer, Zeit	Inhalt	Medien	Lehrform	Sozialform
I (3 min) **11.40**	Begrüßung	–	–	–
II (5 min) **11.43**	Einstieg in Additionsspiel	Spielplan	–	Lehrerervortrag
III (12 min) **11.48**	Durchführung Additions-Spiel	Spielplan	induktiv	Schülertätigkeit
IV (5 min) **12.00**	Ergänzung: Addieren mit Überlauf	Tafel	fragend-entwickelnd	Unterrichtsgespräch
V (10 min) **12.05**	Sicherung	Arbeitsblatt 1	selbsterarbeitend	Einzel- oder Partnerarbeit
VI (5 min) **12.15**	Lösungsvergleich	Arbeitsblatt 1	–	Schülertätigkeit
VII (5 min) **12.20**	Übergangsphase zur Subtraktion	–	fragend-entwickelnd	Unterrichtsgespräch

3.2 Methodischer Gang

I (3 min) **11.40**	Begrüßung	–	–	–

II (5 min) **11.43**	Einstieg in Additionsspiel	Spielplan	–	Lehrerervortrag

Im Rahmen eines Spieles, das einen Vier-Bit-Volladdierer simuliert, addieren die Schüler Binärzahlen. Der Addierer wird als Spielplan (siehe Seite 11) auf den Tischen ausgelegt. Auf dem Spielplan ist ein intuitiv zu verstehender Volladdierer aufgezeichnet. Die Schaltelemente werden in der Stunde nicht weiter erörtert.

Auf die Schaltelemente des Spielplanes wird jeweils eine Spielkarte gelegt, deren Vorderseite eine „0“ und deren Rückseite eine „1“ zeigt.

Vier Schüler simulieren jeweils ein Schaltelement, dass ein Bit des Ergebnisses liefert (unterste Reihe im Spielplan). Vier andere Schüler simulieren jeweils ein Schaltelement, dass den Übertrag generiert (zweite Reihe von unten im Spielplan). Die übrigen Schüler sind für das Einstellen der Operanden (obere Reihen im Spielplan) und die Dokumentation der Rechnungen zuständig.

Ich erkläre den Schülern zunächst die Regeln des Additions-Spiels.

III (12 min) **11.48**	Durchführung Additions-Spiel	Spielplan	induktiv	Schülertätigkeit

Das Spiel wird mit drei Rechenaufgaben durchgeführt. Bei der dritten Aufgabe wird ein Überlauf produziert.

Die Schüler erkennen im Laufe des Spiels, dass das Ergebnis (unterste Zeile) die Summe der beiden Operanden darstellt.

Der Spielplan wird entfernt.

IV (5 min) **12.00**	Ergänzung: Addieren mit Überlauf	Tafel	fragend-entwickelnd	Unterrichtsgespräch

Gemeinsam mit den Schülern wird eine Aufgabe nach dem Grundschulalgorithmus erarbeitet, das Tafelbild entspricht dem Beispiel auf Arbeitsblatt 1.

V (10 min) **12.05**	Sicherung	Arbeitsblatt 1	selbsterarbeitend	Einzel- oder Partnerarbeit

Die Schüler bearbeiten einige Additionsaufgaben mit und ohne Überlauf.

VI (5 min) **12.15**	Lösungsvergleich	Arbeitsblatt 1	–	Schülertätigkeit

Schließlich wird eine Aufgabe von einem Schüler vorgerechnet und die verbleibenden Ergebnisse werden verglichen.

VII (5 min) **12.20**	Übergangsphase zur Subtraktion	–	fragend-entwickelnd	Unterrichtsgespräch

Die Schüler werden aufgefordert, sich darüber Gedanken zu machen, wie rechnerintern subtrahiert wird. Ich erwarte, dass vorgeschlagen wird, einen 4-Bit-Vollsubtrahierer zu bauen, die Schüler aber bald merken, dass dies ziemlich aufwändig wäre. Idealerweise kommen die Schüler darauf durch Addition negativer Zahlen zu subtrahieren.

In der folgenden Stunde wird dann das Zweierkomplement als Repräsentation negativer Zahlen vorgestellt. Dies kann als Eventualphase angesehen werden.

4 Grundlagen der Unterrichtsvorbereitung

[1] COY, WOLFGANG: *Aufbau und Arbeitsweise von Rechenanlagen*. Vieweg, zweite Auflage, 1992.

[2] HENNESY, JOHN L. und DAVID A. PATTERSON: *Computer Organisation and Design. The Hardware/Software Interface*. Morgan Kaufmann Publishers, 1994.

[3] LEXIKONREDAKTION, MEYERS (Herausgeber): *Duden. Informatik. Ein Fachlexikon für Studium und Praxis*. Dudenverlag, dritte Auflage, 2001.

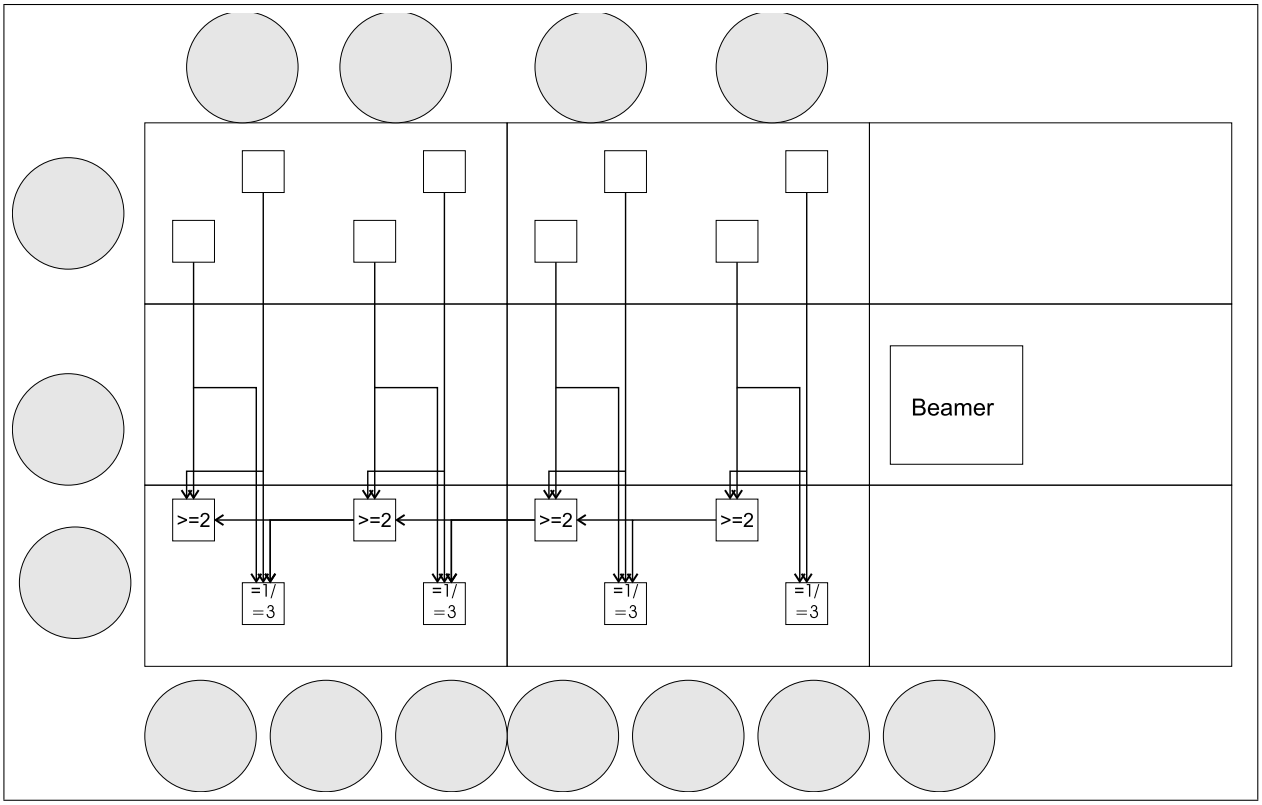
[4] TANENBAUM, ANDREW S.: *Structured Computer Organisation*. Prentice-Hall International, dritte Auflage, 1990.

A Anlagen

Wenn nicht anders angegeben beträgt das Originalformat der Arbeitsbogen und OH-Folien DIN A4.

Verzeichnis der Anlagen

A.1 Spielplan



Größe: $260cm \cdot 195cm$

A.2 Arbeitsblatt 1

Rechnerinterne Addition

Beispiel ($3 + 15$):

	0	0	1	1	($3_{10} = 2 + 1$)
	1	1	1	1	($15_{10} = 8 + 4 + 2 + 1$)
1	1	1	1	0	(Überträge)
	0	0	1	0	Summe ($2_{10} = 2$) **Überlauf!**

Tritt bei der Addition vorzeichenloser Zahlen ein Übertrag *aus* dem *höchstwertigen Bit* (dem am weitesten links stehenden Bit) auf, so kann das Ergebnis mit den zur Verfügung stehenden Bits nicht mehr korrekt dargestellt werden, es ist ein *Überlauf* aufgetreten.

höchstwertiges Bit

Überlauf

Arbeitsauftrag: Führen Sie die folgenden Additionen durch und entscheiden Sie, ob ein Überlauf statt gefunden hat oder ob nicht.

(1) $4 + 7$:

		(4_{10})
		(7_{10})
		(Überträge)
		Summe ( )

❒ Überlauf ❒ kein Überlauf

(2) $1 + 15$:

		(1_{10})
		(15_{10})
		(Überträge)
		Summe ( )

❒ Überlauf ❒ kein Überlauf

(3) $3 + 5$:

		(3_{10})
		(5_{10})
		(Überträge)
		Summe ( )

❒ Überlauf ❒ kein Überlauf

(4) $10 + 9$:

		(10_{10})
		(9_{10})
		(Überträge)
		Summe ( )

❒ Überlauf ❒ kein Überlauf

(5) $11 + 7$:

		(11_{10})
		(7_{10})
		(Überträge)
		Summe ( )

❒ Überlauf ❒ kein Überlauf

info1add.tex Nitzsche (OSZ Handel I), 29. September 2002

A.3 Arbeitsblatt 1 (Lösung)

Rechnerinterne Addition

Beispiel ($3 + 15$):

	0	0	1	1	($3_{10} = 2 + 1$)
	1	1	1	1	($15_{10} = 8 + 4 + 2 + 1$)
1	1	1	1	0	(Überträge)
	0	0	1	0	Summe ($2_{10} = 2$) **Überlauf!**

Tritt bei der Addition vorzeichenloser Zahlen ein Übertrag *aus* dem *höchstwertigen Bit* (dem am weitesten links stehenden Bit) auf, so kann das Ergebnis mit den zur Verfügung stehenden Bits nicht mehr korrekt dargestellt werden, es ist ein *Überlauf* aufgetreten.

höchstwertiges Bit

Überlauf

Arbeitsauftrag: Führen Sie die folgenden Additionen durch und entscheiden Sie, ob ein Überlauf statt gefunden hat oder ob nicht.

(1) $4 + 7$:

	0	1	0	0	(4_{10})
	0	1	1	1	(7_{10})
0	1	0	0	0	(Überträge)
	1	0	1	1	Summe (11_{10})

❒ Überlauf ☒ kein Überlauf

(2) $1 + 15$:

	0	0	0	1	(1_{10})
	1	1	1	1	(15_{10})
1	1	1	1	0	(Überträge)
	0	0	0	0	Summe (0_{10})

☒ Überlauf ❒ kein Überlauf

(3) $3 + 5$:

	0	0	1	1	(3_{10})
	0	1	0	1	(5_{10})
0	1	1	1	0	(Überträge)
	1	0	0	0	Summe (8_{10})

❒ Überlauf ☒ kein Überlauf

(4) $10 + 9$:

	1	0	1	0	(10_{10})
	1	0	0	1	(9_{10})
1	0	0	0	0	(Überträge)
	0	0	1	1	Summe (3_{10})

☒ Überlauf ❒ kein Überlauf

(5) $11 + 7$:

	1	0	1	1	(11_{10})
	0	1	1	1	(7_{10})
1	1	1	1	0	(Überträge)
	0	0	1	0	Summe (2_{10})

☒ Überlauf ❒ kein Überlauf

info1add.tex Nitzsche (OSZ Handel I), 29. September 2002