Bit

Binary	zwei Zustände	0-1 (Zahlensystem mit der Basis 2, Dualzahlen) aus-an | false-true; nenne weitere Beispiele
Digit	Ziffer	mit Ziffern gesteuert, digital; nicht analog
Bit	Binary Digit	jede Ziffer kann nur zwei Zustände haben: 0 oder 1
Byte	8 Bit	**1 Byte hat immer 8 Bit** = 8 Ziffern 0-1, 256 Codierungen möglich kleinste Zahl: 00000000 = 0 (dezimal) größte Zahl: 11111111 = 255 (dezimal)
Dezimal	10 Zustände	0-1-2-3-4-5-6-7-8-9 (Zahlensystem mit der Basis 10) lateinisch decimus (zehnter)
Hexadezimal	16 Zustände	0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F (Zahlensystem mit der Basis 16) zur besseren Lesbarkeit von Bytes werden je 4 Bit als Hex-Digit geschrieben
	Hex-Digit	0000=0 ... 1001=9 1010=A 1011=B 1100=C 1101=D 1110=E 1111=F

Binärzahlen zählen und addieren

0+0=0	0+1=1	1+0=1	1+1=10 (Übertrag)

Übertrag	0000 + 0001	0100 + 0001	1000 + 0001	1100 + 0001	0010 + 0010	0011 + 0011	
Übertrag	= 0001 + 0001 1	= 0101 + 0001 1	= 1001 + 0001 1	= 1101 + 0001 1	+ 0011	+ 0110	
Übertrag	= 0010 + 0001	= 0110 + 0001	= 1010 + 0001	= 1110 + 0001	+ 0100	+ 0001	
Übertrag	= 0011 + 0001 11	= 0111 + 0001 111	= 1011 + 0001 111	= 1111 + 0001 1111	+ 0000	+ 0010	
	= 0100	= 1000	= 1100	= 10000			

Binär in Dezimal umrechnen (2 Byte / 16 Bit)

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
											2*2*2*2	2*2*2	2*2	2	1	
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
																16
																100
																10000

Dezimal in Binär umrechnen (Division durch 2 mit Rest)

Rechne die Dezimal-Zahlen x=16, x=100 und x=10000 in Binärzahlen um mit folgendem Programm:

- teile x durch 2 mit Rest
- schreibe **den Rest** oben in die Tabelle von rechts 2^0 beginnend nach links (Array)
- setze x auf **das Ergebnis** der Division x/2 (ohne Kommastellen)
- wiederhole bis x=0

Rechne mit der Tabelle nach, ob das Ergebnis stimmt. Addiere dazu die Dezimalzahlen.

Speicher können nur Bytes speichern, 1 Byte hat immer 8 Bit

Speicher: RAM, ROM, PROM, EPROM, EEPROM, Lochstreifen, Magnetband, Diskette, ..., Flash
auch Netzwerke übertragen nur Bytes ... von einem Speicher in einen anderen Speicher

Im Speicher kann jedes Daten Byte über eine Adresse gefunden werden, Adressen sind auch (mehrere) Bytes.

Adresse 16 Bit		Daten 8 Bit									Maschinencode
	HEX	HEX	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	8 Bit Prozessor U 880
RAM	2000	C3	1	1	0	0	0	0	1	1	JMP nn springe zur Adresse
	2001	EA	1	1	1	0	1	0	1	0	LOW Byte 2^7 .. 2^0 der Adresse
	2002	04	0	0	0	0	0	1	0	0	HIGH Byte 2^{15} .. 2^8 der Adresse
EPROM	04EA										Spielen der Anfangsmusik LC-80

Speicher-Chip Größen begannen bei 2KByte und wurden immer verdoppelt oder vervierfacht.
Bei Festplatten-Speichern gibt es wegen der Mechanik keine bestimmten Kapazitäten.
1 Kilobyte = 1024 Byte=2^{10} (nicht 1000=10^3); 1 Megabyte = 2^{20} Byte.
Wie viele Bytes sind 64 KByte? Und wie viele Bits sind das? 2 Antworten: _____

Prozessoren verarbeiten auch nur Bytes, allerdings mehrere gleichzeitig

Der erste Mikroprozessor konnte 8 Bit Daten parallel verarbeiten und 16 Bit Speicher (0000-FFFF) adressieren. Er hatte also 8 "Drähte" für den Datenbus und 16 "Drähte" für den Adressbus.
Der Datenbus wurde dann immer verdoppelt auf 16 Bit, 32 Bit, 64 Bit. Dafür musste die doppelte Menge Elektronik auf den Chip passen. Das war möglich weil z.B. Transistoren immer kleiner integriert werden konnten. Auch die Taktfrequenz wurde erhöht, damit die Rechenoperationen schneller gehen.

Daten speichern in Bytes

Zahlen werden in 1 Byte, 2 Byte, 4 Byte oder 8 Byte gespeichert.

	ohne Vorzeichen		mit Vorzeichen	
	kleinster Wert	größter Wert	kleinster Wert	größter Wert
8 Bit	0	255	-128	+127
16 Bit	0	65.535	-32.768	+32.767
32 Bit	0	4.294.967.295	-2.147.483.648	+2.147.483.647
64 Bit	0	18.446.744.073.709.551.615		
Festkommazahlen, Gleitkommazahlen werden auch in 8, 16, 32, 64 Bit gespeichert. Datum und Zeit werden mit Software in diese Zahlen umgerechnet.				

Text: jedes Zeichen wird in 1 Byte, 2 Byte, 3 Byte oder 4 Byte gespeichert

128 ASCII Zeichen belegen 7 Bit und werden in 1 Byte gespeichert. Bei ASCII Zeichen ist das Bit 2^7=0.
→ ASCII Tabelle (American Standard Code for Information Interchange)
Alle anderen Zeichen werden in 2, 3 oder 4 Byte gespeichert. Bei diesen Bytes ist das Bit 2^7=1.

Beispiele für UTF-8 Zeichen: ß € 😀

ß 00011 011111		
2 Byte	HEX	DEZ
1100 0011	C 3	195
1001 1111	9 F	159

€ 0010 000010 101100		
3 Byte	HEX	DEZ
1110 0010	E 2	226
1000 0010	8 2	130
1010 1100	A C	172

😀 000 011111 011000 000000		
4 Byte	HEX	DEZ
1111 0000	F 0	240
1001 1111	9 F	159
1001 1000	9 8	152
1000 0000	8 0	128

logische Werte: false oder true können in 1 Bit gespeichert werden (boolean)

Variablen in Programmiersprachen

Variablen Namen dürfen nur Buchstaben a-z, Ziffern 0-9 und den Unterstrich enthalten. Keine Leerzeichen!

Numerische Variablen bekommen immer einen bestimmten Daten-Typ, der eine feste Anzahl Bytes (im Speicher) reserviert. Die größte Zahl, die in einer Variable gespeichert werden kann, wird durch die reservierten Bytes (1,2,4,8) begrenzt (Tabelle oben). Wenn auch negative Zahlen gespeichert werden sollen, wird das höchste Bit als Minus gewertet und der Zahlenbereich halbiert in negative und positive Zahlen. Für Integer wird standardmäßig 32 Bit mit Vorzeichen verwendet. Es gibt auch Datentypen für Kommazahlen, Datum und Zeit und viele andere. Nur unendlich ist nicht möglich, weil die Hardware Grenzen setzt. Zum Anzeigen werden Zahlen in Text (toString) umgewandelt und dabei formatiert.

String bedeutet Zeichenkette, also Text. **String Variablen** speichern jeden Buchstabe und alle anderen Zeichen als Zeichencode, also als 8 Bit, 16 Bit oder 32 Bit Zahl. Beim Lesen aus dem Speicher und beim Speichern müssen diese Zeichencodes immer umgerechnet werden, weil der Speicher nur Bytes (je 8 Bit) kennt. Um Text anzuzeigen muss das Display oder der Drucker die Zeichen als Pixel kennen oder das Programm muss sie vorher in Images (Bilder) umwandeln. Kleine Computer (z.B. Calliope) sind begrenzt auf 8-Bit-Zeichencodes. Text-Encoding ist so kompliziert, dass immer wieder Fehler auftreten.

Boolean Variablen sind logische Variablen, die nur zwei Werte true oder false (wahr oder falsch) speichern können. Diese können bei der Programmierung direkt in Bedingungen verwendet werden.

Objekt Variablen können alles speichern, was ein Programmierer sich ausgedacht hat.

Arrays: Listen von Variablen gleichen Typs

Array Objekte speichern beliebig viele (0 .. n) Werte in einer Liste. Alle Elemente in einem Array müssen den gleichen Datentyp haben: entweder nur Zahlen, oder nur Texte, oder nur logische true/false Werte. Es gibt auch Arrays von Objekten und Arrays von Arrays... Eigentlich ist ein String schon ein Array von einzelnen Zeichencodes. Es können aber auch viele Strings (unterschiedlicher Länge) in einem Array gespeichert werden. In Arrays kann jedes Element über den Index gefunden werden: arrayVariable[index].

RGB Farbcode speichern in Bytes

RGB steht für die Farben rot, grün und blau. Jedes Farb-Display kann nur genau diese 3 Farben anzeigen. Alle Farben werden durch unterschiedliche Helligkeiten der 3 Farben gemischt. Um die Helligkeit zu speichern hat jede Farbe genau 1 Byte. Das ergibt 16777216 Farben pro Pixel.
Wie ist der Rechenweg?

Weil es keine 3-Byte Datentypen gibt, kommt für den Farbcode nur eine 32 Bit Zahl (4 Byte) in Betracht. Das vierte Byte gibt an, wie durchsichtig oder deckend die Farbe des Pixels sein soll. Das ist aber nur sinnvoll, wenn es einen Hintergrund gibt, der durch ein Vordergrund Bild hindurch scheinen soll.
In der Tabelle ist der Code für die maximale Helligkeit einer Farbe angegeben. Nur mit Hexadezimalzahlen ist der RGB Code auch für Menschen verständlich. Schreibe in die zweite Tabelle die fehlenden Farben!

Transparenz								rot								grün								blau							
2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
								FF0000 16711680								00FF00 65280								0000FF 255							

000000	schwarz	FF0000	
0000FF		FF00FF	
00FF00		FFFF00	
00FFFF		FFFFFF	weiß

Drucker benutzen die Komplementärfarben CMY (cyan, magenta, yellow). Der Unterschied ist: Wenn alle 3 Farben CMY gedruckt werden, wird das Papier schwarz. Wenn alle 3 Farben RGB angezeigt werden, wird das Display weiß. Es kommt also darauf an, ob es ohne Farbe hell oder dunkel war.

Wenn der Bildschirm 1920x1080 Pixel hat, wie viele Bytes müssen übertragen werden? Wie viele Pixel hat die digitale Tafel? Und jetzt stell dir vor, wie schnell ein Bild sich ändern kann.

Übung

1. Schreibe Deinen Name oder einen Text in die Tabelle in Spalte ASC.
2. Schreibe den Zeichencode als Hexadezimalzahl in Spalte HEX.
3. Schreibe die 8 Bit aus der Hexadezimalzahl in Spalte Byte.

HEX	ASC	HEX	ASC	HEX	ASC
2 0		4 0	@	6 0	`
2 1	!	4 1	A	6 1	a
2 2	"	4 2	B	6 2	b
2 3	#	4 3	C	6 3	c
2 4	$	4 4	D	6 4	d
2 5	%	4 5	E	6 5	e
2 6	&	4 6	F	6 6	f
2 7	'	4 7	G	6 7	g
2 8	(	4 8	H	6 8	h
2 9	)	4 9	I	6 9	i
2 A	*	4 A	J	6 A	j
2 B	+	4 B	K	6 B	k
2 C	,	4 C	L	6 C	l
2 D	-	4 D	M	6 D	m
2 E	.	4 E	N	6 E	n
2 F	/	4 F	O	6 F	o
3 0	0	5 0	P	7 0	p
3 1	1	5 1	Q	7 1	q
3 2	2	5 2	R	7 2	r
3 3	3	5 3	S	7 3	s
3 4	4	5 4	T	7 4	t
3 5	5	5 5	U	7 5	u
3 6	6	5 6	V	7 6	v
3 7	7	5 7	W	7 7	w
3 8	8	5 8	X	7 8	x
3 9	9	5 9	Y	7 9	y
3 A	:	5 A	Z	7 A	z
3 B	;	5 B	[	7 B	{
3 C	<	5 C	\	7 C	|
3 D	=	5 D	]	7 D	}
3 E	>	5 E	^	7 E	~
3 F	?	5 F	_	7 F	DEL
HEX	ASC	HEX	ASC	HEX	ASC

Byte		HEX	ASC
0101	1010	5 A	Z
Byte		HEX	ASC

Hilfe: eine HEX Ziffer in 4 Bit umrechnen

2^3	2^2	2^1	2^0		
8	**4**	**2**	**1**		
0	0	0	0	0	
1	0	0	1	9	
1	0	1	0	10	A
				11	B
1	1	0	0	12	C
				13	D
				14	E
1	1	1	1	15	F