Velkommen til MM2
Digitale HW/SW systemer

Hovedemner:

Tal systemer

Konvertering

Kode syntaks

AALBORG UNIVERSITET

DAGENS EMNER/LÆRINGSMÅL

EMNER:

- Tal systemer
- Konvertering mellem talsystemer
- Genopfriskning af C kode syntaks

LÆRINGSMÅL:

At få en dybere forståelse for talsystemer.

At skrive et MATLAB program, der kan konvertere mellem to vilkårlige talsystemer.

At kunne læse og forstå den C kode I finder og dermed være i stand til at skrive jeres egen, herunder at kunne skelne mellem de gængse C operatorer.

Hvor er vi kommet til?

Problems

Algorithms

Language (Program) *Programmable*

Machine (ISA) Architecture ***Computer Specific***

Micro-architecture *Manufacturer Specific*

Circuits

Devices

Hvad er et talsystem

Et talsystem er en systematisk måde at repræsentere tal med **symbolske tegn** på. Et talsystem bruger en **basisværdi** til en passende gruppering af tallene i et kompakt format. Det mest almindelige talsystem er decimalsystemet, som har basisværdien 10 og det symbolske tegnsæt 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9.

$$\textbf{decimal value} = \sum_{k=0}^{number\ of\ digits\ -1} (nth\ digit\ value * base^{n})$$

GENERISK FORMEL FRA ET VILKÅRLIGT TALSYSTEM TIL DECIMAL SYSTEMET

Talsystemer

Positionelle talnotationssystemer

Grundtal	Navn
1	Det unære talsystem
2	Det binære talsystem
3	Det ternære talsystem
4	Det kvaternære talsystem
8	Det oktale talsystem
10	Det decimale talsystem (Det arabiske talsystem)
16	Det hexadecimale talsystem
20	Det vigesimale talsystem (Maya indianernes talsystem)
60	Det sexagesimale talsystem (Det Babyloniske talsystem)

Ikke-positionelle talnotationssystemer

Romertallene er ikke-positionelle - eksempelvis betyder "V" 5 uanset placering.

Romertal fra 1 til 109

	I	II	III	IV	V	VI	VII	VIII	IX
X	XI	XII	XIII	XIV	XV	XVI	XVII	XVIII	XIX
XX	XXI	XXII	XXIII	XXIV	XXV	XXVI	XXVII	XXVIII	XXIX
XXX	XXXI	XXXII	XXXIII	XXXIV	XXXV	XXXVI	XXXII	XXXVIII	XXXIX
XL	XLI	XLII	XLIII	XLIV	VL	XLVI	XLVII	XLVIII	IL
L	LI	LII	LIII	LIV	LV	LVI	LVII	LVIII	LIX
LX	LXI	LXII	LXIII	LXIV	LXV	LXVI	LXVII	LXVIII	LXIX
LXX	LXXI	LXXII	LXXIII	LXXIV	LXXV	LXXVI	LXXVII	LXXVIII	LXXIX
LXXX	LXXXI	LXXXII	LXXXIII	LXXXIV	LXXXV	LXXXVI	LXXXVII	LXXXVIII	LXXXIX
XC	XCI	XCII	XCIII	XCIV	VC	XCVI	XCVII	XCVIII	IC
C	CI	CII	CIII	CIV	CV	CVI	CVII	CVIII	CIX

Bemærk at tallet nul ikke findes

4321 = MMMMCCCXXI
Simpel sammentælling

Opbygningen af Romertal

1. A smaller number in front of a larger number means subtraction, all else means addition.
2. Do not put more than one smaller number in front of a larger number to subtract.
3. You must separate ones, tens, hundreds, and thousands as separate items.

I	The numeral 1
V	The numeral 5
X	The numeral 10
L	The numeral 50
C	The numeral 100
D	The numeral 500
M	The numeral 1000

Sometimes you will see a numeral with a line over it. That means to multiply it by 1000. A numeral V with a line over it means 5000.

Det unære talsystem

Det binære talsystem

Bit: 1 digit
Nibble: 4 digits
Byte: 8 digits
Word: The standard memory bus width in your architecture (e.g. 16-bit, 32-bit, 64-bit words).

Basisværdi: 2
Symbolske tegnsæt: 0, 1

Example:

$$\text{decimal value} = \sum_{k=0}^{\text{number of digits}-1} (\text{nth digit value} * \text{base}^{n})$$

$(10101)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$16 + 0 + 4 + 0 + 1 = (21)_{10}$

Negative binære tal

Signed Magnitude: Leftmost bit has sign (0 is +, 1 is -)

One's Complement: To negate a number switch all 0's and 1's (including sign bit) Obsolete now.

Two's Complement: (Has sign bit 0 for + and 1 for -)

Negating a number:
Switch 0's and 1's (as in one's complement)
Add 1 to the result

Example 00000110 (+6)
(-6 in one's complement) 11111001
(-6 in two's complement) 11111010
any carry over from leftmost bit is thrown away

Hvorfor er Two's Complement smart?

Vi vil gerne udføre følgende regnestykke:

$$8 - 2$$

Men vores CPU kan kun udføre addition. Derfor "omskriver" vi regnestykket til følgende:

$$8 + (-2)$$

Hvis vi bruger two's complement til at repræsentere -2 får vi følgende:

$$\begin{array}{r} 00001000 \\ +\ 11111110 \\ \hline 1\,00000110 \end{array}$$

Vi kan altså lave ægte subtraktion med en add-funktion kombineret med two's complement data.

Det ternære talsystem

Basisværdi: 3

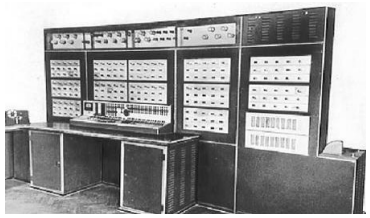
Symbolske tegnsæt:

Set	Name / comments
{0,1,2}	Unbalanced Trinary
{0,1/2,1}	Fractional Unbalanced Trinary
{-1,0,1}	Balanced Trinary
{F,?,T}	Unknown-State Logic
{T,F,T}	Trinary Coded Binary

Cifre i det ternære talsystem

Den negative værdi af en afbalanceret t
antal kan opnås ved simpelthen at udski
hvert + med et - og omvendt.

Trits, Tribbles, and Trytes
(versus bits, nibbles, and bytes)



SETUN computeren bygget af
Nikolay Brusentsov i 1958 ved Moskvas Universitet

decimal	base-3	balanced ternary
−16	−121	−++−
−15	−120	−++0
−14	−112	−+++
−13	−111	−−−
−12	−110	−−0
−11	−102	−−+
−10	−101	−0−
−9	−100	−00
−8	−22	−0+
−7	−21	−+−
−6	−20	−+0
−5	−12	−++
−4	−11	−−
−3	−10	−0
−2	−2	−+
−1	−1	−
0	0	0
1	1	+
2	2	+−
3	10	+0
4	11	++
5	12	+−−
6	20	+−0
7	21	+−+
8	22	+0−
9	100	+00
10	101	+0+
11	102	++−
12	110	++0
13	111	+++
14	112	+−−−
15	120	+−−0
16	121	+−−+

Arabiske talsystem

Basisværdi:
10

Symbolske tegnsæt:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Det Hindu-arabiske talsystem, Arabiske talsystem eller titalssystemet er verdens mest udbredte talsystem.

Det er opfundet af indere og formidlet til Vesten af araberne

Det enestående ved dette system er opfindelsen af tegnet nul, "0", der gør det muligt at nedskrive, at en talklasse (enere, tiere, hundreder...) er tom.

Dog skal det bemærkes at mayaindianerne var før europæerne i anvendelse af nul som ciffer i et positionstalsystem og at Baylonierne var endnu tidligere.

Det hexadecimale talsystem

Basisværdi:
16

Symbolske tegnsæt:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Fra det binære:

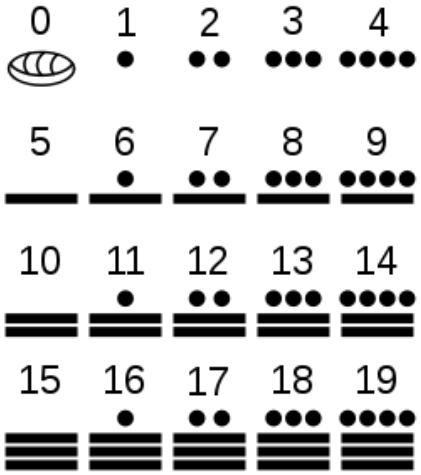
Nibble: $4\ \text{digits}_2 = 1\ \text{digit}_{16}$

Byte: $8\ \text{digits}_2 = 2\ \text{digit}_{16}$

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

Maya indianernes talsystem

Basisværdi:
20
Symbolske tegnsæt:

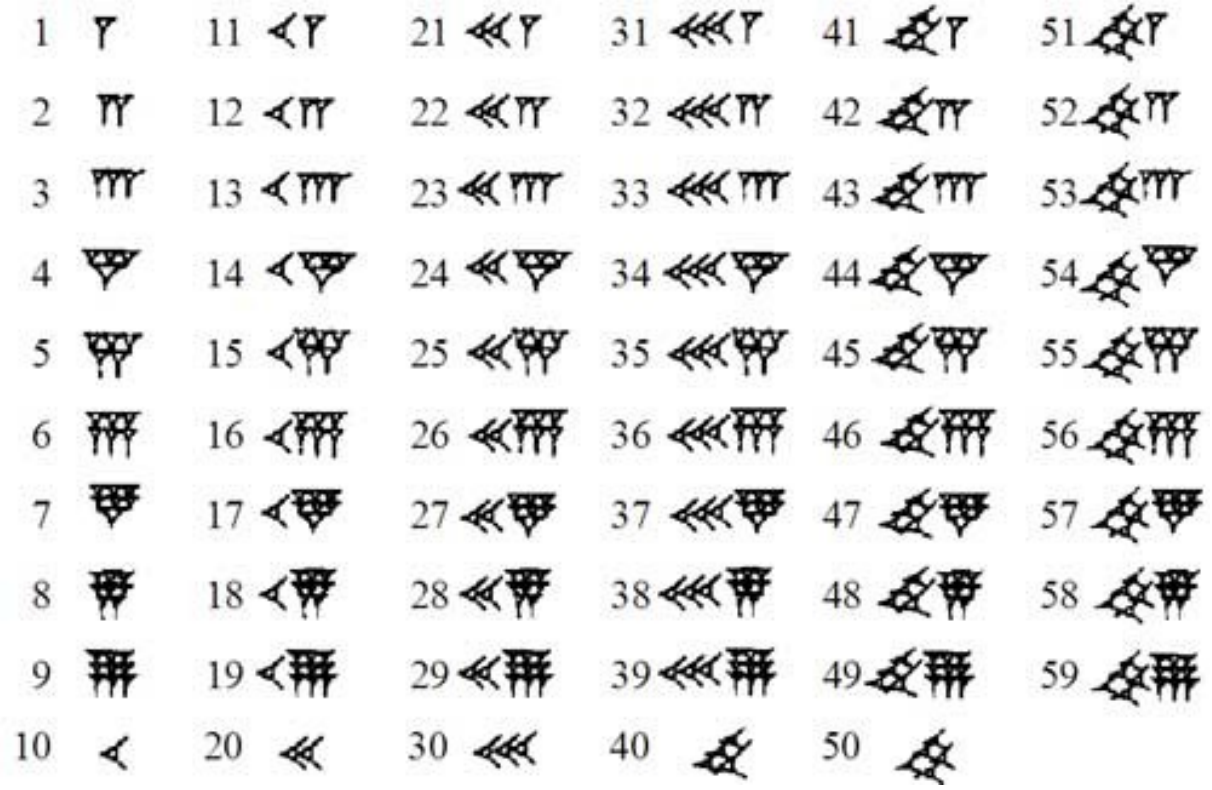


400s		•	••
20s	•	•	•••
1s	•••	••••	—
	33	429	5125

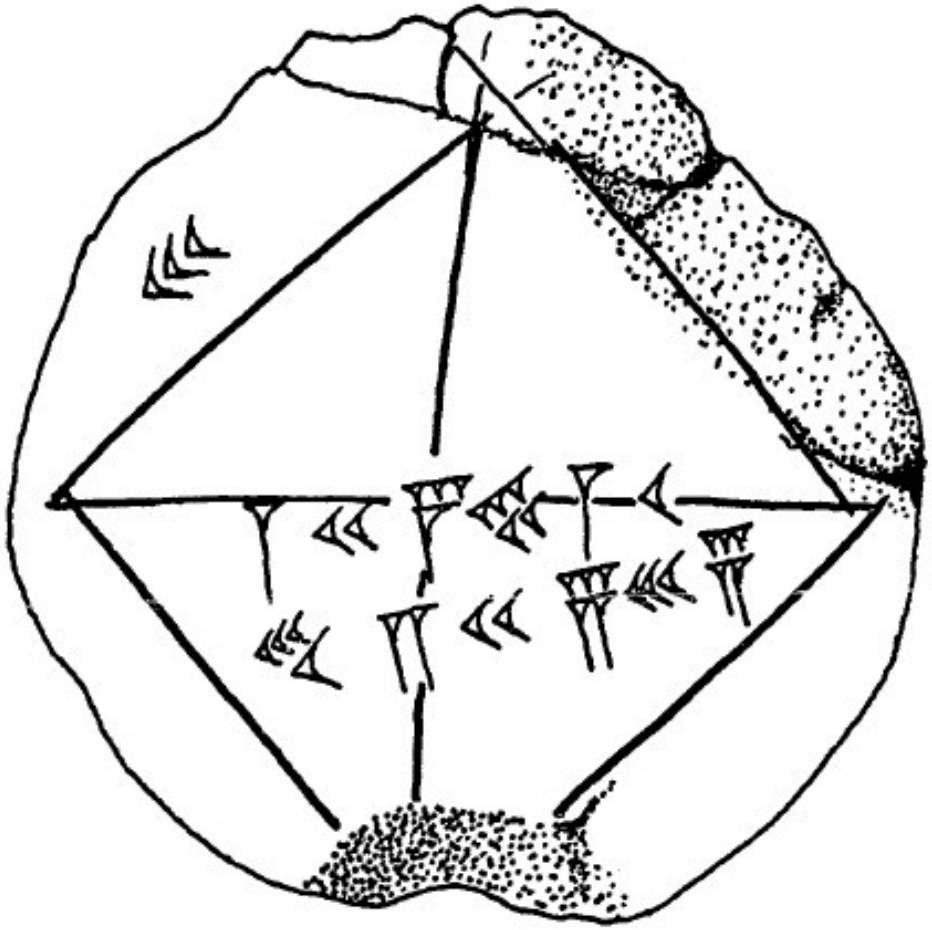
Maya indianerne brugte en kalender bestående af 365 dage.
Maya året bestod af 18 måneder hver på 20 dage.
De sidste 5 dage udgjorde en selvstændig måned fyldt med farer og ulykker

Det Babyloniske talsystem

Rester af det Babyloniske talsystem lever stadig i dag i form af grader, minutter og sekunder i trigonometri og ved måling af tid.



YBC 7289



Talsystemer versus talord

50 - Halvtredsindstyve: dannet af halvtredje, sinde og tyve, altså '2½ gange 20'.

60 - Tresindstyve: dannet af tre, sinde og tyve, altså '3 gange 20'.

70 - Halvfjerdsindstyve: dannet af halvfjerde, sinde og tyve, altså '3½ gange 20'.

80 - Firsindstyve: dannet af fire, sinde og tyve, altså '4 gange 20'.

90 - Halvfemsindstyve: dannet af halvfemte, sinde og tyve, altså '4½ gange 20'.

Konverteringsmetoder

Potensmetoden

FRA ET VILKÅRLIGT TALSYSTEM TIL DECIMAL SYSTEMET

$$TAL = 24AF_{16}$$

$$= 2X16^3 + 4X16^2 + 10X16^1 + 15X16^0$$

$$= 9391_{10}$$

Konverteringsmetoder

Divisionsmetoden

FRA DECIMAL SYSTEMET TIL ET VILKÅRLIGT TALSYSTEM
EKSEMPEL: KONVERTER 521_{10} TIL DET HEXADICIMALE TALSYSTEM

TAL $= 521_{10}$

$= 521/16 = 32 + 9/16$

$= 32/16 = 2 + 0/16$

$= 2/16 = 0 + 2/16$

$= 209_{16}$ CHECK: $2X16^2 + 0X16^1 + 9X16^0 = 521_{10}$

Konverterings-Opgave

Udfyld tabellen:

Basis tal	2	3	8	10	16
	01100111				
		1102			
			765		
				233	
					0xFA

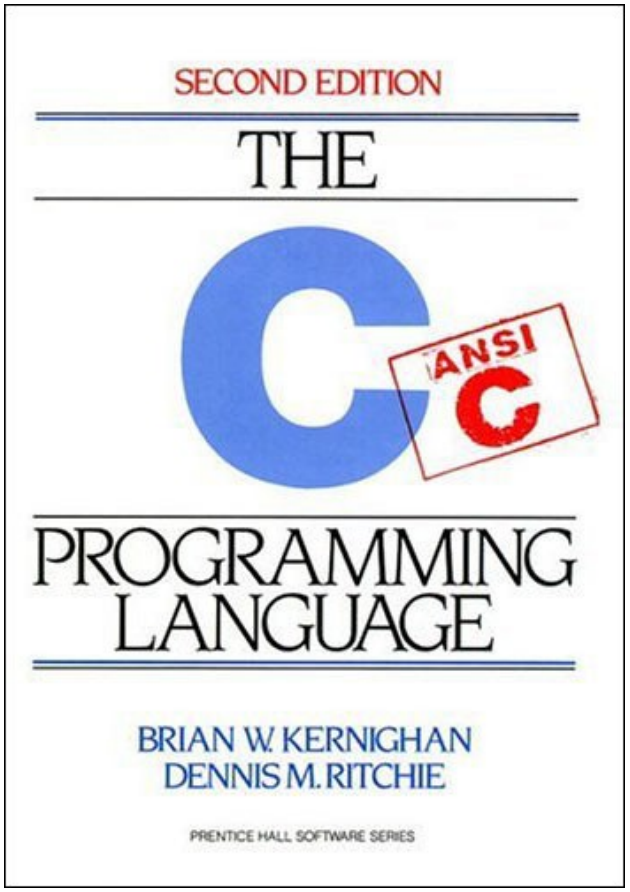
Der skal laves en MATLAB funktion, der konverterer fra én vilkårlig base til en anden. I får (måske) brug for denne funktion til eksamen!

function result = base2base(input_string, old_base, new_base)

Hint: søg evt. på eksisterende MATLAB funktioner til base konverteringer

Genopfriskning af C kode syntaks

PDF udgave findes på moodle under litteratur.



Vær venlig og del jeres viden om online bøger, hvis I har adgang til nogle.

Operators

There are three types of operators

- A **unary** expression consists of either a unary operator prepended to an operand, or the sizeof keyword followed by an expression. The expression can be either the name of a variable or a cast expression. If the expression is a cast expression, it must be enclosed in parentheses.
- A **binary** expression consists of two operands joined by a binary operator.
- A **ternary** expression consists of three operands joined by the conditional-expression operator.

Unary operators take precedence over binary operators

- the unary operation is always performed first.

Operators: Arithmetic

Symbol	Operator	Example
*	multiplication	a*b
/	division	a/b
%	modulo	a%b
+	addition	a+b
-	subtraction	a-b

Relational and Logical

Symbol	Operator	Example
<	less than	a < b
>	greater than	a > b
>=	greater than or equal	a >= b
==	equal to	a==b
!=	not equal	a!=b

Sample	Operator	Example
!	NOT	!(a < b)
&&	AND	a < b && c > d
||	OR	a || d

Bitwise Operators

Operator	Description	Associativity
~	Ones complement	right to left
<<	Left shift	left to right
>>	Right shift	left to right
&	Bitwise AND	left to right
^	Bitwise XOR	left to right
|	Bitwise OR	left to right

Typical Bit Mask Operations

Operation	Code
Set a flag or overlay multiple values	flags | flagbitN
Unset a flag (zero-out a bit or set a bit to zero)	flags & ~flagbitN
Check if a bit is set	(flags & flagbitN) == flagbitN
Invert a pattern of bits	~flags

11 types of Assignment

Symbol	Operator	Example
=	assignment	a = b
+=	addition and assignment	a += b same as a = a + b
-=	subtraction and assignment	a -= b same as a = a - b
*=	multiplication and assignment	a *= b same as a = a * b
/=	division and assignment	a /= b same as a = a / b
%=	modulo and assignment	a %= b same as a = a % b
&=	bitwise AND assignment	a &= b same as a = a & b
|=	bitwise OR and assignment	a |= b same as a = a | b
^=	bitwise XOR and assignment	a ^= b same as a = a ^ b
<<=	shift left and and assignment	a <<= 2 same as a = a << 2
>>=	shift right and assignment	a >>= 4 same as a = a >> 4

Associativity and precedence of operators

Precedence	Operator	Description	Associativity
1	++ -- ( ) [ ] . -> (*type*){*list*}	Suffix/postfix increment and decrement Function call Array subscripting Structure and union member access Structure and union member access through pointer Compound literal(C99)	Left-to-right
2	++ -- + - ! ~ (*type*) * & sizeof _Alignof	Prefix increment and decrement Unary plus and minus Logical NOT and bitwise NOT Type cast Indirection (dereference) Address-of Size-of Alignment requirement(C11)	Right-to-left
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	

Associativity and precedence of operators

Precedence	Operator	Description	Associativity
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	Left-to-right
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13[note 1]	?:	Ternary conditional[note 2]	Right-to-Left
14	= += -= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

1. ↑ Fictional precedence level, see Notes below
2. ↑ The expression in the middle of the conditional operator (between **?** and **:**) is parsed as if parenthesized: its precedence relative to ?: is ignored.

Kilde: http://en.cppreference.com/w/c/language/operator_precedence

A general form for a C and C++ program is as follows:

inclusion of header files
defined constants
declarations of global variables

```
Function-Return-Type main(arguments from command line)
{
    declaration of local variables used by main function;
    body of the main function;
}

Function-Return-Type next_function(argument list)
{
    declaration of local variables for next_function;
    body of the next_function;
}
```

Keywords

The ANSI C (a.k.a. C89) standard is comprised of 32 keywords, or reserved words. Many compilers have extended and added to these for different computing platforms. The keywords are as follows:

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Some keywords where added later

C99 adds five more keywords:	
_Bool	inline
_Complex	long long
_Imaginary	restrict
C11 adds seven more keywords:	
_Alignas	_Noreturn
_Alignof	_Static_assert
_Atomic	_Thread_local

The ANSI C Header Files

assert.h	Diagnostics
ctype.h	Character Handling and Conversion
float.h	Floating Point Support
limits.h	Maximum and Minimum Limits
locale.h	International Support
math.h	Mathematics
setjmp.h	Jumps
signal.h	Signal Handling
stdarg.h	Variable Number of Arguments
stddef.h	Standard Definitions
stdio.h	Input and Output Support
stdlib.h	General Definitions and Utilities
string.h	String Handling
time.h	Time and Date Support

The Alphabet of C

The Standard requires that an alphabet of 96 symbols is available for C as follows:

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

! " # % & ' ( ) * + , - . /

: ; < = > ? [\ ] ^ _ { | } ~

space, horizontal and vertical tab

form feed, newline

ASCII Table

1-31 Control Codes
32-64 Symbols
65-90 Uppercase
91-96 More Symbols
97-122 Lowercase
123-126 More Symbols
127 Delete
128-255 International Symbols

The American Standard Code for Information Interchange (ASCII)

ASCII Table

Regular ASCII Chart (character codes 0 – 127)

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h		048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	‘	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	’	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

Extended ASCII Chart (character codes 128 – 255) Latin1/CP1252

128d	80h	€		144d	90h			160d	A0h		176d	B0h	°	192d	C0h	À	208d	D0h	Ð	224d	E0h	à	240d	F0h	ð
129d	81h			145d	91h	‘		161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h	‚		146d	92h	’		162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h	*f*		147d	93h	“		163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h	„		148d	94h	”		164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h	...		149d	95h	•		165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h	†		150d	96h	-		166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h	‡		151d	97h	--		167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h	ˆ		152d	98h	˜		168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h	‰		153d	99h	™		169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah	Š		154d	9Ah	š		170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh	<		155d	9Bh	>		171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch	Œ		156d	9Ch	œ		172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh			157d	9Dh			173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh	Ž		158d	9Eh	ž		174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh			159d	9Fh	Ÿ		175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

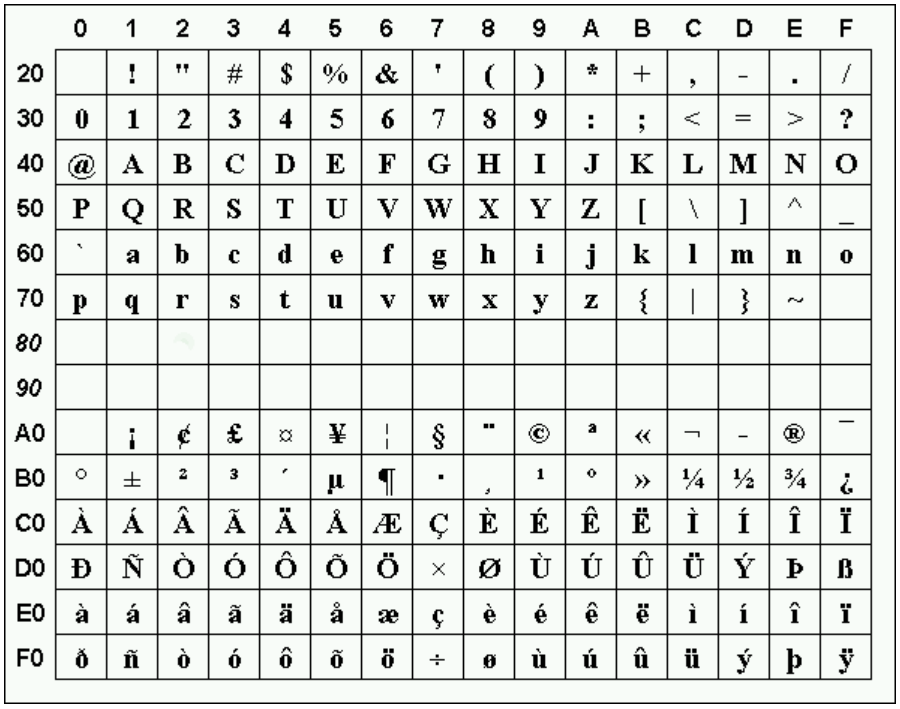
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

ISO-8859-1 international character set

ISO-8859-1 covers the following languages: Afrikaans, Basque, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80																
90																
A0		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

It is the standard character set used on the Internet.

TutorialsPoint demo

< Coding Ground >

http://www.tutorialspoint.com/codingground.htm

Opgave 1: find fejlen

```c
#include <stdio.h>

void main(void)
{
  int bil = 2;
  int cykel = 1;

  if (cykel = bil)
  {
    printf("if blev valgt\n");
  }
  else
  {
    printf("else blev valgt\n");
  }

  printf("\n");
  printf("cykel = %d\n",cykel);
  printf("bil   = %d\n",bil);
  return;
}
```

Filen ”test1.c” kan hentes i kursusmappen for MM2.

Hvilken printf linje eksekveres? Hvis resultatet er overraskende forklar hvorfor.

Hint: Hvad er værdien af bil og cykel for hver linje kode?

Opgave 2: postfix versus prefix

```c
#include <stdio.h>

void main(void)
{
  int n = 5;
  int x = 0;

  x = ++n;

  if (x == 5)
  {
    printf("if blev valgt\n");
  }
  else
  {
    printf("else blev valgt\n");
  }

  printf("\n");
  printf("x = %d\n",x);
  printf("n = %d\n",n);
  return;
}
```

Filen ”test2.c” kan hentes i kursusmappen for MM2.

Hvilken printf linje eksekveres? Forklar hvorfor.

Hvad er værdien af ”x” og ”n” efter linje 8?

Opgave 3: logical versus bitwise

```c
#include <stdio.h>

void main(void)
{
  int n = 5;
  int x = 0;

  x = n++<<2;
  if ((x == 20) &  n )
  {
    printf("if blev valgt\n");
  }
  else
  {
    printf("else blev valgt\n");
  }

  printf("\n");
  printf("x = %d\n",x);
  printf("n = %d\n",n);

  return;
}
```

Filen ”test3.c” kan hentes i kursusmappen for MM2.

Hvilken printf linje eksekveres? Forklar hvorfor.

Opgave 4: Precedence versus Associativity

```c
#include <stdio.h>

void main(void)
{
  int x = 1;
  int y = 2;
  int z = 3;

  y = x - y - z - 10;
  x = y += z -= 4;

  if (x)
  {
    printf("if blev valgt\n");
  }
  else
  {
    printf("else blev valgt\n");
  }

  printf("\n");
  printf("x = %d\n",x);
  printf("y = %d\n",y);
  printf("z = %d\n",z);
  return;
}
```

Filen ”test4.c” kan hentes
kursusmappen for MM2.

Hvad er værdien af ”y” efter linje 9?
Forklar hvorfor.

Hvad er værdien af ”x”, ”y” og ”z”
efter linje 10?
Forklar hvorfor.

Hvilken printf linje eksekveres?
Forklar hvorfor.

Hvornår går man ind i en ”if” sætning

Opgave 5: Array indeksering og adresser

```c
#include <stdio.h>

void main(void)
{
  int a = 1;
  int b[3] = {1,2,3};
  int c = 5;
  int d = 6;

  b[3] = 4;

  if (a == 1)
  {
    printf("if blev valgt\n\n");
  }
  else
  {
    printf("else blev valgt\n\n");
  }

  printf("a = %d\n",a);
  printf("b[0] = %d\n",b[0]);
  printf("b[1] = %d\n",b[1]);
  printf("b[2] = %d\n",b[2]);
  printf("b[3] = %d\n",b[3]);
  printf("c    = %d\n",c);
  printf("d    = %d\n",d);

  printf("\n");

  printf("Address of a: %#012x\n",&a);
  printf("Address of b: %#012x\n",&b);
  printf("Address of c: %#012x\n",&c);
  printf("Address of d: %#012x\n",&d);

  return;
}
```

Filen ”test5.c” kan hentes i kursusmappen for MM2.

Hvilken printf linje eksekveres? Forklar hvorfor.

Hvad er værdierne af ”a” , ”b” , ”c” og ”d”?

Forklar hvorfor.

Opgave 6: Ternary conditional

```c
#include <stdio.h>

void main(void)
{
    int a , b;

    a = 10;

    if (a == 1)
    {
        b = 20;
    }
    else
    {
        b = 30;
    }

    printf("a = %d\n",a);
    printf("b = %d\n",b);

    return;
}
```

Filen ”test6.c” kan hentes i kursusmappen for MM2.

Omskriv koden så linje 9 til og med linje 16 skrives som én linje ved hjælp af ternary conditional operatoren.

Forklar hvorfor man kunne finde på at gøre dette.

Ekstra opgave:

Hvad er sammenhængen mellem nedenstående:

```
char s[5] = "ABCD";
char x[4] = {65,  66,  67,  68};
int y = 1145258561;
float z = 781.035217;
```

Filen ”ekstra_opgave.c” kan hentes i kursusmappen for MM2.

Ekstra opgave: lær brugen af pointere

```c
#include <stdio.h>

void main(void)
{
    int i;
    char *address;

    char s[4] = "ABCD";
    char x[4] = {65,   66,   67,   68};
    int y = 1145258561;
    float z = 781.035217;


    address = (char *) &s;

    printf("Memory content of variable:\n");
    for(i=0;i<sizeof(s);i++)
    printf("Address 0x%.16lx = %d\n",address,*(address++));

    return;
}
```

Næste gang …

- Gates and Boolean Algebra (kap. 3.1)
- Basic Digital Logic Circuits (kap. 3.2)
- Memory (kap. 3.3)
- CPU (kap. 3.4)
- Busser (kap. 3.4)

Litteratur:

Kap. 3 til og med kap. 3.3 i “Structured computer organization”

Dette er et af de vigtigste kapitler for at forstå computerens opbygning.