Prof. Dr. rer. nat. habil. Martin O. Steinhauser

Frankfurt University of Applied Sciences, Germany

Faculty of Computer Science and Engineering

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

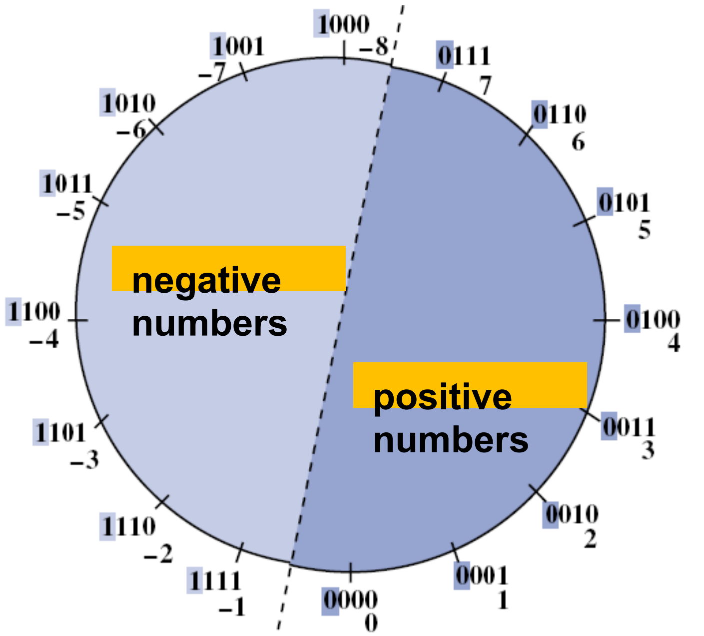
Short Lecture Course:

Introduction to Computational Science with Applications in Molecular Dynamics

Session 2: Bits and Bytes in Computer Memory

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com



Overview of this short course

Topics Covered (subject to change)

1st Session: Lec. 1-2		Introduction & Bits and Bytes
2nd Session: Lec 3	(2x)	Bits and Bytes continued
3rd Session: Lec 4-6		Molecular Dynamics
4th Session: Lec 7-8		MD continued / Algorithms
5th Session: Lec 9	(2x)	Algorithms/ Problem of Sorting
6th Session: Lec 10-11		Asymptotic Analysis of Algorithms
7th Session: Lec 12-13		Monte Carlo/Random Numbers

Session 2: Bit Representation in Computer Memory

- Bit Representation of Numbers and Data Types in Memory

- **Handout 2:** Basics of the UNIX Programing Environment
- **Handout 3:** Bits and Bytes in Memory

To download lecture material, please go to Github:

https://github.com/Kosmokrat/JapanLecture2024

Session 2: Lecture 3

1	Quick Review: Programming Languages and the Compilation Process
2	Basic Data Types
3	Bit Representation of Numbers and Data Types in Computer Memory

Overview of Lecture 3

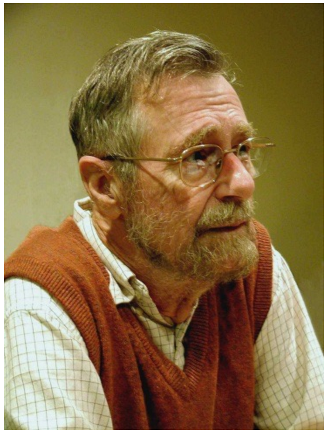
1	Quick Review: Programming Languages and the Compilation Process
2	Basic Data Types
3	Bit Representation of Numbers and Data Types in Computer Memory

Hardware is not Everything!

Hardware without Software is
Noware

Programming Languages

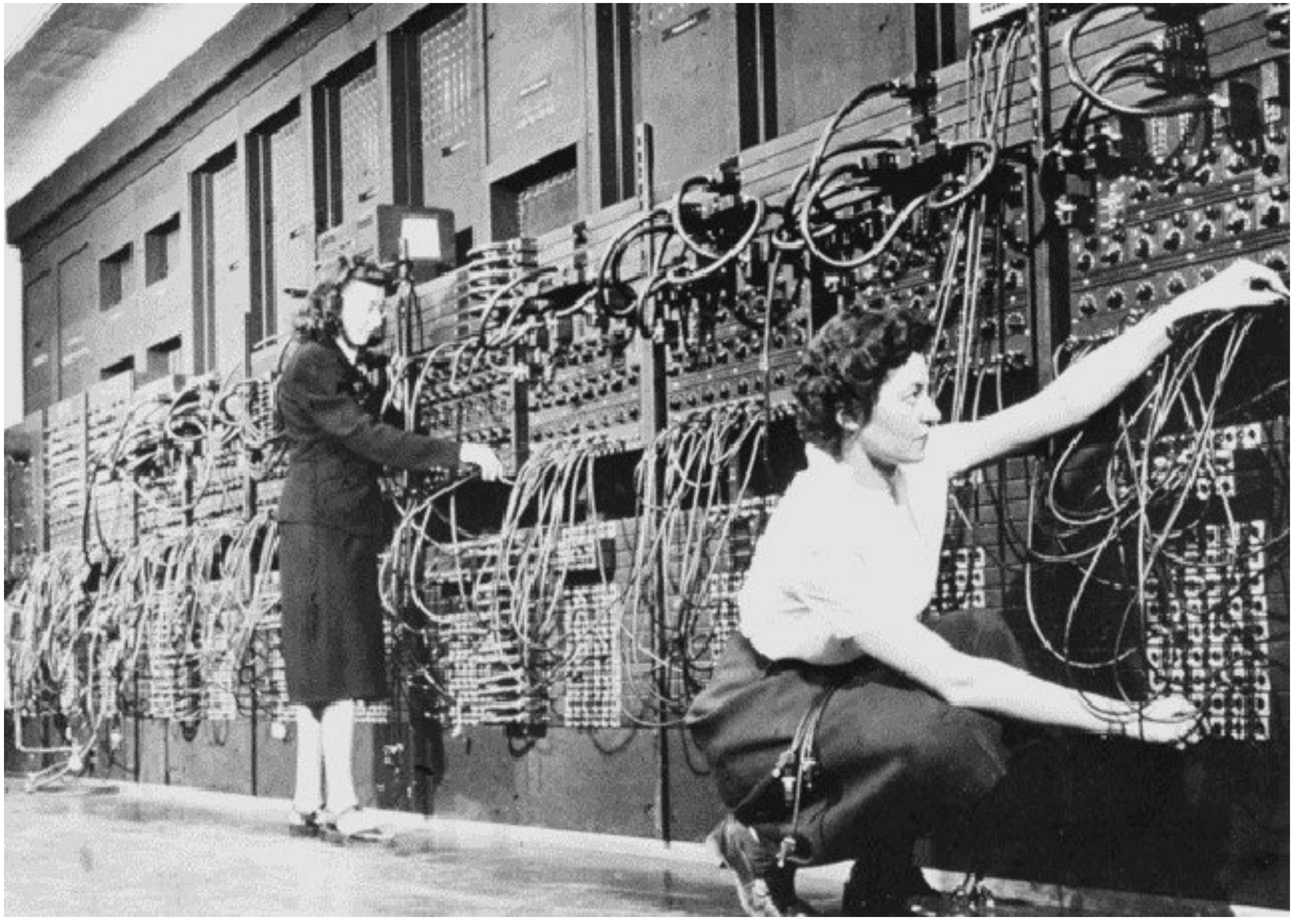
„To put it quite bluntly; as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem; and now we have gigantic computers, programming has become an equally gigantic problem."



E. Dijkstra, 1972 Turing award lecture

Programming Languages

- **1940s**: PROGRAMMING = **MACHINE LANGUAGE** (hardware cable connections)
  - Atomic instructions (zeros and ones)



Programming Languages

- **1940s**: PROGRAMMING = **MACHINE LANGUAGE** (hardware cable connections)
  - Atomic instructions (zeros and ones)

- **1950s: Assembly** (Very first programming language)
  - Displays machine code using (more human-friendly) hexadecimal numbers
  - Assembler is *specific* for every processor!
  - Every processor has a different set of elementary instructions

Address	Machine Language				Assembly Language		
0000 0000	0000	0000	0000	0000	TOTAL	.BLOCK	1
0000 0001	0000	0000	0000	0010	ABC	.WORD	2
0000 0010	0000	0000	0000	0011	XYZ	.WORD	3
0000 0011	0001	1101	0000	0001		LOAD	REGD, ABC
0000 0100	0001	1110	0000	0010		LOAD	REGE, XYZ
0000 0101	0101	1111	1101	1110		ADD	REGF, REGD, REGE
0000 0110	0010	1111	0000	0000		STORE	REGF, TOTAL
0000 0111	1111	0000	0000	0000		HALT	

Programming Languages

- **1940s**: PROGRAMMING = **MACHINE LANGUAGE** (hardware cable connections)
  - Atomic instructions (zeros and ones)

- **1950s: Assembly** (Very first programming language)
  - Displays machine code using (more human-friendly) hexadecimal numbers
  - Assembler is *specific* for every processor!
  - Every processor has a different set of elementary instructions

- **1960s / 70s: Assembly became a problem**
  - Computers could handle larger more complex problems
  - Needed to get abstraction and portability without loosing performance

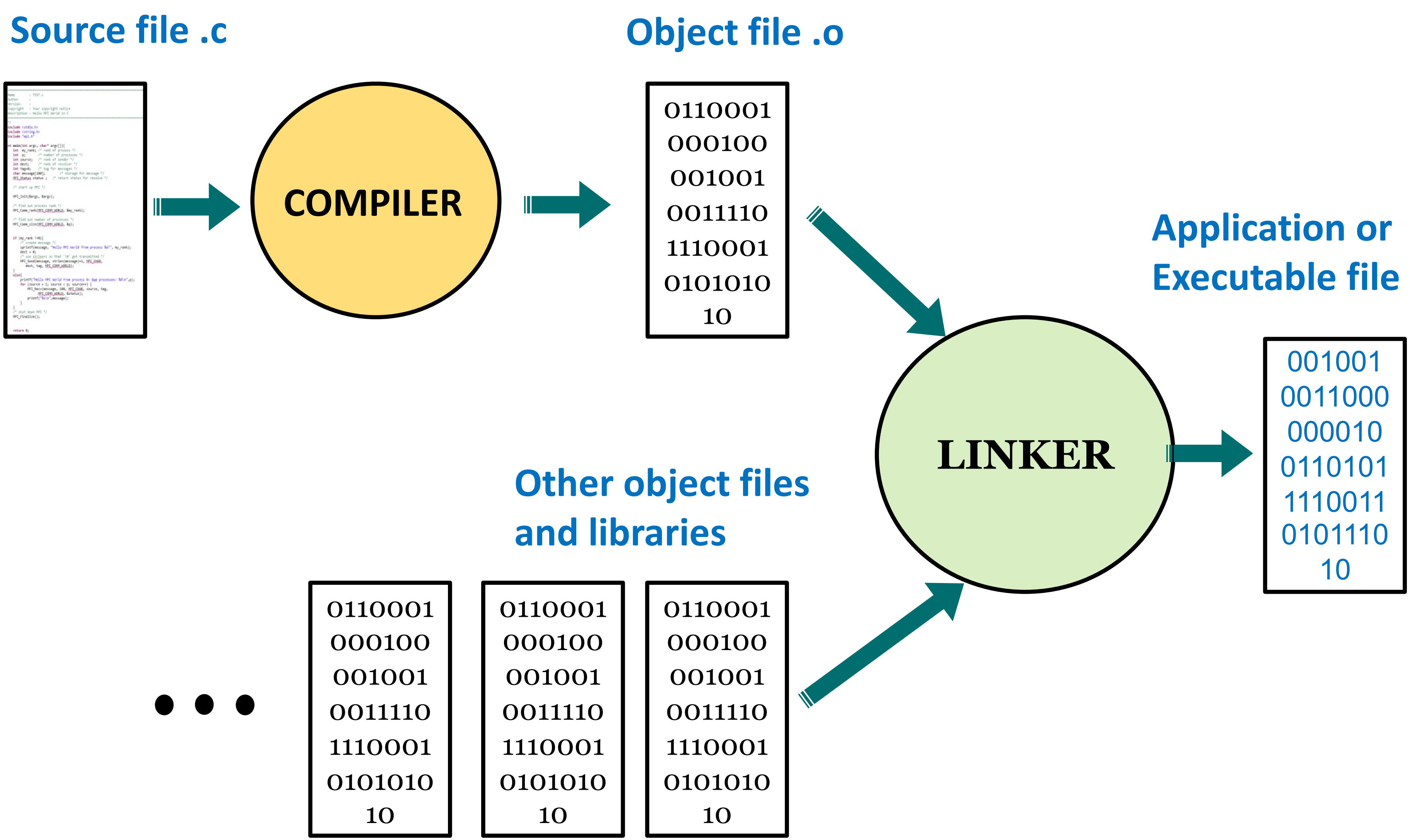
A FIRST software crisis

Programming Languages

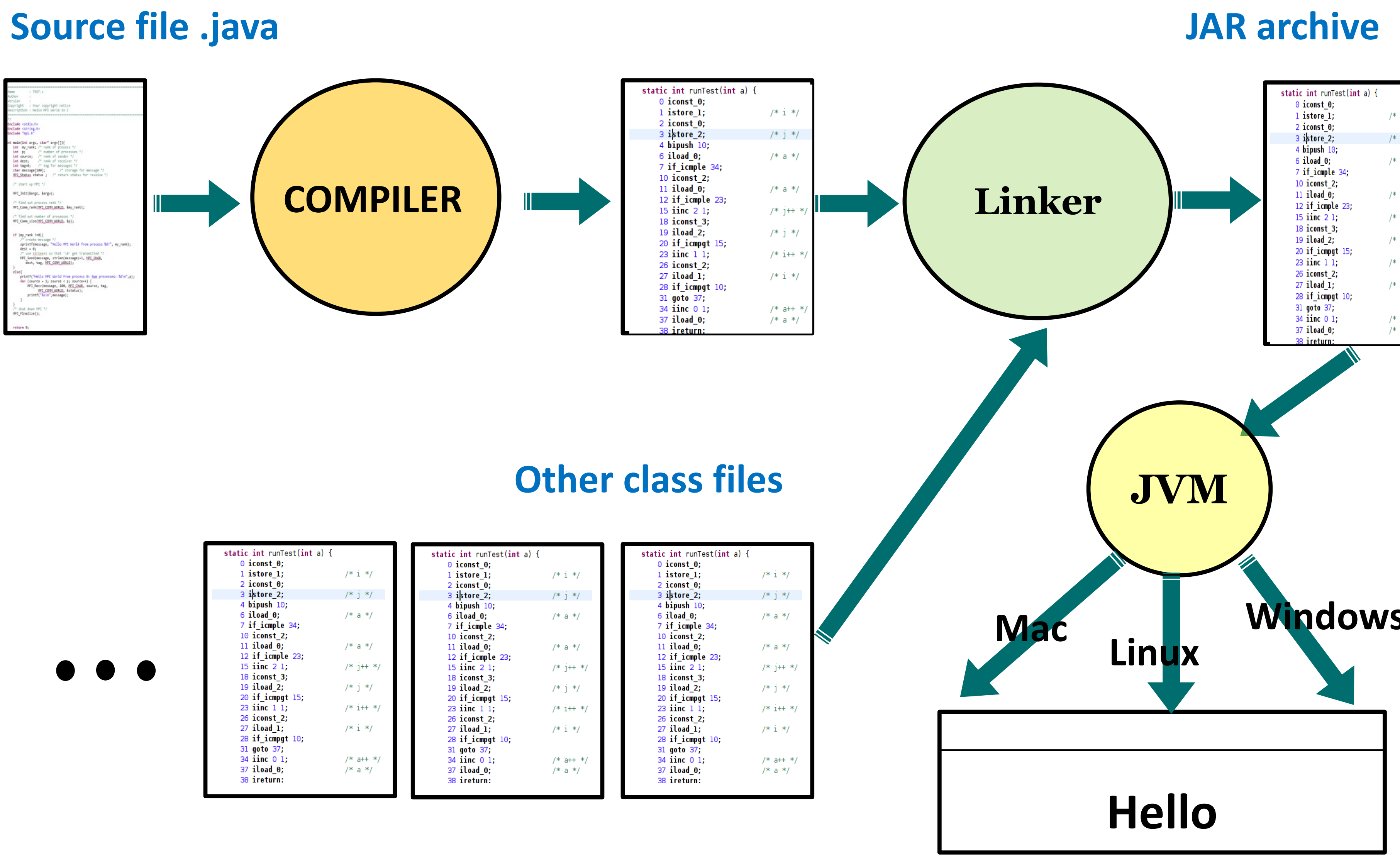
- Solution to the first software crisis:

- **High-Level Languages** (1957: FORTRAN)
  - Offer a common *abstraction* from the specific uni-processor hardware
  - Introduction of an **editor** and of the **compilation process**

The Compilation Process



The Compilation Process



Programing Languages

- Solution to the first software crisis:

- **High-Level Languages** (1957: FORTRAN)
  - Offer a common *abstraction* from the specific uni-processor hardware
  - Introduction of an editor and of the **compilation process**

PROBLEM WITH FORTRAN in 1960s-1980s:

Unstructured Language: **GOTO statement**

CODING HORROR

- **Structured languages:** Algol, Pascal, C,....

Programing Languages

- Solution to the first software crisis:

- **High-Level Languages** (1957: FORTRAN)
  - Offer a common *abstraction* from the specific uni-processor hardware
  - Introduction of an editor and of the **compilation process**

- **Structured languages:** Algol, Pascal, C,….

- **1980s / 90s: Problem**
  - Inability to build and *maintain complex and robust applications* requiring *multi-million lines of code* developed by hundreds of programmers
  - Computers could handle *larger more complex programs*
  - Needed to get *composability, malleability* and *maintainability*
  - High-performance was not an issue and left to Moore’s Law

→ **A SECOND software crisis**

Programing Languages

- Solution to the second software crisis:

- **Object Oriented Programming** (1983: C++)
  - Now also: C# and Java
  - Better tools and reusability through component libraries
  - Better software engineering methodology: specification, testing, reviews

Programing Languages

TODAY?

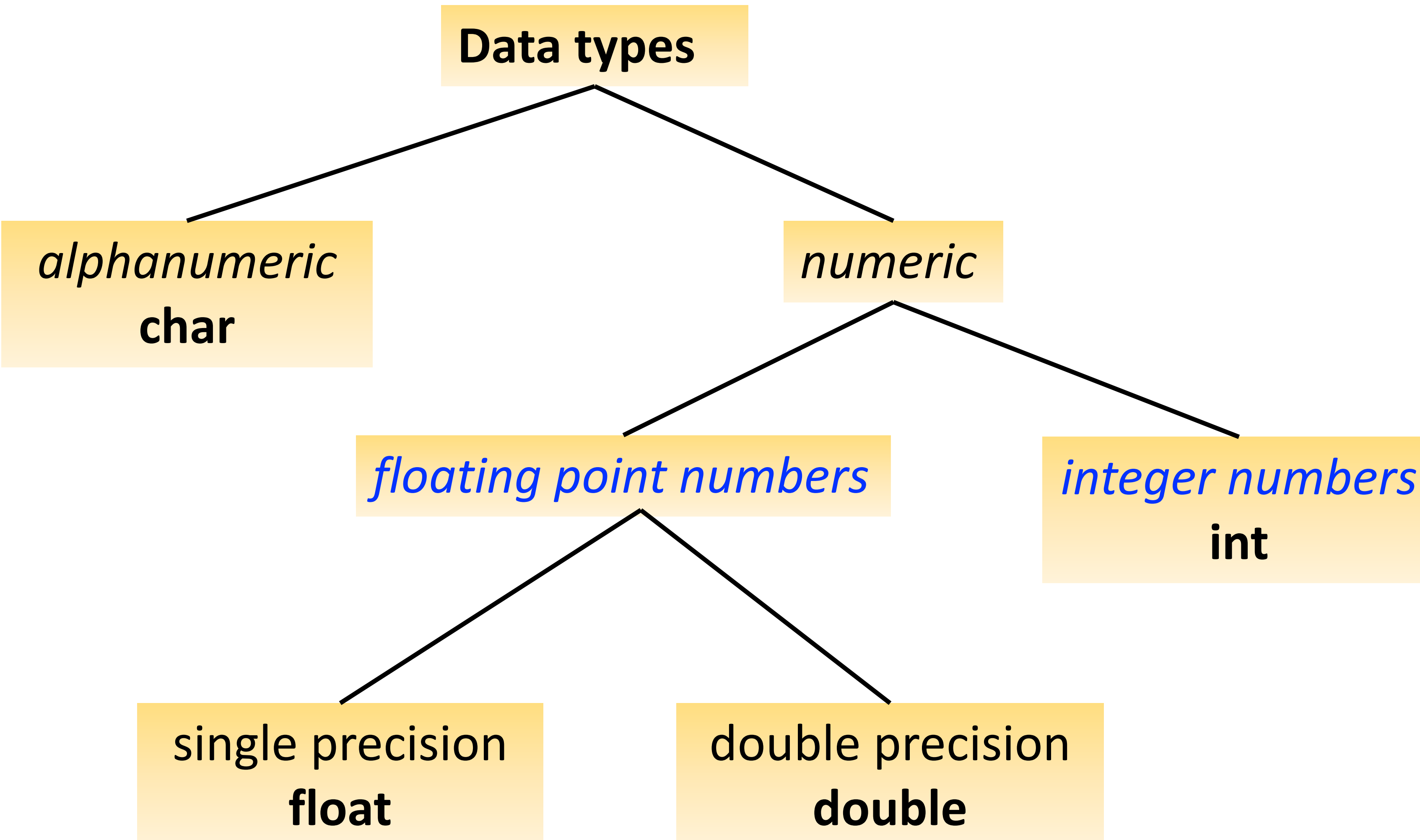
Programing Languages

- TODAY, programmers are oblivious to processors
- Programmers don‘t have to know anything about the particular processor
  - Moore‘s law does not require the programmers to know anything about the processors to get good speedups
  - A program written in 70s using C still works and is much faster today !!

Overview of Lecture 3

1	Quick Review: Programming Languages and the Compilation Process
2	Basic Data Types
3	Bit Representation of Numbers and Data Types in Computer Memory

Basic Data Types in C



Overview of Lecture 3

1	Quick Review: Programming Languages and the Compilation Process
2	Basic Data Types
3	Bit Representation of Numbers and Data Types in Computer Memory

Bit Representation of Characters

- **ASCII-CODE** (**A**merican **S**tandard for **C**oded **I**nformation **I**nterchange)
  - **Latin alphabet plus special characters**
  - **First bit is not used: In standard ASCII: 128 characters**

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Bit Representation of Characters

- **ASCII-CODE** (**A**merican **S**tandard for **C**oded **I**nformation **I**nterchange)
  - **Latin alphabet plus special characters**
  - **First bit is not used: In standard ASCII: 128 characters**
  - **Extended ASCII: 256 characters**

128	Ç	144	É	160	á	176	░	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Bit Representation of Characters

- **ASCII-CODE** (**A**merican **S**tandard for **C**oded **I**nformation **I**nterchange)
  - **Latin alphabet plus special characters**
  - **First bit is not used: In standard ASCII: 128 characters**
- **UNICODE (since 1991)**
  - **Idea: Complete collection of all written characters from all present and past cultures**
  - **At first: 2 bytes, then 4 bytes (2^{32} different characters)**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	[illegible]	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	[illegible]	[illegible]	[illegible] (enquiry)	[illegible]	[illegible]	[illegible]	[illegible]	%	69	45	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	[illegible]	[illegible]	[illegible] new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	[illegible]	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	[illegible]	[illegible]	[illegible]	[illegible]	7	[illegible]	[illegible]	[illegible]	[illegible]	W	[illegible]	[illegible]	[illegible]	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Basic Data Types in C and Their Domain on a 32 bit System

Data Type Name	Size	Range
`char`, `signed char`	8	$-128\ldots127$
`unsigned char`	8	$0\ldots255$
`short`, `signed short`	16	$-32\,768\ldots32\,767$
`unsigned short`	16	$0\ldots65\,535$
`int`, `signed int`	32	$-2\,147\,483\,648\ldots2\,147\,483\,647$
`unsigned`, `unsigned int`	32	$0\ldots4\,294\,967\,295$
`long`, `signed long`	32	$-2\,147\,83\,648\ldots2\,147\,483\,647$
`unsigned long`	32	$0\ldots4\,294\,967\,295$
`float`	32	$1.2\cdot10^{-38}\ldots3.4\cdot10^{38}$
`double`	64	$2.2\cdot10^{-308}\ldots1.8\cdot10^{308}$

Basic Data Types in C and Their Domain on a 32 bit System

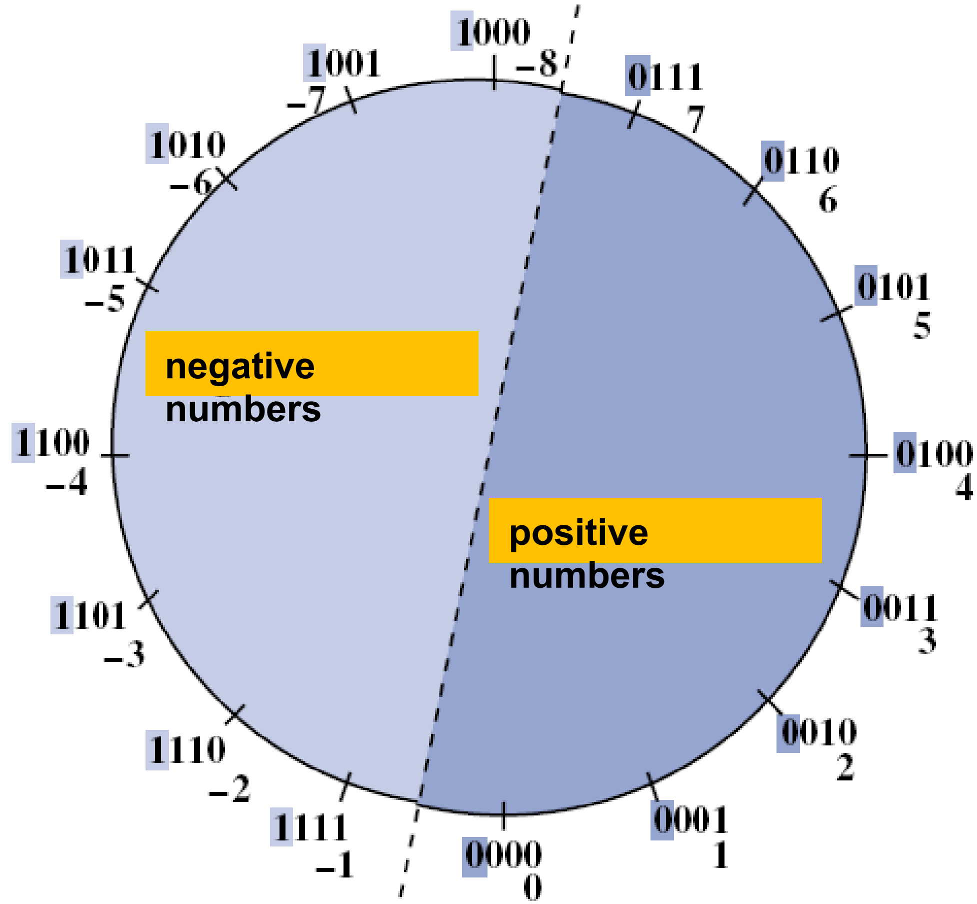
Data Type Name	Size	Range
`char, signed char`	8	$-128 \ldots 127$
`unsigned char`	8	$0 \ldots 255$
`unsigned, unsigned int`	32	$0 \ldots 4\,294\,967\,295$
`long, signed long`	32	$-2\,147\,83\,648 \ldots 2\,147\,483\,647$
`unsigned long`	32	$0 \ldots 4\,294\,967\,295$
`float`	32	$1.2\cdot 10^{-38} \ldots 3.4\cdot 10^{38}$
`double`	64	$2.2\cdot 10^{-308} \ldots 1.8\cdot 10^{308}$

There is NO out of range check in C/C++ when assigning numbers to data types

Bit Representation of Negative Numbers
Two`s Complement with four bits

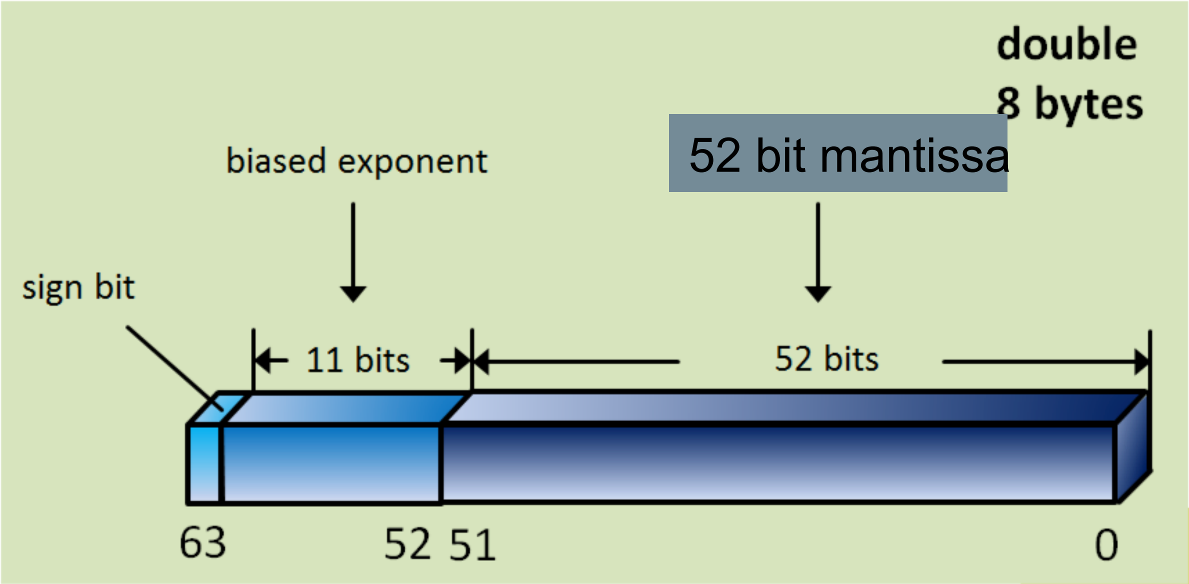
First bit as a sign bit

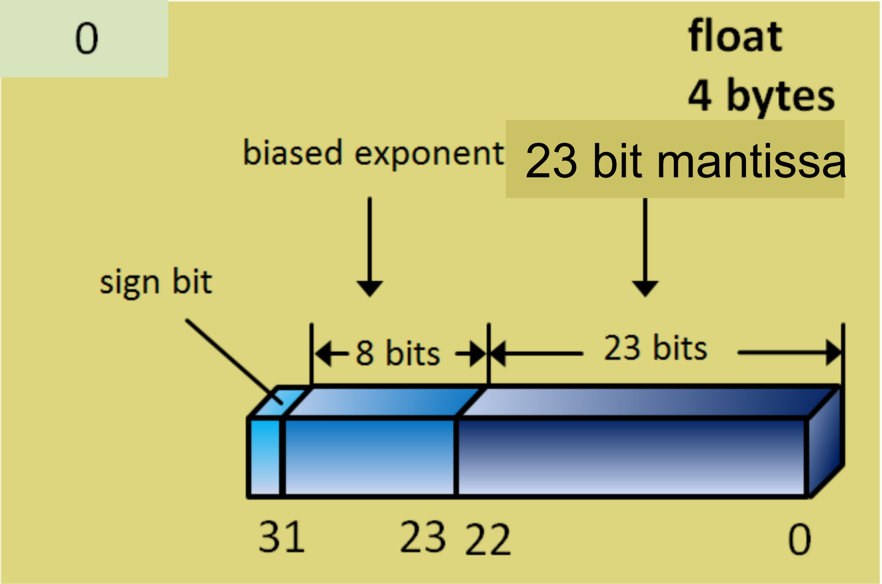
$0000 = 0$	$1000 = -8$
$0001 = 1$	$1001 = -7$
$0010 = 2$	$1010 = -6$
$0011 = 3$	$1011 = -5$
$0100 = 4$	$1100 = -4$
$0101 = 5$	$1101 = -3$
$0110 = 6$	$1110 = -2$
$0111 = 7$	$1111 = -1$



Subtraction can be reduced to the addition (of a negative number)

Bit Representation of Floating Point Numbers





Whiteboard Notes...

Summary

- Two‘s complement is used for adding bit patterns of numbers
- Characters are represented by bit patters outlined in the ASCII Table
- Numbers are just bit patterns which are interpreted in a particular way
- Data types are also just bit patterns, interpreted in a specific way
- There is no out-of-bound checking in C for basic data types

My University Research Page: https://www.frankfurt-university.de/steinhauser

Contact Me: martin.steinhauser@fb2.fra-uas.de

Research Gate: https://www.researchgate.net/profile/Martin-Steinhauser