



Computer Hardware Engineering (IS1200)

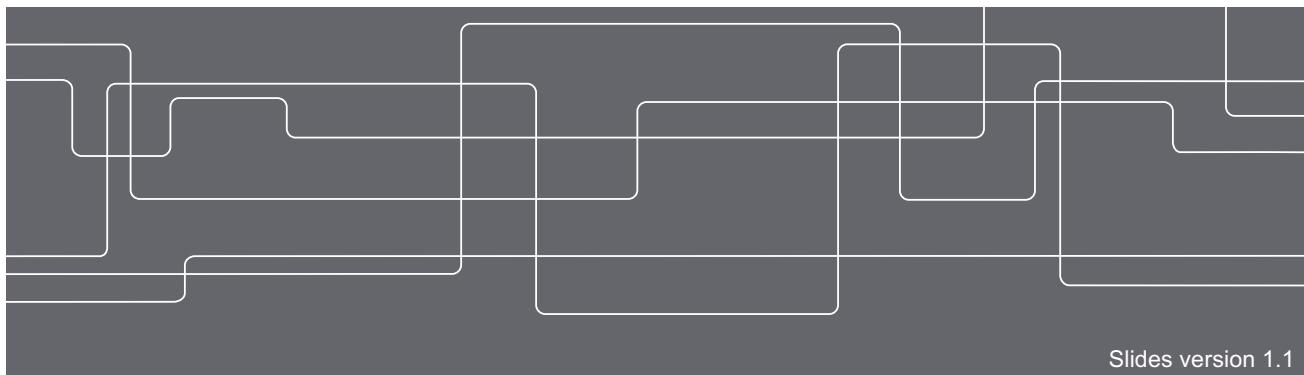
Computer Organization and Components (IS1500)

Fall 2018

Lecture 1: Course Introduction

David Broman

Associate Professor, KTH Royal Institute of Technology



Slides version 1.1

2



Different Kinds of Computer Systems



**Embedded
Real-Time Systems**



**Personal Computers and
Personal Mobile Devices**



**Warehouse
Scale Computers**

Dependability

Energy

Performance



How is this computer revolution possible?



Moore's law:

- Integrated circuit resources (transistors) double every 18-24 months.
- By Gordon E. Moore, Intel's co-founder, 1960s.
- Possible because of refined manufacturing processes. E.g., Intel Core i7-6800 processors uses 14nm manufacturing.
- Sometimes considered a *self-fulfilling prophecy*. Served as a goal for the semiconductor industry.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Have we reached the limit?

Why?

The Power Wall



http://www.publicdomainpictures.net/view-image.php?image=1281&picture=tegelvagg

Increased clock rate implies increased power

We cannot cool the system enough to increase the clock rate anymore...

During the last decade, the clock rate has increased dramatically.

- | | |
|----------------------|---------|
| • 1989: 80486, | 25MHz |
| • 1993: Pentium, | 66Mhz |
| • 1997: Pentium Pro, | 200MHz |
| • 2001: Pentium 4, | 2.0 GHz |
| • 2004: Pentium 4, | 3.6 GHz |

2019: Intel Xeon W, 3.2 GHz, 8 Cores (Turbo 4.2Ghz)

"New" trend since 2006: Multicore

- Moore's law still holds (but will end soon)
- More processors on a chip: multicore
- "New" challenge: parallel programming

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Agenda

Part I

Course Organization



Part II

Introduction to C



David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Part I

Course Organization



David Broman
dbro@kth.se



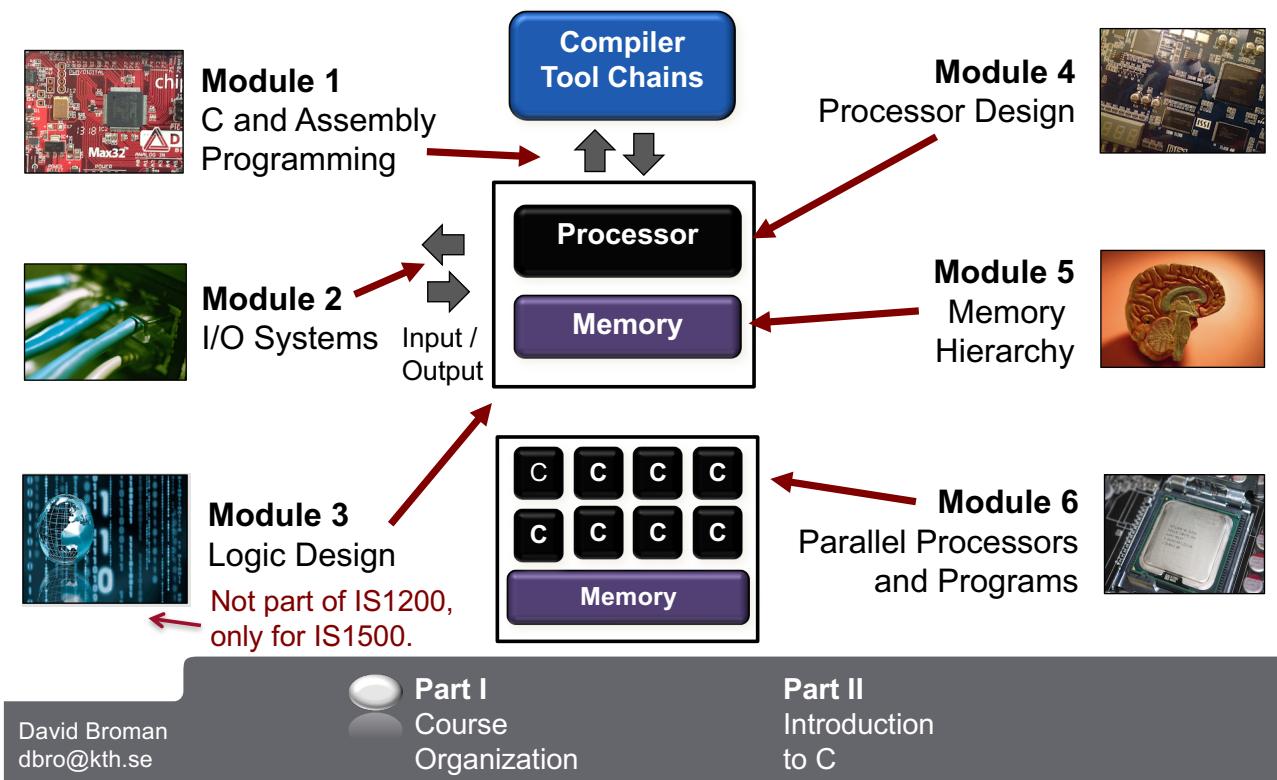
Part I
Course
Organization



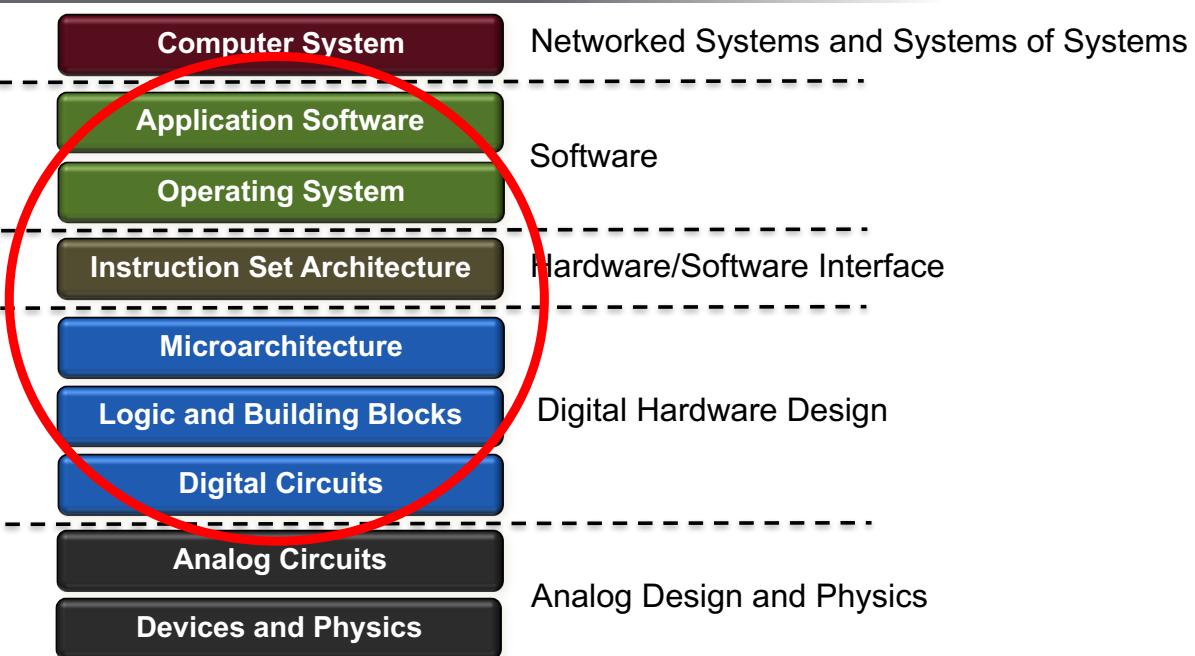
Part II
Introduction
to C



This Course in one Slide



Abstractions in Computer Systems





Learning Activities Overview



Lectures

- 12+2 lectures (2x45 min)
- Active participation
- Preslides before lectures



Empty Slides (Handouts)



Lecture Bugs



Physical Q and A



Exercises and Seminars

- 5+1 exercise classes with teaching assistants.
- 4 optional seminars with bonus (learning) points for the exam



Laboratory Exercises

- 4 laboratory exercises with examination at the KTH class labs
- 1 laboratory exercise for self study (only for IS1500)



Mini-project

- Project work on the ChipKIT board.
- 2 students in each group.

David Broman
dbro@kth.se



Part I
Course
Organization



Part II
Introduction
to C



Exercises and Seminars



Exercises (optional)

- 5+1 traditional KTH exercises lead by teaching assistants (TA).
- Try to prepare solutions in advance.
- Solutions are available on the course web.

Seminars (optional)

- 4 optional seminars.
- Students prepare (individually) solutions and bring them to the seminars.
- If you do not bring solutions, you cannot attend the seminar.
- Exercises are corrected together, while the TA explains the solutions.
- If you pass an exercise, you get 1 bonus point on the fundamental part of the exam.
- The bonus points are valid on the main exam + the following two retake exams.



NOTE: The main purposes of seminars are that you learn and prepare for the exam, not that you get bonus points (although you get this as a bonus).

David Broman
dbro@kth.se



Part I
Course
Organization



Part II
Introduction
to C



Laboratory Exercises

#	Lab	Examination
LAB1	Assembly Programming	KTH Lab Room
LAB2	C Programming	KTH Lab Room
LAB3	I/O Programming	KTH Lab Room
LD-LAB	Logic Design	(IS1500 only)
LAB4	Processor Design	KTH Lab Room

MARS MIPS Simulator

ChipKIT Uno with Basic I/O Shield

Logisim

- Prepare labs at KTH's computers or on your own computer.
- Preparation time for each lab: 8-24h
- 1 surprise exercise at the lab occasion.
- Each student book lab time separately in Canvas.

David Broman
dbro@kth.se



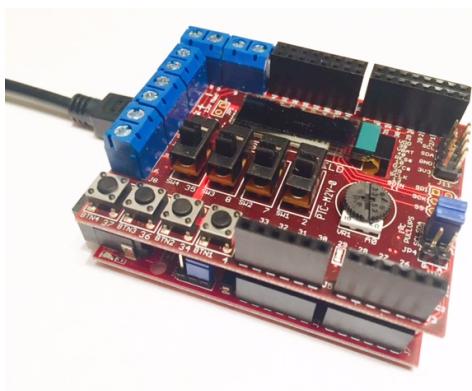
Part I
Course
Organization



Part II
Introduction
to C



Mini Project



- Each group should consist of 2 students. Should be the same as the lab groups.
- Student groups may collaborate, i.e., projects may be connected.
- You must use the ChipKIT hardware and you may add additional hardware components.
- Each group can borrow a hardware kit for free!
- Either you do a basic project or an advanced project (required to be able to get grades A or B)
- More info will come on lecture 6.
See also the course web.

Prestudy

Play around with the hardware, do labs, and think about what you want to do.

Extended abstract

1-2 pages, what you do, why, design, verification, etc.
Draft: February 11
Final abstract: at the EXPO

Project Expo

March 4
One day: everyone show their great project!
Nominations and Awards!



David Broman
dbro@kth.se

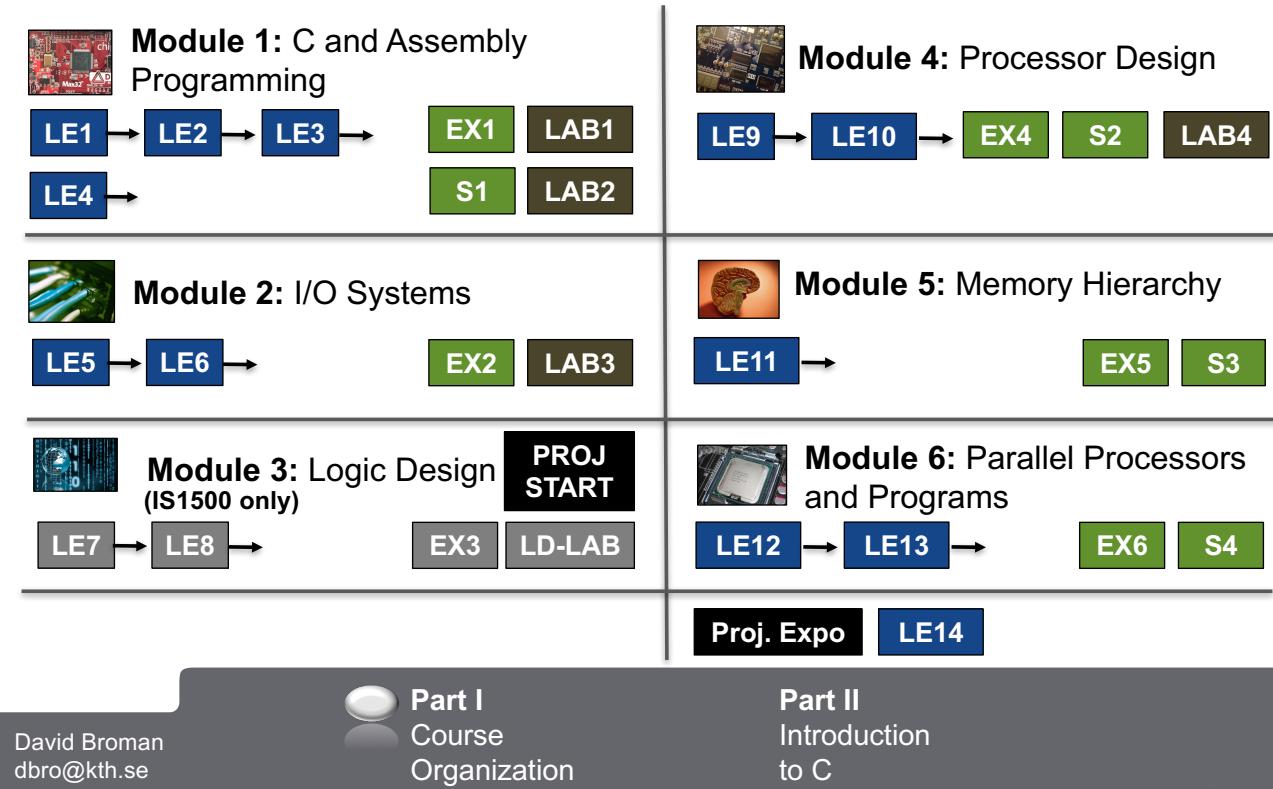


Part I
Course
Organization



Part II
Introduction
to C

Course Structure



David Broman
dbro@kth.se



Examined Course Parts



1.5hp ANN1. LA-LAB, Logic Design
Grades P and F
(IS1500 only, not IS1200)

4.5hp LAB1. Labs 1,2,3,4 and
Mini Project
Grades P and F

3hp TEN1. Written Exam (tenta)
Grades A, B, C, D, E, FX, F

David Broman
dbro@kth.se

Part I
Course Organization

Part II
Introduction to C



Written Exam

Written Exam (Tenta)

- March 15, 8-13, 2019, (5h)
- Retake exams June 2019, January 2020
- Allowed aids: One sheet of handwritten A4 paper (both sides) with notes.

The exam has two parts

• Part I: Fundamentals

- Max 40 points.
- 8 points for each of the 5 modules.
- Short questions with short answers.

• Part II: Advanced

- Max 50 points.
- Comprehensive questions.
Discuss, analyze, construct.

Grading of Exam

- To get a pass grade (A, B, C, D, or E), it is required to get at least 2 points on each module and in total 30 points on Part I (including bonus points).

Grading scale:

- A: 41-50 points on Part II
- B: 31-40 points on Part II
- C: 21-30 points on Part II
- D: 11-20 points on Part II
- E: 0-10 points on Part II
- FX: At least 30 points on Part I, and at most one module with less than 2 points.
- F: otherwise

To get A or B, you also need to have done an advanced project.

David Broman
dbro@kth.se



Part I
Course
Organization

Part II
Introduction
to C



Policy for Plagiarism

Note that all forms of cheating and plagiarism will be reported. Please see KTH's policy for handling plagiarism (see course web page).

Seminar exercises

- You are allowed to discuss the solutions of the exercises, but the final solutions must be written down and solved individually.
- It is not allowed to copy solutions in any way.



Written Exam

- You are allowed to bring one handwritten sheet of A4 paper. You may write on both sides. The A4 paper must not be printed or copied.
- All other aids (for instance calculators or text books) are not allowed.

Labs and Project code

- You may collaborate and discuss with anyone, but you must be able to explain all code you present to us individually, including your lab partner's code.
- When requested in the lab and project instructions, you must clearly declare who has authored the code that you hand in or show at lab examinations.

Written reports and project abstracts

- You are not allowed to copy, cut, or paste any text into your report that is not produced by you.
- The only exception is if you quote text properly, and give a citation to the original source.

David Broman
dbro@kth.se

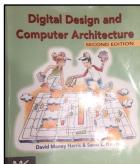


Part I
Course
Organization

Part II
Introduction
to C



Course Literature



Two Recommended Course Books

- David Money Harris and Sarah L. Harris. ***Digital Design and Computer Architecture***, Second Edition, Morgan Kaufmann, 2013.
- D. A. Patterson and J. L. Hennessy, ***Computer Organization and Design – the Hardware/Software Interface***, Fifth Edition, Morgan Kaufmann, 2013.

You may use the
4th edition instead
(available online)



Additional Online Course Material

- Laboratory Exercises
- Comics
- Manuals
- Other online material
- Exercises
- Lecture Slides

See the course webpage for
detailed reading guidelines.

David Broman
dbro@kth.se



Part I
Course
Organization

Part II
Introduction
to C



Teachers and Assistants



David Broman
dbro@kth.se
Examiner and
Course responsible.



Gunnar Johansson
gujo@kth.se
Labs and Admin

If it is not a personal message,
please email is1200@ict.kth.se
(goes to all the above)

David Broman
dbro@kth.se

Lectures

- David Broman
- Elias Castegren

Written Examination

- David Broman

Labs

- Gunnar Johansson
- Saranya Natarajan
- Viktor Palmkvist
- John Wikman
- Erik Båvenstrand
- Daniel Lundén
- Axel Pettersson
- Alexandra Kolonia
- Jacob Wahlgren

Exercises and Seminars

- Fredrik Lundevall
- Saranya Natarajan
- Daniel Lundén
- Viktor Palmkvist

Mini Project

- David Broman
- Fredrik Lundevall
- Gunnar Johansson
- Saranya Natarajan
- Daniel Lundén
- Viktor Palmkvist
- Elias Castegren



Part I
Course
Organization

Part II
Introduction
to C



Feedback



Web-based Course Evaluation

Standard course evaluation (after the course) is used to improve the course next year.



Photo by Julius Schorzman

Personal Feedback

Send me an email (dbro@kth.se) with feedback or let's talk over a cup of coffee!



Muddy Cards Course Evaluation (half time)

3-4 weeks into the course, we will hand out blank cards. Students write (anonymously) pros and cons about the course. The examiner collects, presents, and takes actions!



Course committee (kursnämnd)

A group of students meet up with me at the middle and at the end of the course. Informal oral feedback.

David Broman
dbro@kth.se



Part I
Course
Organization



Part II
Introduction
to C



Course Registration



Registration

- You must register for the course using KTH's web system. See the course web page "Registration" for more info.
- If you are re-registering for the course, you should go to the following webpage: <https://www.kth.se/en/eecs/kontakt/studentexpedition-och-servicecenter-1.21727>
- **Deadline February 1** (needed to take part in labs or seminars)

If you have any questions about course registration

- Please post your questions using the link above.

David Broman
dbro@kth.se



Part I
Course
Organization



Part II
Introduction
to C



Getting Help?



Canvas

- Post all questions about course content on the course website.
- Assistants and teachers will answer within the next working day.
Everyone can see the answers.



Lunch Office Hours

- One day every week, 12.15 – 13.00, teaching assistants will be available to answer questions about labs, project etc.
- See the course schedule for locations.



Email is1200@ict.kth.se

- Send administrative questions to: is1200@ict.kth.se
- Please post questions about exercises, labs, project etc. on Canvas.

David Broman
dbro@kth.se



Part I
Course
Organization

Part II
Introduction
to C



The course web pages in Canvas...

... contain a lot of useful information. Please read them carefully.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Part I

Introduction to C



David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Where is C coming from?

24
 E



The beginning

- Developed at AT&T Bell Labs in the years 1969-1973 by Dennis Ritchie (right in picture).
- C was developed in parallel with UNIX, originally designed by Ken Thompson (left in picture).



Dennis Ritchie and Ken Thompson received the Turing Award in 1983.



Standards

- The K&R book “**The C Programming Language**” (1st edition, 1978) by Brian Kernighan and Dennis Ritchie.
- ANSI C or C89 in 1989. Revised version, C99.
- Current standard, C11, approved December 2011.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C

What is C?



- C is an **imperative** low-level programming language (statements change program states).
- C is **not object-oriented** (as Java and C++), **not functional** (as Haskell, ML, and Ocaml), and **not interpreted** (as Perl, PHP, and Python typically are).
- C has types, but it is **not type safe** (in contrast to e.g., Java or Haskell)
- C allows **low-level memory access**, direct access to hardware, and has no garbage collection.
- C has minimal run-time requirements and can be compiled to many platforms. It is **very portable**.
- C is one of the most **widely used** programming language in the world. It is used in everything from microcontrollers to supercomputers.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Hello World!

Include library functions for handling standard input/output.

Comments start with /* and ends with */. Can be several lines.

Main function. Returns an integer value. Return code 0 = no error.

```
#include <stdio.h>
/* The main function. */
int main(void) {
    printf("Hello World!\n");
    return 0;
}
```

Compilation using **gcc** (GNU Compiler Collection). Without the output flag -o, the compiler produces an executable file named a.out.

```
$ gcc hello.c -o hello
$ ./hello
Hello World!
```

Shortcut
(try it!)

```
$ gcc hello.c && ./a.out
```

Library function **printf** prints to the standard output. "\n" means new line.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Constants and Literals

Integer Literals

233	Decimal
0x1A	Hexadecimal (prefix 0x)
012	Octal (prefix of 0)

Warning. A prefix 0 means that the base is 8 (octal numbers). This number means 10 in decimal representation.

Floating-point Literals

3.1415	With a decimal point
74e-6	With exponent

Character constants

'a'	A character
'\n'	New line
'\\'	\ character
'\'	' character
'\"	" character

String Constant

"This is a string"

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Whitespace, Identifiers, and Keywords

Tokens and Whitespace

Whitespace (space, newline, tab) separates tokens, but does not affect the program otherwise.

```
printf("Hello World!\n");
```

```
printf
("Hello World!\n"
)
;
```

These programs have the same meaning.

An **identifier** is a name used to identify user defined items, such as variables and functions.

foo _myVal
A32 Foo

Case sensitive. **foo** and **Foo** are different.

Can contain underscore, **A** to **Z**, **a** to **z**, and **0** to **9**, but cannot start with a digit (**0** to **9**).

A **keyword** is a reserved words that cannot be used as an identifiers.

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

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C

Example: Variables, Statements, and Expressions

A **variable** is defined by giving it a name and a type.

A variable can be assigned a value.

The right hand side of an assignment is an **expression**.

%d means print out decimal number (special for printf).

All **statements** end with a semicolon.
Statements are executed in sequence.

```
#include <stdio.h>

int main(void) {
    int a, b;
    int c = 5;
    b = 10;
    a = b * c;
    printf("%d\n", a);
    return 0;
}
```

What is the answer if
`int c = 5;`
is replaced with
`int c;`

Answer: Different, depending on what is in the memory.

Variables must be initialized carefully!

What is printed to the standard output?

Answer: 50

Conditional Statements if-statements

```
int x = 0;
if(x)
    printf("true");
if(x) {
    printf("true");
    x = 1;
}
```

C has no boolean type without including any extra library. Integer value 0 is interpreted as false, everything else as true.

Note: From version C99, a boolean type **bool** is available if library `<stdbool.h>` is included.

If there is more than one statement, the sequence of statements should be defined within a **block**, using { and }.

```
int y = 0, x = 1;
if(y)
    printf("true");
else{
    if(x)
        printf("false");
}
```

If-then-else constructs.

If-statements can be **nested**.

What is the output?
Answer: "false"

Binary Operators

```
int z;
z = 3 + 6;           addition: z = 9
z = 3 - 6;           subtraction: z = -3
z = 20 * 6;          multiplication: z = 120
z = 20 / 6;          division: z = 3
z = 20 % 6;          modulo: z = 2
```

```
int z = 10;
int x = 0;
if(++z == 10)
    x = 1;
```

pre-increment:
z = 11, x = 0

Unary Operators

```
int z = 5;
z++;               post-increment: z = 6
z--;               post-decrement: z = 5
++z;               pre-increment: z = 6
--z;               pre-decrement: z = 5
```

```
int z = 10;
int x = 0;
if(z++ == 10)
    x = 1;
```

post-increment:
z = 11, x = 1

```
int z;
int a = 1, b = 0, c = 3, d = 6;
z = a & b;           bitwise AND: z = 0
z = a && b;          boolean AND: z = 0
z = c & d;           bitwise AND: z = 2
z = c && d;          boolean AND: z = 1 (values other than 0 are treated as true)
z = a | b;           bitwise OR: z = 1
z = a || b;          boolean OR: z = 1
z = c | d;           bitwise OR: z = 7
z = c || d;          boolean OR: z = 1
z = c ^ d;           bitwise XOR: z = 5
z = c << 3;          bitwise shift left: z = 24
z = d >> 1;          bitwise shift right: z = 3
```

Physical Q/A (bitwise XOR)
Stand up for $c \wedge d = 5$
On the table for $c \wedge d = 7$



The **relational operators** return 1 if the relation is true and 0 if it is false.

Symbol	Description	Example
<code>==</code>	equal	<code>x == 5</code>
<code>!=</code>	not equal	<code>y != z</code>
<code><</code>	less than	<code>y < 7</code>
<code>></code>	greater than	<code>y > z</code>
<code><=</code>	less than or equal	<code>y ≤ 7</code>
<code>>=</code>	greater than or equal	<code>y ≥ z</code>

Note the difference between assignment `=` and test for equality `==`

```
int x = 0;
while(x < 10) {
    x++;
    printf("%d\n", x);
}
```

Loop while the expression is true (not 0)

Exercise:
A `break`-statement exits the loop.
Rewrite the above using `while(1)`

```
int x = 0;
while(1) {
    if(x >= 10)
        break;
    x++;
    printf("%d\n", x);
}
```

After C99, it is allowed to declare the iteration variable in the for loop.

```
#include <stdio.h>
int main(void){
    for(int i=0; i<128; i++){
        printf("%3d %3x %c\n", i, i, i);
    }
    return 0;
}
```

First element: initiate the counter.

Second element: the condition is tested in each iteration.

Third element: the increment is performed **at the end** of the block

What is this program doing?

It prints out the **ASCII** (American Standard Code for Information Interchange) table. Try it!

Print the ASCII character value

Print the hexadecimal value
(force 3 characters wide formatting)

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Reading Guidelines – Module 1



Introduction

P&H5 Chapters 1.1-1.4, or P&H4 1.1-1.3

Number systems

H&H Chapter 1.4

C Programming

H&H Appendix C

Online links on the literature webpage

Assembly and Machine Languages

H&H Chapters 6.1-6.9, 5.3

The MIPS sheet (see the literature page)

Reading Guidelines

See the course webpage
for more information.

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Almost at the end...



David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C



Summary

Some key take away points:

- **Moore's law:** Integrated circuit resources (transistors) double every 18-24 months.
- **The Power Wall:** Clock rates cannot be increased anymore. Too high power; the chip gets too hot.
- C is a **portable, low-level** language, used in everything from microcontrollers to supercomputers.
- C is **imperative**; states are updated by using assignments.



Thanks for listening!

David Broman
dbro@kth.se

Part I
Course
Organization

Part II
Introduction
to C