Computer Hardware Engineering (IS1200)

Computer Organization and Components (IS1500)

Spring 2021

Lecture 1: Course Introduction

Artur Podobas*, Daniel Lundén**, **David Broman*****

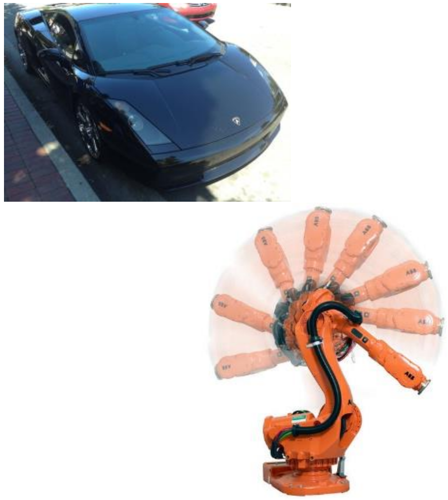
*   Researcher, KTH Royal Institute of Technology

** PhD Candidate, KTH Royal Institute of Technology

*** Associate Professor, KTH Royal Institute of Technology

Slides by David Broman, KTH. Updates by Artur Podobas.

Different Kinds of Computer Systems



Embedded
Real-Time Systems

Photo by Kyro

Personal Computers and
Personal Mobile Devices

Photo by Robert Harker

Warehouse
Scale Computers
Supercomputers

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.

Have we reached the limit?

Why?

The Power Wall

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

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
- **2020: ARM A64FX, 2.0 GHz, 48 Cores**
- *2022: European Processor Initiative Rhea Processor ?*

Increased clock rate implies increased power

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

Trend since 2006: Multicore and Accelerators

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

Agenda

Part I

Course Organization

Part II

Introduction to C

THE
C
PROGRAMMING
LANGUAGE

Part I

Course Organization

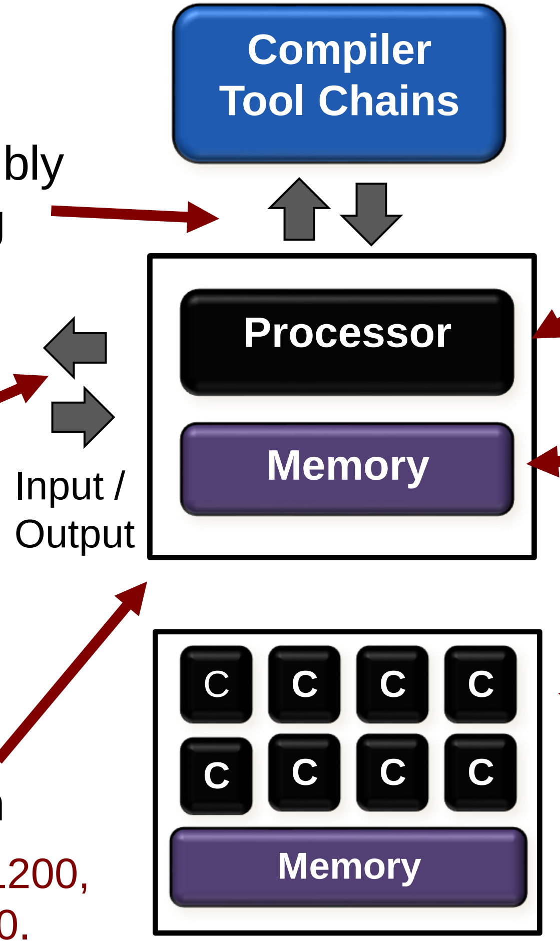
This Course in one Slide

Module 1
C and Assembly Programming

Module 2
I/O Systems

Module 3
Logic Design

Not part of IS1200, only for IS1500.

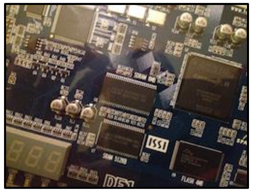


Compiler Tool Chains

Input / Output

Processor

Memory

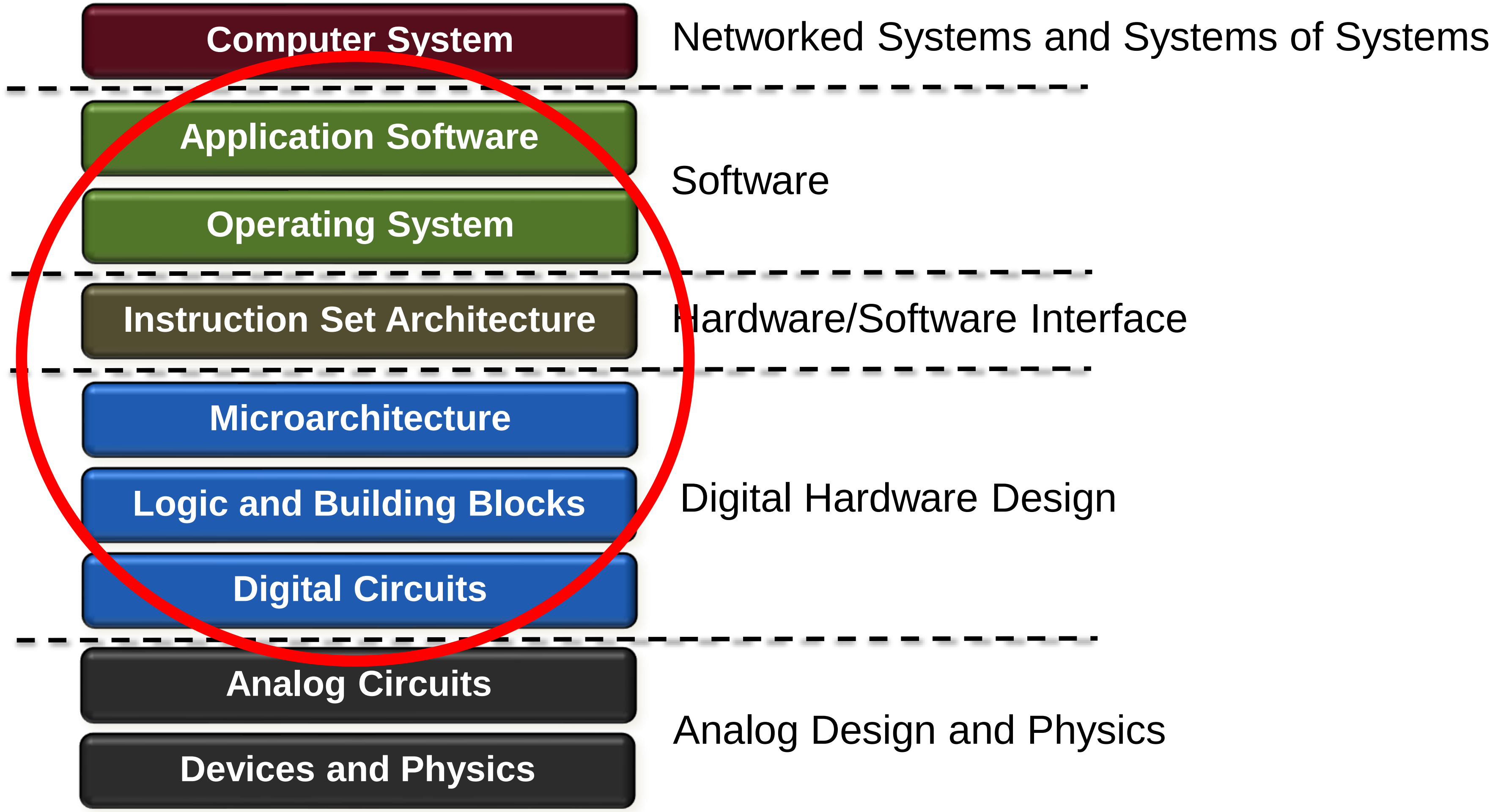


Module 4
Processor Design

Module 5
Memory Hierarchy

Module 6
Parallel Processors and Programs

Abstractions in Computer Systems



Computer System — Networked Systems and Systems of Systems

Application Software

Operating System

Software

Instruction Set Architecture — Hardware/Software Interface

Microarchitecture

Logic and Building Blocks

Digital Circuits

Digital Hardware Design

Analog Circuits

Devices and Physics

Analog Design and Physics

Learning Activities Overview

Lectures

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

Empty Slides (Handouts)

Lecture Bugs

Virtual 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 virtually
- 1 laboratory exercise for self study (only for IS1500)

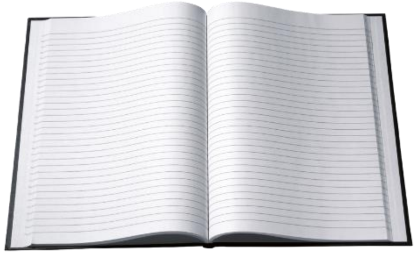
Mini-project

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

Due to COVID-19, Lectures, Labs, Seminars, and Exercises held online

- Zoom links will be posted on the Schedule page
- *Use Chat window to ask questions*
- More information on Canvas

Exercises and Seminars



Exercises (optional)

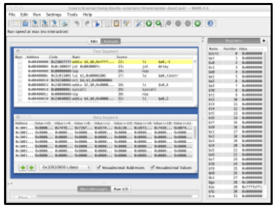
- 5+1 traditional KTH exercises lead by teaching assistants (TA).
- Try to prepare solutions in advance.
- Solutions are available on the course web.
- Select the occasion yourself (given more than on time)

Seminars (optional)

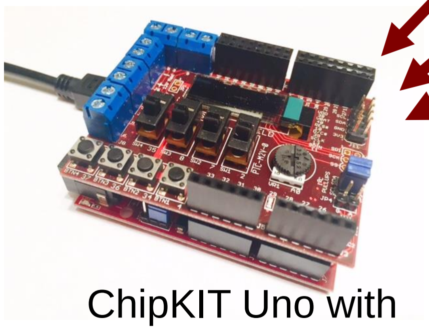
- 4 optional seminars. Select the occasion yourself.
- Students prepare (individually) solutions and send them through Canvas before the seminars.
- If you did not submit solutions then you cannot attend the Seminar.
- Exercises are corrected together, while the TA explains the solutions.
- If you pass 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).

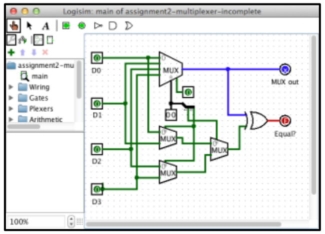
Laboratory Exercises



MARS MIPS Simulator



ChipKIT Uno with Basic I/O Shield

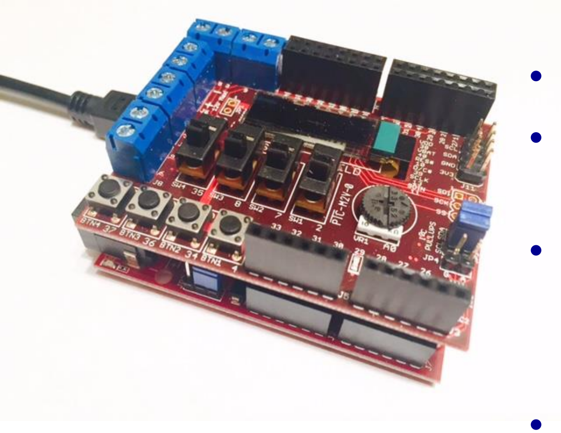


Logisim

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

- 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*.
- Labs will be examined virtually using Zoom rooms
- Borrow ChipKIT for free. Announcements where to pick them up will appear at Canvas shortly. Pick one board per group!

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!
  - Information will appear on Canvas where to pick your board up. Pick up one board per group (more boards on their way!).
- 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: Feb 8th

Final abstract: at the EXPO

Project Expo

March 8th

One day: everyone show their great project!

Nominations and Awards!

Course Structure

Module 1: C and Assembly Programming

LE1 → LE2 → LE3 →

LE4 →

EX1 LAB1

S1 LAB2

Module 2: I/O Systems

LE5 → LE6 →

EX2 LAB3

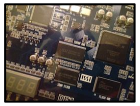
Module 3: Logic Design

(IS1500 only)

PROJ START

LE7 → LE8 →

EX3 LD-LAB



Module 4: Processor Design

LE9 → LE10 →

EX4 S2 LAB4

Module 5: Memory Hierarchy

LE11 →

EX5 S3

Module 6: Parallel Processors and Programs

LE12 → LE13 →

EX6 S4

Proj. Expo LE14

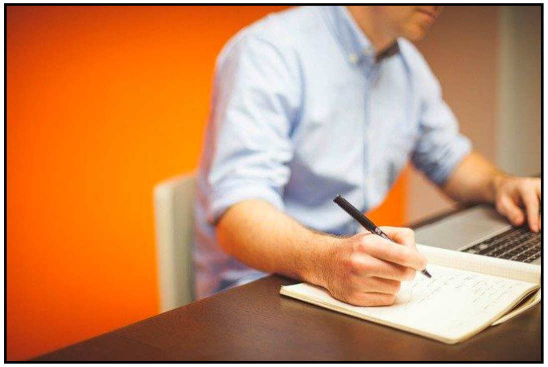
Examined Course Parts

1.5hp ANN1. LD-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

Written Exam (Tenta)



Wrong view, you need to show the computer screen.

Correct view

Main points

- Computer based, using VirtualBox with Ubuntu as the guest OS.
- Takes place either over Zoom or physically at KTH (depending on the Covid situation in March).
- Mobile camera, show computer. Be online on Zoom (if at home)
- Same exam format (kinds of questions), but in Canvas. You can study old exams.
- Everyone gets different questions. Randomized within each module.
- You can solve solutions by hand (on paper) or on the computer (using gcc and MARS).
- Rationale: better examination (real programming), harder to cheat (individual exercises), faster to correct (parts automatic).
- You will get more exact information before the exam.

Written Exam - Grading Criteria

Written Exam (Tenta)

- March 18, 2021, (5h)
- Retake exams June 2021, January 2022
- 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 answers.
- **Part II: Advanced.** 3 questions:
  - 1. Discuss (Focus module 6)
  - 2. Construct (Focus modules 1, 2)
  - 3. Analyze (Focus modules 3, 4)

Criteria: Satisfactory (S), Good(G), Very Good (VG)

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:

On part II:

A:	3VG **or** 2VG & 1G.
B:	1VG & 2G **or** 2VG & 1S.
C:	3G **or** 2G & 1S **or** 1VG & 2S **or** 1VG & 1G & 1F **or** 2VG & 1F **or** 1VG & 1G & 1S
D:	3 S **or** 1G & 1S & 1F **or** 2G & 1F **or** 1VG & 1S & 1F **or** 1VG & 2F **or** 1G & 2S
E:	No requirements on part II.
FX:	At least 30 points on Part I, and at most one module with less than 2 points. No req. on Part II.
F:	otherwise

To get A or B, an advanced project is also needed.

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.

Labs and Project code

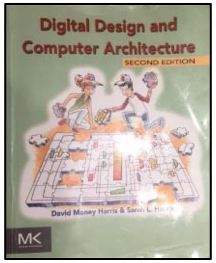
- You may collaborate and discuss with anyone, but you must be able to explain all code you present to us individually, including your lab partner’s code.
- When requested in the lab and project instructions, you must clearly declare who has authored the code that you hand in or show at lab examinations.

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.

KTH EECS Code of Honor
https://www.kth.se/en/eecs/utbildning/hederskodex

Course Literature

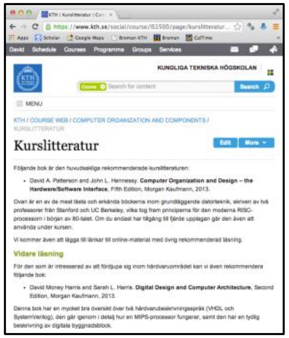


Two Recommended Course Books

- 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)


KTH | Kurslitteratur | Co
https://www.kth.se/social/course/IS1500/page/kurslitteratur...
Apps
Scholar
Google Maps
Broman KTH
Broman
CalTime
David
Schedule
Courses
Programme
Groups
Services
KUNGLIGA TEKNISKA HÖGSKOLAN
Course
Search for content
Search
MENU
KTH / COURSE WEB / COMPUTER ORGANIZATION AND COMPONENTS / KURSLITTERATUR
Edit
More

Kurslitteratur

Följande bok är den huvudsakliga rekommenderade kurslitteraturen:

- David A. Patterson and John L. Hennessy. **Computer Organization and Design – the Hardware/Software Interface**, Fifth Edition, Morgan Kaufmann, 2013.

Ovan är en av de mest lästa och erkända böckerna inom grundläggande datorteknik, skriven av två professorer från Stanford och UC Berkeley, vilka tog fram principerna för den moderna RISC-processorn i början av 80-talet. Om du endast har tillgång till fjärde upplagen går den även att använda under kursen.

Vi kommer även att lägga till länkar till online-material med övrig rekommenderad läsning.

Vidare läsning

För den som är intresserad av att fördjupa sig inom hårdvaruområdet kan vi även rekommendera följande bok:

- David Money Harris and Sarah L. Harris. **Digital Design and Computer Architecture**, Second Edition, Morgan Kaufmann, 2013.

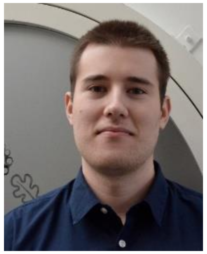
Denna bok har en mycket bra översikt över två hårdvarubeskrivningsspråk (VHDL och SystemVerilog), den går igenom i detalj hur en MIPS-processor fungerar, samt den har en tydlig beskrivning av digitala byggnadsblock.

Additional Online Course Material

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

See the course webpage for detailed **reading guidelines.**

Teachers and Assistants



Daniel Lundén
dlunde@kth.se
Course Responsible

Artur Podobas
podobas@kth.se
Teacher

David Broman
dbro@kth.se
Examiner

If it is not a personal message, *please* email *is1200@ict.kth.se* (goes to several teachers)

Lectures

- Daniel Lundén
- Artur Podobas

Written Examination

- David Broman

Labs

- Wei der Chien
- Lars Hummelgren
- Oscar Eriksson
- Daniel Lundén
- Saranya Natarajan
- Gizem Caylak
- + more assistants

Exercises and Seminars

- Fredrik Lundevall
- Saranya Natarajan
- Daniel Lundén
- Gizem Caylak

Mini Project

- Artur Podobas
- Fredrik Lundevall
- Saranya Natarajan
- Daniel Lundén
- Wei der Chien
- Gizem Caylak

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 an email to dlunde@kth.se with feedback or let’s talk over a cup of coffee!

Battery Evaluation (mid-course evaluation)
3-4 weeks into the (virtual) course, we will hand out (virtual) blank cards. Students write (anonymously) pros and cons about the course. The course responsible 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.

Course Registration

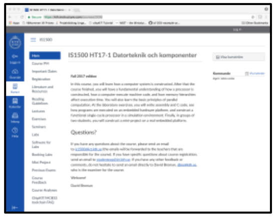
Registration

- 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 January 25th (needed to take part in labs or seminars)

If you have any questions about course registration

- Please post your questions using the link above.

Getting Help?


IS1500

IS1500 HT17-1 Datorteknik och komponenter

Fall 2017 edition

Questions?

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

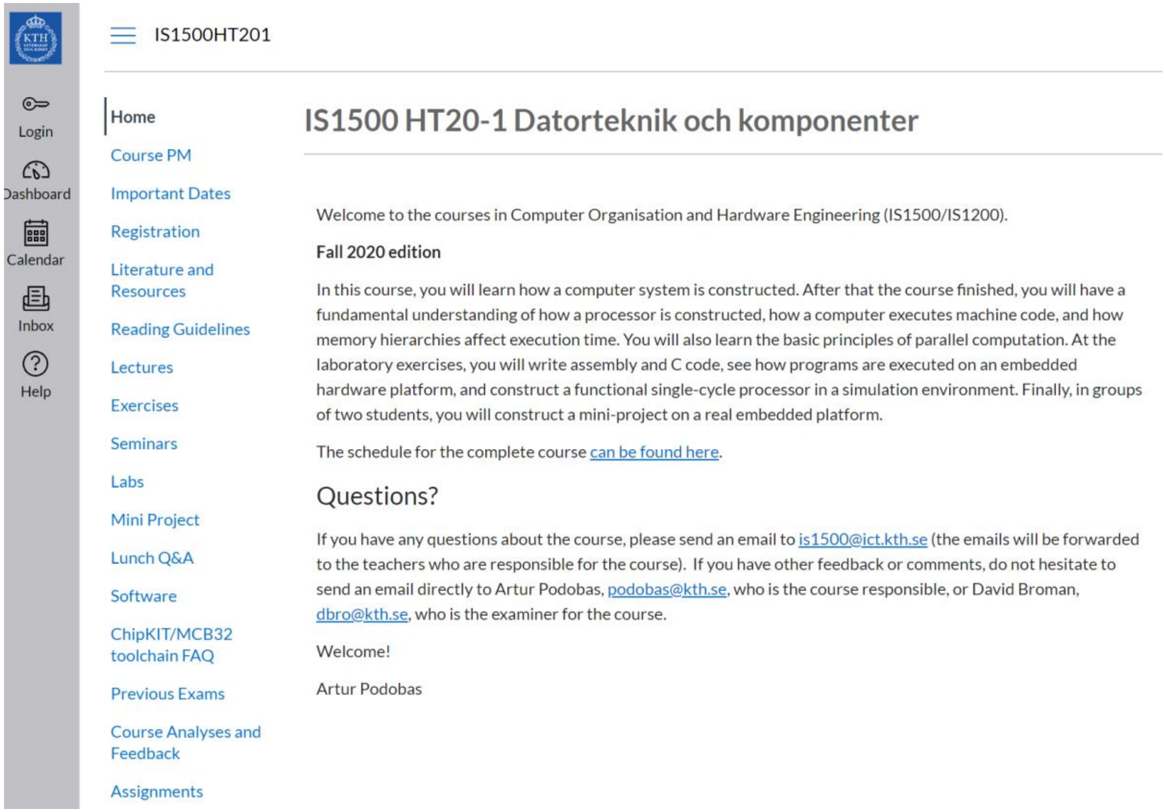
- One day every week, 12.15 – 13.00, teaching assistants will be available to answer questions about labs, project etc. (held online on Zoom.)
- See the Canvas for more information.

Email is1200@ict.kth.se

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

The course web pages in Canvas…

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



IS1500 HT20-1 Datorteknik och komponenter

Welcome to the courses in Computer Organisation and Hardware Engineering (IS1500/IS1200).

Fall 2020 edition

In this course, you will learn how a computer system is constructed. After that the course finished, you will have a fundamental understanding of how a processor is constructed, how a computer executes machine code, and how memory hierarchies affect execution time. You will also learn the basic principles of parallel computation. At the laboratory exercises, you will write assembly and C code, see how programs are executed on an embedded hardware platform, and construct a functional single-cycle processor in a simulation environment. Finally, in groups of two students, you will construct a mini-project on a real embedded platform.

The schedule for the complete course can be found here.

Questions?

If you have any questions about the course, please send an email to is1500@ict.kth.se (the emails will be forwarded to the teachers who are responsible for the course). If you have other feedback or comments, do not hesitate to send an email directly to Artur Podobas, podobas@kth.se, who is the course responsible, or David Broman, dbro@kth.se, who is the examiner for the course.

Welcome!

Artur Podobas

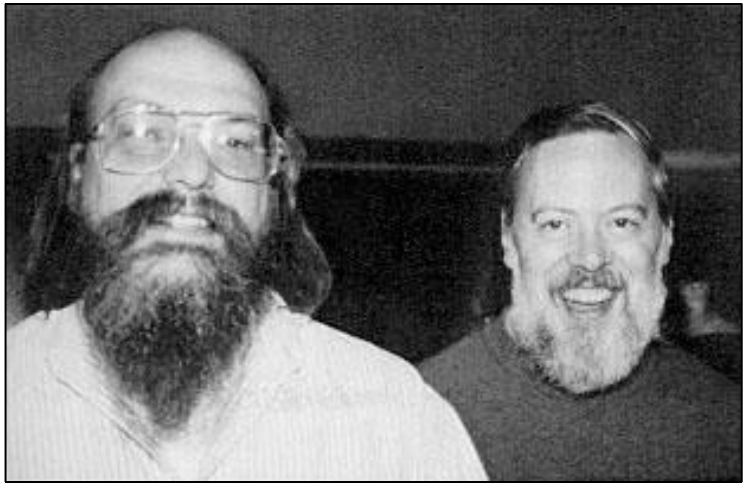
Part I

Introduction to C

(Note: We continue at 09:25)

THE C PROGRAMMING LANGUAGE

Where is C coming from?



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.
- C11, approved December 2011.
- Current standard, C17, 2018
- Upcoming standard: C2x (2021?)

Question: How many have of heard/used C before and the Turing award?

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 from microcontrollers to supercomputers.

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;
}
```

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

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
```

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"`

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
```

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

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

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
```

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 printed to the standard output?

Answer: 50

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!

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”

Operators (1/3)
Arithmetic Operators

Binary Operators

```
int z;
z = 3 + 6;
z = 3 - 6;
z = 20 * 6;
z = 20 / 6;
z = 20 % 6;
```

addition: z = 9

subtraction: z = -3

multiplication: z = 120

division: z = 3

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++;
z--;
++z;
--z;
```

post-increment: z = 6

post-decrement: z = 5

pre-increment: z = 6

pre-decrement: z = 5

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

post-increment:
z = 11, x = 1

Operators (2/3)
Logical and Bitwise Operators

```
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
```

Virtual Q/A:

Mark all statements that you would like to learn more about.

Operators (3/3)
Relational Operators

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

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

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

Loops (1/2)
while

```
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);
}
```

Loops (2/2)
for

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

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

```c
#include <stdio.h>

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

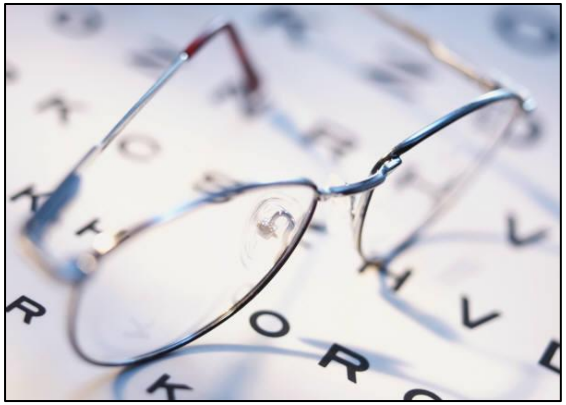
Print the ASCII character value

Print the hexadecimal value (force 3 characters wide formatting)

What is this program doing, and what is the output for iteration i=65?

It prints out the **ASCII** (American Standard Code for Information Interchange) table. **Try it!**
Output: 65 41 A

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.

Almost at the end…

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!