Computer Organization and Components

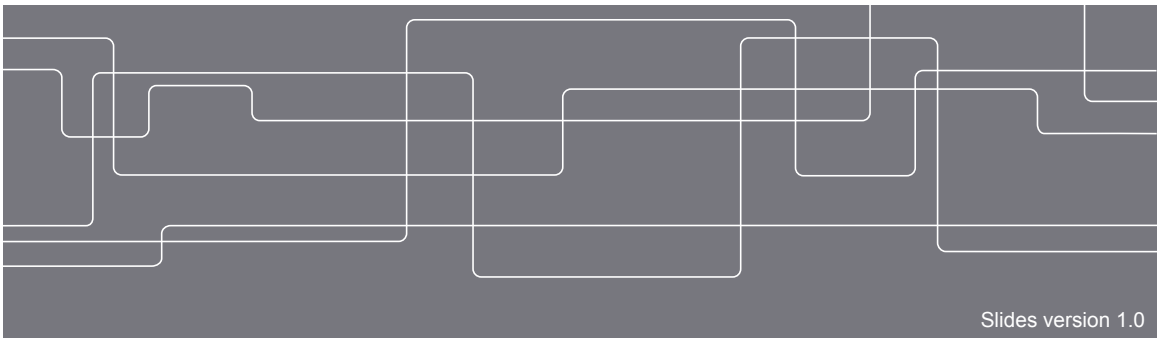
IS1500, fall 2015

Lecture 1: Course Introduction

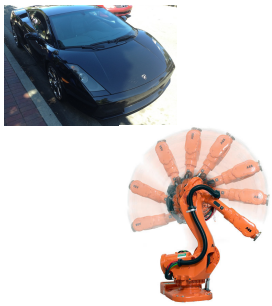
David Broman

Associate Professor, KTH Royal Institute of Technology

Assistant Research Engineer, University of California, Berkeley



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

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 refined manufacturing process. E.g., 4th generation Intel Core i7 processors uses 22nm 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

2015: Core i7, 3.1 GHz - 4 GHz

Increased clock rate implies increased power

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

"New" trend since 2006: Multicore

- Moore's law still holds
- More processors on a chip: multicore
- "New" challenge: parallel programming

Agenda

Part I
Course Organization

Part II
Introduction to C

Part I

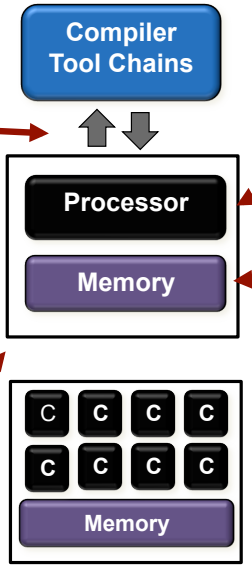
Course Organization

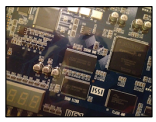
This Course in one Slide

Module 1
C and Assembly Programming

Module 2
I/O Systems

Module 3
Logic Design



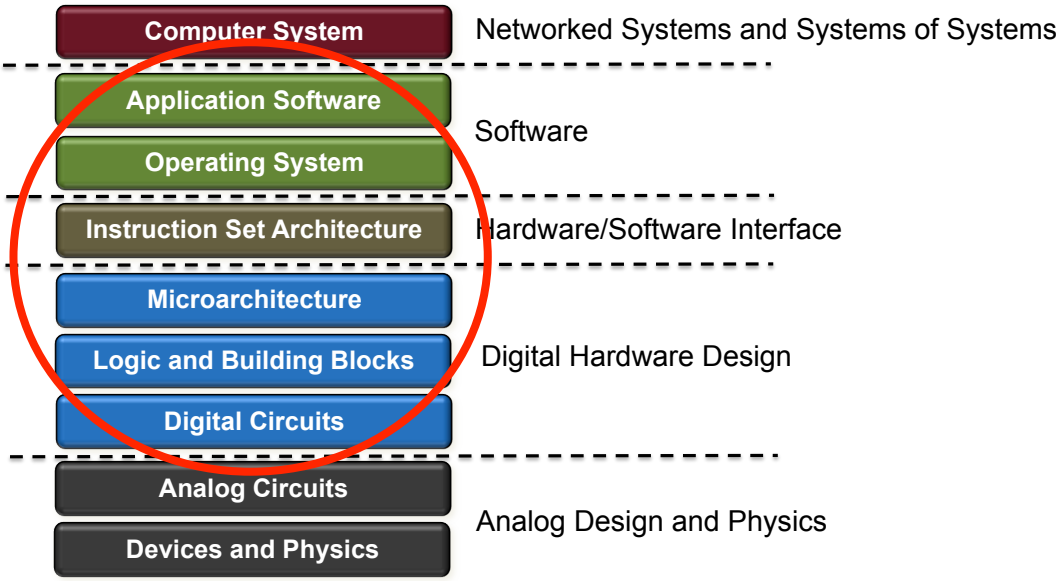


Module 4
Processor Design

Module 5
Memory Hierarchy

Module 6
Parallel Processors and Programs

Abstractions in Computer Systems



Learning Activities Overview

Lectures
- 14 lectures (2x45 min)
- Active participation
- Preslides before lectures

Empty Slides (Handouts)

Lecture Bugs

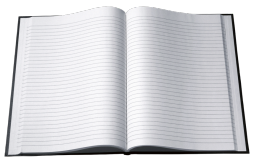
Physical Q and A

Exercises and Seminars
- 5 exercise classes with teaching assistants
- 3 optional seminars with bonus points for the written exam

Laboratory Exercises
- 3 laboratory exercises with examination at the KTH class labs
- 1 laboratory exercise with individual lab report
- 1 laboratory exercise with individual oral examination

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

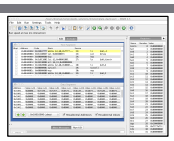
Exercises and Seminars



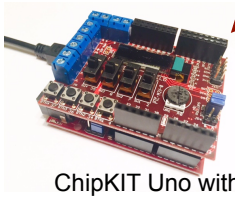
Exercises (optional)
- 5 traditional KTH exercises lead by teaching assistants (TA).
- Try to prepare solutions in advance
- Solutions are available on the course web.

Seminars (optional)
- 3 optional seminars
- Students prepare (individually) solutions and brings them to the seminar
- 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 + two following retake exams.

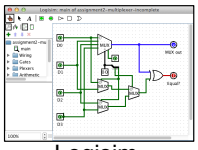
Laboratory Exercises



MARS MIPS Simulator



ChipKIT Uno with Basic I/O Shield

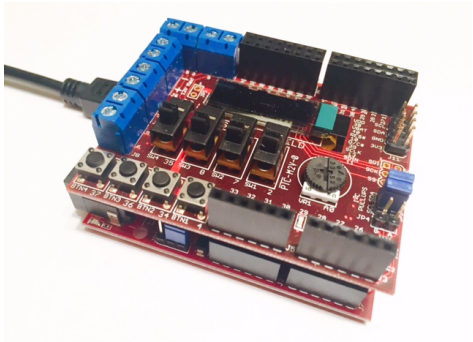


Logisim

#	Lab	Examination
LAB1	Assembly Programming	KTH Lab Room
LAB2	C Programming	KTH Lab Room
LAB3	I/O Programming	KTH Lab Room
LAB4	Logic Design	Individual Lab Report
LAB5	Processor Design	KTH Lab Room
LAB6	Cache Memories	Individual Oral Examination

- Prepare labs at KTH's computers or on your own computer.
- Preparation time for each lab: 4-25h
- 1-2 surprise exercises at the lab occasion.
- Each student book lab time separately in *Daisy*.

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 and electrical components.
- Each group can borrow a hardware kit for free!
- More info will come on lecture 6.

Prestudy (4 Sep – 1 Nov)

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

Extended abstract

1 page, what you do, why, design, verification, etc.

Draft: Nov 1

Peer review: Nov 8

Final abstract: Dec 6

Project Expo

December 6

One day: everyone show their great project!

Nominations and Awards!

Course Structure

Module 1: C and Assembly Programming

LE1 → LE2 → LE3 → EX1 LAB1

LE4 → S1 LAB2

Module 4: Processor Design

LE9 → LE10 → S2 LAB5

Module 2: I/O Systems

LE5 → LE6 → EX2 LAB3

Module 5: Memory Hierarchy

LE11 → EX4 LAB6

Module 3: Logic Design

LE7 → LE8 → EX3 LAB4

Module 6: Parallel Processors and Programs

LE12 → LE13 → EX5 S3

Proj. Expo LE14

Examined Course Parts

1.5hp **ANN1. Lab 4, Digital Design**
Grades P and F

4.5hp **LAB1. Labs 1,2,3,5,6 and Mini Project**
Grades P and F

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

Written Exam

Written Exam (Tenta)
- Monday, Jan 11, 2016, 15.00-20.00
- Retake exam, Spring 2016
- Allowed aids: One sheet of handwritten A4 paper (both sides) with notes.

The exam has two parts
- **Part I: Fundamentals**
  - Max 48 points.
  - 8 points for each of the 6 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 35 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: 32-34 points and 11-50 points on Part II.
- F: otherwise

Cheating

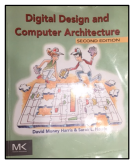
Note that all forms of cheating will be reported to KTH's disciplinary committee.

See the course PM for more information.

Labs and Project
You may discuss the problems with anyone, but you must *create all solutions yourself* (individually or with your lab partner) and clearly declare what you (as an individual) have done.

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.

Course Literature

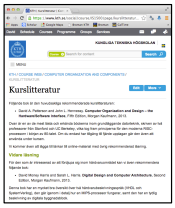


Two Recommended Course Books

- 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.**

Teachers and Assistants

David Broman
dbro@kth.se
Examiner and Lecturer

Fredrik Lundevall
flu@kth.se
Labs and Exercises

Gunnar Johansson
gujo@kth.se
Labs and Admin

If it is not a personal message, *please* email *is1500@ict.kth.se* (goes to all the above)

Lectures
- David Broman

Written Examination
- David Broman

Labs
- Fredrik Lundevall
- Gunnar Johansson
- Johannes Olsson
- Johan Engberg
- Eve (Minjia) Chen
- Ermias Gebremeskel
- Johan Myrsmeden
- Axel Isaksson

Exercises and Seminars
- Fredrik Lundevall
- Johannes Olsson
- Johan Myrsmeden
- Johan Engberg
- David Broman

Mini Project
- Axel Isaksson (Q&A)
- David Broman
- Fredrik Lundevall
- Gunnar Johansson

Feedback and Course Improvements

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

Photo by Julius Schorzman

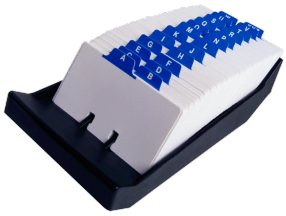
Personal Feedback
Send me an email (dbro@kth.se) with feedback or let's talk over a cup of coffee!

Muddy Cards
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.

Course Registration

Registration
We will register you to the course when you either
- report your first laboratory exercise, or
- write the written exam



To do this, you or your study advisor must have done the following:
- You need to be enrolled on this year's course. You enroll at antagning.se, during the previous semester (swedish: "termin")
  Any problem, talk to your study advisor.
- You need to be registered on the current semester when this course is given (swedish: "terminsregistrerard"). See the KTH's student web.
  Any problem, talk to your study advisor.

Getting Help?

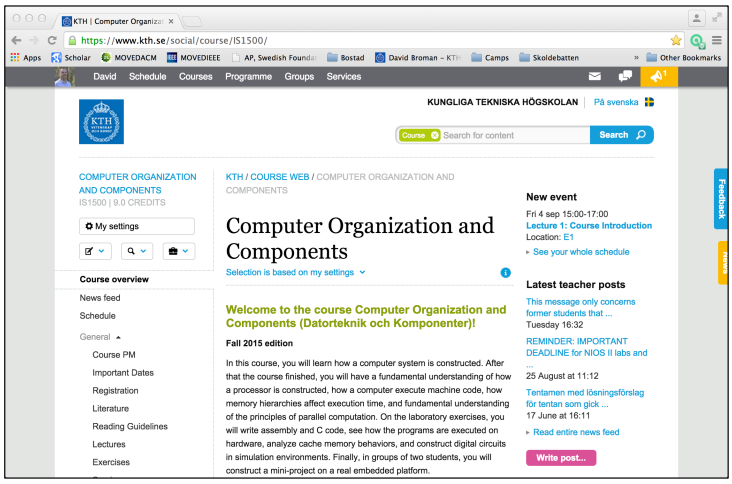
KTH Social
- Post all questions about course content on the course website.
- Assistants and teachers will answer. Everyone can see the answers.

Lunch Office Hours
- Every Wednesday 12.15 – 13.00 teaching assistants will be available to answer questions about labs, project etc.
- See the course schedule for locations.

Email is1500@ict.kth.se
- Send administrative and registration questions to:
- Please post questions about exercises, labs, project etc. on KTH social.

The Course Web page…

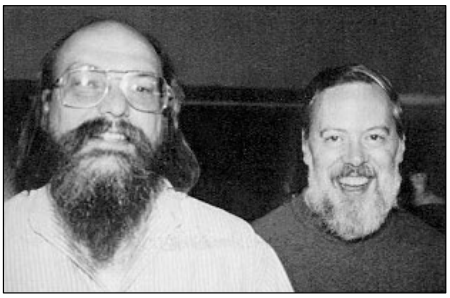
… contains a lot of useful information. Please read it carefully.



Part I

Introduction to C

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 (before a standard).
- ANSI C or C89 in 1989. Revised version, C99.
- Current standard, C11, approved December 2011.

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. 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(){
  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!
```

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

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

What is printed to 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;
z = a && b;
z = c & d;
z = c && d;

z = a | b;
z = a || b;
z = c | d;
z = c || d;

z = c ^ d;

z = c << 3;
z = d >> 1;
```

bitwise AND: z = 0
boolean AND: z = 0
bitwise AND: z = 2
boolean AND: z = 1 (values other than 0 are treated as true)

bitwise OR: z = 0

boolean OR: z = 1
bitwise OR: z = 7
boolean OR: z = 1

bitwise XOR: z = 5

bitwise shift left: z = 24
bitwise shift right: z = 3

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

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`

First element: the condition is tested in the beginning.

Second element, initiates the loop (executed first, once).

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

```
#include <stdio.h>

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

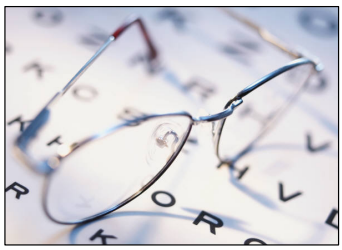
Print the ASCII character value

Print the hexadecimal value (force 3 characters wide formatting)

What is this program doing?

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

Reading Guidelines – Module 1



Reading Guidelines
See the course webpage for more information.

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)

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!