High Performance Programming and Systems
Course Outline - Student Version

Troels Henriksen
Kenneth Skovhede
Teaching Assistants
DIKU

2020/2021
Block: 2
Schedule Group: A

Document Updated: 2020-11-20 12:35:09.224474

Lab 1: Tuesday 17 November, Week 1 (47)

Location: DIKU, Zoom
Lecturers: Kenneth Skovhede and Troels Henriksen

Outline

- General introduction to the course and assignments
- Analog and digital signal primitives
- Working with numbers in a binary system
- Building a simple processor
- Improving a simple processor for speed
- Writing a simple program for a general purpose processor

Learning Goals

- Enumerate differences between analog and digital systems
- Compute basic arithmetic
- Enumerate properties of numbers in Two's complement notation
- Describe the basic properties of a Von Neumann Architecture
- Describe the basic properties of a Turing Machine
- Explain how modern machines are equivalent to a Turing Machine
- Sketch out a conceptual CPU
- Describe the role of registers in a CPU
- Enumerate the pros and cons of a piplelined CPU
- Explain Amdahls Law and relate it to computer programs
- Enumerate the differences between RISC and CISC classes
- Describe and create small programs written in machine code

Reading

Computer Systems: A programmers perspective: Chapter 1, Pages 43—47, Chapter 2, Pages 67—95, Chapter 3, Pages 200—202, Chapter 4, Pages 432—436, Pages 440—457:

2.0 Representing and Manipulating Information
2.1 Information storage
1.4 Processors Read and Interpret Instructions Stored in Memory
4.3.2 SEQ Hardware Structure
4.3.4 SEQ Stage Implementations
4.4 General Principles of Pipelining
3.0 Machine Level Representations of programs

Notes

For this introductory session you can view the video here:

- https://sid.erda.dk/share_redirect/cO5fYbAdIo/1%20-%20Lecture.mp4

For the lab session, the details are here:

- https://github.com/diku-dk/hpps-e2020-pub/tree/master/material/1-l-1

The first session will be online only, join here:

- https://ucph-ku.zoom.us/j/63242906073

Lab 2: Thursday 19 November, Week 1 (47)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Kenneth Skovhede

Outline

- Numeric representations in hardware
- Introduction to the C programming tools and language
- Introducing assignment 1

Learning Goals

- Enumerate different kinds of bitwise integer representations
- Explain how to add and multiply integer numbers in hardware
- Describe how floating point numbers are stored with bits
- Explain the purpose of floating point normalization
- Describe the method for normalizing floating point number
- Read and write a basic C program
- Enumerate common C errors and messages

Reading

Computer Systems: A programmers perspective: Chapter 1, Pages 39—43, Chapter 2, Pages 95—162, Chapter 3, Pages 205—213:

2.2 Integer Representations
2.3 Integer Arithmetic
2.4 Floating Point
1.1 Information Is Bits + Context
1.2 Programs Are Translated by Other Programs into Different Forms
1.3 It Pays to Understand How Compilation Systems Work
3.2 Program Encodings

Modern C: Chapter 3, Pages 21—28:

3. Everything is about control

Notes

The video presentation covers the basics of the C programming language, to the extent it is required for solving the first assignment. If you have never encountered a C-like programming language, or feel unsure about the syntax and operations, we suggest that you also briefly read chapters 1 and 2 in “Modern C”.

The video for this session is here:

- https://sid.erda.dk/share_redirect/hMrFbYiUgC/2%20-%20Lecture.mp4

For the lab session, the details are here:

- https://github.com/diku-dk/hpps-e2020-pub/tree/master/material/1-l-2

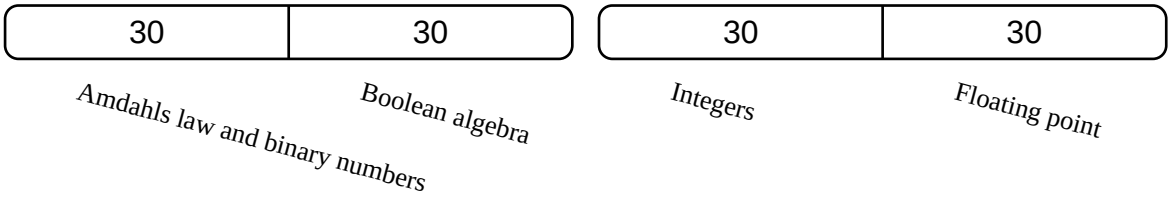
Join by zoom with this link:

- https://ucph-ku.zoom.us/j/63242906073

Exercise 1: Thursday 19 November, Week 1 (47)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Kenneth Skovhede, Troels Henriksen, and Teaching Assistants

Plan



Notes

In this exercise session you will work with hands-on examples that will help you solve the first assignment, as well as train you in exam-like problems.

Amdahls Law
- Problem 1.1 A+B (page 58)

Binary numbers
- Problem 2.1 A-D (page 73)
- Problem 2.3 (page 74)
- Problem 2.4 (page 75)
- Problem 2.7 (page 85) (extra)

Boolean algebra
- Problem 2.9 (page 89)
- Problem 2.10 (page 90)
- Problem 2.14 (page 93)

Integers
- Problem 2.17 (page 101)
- Problem 2.21 (page 112)
- Problem 2.23 (page 116)
- Problem 2.29 (page 129)
- Problem 2.33 (page 131)
- Problem 2.34 (page 134)

Floating point
- Problem 2.45 (page 147)
- Problem 2.48 (page 155)
- Problem 2.52 (page 158)

Lab 3: Tuesday 24 November, Week 2 (48)

Location: Online
Lecturers: Troels Henriksen

Outline

- Compilers and Interpreters
- Interplay between interpreted and compiled languages

Learning Goals

- Explain the key differences between compiled and intepreted programs
- How interpreted languages can be made fast, despite their overhead
- Understanding performance pitfalls when using interpreted languages

Reading

Course notes: Chapter 1, Pages 3—15:
1.1-4 Compiled and Interpreted Languages
Modern C: Chapter 14, Pages 195—196, Chapter 5, Pages 38—48, Chapter 6, Pages 65—79, Chapter 8, Pages 96—105:
5.1-3 Basic values and data
6.1-4 Derived data types
8.3 Input, output, and file manipulation
14.4 Binary streams

Notes

Basic videos:
- Compiled versus interpreted languages
  https://sid.erda.dk/share_redirect/f8RgVGzlET/compiled-interpreted-tombstone.mp4

Videos on C programming:
- Command line arguments: https://sid.erda.dk/share_redirect/f8RgVGzlET/argv.mp4
- Addresses: https://sid.erda.dk/share_redirect/f8RgVGzlET/addresses.mp4
- IO: https://sid.erda.dk/share_redirect/f8RgVGzlET/io-in-c.mp4
- Makefiles: https://sid.erda.dk/share_redirect/f8RgVGzlET/make.mp4
- Using a Debugger: https://www.youtube.com/watch?v=bWH-nL7v5F4

See lab document here: https://github.com/diku-dk/hpps-e2020-pub/tree/master/material/2-l-1

Lab 4: Thursday 26 November, Week 2 (48)

Location: Lille UP-1, DIKU
Lecturers: Troels Henriksen

Outline

- Memory
- Advanced C programming

Learning Goals

- Describe how dynamic memory and statically allocated memory differs
- Understand how to program with manual memory management
- Describe the differences between the stack and heap
- How to separate interface and implementation in C

Reading

Course notes: Chapter 2, Pages 16—21:
2.1 Data Layout
Modern C: Chapter 12, Pages 150—160, Chapter 13, Pages 160—179:
12. The C memory model
13. Storage

Notes

Videos:
- Linked lists (dynamic allocation): https://sid.erda.dk/share_redirect/AjdMSpVIjr/videos/2-l-2/list-alloc.mp4
- Linked lists (Makefiles): https://sid.erda.dk/share_redirect/AjdMSpVIjr/videos/2-l-2/list-makefile.mp4
- Arrays: https://sid.erda.dk/share_redirect/AjdMSpVIjr/videos/2-l-2/arrays.mp4

See lab document here: https://github.com/diku-dk/hpps-e2020-pub/tree/master/material/2-l-2

Exercise 2: Thursday 26 November, Week 2 (48)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Troels Henriksen, Kenneth Skovhede, and Teaching Assistants

Notes

In this exercise session you will work with theoretical problems similar to those that will appear in the exam.

Tombstone Diagrams

You have access to the following:

- an ARM machine,
- an interpreter for x86 written in ARM
- an interpreter for Python written in x86
- a compiler from F# to C written in Python
- a compiler from C to x86 written in ARM
- a compiler from MATLAB to ARM written in F#

Show Tombstone diagrams for the following cases:

1. How to execute an x86 program.
2. How to execute an F# program.
3. How to execute a MATLAB program.

Pointers

Consider the follow fragments of C programs. For each program, fill in the empty underlined spots such that the final value of the variable x is 0.

1.

```
int x;
int *p;
p = &_____;
*p = _____;
```

2.

```
int x;
int *p;
int *pp;
pp = &_____;
p = &_____;
**pp = _____;
```

3.

```
int x, y;
int *p = &_____;
p = _____;
*p = 1;
p = _____;
*p = 0;
```

4.

```
int x, y;
int* arr[2];
arr[0] = _____;
arr[1] = arr[0];
*(arr[1]) = _____;
*(arr[0]) = arr[0] - 1;
```

Arrays

In this exercise you must rewrite multidimensional array indexes to flat array indexes for arrays of various shapes. For each array and index, provide the flat index for when the array is in row-major order and in column-major order, respectively. Also indicate whether the original index is out-of-bounds, regardless of whether the flat index is.

- Array size: 10 x 20 Element index: (5,11)
- Array size: 20 x 10 Element index: (5,11)
- Array size: 2 x 3 x 4 Element index: (1,2,3)

Inverse

Now you must do the opposite of above, and convert *flat* indexes to the corresponding *nested* index for a given array. Again, indicate whether the flat index is out-of-bounds.

- Array size: 10 x 20 Flat index: 111
- Array size: 2 x 3 Flat index: 6
- Array size: 2 x 3 x 4 Flat index: 5

Lab 5: Tuesday 1 December, Week 3 (49)

Location: DIKU, Zoom
Lecturers: Troels Henriksen

Outline

- Physical memory
- Caches

Learning Goals

- Enumerate memory hierachies
- Explain performance impact of memory hierachies
- Describe the difference between spatial and temporal locality
- Enumerating the different reasons for cache misses
- Understand the cache implications of data layout
- Understanding how to program in a cache efficient way

Reading

Computer Systems: A programmers perspective: Chapter 6, Pages 640—683:

6.2 Locality
6.3 The Memory Hierarchy
6.4 Cache Memories
6.5 Writing Cache-Friendly Code
6.6 Putting It Together: The Impact of Caches on Program Performance

Lab 6: Thursday 3 December, Week 3 (49)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Troels Henriksen

Outline

- Looking at processes
- Understanding operating system concepts
- Understanding the purpose of virtual memory
- Virtual memory from an OS perspective

Learning Goals

- Explain the reason for having an operating system
- Understanding the fork/exec model of Unix
- Understanding of children and zombies
- Understanding how and why processes provide isolation
- Understanding at a high level how virtual memory is implemented in hardware
- Understanding the purpose served by virtual memory in Unix
- Knowing how to program in a VM efficient way

Reading

Computer Systems: A programmers perspective: Chapter 1, Pages 768—792, Chapter 9, Pages 839—860:

1.1 Processes
1.2 System Call Error Handling
1.3 Process Control
9.1 Physical and Virtual Addressing
9.2 Address Spaces
9.3 VM as a Tool for Caching
9.4 VM as a Tool for Memory Management
9.5 VM as a Tool for Memory Protection
9.6 Address Translation

Notes

The book will contain some references to low-level mechanics, such as interrupts and interrupt vectors. These are not terribly important at the level we are going to be working at, so don't worry about those details. Also, we will not focus overly much on process handling (e.g. the whole zombie business). You should focus on the idea of the isolation that processes provide.

Exercise 3: Thursday 3 December, Week 3 (49)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Troels Henriksen, Kenneth Skovhede, and Teaching Assistants

Notes

Getting started

- Something with OS and memory

Lab 7: Tuesday 8 December, Week 4 (50)

Location: DIKU, Zoom
Lecturers: Kenneth Skovhede

Outline

- Physical layer (Wire signals)
- Data link layer (Ethernet / MAC)
- Network layer (IP)
- Transport layer (TCP)
- Application layer(s) (Software)

Learning Goals

- List the layers in the OSI model
- Give examples of signal transfer technologies
- Describe how MAC addresses are assigned and their usage scope
- Explain the difference between a switch and a hub
- Describe how IP adresses are assigned and their usage scope
- Describe the role of ports in IP addresses
- List the states required in the TCP protocol
- Enumerate the key differences between UDP and TCP
- Describe how DNS addresses are assigned and their usage scope
- Enumerate the steps required to resolve a hostname to an IP address with DNS
- List examples of applications using networks

Reading

Computer Systems: A programmers perspective: Chapter 11, Pages 953—968:

11.0 Network Programming
11.1 The Client-Server Programming Model
11.2 Networks
11.3 The Global IP Internet

Lab 8: Thursday 10 December, Week 4 (50)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Kenneth Skovhede

Outline

- The worlds most used protocol: HTTP
- The de-facto HPC protocol: MPI
- The new-way protocol: ZeroMQ
- When there is no server: P2P
- Challenges in a distributed system
- Building applications on a network
- Introduction to assignment #3

Learning Goals

- Explain the client/server model
- Describe various high-level network approaches
- Explain tradeoffs between network approaches
- Sketch a basic P2P system
- Describe relation between latency and bandwidth
- Enumerate fundamental problems and solutions in a distributed system

Reading

Computer Systems: A programmers perspective: Chapter 11, Pages 968—1000:

11.4 The Sockets Interface
11.5 Web Servers
11.6 Putting It Together: The TINY Web Server

Exercise 4: Thursday 10 December, Week 4 (50)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Kenneth Skovhede, Troels Henriksen, and Teaching Assistants

Notes

TBD - Assignments related to networks

Lab 9: Tuesday 15 December, Week 5 (51)

Location: DIKU, Zoom
Lecturers: Troels Henriksen

Outline

- Concurrency
- Parallel execution
- POSIX threads
- Semaphores as synchronisation mechanism
- Concurrency abstractions

Learning Goals

- Describe how threads and processes differ
- Enumerate challenges with concurrent programming
- Understand the need for load balancing
- Understand the overhead of multi-threading
- Practical POSIX threads programming
- Building a work-queue abstraction

Reading

Computer Systems: A programmers perspective: Chapter 12, Pages 1009—1056:

12.1 Concurrent Programming with Processes
12.2 Concurrent Programming with I/O Multiplexing
12.3 Concurrent Programming with Threads
12.4 Shared Variables in Threaded Programs
12.5 Synchronizing Threads with Semaphores
12.6 Using Threads for Parallelism

Notes

CSAPP 12.1-12.6

Lab 10: Thursday 17 December, Week 5 (51)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Troels Henriksen

Outline

- GPGPUs as computational accellerators

Learning Goals

- TBD

Exercise 5: Thursday 17 December, Week 5 (51)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Kenneth Skovhede, Troels Henriksen, and Teaching Assistants

Notes

GPGPU exercises

Lab 11: Tuesday 5 January, Week 6 (1)

Location: DIKU, Zoom
Lecturers: Troels Henriksen and Kenneth Skovhede

Notes

This lab session is reserved for any subjects that needs further exploration.

Lecture 12: Thursday 7 January, Week 6 (1)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Kenneth Skovhede and Carl-Johannes Johnsen

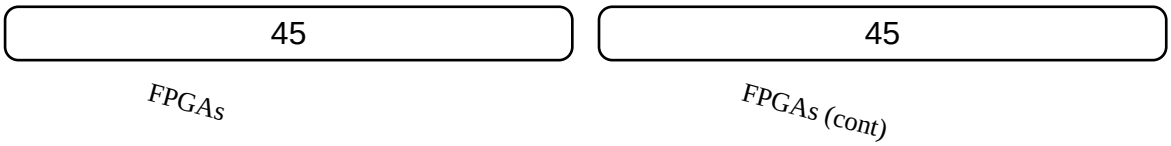
Outline

- What are FPGAs?
- When to use FPGAs, and when NOT to use FPGAs
- Programming models for FPGAs
- Tools for FPGAs
- Next-gen FPGA hybrid devices
- Examples and current research
- Introduction to assignment #4

Learning Goals

- Give examples of applications that fit CPU, GPGPU and FPGA respectively
- Explain how FPGAs are constructed
- Describe FPGA properties and limitations
- Explain how a programming language can be used to describe a circuit
- Enumerate the different approaches to programming FPGAs and their properties

Plan



Exercise 6: Thursday 7 January, Week 6 (1)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Kenneth Skovhede, Troels Henriksen, and Teaching Assistants

Notes

Content TBD

Lab 13: Tuesday 12 January, Week 7 (2)

Location: DIKU, Zoom
Lecturers: Kenneth Skovhede

Outline

- TBD

Learning Goals

- TBD

Lecture 14: Thursday 14 January, Week 7 (2)

Location: DIKU, Lille UP1 + Zoom
Lecturers: Kenneth Skovhede and Markus Jochum

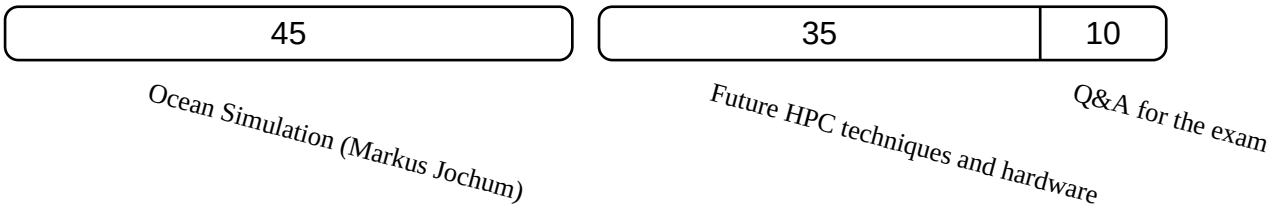
Outline

- Guest Lecture from Markus Jochum - TeamOcean
- Worlds fastest computers: Fugaku, Summit, Sierra, TaihuLight, Tianhe
- Smart-storage
- In-network processing
- Programming a million cores

Learning Goals

- Reflect on large scale simulations
- Enumerate properties of large-scale systems
- Describe metrics for comparing large-scale systems
- Enumerate emerging technologies for HPC

Plan



Notes

This session is *not* a lab session, it is a traditional lecture!

The lecture will be held in-person in Lille UP1 if permitted, but available on zoom regardless.
There is no reading required for attending this lecture.

Exercise 7: Thursday 14 January, Week 7 (2)

Location: DIKU, 1-0-30 + 1-0-34 + 1-0-14 + A111
Supervisors: Kenneth Skovhede, Troels Henriksen, and Teaching Assistants

Notes

TBD

Assignment 1 — Numeric representations in CPUs

Set: Thursday 19 November
Due: Friday 27 November

Guesstimated Time Breakdown

1h	Familiarisation
2h	Finding and reading relevant documentation
2h	Learning to write C code
4h	Implementation and testing
4h	Writing report
Total: 13	hours

Learning Goals

- Writing small C programs
- Learning numeric representations
- Reflecting on bounds of numeric values in a program

Notes

In this first assignment you will be constructing a simple floating point emulation library, which could be used on a CPUs that lacks a floating point unit.

Assignment 2 — Data layouts and dynamic allocation

Set: Monday 30 November
Due: Wednesday 9 December

Guesstimated Time Breakdown

1h	Familiarisation
4h	Implementation and testing of brute-force method
5h	Implementation and testing of k-d tree method
5h	Writing report
Total: 15	hours

Learning Goals

- Understand how to read and write structured data files
- Describing the memory layout of multidimensional arrays
- Develop and debug programs that use dynamic allocation
- Learning how to develop modular C programs
- Reasoning about performance implications of preprocessing

Notes

In the second assignment you will be implementing the k-nearest-neighbours algorithm in C.

Assignment 3 — Building a simple HTTP compliant Web Server

Set: Thursday 10 December
Due: Friday 18 December

Guesstimated Time Breakdown

1h	Familiarisation
2h	Finding and reading relevant RFCs and documentation
8h	Implementation
5h	Writing report
Total: 16	hours

Learning Goals

- Find, read, and understand material used for defining and describing Internet protocols, including RFCs
- Use and understand the socket library for a server
- Be able to implement a simple protocol directly using sockets
- Describe how a URL can map to a file system

Notes

In this third assignment you will be writing a standards compliant webserver that can host static content.

Assignment 4 — Parallelizing existing code

Set: Thursday 7 January
Due: Friday 15 January

Guesstimated Time Breakdown

1h	Familiarisation
2h	Finding and reading relevant documentation
8h	Implementation
5h	Writing report
Total: 16	hours

Learning Goals

- Identify potential for parallelization in existing code
- Measure and evaluate parallelization optimizations
- Implement programs with concurrency

Notes

In this fourth assignment you will be parallelizing existing code to run on a HPC system.

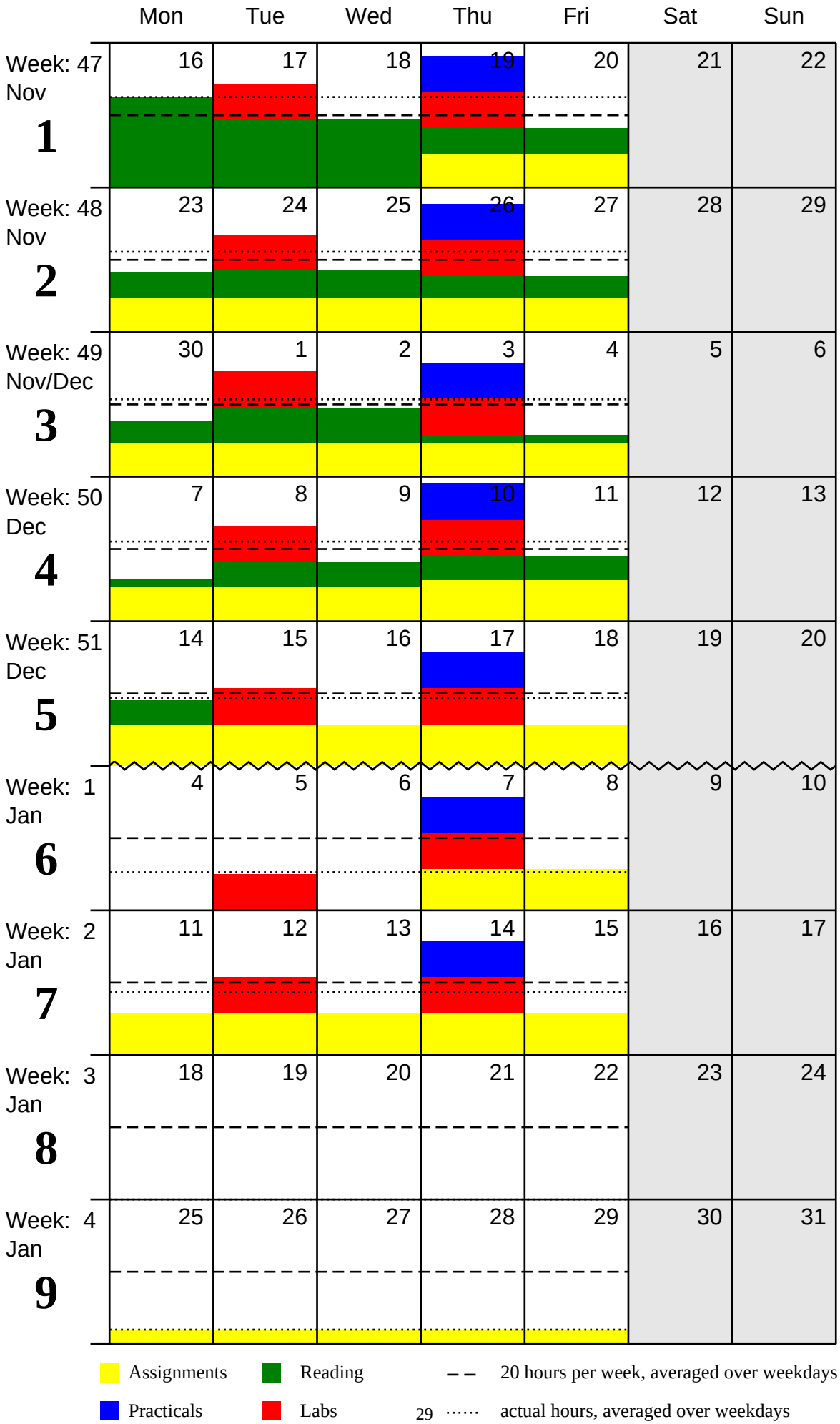
Exam

Set: Monday 25 January
Due: Friday 29 January

Guesstimated Time Breakdown

	4h	Written exam
Total:	4	hours

Guesstimated Work Pressure



Using this Document

We have made this document to help use (the lecturers) prepare the course. We have also, however, made it as a tool to communicate to you (the students) what we expect you to get out of the course, the individual lectures, practical session, and assignment. Thus, we have presented the overview of the content covered in each lecture and practical session, together with a number of learning goals. At the end of a lecture or practical session you can use these learning goals to take stock of what you have learnt and what you need to work on. Do not expect to have the learning goals covered by just going to the lectures without doing any auxiliary work, such as reading other material.

Updates

We will from time to time make updates to this document. You should check regularly on Absalon to see if you have the latest version. The version of the document can be seen on the front page of this PDF.

The Assignments

We have made some guesses as to how long the assignments as a whole should take and provided an estimate of how you may want to consider dividing your time amongst the individual parts of an assignment. These are *guesstimates* and should be taken as a *guide* only.

The Guesstimated Work Pressure

The guesstimated work pressure is divined by taking the time you are going to spend attending lectures and practical sessions, working on your assignments, and reading the set material in the book every week (with a generous amount of time allocated to read a page). Work on the assignments and required reading has been split evenly over the days from when it was set to when it is due. The assumption made in the diagram is that you work a five day week and do not work on the weekends. This may or may not reflect reality and you may of course work however and whenever you like.

You should note that the title of the diagram contains the word *guesstimate*, which a dictionary (*The New Oxford American Dictionary, Second Edition*) defines as: “an estimate based on a mixture of guesswork and calculation.” This is exactly what it is, an estimate of how much time we think you need to spend on the course, it is not a prescription! You may use more time or less, this is fine. Thus, the *Guesstimated Work Pressure* is there for you to be able to visualise the workload in the course and thereby help you arrange your work schedule. Please don’t throw a fit if my guesstimates turn out to be somewhat inaccurate, but do let us know so we can adjust them in future iterations of the course.

Providing Feedback

At the end of the course we will be asking for your feedback. We would like to know how useful this document was to you and how we could improve it. As already mentioned, we will be asking you how much time you spent on different parts of the course to see how they match up with our expectations as to how long I think they should take you. It would therefore be useful if you can keep (rough) notes on the time you spend, for example on reading and your assignments.