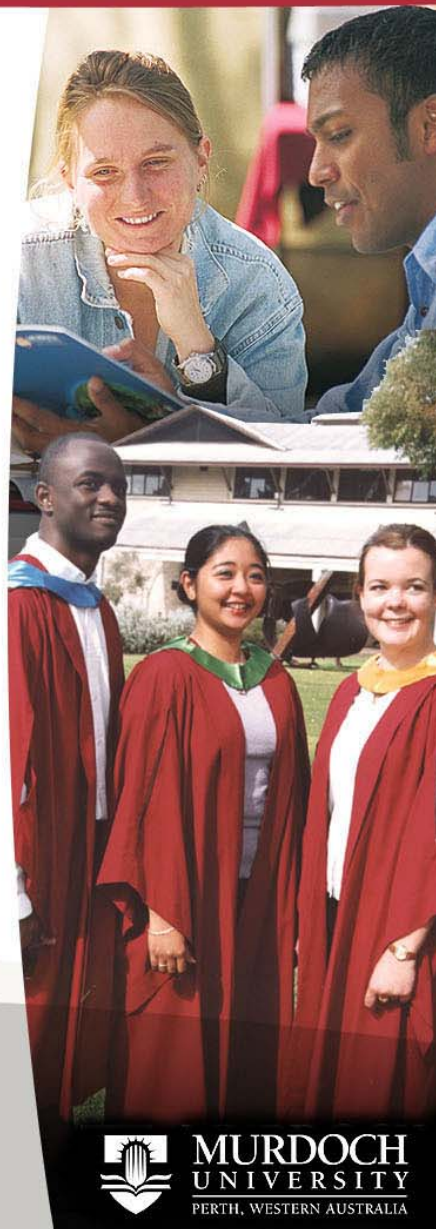
murdochuniversity

ICT 106 Fundamentals of Computer Systems

Associate Professor Lance C.C. Fung
Tel: (08) 9360 7586 / 7507
Office: ECL Building, Room 3.042
Transportable 1, Room 1.06
email: L.Fung@murdoch.edu.au

school of information technology



Location and Time

- South Street – ECL4
- Time: Friday 11:30AM – 14:30AM
- Practical: PS 1.16F (Wednesday 4 sessions)
- ECL 2.046 (Friday 2 sessions)

- Rockingham – Prof Development Room
- Time: Thursday 9:00AM – 12:00AM
- Practical: ACL 109 (1 Session)

Tutors and Lecturers

- Lecturer:
  - South Street – Associate Professor Lance Fung
  - Rockingham – Lance Fung and Eric Li
- Tutors:
  - South Street
    - » Steven Van Der Werf and Eric Li (Wed),
    - » James Mei (Fri)
  - Rockingham
    - » Eric Li (Thur)
  - External
    - » Kien Ping Chung

Content

- Unit Outline
- Practical for Week 2 and 3
- Project
- Lecture 1

What IT jobs are available?
Where are they?


SEEK - Australia's no. 1 jobs, employment, career and recruitment site - Microsoft Internet Explorer
File Edit View Favorites Tools
Address http://www.seek.com.au/
Go
Back
Search
Favorites
Search Web
Mail
My Yahoo!
Answers
Games
Local
Music
My Web
mywebsearch
Upgrade
Search
Screensavers
Smiley Central
Cursor Mania
My Info
Mail Stamp
Mail Stationery
make luck happen
seek
AUSTRALIA'S #1 JOB SITE
131,992 JOBS ONLINE
Welcome. Login or Register now!
SEEK HOME
EXECUTIVE JOBS $80K+
I.T. JOBS
U.K. JOBS
ADVERTISERS
Post a Job Ad
From $150 +GST*
My Account | Job Search | Search by Recruiter | Search by Company | Career Resources | Training
QUICK SEARCH
ADVANCED
Keywords
Search Tips
IT
Postcode Search
Any Location
Any Area
Any Classification
Any Sub-Classification
SEARCH
Email me jobs
Set up a Job Mail and let matching jobs come to you.
Get qualified
Enrol in a uni or TAFE course, get IT certified or upgrade your computer skills at SEEK Learning.
Login
Jobseeker
Advertiser
Username:
Password:
Remember me on this computer
Register now!
Forgotten password
LOGIN
Browse Job Ads
Accounting (14,095)
Administration (10,549)
Advert./Media/Entertain. (1,650)
Banking & Fin. Services (7,123)
Call Centre/Cust. Service (4,767)
Community & Sport (639)
Construction (4,911)
Consulting & Corp. Strategy (525)
Education & Training (1,990)
Engineering (8,078)
Insurance & Superannuation (1,604)
I.T. & T (19,037)
Legal (4,682)
Manufacturing/Operations (2,655)
Mining, Oil & Gas (2,541)
Primary Industry (149)
Real Estate & Property (1,590)
Retail & Consumer Prods. (5,866)
Sales & Marketing (11,351)
Science & Technology (777)
My Last Search
Your most recent search will be displayed here, so you can search again quickly.
seek
Find roles over $80k that
Internet
Start
Labs-S2-2006-v3
timetable Sem ...
Murdoch Univ...
SEEK - Austr...
C:\doc 2005\I...
Microsoft Pow...
3:06 PM

Out of 131,992 advertised, 19,037 are IT & T (about 15%)

Browse Job Ads

- Accounting (14,095)
- Administration (10,549)
- Advert./Media/Entertain. (1,650)
- Banking & Fin. Services (7,123)
- Call Centre/Cust. Service (4,767)
- Community & Sport (639)
- Construction (4,911)
- Consulting & Corp. Strategy (525)
- **Education & Training** (1,990)
- Engineering (8,078)
- **Government/Defence** (1,392)
- Graduate/Entry Level (2,000+)
- **Healthcare & Medical** (6,181)
- Hospitality & Tourism (4,816)
- HR & Recruitment (4,738)
- Insurance & Superannuation (1,604)
- I.T. & T (19,037)
- Legal (4,682)
- Manufacturing/Operations (2,655)
- Mining, Oil & Gas (2,541)
- Primary Industry (149)
- Real Estate & Property (1,590)
- Retail & Consumer Prods. (5,866)
- Sales & Marketing (11,351)
- Science & Technology (777)
- Self-Employment (254)
- Trades & Services (4,051)
- Transport & Logistics (2,965)
- Volunteer (8,000+)

What jobs are they?

Computing Curricula for the New Decade

- A paper presented in the High Education for 21st Century Conference (September 2002)
- Based on 2001 Computing Curriculum prepared by *The Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM)* Task Force.

Changes in the Computing Discipline

- Technical Changes:
  - The World Wide Web and its applications
  - Networking technologies, particularly those based on TCP/IP
  - Graphics and multimedia
  - Embedded systems
  - Relational databases
  - Interoperability
  - Object-oriented programming
  - The use of sophisticated application programmer interfaces (APIs)
  - Human-computer interaction
  - Software safety
  - Security and cryptography
  - Application domains

Changes in the Computing Discipline

- **Cultural Changes**
  - ***Changes enabled by new technologies.*** Examples are networking technology.
  - ***Growth of computing world-wide.*** The students are increasingly familiar with the technology and various applications.
  - ***Economic influence of computing technology.*** Vast amount of resources and expectation have been built around the high-tech and computer industries.
  - ***Acceptance of computer science.*** Due to the acceptance and dependency on computing and information technology in the business and economic arenas, computing discipline is now one of the most popular and dynamic fields.
  - ***Broadening of the discipline.*** With the inclusion of many new applications, computing has grown to a much larger and encompassing discipline.

BODY OF KNOWLEDGE

- Discrete Structures (DS)
- **Programming Fundamentals (PF)**
- **Algorithms** and Complexity (AL)
- **Architecture and Organization (AR)**
- **Operating Systems (OS)**
- Net-Centric Computing (NC)
- **Programming Languages (PL)**
- Human-Computer Interaction (HC)
- Graphics and Visual Computing (GV)
- Intelligent Systems (IS)
- Information Management (IM)
- Social and Professional Issues (SP)
- Software Engineering (SE)
- Computational Science and Numerical Methods (CN)

Professional Practices

- Communication skills (verbal and written)
- Honesty/integrity
- Teamwork skills
- Interpersonal skills
- Motivation/initiative
- Strong work ethic
- Analytical skills
- Flexibility/adaptability
- Computer skills
- Self-confidence

"Characteristics of Computing Graduates"

- *A high-level understanding and appreciation with system-level perspective*.
- *An appreciation of the interaction and linkage between theory and practice*.
- *Familiarity with common themes* such as abstraction, complexity, and evolutionary change which have broad applications in the field of computing.
- *Experience in project development*.
- *Demonstration of adaptability*.

Capabilities and Skills

- ***Cognitive capabilities and skills*** – modeling, requirements, critical evaluation and testing, methods and tools, and, professional responsibility.
- ***Practical capabilities and skills*** - design and implementation, evaluation, information management, human-computer interaction, risk assessment, tools and operation.
- ***Additional transferable skills*** – communication, teamwork, numeracy skills, self management and professional development.

Challenge

- Rapid change in technology and social expectations
- Global competition
- Shortened product development time
- Shortened product life span
- Customer-oriented applications
- Practical and efficient products
- **How can ICT106 help?**

- **Unit Web site:**
  http://online.murdoch.edu.au/public/ICT106/

  The above website contains information from the Handbook, Study Support, Assessment policies and grading.

- **Unit ftp site:**
  http://ftp.it.murdoch.edu.au/pub/units/ICT106/

  The unit ftp site will be the repository of unit materials. Definitive unit details, announcements facility and tutor contact are provided.

Administrative Contacts

- If you have any queries about your **enrolment** in this unit please contact:
- The Division of Arts student administration office:
  Tel: (08) 9360 2420
  Fax: (08) 9310 6958
  Email: arts@murdoch.edu.au

Aims & Objectives

- The aims of this unit are to provide
  - an introduction to the low level machine aspects of a computer system
  - a framework with which to appreciate how a computer works at the machine level
  - an introduction to programming in C and assembly language
  - a basis for understanding how high level languages utilize machine resources
  - an introduction to the machine level aspects of an operating system

At the end of this unit you should:

- be able to program in the high level language C
- gain an understanding of assembly language programming
- be familiar with the relationships between C and assembly language
- gain some understanding of the organisation of 80X86 based computers
- understand some of the low level aspects of an operating system

Unit Organisation

- The unit material is divided into 12 weeks of lectures and associated practical exercises.
- For the internal students, there are 3 hours of lectures and a two-hour practice session each week. The practice sessions are used to exercise and reinforce knowledge of the unit material and are a vital part of the learning activities of the unit.
- External students need to work through the exercises as well. You will also need to spend time on reading and complete the practical work outside the scheduled hours in order to fully understand the unit material.
- It will help your understanding of the material if you read the relevant text book chapter and other related materials in conjunction with the notes for each lecture topic.

Proposed Schedule of topics

- Introduction to Computer System architecture and C programming language
- Information Representation and Storage. Data Input and Output in C
- Arrays, functions and structures in C, Using the Debugger
- Introduction to 80x86 Microprocessors. Bit manipulation and Pointers in C
- Machine language and Introduction to the 80x86 Instruction set
- Assembly Language Fundamentals: Generating, Loading and Executing programs

Proposed Schedule of topics

- Assembly Language Programming Development. Addressing modes. Using the assembler
- High-Level Language interface - Inline assembly in C
- Procedures and Functions. Calling and Exit Conventions, Run-time Stack HLL Control Constructs and Integer arithmetic at Assembly Level
- Concepts of Input, Output and Interrupts
- Introduction to Operating Systems. Memory Management.

Change of Enrolment

- If you are maintaining a steady rate of progress through the unit, then by the end of week 5 you will have a good idea as to your ability to satisfactorily complete the unit. At this stage in the unit, if you have not satisfactorily completed most of the work required for the first three topics, you should discuss your progress with your Tutor or the unit coordinator.
- To make any change of enrolment, submit a Change of Enrolment form to the Division Executive Officer at the Division of Arts. The deadline for such a change is **HECS census date, which for second semester units is August 31**.

Important Deadlines

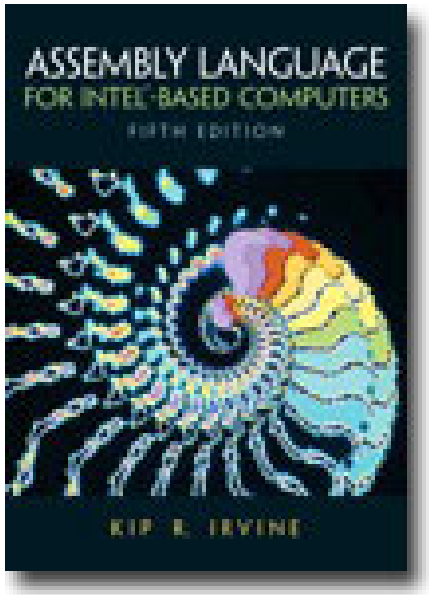
- Students should be aware of the implications of different dates of withdrawal from the unit.
- **Enrolment - end of Week 3**:
  - Does not appear on academic record
- **Week 4 - end of Week 10**
  - Appears as "Withdrawn" on academic record
- **After Week 10**:
  - Appears as "Fail" on academic record.
- **After 31st August in Semester 2:**
  - Incurs HECS liability

Unit Materials and Computing Facilities

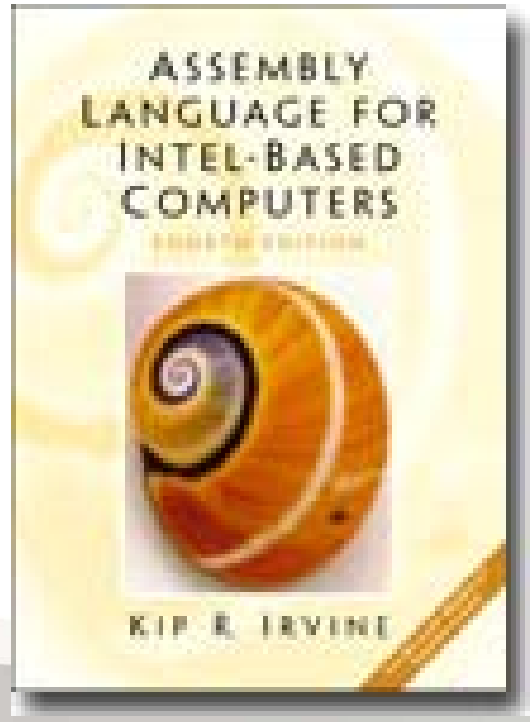
- You will be expected to keep up to date with reading
  - **the textbook,**
  - **additional readings/hand-outs given out during the semester**
  - **lecture notes,**
  - **weekly practice sheets**
- The unit material will be available progressively on the **unit's ftp site**.
- **You must visit the ftp site regularly at**

http://ftp.it.murdoch.edu.au/pub/units/ICT106/

Unit Textbook



Irvine, Kip R., *Assembly Language for Intel-Based Computers*, Fifth Edition, Prentice-Hall (2007)



Irvine, Kip R., *Assembly Language for Intel-Based Computers*, Fourth Edition, Prentice-Hall (2003)

Textbook

- Irvne's book is used as the unit text as it contains all the basic and fundamental topics on assembly language, processor architecture and introduction to operating system including 16-bit DOS and 32-bit Windows. This is a good reference text on the topic if you continue your exploration on the low-level aspects of the computer. However, the 4th Edition is no longer in print and the 5th Edition is supposed to be available in Mid-June. While there are some improvement on 5th Edition, the 4th Edition will be sufficient for the purpose of this unit.

Other References

- Deitel H.M. and Deitel P.J., *C How to Program*, Third Edition, Prentice Hall (2001)

  (This is a useful reference as it also includes C++ and Java. All these languages are used in the course. In addition, the book also includes a CD which contains the Borland C++ 5.5 Compiler for Windows and J Builder Foundation for Windows, Linux and Solaris.)
- Collopy D. *Introduction to C programming* – a Modular Approach, Second Edition, Prentice Hall (2003)

  (This book uses the traditional modular approach to C program development. It contains a CD with a special edition of Microsoft Visual C++ compiler.)

References

- Prata, Stephen, *The Waite Group's C Primer Plus*, Third Edition, Sams Publishing (1999)
- Bronson Gary J., *A First Book of ANSI C*, 3rd edition, Brookes/Cole Thompson Learning (2001)
- Kernighan Brian W. & Ritchie Dennis M., *The C Programming Language*, Second Edition, Prentice-Hall (1988)
- Abel P. *IBM PC Assembly Language and Programming*, Fifth Edition, Prentice Hall (2001)

References

- Mazidi M A & Mazidi J G, *The 80x86 IBM PC and Compatible Computers Assembly Language, Design, and Interfacing*, Second Edition, Prentice-Hall International (1998)
- Tanenbaum, Andrew S., *Structured Computer Organization*, Fourth Edition, Prentice-Hall International (1999)
- Matloff, Norman S., *IBM Microcomputer Architecture and Assembly Language: A Look Under the Hood*, First Edition, Prentice-Hall (1992).

Compiler

- A number of compilers can be used in this unit:
- GCC (to be operated in Cygwin environment)
- Microsoft Visual C
- Borland C++ (version 5) for Windows
- HI-TECH Software's Pacific C which can be downloaded for free from the following site:

  http://www.hitech.com.au/products/pacific.html
- Pacific C is a 16-bit compiler for MS-DOS systems and comes with an IDE that allows editing and debugging of source code. It also allows inline assembly within C.
- Macro Assembler (MASM)
- Turbo Assembler (TASM) with turbo linker and debugger

Assembler

- Microsoft Macro Assembler (MASM) with linker and debugger. *Note that MASM 6.11 and 6.15, and the Code View debugger are included on the CD that comes with the unit textbook (4th Edition).*

Required Course-work and Timetable

- **Workload: This unit represents about one quarter of a full-time load, or about 10 hours per week. Assignments and Regular Practice Work**
- **Continuous assessment for the Unit will consist of *weekly practice work (to be submitted as 3 exercises) and a project*.**
- **The Practice work will be assessed on an on-going basis. All students will be required to keep a folder of Practice Work and submit it to their tutor at the required submission dates for checking and feedback.**

Practical Assessment

Assessment	% mark	Week Due
C Exercise 1	6%	Week 5
Project Abstract	5%	Week 5
C Exercise 2	8%	Week 9
C Project	15%	Week 11
Assembly Exercise	6%	Week 12
Assembly Program	10%	Week 13

Assessment

- This unit will be assessed by *regular practice work, assignments, and a final examination* with the following weights:
  - C Exercises and Project **34%**
  - Assembly Exercises and Program **16%**
  - Final Examination **50%**
- Final Examination: At the end of the unit a three-hour examination will be held, based on all the material of the unit.

Final Grade

In order to pass this unit, you must satisfy the following conditions:

- **achieve an overall aggregate score of 50% or higher for all of the combined assessments (Continuous Assessment and Final Examination);**

and

- achieve a satisfactory performance in the final examination. A satisfactory performance is normally considered to be 45% or higher;

and

- achieve a satisfactory performance in the continuous assessment (C Exercises, C Project, Assembly Language Exercises and Assembly Language Program). A satisfactory performance is normally considered to be 45% or higher.

Letter	Grade	Notional Minimum Percentage Scores
HD	High Distinction	80 - 100
D	Distinction	70 - 79
C	Credit	60 - 69
P	Pass	50 - 59
S	Supplementary Assessment	45 - 49*
N	Fail	Below 50
DNS	Fail and did not submit assignment after census date	

University Policy on Assessment

http://www.murdoch.edu.au/vco/secretariat/admin/codes/assessapp2.html

Assessment roles and responsibilities

http://www.murdoch.edu.au/vco/secretariat/admin/codes/assessappl.html

Honesty in assessment and avoiding plagiarism

http://www.murdoch.edu.au/admin/discipline/

Plagiarism and Collusion

http://www.murdoch.edu.au/vco/secretariat/admin/codes/assessapp2.html#dishonesty

Non-Discriminatory Language

http://www.murdoch.edu.au/teach/studyat/non_disc.html

Teaching Week Table

Week (Monday)	Semester 2, 2006
1	24 Jul
2	31 Jul
3	7 Aug
4	14 Aug
Non-teaching Week : 21 Aug	
5	28 Aug (Work Due)
6	4 Sept

Teaching Week Table

Week Number	Semester 2, 2005
7	11 Sept
8	18 Sept
Non teaching Week: 25 Sept	
9	2 Oct (Work Due)
10	9 Oct
11	16 Oct (Work Due)
12	23 Oct (Work Due)
13	30 Oct (Work Due)

Advice on How to Study this Unit

- The unit is organised around 12 Topics. Each topic has the same basic structure. For each topic, you should:

1. Do the required **Reading.** Find additional resources if necessary.
2. Understand the material in the **Objectives** section
3. Download the practical work sheet from the unit ftp site. Attempt and work through them.
4. When required, demonstrate exercises/programs (External students - submit them to your tutor)

C Programming Project

- **All students** (both internal and external) must submit an Abstract and a Final Report on the project.
- Presentation of the submission must be according to that described in the Project Sheet.
- You will be given a letter grade for the submission.

Contact Your Tutor

- You should not spend an unreasonably long time on one problem. Go on to something else and get in touch with your tutor for some help. Help will be readily available provided you demonstrate to your tutor that you have made a reasonable effort.
- Whether an internal or external student, it always pays to prepare a list of questions you wish to ask and perhaps some of the work done thus far.
- When you telephone (or fax or email) your tutor, you should have your Study Guide, Unit Notes and the list of questions with you, so that you do not waste time (and money) searching for information during your conversation with the tutor.

Study Pace

- The required work is exactly the same for both internal and external students. External students will proceed at their own pace through the unit, using the unit organisation table section and the submission dates as a guide to maintaining an adequate rate of progress through the unit. Internal students have the advantage of lectures and tutorials to pace themselves through the unit. External students should access the lecture notes from the ftp site.

- As you study and work through the unit, remember to balance the **two complementary aspects**: theory versus actually using the computer. Too often, students are more concerned with using the computer than understanding some of the fundamental concepts.

Time allocation

- Plan for a **time allocation** of at least 10 hours per week to your study of this unit. It is best if you maintain a constant, steady rate of progress through the unit.
- Note that this is not a unit in which all of your study can be left to the last moment.
- Additionally, attempt each of the practical exercises at the end of the topics immediately after working through that topic.
- **It may be too late to attempt the work immediately prior to when the assignment is due.**

Labs and Tutors

- Venue and time
  - South Street: PS 1.16F
    - Steven Van Der Werf Wed 9:30-11:30, 11:30-13:30
    - Eric Li Wed 13:30-15:30, 15:30-17:30
  - South Street: ECL 2.046
    - James Mei Fri 8:30 – 10:30, 14:30-16:30
  - Rockingham: ACL109
    - Eric Li Thur 13:00 – 15:00

Changes from 2005

- **Introduction to Unix/Linux**
  - Lab and lecture materials will include and made reference to Unix/Linux. Cygwin will be used as a simulated environment to introduce the Unix/Linux OS and C programming.
- **Assessment**
  - Reduction of one C Exercise. However, the students are still expected to work on the lab EVERY WEEK
- **Weighting for continuous assessment**
  - The weighting between the practical components was adjusted slightly: C - 34% and Assembly Language – 16%

Changes from 2005

- **Passing the Unit**
  - A student must meet satisfactory performance in BOTH Continuous Assessment and Final Examination. Minimum 45% is required in each component and the total score must be at least 50%.
  - In other words, if one fails any component badly (less than 45%) and even the total is above 50%, he/she can still fail the whole unit.
- **Memory Aid**
  - you will not be permitted to take in any calculators. However, you are permitted to bring in ONE PAGE (DOUBLE-SIDED) HAND-WRITTEN NOTES to the Final Examination.

LECTURE 1

- To provide an introduction to computer systems architecture
- To present and revise the fundamental aspects of C Programming Language

Why Look Under the Hood of a Computer System?

- An analogy: car vs computer
  - A look under the hood (bonnet) of a car gives us a view of the engine – the source of its power
- A computer has two sources of power:
  - Hardware including the CPU that executes the computer's machine language
  - Low level software consisting of various services that the operating system makes available to programs

- The analogue of driving a car is programming in a high-level language (HLL)
- Compilation of a HLL program involves translation into machine language & calls to routines of the operating system
- A Computer System consists of Hardware & Software components - both are of equal importance.

Hardware Components

- **Processor** (CPU) – the 'brain' of the computer
- **Main memory** – for storing programs, data and intermediate results while the programs are executed. Memory is divided into blocks of bits called words. Each word has a unique address.
- **Peripherals** – external devices (eg, disks, keyboard, mouse, printer etc.)
- **Bus** – for communication between the different components

Main Components of the Processor

- **Control Unit** – for decoding and executing program intsructions
- **ALU** – for executing all arithmetic and Boolean operations
- **Registers and Accumulators** – internal memory units for storing intermediate results

Main parts of a typical Bus

- **Address Bus** – for sending the address from where to fetch data from memory
- **Data Bus** – for exchanging data between the memory, the processor and the peripherals
- **Status Bus** – for exchanging information about the status of the data and the system
- **Control Bus** – for exchanging control information between the processor and the peripherals (eg, interrupts)

Clock

- Clock – a global clock signal for synchronizing all activities (eg, 2GHz which means at most 2000 million instructions can be executed per second)
- Note:
  - 2000MHZ is 2000 million pulses (clock cycles) per second, which means
  - One clock cycle takes $1/(2000 \times 10^6)$ secs
  - i.e., 0.5 nanoseconds

Software Components

- In order to “drive” the hardware to solve problems, we need systems programs - operating system, assembler, compiler, software development environments etc.
- Therefore, the 3 aspects of a complete computer system are:
  - The hardware
  - The system ‘driving’ programs
  - Application programs

Do we need to know what's under the hood?

- From computing perspective: Do we need to understand assembly language and fundamental components of an operating system?
- It can help in one's understanding of **good programming**
- It can provide increased **appreciation of** why certain **errors, bugs, problems**, ... can/do occur when programming or with software products

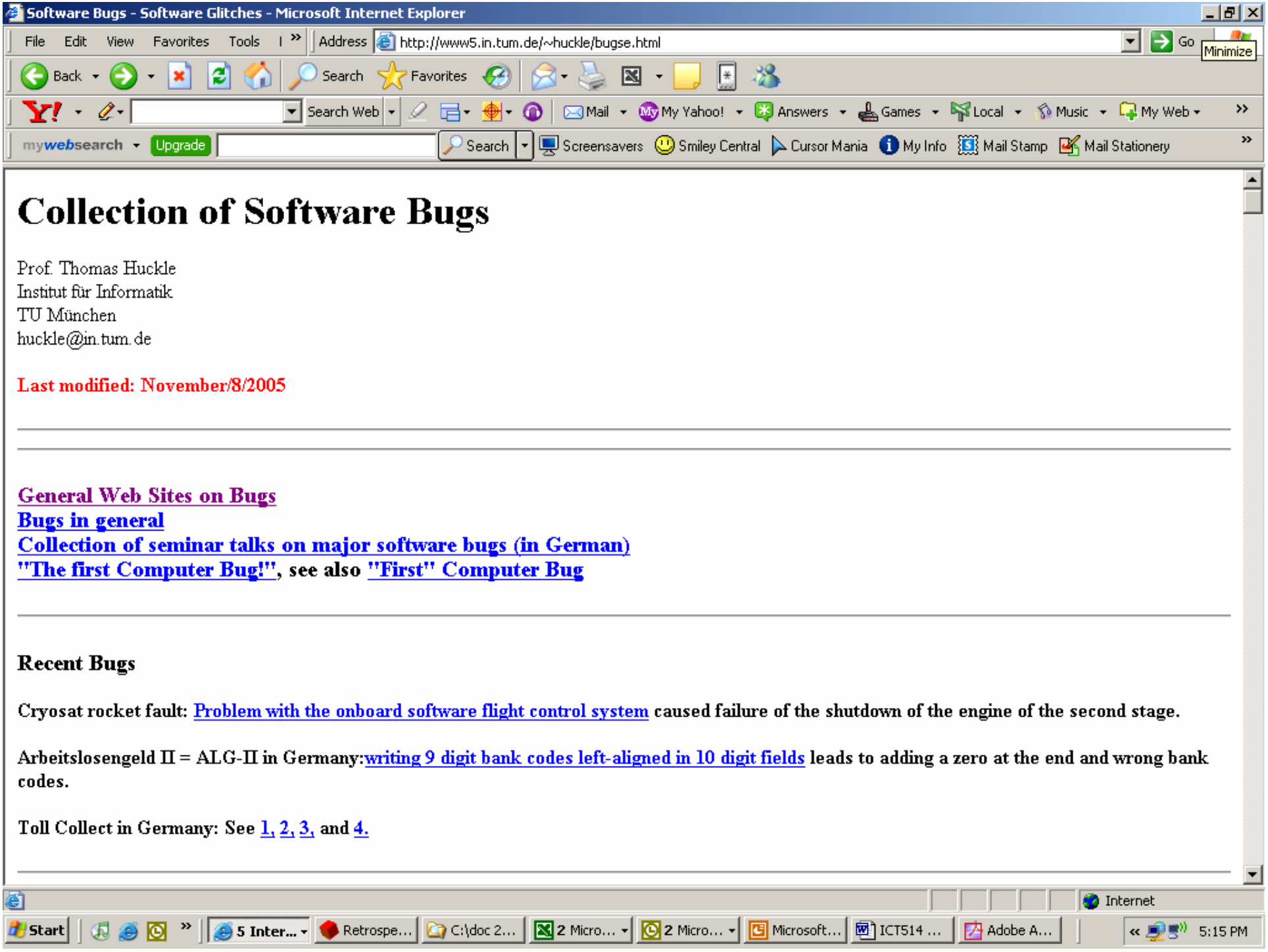
An Example

Typical example:

```
x = 28502;
y = 12344;
sum = x + y;
```

==> sum has the value -24689

Why is this negative & incorrect?



Collection of Software Bugs

Prof. Thomas Huckle
Institut für Informatik
TU München
huckle@in.tum.de

Last modified: November/8/2005

General Web Sites on Bugs
Bugs in general
Collection of seminar talks on major software bugs (in German)
"The first Computer Bug!", see also "First" Computer Bug

Recent Bugs

Cryosat rocket fault: Problem with the onboard software flight control system caused failure of the shutdown of the engine of the second stage.

Arbeitslosengeld II = ALG-II in Germany:writing 9 digit bank codes left-aligned in 10 digit fields leads to adding a zero at the end and wrong bank codes.

Toll Collect in Germany: See 1, 2, 3, and 4.

Do we need to know what's under the hood?

- Can provide increased **appreciation of** the power, limitations, interactions, …. Of **operating systems**
- Can provide a general background or foundation to better **appreciate how/why various constructs in a HLL and operating systems work**

High Level Languages/Low level Languages

- There are many HLLs: Fortran, Cobol, Pascal, Smalltalk, Lisp, Prolog, Modula, C, Ada, Visual Basic, C++, Java, …
- High-level languages resemble human languages in many ways - designed to be easy for humans to write programs in and easy for humans to read
- A ***machine language*** is a language in which the instructions are in a form that can be performed immediately by a computer without any further translation being required (machine code/binary code – 0's and 1's)

- A program that translates a high-level language program (like C/C++) to a machine language program is called a ***compiler*** (a *source* program to an *object* program). Eg, C Compiler - allows programs written in the C language to be translated and executed on the computer
- An ***assembly language*** is designed for a specific computer and provides mnemonics for each machine language instruction.
- An ***assembler*** is a program that translates assembly language program into machine language

Review of High Level Languages

- Different classes of languages: procedural, functional, logic, object oriented
- We will concentrate on procedural (block structured) languages – they have many similar underlying structures

Things in common in HLL's

- **Major Divisions** within any program:
  - Data description section
  - Algorithm section

Data Types

- Simple (primitive) types
- create (define) variables
- associated operations
- typical examples: integer, real, boolean, character, string

Structured data types

- compound type made of simpler or other types
- grouping mechanisms for defining structures
- methods for defining variables
- two or more data cells or elements
- typical mechanisms: arrays, records, files, pointers

Simple data representations

- literals - explicit description of data
- constants - don’t change for lifetime of program
- variables - data changes under direction of program
- last two: mnemonic tools for representing concepts/data

Executable Statements or Actions

- Moving data
- Manipulating or changing data
- Control constructs – control program flow

Moving data (internal vs external)

- internal to internal
  - (assignment: variable = expression)
- how about parameter passing (by value)?
- external to internal
  - (read: item1, item2,...)
- internal to external
  - (write: item1, item2,...)
- external to external: NONE!
- How about copying one file to another?

Manipulating Data

- **expressions** (operands and operations) relevant to types involved (arithmetic, character, string, logical manipulations)
- **data conversion**: type transfers - changing values from one type to another (amount of allocated storage may change)

Control Structures & Program Flow

- sequential flow (default)
- conditional (branching) structures
  - test then action
- mutually exclusive action(s) taken
- *if, if-else, case, switch-case*
- Iteration (looping) constructs
  - test and conditionally execute repeated action
  - order of test and action different
  - Pre-test loop: *while*
  - Post-test loop: *do-while, repeat*
  - Counter controlled loops: *for*

Modularity Constructs

- Above mentioned set of constructs are typical and all that is needed!
- Modern HLL - modularity: above mentioned divisions become overlapped and intertwined - see next (another level or classification)
- Modularity - division of program into syntactic sections which reflect logical structure of program - ideally, also that of the problem being solved! ie. The solution is expressed in terms of the problem.

Modularity Constructs

- Object oriented languages go further. In C++/Java, a type can be defined using a class. The code (i.e. algorithm) and data is encapsulated in one module (object).

Typical tools to enhance modularity of code:

- **Blocks:**

  begin.....end, { ..... }, or just blank lines
- **Subprograms** (named block structures - procedures/subroutines/functions/methods):
- Supported by most HLL although name and syntax changes (eg. *procedures, functions*)
- essentially the same as a program

“called” from another “main” or “calling” program

“called” from other subprograms

Communication between modules

- **Parameters**
  - Information may be explicitly passed to/from modules
  - Named parameters means same subprogram can be used to operate on different data
  - *Formal (dummy) parameters* used in declaration of subprogram
  - *Actual parameters or arguments* used in calling subprogram
  - ***variable*** (by reference) **vs *value*** parameters (passed by address vs by value)

- **Procedures** vs **Functions** (supported by most, although names change)
- A procedure call is a statement where as a function call is part of an expression.
- Recursion can be viewed as a control construct. Here, a subprogram is used for program control when the subprogram calls itself. Recursion is used to solve a certain class of problems where the problem is defined in terms of itself (eg. calculating factorials or tree traversals)

Documentation

- Most HLLs provide some mechanism for commenting code.
- Eg. ***//***, ***/* */***, **{ }**, **‘**

Abstraction

- Most modern HLL's provide support for abstraction. This enables the programmer to deal with the higher level problems rather than the lower level details which can bog the programmer. Some levels of abstraction:
- Blocks of statements
- Statements grouped by control structures – a for loop summing numbers in an array
- Procedures/functions
- Combination of the above 3 methods to group statements into larger groups

- Structured data – arrays and records
- Use of classes (Smalltalk, C++, Java) to group data and operations
- All the above enable the programmer to express the solution in terms of the problem.

Introduction to C

♦ Why C?

- C is one of the most widely used computer programming languages ever introduced.
- It is a powerful language well known for its concise expressive power, which also makes it appear somewhat unfriendly to the newcomer.
- N.B. Unless explicitly stated, references to C throughout this unit implicitly refer to Borland or Turbo, C
- We will be covering only a subset of C's features in this unit – just enough to write simple programs in C
- You are encouraged to learn more C and you'll find it useful as you proceed to subsequent stages of your course and career !!

Revision on C and Week 2 Practice Work

- Search for Information on C from the Web.
- Look for “Tutorials on C” from the Web
- Research some information on Unix and Linux (what’s the difference from Windows?)
- Refer to Lab Sheet for Week 2 practical
- (Eric will do an introduction to Cygwin and developing C program in Cygwin.)