University of Arizona

CSc 340

Foundations of Computer Systems

Christian Collberg

January 16, 2001

Administrivia

Class	: 340 – Foundations of Computer Systems
Instructor	: Christian Collberg
Contact	: collberg@cs.arizona.edu, 621-6612, GS 758
WWW	: http://www.cs.arizona.edu/~collberg
Lectures	: Tue & Thu, 9:30–10:45, M LNG 311
Office hours	: Tue 14:00–15:00, Thu 15:00–16:00
Lab	: Fri, 11:00–11:50, FCR 202
Book	: Maccabe. Computer Systems: Architecture, Organization, and Programming.
TAs	: Ramachandran, Radha, radhakr@cs, BSE 328A, 621-8111
	: Sundararajan, Radha, radha@cs, BSE 328D,621-8119

Slide 0–1

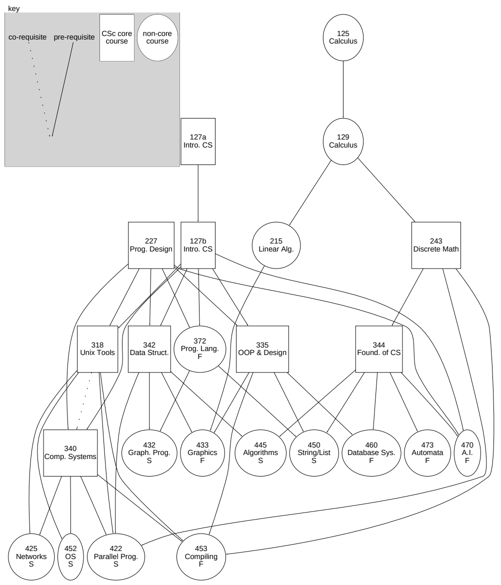


Figure 1: **BS Program: Courses and Pre-requisites.** A summary of the pre-requisite relationships among Computer Science courses contained in the catalog at `http://catalog.arizona.edu/allcats.html`. F = offered in Fall only; S = offered in Spring only.

Slide 0–2

Course Overview

- See the information handout given out in class and course home page: `http://www.cs.arizona.edu/classes/cs340`.
- Overall Goal:
  1. Expose you to the reality of computer systems.
  2. Arguably the single more important systems course in the CSc undergraduate curriculum.
  3. Make the transition from just using computers to understanding computers.

Slide 0–3

Slide 0–4

Course Overview . . .

- More specific goals:
  1. To understand how programs are executed on an actual machine.
- Why?
  1. Executing a program in a HLL is a complex undertaking that helps explain the necessity for different kinds of low-level code.

Slide 0–5

Course Overview . . .

- How?
  1. By understanding the different levels of abstraction in a computing system.
  2. By understanding assembly language programming.
  3. By understanding the basic of low-level systems software.
  4. To understand how the hardware components of a system interact with each other and the software on the system.

Slide 0–6

Abstractions in a Computing System

APPLICATION SOFTWARE

`Editors` `MP3`
`Spreadsheets`

SYSTEM SOFTWARE

`OS` `Linker` `Drivers`
`Assembler` `Loader`

HARDWARE

`CPU` `Memory`
`Disk` `Network`

Slide 0–7

Course Overview . . .

- Why?
  1. The functionality and performance of a program are intricately linked to this interaction.
- How?
  1. By understanding the major hardware components.
  2. By understanding the “language” spoken by the major components.
  3. By understanding the basic policies used to manage these components.

Slide 0–8

Course outline

1. Hardware foundations:
   (a) basic machine organization
   (b) instruction fetch cycle
   (c) number systems and encodings
   (d) CPU operations
2. Assembly language using MIPS as an example:
   (a) syntax
   (b) operations and operands
   (c) using registers, control flow, addressing

Slide 0–9

Course outline . . .

3. Program execution:
   (a) Subroutines and stack
   (b) activation records/stack frames
   (c) calling conventions
   (d) parameter passing
4. Dynamic storage (heap) management
   (a) data structures
   (b) malloc/free/realloc

Slide 0–10

Course outline – Translation

5. Assembling
   (a) structure
   (b) dealing with forward and external references
   (c) symbol tables
6. Linking
   (a) object files
   (b) resolving external references
   (c) Loading

Slide 0–11

Course outline . . .

7. Operating systems:
   (a) paging
   (b) user/supervisor mode
   (c) interrupts, traps and exceptions
   (d) concurrency and mutual exclusion.
8. Learn to program in C and use Unix.

Lectures (preliminary)

0. Administrivia
1. Computer Organization
2. Data Representation
3. Binary Operations
4. Machine Code
5. Introduction to Assembly Code
6. MIPS Instruction Set Overview
7. Simple Control Flow
8. Addressing Modes
9. Static Memory Allocation
10. Subroutines
11. Instruction Set Architectures
12. Delay Slots

Lectures . . .

13. Heap Management
14. Introduction to C
15. C Data Types and Functions
16. C Pointer Arithmetic and Arrays
17. C Structures and Unions
18. C I/O
19. C Misc.
20. Assemblers, Linkers, and Loaders
21. Dynamic Relocation via Segmentation
22. Paging
23. Interrupts, Traps, and Exceptions

Grading

- Grading (Subject to change)
  1. Written homework (10%)
  2. Midterm exam (15%)
  3. One final exam (25%)
  4. Programming assignments (50%)
- Midterm: Tuesday, March 6, in class.
- Final: Tuesday Tues. May 8, 8 a.m. - 10 a.m.

1. All exams are closed book.
2. Without prior arrangement, missed exam ⇒ grade of zero.
3. Fail the final exam ⇒ you might fail the course.

Programming Projects

- There will be six programming assignments, three in assembly code, and three in C. Relative weights for the assignments will be announced when each is assigned.
- Assignments will be graded on design and implementation, in addition to correct functionality. 20% of the grade will be based on the former, 80% on the latter.
- Each student receives a total of three (3) late days for programming assignments. After you have run out of late days, each day late reduces the assignment's grade by 25% of its total value.

Slide 0–16

Academic Integrity

- You will not
  1. turn on another student's work as your own
  2. accept solutions from other students.
  3. give solutions to other students.
  4. tamper with graded papers or exams.
  5. work together on assignments; each student must turn in his or her own work.
- Sanctions typically include:
  - grade reduction, course failure, suspension, expulsion.
- I take this stuff seriously.

Slide 0–17

Academic Integrity ...

- Students who violate the Code are also subject to possible sanctions imposed by the Dean of Students office.
- Submitted solutions will be compared with each other, as well as with solutions from previous semesters.
- All students involved in collusion are equally culpable:
  1. Do not give another student access to your account.
  2. Do not leave printouts in the recycling bin.
  3. Pick up your printouts promptly.
  4. Do not leave your workstation unattended.

  If you suspect that your work has been compromised notify me immediately.

Slide 0–18

Computer Accounts

- Each student will be assigned an account on lectura.
- Go to the Computer Science Department and follow the signs to create your account. It takes several days to complete the process, so get started early.
- If you already have an account on lectura, you should still run the ADDACCT program to update your records. To do this, telnet to lectura and login with username "apply" and password "apply".
- Help on using lectura and Unix can be found at link `http://lww.cs.arizona.edu/help`.

Slide 0–19

A Preview HLL vs. machine code

- Most programs are written in a high-level language (HLL) that is text-based, but computers at the base level use only "0" and "1" (binary digits or bits).
- Implication: must translate between them.
- Machine instructions are represented simply as a sequence of bits:

  1000 1100 1010 0000

  might be the instruction that says

  "add the two numbers b and c and store result in a."
- The "vocabulary" of a specific machine – i.e., the instructions it supports – is called its instruction set.

Slide 0–20

Programming machines

- Early programmers programmed directly with machine language.
- Assemblers translate a symbolic representation called assembly language into machine code:

  `add a, b, c` $\Rightarrow$ `1000 1100 1010 0000`

- HLL are translated into assembly language (compiled), which is then assembled into machine code.

  `a = b + c` $\Rightarrow$ `add a, b, c`

Slide 0–21

"swap" in Assembly and C

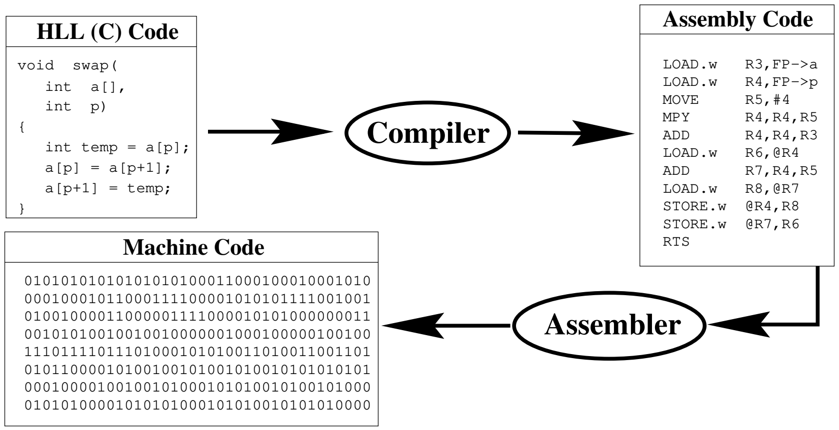
```
void swap(
   int a[],
   int p)
{
   int temp = a[p];
   a[p] = a[p+1];
   a[p+1] = temp;
}
```

$\Longleftrightarrow$

```
LOAD.w    R3,FP->a
LOAD.w    R4,FP->p
MOVE      R5,#4
MPY       R4,R4,R5
ADD       R4,R4,R3
LOAD.w    R6,@R4
ADD       R7,R4,R5
LOAD.w    R8,@R7
STORE.w   @R4,R8
STORE.w   @R7,R6
RTS
```

Slide 0–22

Translation Overview



Slide 0–23

Readings and References

- These handouts have been been adapted from notes originally authored by Rick Schlichting, John Hartman, and Carlos Ugarte.
- Copies of the lecture notes will be provided on the class web page, `http://www.cs.arizona.edu/classes/cs340/index.html`. The actual lecture content may vary, so reading the notes is not a substitute for attending lecture.