Course Overview

- See the information handout given out in class and course home page:

- 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.
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.

APPLICATION SOFTWARE
Editors MP3

SYSTEM SOFTWARE
OS Linker Drivers Assembler Loader

HARDWARE
CPU Memory Disk Network

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

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.
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

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

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.

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
Lectures (preliminary)

0. Administrivia
1. Computer Organization
2. Data Representation
3. Binary Operations
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
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. Final exam (50%)
4. Programming assignments (25%)

- All exams are closed book.
- No prior arrangement, missed exam = grade of zero.
- Midterm: Tuesday, May 8, 8 a.m.
- Final: Tuesday, May 8, 10 a.m.
- Programming assignments will be announced when each is assigned.
- 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.

Programming Projects
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://uw.cs.arizona.edu/help.

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.

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.

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.
**Translation Overview**

HLL (C) Code

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

Assembly Code

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

**Programming machines**

- Early programmers programmed directly with machine language.
- Assemblers translate a symbolic representation called assembly language into machine code:
  \[
  \text{add } a, b, c \Rightarrow 1000 1100 1010 0000
  \]
- HLL are translated into assembly language (compiled), which is then assembled into machine code.
  \[
  a = b + c \Rightarrow \text{add } a, b, c
  \]

**Readings and References**

- These handouts have 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](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.

**“swap” in Assembly and C**

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