Introduction

Dr Patrice Delmas
Email: patrice@cs.auckland.ac.nz
For questions related to 210 contents: see the tutors
Kelvin: tutor210@tcode.auckland.ac.nz
Joe: tutor210@tcode.auckland.ac.nz

Office hours:

By appointment only:
Email subject: 210 appointment

Email
Mistakes in the lecture notes
Deadlinks on the webpage

Forum
- Checking it from time to time
- Don't expect answers from me there
- The tutors are taking care of the forum
-

Tutorials and tutors office hours

Tutors	Monday	Tuesday	Wednesday	Thursday	Friday
11.30 am-12.30pm		K-OH J-OH	J-OH	K-T J-OH	
12.30pm-1.30pm		K-OH J-T	J-OH	K-OH J-T	
1.30pm-2.30pm		K-T J-OH	K-OH	J-OH K-OH	

Assignments:

2 Assignments on Assembly language
(Third in 2 weeks (2 weeks), last in 4 weeks (3 weeks)).
You will have to submit your assignments to tcode.
(You already know how to do that)
You will receive, with your assignment clear directives, regarding file names formatting that will have to be submitted.
An assignment incorrectly submitted will not be marked: 0
Don't pass your work to someone else, don't copy some one else's work: you will receive zero if caught (we won't try to find out who did it, who copied. It will be zero for both)
Markers are doing a fair job.
I will not remark assignments: story

Simulator

You need to know how to use the alpha simulator
Not installed in the undergraduate lab.

- last version available at the following address:

http://www.cs.auckland.ac.nz/compsci210s2c/resources/HOME/bin/simulator.jar

- In the lab, download and install in your G drive
- Start today to understand how it wors
- Tutors will provide lab demos at office hours time

Miscellaneous:

Help: Forum, Tutor, on line guides
Tutor: Weiguo "Kelvin" Jin wjin003@ec.auckland.ac.nz

Email: You **MUST** check your university email account on a weekly basis as vital information may be sent to you regarding assignments, deadlines, etc...
We will have 3 lectures a week. You need to keep up with the pace:

- Data representation and OS (MT lectures) are needed to understand Alpha assembly.
- Each new lecture will require you to know and understand the content of the previous one (s).

Lecture notes and related materials:

- http://tcode.auckland.ac.nz/index.html
- Alpha Architecture Reference manual (Alpha Assembly bible) www.cs.auckland.ac.nz/compsci210s2c/resources/alphaahb.pdf
- Alpha Assembly Language Guide (html guide: useful for quick answers): http://www.cs.auckland.ac.nz/references/unix/digital/APS31DTE/TOC.HTM
- Java http://mirror.cs.auckland.ac.nz/app

Lecture notes, assignments
http://www.citr.auckland.ac.nz/~patrice/lecture_notes.html

Simulator (simulator, documentation)
http://www.cs.auckland.ac.nz/compsci210s2c/resources/
We will use the 2003 syntax:
http://www.cs.auckland.ac.nz/compsci210s2c/resources/HOME/bin/simulator6.006.jar

How to progress while learning

1. **Read the lecture notes after each lecture**
    a. Make a summary of what has been seen on the lectures
    b. Redo examples already treated or do the remaining untreated examples
    c. Do examples without refereeing to the lectures
2. Read the materials provided online
    a. For assembly definitions:
        i. `www.cs.auckland.ac.nz/compsci210s2c/resources/alphaahb.pdf` `(very technical but complete)`
        ii. `www.cs.auckland.ac.nz/references/unix/digital/APS31DTE/TOC.HTM` `(detailed-simple to use)`
    b. For simulator-related issues:
        i. `http://www.cs.auckland.ac.nz/compsci210s2c/resources/` `(look for the assembly section)`
3. If you have questions or do not understand something
    a. Do 1
    b. Do 2
    c. Attend the tutorials
    d. Ask friends
    e. Check the forum
    f. Ask a tutor during office hours
    g. Email me (only if point a to f are completed)

4. How to prepare exams
    a. Do previous years exams
        i. http://www.cs.auckland.ac.nz/compsci210s2c/exams/
        ii. Do 1 and 2
5. How to do assignments
    a. Learn how to use the simulator
    b. Do 1 and 2 on the related lectures before starting the assignment
    c. Write (paper-based) the skeleton of your programme
    d. OR Write the code requested in java if you feel more confident
    e. Write the assembly code
    f. Put comments whenever necessary
    g. Debug the code
    h. Try examples
    i. Submit the assignment properly
    j. Do not cheat

Tentative course overview

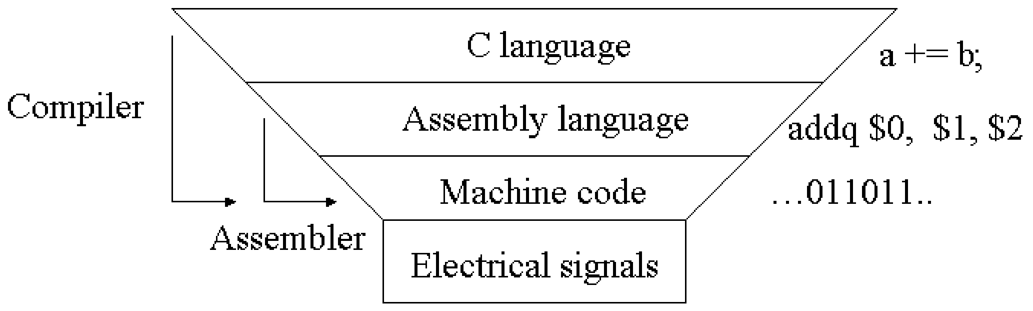
Introduction

1. Introduction, review of CPU architecture
2. First example of assembly programs
3. Registers
4. Instructions
    - Load/store
    - Arithmetic operations
    - Relational instructions
    - Branching
5. I/O in Assembly language (getChar, putChar)
6. Memory allocations
7. More instructions
    - Shifting, clearing, logic
8. Programming
    - Procedures and stack frame
    - Leaf, non-leaf procedures
    - Local variables
9. Coding-decoding instructions
    - 4 sets of instructions

Why study assembly?

"Every Computer Scientists should master Architecture-Algorithms-Applications (AAA)"

From high-level language to low-level language:



Example:
To do a = a + b I can write in C (C++):

- a += b;

In our alpha assembly, it is written as:

- 0x40014400 (addq $0, $1, $0)

In Binary machine code it will look like:

- ..01000000000000100000100000000010…

Machine language

- Flow of 0's and 1's
- Higher-level languages have to be transformed (via a compiler) to it before a program can be run by the computer

Assembly language:

- mnemonics are used to represent the machine instructions (*e.g it is easier to write (and remember) addq to perform the addition between 2 quadwords with the result as a quadword rather than having to provide and hexadecimal code or a set of 0s and 1s*)
- Assembler transforms the assembly code into the machine code
- Unlike high-level languages, assembly languages are not portable between platforms using different architectures.
- This is because instructions in an assembly language are tied to the instruction set of a particular type of CPU.
- Different types of CPUs have different instruction sets.
  - Instruction set: the instructions, which can be understood by the machine.
- Harder to debug (not meant to be done by "normal" humans anymore).
- Needs to reconstruct algorithm architecture
  - *Instructions might have to be modified, orders may differ (we will see examples later on)*
- Time consuming to implement and it is very easy to make mistakes.Fast: takes the maximum from the CPU architecture as it is designed for it.
- Widely use for multimedia applications requiring real time processing.
- Applications that have to run nearly at the limit of the machine's performance, typically graphics-oriented games.
- Applications that must run on a given system
- Embedded applications (main cost: hardware)For the future: bionics, robotics, etc…

Goal:

- Obtain a better understanding of the underlying computer structure that makes your programs work. We are going to learn the Alpha assembly language. The CPUs of "cs26" and "kahu" are Alpha chips.

We could use the assembler on these two servers to generate the machine code and test your programs.

- There are not any tools on cs26 for us to observe the execution of our programs, e.g. set break point, and see the value of the registers.
- This makes debugging very difficult
- Machine might crash each time the programs are not working…

Instead

Use an Alpha simulator written by Dr. Hutton to run our programs. The simulator has a very good user interface and a set of very useful tools for observing and debugging programs.

- Only a simplified version of alpha assembly but it will allow you to understand how instructions are processed, memory allocated, register modified, local variables defined, etc….
- Check out the handout and docs on the web to find more about how to use it.
  http://www.cs.auckland.ac.nz/compsci210s2c/resources/

Why study the Alpha (assembly) ?

The Alpha architecture is the first 64-bit load/store RISC (as opposed to CISC) architecture designed to enhance computer performance by improving clock speeding, multiple instruction issuing and multiple processors management.

- RISC (Reduced Instruction Set Computer) processors are based on the fact that working with internal registers is faster (orders of magnitude) than accessing memory via Bus.
- Intended (in the 80's) as a 25 years long project
  - First 21st century computer architecture

Alpha RISC <u>true</u> 64-bit load/store architecture:

- Use of internal registers which can be loaded from and results transfer back into memory.
- Other work is done between registers
- Instructions interact with each other only by one instruction writing a register or memory and another instruction reading from the same place
  - Can issue multiple instructions every CPU cycle
  - Up to 12 instructions per cycle today

The alpha 21164 was the first processor to break the billion instructions per second threshold in 1996
Alpha processors leader in performance in the 64-bit architecture area
Compaq has announced technology merge with Intel

- Standardisation of future 64-bit architecture will include architectural features of the alpha processors

Alpha Architecture

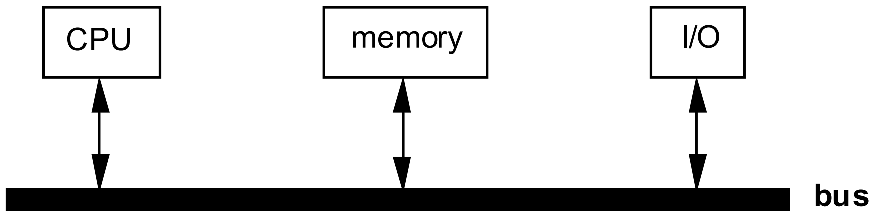
A computer consists of three main components, CPU, memory and I/O devices. The components are connected by a bus, which is used to transmit information amongst the components.
CPU (Central Processing Unit): CPU is used to carry out computation.
Memory: The memory module stores data and programs.

- RAM, ROM, data storage

Alpha: byte addressable memory

- Each memory location has an address and contents
- Memory is a list: the further down the list goes, the higher the address is.

I/O devices: The I/O devices (monitors, keyboards, printers) act as the interface between the computer and the users.



In this part of the course, we are only interested in the CPU of a computer and how it operates.