COMP-573A Microcomputers

Introduction and COMP-273 review

slides by Alexandre Denault

Objective

Students will learn architectural concepts of microprocessors and assembler languages. This includes both theoretical foundations and practical experience. Students will also be required to complete a team project (which include a written report, an oral presentation and a non-trivial amount of programming).

Instructor : Alexandre Denault

- Master student in CS
- Has had experience:
  - Taking this course
  - Being the T.A. for this course
  - Writing the material for this course
- Office: McConnell Room 322
- Email: alexandre.denault@adinfo.qc.ca
- Office Hours:

  Tuesday: 14:30-16:00

  Wednesday: 13:00-14:30

Teacher Assitant (a.k.a. TA)

- Name: Christian Roy
- Office Hours: TBA
- Email: TBA

Topics

- Introduction to the history of microprocessors
- Characteristics of hardware most commonly found on processors (data bus, registers, etc)
- Data representation at the machine level
- Use of assemblers, debuggers and execution monitors
- Architecture of various types of processors

Topics (cont.)

The following architectures will be studies in depth:

- Intel x86 (I32)
- Sparc / UltraSparc
- PowerPC

We will also take a quick look at the following architecture:

- Intel Itanium (I64)
- Motorola m68k
- ARM
- Java VM/Bytecode

Why learn assembler?

- **Gives you a better understanding of computers and the programs you write.**
- Allows you to write code for platform were no adequate compilers exist.
- Allows you to fine tune specific areas of your program.

Why assembler is not popular?

- Because most compiler can write very good assembler and machine code.
- Because it’s very hard to maintain assembler code.
- Because it’s even harder to debug.
- Because it’s a nightmare in large project.
- Because it’s architecture specific.

Why learn assembler at all?

The human should always win and here is why.

First the human writes the whole thing in a high level language.

Second he profiles it to find the hot spots where it spends its time.

Third he has the compiler produce assembly for those small sections of code.

Fourth he hand tunes them looking for tiny improvements over the machine generated code.

The human wins because he can use the machine.

Charles Fiterman, June 1997, comp.compilers newsgroup

Why three learn architectures?

- Because we want to teach methodology, not specialization.
- Because knowing different architectures allows you to better appreciate their differences.
- Because this is a 500 level course.

Evaluation

- Assignments (3) : 20 %
- Midterm (in-class) : 30 %
- Project (presentation and report) : 50 %

Assignments

- You will have three assignments, one for each architecture we will focus on :
  - Intel x86
  - Sparc V8
  - PowerPc 6xx
- These assignment can be completed in teams of two (pair programming)

Assignment Grading

- These three assignments will count for 20% of your grade.
- It is easy to see when student copied their homework or submitted GCC code.
- Students who copy homework will receive a grade of 0.
- Student who submit GCC code will receive a very low mark.

Assignment Grading (cont.)

- No late assignment after the due date will be accepted.
  - This deadline will be called the HARD deadline.
  - Interruption in WebCT services or your own Internet services will not be a valid excuse.

Midterm

- There will be one in-class midterm
  (Thursday, October 30th 2003).
- The format will be similar to previous midterm.
- The only documentation you are allowed to bring is a copy of your assignments.
- To help you prepare :
  - We will do one in-class practice midterm
  - One midterm from the previous year will be available on the web
  - There will be one tutorial (date to be determined) given by the T.A.

Project: Does it have to be in assembler?

- No! In fact I encourage you not to do large projects in assembler.
- The project is there to give you some practical experience in building the architecture for a non-trivial project.
- You are free to chose the technology (i.e. programming language) you will use to complete the project.
- The project topic must be approved by the course instructor.

Project Grading

- There are four deliverables for the project:
  - Project Proposal (due September 18th)

    Short description of your project with your team members name, email and the technology used to complete the project (ex: C, Java, etc)
  - Project Design Report (due October 30th)

    5-6 page document explaining your project and the initial architecture. This document does not need to be very detailed. It is mainly used so I can track your progress on the project. The content of this document is often also used in the project report.
  - Project Presentation (November 4th – December 2nd)

    15-20 minute presentation on the architecture of your project and a small demonstration
  - Project Report (December 2nd)

    20-30 page document on the architecture of your project and the source code to your project.

Examples of past project include …

- Online timetable generator (php)
- Airfoil Design (C and VB)
- Naval Simulation (C++ and OpenGL)
- Board Game (Java)
- 2D Dancing Simulation (Java)
- Internet Radio Station
- 3D Racing Game (3D)
- Online library system
- Dynamic IP System
- Golf Game on Palm
- etc.

- No static websites !

Comp-573 Website

- Can be found at http://www.cs.mcgill.ca/~cs573/ or http://www.cs.mcgill.ca/~adenau/cs573/ .
- Notes, assignments and course syllabus can all be found on the website.
- The "Supplemental Notes" page is a collection of useful resource to do your assignments.

COMP-273 Review

- Byte ordering
- Basic components of a microprocessor
- Cisc VS Risc

Why is byte ordering important?

- Because architectures do not store their numbers the same way.
  - If you want to extract only a section of a number, you have to understand how they are stored in memory or on the stack.
  - If you transfer data from one architecture to another (networking), you have to work with a common byte ordering.

Big-endian vs Little-endian

- The adjectives big-endian and little-endian refer to which bytes are most significant in multi-byte data types and describe the order in which a sequence of bytes is stored in a computer’s memory.
- The terms big-endian and little-endian are derived from the Lilliputians of Gulliver's Travels, whose major political issue was whether soft-boiled eggs should be opened on the big side or the little side. Likewise, the big-/little-endian computer debate has much more to do with political issues than technological merits.

Big-endian vs Little-endian (cont.)

- In a big-endian system, the most significant value in the sequence is stored at the lowest storage address (i.e., first). In a little-endian system, the least significant value in the sequence is stored first. For example, consider the number 1025 (2 to the tenth power plus one) stored in a 4-byte integer:

  00000000 00000000 00000100 00000001

Address	**Big-Endian**	**Little-Endian**
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

Two's Complement

- Binary representation of a negative number.
- To convert to or from two's complement, simply invert the bit representation of a number and add one.
- For example:

```
000111  : 7
111000  : Flip the bits
111001  : Add one and we have -7
000110  : Flip again
000111  : Add one and we have 7 again
```

- Addition and subtraction also work in two's complement.

  http://www.duke.edu/~twf/cps104/twoscomp.html

IEEE Floating Point Numbers

- Small integer numbers can easily be converted to binary and store in memory as-is.
- Larger integer numbers are harder to store because of the large number of bits needed for perfect precision.
- Decimal numbers are also harder to represent because of their fractionnal component.
- The IEEE Floating Point Numbers allow us to represent large and small numbers with a garantied level of precision.

IEEE FP Numbers (cont.)

- A floating point number has three components: Sign, Exponent and Mantissa
  - A sign bit is used to indicate if the number is positive (0) or negative
  - The exponent is used to describe the scale/size of a number.
  - The mantissa describe the fractional part of the number.
- For example : 15.125

  -1111.001 would be converted to

  -1.111001×10^{3}

  Sign: 1 | Exponent: 3 | Mantissa: 111001

IEEE FP Numbers (cont.)

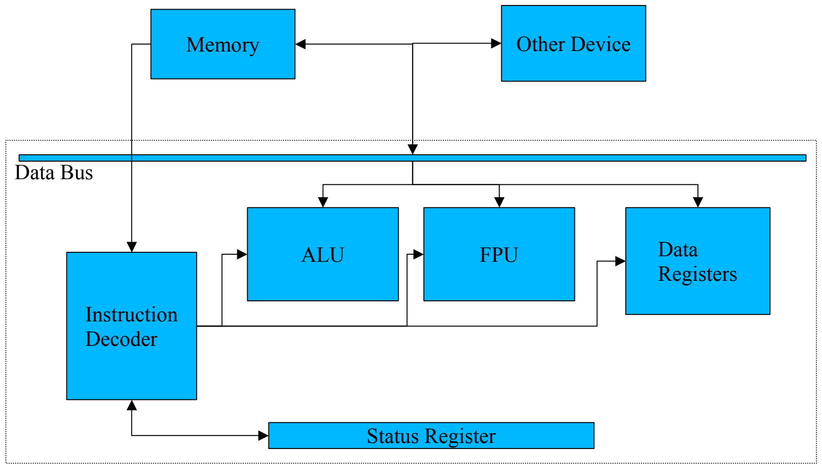
- Floating point numbers have a fixed size in memory. Unused bits must be padded with 0 values.
- Exponents need to be bias (since we do not want to store negative numbers in two's complement).

	Single	Double	Quad
Number of bits taken by:			
Sign	1	1	1
Exponent	8	11	15
Fractional mantissa	23	52	111
Total	**32**	**64**	**128**
Exponent:			
Bias	127	1023	16383
Range of (biased) exponent:	0..255	0..2047	0..32767

IEEE FP Numbers (cont.)

- Taking our previous example
  Sign: 1 | Exponent: 3 | Mantissa: 111001
- And converting it to a “single” floating point number:
  1) Bias the exponent: 3 + 127 = 130
  2) Write the different parts in binary (padding with 0's):
    Sign: 0
    Exponent: 1000 0010
    Mantissa: 1110 0100 0000 0000 0000 000
  3) Append the three parts (and convert to hex):
    0100 0001 0111 0010 0000 0000 0000 0000 –> 4172 0000

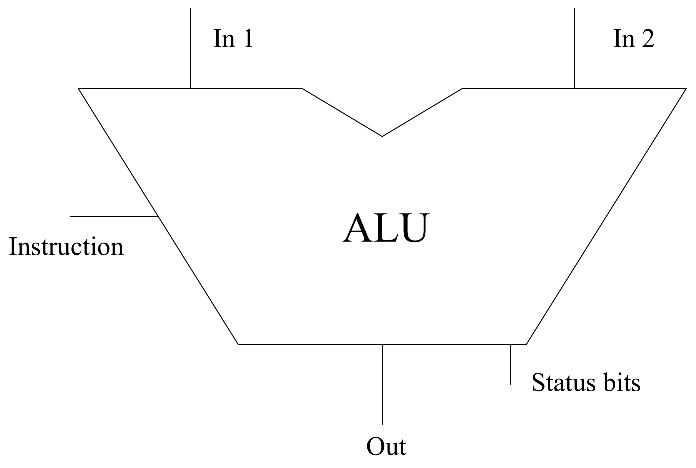
Basic Components of a Microprocessor



Arithmetic & Logical Unit

- Responsible for most integer and logical operations (add, subtract, multiply, and, or, xor, etc)
- Integer division is sometimes handle by the FPU.
- Most ALU's have two inputs and one output.
- After an operation, the status register is often updated to reflect special properties of the output (carry, overflow, zero value, etc)

Arithmetic & Logical Unit (cont.)



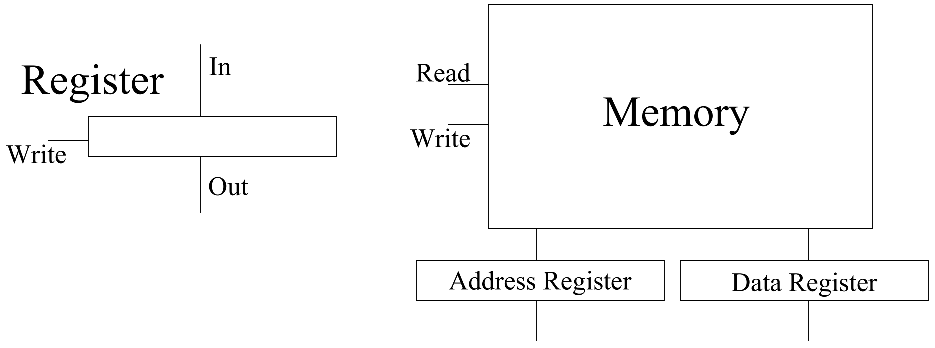
Floating Point Unit

- Responsible for floating-point operations (fadd, fsubtract, fmultiply, fdivision)
- Specialize FPU can also execute exponential or trigonometric calculations (square roots, cosines, etc)
- On some architecture, the FPU is located on a separate chip (not on the microprocessor).
- Some architecture use their ALU and some microcode to emulate a FPU (which is significantly slower).

Memory & Registers

- Collections of AND/OR gates that cycle multiple electrical signals, storing values of 0 and 1.
- Content is lost when computer is powered down.
- Registers and memory share the same basic structures.
  - Registers are much faster because individual registers are connected directly to the data bus.

Memory & Registers (cont.)



Status Register

- In a status register, each individual bit is used to describe a particular state of the computer.
- Popular status bits are those modified by the ALU (carry, overflow, zero value, etc).
- Bits from the Status Register are often used to by conditional branch instructions.
- Each architecture has a different set of status codes.

Instruction Decoder

- The instruction currently being executed is held in an instruction register.
- The instruction decoder uses a microprogram (written in microcode) to look up the instruction and find which actions much be taken.
- Most actions require several actions to be taken over different cycles.
- Evolved instruction decoder can process numerous executions and pipeline them (process them in a way that the CPU can execute numerous action simultaneously).

Instruction Pipeline

- In a non-pipeline system, certain parts of the CPU are often left idle. For example, the data bus is idle while the system is adding.
- Instruction pipelining is accomplished with additional hardware that breaks up instructions into sub-instructions.
- The sub-instruction are then executed in step.
- Once an instruction has passed the initial step of the pipeline, the next instruction is fetched.
- Each step of is called a pipeline stage.
- Today, it is not uncommon to have many pipeline stages (as many as 11 in the Athlon XP and 20 in the Pentium 4)

Instruction Pipeline (cont)

- The following illustrates a simple scenario ("3 stage" pipeline) with 3 add instructions.
- Normally, 3 add instructions would require 9 cycles.
- With our "3 stage" pipeline, it only takes 5 cycles.

load	add	store		
	load	add	store	
		load	add	store

Cisc vs Risc

- The are two different (and often conflicting) design philosophies for microprocessor architecture.
- CISC: Complex Instruction Set Computers
  - Intel x86 processors
  - Motorola m68k processors
- RISC: Reduce Instruction Set Computers
  - Sparc / UltraSparc processors
  - PowerPC processors

Complex Instruction Set Computers

- Design Goal: Reduce the quantity of assembler program should have to write.
- CISC processors are often equipped with a very rich set of instruction.
- Unfortunately, the instructions are often heavy, requiring numerous CPU cycle to complete.
  - Ex: Moving data from memory to memory
- CISC processors traditionally have variable length instructions.

Reduce Instruction Set Computers

- Design Goal: Every instruction should only take one cycle to complete.
- RISC processors are often equipped with a large quantity of general purpose registers.
- Though most RISC process also come with a large number of instructions (comparable to a CISC processor), most of these instructions are very fast to complete.
- RISC is a load and store architecture, no operation is done from memory to memory. Registers are extensively used temporary space.

Comparing the two architectures

- The difference between CISC and RISC have been blured greatly in the last few years.
- “Large number of registers” and “Fixe size instructions” are two features that were usually associate with RISC.
- However, many CISC makers have integrated these two features into their chips.
- A CISC makers optimize there instruction set, CISC are slowly approching the 1 instruction / cycle barrier.
- However, you can still differentiate a CISC chip from a RISC by the complexity of the operations.

References

- The IA-32 Intel Architecture Software Developers Manual
  Volume 1
- Chronology of Personal Computers
  http://www.islandnet.com/%7Ekpolsson/comphist/
- The History of the Microprocessor
  http://www.sit.wisc.edu/%7Emptaylor/microprocessor.html
- CS273 Course Notes
- Linux Assembly HOWTO
  http://www.tldp.org/HOWTO/Assembly-HOWTO/index.html
- Two's complement
  http://www.duke.edu/%7Etwf/cps104/twoscomp.html
- Wikipedia
  http://www.wikipedia.org/