CEG2400 - Microcomputer Systems

Lecture 1: Introduction

Philip Leong

Course Details

- Lecturer: Prof. Philip Leong (phwl)
- Tutors: Matthew Wai Chung Tang (wctang) and Chan Fai (fchan)
- Website: http://www.cse.cuhk.edu.hk/~phwl/teaching/ceg2400
- Lectures/Tutorials:
  - M3-4 10:30-12:30pm (SCL1) and T6 1:30pm (LSB241).
  - M9 4:30-5:15pm (SHB102)
- Credit: 50% Final, 25% Mid-Term, 25% Homework

Thanks to Drs Y.S. Moon, O. Mencer, N. Dulay, P. Cheung for some of the slides used in this course

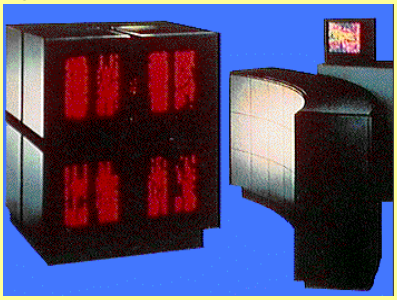
- References:
  - Furber, “ARM System-on-Chip Architecture”, 2000
  - (refer to course website for online references)
- Academic honesty: http://www.cuhk.edu.hk/policy/academichonesty/

What is a Computer?

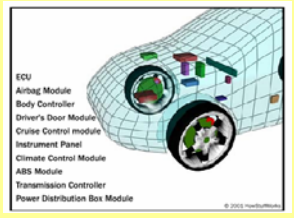
- Apple Macintosh “Personal Computer” (circa 1990)

What is a Computer?

- Thinking Machines “Connection Machine”



What is a Computer?



–ARM is most successful high performance embedded processor (1.3 per mobile phone)

What are they used for?

- Word processing
- Image processing
- Art: Music/Pictures/Movies creation, distribution
- Entertainment / Games
- AI - Robotics
- Simulation
- Communications e.g. Chat, Email, Video conferencing, etc.

What are they used for?

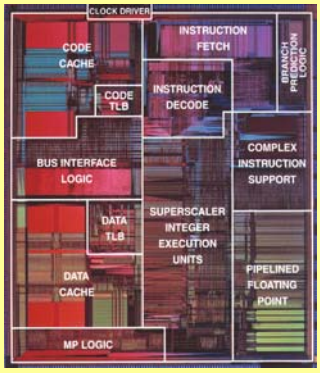
- Email
- Internet Browsing
- Games
- Databases
- Word Processing
- Spreadsheets
- Desktop / Web Publishing
- Accounts/Stock Control/Banking
- Payroll
- Education
- Machines/Appliances/Electronic Devices
- Computer-Aided Design
- Air-Traffic Control
- Weather Prediction
- Weapons
- Designer Drugs
- Oil Exploration
- Human Genome Project
- Financial Markets
- Nuclear Reactor Control
- Exploring Space
- Art: Music, Pictures, Movies, creation and distribution

Computers & CPUs

COMPUTER
Apple
Power Macintosh
7100/66

CPU
Motorola
PowerPC 601/66

CPU on a Chip -> Microprocessor



- Microcontroller is a computer on a chip

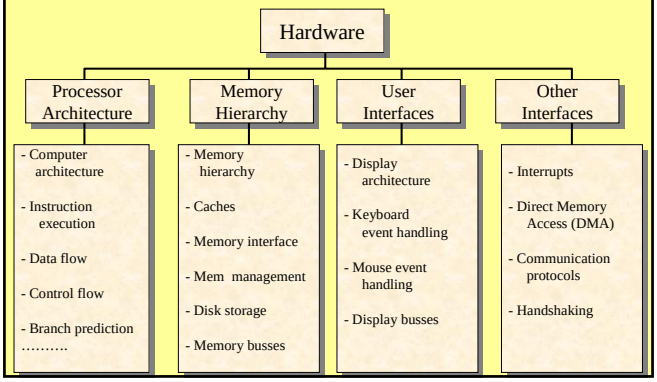
CPUs

- Intel Pentium
- Motorola/IBM PowerPC
- AMD K7
- ARM StrongArm
- Compaq (DIGITAL) Alpha
- Zilog Z80
- Motorola 68000
- 6502
- MIPS

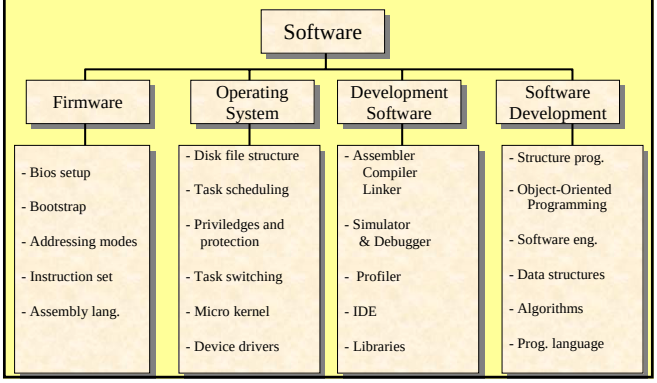
- Interesting details about CPUs

http://bwrc.eecs.berkeley.edu/CIC/archive/cpu_history.html

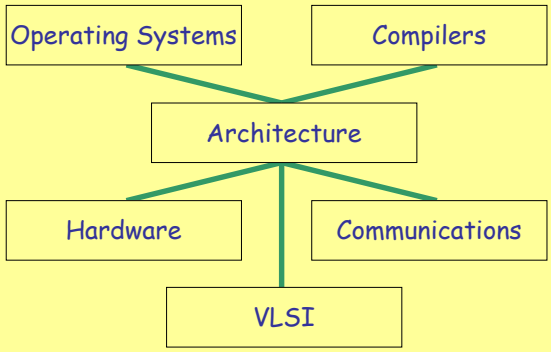
Hardware/Software



Hardware/Software



Why ?



Computer Organization

Binary Numbers	Radixes (2, 16), Conversions & Arithmetic
Data Representation	Integers, Integer Arithmetic, Reals, Text
Main Memory Organisation	Byte-Ordering, Alignment, Interleaving
CPU Organisation & Operation	Instructions, Registers, Fetch-Execute Cycle
CPU & Programming	Registers, Addressing Modes, Mapping High-Level Programs
Input/Output Control	Device Types, Polling, Interrupts, Device Drivers

What will be covered in CEG2400?

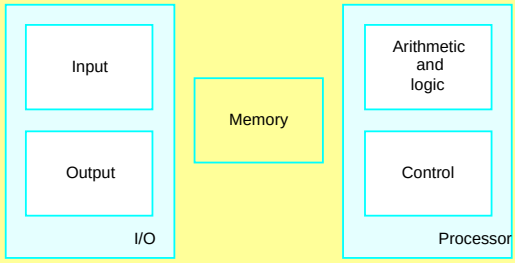
- Lectures
  - Basic Knowledge
  - Machine Instructions & Assembly Language
  - ARM7 instruction set
  - I/O
  - Memory
- Tutorials
  - Reinforce lecture material on computer architecture
  - Assembly language programming
  - Microprocessor interfacing
- We will use the NXP (formerly Philips) LPC2000 family (ARM7) SoC as an example – build your own!

Tutorial Details

"I hear and I forget. I see and I remember. I do and I understand" – Confucius

- Week 1-2 make and test your own LPC2131 board
- Week 3 write and debug a simple program
- Week 4 program a peripheral device
- Week 5-6 develop own seven segment display interface
- Week 7 interrupt programming exercise
- Week 8-11 develop moderate complexity hardware/software design
- Week 12-13 programming exercises

Overview of Organization



Input/Ouput Unit

- Input unit
  - Keyboard, mouse, microphone, CDROM etc
- Output unit
  - Graphical display, printer etc
- Use the collective term input/output unit (I/O unit)
  - Input units, output units, disk drive etc

Memory unit

- Memory used to store programs and data
- Unit of access is an n-bit word
  - Unique location is its address
  - Retrieval is in units of words
  - Commonly 32-bit today, moving to 64-bits
  - Typically 16- 64 bit machines
- Primary storage: random-access memory (RAM)
- Secondary storage: hard disk, CDROM etc

Processor

- Registers
  - Small but fast storage of intermediate values in a computation
- Arithmetic logic unit: performs computations
  - e.g. add, divide, logical operations etc
  - c.f. calculator
  - Operands taken from registers
- Control
  - Orchestrates the transfer of data and sequencing of operations between memory, registers, ALU, I/O devices

How a program is executed

- Program and data reside in memory, on CDROM or hard disk
- Data stored in memory is fetched under program control into ALU and processed
- Processed information sent back to I/O unit
- All activities directed by control unit

Bus

- A bus is used for communications between the processor, memory and I/O
  - advantages?
  - multiple buses are often used for higher performance

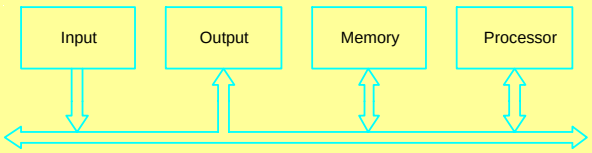


Figure 1.3. Single-bus structure.

Bus

- Allows computer to be customised for different applications e.g. different peripherals
- Design criteria – speed, cost, etc
- Word length can be different depending on application
  - E.g. USB is serial (1-bit), PCI bus is 32-bit

Performance

- Most important measure is how quickly it executes your program
  - Compiler, instruction set and hardware architecture, program all have impact on performance
  - For embedded systems may be measured differently e.g. worse case response time

Numbers and Arithmetic Operations

Counting to 1

- Binary numbers (0, 1) are used in computers as they are easily represented as off/on electrical signals
- Different number systems are used in computers
- Numbers represented as binary vectors
- $B=b_{n-1}\ldots b_1 b_0$
- Unsigned numbers are in range 0 to 2^n-1 and are represented by $V(B)=b_{n-1}\times 2^{n-1}+\ldots+b_1\times 2^1+b_0\times 2^0$
- MSB=Most significant bit (leftmost digit in a binary vector)
- LSB=Least significant bit (rightmost digit in a binary vector)

Negative Numbers

- Sign-and-magnitude
  - Most significant bit determines sign, remaining unsigned bits represent magnitude
- 1's complement
  - Most significant bit determines sign. To change sign from unsigned to negative, invert all the bits
- 2's complement
  - Most significant bit determines sign. To change sign from unsigned to negative, invert all the bits and add 1
  - This is equivalent to subtracting the positive number from 2^n

Number Systems

B	Values represented		
$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+ 7	+ 7	+ 7
0 1 1 0	+ 6	+ 6	+ 6
0 1 0 1	+ 5	+ 5	+ 5
0 1 0 0	+ 4	+ 4	+ 4
0 0 1 1	+ 3	+ 3	+ 3
0 0 1 0	+ 2	+ 2	+ 2
0 0 0 1	+ 1	+ 1	+ 1
0 0 0 0	+ 0	+ 0	+ 0
1 0 0 0	- 0	- 7	- 8
1 0 0 1	- 1	- 6	- 7
1 0 1 0	- 2	- 5	- 6
1 0 1 1	- 3	- 4	- 5
1 1 0 0	- 4	- 3	- 4
1 1 0 1	- 5	- 2	- 3
1 1 1 0	- 6	- 1	- 2
1 1 1 1	- 7	- 0	- 1

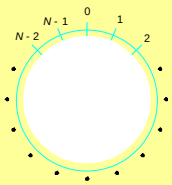
Addition (1-bit)

0	1	0	1
+ 0	+ 0	+ 1	+ 1
0	1	1	1 0

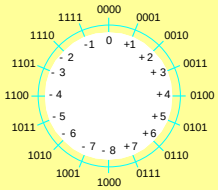
Carry-out

2's Complement

- 2's complement numbers actually make sense since they follow normal modulo arithmetic except when they overflow
- Range is -2^{n-1} to $2^{n-1}-1$



(a) Circle representation of integers mod *N*



(b) Mod 16 system for 2's-complement numbers

Add/sub

- X+Y – use 1-bit addition propagating carry to the next most significant bit
- X-Y – add X to the 2's complement of Y

Add/Sub (2's comp)

(a)	0010	(+2)		(b)	0100	(+4)	
	+0011	(+3)			+1010	(-6)	
	0101	(+5)			1110	(-2)	
(c)	1011	(-5)		(d)	0111	(+7)	
	+1110	(-2)			+1101	(-3)	
	1001	(-7)			0100	(+4)	
(e)	1101	(-3)	⇒		1101		
	-1001	(-7)			+0111		
					0100	(+4)	
(f)	0010	(+2)	⇒		0010		
	-0100	(+4)			+1100		
					1110	(-2)	
(g)	0110	(+6)	⇒		0110		
	-0011	(+3)			+1101		
					0011	(+3)	
(h)	1001	(-7)	⇒		1001		
	-1011	(-5)			+0101		
					1110	(-2)	
(i)	1001	(-7)	⇒		1001		
	-0001	(+1)			+1111		
					1000	(-8)	
(j)	0010	(+2)	⇒		0010		
	-1101	(-3)			+0011		
					0101	(+5)	

Sign Extension

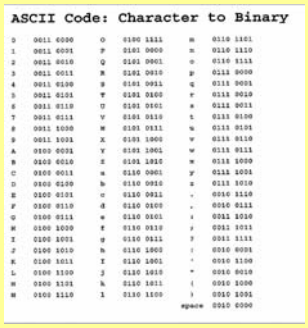
- Suppose I have a 4-bit 2's complement number and I want to make it into an 8-bit number
  - Positive number – add 0's to LHS e.g. 0111 -> 00000111
  - Negative number – add 1's to LHS e.g. 1010 ->11111010
  - c.f. circle representation

Overflow

- In 2's complement arithmetic
  - addition of opposite sign numbers never overflow
  - If the numbers are the same sign and the result is the opposite sign, overflow has occurred
  - E.g. 0111+0100=1011 (but 1011 is -5)
- Unsigned
  - Carry out signals an overflow

Characters

- Typically represented by 8-bit numbers



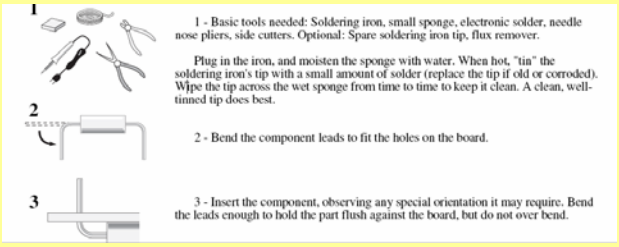
Quiz

- Assuming 4-bit 2's complement numbers
  - What is the binary for -2?
  - Calculate 2+3
  - Calculate -2-3
  - Calculate 5+5
- Assuming 5-bit numbers
  - What is the largest 2's complement number?
  - What is the smallest 2's complement number?
- Convert 56 to unsigned binary
- What is the decimal value of 10110101 in 2's complement? What is the unsigned value of the same binary number?

Nuts and Bolts

How to solder

http://www.robotstore.com/download/How_to_solder_1.pdf



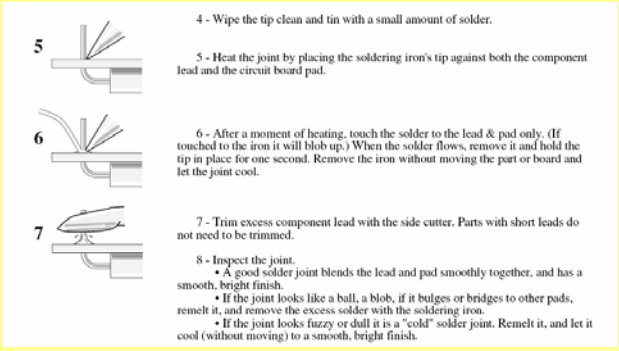
1 - Basic tools needed: Soldering iron, small sponge, electronic solder, needle nose pliers, side cutters. Optional: Spare soldering iron tip, flux remover.

Plug in the iron, and moisten the sponge with water. When hot, "tin" the soldering iron's tip with a small amount of solder (replace the tip if old or corroded). Wipe the tip across the wet sponge from time to time to keep it clean. A clean, well-tinned tip does best.

2 - Bend the component leads to fit the holes on the board.

3 - Insert the component, observing any special orientation it may require. Bend the leads enough to hold the part flush against the board, but do not over bend.

How to solder



4 - Wipe the tip clean and tin with a small amount of solder.

5 - Heat the joint by placing the soldering iron's tip against both the component lead and the circuit board pad.

6 - After a moment of heating, touch the solder to the lead & pad only. (If touched to the iron it will blob up.) When the solder flows, remove it and hold the tip in place for one second. Remove the iron without moving the part or board and let the joint cool.

7 - Trim excess component lead with the side cutter. Parts with short leads do not need to be trimmed.

8 - Inspect the joint.
- A good solder joint blends the lead and pad smoothly together, and has a smooth, bright finish.
- If the joint looks like a ball, a blob, if it bulges or bridges to other pads, remelt it, and remove the excess solder with the soldering iron.
- If the joint looks fuzzy or dull it is a "cold" solder joint. Remelt it, and let it cool (without moving) to a smooth, bright finish.

LPC213x devices

- ARM7DMI-S microcontroller (i.e. computer-on-a-chip)
- 8/16/32kB on-chip static RAM, 32/64/128/256/512kB flash memory
- Debugging interface
- 8-channel 10-bit ADCs, 10-bit DAC
- 32-bit timers for events and watchdog
- Real time clock
- Serial interfaces: UART, I2C bus, SPI and SSP
- Interrupt controller
- Power saving modes
- 60 MHz operation
- 3-3.6V operation with 5V tolerant pads