COMS 1003 Fall 2005

Introduction to Computer Programming in C

Computer Organization (cont.) and Introduction to Programming

September 15th
What's Ahead

• Continuation of Computer Architecture and Organization

• Introduction to programming
  – Computing environment (compiler, make)
  – Design, edit/construct, compile, run/test
CPU Classification

- Accumulator
- Stack
- General Register (modern)
Machine Instruction Set

• The “native language” of the processor
  – Set of directly executable commands
  – Binary string
  – Opcode + address

• programmers use an assembly language to represent the opcodes

• Operands

• Addressing modes
Assembly Language

- Data Transfer
  - Mov, store, load, push, pop
- Control Flow
  - Jmp, jnz, cmp, nop
- Logical
  - And, or, xor, shift, not
- Math
  - Add, mul, sub, inc, div
Taking stock: Tradeoffs

- CS has a number of “vs” relationships:
  - Big vs. Little endian
  - RISC vs. CISC
  - Interrupts vs. Polling
  - Translation vs. Interpretation
  - Open vs. Closed source
  - Combinational vs. Sequential circuits
  - Space vs. Time
Memory Hierarchy

• Space v. Time: Fast memory is expensive
  – Registers
  – caches (L1,L2,L3) (few nsec)
  – main memory (RAM) (tens of nsec)
  – magnetic hard disks
  – Tape
  – CD, DVD
A Penny for your Thoughts

• A 1-bit memory can be formed from a *clocked D latch*

• Basic circuit is the SR latch

• *Flip-flops* are another term for latches; there is a difference, but unimportant for our purposes

• String a bunch of these together and you get a register.
A Block of Memory

- What do we need here?
  - The input bits
  - Address bits
  - Chip Select
  - Read/Write flag
RAM, ROM

• Random Access Memory
  – SRAM (static RAM, from flip-flops)
  – DRAM (capacitors, high density, need refreshing)
  – SDRAM (hybrid)

• Read Only Memory
  – PROM (blow some fuses)
  – EPROM (ultraviolet light)
  – EEPROM (flash memory)
The Bus

• Not #36 Jerome Bettis, but close
• Data and control signals must move between circuits
• The bus is a common grouping of wires that accomplish this between the major circuits and devices
• The CPU usually has a dedicated bus to the memory and another bus for all the devices
The Magic Bus

• Bus carries
  – Data
  – Addresses
  – Control

• We need hardware to 'arbit' the bus with a bus protocol

• Bandwidth and speed
Bus Width

- The more address lines a bus has, the more memory the CPU can address.
- Wider buses take up more space
- The 8088 had a 20 bit bus (how much memory?)
- The 80286 had 24
- The 80386 had 32
- Address and data lines are often multiplexed
  - Leads to serious synchronization issues
Bus Arbitration

- Daisy chain from arbiter
- Decentralized approach (interrupt)
Review

• CPU, ALU, data path
• Von Neumann architecture
• Memory Hierarchy
• Simple half-adder circuit & truth table
• Simple pipeline example
Computer Programming

• Basically, telling the computer:
  – What to do
  – How to do it
  – When to do it

• What operations to perform on what data
  – Structured, organized, precise
  – Concrete expression of abstract algorithms
Translation and Interpretation

- **Source Code**
  - Compiles to:
- **Object Code** (executed by hardware)
  - Or:
- **Interpreted** (executed by some other software)
Software Life Cycle

- Problem Statement
- Requirements Specification
- Design Specification
- Construction or Implementation
- Testing
- Documentation
- Maintenance
Design Strategies

- Top-down (outline)
- Bottom-up (essential details first, building blocks)
- Object Oriented (OOP)
- eXtreme Programming (feedback from customer)
- Many others, which are both design strategies and programming styles or disciplines...
Introduction to OOP

- Object = data + methods (a class)
- Easier to maintain and reuse
- Inheritance (sub & super class relationships)
- Polymorphism
- Encapsulation
What Are Some Programming Languages?

• High-level vs. low-level languages

• Languages are defined by precise grammars: the rules of what alphabet symbols can appear where in a source code listing

• A language is a model of syntax and semantics

• Compiled vs. interpreted
Low Level Languages

• Assembly Language
  – Mneumonics for machine instruction set
  – Opcode + address/operands
  – Function( x, y )
  – Add a, b
  – And a, b
  – Mul a, b

• Execution begins at top and usually falls straight through
High-Level Languages

- Resemble everyday speech, but more structured and precise
- Variables
- Types
- Structured Blocks
- Higher-level control flow constructs (loops)
C

• A mid-level language defined by a compiler
  - small core of operators and key words
  - a standard library of useful functions
  - an operational environment (mostly hidden by the standard library) provided by the Operating System 'system call interface'

• The C preprocessor
  - NOT C – a macro language that is interpreted in a step before compilation
C :: Beginning Execution

• Understand that many 'rules' are conventions: techniques and habits adopted to make programming easier

• Program Execution has to begin somewhere:
  - int main(int argc, char* argv[])
  { }
C Facts

- Whitespace (blank lines, spaces, tabs, newlines) is ignored by the compiler
  - C is case-sensitive
  - statements must end in a ;
- A comment is a piece of text that is not part of the program, but explains something in the source code
  - A comment can be single line or multi-line
  - // starts a single line comment
  - /* denotes a multi-line comment */
More C Facts

• A string of characters in double quotes:
  – “hello world.” is known as a **string**, a character string, or a string literal

• A character is contained in single quotes:
  – 'a'

• Some strings/words have special meaning in the C language. These words are reserved and the programmer cannot use them as a variable name.
end