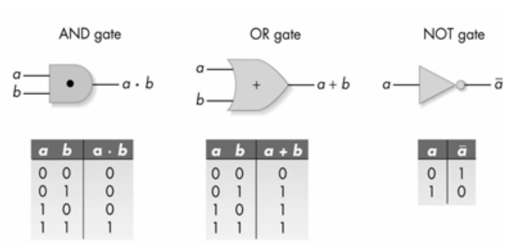
CS1004: Intro to CS in Java, Spring 2005

Lecture #9: Computer organization, Java OO

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW#2 due next Tuesday



The Three Basic Gates and Their Symbols
(see pages 156-157 for implementation)

A few examples of computation circuits

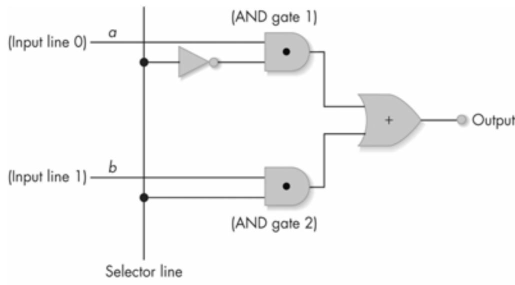
- 1-bit equality
  - Two inputs, one output
- *n*-bit equality
  - Composed of many 1-bit equality circuits ANDed together
- 1-bit adder
  - Three inputs, two outputs
- *n*-bit adder
  - Composed of many 1-bit adders chained together
- Let's do these on the board
  - Pages 165-172

Control Circuits

- Do not perform computations
- Choose order of operations or select among data values
- Two major types:
  - Multiplexors
    - Select one of a number of inputs to send to output
  - Decoders
    - Sends a 1 on one output line, based on what input line indicates

Muxes

- Multiplexor form
  - 2^N regular input lines
  - N selector input lines
  - 1 output line
- Purpose
  - Given a code number for some input, selects that input to pass along to its output
  - Used to choose the right input value to send to a computational circuit

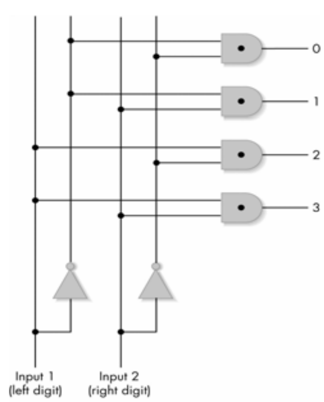


A Two-Input Multiplexor Circuit (p. 174)

Decoders

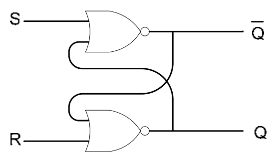
- Decoder
  - Form
    - N input lines
    - 2N output lines
  - N input lines indicate a binary number, which is used to select one of the output lines
  - Selected output sends a 1, all others send 0
- Purpose
  - Given a number code for some operation, trigger just that operation to take place
  - Numbers might be codes for arithmetic: add, subtract, etc.
  - Decoder signals which operation takes place next

A 2-to-4 Decoder Circuit (page 175)



Memory circuit

- Trick: route the outputs of our logic gates back into the inputs
  - *SR flip-flop*
- Use NOR gates (OR, with a NOT attached to the end)
- You don't have to know this, but the basic principle is useful
- Let's draw out the truth table…



Big picture of control circuits

- Mux: select data elements to be computed upon
  - If we have n *registers* (memory cells), can we choose the two to be used for a math operation
- Decoder: Which operation do we want to do?
  - Store it as a two-bit command
- A computer instruction basically consists of four parts
  - A command
  - Cell for the first operand
  - Cell for the second operand
  - Cell for the result

What's a processor?

- Consists of control circuits and an ALU (Arithmetic Logic Unit)
  - An ALU is capable of doing addition, subtraction, etc.
  - Uses *machine language* and *decodes it* to decide what to do – basically, a very fast calculator
- Part of the study of *computer organization*, utilizing abstraction
  - Given semiconductors, we can create basic logic operators
  - Given basic logic operators, we can create 1-bit adders
  - Given 1-bit adders, we can create *n*-bit adders
  - With adders and the like, we can create an ALU
  - Given an ALU, we can create a microprocessor
  - Given a microprocessor and some other things, we can build a computer
- The *Von Neumann* architecture is what modern PCs use

Von Neumann architecture

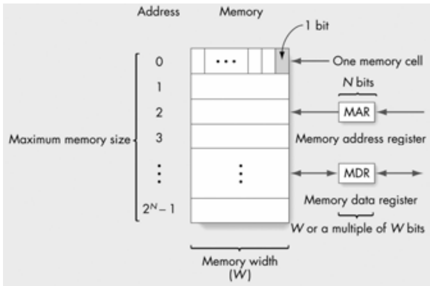
- Von Neumann architecture has four functional units:
  - Memory
  - Input/Output
  - Arithmetic/Logic unit
  - Control unit
- Microprocessor has the ALU, control unit, and temporary memory for operations
- Sequential execution of instructions
- Stored program concept

Memory and Cache

- Information stored and fetched from memory subsystem to processor
- Random Access Memory maps addresses to memory locations
- Cache memory keeps values currently in use in faster memory to speed access times

Memory and Cache (continued)

- RAM (Random Access Memory)
  - Computer's main *volatile* memory
  - Memory made of addressable "cells"
  - Current standard cell size is 8 bits (byte)
  - All memory cells accessed in equal time
  - Memory address
    - Unsigned binary number N long
    - Address space is then 2^N cells



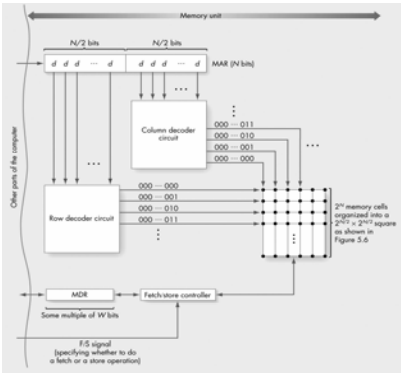
Structure of Random Access Memory
(page 188)

Doing the math…

- N = 10, 2^N = 1,024 (one kilobyte)
- N = 20, 2^N = 1,048,576 (one megabyte)
- N = 30, 2^N = 1,073,741,824 (one gigabyte)
- N = 40, 2^N = 1,099,511,627,776 (one terabyte)
- Pentium I, II, III, IV are *32-bit* machines, so a single program can access 2^{32}, or 4294967296 bytes
  - Up to 4 *gigabytes* of memory (without tricks)
- Newer processors are *64-bit*, or 2^{64}, or 18446744073709551616 bytes
  - Up to 16 *exabytes* of memory
  - One exabyte = 1024 petabytes = 1024^2 terabytes = 1024^3 gigabytes
  - One exabyte = 2^{60} = 18 million 60GB iPods
  - What's after exabyte? Zettabyte = 2^{70}, Yottabyte = 2^{80}

Memory Subsystem

- Fetch/store controller
- Two special, fast memory *registers*:
  - Memory address register (MAR): contains *location* of data to fetch/store
  - Memory data register (MDR): contains *content* of data to fetch into/store from
- Memory cells, with decoder(s) to select individual cells



Overall RAM Organization (p. 195)

Cache Memory

- Problem
  - Fast memory is expensive
  - Slow memory isn't fast enough for the CPU
- Solution
  - Use both
  - Fast memory acts as a *cache* for the slow memory
  - Locality principle: once a value is used, it is likely to be used again

Next time

- Finish computer architecture
- Continue Java OO concepts