CS 5515
Computer Architecture

VA Tech
The Computer as a Tool of the Mind 1

- Overcoming human mental limitations
  - speed
  - accuracy / avoidance of error caused by distraction, fatigue, etc
  - precision memory

- Distinction between data and process
  - definition of process/algorithm as an abstraction concept

- Distinction between numeric calculation and information management
  - needs for automated numeric calculation pre-dated for automated information management

- Human advantages as a processing system
  - unlimited long term memory
  - tight integration with analog senses
  - heuristic processing
  - adaptive processing
  - parallel cognitive understanding
  - subjective & judgmental
The Computer as a Tool of the Mind 2

- **Unsolvable problems** (undecidable)
  - no theoretic machine could solve
  - usually involves problems dealing with
general solutions, not merely specific cases
  - e.g. *The Turing Machine Halting Problem*
    - determine if an arbitrary Turing Machine
      will halt given any arbitrary input set
    - an infinite loop detecting program
  - problems may be practically unsolvable within a
  finite number of states

- **Intractable problems**
  - can not be solved with a computer of
  reasonable size and cost in reasonable
  amount of time
  - e.g. *Euler Circuit Problem*
    - transit all edges of a graph once and end up at the
      starting point
    - generating and checking every permutations
grows exponentially with the number of nodes
  - problems with solution methods that grow
    exponential are usually intractable

- **Speed limitations**
  - an algorithm of time complexity of order $f(n)$, $O(f(n))$
  - e.g. *The Travelling Salesman Problem*
    - visit all nodes of an network with minimum distance
  - use heuristic techniques to approximate solutions
    use parallel processing to reduce time complexity
Historical Highlights

**Traditional**

- **1700s**
  - Chinese Abacus

- **1600s**
  - Napier's Bones

- **1700s**
  - Jacquard Loom
Historical Highlights

**1800s**

- Babbage's Difference Engine
- Hollerith Tabulation

**1900s**

- ENIAC: Electronic Numerical Integrator and Calculator
- Altair 8800
Historical Highlights

2000s

Cray X-MP

Happy New Year
Babbage's and His Engines

• Charles Babbage (1791-1871) was a product of the industrial revolution
  – explosive growth in the need for pre-calculated numeric tables
  – driven by the growth in industrial technologies

• Designed several table-making Difference Engines, and prepared plans for far more ambitious Analytical Engines
  – both were flexible and powerful, punched-card controlled general purpose calculators
  – embodied many features of a modern stored program computer including punched card control; separate store and mill; internal registers; fast multiplier/divider; a range of peripherals; array processing

  – a complete version of DE2 was built in 1973
  – the more ambitious task of constructing a working Analytical Engine remains to be undertaken
Babbage's and His Engines 2

- Babbage was also an important economist and technology theorist
  - first advanced factories as the key generator of national wealth vice Adam Smith who thought it was agriculture
  - pre-dated John Stuart Mill in the theory of division of labor
  - advocated technology-based production as fundamental to socio-economic development
  - developed mathematical techniques in codebreaking

- In 1833, Augusta Ada Byron, Countess of Lovelace met Babbage and was fascinated with him and his engines
  - she translated and annotated a paper on Babbage's engines which constitute their best contemporary description of the engines and their capabilities
  - Not accurate to call Ada the world's first programmer vice the world’s first documentation writer

The idea for the novel Frankenstein arose in the summer of 1816 when Mary Shelley was staying at Lord Byron’s (Augusta Ada’s father) villa in Geneva, Switzerland.
Programming the Analytical Engine 1

- Programs were to be punched on pasteboard Jacquard-type cards
  - Babbage envisioned three different kinds of cards each with a different format and its own independent reader
  - the separation of card types simplified the mechanics of the Engine

- Operation Cards
  - correspond to the "operation codes" in the instruction set of modern computers
  - Operations Cards command the Mill to perform the various arithmetic operations: Addition, Subtraction, Multiplication, and Division
  - an Operation card used four holes to encode the arithmetical operations

- Combinatorial Cards and Index Cards advance or back the chain of cards in the reader, correspond to the jump/branch and loop control instructions of modern computers
Programming the Analytical Engine 2

- **Number Cards**
  - supply numerical constants punched upon them to the Store as required
  - permits more constants to be used in a computation than can be contained in the Store
  - included the result of previous calculations that were punched by the Card Punching Apparatus
  - a Number card used 50 columns of 10 holes for the digits plus a column to indicate the sign

- **Variable Cards**
  - direct the transfer of values from the Store into the Mill to serve as arguments to an operation, and the transfer of the result of a computation by the Mill back to one or more locations in the Store
  - can, when transferring a value to the Mill, either zero the column in the Store or leave it as before
  - specified the location in the Store, and the source or destination axis in the Mill, and whether, on a transfer to the Mill, the column in the Store is to be zeroed
The Turing Machine

• The Turing Machine was based on the philosophical question:
  – could any definite method or process exist by which all mathematical questions could be decided?

• Turing proved that a finite set of rules could theoretically be written to solve any problem that is humanly solvable

• Triple correspondence between logical instructions, the action of a human mind, and a machine which could in principle be embodied in a practical physical form:
  – was Turing's definitive contribution to computer science and specifically Programming Languages

• Turing’s theoretical and practical work significantly contributed to the Allies’ ability to break Nazi codes during World War II

Alan Turing died in 1954 by eating a poison apple.
In 1936 Alan Turing, a British Mathematician, developed a theoretical machine

- the machine would carry out all computations on numbers and symbols
- required a set of rules describing the computation along with the data

Turing's Machine is the cornerstone of the modern theory of computation and computability

The Turing Machine consists of

- an **Input/Output Tape**,
- the **Turing Machine** itself,
- and a **Rule List**
The Turing Machine 3

• The Input/Output Tape
  – infinitely long and can be wound forwards and backwards
  – divided into cells containing symbols and

• The Turing Machine
  – can be conceived as a mechanical 'black box' that reads/writes symbols the tape one at a time from its Read/Write head
  – the machine is always in a particular internal State indicated by a number on the box.

• The Rule List is what determines the Machine's move at any particular point.

• The Turing Machine reads a symbol from the Input/Output Tape and consults its Rule List. It then performs two actions.
  1. It modifies its internal State
  2. It writes a symbol on the tape
     Or
     Moves its Read/Write head left or right.
The Turing Machine 4

• The tape head of the Turing Machine is capable of only three actions:
  – **Write** on the tape (or erase from tape), only on the section being viewed
  – **Change** the internal state
  – **Move** the tape 0 or 1 space, to the left or right

• The Turing Machine is a finite state machine or a finite automaton
  – it separates information into two elements - that from its internal state, and that which is derived externally
  – at any particular moment in time, it is in a describable state
  – between this moment and the next discrete time step, the machine reads its input from the tape, refers to rules controlling its behavior, and considering both the input and its own current state, in determining what behavior to exhibit

*For example*

If the machine is in State 10 and the Read/Write head is positioned on the letter 'A'. The Turing Machine Rule List might specify:

```
10 1 11 R
```

This rule says: If you are in State 10 and reading an '1' change to State 11 and advance the tape head to the right.
A Simple Turing Machine

Inputs = \{0, 1, 2\}
States = \{\alpha, \beta, \text{halt}\}
Directions = \{L, R, N\}

<table>
<thead>
<tr>
<th>States</th>
<th>Inputs</th>
<th>α</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1 R</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>halt N</td>
<td></td>
</tr>
</tbody>
</table>

Initial configuration

---
2 2 1 1 1 0 2 2
---

Read/write head

State \(\alpha\)
The von Neumann Architecture

- John von Neumann
  - Hungarian-American
    - mathematician and physicist
    - promoter of the stored program concept
    - contributed to modern gaming theory

- In a computer with a von Neumann architecture
  - both data and programs are stored in the same memory
  - the central processing unit (CPU), which actually executes instructions, is separate from the memory
  - instructions and data must be piped, or transmitted, from memory to the CPU.
  - results of operations in the CPU must be moved back to memory.
  - nearly all digital computers built since the 1940s have been based on the von Neumann architecture

John von Neumann was almost denied U.S. citizenship when he began to argue logical inconsistencies within the U.S. Constitution with immigration officials.
The Von Neumann computer
Architecture

Central Processing Unit

Arithmetic and logic unit

Control Unit

Input and output devices

“Secondary Storage”

Instructions and data

Results of operations

Memory (stores both instructions and data)
“Primary Storage”

Copyright 1999 ©Roy M. Wnek. All rights reserved.
The von Neumann Architecture

- Most programming languages have been designed for computers using the von Neumann architecture.

- Because of the von Neumann architecture, most computer programming languages are imperative in nature.

- Central features of imperative languages are:
  - variables which model the memory cells
  - assignment statements, which are based on the piping operation
  - the iterative or looping form of repetition
    - The most efficient method because the instructions in a von Neumann computer are stored in adjacent cells of memory

- Operands in expressions are piped from memory to the CPU, and the result of evaluating the expression is piped back to the memory cell represented by the left side of the assignment.

- “The von Neumann bottleneck”
Physical Limitation to Computer Performance

- Navy Rear Admiral Grace Hopper
  - early computer programmer,
    worked on the Harvard Mark I
  - advocate for standardized COBOL
  - lectured on computer performance barriers

- Computer performance is bounded by the speed of light
  - using electronic signals to transfer data places a speed limit on that movement
  - conversely, electronic data transfer takes a finite amount of time
    - therefore physical size limits how fast a computer can operate
    - parallelism is needed to help overcome size / speed limitations
  - Grace Hopper’s lectures featured her handing out shorts lengths of wire which represented the distance that electrons could travel in 1 nanosecond

Grace Hopper discovered the first computer bug in 1946.