CSC104 winter 2013
Why and how of computing
week 3

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/104/W13/
416-978-5899

Text: Picturing Programs
Clearly there’s no fool-proof method, but there’s some techniques that often make progress. It helps to write down the whole process:

- Understand the problem
- Devise (one or more) plan(s)
- Try the plan
- Look back
paper folding?
try it out

- Understand the problem (what’s given, what’s required)?
- Devise a plan
- Try at least one plan (be ready to abandon it too)
- Look back
Representing even simple information is hard. Let’s race through this table:

<table>
<thead>
<tr>
<th>Bits</th>
<th>b₇ b₆ b₅ b₄ b₃ b₂ b₁</th>
<th>0 0 0 0</th>
<th>0 0 0 1</th>
<th>0 0 1 0</th>
<th>0 0 1 1</th>
<th>0 1 0 0</th>
<th>0 1 0 1</th>
<th>0 1 1 0</th>
<th>0 1 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td>NUL DLE SP</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>`,</td>
<td>p</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 1 1 1</td>
<td>SOH DC1 I</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 2 2</td>
<td>STX DC2 &quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 1 3 3</td>
<td>ETX DC3 #</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 4 4</td>
<td>EOT DC4 $</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 1 5 5</td>
<td>ENQ NAK %</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 0 6 6</td>
<td>ACK SYN &amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1 1 7 7</td>
<td>BEL ETB `'</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 0 8 8</td>
<td>BS CAN (</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 1 9 9</td>
<td>HT EM )</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 0 10</td>
<td>LF SUB `</td>
<td></td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1 1 11</td>
<td>VT ESC +</td>
<td></td>
<td>K</td>
<td>[</td>
<td>k</td>
<td>{</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 12</td>
<td>FF FC .</td>
<td></td>
<td>&lt;</td>
<td>L \</td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 1 1 0 1 13 | CR GS - | | = | M ] | m | }
| 1 1 1 0 14 | SO RS > | | N ^ | n | ~ |
| 1 1 1 1 15 | SI US / | | ? | O _ | o | DEL |
Early devices

Tally systems

Clay tablets, read-only when baked, read/write when sundried, have been in use for at least 5,000 years.

Abacuses, or abaci, have been in use for nearly as long.
Number systems and gears
ancient world

Hindu-Arabic numbers: positional notation, and zero over 2000 years ago
slick algorithms, e.g. long multiplication

Antikythera mechanism
make us re-think
ancient technical skills
Gears and rules
machine age

Add powers (logs) to multiply quickly, extract roots

Read the gears to extract taxes — Pascaline
Looms and engines
industrial revolution

Jacquard loom combined steam and punch cards for automatic patterns

Babbage's difference engine would have evaluated polynomials like $3x^3 + 5x^2 - 7x + 9$
gears, pins, and electricity
digital and analog before tubes

data stored in punched cards
manipulated by pins and
electricity last for decades

analog computers model world
using smoothly-varying quantities
such as water
programmable or electronic... 
...but not both?

“programmable” (cards) but not electronic (relays) 
the Zuse Z1

electronic but not programmable dedicated to one calculation 
the Atanosoff-Berry
when computers were women for a while

Eniac’s first programmers were women known for clear-thinking, manual dexterity, and speed. Human labour was cheaper than computer cycles. Dozens of cubic metres were programmed by connecting pins.
stored programs, faster switches

getting modern

the same memory for data and programs is now the typical design

tubes were big, hot, slow compared to transistors ... which just keep shrinking
your (grand)parent’s computer
smaller, faster . . .

perhaps thanks to sputnik
the computing power of eniac
fits in your hand by 1970

mass-produced desktops
landed with a clunk
by 1980s