Level ISA3: Information Representation

Information as electrical current

- At the lowest level, each storage unit in a computer’s memory is equipped to contain either a high or low voltage signal
- Each storage unit exists, therefore, in one of two states, each of which is discrete
- Since two values are possible, and these values are usually represented by numeric values, we refer to the storage units as binary digits, abbreviated bits

Memory cells & data storage

- Computer memory is organized into groups of bits called cells
- The number of bits in a cell may vary across hardware platforms, although the 8-bit byte is the most common
- All data and instructions must be represented in binary form in order to be stored in a computer’s memory

Integer storage

- The simplest code for integer storage is unsigned binary form
- Binary is to base 2 as decimal is to base 10
- The binary system is similar to the decimal system, in that it uses positional notation to represent the magnitude of a number
- In decimal, each digit’s position represents a power of 10; in binary, each position is a power of 2
- Any value can be used as the base number for integer representation; the next slide illustrates this

Equivalent values in different bases

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Value</th>
<th>Base 4</th>
<th>Base 6</th>
<th>Base 8 (Octal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0010</td>
<td>02</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0011</td>
<td>03</td>
<td>03</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0100</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0101</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0110</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0111</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1000</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1001</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td>1010</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
<td>1011</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>1100</td>
<td>1100</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>1101</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
<td>1110</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>1111</td>
<td>33</td>
<td>33</td>
</tr>
</tbody>
</table>

Hexadecimal numbers

- Numeric representation with a base number (called the radix) greater than 10 is also possible
- Of particular interest in computer systems is radix 16, or hexadecimal numbers
- As with other bases, base 16 numbers use positional notation
- Each position represents a power of 16
- The symbols employed include the digits 0…9 and the letters A (for decimal value 10) through F (decimal 15)
Hexadecimal / Decimal Equivalence

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
</tr>
<tr>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
</tr>
</tbody>
</table>

Hexadecimal / Binary Equivalence

Hexadecimal numbers provide convenient abbreviations for equivalent binary values.

Each hex digit represents 4 binary digits.

Octal numbers can also be used to abbreviate binary (with 1 octal digit representing 3 bits) but hex is far more common in modern computer systems.

Converting from one base to another

- The next several slides describe different methods by which to convert from one base to another.
- It is especially useful to be able to convert back and forth between binary, decimal and hexadecimal notations.

Converting from any base to decimal

- The value of a number in any base is the sum of the values of each digit multiplied by successive powers of the base, we can easily convert a number from any other base to decimal if we know the powers of the base.
- The rightmost digit will always be multiplied by 1, since any number to the 0 power is 1.
- For a 5-digit number with base n, for example, its decimal equivalent would be: digit1 x n^4 + digit2 x n^3 + digit3 x n^2 + digit4 x n + digit5

Positional notation

1101 base 10 =
1 x 10000 + 1 x 1000 + 1 x 100 + 0 x 10 + 1 x 1
1101 base 2 =
1 x 16 + 1 x 8 + 1 x 4 + 0 x 2 + 1 x 1 (29_{10})
1101 base 4 =
1 x 256 + 1 x 64 + 1 x 16 + 0 x 4 + 1 x 1 (337_{10})
1101 base 8 =
1 x 4096 + 1 x 512 + 1 x 64 + 0 x 8 + 1 x 1 (4673_{10})

Converting from hex to decimal

- You can use positional notation when converting from hex to decimal as well, but you need to remember to use the decimal values of the digits beyond 9 (A=10, B=11, C=12, D=13, E=14, F=15), multiplying the digit’s value by the power of 16 indicated by the digit’s position.
Example

The decimal value of F78EC2 is:

| 15 x 16^7 | (15,728,640) |
| 7 x 16^6 | (458,752) |
| 8 x 16^5 | (32,768) |
| 14 x 16^4 | (3,584) |
| 12 x 16^3 | (192) |
| 2 | (2) |

______________________________

16,223,938_{10}


Division algorithm for converting decimal value n to base b

- Divide n by b, obtaining quotient & remainder:
  \[ n = bq_0 + r_0 \] where \( 0 \leq r_0 < b \)
  remainder (r_0) is rightmost digit in base b expansion
- Divide q_0 by b, obtaining:
  \[ q_0 = bq_1 + r_1 \] (0 <= r_1 < b)
  r_1 is second digit from right in base b expansion
- Continue successive divisions until we obtain a q = 0


Example

Find octal (base 8) expansion of 4745_{10}

\[
\begin{align*}
4745 &= 8 \times 593 + 1 \text{ (rightmost digit)} \\
593 &= 8 \times 74 + 1 \\
74 &= 8 \times 9 + 2 \\
9 &= 8 \times 1 + 1 \\
1 &= 8 \times 0 + 1 \text{ (leftmost digit)}
\end{align*}
\]

Result is 11211_{8}


C++ for base expansion algorithm (for base <= 10)

```cpp
int baseExpand(int n, int b)
{
    int k = 0, digit, expansion = 0;
    while (n != 0)
    {
        digit = n % b;
        n = n / b;
        expansion = expansion + digit * pow(10,k);
        k++;
    }
    return expansion;
}
```


Converting binary to hex

- The easiest way to convert binary numbers to hex involves memorizing the table of equivalence given in slide 8
- An awareness of the hex/binary equivalence will also help keep you mindful of the equivalent decimal values for both bases
- If you know the equivalences, it is easy to apply the conversion method given on the next slide


Converting binary to hex (and vice-versa)

- Break the binary expansion into four digit groupings (hextets), starting from the right; expand the leftmost hextet with leading 0s, if necessary
- Substitute the hexadecimal digit corresponding to each hextet
- The reverse is also easy; simply convert each hex digit into its equivalent 4-bit group to convert from hex to binary
Examples
Convert binary value 1001000011011110111011 to its hex equivalent:

Breaking the expansion into groups of 4 digits, we have:
0100 1000 0111 0111 1011 or:
4 8 7 7 B

Convert hexadecimal value 92AF to its binary equivalent:
9 = 1001
2 = 0010
A = 1010
F = 1111
Value is 1001 0010 1010 1111

Another method for converting binary to decimal

- The fastest way to convert binary to decimal is a method called double-dabble (or sometimes double-dibble)
- This method uses the idea that a subsequent power of 2 is double the previous power of 2 in a binary number
- Starting with the leftmost bit and working to the right:
  - Double the first bit and add it to the second
  - Double the sum and add it to the next bit
  - Repeat for each bit until the rightmost bit is used

Convert 10010011 to decimal

Step 1: write the binary expansion, leaving spaces between digits:

1 1 0 0 0 1 1

Step 2: double the high-order bit and copy it under the next bit:

1 1 0 0 0 1 1 1

Step 3: Add the next bit and double the sum; copy result under next bit:

1 1 0 0 0 1 1 1

Step 4: Repeat step 3 until you run out of bits (see next slide)

Double-dabble conversion of binary to decimal (result)

Original value:

1 0 0 1 0 0 1 1 1

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>18</th>
<th>36</th>
<th>72</th>
<th>146</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>1</td>
<td>+</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

Final result

Hex to decimal conversion

- You can combine hex to binary and double-dabble to do hexadecimal to decimal conversion
- For example, to convert 02CA₁₆ to decimal:
  - Convert hex to binary by grouping into hexets:
    0000 0010 1100 1010
  - Apply double-dabble method on the binary form

Converting fractions

- Fractions in any base system can be approximated in any other base system using negative powers of the radix (radix point separates the whole part of a number from its fractional part; for decimal numbers, we call this a decimal point)
- One of the previous algorithms for conversion of whole numbers used division with remainder to find the digits of the converted base expansion; a similar method for converting fractional numbers uses multiplication by the radix, with the whole part of each product becoming a digit of the result
### Convert 0.4304₇ to base 5:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Base 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4304₇</td>
<td>0.20₀₅</td>
</tr>
</tbody>
</table>

1. **First digit is 2:**

   
   \[
   \begin{align*}
   0.4304 & \times 5 \\
   2.1520 & \text{first digit is 2; use fractional part for next calculation} \\
   \end{align*}
   \]

2. **Second digit is 0:**

   \[
   \begin{align*}
   0.1520 & \times 5 \\
   0.7600 & \text{second digit is 0; continuing …} \\
   \end{align*}
   \]

3. **Third digit is 3:**

   \[
   \begin{align*}
   0.7600 & \times 5 \\
   3.8000 & \text{third digit is 3; continuing…} \\
   \end{align*}
   \]

4. **Fourth digit is 4,** and fractional part is now 0; done

Base 5 equivalent of is 0.20₀₅

---

### Notes on fractional conversion

- Things don’t always work out as neatly as in the previous example
- For example, we might end up with repeating fractions, or we may simply run into a hard limit on the number of digits that can be stored
- Most computer systems implement specialized rounding algorithms to provide a predictable degree of accuracy

---

### Addition with unsigned binary numbers

- The rules for adding bits are:
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 10$

- In the last instance, the result is a place-holder with a carry bit; if two numbers add up to a sum greater than 1, you must carry a 1 to the next column

---

### Example

Calculate the sum of 1011₂ and 0010₂:

\[
\begin{align*}
1011 & \\
+ 0010 & \\
\hline
1101 & \text{(result)}
\end{align*}
\]

---

### The Carry Bit

- Every computer system has a hard limit on the amount of memory used to represent a binary integer
- When two numbers are added, even if each of their values fits within this limit, the value of the sum may be too large to fit into the available space
- To flag this possibility, the CPU contains a special bit called the carry bit
- If the sum of the leftmost digits (most significant bits) of two numbers produces a carry, the carry bit is set to 1, indicating the result was out of range

---

### The carry bit and subtraction

- Instead of carries, binary subtraction (like decimal subtraction) is performed with borrows
- When a larger unsigned number is subtracted from a smaller unsigned number, the result would be negative, except there’s no such thing as a negative unsigned number
- In other words, the result value would be out of range, and the carry bit is set to reflect this error condition