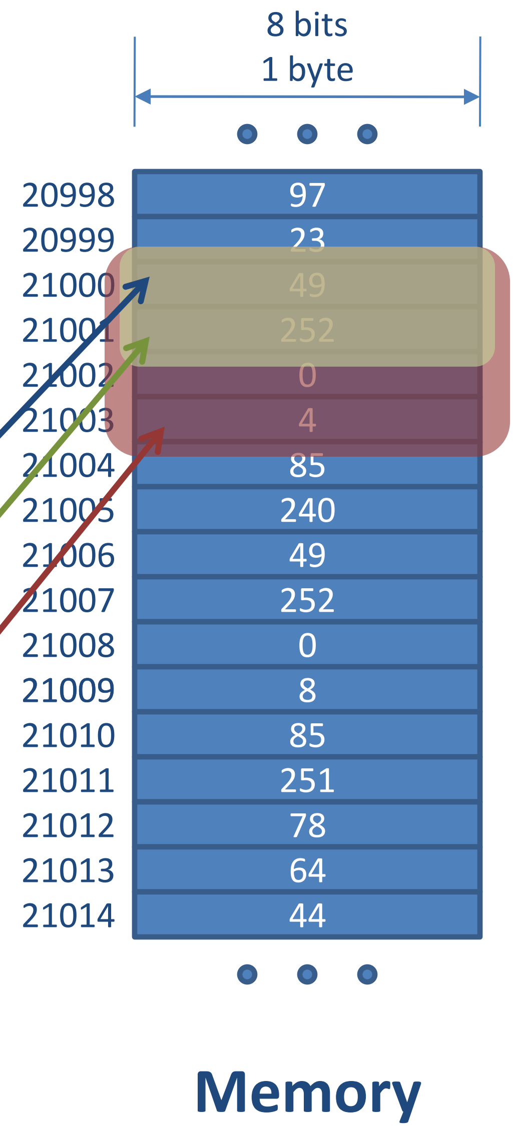
Memory Structure

- Memory is implemented as an array of electronic switches
  - Each switch can be in one of two states
  - **0** or **1**, **on** or **off**, **true** or **false**, **purple** or **gold, sitting** or **standing**
  - **BI**nary digi**T**s (**bits**) are the fundamental unit of data storage in a computer
  - Accessing each bit individually isn't very useful
  - We want to store data that can take a wider range of values
    - the value 214
    - the letter "b"
    - 

Memory Structure

- By grouping bits together we can store more values
  - 8 bits = 1 **byte**
  - 16 bits = 2 bytes = 1 **halfword**
  - 32 bits = 4 bytes = 1 **word**
- When we refer to memory locations by address (using the ARM7), we can only do so in units of **bytes**, **halfwords** or **words**
  - the byte at address 21000
  - the halfword at address 21000
  - the word at address 21000



Larger Units of Information Storage

- Larger units of information storage
  - 1 **kilobyte** (kB) = 2^{10} bytes = 1,024 bytes
  - 1 **megabyte** (MB) = 1,024 KB = 2^{20} bytes = 1,048,576 bytes
  - 1 **gigabyte** (GB) = 1,024 MB = 2^{30} bytes = ...
- The following units of groups of bits are also used, usually when expressing **data rates**:
  - 1 **kilobit** (kb) = 1,000 bits
  - 1 **megabit** (Mb) = 1,000 kilobits = 1,000,000 bits
- IEC prefixes, KiB, MiB, GiB, ...

Numeral Systems

- We use the decimal (base-10) numeral system most frequently
  - We have symbols (digits) that can represent ten integer values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - We represent integer values larger than 9 with combinations of two or more digits, e.g.: 10, 11, 12, ..., 112, ..., 247
  - e.g.: 247
    - $= (7 \times 10^0) + (4 \times 10^1) + (2 \times 10^2)$
    - 2 is the **Most Significant Digit**
    - 7 is the **Least Significant Digit**
  - Given *n* decimal digits, how many different integer values can we represent?
    - $[0 \ldots (10^n - 1)]$
    - e.g., $n = 3$ allows us to represent values in the range [0 ... 999]

Binary Numeral System (Base-2)

- Computer systems store information electronically using bits (binary digits)
  - Each bit can be in one of two states, which we can take to represent the binary (base-2) digits 0 and 1
  - The **binary** number system is a natural number system for computing (rather than the decimal system)
  - Using a single bit, we can represent integer values 0 and 1
  - Using two bits, we can represent 0, 1, 10, 11
    - i.e. four different values
  - How many unique values can we represent with, for example, eight bits?
    - [0 ... 11111111] in binary notation
    - [0 ... (2^8 - 1)] = [0 ... 255] in decimal notation
    - 256 unique values

Base Subscript Notation

There are 10_2 types of people in the world: those who understand binary and those who don't ...

- The same sequence of symbols can have a different meaning depending on the base being used
  - We can use the subscript notation to make it clear which base we are using
  - $12_{10} = 1100_2$
  - $1_{10} = 1_2$

- Using binary all the time would become quite tedious
  - The 3D1 exam is worth 1010000_2% of the final mark

Binary to Decimal Conversion

- Convert %100101 to its decimal equivalent

100101 =	(1×2^0) +
	(0×2^1) +
	(1×2^2) +
	(0×2^3) +
	(0×2^4) +
	(1×2^5) +
	= 37

Decimal to Binary Conversion

- Convert 37 to its binary equivalent?

	Quotient	Remainder	Binary digit
37 / 2 =	18	1	1
18 / 2 =	9	0	0
9 / 2 =	4	1	1
4 / 2 =	2	0	0
2 / 2 =	1	0	0
1 / 2 =	0	1	1

Hexadecimal Numeral System (Base 16)

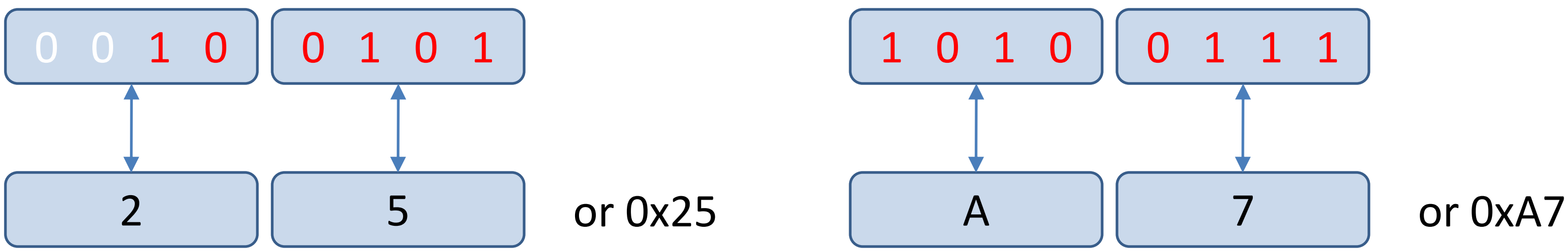
- Base-16 (**hexadecimal**) is a more convenient number system for computer scientists:
  - With binary, we needed 2 symbols (0 and 1)
  - With decimal, we needed 10 symbols (0, 1, ..., 9)
  - With hexadecimal, we need 16 symbols
  - Use the same ten symbols as the decimal system for the first ten hexadecimal digits
  - "Borrow" the first six symbols from the alphabet for the last six symbols
  - ☺, ⚐, ✋, ☯, 🕯, ☎, ★, ➚, ☑, ❖, ■, ●, ❀, 🕑, 🗏, ✂
- Why is hexadecimal useful?
  - 16 is a power of 2 (2^4), so exactly one "**hex**" digit can represent the same sixteen possible values as four bits

Decimal, Binary and Hexadecimal

base-10	base-2	base-16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Converting Between Binary and Hex

- One hexadecimal digit represents the same number of values as four binary digits
  - ⇒conversion between hex and binary is trivial
  - ⇒the hexadecimal notation a convenient one for us



- Hexadecimal is used by convention when referring to memory addresses: e.g. address 0x1000, address 0x4002

"0x" denotes a hexadecimal value

Alternative Base Notation

- Without a fancy word processor, we won't be able to use the subscript notation to represent different bases
  - How would we tell a computer whether we mean 10 or 10?
- Instead we can prefix values with symbols that provide additional information about the base
- In ARM assembly language (which we will be using) we use the following notation:
  - 1000 No prefix usually means decimal
  - **0x**1000 Hexadecimal, used frequently
  - **&**1000 Alternative hexadecimal notation
  - **2_**1000 Binary
  - **n_**1000 Base n

Representing Text

- So far, we have only considered how computers store integer values using binary digits
- What about representing other information, for example text composed of alphanumeric symbols?

‘T’, ‘h’, ‘e’, ‘ ‘, ‘q’, ‘u’, ‘i’, ‘c’, ‘k’, ‘ ‘, ‘b’, ‘r’, ‘o’, ‘w’, ‘n’, ‘ ‘, ‘f’, ‘o’, ‘x’, ...

- We’re still restricted to storing binary digits (bits) in memory
- To store alphanumeric symbols or “**characters**”, we can assign each character a value which can be stored in binary form in memory

ASCII

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- ASCII is a standard used to encode alphanumeric and other characters associated with text
  - e.g. representing the word “hello” using ASCII

‘H’	‘E’	‘L’	‘L’	‘O’

$48	$45	$4C	$4C	$4F

- Each character is stored in a single byte value (8 bits)
  - 1 byte = 8 bits means we can have a possible 256 characters
  - In fact, ASCII only uses 7 bits, giving 128 possible characters
  - Only 96 of the ASCII characters are **printable**
  - Remaining values are **control codes** – examples??

ASCII

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>0</td><td>NUL</td><td>DLE</td><td>SPACE</td><td>0</td><td>@</td><td>P</td><td>`</td><td>p</td></tr>
<tr><td>1</td><td>SOH</td><td>DC1</td><td>!</td><td>1</td><td>A</td><td>Q</td><td>a</td><td>q</td></tr>
<tr><td>2</td><td>STX</td><td>DC2</td><td>“</td><td>2</td><td>B</td><td>R</td><td>b</td><td>r</td></tr>
<tr><td>3</td><td>ETX</td><td>DC3</td><td>#</td><td>3</td><td>C</td><td>S</td><td>c</td><td>s</td></tr>
<tr><td>4</td><td>EOT</td><td>DC4</td><td>$</td><td>4</td><td>D</td><td>T</td><td>d</td><td>t</td></tr>
<tr><td>5</td><td>ENQ</td><td>NAK</td><td>%</td><td>5</td><td>E</td><td>U</td><td>e</td><td>u</td></tr>
<tr><td>6</td><td>ACK</td><td>SYN</td><td>&</td><td>6</td><td>F</td><td>V</td><td>f</td><td>v</td></tr>
<tr><td>7</td><td>BEL</td><td>ETB</td><td>‘</td><td>7</td><td>G</td><td>W</td><td>g</td><td>w</td></tr>
<tr><td>8</td><td>BS</td><td>CAN</td><td>(</td><td>8</td><td>H</td><td>X</td><td>h</td><td>x</td></tr>
<tr><td>9</td><td>HT</td><td>EM</td><td>)</td><td>9</td><td>I</td><td>Y</td><td>i</td><td>y</td></tr>
<tr><td>A</td><td>LF</td><td>SUB</td><td>*</td><td>:</td><td>J</td><td>Z</td><td>j</td><td>z</td></tr>
<tr><td>B</td><td>VT</td><td>ESC</td><td>+</td><td>;</td><td>K</td><td>[</td><td>k</td><td>{</td></tr>
<tr><td>C</td><td>FF</td><td>FS</td><td>,</td><td><</td><td>L</td><td>\</td><td>l</td><td>|</td></tr>
<tr><td>D</td><td>CR</td><td>GS</td><td>-</td><td>=</td><td>M</td><td>]</td><td>m</td><td>}</td></tr>
<tr><td>E</td><td>SO</td><td>RS</td><td>.</td><td>></td><td>N</td><td>^</td><td>n</td><td>~</td></tr>
<tr><td>F</td><td>SI</td><td>US</td><td>/</td><td>?</td><td>O</td><td>_</td><td>o</td><td>DEL</td></tr>
</table>

ASCII

- Some things to note about ASCII
  - The value 0 is not the name as the character ‘0’
  - Similarly, the value 1 is not the same as the character ‘1’, ...
  - The characters ‘0’, ‘1’, ... are used in text to display values in human readable form
  - Upper and lower case characters have different codes
  - The first printable character is the space symbol, ‘ ’ and it has code 32_{10}
  - ASCII is a base-128 number system and each value has a different symbol
  - It is more efficient to store a value in its value form than its text form (i.e. Store the value 10_{10} instead of the ASCII symbols ‘1’ followed by ‘0’.)

Types of Memory

- **Random Access Memory (RAM)**
  - The CPU can both read and modify the contents of memory
  - **Volatile** (information lost when power is turned off)
  - **Non-volatile** (information retained when power is turned off)
- **Read Only Memory (ROM)**
  - CPU can only read the contents of memory
  - Non-volatile
  - Further classified by ability to change contents
    - Programmable ROM (**PROM**): Can only write contents once
    - Erasable Programmable ROM (**EPROM**): Can erase contents using ultra-violet light and re-write
    - Electronically Erasable Programmable ROM (**EEPROM**): Can erase contents electronically (including "**Flash Memory**")