Arithmetic and Logical Operations: ALU

Gabriel Hugh Elkaim
MAGIC NUMBER:

Announcements 378 / 402

MIPS Assembly Lab

Incremental Development

Git commit/push/submit

Reading: Kann ch. 2, 3, 47.
Integer Numbers

N-bits unsigned  $\rightarrow$ $0 - (2^n - 1)$

- 8 bits = $0 - 255$
- 16 bits = $0 - 65535$

N-bits signed  $(2S)$  $\rightarrow$

- $-2^{(n-1)}$ $\rightarrow$ $2^{(n-1)} - 1$

- 8 bits = $-128 \rightarrow +127$
- 16 bits = $-32k \rightarrow +32k$
Character Representation (1.3)

Memory location for a character usually contains 8 bits:

0000 0000 → 1111 1111
0x00 0xFF

A-Z
a-z
0-9
[!@#$%^&*()]

Special characters {\, \, \}

Alphabetical order:

DOES \, COME \, BEFORE \, OR \, AFTER \, 2C?

Gabriel Hugh Elkaim
Character Representation (2.3)

ASCII - American Standard Code for Information Interchange

Each character is mapped to a specific bit pattern.

0100 0001 → 0x41 → C5₁₀ → “A”
0100 0010 → 0x42 → C6₁₀ → “B”
Character Representation (3.3)

ASCII, THE AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE.
ASCII Properties (1.2)

IF THE BIT PATTERN IS COMPARED (TREAT LETTER AS UNSIGNED NUMBERS)

'A' < 'B'
65 < 67

'a' is 0x61 ≠ 'A' is 0x41

"x" (0x38) ≠ 8.

x = c - 'φ'
## ASCII Properties (2.2)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Ch</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Ch</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td> </td>
<td>64</td>
<td>40</td>
<td>100</td>
<td> </td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td> </td>
<td>65</td>
<td>41</td>
<td>101</td>
<td> </td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td> </td>
<td>66</td>
<td>42</td>
<td>102</td>
<td> </td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td> </td>
<td>67</td>
<td>43</td>
<td>103</td>
<td> </td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td> </td>
<td>68</td>
<td>44</td>
<td>104</td>
<td> </td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td> </td>
<td>69</td>
<td>45</td>
<td>105</td>
<td> </td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td> </td>
<td>70</td>
<td>46</td>
<td>106</td>
<td> </td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td> </td>
<td>71</td>
<td>47</td>
<td>107</td>
<td> </td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td> </td>
<td>72</td>
<td>48</td>
<td>110</td>
<td> </td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>HT (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td> </td>
<td>73</td>
<td>49</td>
<td>111</td>
<td> </td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td> </td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td> </td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td> </td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td> </td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td> </td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td> </td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td> </td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td> </td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td> </td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td> </td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td> </td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td> </td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td> </td>
<td>80</td>
<td>50</td>
<td>120</td>
<td> </td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td> </td>
<td>81</td>
<td>51</td>
<td>121</td>
<td> </td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td> </td>
<td>82</td>
<td>52</td>
<td>122</td>
<td> </td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td> </td>
<td>83</td>
<td>53</td>
<td>123</td>
<td> </td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td> </td>
<td>84</td>
<td>54</td>
<td>124</td>
<td> </td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td> </td>
<td>85</td>
<td>55</td>
<td>125</td>
<td> </td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td> </td>
<td>86</td>
<td>56</td>
<td>126</td>
<td> </td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td> </td>
<td>87</td>
<td>57</td>
<td>127</td>
<td> </td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>068</td>
<td> </td>
<td>88</td>
<td>58</td>
<td>130</td>
<td> </td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>069</td>
<td> </td>
<td>89</td>
<td>59</td>
<td>131</td>
<td> </td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>070</td>
<td> </td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td> </td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td> </td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td> </td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td> </td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td> </td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td> </td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td> </td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td> </td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td> </td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td> </td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td> </td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Special Characters

\n  \n  newline

\t\n  tab

\n  escape

\n  "Hello world"

\n  \n  NULL or NULL

\n  \n  0x00
## Logical Operations – 1 Bit

<table>
<thead>
<tr>
<th>In 0</th>
<th>In 1</th>
<th>AND</th>
<th>OR</th>
<th>NAND</th>
<th>NOR</th>
<th>XOR</th>
<th>XNOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Logical Operations – Many Bits

"Bitwise" logical operations, we do the logical operation in parallel for each bit.

\[ \begin{align*}
  \text{If } x &= 0011, \\
  \text{and } y &= 1010,
\end{align*} \]

\[ \text{\(x \land y\)} = 0011 \]
\[ \text{\(x \lor y\)} = 0010 \]

\[ \text{\(x \oplus y\)} = 1011 \]

\[ \text{\(x \Rightarrow y\)} = 1010 \]
\[ \text{\(x \Leftarrow y\)} = 1001 \]
Masking (1.3)

Want to look at certain bits

Use "mask" to remove uninteresting bits

VALUE: 1001 110

AND 0000 1111

RESULT 0000 1101

← MASK
Masking (2.3)

- Clean a bit
  - $1111 \rightarrow 0111$ (AND with $\overline{mnc}$)

- Set a bit
  - $0000 \times 0100$
  - $0000 \times 0100$

- Toggle a bit
  - Switch states
Masking (3.3)

\[ \text{If } (x < 0) \rightarrow \text{Test top bit} \times \begin{array}{c} 1000 \ 0000 \end{array} \land \frac{1}{2} \]
Overflow (1.4)

Carry out is usually thrown away.

"Overflow" = carry at bit = 1, cause invalid result.

\[ \begin{align*}
31 & \xrightarrow{+} 5 \\
\hline
0 & 4
\end{align*} \]
Overflow (2.4)

 UNSIGNED ADDITION

\[
\begin{array}{c}
1000 & 6 \\
+ & 1001 & 9 \\
\hline
0001 & \rightarrow & 1
\end{array}
\]

\[\text{Overflow}\]

"WEIRD"
Overflow (3.4)

2's Complement

When the signs of the addends are the same, but the result is different:

If signs of addends are different, no overflow is possible.
Overflow (4.4)

In subtraction:

Unsigned — if result would be negative

2's comp — never do subtraction, add instead.

"Underflow"
Sign Extension (1.4)

How to change smaller number of bits into a bigger container.

Common thing within the CPU.
SIGN EXTENSION (2.4)

UNSIGNED → Copy original bits into LSB's and put 0's everywhere above.

00001011
Sign Extension (3.4)

2’s Complement (2SC)

Copy lower n-1 bits into LSBs.

Replicate MSB to n other places.

\[
\begin{array}{c}
\ 1011 \\
\ 11111011
\end{array}
\]
Sign Extension (4.4)

1 bit 2sc

4 bits → 8 bits

0011 (3)

0000 0011 (3)

1 1 0 0

1 1 0 1 (-3)

\[ 0 \quad -1 \]

\[ 0 1 1 \quad -2 + 1 = -1 \]

\[ 0 1 1 \quad -4 + 2 + 1 = -1 \]

MSB show bit \(-2^{n-1}\) plus

\[ -2^n + 2^{n-1} = -2^{n-1} \]
Multiplication (Base 10)

\[
\begin{align*}
12 & \times 4 \\
\underline{\times} & \\
48 & \\
\end{align*}
\]

\[
\begin{align*}
40 & \times 12 \\
\underline{\times} & \\
80 & \\
+ 400 & \\
\underline{+} & \\
480 & \\
\end{align*}
\]
Multiplication (Any Base)

 multiplicand \times \text{multiplier} \rightarrow \text{product, sum of partial products}

\[
\begin{array}{c}
0011 \\
0110
\end{array}
\]

\[\begin{array}{c}
00000 \\
00110 \\
01110 \\
01000 \\
00000
\end{array}\]

\[= 165.5\]
Unsigned Binary Multiplication (1.4)

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 0 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Unsigned Binary Multiplication (2.4)

\[ 1010 \times 10 = 101000 \]
Unsigned Binary Multiplication (3.4)
Unsigned Binary Multiplication (4.4)
2SC Binary Multiplication (1.4)

2. IF multiplier is **NEGATIVE**, sign extend it.

1. IF multiplier is **NEGATIVE**, take 2SC of both

\((-7 \times -3) = 7 \times 3 \quad 7 \times -3 = -7 \times 3\)

\[
\begin{array}{c}
1101 -3 \\
0101 5 \\
\end{array} \quad \begin{array}{c}
0011 3 \\
1011 -5 \\
\end{array} \quad \begin{array}{c}
1111111001 \\
11111111001 \\
000000000000 \\
111111101000 \\
\end{array}
\]
2SC Binary Multiplication (2.4)

\[ \begin{array}{cccc}
0011 & - & 1101 & (-3) \\
\hline
0110 & x & 1010 & (-6) \\
\hline
\end{array} \]

\[ \begin{array}{cccc}
0011 \\
\hline
0110 \\
1000 \\
1000 \\
\hline
0010 \\
0000 \\
0000 \\
0000 \\
\hline
00010000 \\
\end{array} \]
2SC Binary Multiplication (3.4)

0011 (3)  x  11111010  (-6)
1010 (-6)  

0001 0010
2SC Binary Multiplication (4.4)
Division (Base 10)

\[ 10 \div 5 \]

\[ 6 \overline{)65} \]

\[ 6 \]

\[ 05 \]

\[ 32 \overline{)85} \]

\[ 64 \]

\[ 21 \]
Division (Any Base)

- **Quotient**
- **Dividend**
- **Divisor**
- **Remainder**

\[
\text{Dividend} \div \text{Divisor} = \text{Quotient} \cdot \text{Divisor} + \text{Remainder}
\]
Unsigned Binary Division (1.2)

- Only going to require UNSIGNED division

\[
\begin{array}{c}
\frac{14}{2} \\
\end{array}
\]

\[
\begin{array}{c}
\frac{14}{2} \\
2 \overline{14} \\
- 7 \text{ remainder 0}
\end{array}
\]

\[
\begin{array}{c}
0111 \quad (7) \\
\end{array}
\]

\[
\begin{array}{c}
(2) \quad 10 \overline{1110} \quad (14) \\
- 10 \\
\hline
- 110 \\
- 10 \\
\hline
10
\end{array}
\]
Unsigned Binary Division (2.2)

\[
\begin{array}{c}
15 \\
\hline
3 \\
\end{array}
\]

\[
3 \sqrt{15} \\
\hline
15 \\
\hline
15
\]

\[
0101 \\
\hline
11 \\
\hline
1111 \\
\hline
011
\]

Gabriel Hugh Elkaim
Shifts and Rotates (1.2)

**Logical Right**
- Move bits to right
- Throw away bits that fall off of LSB
- Insert 0's at MSB

**Logical Left**
- Move bits to left
- Throw away MSB
- Introduce 2's complement at LSB

Example:
- $1010 \rightarrow 0101$
  - (10) \rightarrow (5)
- $0101 \leftarrow 1010$
  - $x2$
**Shifts and Rotates (2.2)**

**Rotate Right**
- Move bits to right
- Bits that fall off MSB are re-inserted at MSB

**Rotate Left**
- Move bits to left
- Bits that fall off LSB are re-inserted at LSB

1010 → 0101 → 1010

1101 ← 1101

Not

Gabriel Hugh Elkaim
Arithmetic Shifts (1.2)

**Arithmetic Right Shift (SRN)**
- Move bits to right
- Throw away bits that fall off LSB
- Sign extend MSB bits

**Arithmetic Shift Left**
- Move bits to left
- Introduce 0 at LSB
- Throw away bit that pops off MSB

Example:
- (−) 1100 → ¬ 110 (−)
- 00110101 → 00011010

Examples:
- 4 1 2 0 ↔ 2 1 0 4
- 2 1 0 4 ↔ 0 1 2 4
Arithmetic Shifts (2.2)

1 $\leftarrow$ 101 (5)
1010 (10)
10100 (20)
101000 (40)
1010000 (80)

1010000 (80) $\rightarrow$ 3

$\div 2^3 \text{ or } 8$

1010 (10)

$\leftarrow$ 1011 (-5)
11110110 (6)
$\gamma$ (-6)

SRL

0101 (5) $\rightarrow$ 1
10 (2)

SRX

10111 (-5)
10101 (-3)
Rotations (1.2)
Rotations (2.2)
The ALU (1.5)

Now we have all operations needed and we can design hardware to do it.
The ALU (2.5)

**Arithmetic Logic Unit** (or **ALU**, for short)
The ALU (3.5)
The ALU (4.5)
The ALU (5.5)
Abstractions and Computer Systems

Gabriel Hugh Elkaim
Computing Machines are Everywhere!

- General Purpose
- Special Purpose (Embedded)
- Embedded
Computing Machines: Distinguishing Features

- Cost: 1st million $ \#2 \to \infty, < \$$ 1
- Speed: clock speed (20 MHz)
- Cost: 10k
- Price/Performance
- Ease of Use (software + support)
- Interchange
- Power
- Size (4 bits - 64 bits)
Computing Systems
The Turing Machine (1.4)

While Ada Lovelace was the original programmer, the first person to prove the full power of software was Alan Turing (1912-1954).

Gabriel Hugh Elkaim
The Turing Machine (2.4)

Universal Computational Device:
- Given enough time and memory, all computers are capable of calculating the exact same thing.

Turing's Thesis:
Every computation can be performed by some Turing machine, a universal Turing machine can simulate any other.
The Turing Machine (3.4)
The Turing Machine (4.4)
Problem Transformation and Abstraction

- Ultimate objective: transform a problem from human speech to electrons in a circuit.
- This is computer science & computer engineering.
Computer Architecture (1.2)

Problems

Accounting

Higher-Level Languages

Assembly Language

Microcode
Computer Architecture (2.2)

- Problems
- Algorithms
- Language
  - Instruction Set Architecture
  - Microarchitecture
- Circuits
- Devices
Computer Science

**Definition** - The study of algorithms and data structures to solve problems.

**Abstraction** - Software design that allows the programmer to focus on critical aspects of a problem.

\[ \text{pow}(\text{num}, 2) \]
Procedure or Function

```c
int average(int a, int b)
{
    int avg;
    avg = (a+b)/2;
    return (avg);
}

main()
{
    x = 4;
    y = 2;
    k = average(x, y);
}
```
Programming Flow

Source Code
  ↓
Compiler or Interpreter
  ↓
Object Code

High-Level Language
  ↓
Compiler

Assembly
  ↓
Assembler

Machine Language
Computer Engineering

Definition: Cross-disciplinary application of engineering principles and methods to the design & development of hardware and software systems.
Instruction Set Architecture (ISA)

Interface between a computer's hardware and its software. Defines exactly what the computer's instructions do, how they are specified.
Central Processing Unit (CPU)

1980

2000
Motherboard: System
CPU: Package
SoC—System on a Chip

**800 Processor**

- Krait 400 CPU
  - Features 28nm process technology
  - 2GHz+ performance
- Adreno 330 for advanced graphics
- Hexagon QDSP6
  - For ultra-low power applications and custom programmability
- Integrated LTE, 802.11ac, USB 3.0
  - And BT 4.0 offers broad array of high-speed connectivity

**Multimedia**
- Audio, video, and captures

**Connectivity**
- 4G LTE, WiFi, USB, BT, and FM

**Camera**
- 55MP with dual ISP

**Navigation**
- Support for up to 2560x2048 display
- Miracast 1080p
- HD support
- EZat GNSS with support for three GPS constellations

Ultra HD Capture and Playback
- DTS-HD and Dolby Digital Plus audio
- Expanded gestures
CPU: Microarchitecture (1.3)
CPU: Microarchitecture (2.3)
CPU: Microarchitecture (3.3)
Advanced Microarchitecture (1.3)
Advanced Microarchitecture (2.3)
Advanced Microarchitecture (3.3)
Abstraction and Framing
Hardware vs. Software
Computer Organization
General Purpose Computers
Simplified Computer Model