Multiplication by an Integer Constant: Lower Bounds on the Code Length

Vincent LEFÈVRE

Loria, INRIA Lorraine

RNC’5

September 3 – 5, 2003
Introduction

Problem: to generate (optimal) code with elementary operations (left shifts, i.e. multiplications by powers of 2, additions and subtractions).

Example: compute $1997x$ (constant $n = 1997$).

1. $17x \leftarrow (x \ll 4) + x$
2. $51x \leftarrow (17x \ll 2) - 17x$
3. $1997x \leftarrow (x \ll 11) - 51x$

Can we get a very short code that computes $nx$?

Same question as with compression methods! (i.e. compress $n$.)

Other similarities: my heuristic, based on common patterns in the base-2 representation of $n$. 
Formulation of the Problem

Given: odd positive integer $n$ (our constant). We consider a sequence of positive integers $u_0, u_1, u_2, \ldots, u_q$ such that:

- initial value: $u_0 = 1$;
- for all $i > 0$, $u_i = |s_i u_j + 2^{c_i} u_k|$, with
  \[ j < i, \quad k < i, \quad s_i \in \{-1, 0, 1\}, \quad c_i \geq 0; \]
- final value: $u_q = n$.

Same operations with $u_0 = x$: we get code (called program in the following) that computes the $u_i x$, and in particular, $nx$.

Minimal $q$ associated with $n$ (denoted $q_n$)?
Outline:

1. Introduction / formulation of the problem (done).
2. Bounds on the shift counts.
3. A prefix code for the nonnegative integers.
4. How programs are encoded.
5. Lower bounds on the program length.
Bounds on the Shift Counts

Two data contribute to the size $\sigma$ of a program:

- the number $q$ of elementary operations (i.e. the length);
- the size of the parameters, in particular the shift counts $c_i$.

Information theory will give us information on $\sigma$. To deduce lower bounds on $q$, we need bounds on $c_i$.

Notation: for any positive integer $m$, let $\mathcal{P}_m$ be a subset of programs multiplying by $m$-bit constants; $S$ denotes a function such that for any program $\in \mathcal{P}_m$ and any $i$, $c_i \leq S(m)$.

$\mathcal{P}_m$: optimal programs, programs generated by some algorithm, etc.
$S(m)$: bound on the shift counts for any considered program (i.e. in $\mathcal{P}_m$) associated with $m$-bit constants.

For $n = 2^m - 1$, the optimal program will always be in $\mathcal{P}_m$. Therefore, $S(m) \geq m$.

For the set of programs generated by algorithms used in practice, $c_i \leq m$, therefore $S(m) = m$.

Proved upper bound for optimal programs:

$S(m) \leq 2^{\lceil m/2 \rceil} - 2(m + 1)$, but useless here.

For adequately chosen optimal programs, it seems that $c_i \leq m$. If this is true, then $S(m) = m$. → Lower bound on the length of any program.
But for the set of all optimal programs, consider the following example for $m = 6h + 1$: $n = (1 + 2^h)(1 + 2^{2h})(1 + 2^{4h}) - 2^{7h}$.

One of the optimal programs (4 operations):

\begin{align*}
  u_0 &= 1 \\
  u_1 &= u_0 << h + u_0 \\
  u_2 &= u_1 << 2h + u_1 \\
  u_3 &= u_2 << 4h + u_2 \\
  u_4 &= u_3 - u_0 << 7h.
\end{align*}

This gives: $S(m) \geq 7h = \frac{7}{6} (m - 1)$.

→ The choice of the optimal program for a constant $n$ is important.

We will also consider $S(m) = k.m$, with $k > 1$. 
A Prefix Code for the Nonnegative Integers

Linked to the *unbounded search problem*: there exists a code in \( \log_{\max}(n) + O(\log^*(n)) \).

Here, we are only interested in a code in \( \log_2(n) + o(\log_2(n)) \).

For \( n \geq 4 \):

- \( k \): number of bits of \( n \) minus 1;
- \( h \): number of bits of \( k \) minus 1;
- code word of \( n \): 3 concatenated subwords  
  - \( h \) digits 1 and a 0
  - \( h \) bits of \( k \) without the first 1
  - \( k \) bits of \( n \) without the first 1.
<table>
<thead>
<tr>
<th>integer</th>
<th>code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>10 0 00</td>
</tr>
<tr>
<td>5</td>
<td>10 0 01</td>
</tr>
<tr>
<td>6</td>
<td>10 0 10</td>
</tr>
<tr>
<td>7</td>
<td>10 0 11</td>
</tr>
<tr>
<td>8</td>
<td>10 1 000</td>
</tr>
<tr>
<td>15</td>
<td>10 1 111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>integer</th>
<th>code word</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>110 00 0000</td>
</tr>
<tr>
<td>31</td>
<td>110 00 1111</td>
</tr>
<tr>
<td>32</td>
<td>110 01 0000</td>
</tr>
<tr>
<td>63</td>
<td>110 01 1111</td>
</tr>
<tr>
<td>64</td>
<td>110 10 0000</td>
</tr>
<tr>
<td>127</td>
<td>110 10 1111</td>
</tr>
<tr>
<td>128</td>
<td>110 11 0000</td>
</tr>
<tr>
<td>255</td>
<td>110 11 1111</td>
</tr>
<tr>
<td>256</td>
<td>1110 000 0000</td>
</tr>
<tr>
<td>511</td>
<td>1110 000 1111</td>
</tr>
</tbody>
</table>
Encoding an Elementary Operation

Elementary operation: \( u_i = |s_i u_j + 2^{c_i} u_k| \).

→ Encode \( s_i, c_i, j \) and \( k \).

- \( s_i \): 3 possible values \((-1, 0 \text{ and } 1) \to 2\) bits.
  4th one for the end of the program.
- Integers \( c_i, j \) and \( k \): prefix code.
- Concatenate the 4 code words.
Bounds on the integers:

- $c_i$ bounded above by $S(m) = k.m$.
- $j$ and $k$ bounded by $i - 1$, and without significant loss, by $q - 1$.

→ Upper bound on the size of the encoded program:

$$B(m, q) = q (2 + C(S(m)) + 2 C(q - 1)) + 2.$$  

with $C(n) = \begin{cases} 
3 & \text{if } n \leq 3, \\
\lfloor \log_2(n) \rfloor + 2 \lfloor \log_2(\log_2(n)) \rfloor + 1 & \text{if } n \geq 4. 
\end{cases}$

Asymptotically: $B(m, q) \sim q (\log_2(S(m)) + 2 \log_2(q))$.

With $S(m) = k.m$: $B(m, q) \sim q (\log_2(m) + 2 \log_2(q))$. 


Let $f$ and $g$ be two positive functions on some domain.

$f(x) \gtrsim g(x)$ if there exists a function $\varepsilon$ such that

$$|\varepsilon(x)| = o(1) \quad \text{and} \quad f(x) \geq g(x) \left(1 + \varepsilon(x)\right).$$

Note: it is equivalent to say that there exists a function $\varepsilon'$ such that

$$|\varepsilon'(x)| = o(1) \quad \text{and} \quad f(x) (1 + \varepsilon'(x)) \geq g(x).$$
Lower Bounds: Worst Case

We consider the $2^{m-2}$ positive odd integers having exactly $m$ bits in their binary representation, and for each integer, an associated program in $\mathcal{P}_m$. The $2^{m-2}$ programs must be different.

⇒ There exists a program whose size $\sigma$ is $\geq m - 2$, and its length $q$ satisfies: $m - 2 \leq \sigma \leq B(m, q) \leq B(m, q_{\text{worst}})$.

We recall that asymptotically, with $S(m) = k.m$, we have:

$$B(m, q_{\text{worst}}) \sim q_{\text{worst}} (\log_2(m) + 2 \log_2(q_{\text{worst}})).$$

We can guess that $\log_2(q_{\text{worst}}) \sim \log_2(m)$. Thus we choose to bound $q_{\text{worst}}$ by $m$ and write: $q_{\text{worst}} (3 \log_2(m)) \gtrsim B(m, q_{\text{worst}})$. 
We recall that \( q_{\text{worst}} (3 \log_2(m)) \gtrsim B(m, q_{\text{worst}}) \geq m - 2 \).

As a consequence: \( q_{\text{worst}} \gtrsim \frac{m}{3 \log_2(m)} \).

Note: this also proves that \( \log_2(q_{\text{worst}}) \sim \log_2(m) \), thus we didn’t lose anything significant when bounding \( q_{\text{worst}} \) by \( m \).

Exact lower bound for \( m \geq 4 \):

\[
\frac{m - 4}{3 \log_2(m) + 4 \lceil \log_2(\log_2(m)) \rceil + 2 \lceil \log_2(\log_2(k.m)) \rceil + \log_2(k) + 6}
\]

(note: very optimistic for small \( m \) — e.g., \( < 1 \) for all \( m \leq 37 \)).
Lower Bounds: Average Case

We consider the set $O_m$ of the $2^{m-2}$ positive odd integers having exactly $m$ bits in their binary representation, and for each integer, an associated program in $P_m$.

The $2^{m-2}$ programs must be different:

$$
\frac{1}{2^{m-2}} \sum_{i \in O_m} B(m, q_i) \geq \frac{1}{2^{m-2}} \sum_{i=1}^{2^{m-2}} \lceil \log_2 i \rceil = m - 4 + \frac{m}{2^{m-2}},
$$

As a consequence,

$$
2 + (2 + C(S(m)) + 2 C(m)) \frac{1}{2^{m-2}} \sum_{i \in O_m} q_i \geq m - 4 + \frac{m}{2^{m-2}}.
$$
We recall that
\[
2 + (2 + C(S(m)) + 2 C(m)) \frac{1}{2^{m-2}} \sum_{i \in O_m} q_i \geq m - 4 + \frac{m}{2^{m-2}}.
\]

Thus \( q_{av} \geq \frac{m - 6 + m/2^{m-2}}{2 + C(S(m)) + 2 C(m)} \).

Asymptotically, with \( S(m) = k.m \), the average length \( q_{av} \) satisfies:
\[
q_{av} \geq \frac{m}{3 \log_2(m)},
\]
i.e. the same bound as in the worst case.
For random $m$-bit constants: approximated upper bounds on $q_{av}$ (obtained with my algorithm), lower bounds on $q_{av}$ and the ratio.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$q_{av}^+$</th>
<th>$q_{av}^-$</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2.6</td>
<td>0.11</td>
<td>24.5</td>
</tr>
<tr>
<td>16</td>
<td>4.4</td>
<td>0.34</td>
<td>12.8</td>
</tr>
<tr>
<td>32</td>
<td>7.6</td>
<td>0.81</td>
<td>9.35</td>
</tr>
<tr>
<td>64</td>
<td>13.4</td>
<td>1.66</td>
<td>8.09</td>
</tr>
<tr>
<td>128</td>
<td>23.7</td>
<td>3.21</td>
<td>7.38</td>
</tr>
<tr>
<td>256</td>
<td>42.2</td>
<td>5.32</td>
<td>7.93</td>
</tr>
<tr>
<td>512</td>
<td>75.5</td>
<td>10.1</td>
<td>7.46</td>
</tr>
<tr>
<td>1024</td>
<td>135</td>
<td>19.2</td>
<td>7.05</td>
</tr>
<tr>
<td>2048</td>
<td>243</td>
<td>36.5</td>
<td>6.67</td>
</tr>
<tr>
<td>4096</td>
<td>440</td>
<td>69.3</td>
<td>6.35</td>
</tr>
<tr>
<td>8192</td>
<td>803</td>
<td>132</td>
<td>6.08</td>
</tr>
</tbody>
</table>