Department of Electrical and Computer Engineering
University of Texas at Austin

EE 306
Fall, 2003
Instructor: Tony Ambler
TAs: Sundar Balasubramaniam, Ming Ding, Ashwin Gunasekar, Aamir Hasan, Harshal Ingole, Ritu Kar, Paroma Sen, Sujata Sharma, Raghuveer Simha

Programming Assignment 2
Due: Midnight – 31st October, 2003

Instructions
Name your file program2.asm
Write your name at the beginning of the program file you submit. You will be penalized if you don’t. Be sure to write it as a comment, like so
; Last name, First name
No late submissions will be accepted. So make sure you start early.
Programs you submit must be your own work. You may discuss algorithms with your classmates. You may not discuss your program with others. Do not show your code to any of your fellow students. Plagiarism of any kind will not be tolerated.

All My Strings are Packed

The ASCII character set is an 8-bit code for representing characters (letters, digits, punctuation marks, spaces). A sequence of characters is called a string. The null character (all zeroes) is usually used to mark the end of a string.

There are two ways of storing strings in the physical memory of the LC-3 (which consists of 16-bit locations). We could store a single character in each memory location, in which case the higher 8-bits would be 0s. But this is inefficient; we waste the higher 8-bits. A better way of storing strings is to pack 2 characters per memory location. This technique is called (rather unimaginatively) packing. This assignment deals with such a packed string.

Description and Specifications
You will be given a set of 16 bit numbers, stored location x3151 onwards. Each of these numbers is actually 2 ASCII characters packed into one 16 bit location. The number (N) of memory locations to unpack is stored at x3150.

Your goal is to split each of the 16 bit numbers into two 8 bit characters and store these (a character per location) starting location x4000. You would need
2N locations to store the string. Print the string to the console. You should use TRAP x22 to print the string, with Ro set to x4000. TRAP x22 requires that the string be terminated by the null character, so make sure you insert the null.

Example

Given the following packed string

<table>
<thead>
<tr>
<th>Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x3150</td>
<td>x002</td>
</tr>
<tr>
<td>x3151</td>
<td>x4F3E</td>
</tr>
<tr>
<td>x3152</td>
<td>x2A69</td>
</tr>
</tbody>
</table>

The unpacked string will be

x4000 x003E
x4001 x004F
x4002 x0069
x4003 x002A
x4004 x0000

Output produced by the program is >Oi*

This is the ASCII representation of the numbers in the locations 4000, 4001, 4002 and 4003. The null character in the end (4004) specifies the end of the string. The ASCII chart can be obtained from Appendix E of your textbook.

Hints

To split the 16 bit number into 8 bit characters, you need to use a technique called Masking. Masking involves ANDing a given number with a particular value (mask) to extract certain bits. For example, when we AND 1101 with 0011 we get 0001. The two higher bits are reset to 0 (masked out) while the lower bits remain unchanged. Extending this principle to 8 bit numbers in hex notation, when we AND xD9 with the mask xoF, we get x09. In this case the higher 4 bits are reset to 0 while the lower 4 bits are unchanged. You need to figure out the mask needed to extract 8 bits from the 16 bit numbers.

You will also need to shift bits in this assignment. To shift a number 1 bit to the left, you multiply it by 2. Conversely, to shift it to right by a bit, you divide the number by 2. This can be extended to shift an arbitrary number of bit positions.

Shifting 1 bit position to the left

\[
\begin{array}{c}
01110110 \\
\times \quad 10 \quad \text{(Binary Equivalent of 2)} \\
\hline
11101100
\end{array}
\]

Shifting 1 bit position to the right

\[
\begin{array}{c}
01001000 \\
/ \quad 10 \quad \text{(Binary Equivalent of 2)} \\
\hline
00100100
\end{array}
\]