The objective of this lab is to help you learn the skill of programming in 68HC12 assembly language – quickly. When you work on the term project later in the quarter, you will need to have learned this skill so that you can concentrate on the details of the project rather than trying to figure out how to write a basic program.

The exercise will take two weeks and be done by a team of two people. During the first week you will write the pieces of the program, and during the second week you will put everything together into a working product. The report and working code will be due at the beginning of class after the second week (that is, on October 11).

There are numerous resources to help you write your program. Some of the most important are the library section of the course webpage, the course text, and the Y:\documents folder on the computers in the lab. Begin with the ME405 Library file from the web page.

The Assignment

Write a program which implements a hexadecimal calculator that can add or subtract two signed eight-bit operands in accordance with the specifications below. The calculator is to be used as follows:

- The user keys in the first operand
- The user presses the function key “+” or “-”
- The user then keys in the second operand
- The user presses the “equals” key, and the calculator performs the requested operation and displays the result
- The user can then press another function key and another number, and the calculator will continue to compute results
- There must be some provision for clearing the calculator to allow a new calculation to be made; using the reset key is acceptable

Other specifications which the calculator program must meet are as follows:

1. Operands are typed into the hex keyboard. Each operand is one byte, and it is entered as one or two digits as needed. A single-digit number such as five could be entered as “5” or “05” and must work correctly either way.

2. There must be a user-friendly display. One possibility is to show a prompt, and echo the keys typed in, on the top line. The bottom line can then show output and error messages.
3. The calculator must clearly indicate error conditions, such as no digits typed in, too many digits, or a digit entered when a function key should have been pressed (there may be other error conditions, and the calculator should respond appropriately to those as well).

4. The arithmetic is **signed**, so if a result is negative, it must be displayed as negative. However, unsigned numbers are entered at the keyboard.

5. Any possible result must be displayed correctly, even if it is negative or is a 9-bit number requiring an extra digit.

6. The keys must be assigned as shown in the diagram below. Use of the backspace key to allow editing of the number typed in is optional (but it sure is nice).

7. Code must be thoroughly commented.

8. Your program should be thoroughly tested with as many different possible inputs as you can think of. If there are any bugs, you will score higher by finding them and describing them in your report than if I find them when testing your program!

---

**Deliverables**

**Week 1** By the beginning of lab after the first week (October 4), you are to turn in a working program which displays a prompting message on the LCD keyboard, reads in a number in unsigned hexadecimal form, and displays that same number as a *signed* hexadecimal number on the screen. The program will be turned in according to the instructions given below.

**Week 2** By the beginning of lab after the second week (October 11), you are to turn in a complete working calculator program as specified above, using the same submission procedure given below. In addition, you will turn in a brief memorandum-format report (1 page is ideal, 2 pages maximum) describing what you accomplished. If there are any flaws, bugs or “undocumented features” in your program, describe them here and say briefly how you would fix them if you had the time.

The way in which each program will be turned in is as follows: You will place your source code file (in *filename.s* text format) and your downloadable machine language file (in *filename.s19* format) in a folder which is named according to the last names of your group’s members: if your members were named S. Laurel and O.
Hardy, the folder would be called LaurelHardy. This folder, along with its contents, will be copied to a location which will be specified in lab.

Hints

- When you convert numbers from binary (8-bit number) to ASCII (two or three displayed digits) form, it is easiest to use look-up tables with indexed addressing modes. See example 2.6 in the text.

- In creating the tables, you can use a bunch of .byte commands in a row. The data will be stored in successive memory locations, thus forming your table:

```assembly
.byte 'M'  ; Table to convert weekday numbers
.byte 'T'  ; into weekday letter codes
.byte 'W'
.byte 'R'
.byte 'F'
...```

- For creating strings to display, the .asciz directive is recommended. It’s not in the text but is described in the ICC12 manuals.

- A general programming note: Try to get something simple working, then modify it bit by bit, testing each modified version thoroughly. If you try to create large portions of a program and then test the whole thing, it’s really hard to find where in your code problems are occurring.

- Sometimes printing diagnostic information to the LCD display is even easier than using the debugger.

- Watch out for errors with using immediate addressing when you meant to use direct/extended, and vice versa. Historically, about half of the programming problems in ME 405 have been caused by forgetting the # sign or putting it in when it oughtn’t be there.

### ASCII Character Table

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>\0</td>
<td>SO</td>
<td>ST</td>
<td>ET</td>
<td>EO</td>
<td>EQ</td>
<td>AK</td>
<td>BL</td>
<td>BS</td>
<td>HT</td>
<td>LF</td>
<td>VT</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
</tr>
<tr>
<td>10</td>
<td>DL</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>NK</td>
<td>SY</td>
<td>ET</td>
<td>CN</td>
<td>EM</td>
<td>SB</td>
<td>ES</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
</tr>
<tr>
<td>20</td>
<td>spec</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>,</td>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>30</td>
<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>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
</tr>
<tr>
<td>40</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>50</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>^</td>
</tr>
<tr>
<td>60</td>
<td>'</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>70</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td></td>
<td>}</td>
<td>^</td>
</tr>
</tbody>
</table>