Physics 116B Winter 2007: Optional Easy68K Problem

1. Consider the EASY68k program on the next page. The lines are numbered in the left column. This time, we are just summing any ASCII characters that happen to be decimal integers. The data are read from the keyboard and output to the screen using EASY68k I/O routines which rely on the trap #15 instruction. The use of these routines is explained in the included table from the EASY68k help commands. The trap instruction itself is discussed on p. 6-13 of the M68000 Microprocessor User’s Manual. It is useful for implementing system program calls as is done here. The result is equivalent to a MAS system subroutine call although the mechanism is different.

Notice the use of absolute addressing for the code and data via the org directive for the assembler. That is, the first instruction with the label start will be placed at location $1000. The storage area for msg will begin at $2000.

(a) On line 8, the instruction lea msg, a1 is assembled as 43F9 00002000 (numbers are hex).
   i. Look up LEA in the M68000 Programmer’s Reference Manual. Compare the binary pattern of the first word of this instruction and verify it is what you expect (explain). Check the register field and effective address field (mode, register) to determine the addressing mode for this operation.
   ii. Look up the behavior of this addressing mode in Sec. 2.2 of the manual. Verify that the last 4 bytes are what you expect (explain).
   iii. Would you expect this addressing mode to be used in the Mac program with the MAS system? Explain briefly.

(b) The instruction jsr addit (line 18) is located at address $1022 and the next instruction (line 19) at $1028. Prior to executing line 18, the stack pointer contains $1000000. Upon entry to the addit subroutine on line 32, answer the following.
   i. What is the contents of the stack pointer?
   ii. What is the 32 bit word contained in the memory address referenced by the stack pointer?

(c) If the ASCII character for zero (≠ 0) is in d0 on entry to the addit subroutine on line 32, what is the contents (i) of the Z condition code and (ii) the N condition code after line 33 is executed?

Note: You are strongly encouraged to get EASY68k to run or modify this program. A text file of the code is on the Physics 116 site. You can use the EASY68k simulator to watch the registers, memory and stack as you run the program. Breakpoints can be set to stop the execution if you want to look at registers. Also, there are two modes of single-stepping. One steps over user subroutines (runs them but does not trace execution) while the other traces execution into the subroutine.
1 ; Program to read, store and echo string of 20 ASCII characters and sum up any numbers
2 ; Modified for Easy68K simulator - uses Easy68K trap instructions for I/O
3 ; Sum in D2 at end, display on screen
4
5 start: org $1000 ; set up registers for trap instruction to read string
6 move.w #2,d0 ; number of characters to read in d1 (80 max)
7 lea msg,a1 ; load the address of msg in a1 (must use LEA)
8 trap #15 ; do EASY68k I/O operation
9 move.w #0,d0 ; display string on screen
10 trap #15 ; do EASY68k I/O operation
11
12 ; begin processing
13
14 clr d2 ; clear the register for the sum
15 jmp enter ; enter loop at end
16 loop: move.b (a1)+,d0 ; move the character from msg array to d0
17 jsr addit ; subroutine to test and add integers
18 enter: dbra d1,loop ; subtract 1 from d1 and see if done
19
20 ; all data processed, sum in d2 - now display number on screen
21
22 move.l d2,d1 ; put result in d1 for EASY68k I/O
23 move.w #3,d0 ; display number on screen
24 trap #15 ; do EASY68k I/O operation
25
26 STOP #2000 ; end of program
27
28 ; Subroutine addit tests ASCII characters to see if they represent numbers
29 ; and if so, adds them to the sum in d2
30
31 addit: and.b #$7F,d0 ; mask off parity bit of character
32 cmp.b #$30,d0 ; see if it is less than $30
33 blt skip ; if so, skip to return statement
34 cmp.b #$39,d0 ; see if it is greater than $39
35 bgt skip ; if so, skip to return statement
36 and.w #$000F,d0 ; get the number
37 add.w d0,d2 ; add to sum in d2
38 skip: rts ; return from subroutine
39
40 org $2000 ; data storage area
41 msg: ds.b 80 ; storage for string of ASCII characters
42 nchar: dc.w 20 ; number of characters to read
43
44 END START

Sim68K Input/Output

TRAP #16 is used for I/O.
DO contains the task number:

<table>
<thead>
<tr>
<th>Text</th>
<th>10</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Debug entry at A11</td>
<td>D0 is pointer to breakpoint without ULP.</td>
</tr>
<tr>
<td>1</td>
<td>Debug entry at A11</td>
<td>D0 is pointer to breakpoint (must use ULP).</td>
</tr>
<tr>
<td>2</td>
<td>Read keyboard string starting at (A1) length returns in D1 (W max 80)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Display number in D1 in decimal window (A0)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Read a number from keyboard and store in D1 (W)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Read single character from keyboard and store in D1 (B)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Get D1 to D1+1 keyboard input pending, otherwise set to 0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>The code to read pending key</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Return back if last byte of a second since last byte = D1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Terminate the program. (Not Implemented)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Print screen area (window) assigned to D1 and D2 by DO</td>
<td></td>
</tr>
</tbody>
</table>

*Read these, see bit 4 of D1.*

<table>
<thead>
<tr>
<th>Text</th>
<th>11</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Print byte at D1, COL.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>The high byte of D1 is stored for D1, number (0-79).</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The low byte is stored for the current position (0-79).</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Out of range characters are ignored.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>The screen. See bit 4 of D1.</td>
<td></td>
</tr>
</tbody>
</table>

*These sequences, see bit 5 of D1.*

<table>
<thead>
<tr>
<th>Text</th>
<th>12</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Keyboard 0-9.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>See D1 to D1+1 keyboard input.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>See D1 to enable key (default)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Enter terminated string at (A1) from D0 with BCRB.</td>
<td></td>
</tr>
</tbody>
</table>

*Key codes 0 - 12 are compatible with the terminate method.*