Physics 116B Winter 2004: Problem on Programming

3/9/2004

1. Consider the M68000 Mac program on the next page (from the M68000 programming notes).

   (a) Which lines contain assembler directives rather than M68000 instructions?

   (b) In line 2, why not use the address register A5 to hold the pointer to msg2 instead of A1?

   (c) In line 5, do you think the assembler uses absolute addressing for the jmp enter command? If not, why not and what kind of addressing do you think is used? Hint: this is similar to the case with the jsr commands described in the programming notes.

   (d) Which lines use immediate addressing?

   (e) What would be the problem with replacing the jsr with jmp on line 8?

   (f) Just before the subroutine addit is called on line 8, the stack pointer contains $69C778. What is the contents of the stack pointer in the subroutine?

   (g) Consider line 19 containing cmp b. Refer to the instruction description and see if you think the assembler has actually inserted a (slightly) different instruction here. If so which?

   (h) Could the instruction on line 19 have the arguments reversed (i.e., register first, number second) if the next line were changed to bge?

   (i) If the ASCII character was $37, what is contained in register D0 when the subroutine is exited on line 25?

   (j) What is the decimal representation of $37?
; Program to read and store 20 ASCII characters and sum up any integers

1 xref getchar, strout, decout, newline, stop
2 start: lea msg,a1 ; put address of msg in a1
3 move.w nchar,d1 ; number of bytes to read
4 clr d2 ; clear the register for the sum
5 jmp enter ; enter loop at end
6 loop: jsr getchar ; getchar puts the character in d0
7 move.b d0,(a1)+ ; move the character to memory
8 jsr addit ; subroutine to test and add integers
9 enter: dbra d1,loop ; subtract 1 from d1 and see if done

; now output the information
10 lea msg,a0 ; set up for outputting character string
11 move.w nchar,d0
12 jsr strout ; output the string
13 jsr newline
14 move.w d2,d0 ; output the sum
15 jsr decout
16 jsr newline
17 jsr stop ; end of program

; Subroutine addit tests ASCII characters to see if they represent numbers
; and if so, adds them to the sum in d2
18 addit: and.b #$7F,d0 ; mask off parity bit of character
19 cmp.b #$30,d0 ; see if it is less than $30
20 blt skip ; if so, skip to return statement
21 cmp.b #$39,d0 ; see if it is greater than $39
22 bgt skip ; if so, skip to return statement
23 and.w #$000F,d0 ; get the number
24 add.w d0,d2 ; add to sum in d2
25 skip: rts ; return from subroutine
26
27 msg: ds.b 100 ; set aside 100 bytes of storage
28 nchar: dc.w 20 ; number of characters to read
29 end
### Table C-2. ASCII Code

<table>
<thead>
<tr>
<th>Least Significant Digit (Ref)</th>
<th>Most Significant Digit (Hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 NUL DLE SP 0 @ P ' p</td>
<td>1 SOH DC1 ! 1 A Q a q</td>
</tr>
<tr>
<td>2 STX DC2 * 2 B R b r</td>
<td>3 ETX DC3 # 3 C S c s</td>
</tr>
<tr>
<td>4 EOT DC4 $ 4 D T d t</td>
<td>5 ENQ NAK % 5 E U e u</td>
</tr>
<tr>
<td>6 ACK SYN &amp; 6 F V f v</td>
<td>7 BEL ETB ' 7 G W g w</td>
</tr>
<tr>
<td>8 BS CAN ( 8 H X h x</td>
<td>9 HT EM ) 9 I Y i y</td>
</tr>
<tr>
<td>A LF SUB * : J Z j z</td>
<td>B VT ESC + ; K [ k (</td>
</tr>
<tr>
<td>C FF FS . &lt; L \ I l l</td>
<td>D CR GS - = M ] m )</td>
</tr>
<tr>
<td>E SO RS . &gt; N ^ n ~</td>
<td>F SI US / ? O _ o DEL</td>
</tr>
</tbody>
</table>

Example:

'A' = $41