C SC 230 F01 2001
Midterm Test
October 18, 2001

Solutions

1. (20 marks) Perform each of the following operations using 8 bit 2's complement numbers and show the condition code flag settings that will result . As shown all operations are to be done as additions.

Decimal	**Answer** (show all values in hex)	**C**	**V**	**N**	**Z**
-7 + 5	____F9_____ + ____05_____ = __FE___	0	0	1	0
14 - 14	____0E_____ + ____F2_____ = __00___	1	0	0	1
-128 - 3	____80_____ + ____FD_____ = __7D___	1	1	0	0

2. (3 marks) For the IEEE single precision floating point representation of the decimal value -12.125, complete the following:

   the value of the sign bit (binary) ___1___
   the value stored for the exponent (decimal) __130__
   the stored mantissa (do not show the hidden bit or trailing 0's) (binary)

   _____100001_____

3. (4 marks) What is the purpose of the PC register on the 6811?

   __ It points to next byte of program code to be executed._____

   State three distinct situations in which the value in the PC is modified:

   Any three of:
   __A branch instruction is executed._____
   __A jump instruction is executed._____
   __It is updated as bytes of code are fetched._____
   __A JSR instruction is executed._____
   __An RTS instruction is executed_____

4. (1 marks) Explain the purpose of the JMP $E000 instruction used in programs run on the lab board.

____It causes the processor to continue execution at the _____

____start of the Buffalo monitor._____

5. (15 marks) Write a complete 6811 assembly language program that sums the odd integers from 1 to 149 inclusive leaving the sum in IX upon completion of the program. Include brief comments to explain your approach.

```
            ORG   $C000
            LDX   #0    / INITIALIZATION
            LDAB  #1
  LOOP      ABX         / ADD B TO IX
            INCB
            INCB
            CMPB  #149  / CHECK IF DONE
            BLS   LOOP  / IF NOT REPEAT
            STOP
            END
```

6. (7 marks) Fill-in the blanks (indicated by ____) in the program shown below so that it behaves as described in the comments.

```
SBASE       EQU   $CFFF
            ORG   $D000
STR1        FCC   "Sample string"
            FCB   $00
STR2        RMB   20
            ORG   $C000
START       LDS   #SBASE
            LDX   #STR1       / PUT PARAMETERS ON STACK
            PSHX
            LDX   #STR2
            PSHX
            JSR   CPYSTR      / EXECUTE CPYSTR
            INS               / CLEAR PARMETERS FROM STACK
            INS
            INS
            INS
            STOP

; CPYSTR copies a string (source) from the location specified
; by the first parameter on the stack to the location
; (destination) specified by the second parameter (closest to
; the top of the stack).  The string terminator is $00.

CPYSTR      PSHX        / SAVE REGISTERS
            PSHY
            PSHA

            TSX
```

```
            LDY   _7_,X / SET Y TO POINT TO DESTINATION STRING
            LDX   _9_,X / SET X TO POINT TO SOURCE STRING

CPY1        LDAA  0,X   / COPY STRING ONE CHARACTER AT A TIME
            STAA  0,Y   / INCLUDING THE TERMINATOR
            TSTA

            _BEQ_ CPY2  / EXIT LOOP AT TERMINATOR

            _INX_

            _INY_
            BRA   CPY1

CPY2        _PULA_            / RESTORE REGISTERS

            _PULY_

            _PULX_
            RTS

            END
```

7. (15 marks) Consider the routine 'outd' shown below as used in the labs.

```
; Subroutine outd displays the value in ACCD
; on the PC screen as an unsigned integer.
(1)   outd    psha            / protect registers
(2)           pshb
(3)           pshx
(4)           pshy
(5)           ldx     #$FFFF  / mark top of stack
(6)           pshx
(7)   outd1   cpd     #0      / loop while ACCD != 0
(8)           beq     outd2
(9)           ldx     #10     / find rightmost digit
(10)          idiv            / D/X: quotient->X remainder->D
(11)          ldaa    #'0'    / convert to ASCII character
(12)          aba
(13)          psha            / stack it
(14)          xgdx
(15)          bra     outd1   / repeat
(16)  outd2   pula            / pull top byte from stack
(17)          cmpa    #$FF    / if marker we are finished
(18)          beq     outd3
(19)          jsr     OUTA    / output a digit
(20)          bra     outd2   / repeat
(21)  outd3   jsr     OUTCRLF / end of line
(22)          pula            / pull 2nd byte of marker
(23)          puly            / restore registers
(24)          pulx
(25)          pulb
(26)          pula
(27)          rts
```

(a) The version given does not work if given ACCD=0. What would you change so that it will? Refer to the numbers in brackets above to identify lines to be removed or replaced, as well as any new lines (for new lines, number them 1.1, 1.2 etc.)

___REMOVE LINES (7) AND (8)_____

___REPLACE (9) WITH outd1 ldx #10_____

___REPLACE (15) WITH cpd #0 _____

___ADD (15.1) bne outd1 _____

(b) What is the minimal change so that the value will be displayed in binary rather than decimal?

___CHANGE THE IMMEDIATE VALUE 10 IN LINE (9) TO 2 __

(c) What would you add so that the routine will also work for negative values (number new lines as indicated above)?

use this column first	*continue here if necessary*
__(6.1) cpd #0 _____	**___(6.7) pulb _____**
__(6.2) bge outd1 ___	**___(6.8) pula _____**
__(6.3) psha _____	**___(6.9) coma _____**
__(6.4) pshb _____	**___(6.10) comb _____**
__(6.5) ldaa #'-' _____	**___(6.11) addd #1_____**
__(6.6) jsr outa ____	**_____**

Recall: OUTA is a Buffalo monitor routine that displays the ASCII character passed to it in ACCA. OUTCRLF is a Buffalo monitor routine that displays a carriage return and line feed. Neither routine protects the CPU registers.

You may not need all the spaces provided.