Experiment No. 4
CODE CONVERSION and BIT MANIPULATION
ECE 441

Peter Chinetti

October 10, 2013

Date Performed:	September 26, 2013
Partners:	Zelin Wu
Instructor:	Professor Saniie

1 Introduction

1.1 Purpose

The purpose of this experiment is to accomplish the following:

- perform ASCII, BCD and Hexadecimal Code Conversion
- gain familiarity with the 68000's bit manipulation instructions
- learn how to download programs from a host computer into the SANPER-1 ELU.

1.2 Background

1.2.1 Bit Manipulation

Bit manipulation is the ability to modify each bit according to some algorithm. The 68000 has the following four Bit Manipulation Instructions:

- BCHG Test a Bit and Change
- BCLR Test a Bit and Clear
- BSET Test a Bit and Set
- BTST Test a Bit

Bit manipulation can also be performed with Logical Instructions such as:

- AND Logical AND
- ANDI Logical AND Immediate
- OR, Logical Inclusive OR
- ORI Logical Inclusive OR Immediate
- EOR Logical Exclusive OR
- EORI Logical Exclusive OR Immediate
- NOT Logical Complement

Lastly, bit manipulation can be performed with Shift and Rotate Instructions such as:

- ASL Arithmetic Shift Left
- ASR Arithmetic Shift Right
- LSL Logical Shift Left
- LSR Logical Shift Right
- ROL Rotate Left
- ROR Rotate Right
- ROXL Rotate Left with Extend
- ROXR Rotate Right with Extend

1.2.2 Downloading Capability

Through a combination of hardware and software, the SANPER-1 ELU is capable of receiving MC68000 programs from an external computer, and storing these programs into the SANPER-1 ELUs memory. This downloading capability is achieved in hardware by connecting the serial port of the computer to one of the serial ports of the SANPER-1 ELU. The download functionality is achieved in software through the TUTOR firmware. Invoking TUTORs Transparent Mode Command ("TM") sets up the SANPER-1 hardware to wait for data to arrive through one of its serial ports. The external computer then transmits a file out of its serial port. The file is sent in Motorola S-Record format. The TUTOR firmware reads in the data from its serial ports and stores it into memory. The procedure to download a program from a personal computer to the SANPER-1 ELU is described in the SANPER-1 Educational Lab Unit Users Manual.

2 Lab Procedure and Equipment List

2.1 Equipment

- SANPER System
- Computer with TUTOR software

2.2 Procedure

Execute each program and record data when requested.

3 Results, Analysis and Discussion

3.1 Bit Manipulation Program

```
                ORG $1000
START:
* Initialize registers that will be used to 0
        CLR.L   D0
        CLR.L   D1              ;The sum of binariesx
        CLR.L   D2              ;Mutiplication place
        CLR.L   D3              ;The mutiplier
*Prompt input from the terminal
        LEA     $3000,A5
    LEA $3000,A6
        MOVE.B  #241,D7         ;Move function #241 to register D7
        TRAP    #14             ;Input String from the terminal
*ASCII to decimal converter
        MOVEA.L A5,A4           ;Copy the starting address to A4
LOOP:
    CMPA    A6,A4           ;Check if A4 is greater than A6
    BGE NEXT                ;If A>=A6, done converting and branches to NEXT
    SUB.B   #$30,(A4)+      ;Else subtract the content of A4 by #$30 then increment
    BRA LOOP
*Decimal to binary converter
NEXT:
    MOVEA.L A6,A4           ;Copy the ending address to A4
    MOVE.B  #1,D3           ;Store mutiplier in D1
MUTIPLICATION:
    SUBA    #1,A4           ;Let A4 point to te end of the String
    CMPA    A5,A4           ;Compare the address of A5 and A4
    BLT DONE                ;if A4<A5, done converting
    MOVE.B  (A4),D2         ;Else move the byte from A4 to D2
    MULU    D3,D2           ;Mutiply D2 by D3(1,10,100)
    ADD.W   D2,D1           ;Add the value from D2 to D1
```

```
    CLR.L     D2              ;Clear D2 to give empty space for next multiplication
    MULU      #$A,D3          ;Times the mutiplier by 10 for next multiplication
    BRA MUTIPLICATION         ;Branches back to the converting process
DONE:

    MOVE.B D1,$2000
***********************************************************NEW


*BIT MANIPULATION
*D0= OUTPUT
*D1= INPUT BINARY NUMBER
*D2= BIT A
*D3= BIT B

*O1 = I1 NAND I7
        MOVE.L  D1, D3          ;Copy the binary from D1 to D3 for temp manipula
        MOVE.L  D1, D2          ;Copy the binary from D1 to D2 for temp manipula
        ROR     #6,D2           ;MOVE I7 IN D2 to the location I1
        AND     D3,D2           ;AND D2 AND D3 then store the result in D2
        NOT     D2              ;Complement the binary numbers in D2
        ANDI    #$2,D2          ;Clear all bits except I1
        OR      D2,D0           ;Store the updated I1 from D2 to D0

*O0 = I0 XNOR O1
        MOVE.L  D0,D2           ;Copy the current output to D2
        ROR     #1,D2           ;Move bit I1 TO I0
        EOR     D3,D2           ;D2 = D3(I0) EOR D2(01)
        NOT     D2              ;D2 = compliment of D2
        ANDI    #$1,D2          ;Clear all bits except I0
        OR      D2,D0           ;Store the O1 from D2 to D0

*O2 = I0 EOR I5
        MOVE.L  D1,D2           ;Copy the binary number from in D2
        ROL     #2,D2           ;Move bit I0 to I2 of D3
        ROR     #3,D3           ;Move bit I5 tp I2 of D3
        EOR     D3,D2           ;D2 = D3 EOR D2
        ANDI    #$4,D2          ;Clear all bits except the 2nd bit
        OR      D2,D0           ;Store the O2 from D2 to D0

*O3 = O2 AND I6
        MOVE.L  D0,D2           ;Copy the current output to D2
        MOVE.L  D1,D3           ;Copy the input to D3
        ROL     #1,D2           ;Move bit O2 to the fourth bit(O3)
        ROR     #3,D3           ;MOVE bit I6 to the fourth bit(O3)
        AND     D3,D2           ;D2 = O2 AND I6
```

```
        ANDI     #$8,D2          ;Clear all bits except the 4th(O3) bit
        OR       D2,D0           ;Store bit O3 to D0

*O6 = O3
        MOVE.L  D0,D2            ;Copy the current output to D2
        ROL      #3,D2           ;Move bit O3 to location O6
        ANDI     #$40,D2         ;Clear all bits except the 7th(O6) bit
        OR       D2,D0           ;Store bit O6 to D0

*05 = compliment of O6
        MOVE.L   D0,D2           ;Move the current output to D2
        ROR      #1,D2           ;Move bit O6 to location O5
        NOT      D2              ;D2= compliment of D2
        ANDI     #$20,D2         ;Clear all bits except the 6th(O5) bit
        OR       D2,D0           ;Store bit O5 to D0

*O4 = I2 AND I3
        MOVE.L   D1,D2           ;Move input to D2
        MOVE.L   D1,D3           ;Move input to D3
        ROL      #2,D2           ;Move bit I2 to location O4
        ROL      #1,D3           ;Move bit I3 to location O4
        AND      D3,D2           ;D2= D2 AND D3
        ANDI     #$10,D2         ;Clear all bits except the 5th(O4) bit
        OR       D2,D0           ;Store bit O4 to D0

*O7 = compliment of I4
        MOVE.L   D1,D2           ;Move input to D2
        ROL      #3,D2           ;Move bit I4 to location O7
        NOT      D2              ;D2 = compliment of D2
        ANDI     #$80,D2         ;Clear all bits except the 8th(O7) bit
        OR       D2,D0           ;Store bit O7 to D0

        MOVE.B D0, $800
 ********************************************************NEW


*BINARY TO BCD CONVERTER
*D2:A PLACE FOR TEMP CALULATION
*D3:HIGHEST BCD BYTE
*D4:SECOND BCD BYTE
*D5:LOWEST BCD BYTE
    CLR D3
    CLR D4
    CLR D5
        MOVE.L   D0,D2           ;Copy output to D2
        DIVU     #100,D2         ;Get the highest byte
```

```
        MOVE.B  D2,D3           ;Move the highest BCD byte to D3
        MOVE.W  D0,D2           ;Copy output to D2
        CLR.W   D2              ;Clear the lower word of D2
        SWAP    D2              ;SWAP to get the higher word
        DIVU    #10,D2          ;Get the second BCD byte to D2
        MOVE.B  D2,D4           ;Move the second BCD byte TO D4
        SWAP    D2              ;SWAP to get the lowest BCD, which is quotient
        MOVE.B  D2,D5           ;Move the lowest BCD byte to D5

        MOVE.B  D3,$900
        MOVE.B  D4,$901
        MOVE.B  D5,$902
        MOVEA.L #$900,A6
        MOVEA.L #$900,A5
        ADD.B   #$30,(A6)+
        ADD.B   #$30,(A6)+
        ADD.B   #$30,(A6)+
        MOVE.B  #227,D7
        TRAP    #14

*OUTPUT TO USER DATA DISPLAY
    CLR.L   D6
        MOVE.B  D4,D6           ;Move second BCD byte to D6
        ROL.B   #4,D6
        ADD.B   D5,D6


*SINCE USER DATA DISPLAY IS 3 BYTES
        MOVE.B  #00,$A0001
        MOVE.B  D3,$90000
        MOVE.B  D6,$90001

*******************************************************
    MOVE.B #228,D7
    TRAP #14

    END     START       ; last line of source
```

3.2 Sample Program S-Records

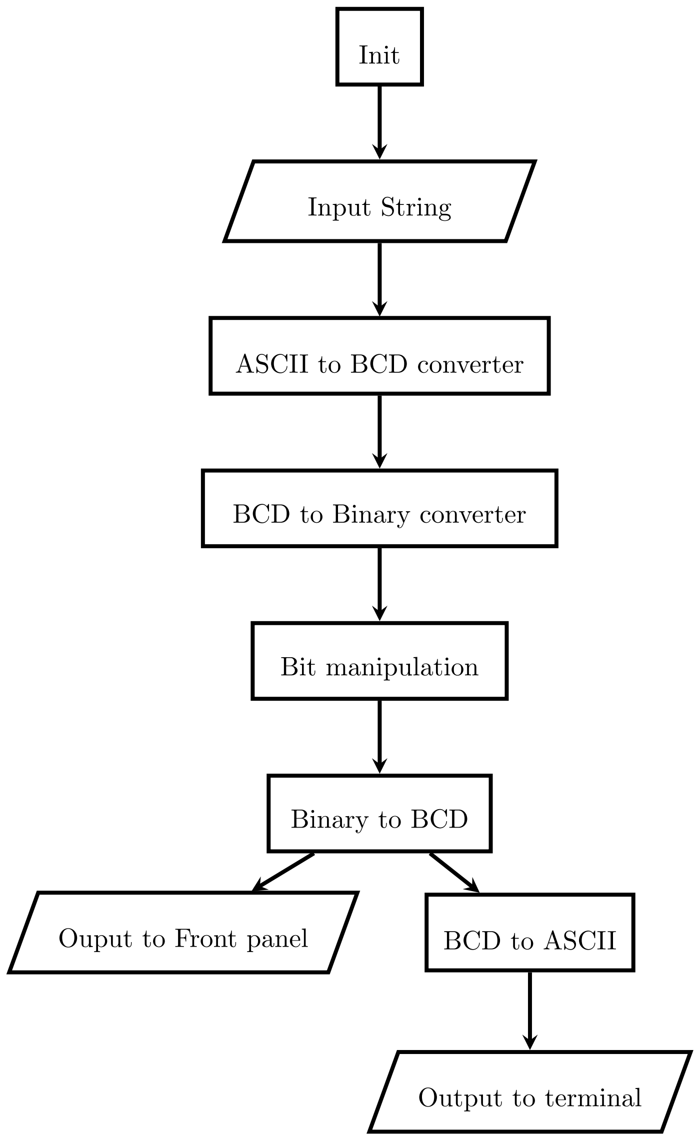
a.
- Type: S0
- Length: 21
- Address: 0000
- Data: 36384B50524F4720202032304352454154454420425920454153593638 4B
- Checksum: 6D

b.
- Type: S1
- Length: 0C
- Address: 0900
- Data: 495420574F524B5321
- Checksum: 76

c.
- Type: S1
- Length: 23
- Address: 1000
- Data: 4FF82FFF4BF809004DF809091E3C00F34E4E1E3C00F14E4E1E3C00E34E4E60E4
- Checksum: C7

d.
- Type: S8
- Length: 04
- Address: 0000
- Data: 00
- Checksum: FB

3.3 Block Diagram



4 Conclusions

This experiment was accomplished. TUTOR was introduced, as well as M68k instructions. From this building block, students can work on more and more complex programs for SANPER and can continue to learn about the functioning of the machine.