Experiment No. 02:

TUTOR COMMAND UTILIZATION and PROGRAM EXPERIMENTATION

Instructor: Dr. Jafar Saniie

ECE 441-001

Lab Date: 09-10-2015

Due Date: 09-24-2015

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature : ______________________

I. Introduction

A. Purpose

The Purpose of this Lab is to utilize SANPER-1 Educational Lab Unit to write 6 programs. This will help us understand assembly language (MC68000) better and be proficient in upcoming labs.

B. Background

Best way to learn MC68000 is to write assembly language program. This experiment will have users write 6 different codes, execute it and debug it. SANPER-1 ELU will be used to write the TUTOR commands.

II. Lab Procedure and Equipment List

A. Equipment

Equipment

- SANPER-1 system
- PC with TUTOR software
- EASY 68k

B. Procedure

1. Download the Lab manual for
2. Perform the programs instructed in Sample Program 2.1,2,3,4,5,6
3. Record the results gathered by running this program.
4. Allow TA to check the code.
5. Redo any code that does not work properly

III. Results and Analysis

SAMPLE 2.1

```
TUTOR 1.X > MM            Instruction               Comments
$300C;DI <CR>
  Address
00300C                 MOVE.W D0,(A0)+ <CR>      To be determined by user
00300E                             CMP.W A0,A1 <CR>
003010                             BGE $300C <CR>
003012                             MOVE.B #228,D7 <CR>
003016                             TRAP #14 <CR>
003018                             . <CR>
```

Code

```
   ORG    $300C

INIT:
    ADD.L #2,A1

START:

    MOVE.W D0,-(A1)
    CMP.W A0,A1
    BGE START
    MOVE.B #228,D7
    TRAP #14

    SIMHALT

* Put variables and constants here
```

SAMPLE 2.2

```
    END     START
TUTOR 1.3 > MM
$900;DI<CR>
```

Address	Instruction	Comments
000900	**MOVE.B #$??,D0 <CR>**	To be determined by user
000904	**MOVE.B #248,D7 <CR>**	
000908	**TRAP #14 <CR>**	
00090A	**MOVE.L #$FFFF,D5 <CR>**	
000910	**DBEQ D5,$910 <CR>**	
000914	**BRA $900 <CR>**	
000916	**. <CR>**	

CODE

```
    ORG     $900

START:                      ;

    MOVE.B #'A',D0
    MOVE.B #248,D7
    TRAP #14
    MOVE.L #$FFFF,D5
    DBEQ D5,$910
    BRA $900

    SIMHALT

* Put variables and constants here

    END     START
```

SAMPLE 2.3

Address	Instruction	Comments
000950	**MOVE.L #$??,A5 <CR>**	To be determined by user
000956	**MOVE.L #$??,A6 <CR>**	
00095C	**MOVE.B #227,D7 <CR>**	
000960	**TRAP #14 <CR>**	
000962	**MOVE.B #228,D7 <CR>**	

```
000966                    TRAP #14 <CR>
000968                    . <CR>
```

CODE

```
   ORG    $950

START:

        LEA.L STR1,A5
        LEA.L END1,A6
        MOVE.B #227,D7
        TRAP #14
        MOVE.B #228,D7
        TRAP #14

        SIMHALT
        ORG $1000
STR1:   DC.B    'WELCOME TO THE JUNGLE'
END1:   DC.B    0

        END    START
```

SAMPLE 2.4

```
TUTOR 1.3 > MM
$1000;DI <CR>
Address                 Instruction             Comments
001000                  MOVE.L #$????,A0        To be determined
                        <CR>                    by user
001006                            MOVE.L #$????,A1 <CR>
00100C                            MOVEQ.L #-1,D1 <CR>
00100E                            MOVEQ.L #0,D0 <CR>
001010                            MOVE.B (A0),D0 <CR>
001012                            CMPM.B (A0)+,(A1)+ <CR>
001014                            DBNE D0,$???? <CR>
001018                            BNE.S $101C <CR>
00101A                            NOT.B D1 <CR>
00101C                            MOVE.B D1,$1100 <CR>
001020                            MOVE.B #228,D7 <CR>
001024                            TRAP #14 <CR>
001026                            . <CR>
```

CODE

```
  ORG    $1000

START:
        MOVE.L  #$AAAA, $1100
        MOVE.L  #$2000, A0
        MOVE.L  #$3000, A1
        MOVEQ.L #-1,D1
        MOVEQ.L #0,D0
        MOVE.B  (A0),D0

LOOP:   CMPM.B  (A0)+,(A1)+
        DBNE    D0, LOOP

        BNE.S   $101C
        NOT.B   D1
SKIP:   MOVE.B  D1, $1100
        MOVE.B  #228,D7
        TRAP    #14

    SIMHALT

    ORG    $2000
    DC.B   22,'MC68000 MICROPROCESSOR'

    ORG    $3000
    DC.B   22,'MC68000 MICROPROCESSOR'

    END    START
```

SAMPLE 2.5

TUTOR 1.3 > **MM**
$2000;DI <CR>

Address	Instruction	Comments
002000	**MOVE.L A0,A2 <CR>**	To be determined by user
002002	**MOVE.L A2,A0 <CR>**	
002004	**CMP.W (A0)+,(A0)+ <CR>**	
002006	**BHI.S $2014 <CR>**	
002008	**SUBQ.L #2,A0 <CR>**	
00200A	**CMP.L A0,A1 <CR>**	
00200C	**BNE $2004 <CR>**	
00200E	**MOVE.B #228,D7 <CR>**	
002012	**TRAP #14 <CR>**	
002014	**MOVE.L –(A0),D0 <CR>**	
002016	**SWAP.W D0 <CR>**	
002018	**MOVE.L D0,(A0) <CR>**	
00201A	**BRA $2002 <CR>**	
00201C	**. <CR>**	

CODE

```
ORG    $2000

START:

        LEA TABLE,A0
        LEA TBEND,A1

        MOVE.L A0,A2

LBL0:   MOVE.L A2,A0
LBL1:   CMP.W (A0)+,(A0)+
        BHI.S LBL2
        SUBQ.L #2,A0
        CMP.L A0,A1
        BNE LBL1
        MOVE.B #228,D7
        TRAP #14
LBL2:   MOVE.L -(A0),D0
        SWAP.W D0
        MOVE.L D0,(A0)
        BRA LBL0

    SIMHALT

    ORG     $2100

TABLE:  DC.W    $00, $10, $20, $30, $40, $50, $60, $70
TBEND:  DC.W    0

    END    START
```

SAMPLE 2.6

```
TUTOR 1.3 > MM                 Instruction               Comments
$3000;DI <CR> Address

003000                         CMP.W (A0),D0 <CR>        To be determined by user

003002                                    BCC $300C <CR>

003004                                    MOVE.W (A0),-(A0) <CR>

003006                                    ADDQ #4,A0 <CR>

003008                                    CMPA.L A0,A1 <CR>

00300A                                    BCC $3000 <CR>

00300C                                    MOVE.W D0,-(A0) <CR>

00300E                                    MOVE.B #228,D7 <CR>

003012                                    TRAP #14 <CR>

003014                                    . <CR>
```

CODE

```
ORG     $1000

        LEA     Str1, A5
        LEA     End1, A6
        MOVE.B  #243,D7
        TRAP    #14

        MOVE    #241,D7
        TRAP    #14

        MOVE    #226,D7
        TRAP    #14

        BRA     INSERT

    ORG    $1500

INSERT: CMP.W   (A0),D0
        BCC     LBL1
        MOVE.W  (A0),-(A0)          ADDQ    #4,A0
        CMPA.L  A0,A1
        BCC     START

LBL1:   MOVE.W  D0,-(A0)

        MOVE.B  #228,D7
        TRAP    #14

    SIMHALT

        ORG     $900
Str1    DC.B    'Enter 4 Hex Digits: '
End1    DC.B    0

    END    START
```

A. Discussion

SAMPLE 2.1

1. A fully commented version of the original program (it must include both global and local comments)

```
   ORG      $300C       ; Starting Location
   MOVE.W   D0,(A0)+    ; Move word D0 to A0 and post increment
   CMP.W    A0,A1       ; Compare address A0 and A1
   BGE      $300C       ; Branch if A0 is greater than or equal to AL
   MOVE.B   #228,D7     ; EXIT
   TRAP     #14
```

2.

```
     ORG    $300C

INIT:
    ADD.L #2,A1      ;Add 2 ( 1 word ) to A1 so pre-decrement can reach full
address.

START:

    MOVE.W D0,-(A1); Move word from D0 to A1 and pre decrement A1 first
    CMP.W A0,A1    ; Compare A0 and A1
    BGE START      ; Branch to Start if A0 is >= to A1
    MOVE.B #228,D7 ; Exit
    TRAP #14       ; Return to TUTOR

    SIMHALT
```

3. Discuss the function of each register used in the original program.
   - D0 saves new contents so it can be moved to memory later.
   - A0 starting memory address
   - A1 end memory address
   - D7 Saves Trap #14.

4. Discuss the advantages of the pre-decrementing and post-incrementing addressing modes.

- Pre-decrementing allows instructions to flow without using another line of code to tell the register to move to the next one. It will automatically increase the index value of program instructions while other modes execute regularly.

- Post-increment allows program to access memory before it increments index value. It will automatically keep receiving the next high memory address.

Sample 2.2

1. A fully commented version of the original program (it must include both local and global comments).

- lab2 2.2

```
ORG         $900         ; Original location
MOVE.B      #$41,D0      ;Move byte "A" to D0
MOVE.B      #248,D7      ; Print function
TRAP        #14          ; Print
MOVE.L      #$FFFF,D5    ; Move long FFFF to D5
DBEQ        D5,$910      ;Branch when D5 equals to 0
BRA         $900         ; Branch back to beginning.
```

2. Describe the output results of procedure #10, and explain what caused the difference.

- Procedure 10 changes $000F to $FFFF which in original code it prints infinite 55555... But changing the code prints only one 5. This is because the infinite loop is changed.

3. Write a subroutine that outputs any character once. The character to be outputted will be passed to this subroutine through Data Register D1. Note: you are not required to execute this program on the SANPER-1 ELU.

```
MOVE.B D1,D0
MOVE.B #248,D7
TRAP #14
MOVE.L #$000F,D5
DBEQ D5,$910
BRA $900
```

4. What is the effect of changing the instruction at address $914 to "BRA 904"?
- Nothing will happen to the output until D0 register is changed.

5. Outline the steps involved in the execution of the instructions at addresses $90A and $910. Discuss the usefulness of this combination of instructions.

- $90A is where the counter is located while $910 is decrementing the D5 register. This function delays the program.

6. List the major benefits of using TRAP instructions.

- TRAP function is where you can use I/O instructions.

- TRAP automatically converts majority of ASCII codes.

SAMPLE 3

1.A fully commented version of the original program (it must include both global and local comments)

```
- ORG           $950            ; Origin of the address
  MOVE.L        #$1000,A5       ; Move address $1000 to A5
  MOVE.L        #$1018,A6       ; Move long $1018 (end) to A6
  MOVE.B        #227,D7         ; Output Trap
  TRAP          #14             ; Output
  MOVE.B        #228,D7         ; Exit
  TRAP          #14
```

2. Discuss how you would have implemented this program were TRAP Function No. 227 not available.

- TRAP 242 is an alternate function to the TRAP 227.

3. After executing the program, what is the final value of A5, and why?
- Final value will be $1018, following the program logic.

SAMPLE 4

1. A fully commented version of the original program (it must include both global and local comments).

```
- ORG       $1000                 ; Original location
  MOVE.L #$2000,A0                ; Move $2000(start) to A0
  MOVE.L #$3000,A1                ; Move $3000(start) to A1
  MOVEQ.L       #-1,D1            ; Default is zero
  MOVEQ.L         # 0,D0          ; String length to 0 in D0
  MOVE.B (A0),D0                  ; Move byte A0 to Do
  CMPM.B (A0)+,(A1)+              ; Compare post decremented A0 with A1 post decremented.
  DBNE    D0,$1012                ; If the compare did not end loop again.
  BNE.S   $101C                   ; if it does equal stop d1
  NOT.B   D1
```

2. Draw a flowchart for the program.

3. Describe the differences between the MOVE and MOVEQ instructions. Under what conditions is it advantageous to use one instruction over the other?

- MOVEQ is better when you need to move data immediately to the data register.

4. Discuss the usefulness of the CMPM instruction
- CMPM is useful because it allows user to post increment both side of the source and destination.

5. What instruction sets the Condition Code bits for the BNE instruction at address $1018?
- CMPM.B (A0)+,(A1)+

Sample 5

1. A fully commented version of the original program (it must include both global and local comments).

TUTOR 1.3 > **MM $2000;DI <CR>** Address	Instruction	Comments
002000	**MOVE.L A0,A2 <CR>**	To be determined by user
002002	**MOVE.L A2,A0 <CR> ; MOVE long A2 to A0**	
002004	**CMP.W (A0)+,(A0)+ <CR> ; Compare A0 and A0 and Post decrement**	
002006	**BHI.S $2014 <CR> ; Branch**	
002008	**SUBQ.L #2,A0 <CR> ; Subtract load 2 from A0**	
00200A	**CMP.L A0,A1 <CR> ; Compare A0 to A1**	
00200C	**BNE $2004 <CR> ; Branch if not equal to $2004 data**	
00200E	**MOVE.B #228,D7 <CR> ; Ready write**	
002012	**TRAP #14 <CR>**	
002014	**MOVE.L –(A0),D0 <CR> ; MOVE long A0 to D0 and pre-decrement -**	
002016	**SWAP.W D0 <CR> ; Reverse D0**	
002018	**MOVE.L D0,(A0) <CR> ; MOVE long D0 to A0**	
00201A	**BRA $2002 <CR> ; Branch to $2002**	
00201C	**. <CR>**	

2. Examine the program and describe how the sorting algorithm has been implemented.
-It adds value to two different address and starts comparing until it reaches the end
-Then it swaps to $2014

3.Describe the significance of the SWAP instruction. Assume for a moment that the 68000 does not have a SWAP instruction. List the set of instructions, in the proper sequence that are necessary to replace the SWAP instruction.

SWAP.W D0

- MOVE.W D0, D1 ; Save D0 to D1

- MOVE.L #$FFFFFFFF, D2 ; Register temporary D2
- ROL #16, D0 Lowest goes to highest
-ROL #16, D1 Highest goes to lowest
-AND.L D0,D2
-AND.L D1,D2
-MOVE.L D2,D0 Result goes to D0

4. Describe the function and advantages of the ADDQ and SUBQ instructions.
-Like MOVEQ it moves the value immediately to the location basically making it faster.

5. Describe the differences between the ADD and ADDQ instructions.
   - This takes immediate data and moves it right to the destination.

6. Describe the differences between the SUB and SUBQ instructions.
   -This takes immediate data and moves it right to the destination.

7. Describe the sequence of events that occurs during the execution of the instruction located at address $2004.

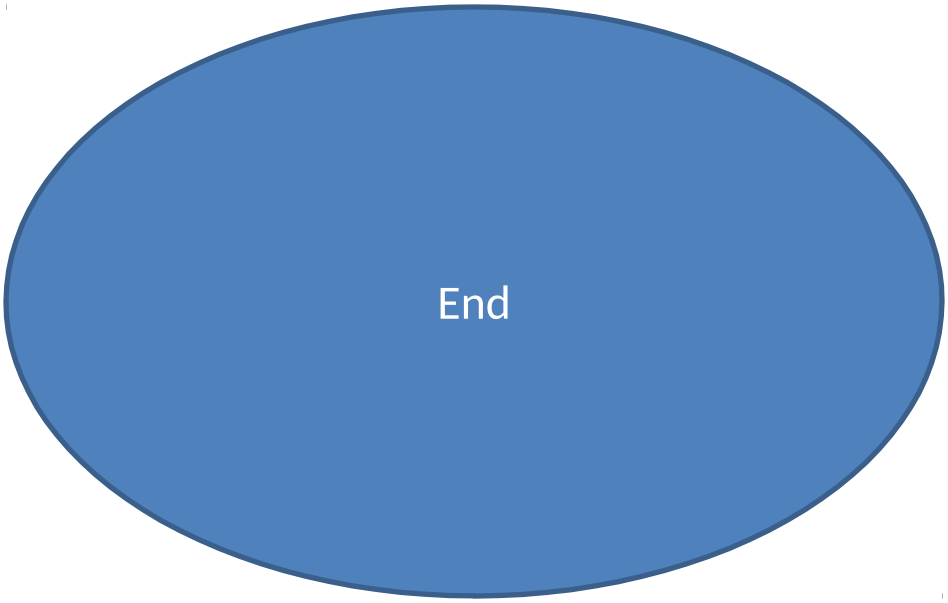
- Comparison of2000 and 2002 and A0 loads to 2004
- High goes to 2014
- 2000 and 2002 swaps
- Then it repeats the comparison process.

Sample 6

1. A fully commented version of both programs (they must include both global and local comments)

```
ORG     $1000

        LEA     Str1, A5
        LEA     End1, A6
        MOVE.B  #243,D7
        TRAP    #14         ; TRAP #14 ready prompt

        MOVE    #241,D7
        TRAP    #14         ; TRAP #14 input collections

        MOVE    #226,D7
        TRAP    #14         ; TRAP #14 store in D0 after conversion

        BRA     INSERT

    ORG    $1500

INSERT: CMP.W   (A0),D0     ; compare A0 and D0
        BCC     LBL1        ; If table value is less than inset value, branch
to LBL1
        MOVE.W  (A0),-(A0)  ; increment the value
        ADDQ    #4,A0
        CMPA.L  A0,A1       ; Compare the memory address
        BCC     START       ; if its in the table Loop

LBL1:   MOVE.W  D0,-(A0)    ; insert the value into A0

        MOVE.B  #228,D7
        TRAP    #14         ; Trap #14 : Return to TUTOR

    SIMHALT                 ; halt simulator

        ORG     $900
Str1    DC.B    'Enter 4 Hex Digits: '
End1    DC.B    0

    END    START        ; last line of source
```

2. Draw a flowchart of the program, and discuss the insertion algorithm.



IV. Conclusions

The Lab was completed successfully. All the experiment was completed with working code. This increased our knowledge about assembly language and prepared for further lab session.

References

1) Experiment 2 Lab Manual

Attachments

None