SAMPLE PROGRAM LISTING:

```
dumbo 29: more labtest.s

        ORG       $1000       ; start at $1000

main    MOVE.W    #$00FFFE,D0 ; x:=y
        ADD.W     #$0001,D0   ; x:=x+1
        ADD.W     #$0001,D0
        ADD.W     #$00FFFE,D0 ; x:= x + $FFFE
        ADD.W     #$0002,D0   ; x:= x+ 2
        LEA.L     $6000,A0    ;
        LEA.L     adr,A2
        MOVEA.W   $2000,A1
        MOVE.W    D0,(A0)

        ORG       $6000
stuff   DC.W      $2000
adr     DC.W      $4000
        DC.W      $5000
        END       main        ; forces debug start at main
```

This program has two parts:

code — begins at $1000 and contains the program instructions

data — begins at $6000 and contains data and storage for program variables

The normal sequence of running a program is:

1. assemble the program
   The input file has a .s extension and the output file has a .o extension
2. link the program
   The input file(s) has a .o extension and the output file has a .x extension.
3. debug the program
   The input file has a .x extension. There is no output file.

ASSEMBLE THE PROGRAM:

```
dumbo 24: as68k -L demoprog.s > demoprog.lis
```

The input file has the extension .s. The assembler creates an output file demoprog.o for use by the linker and a list file demoprog.lis from the assembler. This list file shows the program, the assembled form of each instruction, memory locations used by the program, and all symbols defined in the program and their values. In particular, note the values of `stuff`, `main` and `adr` in the following listing.

```
dumbo 25: more demoprog.lis
Hewlett Packard AS68000 V(01.20 10Mar88) Page 1 Thu May 21 22:50:52

Cmdline - as68k -L demoprog.s
Line Address
1
2
3                                       ORG     $1000         ;start at
$1000
4
5   00001000 303C FFFE          main   MOVE.W  #$00FFFE,D0  ;x:=y
6   00001004 5240                      ADD.W   #$0001,D0     ;x:=x+1
7   00001006 5240                      ADD.W   #$0001,D0
8   00001008 0640 FFFE                 ADD.W   #$00FFFE,D0  ;x:= x + $FFF2
10  0000100E 41F8 6000                 LEA.L   $6000,A0      ;
11  00001012 45F8 6002 4E71            LEA.L   adr,A2
12  00001018 3278 2000                 MOVEA.W $2000,A1
13  0000101C 3080                      MOVE.W  D0,(A0)
14
15                                      ORG     $6000
16  00006000 2000               stuff  DC.W    $2000
17  00006002 4000               adr    DC.W    $4000
18  00006004 5000                      DC.W    $5000
19                                      END     main          ;starts at
main


                    Symbol Table

Label       Value

adr        00006002
main       00001000
stuff      00006000
```

NOTE: There is an important difference between the LEA commands in the above program. The instruction `LEA.L $6000,A0` puts the number $6000 into address register A0. The instruction `LEA.L adr,A2` puts the symbol "`adr`" (which has a value of $6002) into the instruction and has the effect of putting the number $6002 into A2. The `4E71` is a filler for the possibility that the linker would change the absolute value of "`adr`" from a word length $6002 to a different long word value.

LINK THE PROGRAM:

```
dumbo 26: ld68k -L -o demoprog.x demoprog >demoprog.llis
```

The input file is assumed to have the extension .o. The assembler creates an output file `demoprog.x` for use by the debugger and a list file `demoprog.llis` from the linker. This list file shows the location in memory of the program and it's components. In particular, note that there are two components of the program: one begins at address $1000 and the other begins at address $6000. These are the code and data components of the program as defined by the ORG statements in the program. A printout of `demoprog.llis` is shown below:

```
dumbo 27: more demoprog.llis
Hewlett-Packard LD68000 V(01.20 10Mar88)         Thu May 21 22:57:02

Command line: ld68k  -L -o demoprog.x demoprog

LOAD demoprog


OUTPUT MODULE NAME:    demoprog
OUTPUT MODULE FORMAT:  IEEE


SECTION SUMMARY
---------------

SECTION    ATTRIBUTE      START      END        LENGTH     ALIGN

           ABSOLUTE       00001000   0000101D   0000001E   0 (BYTE
           ABSOLUTE       00006000   00006005   00000006   0 (BYTE


MODULE SUMMARY
--------------

MODULE     SECTION:START     SECTION:END        FILE

demoprog          :00001000         :0000101D
/users/merat/eeap282/labs
                  :00006000         :00006005



START ADDRESS:   00001000

Load Completed
```

DEBUGGING/TESTING THE PROGRAM:

`dumbo 28: db68k demoprog`

It is assumed that the input file has the extension .x. The debugger uses this file which contains the program, memory assignments, and symbol definitions and values to simulate the operation of the program on a 68000 microprocessor.

```
=======================Monitor====================12===========Stack=======14=
|1                                                  ||    00000010=4E724E72
|2                                                        0000000C=4E724E72
|3                                                        00000008=4E724E72
|4                                                        00000004=4E724E72
|5                                                      SP->00000000=4E724E72
================================================================================
                         Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=00001000 pi=00000000
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00000000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00000000
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00000000
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
=====================================Journal=================================10=
|  auto halt at address 0x00001000.
|
|
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>

 Breakpoint  Debugger Expression  File  Memory  Program  Symbol  Window
  Execution  HP-UX_Shell  Macro  Option  Pause  Quit  Level  Directory  ?(Help
```

The above screen is what you will see as a result of the db68k command. The monitor and stack areas of the display show information that will be important in future labs. The code area shows the next instruction to be executed and, approximately, the eight instructions following it. This code area displays your program in a format that is very similar to the assembler listing. Shown are the memory address (`00001000`), the assembled form of the instruction (`303C FFFE`), and the mnemonic form of the instruction (`MOVE.W #$FFFE,D0`). The register window shows the current contents of the data and address registers, the program counter (PC), and the status register (SR). When the debugger first starts not all registers and the SR are shown in the register area but will appear after the first instruction is executed. The PC was automatically set to $1000 by using the `END main` instruction in the program. If your program is not visible when you enter the debugger you will need to set the

current value of the PC to the correct starting location of your program using the command
Memory **R**egister @PC=&&&&h
where &&&& is the hexadecimal address of the first instruction to be executed.

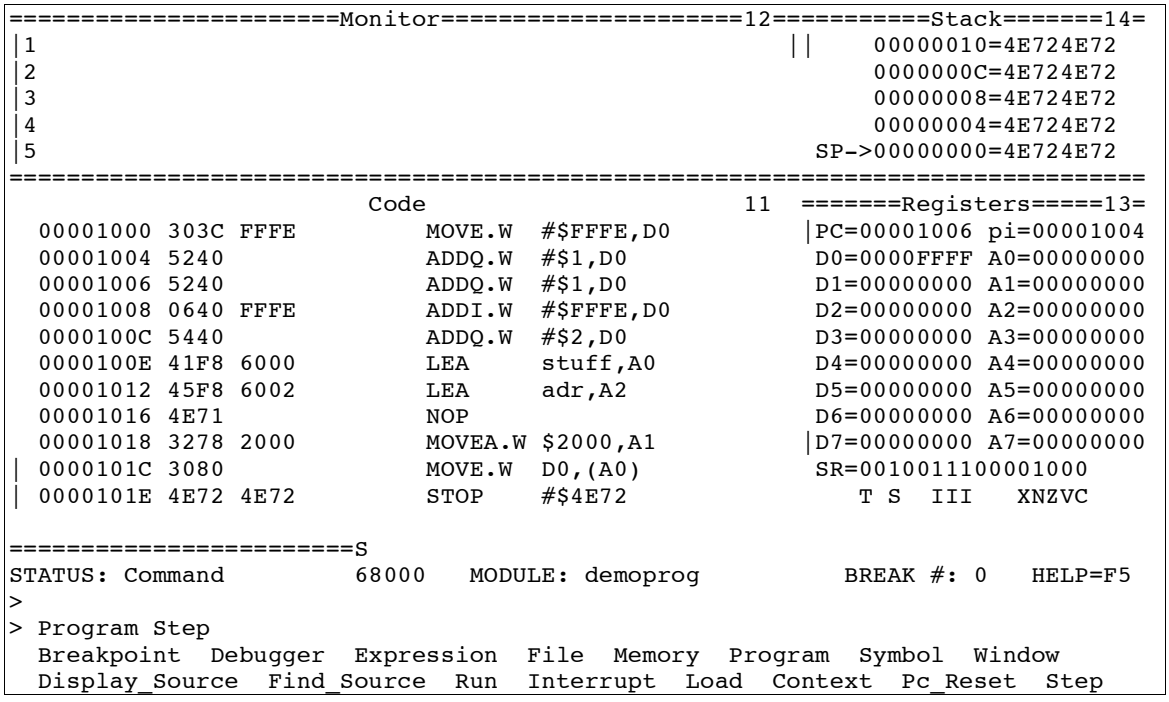
The MOVE instruction at $00001000 can be executed using the debugger command
Program **S**tep
which will execute the instruction pointed to by the PC (in this the MOVE instruction at $00001000), update all register and memory values affected by the instruction, advance the PC to the next instruction and display the new values as shown below. Note that the register window has been enlarged to show the remaining registers and the SR.

```
======================Monitor====================12===========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                        Code                       11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0         |PC=00001004 pi=00001000
  00001004 5240              ADDQ.W  #$1,D0             D0=0000FFFE A0=00000000
  00001006 5240              ADDQ.W  #$1,D0             D1=00000000 A1=00000000
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0          D2=00000000 A2=00000000
  0000100C 5440              ADDQ.W  #$2,D0             D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0           D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2             D5=00000000 A5=00000000
  00001016 4E71              NOP                        D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1          |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)            SR=0010011100001000
| 0000101E 4E72 4E72         STOP    #$4E72                T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

The X,N,Z,V and C bits of the SR are all updated as a result of the MOVE instruction. For a MOVE instruction the XNZVC bits will be updated according to _**00 (See the Programmer's Reference Manual). This means that the X bit doesn't change (indicated by _) from the previous value of zero, the N and Z bits are set according to the number being moved (as indicated by the *, in this case $FFFE is negative so the N bit becomes 1 and the Z bit remains 0), and the V and C bits are always set to 0 (as indicated by the 0) for a MOVE instruction.

You can continue to type
Program **S**tep
to execute the remaining program instructions one at a time. In this case the debugger will execute the ADD.W $0001,D0 instruction which adds a word length $0001 to the contents of D0 to get (D0)=$0000FFFF



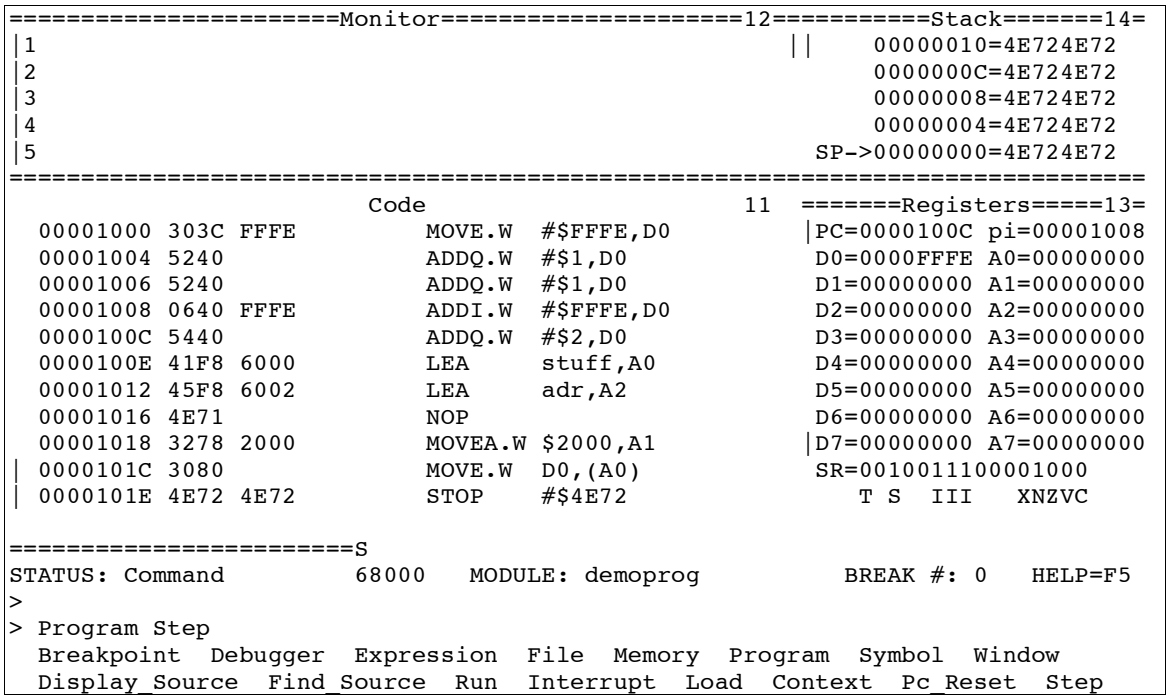
For an ADD instruction the XNZVC bits will be updated according to ***** (See the Programmer's Reference Manual). This means that these bits are set according to the results of the addition. The addition added $0001 to $FFFE to get $FFFF. There is no overflow or carry. The X , C and V bits are accordingly set to zero. Because the result is negative the N bit is set to 1. The result is not zero so the Z bit is set to 0.

You can continue to type
Program **S**tep
to execute the remaining program instructions one at a time.

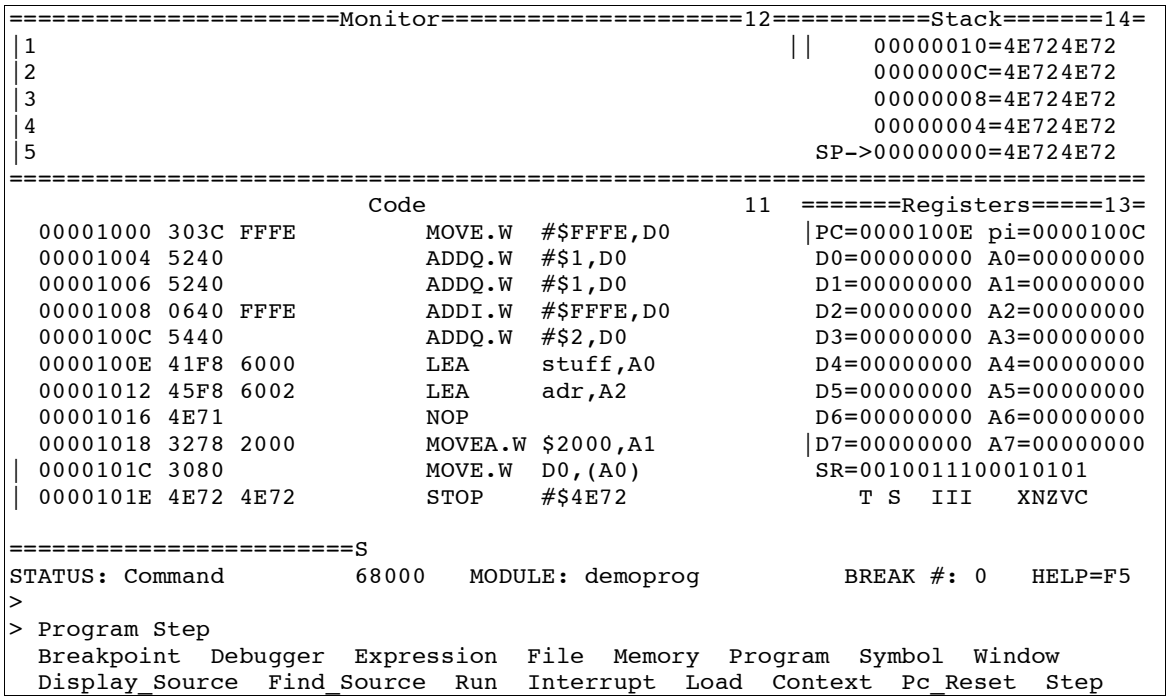
```
======================Monitor====================12==========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                         Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=00001008 pi=00001006
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00000000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00000000
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00000000
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1           |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)             SR=0010011100010101
| 0000101E 4E72 4E72         STOP    #$4E72                 T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

This addition added $0001 to $FFFF to get $10000. However, because the addition is word length the 1 cannot be placed in the lower word of D0 and a carry occurs. This sets the C and X bits to 1. Because the signs of the numbers are mixed no overflow occurs and V is set to 0. Because the result put into D0 is zero the Z bit is set to 1. The N bit is set to 0.



This addition added $FFFE to $0000 to get $FFFE. Because there is no overflow or carry the X , C and V bits are set to zero. Because the result is negative the N bit is set to 1. The number is not zero so the Z bit is set to 0.



This addition added $0002 to $FFFE to get $10000. Note that additions take place in 2's complement. Because $FFFE is -2, adding +2 to to -2 should result in zero and that is what happened. Because the addition is word length the 1 cannot be placed in the lower word of D0 and a carry occurs. This sets the C

and X bits to 1. Because the signs of the numbers are mixed no overflow occurs and V is set to 0. Because the result put into D0 is zero the Z bit is set to 1. The N bit is set to 0.

```
======================Monitor====================12==========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                        Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=00001012 pi=0000100E
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00000000
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00000000
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1           |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)             SR=0010011100010101
| 0000101E 4E72 4E72         STOP    #$4E72                 T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

The instruction `LEA stuff,A0` put the number $6000 into address register A0. Note that the original instruction in our program was `LEA.L $6000,A0.` The debugger recognized that $6000 was the value of the symbol stuff and automatically substituted it. The LEA instruction does not affect the SR so no SR values are changed.

```
======================Monitor====================12==========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                        Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=00001018 pi=00001016
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00000000
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00006002
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1           |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)             SR=0010011100010101
| 0000101E 4E72 4E72         STOP    #$4E72                 T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

The instruction `LEA adr,A2` put the number $6002 into address register A2. Note that the PC automatically advances beyond the NOP (the $4E71) inserted

by the assembler. No long word address was needed since our address $6000 is representable by a single word. The LEA instruction does not affect the SR so no SR values are changed.

```
======================Monitor====================12==========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                        Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=0000101C pi=00001018
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00004E72
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00006002
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1           |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)             SR=0010011100010101
| 0000101E 4E72 4E72         STOP    #$4E72                 T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

The instruction `MOVEA.W $2000,A1` put the contents of memory location $2000 into address register A1. Since we did not put anything into that particular memory location the debugger found the default value of $4E72 there. (The debugger initializes all memory to $4E72.) The MOVEA instruction does not affect the SR so no SR values are changed.

```
======================Monitor====================12==========Stack=======14=
|1                                                ||    00000010=4E724E72
|2                                                      0000000C=4E724E72
|3                                                      00000008=4E724E72
|4                                                      00000004=4E724E72
|5                                                    SP->00000000=4E724E72
================================================================================
                        Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=0000101E pi=0000101C
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00004E72
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00006002
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
  00001018 3278 2000         MOVEA.W $2000,A1           |D7=00000000 A7=00000000
| 0000101C 3080              MOVE.W  D0,(A0)             SR=0010011100010100
| 0000101E 4E72 4E72         STOP    #$4E72                 T S  III   XNZVC

========================S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

The instruction `MOVE.W D0,(A0)` puts the contents of D0 (in this case, $0000) into the memory location whose address is in A0 (in this case, $6000). Because

the number is zero the Z bit is set to 1. The N, V and C bits are set to 0. The X bit remains unchanged.

```
=======================Monitor=====================12===========Stack=======14=
|1                                                  ||    00000010=4E724E72
|2                                                        0000000C=4E724E72
|3                                                        00000008=4E724E72
|4                                                        00000004=4E724E72
|5                                                      SP->00000000=4E724E72
================================================================================
                         Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=0000101E pi=0000101E
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00004E72
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00006002
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
=====================================Journal=====================================10=
|
|  Exception vector 8 at 101E: privilege violation
|
|  Stopped due to exception interrupt
S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Program Step
  Breakpoint  Debugger  Expression  File  Memory  Program  Symbol  Window
  Display_Source  Find_Source  Run  Interrupt  Load  Context  Pc_Reset  Step
```

When we attempt to execute the next instruction we get a error. This is because our program did not extend to this memory location and it contained the initial value of $4E72 put there by the debugger. There is no elegant way to end your program in the debugger (that we have shown you yet).

```
=======================Monitor=====================12===========Stack=======14=
|1                                                  ||    00000010=4E724E72
|2                                                        0000000C=4E724E72
|3                                                        00000008=4E724E72
|4                                                        00000004=4E724E72
|5                                                      SP->00000000=4E724E72
================================================================================
                         Code                        11  =======Registers=====13=
  00001000 303C FFFE         MOVE.W  #$FFFE,D0          |PC=0000101E pi=0000101E
  00001004 5240              ADDQ.W  #$1,D0              D0=00000000 A0=00006000
  00001006 5240              ADDQ.W  #$1,D0              D1=00000000 A1=00004E72
  00001008 0640 FFFE         ADDI.W  #$FFFE,D0           D2=00000000 A2=00006002
  0000100C 5440              ADDQ.W  #$2,D0              D3=00000000 A3=00000000
  0000100E 41F8 6000         LEA     stuff,A0            D4=00000000 A4=00000000
  00001012 45F8 6002         LEA     adr,A2              D5=00000000 A5=00000000
  00001016 4E71              NOP                         D6=00000000 A6=00000000
=====================================Journal=====================================10=
|
|  Exception vector 8 at 101E: privilege violation
|
|  Stopped due to exception interrupt
S
STATUS: Command         68000   MODULE: demoprog          BREAK #: 0   HELP=F5
>
> Debugger Quit Yes
  <return>
```

We can exit the debugger with the command

Debugger **Q**uit **Y**es