EXAMPLE: SIMPLE MATH PROGRAM

The correct way to design a program is by starting with your inputs, outputs and functional requirements.

This program accepts as input a 16-bit signed number N and outputs the following values:

N 2*N 16*N N DIV 2 N DIV 16

Functional specification (pseudocode)

```
get signed number N
multiply by 2 using left shift by 1
multiply by 16 using left shift by 4
divide by 2 using right shift by 1
divide by 16 using right shift by 1
```

MC68000 assembly code for simple math program:

```
            ORG       $5000           ;put data here
NEWLINE     DC.B      $0A,$0          ;ascii code for carriage return
                                      followed by end of string
                                      character "0"
            include   io.s            ;insert appropriate code for io
                                      routines
start       JSR       HexIn           ;get N and put in D0
            MOVE.W    D0,D1           ;copy N to D1 for safekeeping
            JSR       HexOut          ;output N
            ASL.W     #1,D0           ;multiply N by 2 by shifting left by
                                      1
            JSR       HexOut          ;output 2*N
            MOVE.W    D1,D0           ;get new copy of N
            ASL.W     #4,D0           ;multiply N by 2^4 by shifting left
                                      by 4
            JSR       HexOut          ;output 2^4*N
            MOVE.W    D1,D0           ;get new copy of N
            ASR.W     #1,D0           ;divide N by 2 by shifting right by
                                      1
            JSR       HexOut          ;output N DIV 2
            MOVE.W    D1,D0           ;get new copy of N
            ASR.W     #4,D0           ;divide N by 2^4 by shifting right by
                                      4
            JSR       HexOut          ;output N DIV 2^4
            LEA       NEWLINE,A0      ;load starting address of new line
                                      control characters into A0
            JSR       PrintString
            END       start
```

EXAMPLE: BLANK SEARCH PROGRAM

This program will search a string of ASCII characters for the first non-blank character and return the address of this character.

STRING	sequence of ASCII characters
START	starting address of STRING in memory
POINTER	address of first non-blank character in STRING

Functional specification (pseudocode)

```
point = START;

LOOP:
IF character(point) = blank THEN
        point = point + 1;
        goto LOOP;
        END
POINTER = point;
```

MC68000 assembly code for blank search program:

```
            ORG     $2000
START       DS.L    1           ;contains starting address of
                                string
POINTER     DS.L    1           ;answer, will contain address of
                                first non-blank character
BLANK       equ     $32         ;ASCII code for blank space
            include io.s        ;insert appropriate code for io
                                routines
start       MOVEA.L START,A0    ;set A0 to start of string
            MOVEA.L POINTER,A1  ;set A1 to answer
            MOVE    #BLANK,D1   ;put ASCII blank into D1
LOOP        CMP.B   (A0)+,D1    ;is current character a blank?
            BEQ     LOOP        ;if YES, then continue looping
            SUBA    #1,A0       ;if NO, then point = point -1 to
                                correct for previous (A0)+
            MOVE.L  A0,(A1)     ;save address of first non-blank
                                character in POINTER
            END     start
```

EXAMPLE: ASCII SEARCH PROGRAM (2)

This program will search a block of memory containing ASCII characters for a specified character and return the address of the first occurrance of the specified character.

CHAR character to search for
BLOCK memory block containing ASCII characters
START starting address of BLOCK in memory
STOPA ending address of BLOCK in memory
POINTER address of specified character in BLOCK

Functional specification (pseudocode)

```
point = START;

LOOP:
IF character(point) ≠ CHAR THEN
        BEGIN
        point = point + 1;
        IF point ≤ STOP THEN goto LOOP;
        END
POINTER = point - 1;
```

MC68000 assembly code for ascii search program:

```
        ORG     $70000
BSTART  DC.L    $2000       ;start of BLOCK to search
BSTOP   DC.L    $4000       ;end of BLOCK to search

CHAR    equ     $40         ;ASCII character to search for
prog    MOVEA.L BSTART,A0   ;set A0 to start of BLOCK
        MOVEA.L BSTOP,A1    ;set A1 to end of BLOCK
        MOVE    #CHAR,D1    ;put ASCII character into D0
LOOP    CMP.B   (A0)+,D0    ;is current character what we are
                            searching for?
        BEQ     DONE        ;if YES, then get out of here
        CMPA.L  A0,A1       ;if NO, then have we searched
                            entire block?
        BCC     LOOP        ; this is a CARRY CLEAR
                            instruction and is equivalent to ≤
                            comparison since there will be no
                            carry (actually borrow in this
                            case) if A0 ≤ A1
DONE    SUBA    #1,A0       ;adjust A0 to correct for the post
                            increment in the CMP instruction
        END     prog
```

EXAMPLE: WORD SEARCH PROGRAM

This program will search for a given word in memory.

WORD word to search for
BLOCK block of memory containing ASCII characters
START starting address of BLOCK in memory
STOP ending address of BLOCK in memory
POINTER address of specified character in BLOCK

Functional specification (pseudocode)

```
point = START;

LOOP:
IF word(point) = WORD THEN
        BEGIN
        point = point + 2;
        IF point ≤ STOP THEN goto LOOP;
        END
POINTER = point - 2;
```

MC68000 assembly code for word search program:

```
        ORG     $3000
START   DC.L    $2000       ;start of memory to search
STOPA   DC.L    $4000       ;end of memory to search
WORD    DC.W    $4E40       ;word to search for

prog    MOVEA.L START,A0    ;set A0 to starting address of
                            search
        MOVEA.L STOPA,A1    ;set A1 to ending address of
                            search
        MOVE    WORD,D0     ;put search word into D0
LOOP    CMP.W   (A0)+,D0    ;is current word what we are
                            searching for?
        BEQ     DONE        ;if YES, then get out of here
        CMPA.L  A0,A1       ;if NO, then have we searched all
                            required memory?
        BCC     LOOP        ; this is a CARRY CLEAR
                            instruction and is equivalent to ≤
                            comparison since there will be no
                            carry (actually borrow in this
                            case) if A0 ≤ A1
DONE    SUBA.L  #2,A0       ;adjust A0 to correct for the post
                            increment in the CMP instruction.
                            Note that since it was
                            incremented by a word we must
                            subtract 1 word (2 bytes).
        END     prog
```

EXAMPLE: SEQUENTIAL SEARCH PROGRAM

This program implements a sequential search program defined as:

> Given an N-element list of 16-bit numbers and a KEY, store the KEY in the N+1-st element of the list. Execute a sequential search of the list for KEY. KEY will always be found. If the address of the matching location is NOT the N+1-st element's address, the KEY was in the list. Otherwise, it is not present.

The program uses:

N	the number of elements in the list to search
KEY	the 16-bit number to search for
LIST	set of 16-bit numbers to search

The program outputs one of the following:

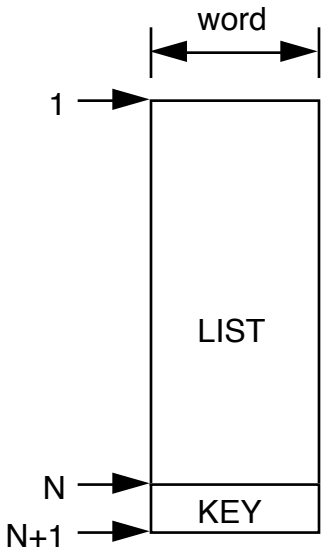
> <value of KEY> is in the list.
> <value of KEY> is NOT in the list.

The program uses the DBEQ instruction to implement the search loop.

Functional specification (pseudocode)

```
input (N);
input (KEY);
LIST(N+1)=KEY;

FOR j=0 to N+1
        IF LIST(j) = KEY THEN KEYADDR=j;
IF KEYADDR≠N+1 THEN
        output(KEY in list.")
        ELSE
        output(KEY NOT in list.");
```



MC68000 assembly code for key search program:

```
            ORG     $5000
LIST:       DS.W    20              ;reserve space for 20 words
FNDMSG      DC.B    'IS IN THE LIST',0
NOTMSG      DC.B    'IS NOT IN THE LIST',0
NEWLINE     DC.B    $0A,0           ;new line command message
            include io.s            ;enter i/o declarations
START       JSR     HexIn           ;enter N into D0
            MOVE.W  D0,D1           ;store N in D1 for DB instructions
            SUBQ.W  #1,D1           ;correct N for DB instruction
            MOVE.W  D0,D2           ;save original N in D2
            LEA     LIST,A0         ;put starting LIST address into A0
LOAD        JSR     HexIn           ;enter entire LIST from keyboard
            MOVE.W  D0,(A0)         ;put in LIST
            ADDA    #2,A0           ;increment LIST address
            DBRA    D1,LOAD         ;decrement and repeat until done
            JSR     HexIn           ;get KEY
            LEA     LIST,A0         ;reset starting address
            LEA     LIST,A1         ;set working address
            ASL.W   #1,D2           ;double D2 for byte count since
                                    words
            ADDA    D2,A1           ;set A1 to end of LIST
            MOVE.W  D0,(A1)         ;put KEY at end of LIST
COMPARE     CMP.W   (A0),D0         ;LIST(j) = KEY?
            BEQ.S   OUTPUT          ;if yes then stop
            ADDA    #2,A0           ;if no then increment by one word
            BRA     COMPARE         ;and repeat
OUTPUT      MOVE.L  A0,D0
            JSR     HexOut          ;print address of where key was
                                    found
            CMPA.L  A0,A1           ;was KEY found in LIST? Is A0
                                    equal to end of LIST?
            BEQ.S   NOTFND          ;if not equal then KEY was not in
                                    LIST
```

```
            LEA     FNDMSG,A0       ;load starting address of
                                    message for KEY found
            BRA     PRINTIT
NOTFND      LEA     NOTMSG,A0       ;load starting address of
                                    message for KEY not found
PRINTIT     JSR     PrintString
            LEA     NEWLINE,A0      ;load starting address of new line
                                    command
            JSR     PrintString
            END     START
```

Comments on use of DBcc instruction in this program:

```
            MOVE.W  D2,D1           put N-1 into D1 for loop count
            SUBQ.W  #1,D1

COMPARE     CMP.W   (A0)+,D0        compare (A0) with KEY
            DBEQ    D1,COMPARE      if they are equal then fall through
                                    else goto compare.
```