TMS320 DSP
Number 42
DESIGNER'S NOTEBOOK
TEXAS INSTRUMENTS

Binary Search Algorithm on the TMS320C5x

Contributed by Lawrence Wong

Design Problem How do I implement an efficient binary search algorithm on the TMS320C5x Family that will take advantage of the 'C5x's capability?

Solution There are many ways to implement the classical binary search algorithm but very few would take advantage of the 'C5x's advanced architecture and instruction set. The following is one of the many possible examples using the TMS320C5x executing the binary search algorithm.

The program takes advantage of the 'C5x's capability of performing bit-reversed addressing to half the search after each testing and therefore freeing the accumulator for other tasks. Also, instead of using conditional branching to perform the testing, the execute conditional (XC) instruction is used, thereby saving cycles and increasing performance.

This routine performs a binary search on an ordered table. It assumes that the table is ordered from low to high, where the largest number is located in the highest memory of the array. Modifications can be made to reverse the ordering, if necessary.

This program also assumes that the size of the search table is some integer power of 2 (i.e., 2^N where N=11 in the following program). As a result, the search would never pass the last entry in the array. A maximum of N iterations is required to complete the search or determine that the search failed. Modifications can be made if the size of the array is not a power of 2. In order to do this, test conditions will have to be included to determine if the last entry has been passed.

This function returns the address of the found number and it is stored in the ACCUMULATOR. A 0x0000 address in the ACCUMULATOR signifies that the search was unsuccessful.

```
    .bss    NTABLE,800h                 ;Sorted search table from low to high
    .bss    LOOK,1                      ;Search value
    .mmregs
    .text
    •
    •
    •
    call    bsearch
    •
    •
    •
bsearch     lar  AR0,#0800h             ;AR0 size of array
            mar         *,AR0           ;
            mar         *BR0+,AR3       ;Half the size of the array
            lar         AR3,#NTABLE     ;AR3 points to beginning of array
            lacl        #11             ;RPT N Times, Size of Array is 2^N
            samm        BRCR            ;Setup Repeat Block
            ldp         #LOOK
            lacc        LOOK            ;Begin search
            sub         *               ;Compare data at AR3
            bcnd        nothere,LT      ;ERROR not found in this array
            rptb        nothere-1
            bcnd        found,EQ        ;Check if found
            xc          1,GT            ;If too low on array
            mar         *0+,AR0         ;Jump forward
            xc          1,LT            ;If too high on array
            mar         *0-,AR0         ;Jump back
            mar         *BR0+,AR3       ;Half the search space
            lacc        LOOK
            sub  *

nothere     retd                        ;Did not find value in the table
            zac                         ;return 0x0 for failed search
            nop

found       ldp         #0
            apl         #0fffeh,PMST    ;disable block repeat bit
            retd
            lamm AR3                    ;return address of search
            nop
```

Figure 1.