TMS320 DSP
DESIGNER'S NOTEBOOK
Number 59
TEXAS INSTRUMENTS

Designing Macros for the TMS320C5x

Contributed by Jay Reimer

Design Problem

What are the tricks to creating a macro that supports all the normal addressing modes for its parameter fields?

How can I be sure that I'm using the correct instruction (CRGT/CRLT) when I want to do apply a lower/upper limit to a variable?

Solution

The TMS320 Assembler includes a powerful macro capability. It provides an effective way to generate a macro function that appears to be a normal instruction, including support of all the normal addressing modes. An example illustrating a lower-limit macro is used to demonstrate the capability. The lower-limit macro has the characteristics typical of many of the native 'C5x instructions.

```
llimit  .macro  limit,shift,nextar
        .nolist
;/*************************************************************
;*                   FILE INFORMATION                        *//
;*                                                           *//
;*(C) Copyright 1993 Texas Instruments. All rights reserved.*//
;*  Use of copyright notice is precautionary and does imply  *//
;*  publication.                                             *//
;*                                                           *//
;*************************************************************/
;* File     : llimit.asm
;*
;* Comments : Perform a lower limit test. The result will be
;*            in both the accumulator and the accumulator
;*            buffer on exit.
;*            The accumulator buffer must contain the value to
;*            be tested upon entry. A total of three
;*            parameters may be passed to llimit. These
;*            parameters must satisfy the syntax for the
;*            lacc/lacl instructions. The limit parameter may
;*            be an immediate value or an address in data
;*            memory containing the value. If the limit
;*            parameter is provided as a direct address or as
;*            an indirect address, the data page pointer or the
;*            ARP, respectively, must be properly set on entry.
;*            In any case, the limit will be loaded into the
;*            accumulator and accumulator buffer compared.
```

Figure 1: `llimit.asm` *file listing*

```
;/           The greater of the two values, either the original or the lower limit,
;/           will be loaded into both the accumulator and the accumulator buffer.
;/
;/ Usage    : direct:        [label]  LLIMIT  dma [,shift]
;/           indirect:      [label]  LLIMIT  {ind} [,shift [,next ARP]]
;/           long immed:    [label]  LLIMIT  #lk [,shift]
;/           short immed:   [label]  LLIMIT  #k
;/
;/ History  : 12/16/93 - created - Jay Reimer
;/^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
;
;   create macro substitution symbols and assign them components of the limit parameter
;   passed to the macro
;
        .var  tmp1,tmp2,len,shft,nxt
;   copy the first character of limit to tmp1
        .asg  :limit(1):,tmp1
;   get the length of the limit string
        .asg  $symlen(limit),len
;   copy the remaining characters of limit to tmp2
        .asg  :limit(2,len):,tmp2
;   if nextar is a non-zero length string
        .if     ($symlen(nextar))
;   copy nextar to nxt with a leading comma
        .asg  “,:nextar:”,nxt
        .endif
;   if shift is a non-zero length string
        .if     ($symlen(shift))
;   copy it to shift with a leading comma
        .asg  “,:shift:”,shft
;   otherwise, if nxt is a non-zero length string
        .elseif ($symlen(nxt))
;   add a leading “,0" to nxt
        .asg  “,0:nxt:”,nxt
        .endif
;
;   generate the appropriate load accumulator instruction (lacc|lacl) followed by the
;   compare instruction (crgt)
;
;   use direct addressing if limit is a symbol
        .if     ($isname(limit))
        .list
        lacc  :limit::shft:
       .nolist
;   use indirect addressing if the first character of limit is “*”
        .elseif ($symcmp(tmp1,"*")==0)
        .list
        lacc  :limit::shft::nxt:
        .nolist
;   use immedate addressing if the first character of limit is “#”
;   if the value is 0–255 use short immediate addressing
        .elseif (($symcmp(tmp1,"#")==0)&((tmp2&0ffh)==tmp2))
        .list
        lacl  :limit:
        .nolist
;   otherwise, use long immediate addressing
        .elseif ($symcmp(tmp1,"#")==0)
        .list
        lacc  :limit::shft:
```

Figure 1: `llimit.asm` *file listing (continued)*

```
        .nolist
        .else
        .list
        .emsg “ERROR - invalid macro parameter to llimit.”
        .mexit
        .endif
        .list
        crgt
        .endm
```

Figure 1: `llimit.asm` *file listing (continued)*