Context Switch Routine for XINU System on LSI-11

```
/*------------------------------------------------------------------------
 *  ctxsw -- assembler routine for performing context switch,
 *  saving/loading registers
 *
 *  The stack contains three items upon entry to this routine:
 *
 *  SP+4 => address of 9 word save area with new registers + PS
 *  SP+2 => address of 9 word save area for old registers + PS
 *  SP   => return address
 *
 *  The saved state consists of: the values of R0-R5 upon entry, SP+2, PC
 *  equal to the return address, and the PS (i.e., the PC and SP are saved
 *  as if the calling process had returned to its caller.
 *------------------------------------------------------------------------
 */
        .globl _ctxsw           /* declare name global */
_ctxsw:                         /* entry point to proc. */
        mov   r0,*2(sp)         /* save old R0 in old register area */
        mov   2(sp),r0          /* get address of old register area in R0 */
        add   $2,r0             /* increment to saved pos. of R1 */
        mov   r1,(r0)+          /* save R1-R5 in successive locations of old
        mov   r2,(r0)+          /*  process register save are */
        mov   r3,(r0)+
        mov   r4,(r0)+
        mov   r5,(r0)+
        add   $2,sp             /* move SP beyond the return address, as if a */
                                /*  return had occurred */
        mov   sp,(r0)+          /* save stack pointer */
        mov   -(sp),(r0)+       /* save caller's return address as PC */
        mfps  (r0)              /* save processor status beyond registers */
        mov   4(sp),r0          /* get address of start of new register area */
                                /* ready to load registers for the new process */
                                /*  and abandon the old stack */
        mov   2(r0),r1          /* load R1-R5 and SP from the new area */
        mov   4(r0),r2
        mov   6(r0),r3
        mov   8.(r0),r4         /* dot following a number makes it decimal; */
        mov   10.(r0),r5        /*  otherwise it is octal */
        mov   12.(r0),sp        /* have switched stacks now */
        mov   16.(r0),-(sp)     /* push new process PS on new process stack */
        mov   14.(r0),-(sp)     /* push new process PC on new process stack */
        mov   (r0),r0           /* load R0 from new area */
        rtt                     /* load PC, PS, and reset SP all at once */
```