```
;       FILE MESA-NOVA2.ASM
;       R. JOHNSSON
;       LAST MODIFIED May 17, 1978  7:57 AM
        .TITL MesaNova2
        .TXTM B
;
        .ENT PScode
        .ENT MesaNova2
        .ENT MesaNovaSize2
        .ENT STOPImplementer
        .ENT CleanUpQueueImplementer
        .ENT RequeueSubImplementer
        .ENT WakeHeadUser
        .ENT OSFPtr
        .ENT OutLdPtr
        .ENT InLdPtr
        .ENT FinishPtr
        .ENT FinProcPtr
;
        .SREL
PScode: PCORR
MesaNova2: START
MesaNovaSize2: END-START
STOPImplementer: STOP-START+X
CleanUpQueueImplementer: CleanUpQueue-START+X
RequeueSubImplementer: RequeueSub-START+X
WakeHeadUser: WakeHead
OSFPtr: RTRN
OutLdPtr:       OUTLDP
InLdPtr:        INLDP
FinishPtr: FINISH-START+X
FinProcPtr: FINPROC
;
        .NREL
        .GET "Mesa-NovaDefs.asm"

X = 174400              ; where this code will be loaded

START:
PCORR:  PS0-START+X-1+2 ; Absolute address of PS0-1+2

; PS0 thru PS16 must be consecutive locations

PS0:    JSR PSWITCH
PS1:    JSR PSWITCH
PS2:    JSR PSWITCH
PS3:    JSR PSWITCH
PS4:    JSR PSWITCH
PS5:    JSR PSWITCH
PS6:    JSR PSWITCH
PS7:    JSR PSWITCH
PS10:   JSR PSWITCH
PS11:   JSR PSWITCH
PS12:   JSR PSWITCH
PS13:   JSR PSWITCH
PS14:   JSR PSWITCH
PS15:   JSR PSWITCH
PS16:   JSR PSWITCH

pINTPC: INTPC

PSWITCH:
        LDA 0 PCORR
        SUB 0 3         ; AC3 now has interrupt channel number
        LDA 0 CVA,3     ; AC0 now has CVptr to NOTIFY
        LDA 1 @pINTPC   ; Find out where we interrupted from
        MOVL# 1 1 SZC
          BRI           ; Something wrong, probably SWAT abort

NakedNotify:    ;(cvptr)
        mov     0 2 snr         ; test for no cv
          bri
        lda     1 0,2           ; cvptr↑
        movzl#  1 1 szr         ; test for empty, ignore possible ww
          jmp   DoNotify
        subzr   0 0
```

```
        sta     0 0,2           ; cvptr↑ ← [ww,empty]
        bri
DoNotify:
        jsr     CleanUpQueue
        jsr     @WakeHead
        bri

WakeHead: 0

STOP:   LDA 2 currentState      ; COPY STATE POINTER TO AC2
        JSR NOVACODE    ; ADDRESS OF DISPATCH TABLE TO AC3
        JMP @RTRN
        JMP @RTRN
        JMP DOOUTLD
        JMP DOINLD
        JMP DISASTER
        JMP NOVAJSR
NOVACODE:       LDA 0 0,2               ; PICK UP CODE
        ADD 0 3         ; ADD TO TABLE BASE
        JMP 0,3


FINPROC:        0       ; POINTS TO PD FOR FINISH PROCEDURE

FINISH: DIR
        LDA 1 @FINPROC
        MOV# 1 1 SNR
        JMP 1,3
        INC 3 3
        STA 3 RTRN              ; NEW RETURN ADDRESS
        LDA 2 currentState
        STA 1 11,2              ; X
        SUB 1 1
        STA 1 12,2              ; Y
        STA 0 0,2               ; THE FINISH CODE
        SUBZL 0 0
        STA 0 10,2              ; STKP ← 1
        STA 1 @pACTIVE  ; DISABLE ALL INTERRUPTS
        ISZ SDC         ; disable reschedule
        JMP @Emulate

pACTIVE: ACTIVE

DOOUTLD:        SUBZL 0 0
        STA 0 10,2      ; STKP ← 1
        LDA 0 1,2               ; FP
        LDA 1 2,2               ; @MESSAGE
        JSR @OUTLDP
OUTLDP: 0
        STA 0 0,2               ;RETURN VALUE
        JMP @Emulate

DOINLD: SUB 0 0
        STA 0 10,2      ; STKP ← 0
        LDA 0 1,2
        LDA 1 2,2
        JMP @INLDP              ; NEVER RETURNS

DISASTER:       LDA 2 @PuntData ; POINTER TO PUNT DATA
        MOV# 0 0 SNR
          JMP SwatPunt          ; NOT SET UP YET
        INC 2 2
        INC 2 2
        LDA 0 -1,2              ; MESACOREFP
        MOV# 0 0 SNR
          JMP SwatPunt          ; NOT SET UP YET
        SUB 1 1
        JSR @OUTLDP
INLDP:  0
        LDA 0 -2,2              ; MESADEBUGGERFP
        MOV 2 1
        JMP @INLDP
PuntData:       456

RTRN:   0
```

```
SWATTrap:       567
SwatPunt:
        JSR Swat1
        .TXT "Punt!"
Swat1:  MOV 3 1
        LDA 3 @SWATTrap
        JMP 16,3        ; does a CallSwat

; CALL LEAVES FCN-CODE, ADDRESS, ARG ON TOP OF STACK
;       tos     ARG
;       tos-1   ADDRESS
;       tos-2   FCN-CODE
; CALLS NOVA CODE WITH ARG IN AC0, RETURNS AC0 TO TOS
; INTERRUPTS MUST REMAIN OFF!
NOVAJSR:        DSZ 10,2                ; STKP ← STKP - 1
        DSZ 10,2                ; STKP ← STKP - 1
        ; THESE SHOULD NEVER SKIP
        LDA 3 10,2      ; AC3 ← STKP
        ADD 2 3         ; AC3 ← POINTER TO NEW TOS+1
        LDA 0 1,3               ; AC0 ← ARG
        JSR @0,3
        LDA 2 currentState
        LDA 3 10,2      ; AC3 ← STKP
        ADD 2 3
        STA 0 -1,3
        JMP @Emulate

CleanUpQueue:   ;(q) returns (q)
        sta     3 QSreturn
        mov     0 2
        lda     3 0,2   ; p ← q↑
        movzl#  3 3 snr         ; test p = NIL; ignore ww
          jmp   @QSreturn
        lda     1 cleanUpLink,3
        snz     1 1             ; test p.cleanUpLink = NIL
          jmp   @QSreturn

findhead:
        sne     1 3             ; test p.cleanUpLink # p
          jmp   queueempty
        snz     1 1             ; test p.cleanUpLink # 0
          jmp   foundhead
        mov     1 3             ; p ← p.cleanUpLink
        lda     1 cleanUpLink,3 ; load p.cleanUpLink
        jmp     findhead

foundhead:
        mov     3 0             ; head ← p
findtail:
        lda     1 link,3
        sne     0 1             ;test p.link = head
          jmp   foundtail
        mov     1 3             ; p ← p.link
        jmp     findtail

queueempty:
        sub     3 3             ; set q↑ ← NIL when p.cleanUpLink=p
foundtail:
        sta     3 0,2           ; q↑ ← p
        mov     2 0             ; return q
        jmp     @QSreturn

q1:     0
q2:     0
p:      0
QSreturn: 0

RequeueSub:             ; (q1,q2,p)
                        ; returns with p still in AC2
        sta     0 q1
        sta     1 q2
        sta     2 p
        sta     3 QSreturn

        lda     3 link,2
        sub     2 3 szr         ; test p.link = p
```

```
          jmp   delink          ; not equal; must delink
        ; pp = 0 = AC3
        sz      0 0             ; if q1=0
          jmp   cleanStore
        lda     1 link,2                ; AC1 = p.link (added...ddr)
        jmp     cleanLater

delink: movz    0 3 snr         ; if q1=0
          movo  2 3 skp         ; then p; carry=1 iff q1=0
search: lda     3 0,3           ; else q1↑; assumes link = 0
        ; pp is in 3
        lda     1 link,3
        seq     1 2
          jmp   search          ; assumes link = 0

        ; now pp.link = p
        lda     1 link,2
        sta     1 link,3        ; pp.link ← p.link

        mov#    0 0 szc         ; test q1=0 (carry set above)
          jmp   cleanLater      ; cleanup later
cleanNow:
        lda     0 @q1
        sub#    0 2 snr         ; if q1↑=p
cleanStore: sta 3 @q1           ; q1↑ ← pp
        jmp insert
cleanLater:
        sta     1 cleanUpLink,2 ; p.cleanUpLink ← p.link

insert: lda     3 @q2           ; pp ← q2↑
        sz      3 3             ; test for zero
          jmp   RQnonempty
        ; here if queue was empty
        sta     2 link,2        ; p.link ← p
        sta     2 @q2           ; q2↑ ← p;
        jmp     @QSreturn
mPriority: priority
RQnonempty:
        lda     0 mPriority             ; mask
        lda     1 bitsandpriority,2
        and     0 1             ; p.priority
        lda     2 bitsandpriority,3
        and     0 2             ; pp.priority
        sleu    1 2             ; skip if p.priority <= pp.priority
          jmp   iSearch
        ; here if item is new queue head
        lda     2 p
        sta     2 @q2           ; q2↑ ← p
        jmp setLinks

iSearch:
        lda     2 link,3        ; pp.link
        lda     2 bitsandpriority,2
        and     0 2             ; pp.link.priority
        sleu    1 2             ; skip if p.priority <= pp.link.priority
          jmp searchdone
        lda     3 link,3
        jmp     iSearch
searchdone:
        lda     2 p
setLinks:
        lda     1 link,3
        sta     1 link,2        ; p.link ← pp.link
        sta     2 link,3        ; pp.link ← p
        jmp     @QSreturn

END:    jmp     START+200               ; generate error if too big

;
;
        .END
```