```
;-----------------------------------------------------------------
; Mesab.Mu - Registers, miscellaneous symbols and constants
; Last modified by Levin - July 5, 1978  9:08 AM
;-----------------------------------------------------------------


;-----------------------------------------------------------------
; Get standard Alto Definitions
;-----------------------------------------------------------------

#Altoconsts23.mu;
```

```
;-----------------------------------------------------------------
; R memories used by code in ROM0, correct to AltoCode23.Mu
;-----------------------------------------------------------------

$NWW            $R4;
$CBA            $R22;
$AECL           $R23;
$SLC            $R24;
$MTEMP          $R25;
$HTAB           $R26;
$YPOS           $R27;
$DWA            $R30;
$CURX           $R20;
$CURDATA        $R21;
$R37            $R37;
$ECNTR          $R12;   COUNT OF WORDS YET TO BE PROCESSED IN MAIN LOOP
$EPNTR          $R13;   POINTS AT WORD BEFORE THE WORD NEXT TO BE PROCESSED
$CLOCKTEMP      $R11;
$SAD            $R5;
$PC             $R6;    USED BY MEMORY INIT
$AC0            $R3;    AC'S ARE BACKWARDS BECAUSE THE HARDWARE SUPPLIES
;                       COMPLEMENT OF ADDRESS WHEN ADDRESSING FROM IR
$AC1            $R2;
$AC2            $R1;
$AC3            $R0;
$XREG           $R7;
$CYRET          $R5;    Shares space with SAD.
$CYCOUT         $R7;    Shares space with XREG.
$XH             $R10;
$DWAX           $R35;
$MASK           $R36;
$MASK1          $R0;
$YMUL           $R2;    HAS TO BE AN R-REG FOR SHIFTS
$RETN           $R2;
$SKEW           $R3;
$TEMP           $R5;
$WIDTH          $R7;
$PLIER          $R7;    HAS TO BE AN R-REG FOR SHIFTS
$DESTY          $R10;
$WORD2          $R10;
$STARTBITSM1    $R35;
$SWA            $R36;
$DESTX          $R36;
$LREG           $R40;   HAS TO BE R40 (COPY OF L-REG)
$NLINES         $R41;
$RAST1          $R42;
$SRCX           $R43;
$SKMSK          $R43;
$SRCY           $R44;
$RAST2          $R44;
$CONST          $R45;
$TWICE          $R45;
$HCNT           $R46;
$VINC           $R46;
$HINC           $R47;
$NWORDS         $R50;
$MASK2          $R51;   WAS $R46;
$DCBR           $R34;
$KNMAR          $R33;
$CKSUMR         $R32;
$KWDCT          $R31;
$KNMARW         $R33;
$CKSUMRW        $R32;
$KWDCTW         $R31;
$AudioWdCt      $R71;
$AudioData      $R72;
```

```
;-------------------------------------------------------------
; Registers used by Mesa Emulator
;-------------------------------------------------------------

; R registers

$temp           $R35;                   Temporary (smashed by BITBLT)
$temp2          $R36;                   Temporary (smashed by BITBLT)
$mpc            $R15;                   R register holds Mesa PC (points at word last read)
$stkp           $R16;                   stack pointer [0-10] 0 empty, 10 full
$XTSreg         $R17;                   xfer trap state

; Registers shared by Nova and Mesa emulators
;       Nova ACs are set explicitly by Mesa process opcodes and for ROM0 calls
;       Other R-registers smashed by BITBLT and other ROM0 subroutines

$brkbyte        $R0;                    (AC3) bytecode to execute after a breakpoint
;                                       Warning! brkbyte must be reset to 0 after ROM calls!
;                                       (see BITBLT)
$mx             $R1;                    (AC2) x register for XFER
;                                       Warning! smashed by BITBLT and MUL/DIV/LDIV
$saveret        $R2;                    (AC1) R-temporary for return indices and values
$newfield       $R3;                    (AC0) new field bits for WF and friends
;                                       Warning! must be R-register; assumed safe across CYCLE

$count          $R5;                    scratch R register used for counting
$taskhole       $R7;                    pigeonhole for saving things across TASKs
;                                       Warning! smashed by all ROM calls!
$ib             $R10;                   instruction byte, 0 if none (0,,byte)
;                                       Warning! smashed by BITBLT
$clockreg       $R37;                   low-order bits of real-time clock

; S registers, can't shift into them, BUS not zero while storing.

$my             $R51;                   y register for XFER
$lp             $R52;                   local pointer
$gp             $R53;                   global pointer
$cp             $R54;                   code pointer
$ATPreg         $R55;                   allocation trap parameter
$OTPreg         $R56;                   other trap parameter
$XTPreg         $R57;                   xfer trap parameter
$wdc            $R70;                   wakeup disable counter

; Mesa evaluation stack

$stk0           $R60;                   stack (bottom)
$stk1           $R61;                   stack
$stk2           $R62;                   stack
$stk3           $R63;                   stack
$stk4           $R64;                   stack
$stk5           $R65;                   stack
$stk6           $R66;                   stack
$stk7           $R67;                   stack (top)

; Miscellaneous S registers

$mask           $R41;                   used by string instructions, among others
$unused1        $R42;                   not safe across call to BITBLT
$unused2        $R43;                   not safe across call to BITBLT
$alpha          $R44;                   alpha byte (among other things)
$index          $R45;                   frame size index (among other things)
$entry          $R46;                   allocation table entry address (among other things)
$frame          $R47;                   allocated frame pointer (among other things)
$righthalf      $R41;                   right 4 bits of alpha or beta
$lefthalf       $R45;                   left 4 bits of alpha or beta
$unused3        $R50;                   not safe across call to BITBLT
```

```
;-------------------------------------------------------------
; Mnemonic constants for subroutine return indices used by BUS dispatch.
;-------------------------------------------------------------

$ret0           $L0,12000,100;                  zero is always special
$ret1           $1;
$ret2           $2;
$ret3           $3;
$ret4           $4;
$ret5           $5;
$ret6           $6;
$ret7           $7;
$ret10          $10;
$ret11          $11;
$ret12          $12;
$ret13          $13;
$ret14          $14;
$ret15          $15;
$ret16          $16;
$ret17          $17;
$ret20          $20;
$ret21          $21;
$ret22          $22;
$ret23          $23;
$ret24          $24;
$ret25          $25;
$ret26          $26;
$ret27          $27;
$ret30          $30;
$ret31          $31;
$ret37          $37;
```

```
;-----------------------------------------------------------------
; Mesa Trap codes - index into sd vector
;-----------------------------------------------------------------

$sBRK                   $L0,12000,100;          Breakpoint
$sStackUnderflow        $2;                     (trap handler distinguishes underflow from
$sStackOverflow         $2;                     overflow by stkp value)
$sXferTrap              $4;
$sAllocListEmpty        $6;
$sControlFault          $7;
$sCsegSwappedOut        $10;
$sUnbound               $13;


;-----------------------------------------------------------------
; Low-core address definitions
;-----------------------------------------------------------------

$CurrentState   $23;                            location holding address of current state
$NovaDVloc      $25;                            dispatch vector for Nova code
$avm1           $777;                           base of allocation vector for frames (-1)
$sdoffset       $60;                            offset to base of sd from av
$gftm1          $1377;                          base of global frame table (-1)

;-----------------------------------------------------------------
; Constants in ROM, but with unpleasant names
;-----------------------------------------------------------------

$12             $12;                            for function calls
$-12            $177766;                        for Savestate
$400            $400;                           for JB


;-----------------------------------------------------------------
; Frame offsets and other software/microcode agreements
;-----------------------------------------------------------------

$lpoffset       $6;                             local frame overhead + 2
$nlpoffset      $177771;                        = -(lpoffset + 1)
$nlpoffset1     $177770;                        = -(lpoffset + 2)
$pcoffset       $1;                             offset from local frame base to saved pc
$npcoffset      $5;                             = -(lpoffset+1+pcoffset) [see Savpcinframe]
$retlinkoffset  $2;                             offset from local frame base to return link
$nretlinkoffset $177774;                        = -(lpoffset-retlinkoffset)

$gpoffset       $4;                             global frame overhead + 1
$ngpoffset      $177773;                        = -(gpoffset + 1)
$gfioffset      $L0,12000,100;                  offset from global frame base to gfi word (=0)
$ngfioffset     $4;                             = gpoffset-gfioffset [see XferGfz]
$cpoffset       $1;                             offset from global frame base to code pointer

$gfimask        $177600;                        mask to isolate gfi in global frame word 0
$enmask         $37;                            mask to isolate entry number/4

$maxallocslot   $23;                            largest fsi microcode can handle


;-----------------------------------------------------------------
; Symbols to be used instead of ones in the standard definitions
;-----------------------------------------------------------------

$mACSOURCE      $L024016,000000,000000;         sets only F2.  ACSOURCE also sets BS and RSEL
$msr0           $L000000,012000,000100;         IDISP => 0, no IR← dispatch, a 'special' zero
$BUSAND~T       $L000000,054015,000040;         sets ALUF = 15B, doesn't require defined bus
```