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

; Get standard Alto Definitions

#Altoconsts23.mu;
Mesab.mu

Page 2

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

$NW  $R4;
$CB  $R22;
$AE  $R23;
$SL  $R24;
$TE  $R25;
$HT  $R26;
$YP  $R27;
$DA  $R30;
$UR  $R20;
$URD $R21;
$37  $R37;
$CN  $R12;
$PN  $R13;
$CLOC $R11;
$AD  $R5;
$AC  $R6;
$AO  $R3;
$AC  $R2;
$AC  $R1;
$AC  $R0;
$XRE $R7;
$YR  $R5;
$CY  $R7;
$CYC $R7;
$XX  $R10;
$DM  $R35;
$MS  $R36;
$MKS $R0;
$YM  $R2;
$RT  $R2;
$SKW $R3;
$TE  $R5;
$WD  $R7;
$LI  $R7;
$DE  $R10;
$WD2 $R10;
$STB  $R36;
$SWA $R36;
$DE  $R36;
$LR  $R40;
$NL  $R41;
$RA  $R42;
$SRC $R43;
$SK  $R43;
$RC  $R44;
$RA  $R44;
$CON  $R45;
$TW  $R45;
$HCN $R46;
$VI  $R46;
$HI  $R47;
$NW  $R50;
$MA  $R51;
$DC  $R34;
$KN  $R33;
$CS  $R32;
$KD  $R31;
$KN  $R33;
$CS  $R32;
$KD  $R31;
$Au  $R71;
$Au  $R72;

COUNT OF WORDS YET TO BE PROCESSED IN MAIN LOOP
POINTS AT WORD BEFORE THE WORD NEXT TO BE PROCESSED
USED BY MEMORY INIT
AC'S ARE BACKWARDS BECAUSE THE HARDWARE SUPPLIES
COMPLEMENT OF ADDRESS WHEN ADDRESSING FROM IR
Shares space with SAD.
Shares space with XREG.
HAS TO BE AN R-REG FOR SHIFTS
HAS TO BE AN R-REG FOR SHIFTS
HAS TO BE R40 (COPY OF L-REG)
WAS $R46;
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)
$stkpt $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 ROMO calls
; Other R-registers smashed by BITBLT and other ROMO 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
; (AC0) new field bits for WF and friends
; Warning! must be R-register; assumed safe across CYCLE
$count $R3; scratch R register used for counting
$taskhole $R7; pigeonhole for saving things across TASKS
; Warning! smashed by all ROM calls!
$tb $R10; instruction byte, 0 if none (0,byte)
; Warning! smashed by BITBLT
$clockreg $R3; 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)
$lefhalf $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  \$LO,12000,100;  
\$ret1  \$1;            
\$ret2  \$2;            
\$ret3  \$3;            
\$ret4  \$4;            
\$ret5  \$5;            
\$ret6  \$6;            
\$ret7  \$7;            
\$ret8  \$10;           
\$ret9  \$11;           
\$ret10 \$12;           
\$ret11 \$13;           
\$ret12 \$14;           
\$ret13 \$15;           
\$ret14 \$16;           
\$ret15 \$17;           
\$ret16 \$20;           
\$ret17 \$21;           
\$ret18 \$22;           
\$ret19 \$23;           
\$ret20 \$24;           
\$ret21 \$25;           
\$ret22 \$26;           
\$ret23 \$27;           
\$ret24 \$30;           
\$ret25 \$31;           
\$ret26 \$32;           
\$ret27 \$33;           
\$ret28 \$34;           
\$ret29 \$35;           
\$ret30 \$36;           
\$ret31 \$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 $17776; 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)
$gfoffset $L0,12000,100; offset from global frame base to gfi word (=0)
$ngfoffset $4; = gpoffset-gfoffset [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

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