S PROGRAM DEFINES THE MICROINSTRUCTION FORMAT
; AND MNEMONICS FOR THE METHEUS DISPLAY MODULE

; WRITTEN BY BOB BRUCE 2-SEP-81
; MINOR REVISIONS 7-OCT-81 TO INCLUDE NOP, CLR,
; AND TO REMOVE I/O FLAG SET/CLEAR CONTROL BITS

; WRITTEN BY BOB BRUCE 2-SEP-81
; MINOR REVISIONS 7-OCT-81 TO INCLUDE NOP, CLR,
; AND TO REMOVE I/O FLAG SET/CLEAR CONTROL BITS

; ADDITIONAL REVISIONS 3-NOV-81 TO PROVIDE DEFAULT
; OF -1 IN PIPELINE REGISTER CONSTANT FIELD AND OF
; OF CONT IN BRANCH FIELD, CHANGE DEFAULT OF DATA BUS
; SOURCE TO ALUY, ADD SHIFT CONTROL TO IMAGE READBACK
; SHIFT REGISTER AND ALLOW IMAGE READBACK AS HIGH AND
; LOW NIBLES RATHER THAN AS ONE BYTE. ALSO, THE LP2 CONDITION
; IS RENAMED TO LPC FOR CONSISTENCY. SOME OF THESE CHANGES
; ARE IN ANTICIPATION OF MODIFICATIONS TO BE MADE TO THE
; PROTOTYPE BOARD.

; REVISION OF 10-DEC-81 TO FIX ERROR IN NOP DEFINITION
; REVISION OF 11-DEC-81 TO ADD SUPPORT FOR UCTL1 AND 2
; REVISION OF 13-DEC-81 TO FIX PASS CMD
; REVISION OF 25-DEC-81 TO CORRECT ADDRESS OF IMRDH

; DEFINE MICROINSTRUCTION WORD LENGTH
WORD 64

; THE MICROINSTRUCTION WORD IS DIVIDED INTO SIX FIELDS.
; STARTING WITH THE LEAST SIGNIFICANT BIT THEY ARE:
; 1. CONSTANT FIELD FOR DATA OR BRANCH ADDRESS (12 BITS)
; 2. ALU FUNCTION, SOURCE SELECT AND A AND B ADDRESSES (15 BITS)
; 3. ALU DESTINATION AND SHIFT CONTROL (5 BITS)
; 4. MICROPROGRAM CONTROL: BRANCH AND SUBROUTINES (11 BITS)
; 5. DATA BUS SOURCE AND DESTINATION (8 BITS)
; 6. CONTROL BITS FOR DRAWING PROCESSOR AND MISC. (13 BITS)

; THE CONSTANT FIELD IS EITHER EXPLICITLY DEFINED
; WITH THE "CONST" INSTRUCTION OR IMPLICITLY WITH
; THE "DBUS" INSTRUCTION.
2901 SOURCE AND ALU FUNCTIONS

;THE ALU FUNCTION TAKES THREE ARGUMENTS: ALU SOURCE CODE, B AND A INPUTS.

2901 ALU FUNCTIONS

ADD: DEF 37X, H#0, 3VQ#, 4VH#, 4VH#, 12X
ADDI: DEF 37X, H#1, 3VQ#, 4VH#, 4VH#, 12X
SUBRD: DEF 37X, H#2, 3VG#, 4VH#, 4VH#, 12X
SUBR: DEF 37X, H#3, 3VG#, 4VH#, 4VH#, 12X
SUBD: DEF 37X, H#4, 3VQ#, 4VH#, 4VH#, 12X
SUB: DEF 37X, H#5, 3VQ#, 4VH#, 4VH#, 12X
OR: DEF 37X, H#6, 3VQ#, 4VH#, 4VH#, 12X
AND: DEF 37X, H#7, 3VQ#, 4VH#, 4VH#, 12X
NOTR: DEF 37X, H#A, 3VQ#, 4VH#, 4VH#, 12X
EXOR: DEF 37X, H#C, 3VQ#, 4VH#, 4VH#, 12X
EXNOR: DEF 37X, H#E, 3VQ#, 4VH#, 4VH#, 12X

ALU SOURCES

A0: EQU Q#0
AB: EQU Q#1
Z0: EQU Q#2
Z1: EQU Q#3
Z2: EQU Q#4
D0: EQU Q#5
D1: EQU Q#6
D2: EQU Q#7

REGISTER DEFINITIONS

R0: EQU H#0
R1: EQU H#1
R2: EQU H#2
R3: EQU H#3
R4: EQU H#4
R5: EQU H#5
R6: EQU H#6
R7: EQU H#7
R8: EQU H#8
R9: EQU H#9
R10: EQU H#A
R11: EQU H#B
R12: EQU H#C
R13: EQU H#D
R14: EQU H#E
R15: EQU H#F
USEFUL COMBINATIONS. TAKE AS FIRST ARGUMENT A, B OR Q. SECOND AND THIRD ARGUMENTS ARE B AND A ADDRESSES.

THESE INSTRUCTIONS DEFINE USEFUL COMBINATIONS OF ALU SOURCE AND FUNCTION.

PASS: DEF 37X, B#0000, 3V, 4VH#, 4VH#, 12X
NOT: DEF 37X, B#1110, 3V, 4VH#, 4VH#, 12X
INC: DEF 37X, B#0001, 3V, 4VH#, 4VH#, 12X
DEC: DEF 37X, B#0010, 3V, 4VH#, 4VH#, 12X
NEG: DEF 37X, B#0101, 3V, 4VH#, 4VH#, 12X

ARGUMENT DEFINITIONS FOR ABOVE

A: EQU B#100
B: EQU B#011
Q: EQU B#010

D INSTRUCTIONS—NOT COMBINED WITH THE ABOVE BECAUSE OF DIFFERENT BIT PATTERNS
PASSD: DEF 37X, B#0000111, 4VH#, 4VH#, 12X
NOTD: DEF 37X, B#1110111, 4VH#, 4VH#, 12X
INCD: DEF 37X, B#0001111, 4VH#, 4VH#, 12X
DECD: DEF 37X, B#0100111, 4VH#, 4VH#, 12X
NEGD: DEF 37X, B#0011111, 4VH#, 4VH#, 12X

NOP INSTRUCTION: PASS RO AND NOLD TAKES NO ARGUMENTS
NOP: DEF 32X, B#00, Q#1, B#1100100000000000, 12X

CLEAR INSTRUCTION: PUT OUT ZERO ON F BUS TAKES TWO ARGUMENTS: A AND B REGISTER DESIGNATIONS
CLR: DEF 37X, B#1000111, 4VH#, 4VH#, 12X
2901 DESTINATION CONTROL

The destination control functions select the B or Q registers to be loaded with the output of the ALU. This result can be shifted up or down before loading. The shifting mode is selected by the "shift control" argument.

LDQ: DEF 32X,B#00,Q#0,27X
NOLD: DEF 32X,B#00,Q#1,27X
LDB.A: DEF 32X,B#00,Q#2,27X
LDB.F: DEF 32X,B#00,Q#3,27X
LDBQ.DN: DEF 32X,2V,Q#4,27X
LDB.DN: DEF 32X,2V,Q#5,27X
LDBQ.UP: DEF 32X,2V,Q#6,27X
LDB.UP: DEF 32X,2V,Q#7,27X

SHIFT CONTROL

SHIFTO: EQU B#00
SHIFT1: EQU B#01
ROTATE: EQU B#10
ARSHIFT: EQU B#11
; MICROSEQUENCER CONTROL

; FOUR-WAY BRANCH (TO R OR D)
; TAKES AS ARGUMENTS DESTINATION (R OR D) AND TWO CONDITIONS
FWYBR: DEF 21X,B#00,1V,3V,5V,32X

; CONDITIONAL BRANCH (TO R OR D)
; TAKES AS ARGUMENTS DESTINATION (R OR D) AND CONDITION
; IF NO CONDITION IS SPECIFIED THEN BRANCH UNCONDITIONALLY.
BR: DEF 21X,B#01,1V,B#000,5VB#10000,32X

; CONDITIONAL BRANCH TO SUBROUTINE (ADR. IN R OR D)
; TAKES AS ARGUMENTS DESTINATION (R OR D) AND CONDITION
; IF NO CONDITION IS SPECIFIED THEN BRANCH TO SUBROUTINE IS UNCONDITIONAL.
BSR: DEF 21X,2B#10,1V,B#000,5VB#10000,32X

; BRANCH ADDRESS SOURCES: R (R REGISTER) OR D (DATA BUS)
R: EQU B#0
D: EQU B#1

; CONDITIONAL RETURN FROM SUBROUTINE
; TAKES AS AN ARGUMENT A CONDITION
; IF NO CONDITION IS SPECIFIED THEN RETURN IS UNCONDITIONAL
RTS: DEF 21X,B#111,B#000,5VB#10000,32X

; CONTINUE (ACTUALLY BRANCH NEVER)
CONT: DEF 21X,B#010,B#000,B#00000,32X
BRANCH CONDITIONS, GROUP ONE (USED WITH BR, BSR, RTS AND
SECOND CONDITION IN 4-WAY BRANCH)

188 ; MINUS RESULT IN ALU
189 MI: EQU B#00001
190 ;
191 ; PLUS (NONNEGATIVE) RESULT IN ALU
192 PL: EQU B#10001
193 ;
194 ; ARITHMETIC CARRY (CA)
195 CA: EQU B#00010
196 ; NO ARITHMETIC CARRY
197 NCA: EQU B#10010
198 ;
199 ; ARITHMETIC OVERFLOW
200 OV: EQU B#00011
201 ; NO OVERFLOW
202 NOV: EQU B#10011
203 ;
204 ; ZERO RESULT IN ALU
205 Z: EQU B#00100
206 ; NONZERO RESULT IN ALU
207 NZ: EQU B#10100
208 ;
209 ; LOOP COUNTER EQUALS ZERO
210 LPC: EQU B#10101
211 ; LOOP COUNTER NOT EQUAL TO ZERO
212 NLPC: EQU B#00101
213 ;
214 ; RAM RIGHT SHIFT CARRY
215 RRC: EQU B#00110
216 ; NO RAM RIGHT SHIFT CARRY
217 NRRC: EQU B#10110
218 ;
219 ; QREG RIGHT SHIFT CARRY
220 QRC: EQU B#00111
221 ; NO QREG RIGHT CARRY
222 NOQRC: EQU B#10111
223 ; LEFT RAM CARRY
224 LRC: EQU B#01000
225 ; NOT LEFT RAM CARRY
226 NLRC: EQU B#11000
227 ;
228 ; LEFT Q CARRY
229 LQC: EQU B#01001
230 ; NOT LEFT Q CARRY
231 NLQC: EQU B#11001
232 ;
233 ; DISSEL CONTROL LINE
234 DSL: EQU B#01010
235 ; INVERTED DISSEL CONTROL LINE
237  NDSL:  EQU  B#11010
238  ;
239  ;BYTE NEGATIVE FLAG
240  BMI:  EQU  B#01011
241  ;BYTE NONNEGATIVE FLAG
242  BPL:  EQU  B#11011
243  ;
244  ;I/O FLAG TRUE
245  IOF:  EQU  B#01100
246  ;I/O FLAG NOT TRUE
247  NIOF:  EQU  B#11100
248  ;
249  ;VERTICAL BLANK FLAG
250  VBL:  EQU  B#11101
251  ;VERTICAL BLANK NOT
252  NVBL:  EQU  B#01101
253  ;
254  ;OUTPUT DATA ACCEPTED
255  ODAC:  EQU  B#01110
256  ;OUTPUT DATA NOT ACCEPTED
257  NODAC:  EQU  B#11110
258  ;
259  ;INPUT DATA READY
260  IDRDY:  EQU  B#01111
261  ;INPUT DATA NOT READY
262  NIDRDY:  EQU  B#11111
264 ;
265 ;
266 ;BRANCH CONDITIONS, GROUP TWO. (USED ONLY
267 ;AS FIRST CONDITION IN FOUR-WAY BRANCH).
268 ;
269 ;ALTHOUGH THE CONDITIONS ARE THE SAME AS
270 ;SOME OF THE GROUP ONE CONDITIONS,
271 ;THEY ARE GIVEN DIFFERENT NAMES HERE SINCE
272 ;THEY HAVE DIFFERENT BIT PATTERNS.
273 ;
274 ;LOOP COUNTER CARRY TRUE. (NOTE THAT
275 ;THIS FLAG IS LOW TRUE IN HARDWARE
276 ;AND IS INVERTED HERE BY SETTING MOST SIG BIT HIGH
277 ;
278 ;LOOP COUNTER CARRY NOT TRUE
279 ;
280 ;ALU RESULT ZERO AND NONZERO
281 Z: EQU B#000
282 NZ: EQU B#101
283 ;
284 ;ALU ARITHMETIC CARRY AND NOT CARRY
285 CA: EQU B#010
286 NCA: EQU B#110
287 ;
288 ;RAM RIGHT CARRY AND NOT RAM RIGHT CARRY
289 RRC: EQU B#011
290 NRR: EQU B#111
; DATA BUS SOURCE AND DESTINATION.
; OF CODE IS FOLLOWED BY TWO ARGUMENTS: FIRST
; THE DESTINATION, THEN THE SOURCE. IF THE SOURCE IS
; THE PIPELINE REGISTER THEN A THIRD ARGUMENT PROVIDES
; THE PIPELINE REGISTER CONTENTS.
; DEFAULTS ARE NO DESTINATION, ALU Y OUTPUT
; SOURCE AND -1 (ALL 1'S) PIPELINE REGISTER DATA

DBUS: DEF 13X,5VB#00000,3VB#001,31X,12V%H#FFF
; DATA BUS SOURCE OPERANDS
; ALU "Y" OUTPUT
ALUY: EQU B#001

; PIPELINE REG "CONSTANT"
PLREG: EQU B#010

; SCRATCH PAD RAM READ
SPRAM: EQU B#011

; IMAGE MEMORY READBACK LOW NIBBLE
IMRDL: EQU B#100

; IMAGE MEMORY READBACK HIGH NIBBLE
IMRDH: EQU B#111

; CRC TESTER DATA READ
CRCRD: EQU B#101

; INPUT DATA REGISTER LOW AND HIGH BYTE READ
INL: EQU B#110

; INH: EQU B#000
; DATA BUS DESTINATION OPERANDS

; NO DESTINATION
336  NODEST:  EQU  B#00000
337
; LOOP COUNTER LOAD
339  LPCNTLD:  EQU  B#00001
340
; IMAGE MEMORY DATA MASK REGISTER LOAD
342  DMSKLD:  EQU  B#00010
343
; PATTERN CONTROL REGISTER
345  PATRLD:  EQU  B#00011
346
; DRAWING PROCESSOR X REGISTER LOAD
348  PXLD:  EQU  B#00100
349
; DRAWING PROCESSOR Y REGISTER LOAD
350  PYLD:  EQU  B#00101
351
; DISPLAY X START (X PAN)
354  XSTLD:  EQU  B#01100
355
; DISPLAY X ZOOM AND FRACTIONAL PIXEL PAN REGISTER LOAD
357  XFPZLD:  EQU  B#01101
358
; DISPLAY Y START (Y PAN)
361  YSTLD:  EQU  B#01000
362
; DISPLAY Y ZOOM AND FRACTIONAL PIXEL PAN REGISTER LOAD
363  YFPZLD:  EQU  B#01001
364
; OUTPUT DATA REGISTER LOW AND HIGH BYTE LOAD
366  ODLLD:  EQU  B#01010
367
368  ODHLD:  EQU  B#01011
369
; CHARACTER GENERATOR CHAR REG LOAD
371  CGENLD:  EQU  B#01100
372
; CRT CONTROLLER LOAD
374  CRTCLD:  EQU  B#01101
375
; COLOR MAP CONTROL REGISTER LOAD
377  CMAPLD:  EQU  B#01110
378
; CRC TESTER CONTROL REGISTER LOAD
380  CRTCCLD:  EQU  B#01111
381
; SCRATCH PAD RAM ADDRESS LOAD
383  SPADRLD:  EQU  B#10000
384  ; SCRATCH PAD RAM DATA LOAD
385  SPDATLD:     EQU     B#10001
386  7  ;
388  ; USER CONTROL 1 FOR DEBUG
389  UCTL1:     EQU     B#10010
390  8  ;
391  ; USER CONTROL 2 FOR DEBUG
392  UCTL2:     EQU     B#10011
393  ;
395 ;
396 ; CONSTANT
397 ; A SPECIAL CASE: CONSTANT FROM PIPELINE REGISTER
398 ; TAKES ARGUMENT WHICH IS A LABEL OR DECIMAL CONSTANT.
399 CONST: DEF 13X,5B#00000,3B#010,31X,12V%H#
401 ;
402 ; CONTROL FUNCTIONS
403 ; EACH CONTROL BIT IS HANDLED BY A SEPARATE
404 ; INSTRUCTION FORMAT TO ALLOW THEM TO BE SPECIFIED IN ANY
405 ; ORDER. IN ADDITION, A CONTROL DEFAULTS INSTRUCTION ALLOWS ALL THE
406 ; CORRECT DEFAULTS FOR THOSE CONTROL BITS NOT SPECIFIED TO BE USED.
407 ; THE DEFINITIONS ARE SET UP WITH DEFAULTED VARIABLE FIELDS TO
408 ; ALLOW OVERRIDE. OVERRIDE IS NOT PERMITTED WITH CONSTANT FIELDS.
409 ;
410 ; CONTROL DEFAULTS
411 DEFC: DEF 1VB#0,1VB#0,1VB#1,1VB#0,1VB#1,1VB#0,2VB#10,2VB#10,
412 /1VB#1,1VB#0,1VB#1,51X
413 ;
414 ; DRAWING PROCESSOR IMAGE MEMORY WRITEL
415 MEMWR: DEF 11X,2VB#00,51X
416 ;
417 ; DRAWING PROCESSOR BLOCK WRITE
418 WALL: DEF 11X,2VB#10,51X
419 ;
420 ; LOOP' COUNTER COUNT ENABLE
421 LPCNTEN: DEF 10X,1VB#0,53X
422 ;
423 ; DRAWING PROCESSOR X UP/DOWN COUNT CONTROL
424 XUP: DEF 8X,2VB#01,54X
425 XDWN: DEF 8X,2VB#00,54X;
426 ;
427 ; DRAWING PROCESSOR: Y UP/DOWN COUNT CONTROL
428 YUP: DEF 6X,2VB#01,56X
429 YDWN: DEF 6X,2VB#00,56X
430 ;
431 ; CHARACTER GENERATOR SHIFT ENABLE
432 CGENSHFT: DEF 5X,1VB#1,58X
433 ;
434 ; CHARACTER GENERATOR COUNT ENABLE
435 CGENCNT: DEF 4X,1VB#0,59X
436 ;
437 ; IMAGE MEMORY READBACK LOAD
438 RDBCKLD: DEF 3X,1VB#1,1X,1VB#1,53X
439 ;
440 ; IMAGE MEMORY READBACK SHIFT RIGHT
441 RDBCKSR: DEF 3X,1VB#1,1X,1VB#0,53X
442 ;
443 ; IMAGE MEMORY READBACK SHIFT LEFT
444 RDBCKSL: DEF 3X,1VB#0,1X,1VB#1,58X
445 ;
446 ; LOAD R REGISTER
447 LDR: DEF 2X,1VB#0,61X
448 ;
449 ; BREAKPOINT SET
450 BPT: DEF 1VB#1,63X
451 ;
452 ;
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00004</td>
<td>AB</td>
</tr>
<tr>
<td>B</td>
<td>00001</td>
<td>AND</td>
</tr>
<tr>
<td>BR</td>
<td>D</td>
<td>BSR</td>
</tr>
<tr>
<td>CGENCT</td>
<td>D</td>
<td>CGENLDER</td>
</tr>
<tr>
<td>CMAPLD</td>
<td>A</td>
<td>CONST</td>
</tr>
<tr>
<td>CRCTLD</td>
<td>A</td>
<td>CRCTLD</td>
</tr>
<tr>
<td>DBUS</td>
<td>D</td>
<td>DEC</td>
</tr>
<tr>
<td>DMSKLD</td>
<td>A</td>
<td>DQ</td>
</tr>
<tr>
<td>EXNOR</td>
<td>D</td>
<td>EXOR</td>
</tr>
<tr>
<td>IMRDH</td>
<td>A</td>
<td>IMRD</td>
</tr>
<tr>
<td>INH</td>
<td>A</td>
<td>INL</td>
</tr>
<tr>
<td>LDBG.UP</td>
<td>D</td>
<td>LDB.A</td>
</tr>
<tr>
<td>LPC2</td>
<td>A</td>
<td>LPCNTE</td>
</tr>
<tr>
<td>LRD</td>
<td>A</td>
<td>MEMWR</td>
</tr>
<tr>
<td>NCA</td>
<td>A</td>
<td>NCA2</td>
</tr>
<tr>
<td>NEG</td>
<td>D</td>
<td>NIDRDY</td>
</tr>
<tr>
<td>NLPC2</td>
<td>A</td>
<td>NLPC</td>
</tr>
<tr>
<td>NODEST</td>
<td>A</td>
<td>NOLD</td>
</tr>
<tr>
<td>NOTD</td>
<td>D</td>
<td>NOTR</td>
</tr>
<tr>
<td>NRRC</td>
<td>A</td>
<td>NRRC2</td>
</tr>
<tr>
<td>NZ2</td>
<td>A</td>
<td>NDAC</td>
</tr>
<tr>
<td>OR</td>
<td>D</td>
<td>OQ</td>
</tr>
<tr>
<td>PATRLD</td>
<td>A</td>
<td>PL</td>
</tr>
<tr>
<td>R</td>
<td>A</td>
<td>R</td>
</tr>
<tr>
<td>R12</td>
<td>A</td>
<td>R13</td>
</tr>
<tr>
<td>R2</td>
<td>A</td>
<td>R3</td>
</tr>
<tr>
<td>R6</td>
<td>A</td>
<td>R7</td>
</tr>
<tr>
<td>RDBCKLD</td>
<td>D</td>
<td>RDBCKSL</td>
</tr>
<tr>
<td>RRC</td>
<td>A</td>
<td>RRC2</td>
</tr>
<tr>
<td>SHIFT1</td>
<td>A</td>
<td>SPADRLD</td>
</tr>
<tr>
<td>SUB</td>
<td>D</td>
<td>SUBD</td>
</tr>
<tr>
<td>UCTL1</td>
<td>A</td>
<td>UCTL2</td>
</tr>
<tr>
<td>XDWN</td>
<td>D</td>
<td>XFPZLD</td>
</tr>
<tr>
<td>YDWN</td>
<td>D</td>
<td>YFPZLD</td>
</tr>
<tr>
<td>Z</td>
<td>A</td>
<td>Z2</td>
</tr>
<tr>
<td>Z0</td>
<td>A</td>
<td>Z</td>
</tr>
</tbody>
</table>
Graphin
5  sample current position in screen coord.
6  sample current position in tablet coord.
7  modify mode reg (takes 3+4 byte
8  select device

mode reg

\[ \begin{array}{c}
\text{mode reg} \\
\hline \\
2 & 1 & 0
\end{array} \]

- clip, if 1, wrap if 0
- edge bit \( \varnothing = \) level
- ignore button pressed in tracking mode

TAB1D

\[ \begin{array}{c}
\text{t \# bytes returned in a message} \\
\text{Bit pad 1} = 15 \\
\text{GETCO} = 16 \quad \text{(in high res.)} \\
\text{no tablet} = \text{ff}
\end{array} \]

Read config3 2 \& 0

\[ \begin{array}{c}
\varnothing = \text{no intf} \\
1 = \text{tablet interface installed}
\end{array} \]

MISC

43  XOR DRAW (pixel = pixel \& active window)
44  POLY C - close eplotger
45  POLY M - (x, y)
46  CSET WR (x, value)
47  SBCUR (\( \frac{1}{2} x, \frac{1}{2} y \))
48  CSPACE (dw, dh)
49  FONT 52 (w, h)

change zoom and scroll parameter interpreted as follows

\[ \begin{array}{c}
1 \\
\hline
\end{array} \]

If \( x = \varnothing \), \( x_{\text{zoom}} = y_{\text{zoom}} \)
/*
 * the following table is a list of assembler mnemonics;
 * This table is for the Omega instruction set.
 * for each mnemonic the associated machine-code bit pattern
 * and symbol type are given.
 */

struct item  keytab[] = {  
    "align",  0,  ALIGN,  
    "cir",  0x47,  ALU2,  
    "add",  0,  ALU3,  
    "addi",  1,  ALU3,  
    "and",  8,  ALU3,  
    "exnor",  0xe,  ALU3,  
    "exor",  0xc,  ALU3,  
    "notrs",  0xa,  ALU3,  
    "or",  6,  ALU3,  
    "sub",  5,  ALU3,  
    "subd",  4,  ALU3,  
    "subr",  3,  ALU3,  
    "subrd",  2,  ALU3,  
    "dec",  2,  ALU3MAGIC,  
    "inc",  1,  ALU3MAGIC,  
    "neg",  5,  ALU3MAGIC,  
    "not",  0xe,  ALU3MAGIC,  
    "pass",  0,  ALU3MAGIC,  
    "a",  4,  ALUSRC1,  
    "b",  3,  ALUSRC1,  
    "q",  2,  ALUSRC1,  
    "zq",  2,  ALUSRC2,  
    "ab",  1,  ALUSRC2,  
    "aq",  0,  ALUSRC2,  
    "da",  5,  ALUSRC2,  
    "dq",  6,  ALUSRC2,  
    "dz",  7,  ALUSRC2,  
    "za",  4,  ALUSRC2,  
    "zb",  3,  ALUSRC2,  
    "include",  3,  ARGPSEUDO,  
    "rsym",  1,  ARGPSEUDO,  
    "title",  0,  ARGPSEUDO,  
    "wsym",  2,  ARGPSEUDO,  
    "br",  1,  BR,  
    "bsr",  2,  BR,  
    "bmi",  0x0b,  CC1,  
    "bpl",  0x1b,  CC1,  
    "ca",  0x02,  CC1,  
    "db0",  0x09,  CC1,  
    "dsl",  0x0a,  CC1,  
    "idrdy",  0x0f,  CC1,  
    "iof",  0x0c,  CC1,  
    "lpc",  0x5,  CC1,  
    "lrc",  0x08,  CC1,  
    "mi",  0x1,  CC1,  
    "nca",  0x12,  CC1,  
    "ndbo",  0x19,  CC1,  
    "ndsl",  0x1a,  CC1,  
    "nidrdy",  0x1f,  CC1,  
}
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;spram&quot;</td>
<td>0x3</td>
<td>DBSRC</td>
</tr>
<tr>
<td>&quot;dbus&quot;</td>
<td>0</td>
<td>DBUS</td>
</tr>
<tr>
<td>&quot;end&quot;</td>
<td>0</td>
<td>END</td>
</tr>
<tr>
<td>&quot;endif&quot;</td>
<td>0</td>
<td>ENDIF</td>
</tr>
<tr>
<td>&quot;endm&quot;</td>
<td>0</td>
<td>ENDM</td>
</tr>
<tr>
<td>&quot;equ&quot;</td>
<td>0</td>
<td>EQU</td>
</tr>
<tr>
<td>&quot;defl&quot;</td>
<td>0</td>
<td>DEFL</td>
</tr>
<tr>
<td>&quot;fwaybr&quot;</td>
<td>0</td>
<td>FWAYBR</td>
</tr>
<tr>
<td>&quot;bpt&quot;</td>
<td>0x1000</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;cgenccnt&quot;</td>
<td>0x100</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;cgenshft&quot;</td>
<td>0x80</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;crcstop&quot;</td>
<td>0x1000</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;crcstrt&quot;</td>
<td>0x800</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;ldr&quot;</td>
<td>0x400</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;lpcnten&quot;</td>
<td>0x4</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;memwr&quot;</td>
<td>0x1</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;rdbackld&quot;</td>
<td>0x280</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;rdbacksl&quot;</td>
<td>0x80</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;rdbacksr&quot;</td>
<td>0x200</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;wall&quot;</td>
<td>0x3</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;xdwn&quot;</td>
<td>0x10</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;xup&quot;</td>
<td>0x18</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;ydwn&quot;</td>
<td>0x40</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;yup&quot;</td>
<td>0x60</td>
<td>HWBITS</td>
</tr>
<tr>
<td>&quot;if&quot;</td>
<td>0</td>
<td>IF</td>
</tr>
<tr>
<td>&quot;ifd&quot;</td>
<td>0</td>
<td>IFDEF</td>
</tr>
<tr>
<td>&quot;d&quot;</td>
<td>7</td>
<td>IM</td>
</tr>
<tr>
<td>&quot;eject&quot;</td>
<td>1</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;elist&quot;</td>
<td>3</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;flist&quot;</td>
<td>4</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;ilist&quot;</td>
<td>5</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;list&quot;</td>
<td>0</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;mlist&quot;</td>
<td>6</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;nolist&quot;</td>
<td>-1</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;space&quot;</td>
<td>2</td>
<td>LIST</td>
</tr>
<tr>
<td>&quot;ldb.a&quot;</td>
<td>2</td>
<td>LOAD</td>
</tr>
<tr>
<td>&quot;ldb.f&quot;</td>
<td>3</td>
<td>LOAD</td>
</tr>
<tr>
<td>&quot;ldq&quot;</td>
<td>1</td>
<td>LOAD</td>
</tr>
<tr>
<td>&quot;ldoldn&quot;</td>
<td>5</td>
<td>LOADSHIFT</td>
</tr>
<tr>
<td>&quot;ldb.up&quot;</td>
<td>7</td>
<td>LOADSHIFT</td>
</tr>
<tr>
<td>&quot;ldbq.dn&quot;</td>
<td>4</td>
<td>LOADSHIFT</td>
</tr>
<tr>
<td>&quot;ldbq.up&quot;</td>
<td>6</td>
<td>LOADSHIFT</td>
</tr>
<tr>
<td>&quot;macro&quot;</td>
<td>0</td>
<td>MACRO</td>
</tr>
<tr>
<td>&quot;mod&quot;</td>
<td>0</td>
<td>MOD</td>
</tr>
<tr>
<td>&quot;defc&quot;</td>
<td>0</td>
<td>NOP</td>
</tr>
<tr>
<td>&quot;nop&quot;</td>
<td>0</td>
<td>NOP</td>
</tr>
<tr>
<td>&quot;org&quot;</td>
<td>0</td>
<td>ORG</td>
</tr>
<tr>
<td>&quot;pplreg&quot;</td>
<td>0</td>
<td>PLREG</td>
</tr>
<tr>
<td>&quot;r0&quot;</td>
<td>0</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r1&quot;</td>
<td>1</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r10&quot;</td>
<td>10</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r11&quot;</td>
<td>11</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r12&quot;</td>
<td>12</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r13&quot;</td>
<td>13</td>
<td>REG</td>
</tr>
<tr>
<td>&quot;r14&quot;</td>
<td>14</td>
<td>REG</td>
</tr>
</tbody>
</table>
"r15", 15, REG,
"r2", 2, REG,
"r3", 3, REG,
"r4", 4, REG,
"r5", 5, REG,
"r6", 6, REG,
"r7", 7, REG,
"r8", 8, REG,
"r9", 9, REG,
"r", 0, RREG,
"rts", 0, RTS,
"arshift", 3, SHIFT,
"rotate", 2, SHIFT,
"shift0", 0, SHIFT,
"shift1", 1, SHIFT,
NULL, NULL, NULL

);
<table>
<thead>
<tr>
<th>String</th>
<th>Address</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bpt</code></td>
<td>0x1000</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>cgecnt</code></td>
<td>0x100</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>cgensht</code></td>
<td>0x80</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>crcstop</code></td>
<td>0x1000</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>crcstrt</code></td>
<td>0x800</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>ldr</code></td>
<td>0x400</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>lpcnten</code></td>
<td>0x4</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>memwr</code></td>
<td>0x1</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>rdbckld</code></td>
<td>0x280</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>rdbcksl</code></td>
<td>0x80</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>rdbcksr</code></td>
<td>0x200</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>wall</code></td>
<td>0x3</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>xdwn</code></td>
<td>0x10</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>xup</code></td>
<td>0x18</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>ydwn</code></td>
<td>0x40</td>
<td>HWBITS</td>
</tr>
<tr>
<td><code>yup</code></td>
<td>0x60</td>
<td>HWBITS</td>
</tr>
</tbody>
</table>
ALU - 1 source

OPCODES:
DEC
INC
NEG
NOT
PASS

FORMAT:
OP src-CNTL, regA, regB
OP src-CNTL, regAB

src-CNTL:
A
B
D
Q

EXAMPLES:
PASS D, R3
NOT A, R2, R3
ALU - 2 source

O P E R A T I O N S:

- ADD
- ADDI
- AND
- EXNOR
- EXOR
- OR
- SUB
- SUBL
- SUBR
- SUBRS
- NOT RS

- add
- add and increment
- and
- exclusive nor
- exclusive or
- or
- sub
- sub and decrement
- sub reverse
- sub reverse and decrement
- not R and I

F O R M A T:

- OP SRC_CNTL, REG A, REG B
- OP SRC_CNTL, REG A, REG B

S R C _ C N T L:

- a
- b
- c
- d
- e
- f

E X A M P L E S:

- ADD a, b, R0, R2
- AND a, b, R5
ALU - MISC

OPCODES
CLR
NOP

FORMAT
op reg, reg
or
op reg
or
op

EXAMPLES
CLR R5
CLR 3, R7
LOAD FIELD CONTROLS THE LOADING OF THE ALU REGS

LOAD CODES:

LDB. A       LOAD SPECIFIED B reg; A reg data → output bus
LDB. F       " " " " alu output → " " " "
LDQ          Q reg
NOLD         NO LOAD
LDBQ

LOAD AND SHIFT:

LDB. DN     SHIFT-CNTL
LDB. UF     ""
LDQ. DN     ""
LDQ. UF     ""

SHIFT-CNTL:

ARSHIFT     ARITHMETIC SHIFT
ROTATE      ""
SHIFTØ      ""
SHIFT1      ""

EXAMPLES

LDB. F
LDB. DN    ROTATE
LDQ. UP    SHIFTØ
Program Counter Control

Opcodes:

FWYBR SRC, CCl, CCl, four way branch
BR SRC [CC2] branch
BSR SRC [CC2] branch subroutine
CONT SRC [CC2] no branch, goto next instr.
RTS [CC2] return from subr.

SRC:
R     R reg has jump addr.
D     Data Bus

Examples:

BR D; NZ
BSR R
FWYBR R, 22, NC
RTS

CC1:

BPL          QRC
CA            RRC
DBT          VBL
DSL          Z

IDDY
IDF

CC2:

LPC          CA2
NCA          LPC2
NCA2          NCA2
NLPC         NLRCC2
NLRCC2        NLRCC2
NVLAC        NZ2
NZ2           NZ2

QV

OPLAC
QV
HARDWARE CONTROL FIELD

These bits control various hardware functions of the drawing processor, such as pad roam, etc.

OPCOCES

DEFC  set default values
MEMWR  \ pixel memory write cmd
WALL   \ loop counter enable
LPCNTEN \ pixel address controls
XUP    \ CHAR GEN
YDOWN  \ READ BACK
YUP    \ LOAD R REG
C6GENSHIFT
C6GNCNT
RDBCKLD
RDBCKSR
RDBCKSL
LDR
CRCSTDPA
CRCSTRT

EXAMPLES

DEFC
WALL LPCNTEN XUP YDOWN
DATA BUS CNTL FIELD — PIPELINE REGISTER FIELD

Controls the data bus source and destination.

OPCODES:

DBUS
DBUS dest, src
DBUS dest, PLREG, expression

CONST expression

SRC:
ALU Y
SPRAM
IMRD1
IMRDH
EXTSRC
INL
INH

DEST:
NODEST
LCENTLD
WDATLD
PATRLD
PXLD
PYLD
XSTLD
YSTLD
ZOMLD
*
CRTCRD
CRTCLD
CGSLD
OQBLLD
OBHLID
CMAPLD
LEDLD
SPARLD
SPDATLD
WMSKLID
*
CRCRD
SPWRUP
RMSKLID
UARTLD
*
UARTRD

* These are actually data sources, use EXT SRC with them

EXAMPLES:

DBUS XSTLD, ALU Y
DBUS SPARLD, PLREG, ØFFH
CONST 3
DBUS UARTRD, EXT SRC
Micro Code

<table>
<thead>
<tr>
<th>Control</th>
<th>Data Bus Sequences</th>
<th>ALU</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>5 3 3 3 5</td>
<td>2 3</td>
</tr>
</tbody>
</table>

Data field
- can go onto data bus
- or< ALU to I/O device
- used to branch address for micro sequence.

HW Control field
- Write refresh ram with content of buffer to x/y counter address
- Wall write all (flash fill)
- Loop cnt+enL enable loop counter hw
- PX UD refresh ram x counter w/b
- PX CNT " " " count enable
- PY UD " " y w/d
- PY CNT " " " count enable
- CGEN SHIFT Chan Gen shift & row cnt Enables
- CGEN CNTENL
- RD BCK LD 2 bits, oe is shaped with f. Rd back shift control.
- RD BCK LD RREGL micro sequencer & reg all enables.
- BPT SPARE Single step & bpt control
Assemblers Format

5 Fields
1. Control bits
2. ALU src + op
3. Branch, sequence
4. ALU dest.
5. Data bus

note this is different from schematics

<table>
<thead>
<tr>
<th>ALU SRC + OP</th>
<th>ALU DEST</th>
<th>Sequence</th>
<th>Data Bus Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFC &amp; EXOR</td>
<td>AB, R6, R7</td>
<td>LDB, F &amp; CONT</td>
<td>DBUS, PYLD, ALUV</td>
</tr>
</tbody>
</table>
Sequencer Control

3 bits function
5 " cc1 \{ typical condition codes, carry, zero, etc. \}
3 " cc2 four way branch condition codes

BRANCH

D BR D, cc1
E BR R, cc1
BR D

CONT
BSR D, cc1
BSR R, cc1
RTS cc1
FWY BR R, cc1, cc2, cc3

Data Bus Branch
R reg Branch
unconditional Branch if no cc1
actually branch never.
subroutine branches, unconditionally if cc1 missing

Can't do a branch based on conditions of an immediate value or
any data bus source. CC just can't make it through logic in time.

Four Way Branch

![Four Way Branch Diagram]
DATA Bus

CONST tells the arm that a constant is to be put in the Data field and also gates the immediate data field onto the data bus

CONST LABEL puts dest address in bus
CONST H H EA put hex value in bus

DBUS DST, SRC ld dest with src.

DBUS PXLD, ALUX ld PX counter with ALU output.

DBUS PXLD, PCREG, #H #10 immediate ld to PXLD & pipe line reg. immediate data (which is the pipe line reg)