```
;-----------------------------------------------------------------------
; FILE  D.text
;
;       Disassembler for the Motorola 68000
;
;
;       Written by Rich Page,  June 8, 1980
;
;
;       Modification History
;
;
;       15-Sep-84       Lookup calls LookupPC
;       27-Sep-84       Line A instructions translate into trap names
;       18-Oct-84       EA *+$HHHH prints absolute address, mask high PC byte
;
;-----------------------------------------------------------------------
;
;
;
;
;
;
;       XJPW    table,index,areg,dreg
;
;
        .MACRO  XJPW
        LEA     %1,%3
        ADD.W   %2,%3
        MOVE.W  0(%3,%2),%4
        JMP     0(A6,%4)
        .ENDM
;
;       XJPB    table,index,areg,dreg,label
;
;
        .MACRO  XJPB
        LEA     %1,%3
        CLR.W   %4
        MOVE.B  0(%3,%2),%4
        LEA     %5,%3
        JMP     0(%3,%4)
        .ENDM
;
;       ILLEGAL
;
;
        .MACRO  ILLEGAL
        BSET    #16,D7
        .ENDM
;
;       NXTWORD
;
;
        .MACRO  NXTWORD
        MOVE.W  (A5)+,-(A7)
        .ENDM
;
;       APPCHAR char
;
;
        .MACRO  APPCHAR
        MOVEQ   #%1,D0
        BSR     APPCH
        .ENDM
;
;       USE8IFO dreg
;
;
        .MACRO  USE8IFO
        MOVE.W  %1,D0
        TST.W   D0
        BNE.S   @1
        ADDQ    #8,D0
@1
        .ENDM
;
;       B6TO7   dreg
;
        .MACRO  B6TO7
;        MOVE.W  D4,%1
;        AND.W   #3,%1
        MOVEQ   #3,%1
        AND     D4,%1
        .ENDM
;
;       DATAREG dreg
;
        .MACRO  DATAREG
        MOVE.W  %1,-(A7)
        BSR     DR
        .ENDM
;
;       ADDRREG areg
;
        .MACRO  ADDRREG
        MOVE.W  %1,-(A7)
        BSR     AR
        .ENDM
;
;       MAKEIMM size
;
        .MACRO  MAKEIMM
        MOVE.W  %1,-(A7)
        BSR     MI
        .ENDM
;
;       SHORTIM value
;
        .MACRO  SHORTIM
        MOVE.W  %1,-(A7)
        BSR     SI
        .ENDM
;
;       EA      mode,reg,long
;
        .MACRO  EA
        MOVE.W  %1,-(A7)
        MOVE.W  %2,-(A7)
        .IF     '%3'='LONG'
        MOVEQ   #3,D0
        AND     D4,D0
        CMP.W   #2,D0
        SEQ     -(A7)
        .ENDC
        .IF     '%3'='#FALSE'
        SF      -(A7)
        .ENDC
        .IF     '%3'='#TRUE'
        ST      -(A7)
        .ENDC
        BSR     EFFADDR
        .ENDM
;
;       APPNDSZ size
```

```
;
        .MACRO  APPNDSZ
        MOVE.W  %1,-(A7)
        BSR     APPSIZE
        .ENDM
;
;       APPND   slen,string
;
        .MACRO  APPND
        MOVE.W  #%1,-(A7)
        BSR     APP
        .IF     %1&1 = 0
        .ASCII  %2
        .ELSE
        .ASCII  %2
        .ASCII  ' '
        .ENDC
        .ENDM
;
;       OPCODE  slen,string
;
        .MACRO  OPCODE
        .IF     %1 = 2
        BSR     OPC2
        .ASCII  %2
        .ENDC

        .IF     %1 = 3
        BSR     OPC3
        .ASCII  %2
        .ASCII  ' '
        .ENDC

        .IF     %1 = 4
        BSR     OPC4
        .ASCII  %2
        .ENDC

        .IF     %1 = 5
        BSR     OPC5
        .ASCII  %2
        .ASCII  ' '
        .ENDC

        .IF     %1 = 6
        BSR     OPC6
        .ASCII  %2
        .ENDC

        .IF     %1 = 7
        BSR     OPC7
        .ASCII  %2
        .ASCII  ' '
        .ENDC

;        MOVE.W  #%1,-(A7)
;        BSR     OPC
;        .IF     %1&1 = 0
;        .ASCII  %2
;        .ELSE
;        .ASCII  %2
;        .ASCII  ' '
;        .ENDC
        .ENDM
;
; start here and follow the yellow brick road (if ever a wizard of Oz there was)
;
;
;       DISASM -- disassemble one line of code
;
;       Enter with:
;               A5 as pointer to the code to disassemble
;               A4 as pointer to string for opcode      (max len is 8)
;               A3 as pointer to string for operand     (max len is 64)
;
;       Registers:
;               D7 is first word of opcode
;               D6 is bits0to2
;               D5 is bits3to5
;               D4 is bits6to8
;               D3 is bits9to11
;

DISASM  MOVE.L  A5,D0
        AND.L   MaskBC,D0       ; mask off
        MOVE.L  D0,A5           ; and restore new PC
        MOVE.L  A5,-(SP)        ; save PC on stack
        CLR.W   TEMP            ; clear out special op-code length value
        BSR     TRY2DIS         ; return with D7 upper as validity flag
        SWAP    D7
        TST.W   D7              ; was diassembly valid ?
        BEQ.S   DISASMX
        MOVE.L  (SP)+,A5        ; no, get old PC address
        ADDQ    #2,A5           ; bump PC one word
        CLR.B   (A4)            ; reset opcode and operand strings
        CLR.B   (A3)
        OPCODE  4,'$$$$'        ; push 'bad disassembly' sign
        RTS

DISASMX TST.W   TEMP            ; and special op-code length set?
        BEQ.S   @0              ; nope
        MOVE.W  TEMP,(A3)       ; stuff long length

@0      TST.L   (A7)+           ; yes, return
        RTS

;
;       LOOKUP -- lookup address in symbol table, stack has output string ptr
;                 and address to lookup.
;
;

LOOKUP
        MOVE.L  (A7)+,A2        ; return address
        MOVE.L  (A7)+,A1        ; where to print symbol
        MOVE.L  (A7),A0         ; location of PC
        .IF     FullSized
        MOVE.L  A2,-(SP)        ; push return address
        BSR     LookupPC        ; print out what proc the user is in
        MOVE.L  (SP)+,A2        ; restore return address
        .ENDC
        .IF     0
        LEA     STRSYM,A0   ; symbol table base
        MOVE.L  (A0),A0
        LEA     ENDSYM,A1   ; symbol table limit
        MOVE.L  (A1),A1
@1      CMP.L   A0,A1
        BEQ.S   @4
        CMP.L   8(A0),D0
        BEQ.S   @2
        ADD     #$C,A0
        BRA.S   @1
@2      MOVE.L  #8,D0
        MOVE.L  D1,A1
@3      MOVE.B  (A0)+,(A1)+
        SUBQ    #1,D0
        BNE.S   @3
@4
        .ENDC
        JMP     (A2)
;
;       APPCH -- appends a character to the operand
;

APPCH   MOVE.B  D0,-(A7)                ; save char
        MOVE.L  A3,A0                   ; get operand ptr
        MOVE.B  (A0)+,D0                ; get operand length
        MOVE.B  (A7)+,0(A0,D0)          ; move char to one past end
        ADDQ.B  #1,(A3)                 ; bump position
        RTS

;
;       COMMA -- appends a comma to the operand
;
;       uses registers: A0 and D0
;
COMMA   APPCHAR 44
        RTS
;
;       TABDEST -- tabs to column 15 of operand and appends "; "
;
;       uses registers: A0 and D0
;
TABDEST APPCHAR 32
        MOVE.B  (A3),D0
        CMP.B   #15,D0
        BLT.S   TABDEST
        APPCHAR 59
        APPCHAR 32
        RTS
;
;       DR -- appends a data register to the operand
;
;       uses registers: A0, A1, D0 and D1
;
DR      MOVE.L  (A7)+,A1
        APPCHAR 68
DIGIT   CLR.W   D0
        MOVE.L  A3,A0
        MOVE.B  (A0)+,D0
        MOVE.W  (A7)+,D1
        AND.W   #7,D1
        ADD.W   #48,D1
        MOVE.B  D1,0(A0,D0)
        ADDQ.B  #1,(A3)
        JMP     (A1)
;
```

```
;       AR -- appends an address register to the operand
;
;       uses registers: A0, A1, D0 and D1
;
AR      MOVE.L  (A7)+,A1
        APPCHAR 65
        BRA.S   DIGIT
;
;       REGLIST -- make register list
;
;       uses registers: A0, A1, A2, D0, D1 and D2
;
REGLIST
        NXTWORD
reg2list
        CMP.W   #4,D5
        BNE.S   PREDECR
        MOVE.W  (A7)+,D0
        MOVE.W  #$10,D2
@1      LSR.W   #1,D1
        BTST    #15,D0
        BEQ.S   @2
        BSET    #15,D1
@2      LSL.W   #1,D0
        SUBQ    #1,D2
        BNE.S   @1
        MOVE.W  D1,-(A7)
PREDECR CLR.W   D0
DLOOP   MOVE.W  D0,-(A7)
        BSR     MAKEMSK
        MOVE.L  (A7),D0
        AND.W   D1,D0
        BEQ.S   @1
        MOVE.W  (A7),-(A7)
        BSR     DR
        APPCHAR 47
@1      MOVE.W  (A7)+,D0
        ADDQ    #1,D0
        CMP.W   #$8,D0
        BNE.S   DLOOP

ALOOPS  MOVE.W  D0,-(A7)
        BSR     MAKEMSK
        MOVE.L  (A7),D0
        AND.W   D1,D0
        BEQ.S   @1
        MOVE.W  (A7),-(A7)
        BSR     AR
        APPCHAR 47
@1      MOVE.W  (A7)+,D0
        ADDQ    #1,D0
        CMP.W   #$10,D0
        BNE.S   ALOOPS
        SUBQ.B   #1,(A3)
        TST.W   (A7)+
        RTS
;
MAKEMSK MOVE.L  (A7)+,A2
        MOVE.W  (A7),D0
        MOVE.W  #1,D1
@1      TST.W   D0
        BEQ.S   MSKEXIT
        SUBQ    #1,D0
        LSL.W   #1,D1
        BRA.S   @1
MSKEXIT JMP     (A2)
;
;       APP -- append string to operand
;
;       uses registers: A0, A1, D0 and D1
;
APP     MOVE.L  (A7)+,A0
        MOVE.W  (A7),D0
        ADDQ    #1,D0
        LSR.W   #1,D0
        MOVE.L  A3,A1
        TST.B   (A1)+
        CLR.W   D1
        MOVE.B  (A3),D1
        ADD.W   D1,A1
        BSR.S   APPLOOP
        MOVE.W  (A7)+,D0
        ADD.B   D0,(A3)
        JMP     (A0)
;
;       APPLOOP - actual append loop shared by APP and OPC
;
APPLOOP MOVE.W  (A0)+,D1
        ROR.W   #8,D1
        MOVE.B  D1,(A1)+
        ROR.W   #8,D1
        MOVE.B  D1,(A1)+
        SUBQ    #1,D0
        BNE.S   APPLOOP
        RTS
;
;       OPC -- append string to opcode
;
;       uses registers: A0, A1, D0 and D1
;
opc2
        MOVEQ   #2,D0
        BRA.S   OPC
opc3
        MOVEQ   #3,D0
        BRA.S   OPC
opc4
        MOVEQ   #4,D0
        BRA.S   OPC
opc5
        MOVEQ   #5,D0
        BRA.S   OPC
opc6
        MOVEQ   #6,D0
        BRA.S   OPC
opc7
        MOVEQ   #7,D0

OPC     MOVE.L  (A7)+,A0
;        MOVE.W  (A7),D0
        MOVE    D0,-(SP)
        ADDQ    #1,D0
        LSR.W   #1,D0
        MOVE.L  A4,A1
        TST.B   (A1)+
        CLR.W   D1
        MOVE.B  (A4),D1
        ADD.W   D1,A1
        BSR.S   APPLOOP
        MOVE.W  (A7)+,D0
        ADD.B   D0,(A4)
        JMP     (A0)
;
;       APPSIZE -- append size to opcode
;
;       uses registers: A0, A1, A2, D0 and D1
;
APPSIZE MOVE.L  (A7)+,A2
        MOVE.W  (A7)+,D0
        AND.W   #3,D0
        BEQ.S   @1
        CMP.W   #2,D0
        BEQ.S   @2
        OPCODE  2,'.W'
        JMP     (A2)
@1      OPCODE  2,'.B'
        JMP     (A2)
@2      OPCODE  2,'.L'
        JMP     (A2)
;
;       HEX1 -- append a single hex digit to operand
;
;       uses registers: A0, D0 and D1
;
HEXCH   .ASCII  '0123456789ABCDEF'
HEX1    AND.W   #$F,D0
        LEA     HEXCH,A0
        MOVE.B  0(A0,D0),D1
        CLR.W   D0
        MOVE.L  A3,A0
        MOVE.B  (A0)+,D0
        MOVE.B  D1,0(A0,D0)
        ADDQ.B  #1,(A3)
        RTS
;
;       HEX2 -- append two hex digits to operand
;
;       uses registers: A0, A2, D0 and D1
;
HEX2    MOVE.L  (A7)+,A1
        MOVE.W  (A7),D0
        LSR.W   #4,D0
        BSR     HEX1
        MOVE.W  (A7)+,D0
        BSR     HEX1
        JMP     (A1)
;
;       HEX4 -- append four hex digits to operand
;       Arguments (SP.W)+ : hex value to print out
;       uses registers: A0, A1, A2, D0 and D1
;

HEX4    MOVE.L  (A7)+,A2
        MOVE.W  (A7),D0
        LSR.W   #8,D0
        MOVE.W  D0,-(A7)
        BSR     HEX2
        BSR     HEX2
        JMP     (A2)
;
;       MI -- makeimmediate appends #$data to operand
;
;       uses registers: A0, A1, A2, D0 and D1
;
MI      MOVE.L  (A7)+,A2
        APPCHAR 35
        APPCHAR 36
        MOVE.W  (A7)+,D0
        MOVE.L  A2,-(A7)
        AND.W   #3,D0
        BEQ.S   @1
        CMP.W   #2,D0
        BEQ.S   @2
        NXTWORD
        BSR     HEX4
        RTS
@1      NXTWORD
        BSR     HEX2
        RTS
@2      NXTWORD
        BSR     HEX4
        NXTWORD
        BSR     HEX4
        RTS
;
;       SI -- shortimmediate appends #$data to operand
;
;       uses registers: A0, A1, A2, D0 and D1
;
SI      MOVE.L  (A7)+,A2
        APPCHAR 35
        APPCHAR 36
        MOVE.W  (A7)+,D0
        MOVE.L  A2,-(A7)
        BRA     HEX1
;
;       EFFADDR -- generates an effective address
;
EFFADDR MOVE.L  (A7)+,A0
;        MOVE.W  (A7)+,D2
        MOVEQ   #0,D2
        MOVE.B  (A7)+,D2
        MOVE.W  (A7)+,D1
        MOVE.W  (A7)+,D0
        MOVE.L  A0,-(A7)
        MOVEM.W D3/D4/D5,-(A7)
        MOVE.W  D0,D3
        MOVE.W  D1,D4
        MOVE.W  D2,D5
        XJPW    EAMAIN,D3,A0,D1
EAXIT   MOVEM.W (A7)+,D3/D4/D5
        RTS
;
MAKEREG MOVE.W  4(A7),D0
        AND.W   #$F,4(A7)
        LSL.B   #4,D0
        BMI     AR
        BRA     DR
```

```
;
LPARIRP APPCHAR 40
        MOVE.W  D4,-(A7)
        ADDQ    #8,(A7)
        BSR     MAKEREG
        BSR     COMMA
        MOVE.W  (A7)+,D0
        LSR.W   #8,D0
        LSR.W   #4,D0
        MOVE.W  D0,-(A7)
        BSR     MAKEREG
        APPCHAR 41
        BRA     EAXIT
;
LPANRP  APPCHAR 40
        MOVE.W  D4,-(A7)
        ADDQ    #8,(A7)
        BSR     MAKEREG
        APPCHAR 41
        RTS
;
EACAS0  DATAREG D4
        BRA     EAXIT
EACAS1  ADDRREG D4
        BRA     EAXIT
EACAS2  BSR     LPANRP
        BRA     EAXIT
EACAS3  BSR     LPANRP
        APPCHAR 43
        BRA     EAXIT
EACAS4  APPCHAR 45
        BSR     LPANRP
        BRA     EAXIT
EACAS5  APPCHAR 36
        NXTWORD
        BSR     HEX4
        BSR     LPANRP
        BRA     EAXIT
EACAS6  APPCHAR 36
        NXTWORD
        MOVE.W  (A7),D0
        AND.W   #$FF,D0
        MOVE.W  D0,-(A7)
        BSR     HEX2
        BRA     LPARIRP

EACAS7  XJPW    EAC7,D4,A0,D1

; $HHHH

EAC7S0  APPCHAR 36
        NXTWORD
        BSR     HEX4
        BRA     EAXIT

; $HHHHHHHH

EAC7S1  APPCHAR 36
        NXTWORD
        BSR     HEX4
        NXTWORD
        BSR     HEX4
        BRA     EAXIT

; *+$HHHH

EAC7S2  APPND   3,'*+$'
        NXTWORD
        MOVE.W  (SP),-(SP)      ; save off offset (word)
        BSR     HEX4            ; print the hex value
        MOVE.W  (A3),TEMP       ; save the length of operand
        BSR     TABDEST         ; tab out
        MOVE.L  A5,A0           ; get location we're disassembling
        SUBQ.L  #2,A0           ; adjust PC back 2
        ADD.W   (SP)+,A0        ; A0 = location we're offsetting to
        MOVE.L  A0,-(SP)        ; push location on stack
        BSR     HEX4            ; print it out
        BSR     HEX4            ;

        MOVE.W  (A3),D0         ; stash length of operand
        MOVE.W  TEMP,(A3)       ; restore old length
        MOVE.W  D0,TEMP         ; save true length
        BRA     EAXIT           ; and exit


; *+$HH(AX)

EAC7S3  APPCHAR 42
        APPCHAR 43
        APPCHAR 36
        NXTWORD
        MOVE.W  (A7),D0
        AND.W   #$FF,D0
        MOVE.W  D0,-(A7)
        BSR     HEX2
        APPCHAR 40
        MOVE.W  (A7)+,D0
        LSR.W   #8,D0
        LSR.W   #4,D0
        MOVE.W  D0,-(A7)
        BSR     MAKEREG
        APPCHAR 41
        BRA     EAXIT
EAC7S4  TST.W   D5
        BEQ.S   @1
        MAKEIMM #2
        BRA     EAXIT
@1      MAKEIMM #1
        BRA     EAXIT
EAC7S5
EAC7S6
EAC7S7  ILLEGAL
        BRA     EAXIT
;
;       TRY2DIS -- try to disassemble
;
TRY2DIS CLR.B   (A4)            ; initialize opcode string
        CLR.B   (A3)            ; initialize operand string
        CLR.L   D7              ; validity flag false
        MOVE.W  (A5)+,D7        ; get first opcode word
        MOVE.W  D7,D6
        MOVE.W  D6,D5
        LSR.W   #3,D5
        MOVE.W  D5,D4
        LSR.W   #3,D4
        MOVE.W  D4,D3
        LSR.W   #3,D3
        MOVE.W  D3,D0
        LSR.W   #3,D0
        MOVE.L  #7,D2           ; field mask after shifts
        AND.W   D2,D6           ; D6 is bits0to2
        AND.W   D2,D5           ; D5 is bits3to5
        AND.W   D2,D4           ; D4 is bits6to8
        AND.W   D2,D3           ; D3 is bits9to11
        XJPW    MAIN,D0,A0,D1
;
;       BITOPC -- generate opcodes for bit instructions
;
BITOPC  MOVE.W  D4,D0
        AND.W   #3,D0
TO      XJPB    BITTBL,D0,A0,D1,TO
OPBTST  OPCODE  4,'BTST'
        RTS
OPBCHG  OPCODE  4,'BCHG'
        RTS
OPBCLR  OPCODE  4,'BCLR'
        RTS
OPBSET  OPCODE  4,'BSET'
        RTS
;
; bit manipulation, move peripheral, immediate instructions
;
CASE0   BTST    #8,D7
        BEQ.S   CASE0C
        CMP.W   #1,D5
        BEQ.S   CASE0A
        BSR.S   BITOPC
        BSR.S   DATARD3
COMMAEF BSR     COMMA
EA56F   EA      D5,D6,#FALSE
        RTS
CASE0A  OPCODE  5,'MOVEP'
        MOVE.W  D4,D0
        AND.W   #1,D0
        ADD.W   #1,D0
        APPNDSZ D0
        TST.B   D7
        BPL.S   CASE0B
        BSR.S   DATARD3
        BSR     COMMA
        EA      #5,D6,#FALSE
        RTS
CASE0B  EA      #5,D6,#FALSE
COMMAD3 BSR     COMMA
DATARD3 DATAREG D3
        RTS
CASE0C  XJPB    C0TBL,D3,A0,D1,CASE0C
C0S0    OPCODE  3,'ORI'
IMMELOG APPNDSZ D4
        MAKEIMM D4
        BSR     COMMA
        CMP.W   #7,D5
        BNE.S   EALONG
        CMP.W   #4,D6
        BNE.S   EALONG
        B6TO7   D0
        BNE     APPNDSR
        BRA     APPNCCR
COS1    OPCODE  4,'ANDI'
        BRA.S   IMMELOG
COS2    OPCODE  4,'SUBI'
IMMEARI APPNDSZ D4
        MAKEIMM D4
COMMALN BSR     COMMA
EALONG  EA      D5,D6,LONG
        RTS
COS3    OPCODE  4,'ADDI'
        BRA.S   IMMEARI
COS4    BSR.S   BITOPC
        EA      #7,#4,#FALSE
        BRA     COMMAEF
COS5    OPCODE  4,'EORI'
        BRA.S   IMMELOG
COS6    OPCODE  4,'CMPI'
        BRA.S   IMMEARI
COS7    ILLEGAL
        RTS
;
; Move byte
;
CASE1   OPCODE  6,'MOVE.B'
EA2EA   BSR     EA56F
        BSR     COMMA
        EA      D4,D3,#FALSE
        RTS
;
; Move long
;
CASE2   OPCODE  6,'MOVE.L'
        EA      D5,D6,#TRUE
        BSR     COMMA
        EA      D4,D3,#TRUE
        RTS
;
; Move word
;
CASE3   OPCODE  6,'MOVE.W'
        BRA.S   EA2EA
;
; Misc.
;
CASE4   CMP.W   #6,D4
        BNE.S   @1
        OPCODE  3,'CHK'
        BRA     EA2D3
@1      CMP.W   #7,D4
        BNE.S   @2
        OPCODE  3,'LEA'
        EA      D5,D6,#TRUE
        BSR     COMMA
        ADDRREG D3
        RTS
@2      XJPW    C4TBL,D3,A0,D1
C4S0    B6TO7   D0
        CMP.W   #3,D0
        BNE.S   @1
        OPCODE  4,'MOVE'
        BSR     APPNDSR
        BRA     COMMAEF
@1      OPCODE  4,'NEGX'
D4EALNG APPNDSZ D4
        BRA     EALONG
C4S1    B6T07   D0
        CMP.W   #3,D0
        BNE.S   @1
        ILLEGAL
        RTS
@1      OPCODE  3,'CLR'
        BRA     D4EALNG
C4S2    B6T07   D0
        CMP.W   #3,D0
        BNE.S   C4S2A
        OPCODE  4,'MOVE'
        BSR     EA56F
        BSR     COMMA
APPNCCR APPND   3,'CCR'
        RTS
C4S2A   OPCODE  3,'NEG'
        BRA     D4EALNG
C4S3    B6T07   D0
        CMP.W   #3,D0
        BNE.S   C4S3A
        OPCODE  4,'MOVE'
        BSR     EA56F
        BSR     COMMA
APPNDSR APPND   2,'SR'
        RTS
C4S3A   OPCODE  3,'NOT'
        BRA     D4EALNG
C4S4    B6T07   D0
T1      XJPB    C4S4TBL,D0,A0,D1,T1
C4S4S0  OPCODE  4,'NBCD'
        BRA     EA56F
C4S4S1  TST.W   D5
        BNE.S   NOTSWAP
        OPCODE  4,'SWAP'
DATARD6 DATAREG D6
        RTS
NOTSWAP OPCODE  3,'PEA'
        BRA     EA56F
C4S4S2  TST.W   D5
        BNE.S   @1
        OPCODE  5,'EXT.W'
        BRA     DATARD6
@1      OPCODE  7,'MOVEM.W'
REGLSEA BSR     REGLIST
        BRA     COMMAEF
C4S4S3  TST.W   D5
        BNE.S   @1
        OPCODE  5,'EXT.L'
        BRA     DATARD6
@1      OPCODE  7,'MOVEM.L'
        BRA     REGLSEA
C4S5    B6T07   D0
        CMP.W   #3,D0
        BNE.S   @1
        OPCODE  3,'TAS'
        BRA     EA56F
@1      OPCODE  3,'TST'
        BRA     D4EALNG
```

```
C4S6    OPCODE  5,'MOVEM'
        MOVE.W  D4,D0
        AND.W   #1,D0
        ADDQ    #1,D0
        APPNDSZ D0
        NxtWord                 ; pop reglist on
        BSR     EA56F
        BSR     COMMA
        BRA     reg2list
;        RTS
C4S7    B6T07   D0
T2      XJPB    C4S7TBL,D0,A0,D1,T2
C4S7S0  ILLEGAL
        RTS
C4S7S1  MOVE.W  D5,D0
        AND.W   #6,D0
        BNE.S   T3
        OPCODE  4,'TRAP'
        SHORTIM D7
        RTS
T3      XJPB    C4S7S1T,D5,A0,D1,T3
C4S7S2  OPCODE  3,'JSR'
        BRA     EA56F
C4S7S3  OPCODE  3,'JMP'
        BRA     EA56F
C4S7S10
C4S7S11 ILLEGAL
        RTS
C4S7S12 OPCODE  4,'LINK'
        ADDRREG D6
        BSR     COMMA
        EA      #7,#4,#FALSE
        RTS
C4S7S13 OPCODE  4,'UNLK'
        ADDRREG D6
        RTS
C4S7S14 OPCODE  4,'MOVE'
        ADDRREG D6
        BSR     COMMA
        APPND   3,'USP'
        RTS
C4S7S15 OPCODE  4,'MOVE'
        APPND   3,'USP'
        BSR     COMMA
        ADDRREG D6
C4S7S17 ILLEGAL
        RTS
C4S7S16 XJPB    MISC,D6,A0,D1,C4S7S16
MISC0   OPCODE  5,'RESET'
        RTS
MISC1   OPCODE  3,'NOP'
        RTS
MISC2   OPCODE  4,'STOP'
        EA      #7,#4,#FALSE
        RTS
MISC3   OPCODE  3,'RTE'
        RTS
MISC4   ILLEGAL
        RTS
MISC5   OPCODE  3,'RTS'
        RTS
MISC6   OPCODE  5,'TRAPV'
```

```
        RTS
MISC7   OPCODE  3,'RTR'
        RTS
;
; Add and subtract quick, set conditionally, decrements
;
CASE5   B6T07   D0
        CMP.W   #3,D0
        BNE     CASE5A
        CMP.W   #1,D5
        BNE     SETCC
        MOVE.W  D7,D0
        LSR.W   #8,D0
        AND.W   #$F,D0
        XJPW    C5DBCC,D0,A0,D1
C5DB0   OPCODE  3,'DBT'
        BRA     DBOPNDS
C5DB1   OPCODE  4,'DBRA'
        BRA     DBOPNDS
C5DB2   OPCODE  4,'DBHI'
        BRA     DBOPNDS
C5DB3   OPCODE  4,'DBLS'
        BRA     DBOPNDS
C5DB4   OPCODE  4,'DBCC'
        BRA     DBOPNDS
C5DB5   OPCODE  4,'DBCS'
        BRA     DBOPNDS
C5DB6   OPCODE  4,'DBNE'
        BRA.S   DBOPNDS
C5DB7   OPCODE  4,'DBEQ'
        BRA.S   DBOPNDS
C5DB8   OPCODE  4,'DBVC'
        BRA.S   DBOPNDS
C5DB9   OPCODE  4,'DBVS'
        BRA.S   DBOPNDS
C5DB10  OPCODE  4,'DBPL'
        BRA.S   DBOPNDS
C5DB11  OPCODE  4,'DBMI'
        BRA.S   DBOPNDS
C5DB12  OPCODE  4,'DBGE'
        BRA.S   DBOPNDS
C5DB13  OPCODE  4,'DBLT'
        BRA.S   DBOPNDS
C5DB14  OPCODE  4,'DBGT'
        BRA.S   DBOPNDS
C5DB15  OPCODE  4,'DBLE'
DBOPNDS BSR     DATARD6
        BSR     COMMA
        EA      #7,#2,#FALSE
        RTS
SETCC   MOVE.W  D7,D0
        LSR.W   #8,D0
        AND.W   #$F,D0
        XJPW    C5SCC,D0,A0,D1
C5SCC0  OPCODE  2,'ST'
        BRA     SCOPNDS
C5SCC1  OPCODE  2,'SF'
        BRA     SCOPNDS
C5SCC2  OPCODE  3,'SHI'
        BRA     SCOPNDS
C5SCC3  OPCODE  3,'SLS'
        BRA     SCOPNDS
```

```
C5SCC4  OPCODE  3,'SCC'
        BRA     SCOPNDS
C5SCC5  OPCODE  3,'SCS'
        BRA     SCOPNDS
C5SCC6  OPCODE  3,'SNE'
        BRA.S   SCOPNDS
C5SCC7  OPCODE  3,'SEQ'
        BRA.S   SCOPNDS
C5SCC8  OPCODE  3,'SVC'
        BRA.S   SCOPNDS
C5SCC9  OPCODE  3,'SVS'
        BRA.S   SCOPNDS
C5SCC10 OPCODE  3,'SPL'
        BRA.S   SCOPNDS
C5SCC11 OPCODE  3,'SMI'
        BRA.S   SCOPNDS
C5SCC12 OPCODE  3,'SGE'
        BRA.S   SCOPNDS
C5SCC13 OPCODE  3,'SLT'
        BRA.S   SCOPNDS
C5SCC14 OPCODE  3,'SGT'
        BRA.S   SCOPNDS
C5SCC15 OPCODE  3,'SLE'
SCOPNDS BRA     EA56F
CASE5A  BTST    #8,D7
        BEQ.S   CASE5B
        OPCODE  4,'SUBQ'
        BRA.S   QUICK
CASE5B  OPCODE  4,'ADDQ'
QUICK   APPNDSZ D4
        USE8IFO D3
        SHORTIM D0
        BRA     COMMALN
;
; Branch conditionally
;
CASE6   MOVE.W  D7,D0
        LSR.W   #8,D0
        AND.W   #$F,D0
        XJPW    C6BCC,D0,A0,D1
C6BCC0  OPCODE  3,'BRA'
        BRA     BCOPNDS
C6BCC1  OPCODE  3,'BSR'
        BRA     BCOPNDS
C6BCC2  OPCODE  3,'BHI'
        BRA     BCOPNDS
C6BCC3  OPCODE  3,'BLS'
        BRA     BCOPNDS
C6BCC4  OPCODE  3,'BCC'
        BRA     BCOPNDS
C6BCC5  OPCODE  3,'BCS'
        BRA     BCOPNDS
C6BCC6  OPCODE  3,'BNE'
        BRA.S   BCOPNDS
C6BCC7  OPCODE  3,'BEQ'
        BRA.S   BCOPNDS
C6BCC8  OPCODE  3,'BVC'
        BRA.S   BCOPNDS
C6BCC9  OPCODE  3,'BVS'
        BRA.S   BCOPNDS
C6BCC10 OPCODE  3,'BPL'
        BRA.S   BCOPNDS
```

```
C6BCC11 OPCODE  3,'BMI'
        BRA.S   BCOPNDS
C6BCC12 OPCODE  3,'BGE'
        BRA.S   BCOPNDS
C6BCC13 OPCODE  3,'BLT'
        BRA.S   BCOPNDS
C6BCC14 OPCODE  3,'BGT'
        BRA.S   BCOPNDS
C6BCC15 OPCODE  3,'BLE'
BCOPNDS TST.B   D7
        BNE.S   SHORTBR
        NXTWORD
        TST.W   (A7)
        BPL.S   @1
        APPND   3,'*-$'
        MOVE.W  (A7),-(A7)
        NEG.W   (A7)
        BRA.S   @2
@1      APPND   3,'*+$'
        MOVE.W  (A7),-(A7)
@2      BSR     HEX4
        BSR     TABDEST
        MOVE.L  A5,A0
        SUBQ    #2,A0
        BRA     DOLOC
SHORTBR OPCODE  2,'.S'
        MOVE.W  D7,D0
        EXT.W   D0
        MOVE.W  D0,-(A7)
        BPL.S   @1
        APPND   3,'*-$'
        MOVE.W  (A7),-(A7)
        NEG.W   (A7)
        BRA.S   DOSMALL
@1      APPND   3,'*+$'
        MOVE.W  (A7),-(A7)
DOSMALL BSR     HEX2
        BSR     TABDEST
        MOVE.L  A5,A0
DOLOC   ADD.W   (A7)+,A0
        MOVE.L  A0,-(A7)
        MOVE.L  A3,D0
        ADDQ.L  #1,D0
        MOVE.L  D0,-(A7)
        BSR     LOOKUP
        BSR     HEX4
        BSR     HEX4
        RTS
;
; Move quick
;
CASE7   BTST    #8,D7
        BEQ.S   @1
        OPCODE  3,'SOB'
        BSR     DATARD3
        BSR     COMMA
        APPND   3,'*-$'
        MOVE.W  D7,D0
        AND.W   #$FF,D0
        MOVE.B  D0,D1
        OR.W    #$FF00,D1
        MOVE.W  D1,-(A7)
        NEG.W   D0
        MOVE.W  D0,-(A7)
        BSR     HEX2
        BSR     TABDEST
        MOVE.L  A5,A0
        ADD.W   (A7)+,A0
        MOVE.L  A0,-(A7)
        BSR     HEX4
        BSR     HEX4
        RTS
@1      OPCODE  5,'MOVEQ'
        TST.B   D7
        BPL.S   @2
        APPND   3,'#-$'
        BRA.S   @3
@2      APPND   2,'#$'
@3      MOVE.W  D7,D0
        AND.W   #$7F,D0
        MOVE.W  D0,-(A7)
        BSR     HEX2
        BRA     COMMAD3
;
; Or, divide, subtract decimal
;
CASE8   MOVE.W  D7,D0
        AND.W   #$1F0,D0
        CMP.W   #$100,D0
        BNE.S   SDIV
        OPCODE  4,'SBCD'
        BRA     AORSBCD
SDIV    CMP.W   #7,D4
        BNE.S   UDIV
        OPCODE  4,'DIVS'
EA2D3   BSR     EA56F
        BRA     COMMAD3
UDIV    CMP.W   #3,D4
        BNE.S   @1
        OPCODE  4,'DIVU'
        BRA     EA2D3
@1      OPCODE  2,'OR'
DEANEAD APPNDSZ D4
        BTST    #8,D7
        BEQ.S   EADATA
DATAEA  BSR     DATARD3
        BRA     COMMALN
EADATA  BSR     EALONG
        BRA     COMMAD3
;
; Subtract
;
CASE9   BTST    #8,D7
        BEQ.S   CASE9A
        MOVE.W  D7,D0
        AND.W   #$30,D0
        BNE.S   CASE9A
        MOVE.W  D7,D0
        AND.W   #$C0,D0
        CMP.W   #$C0,D0
        BEQ.S   CASE9A
        OPCODE  4,'SUBX'
ADDSUBX APPNDSZ D4
AORSBCD CMP.W   #1,D5
```

```
        BNE.S   @1
        EA      #4,D6,LONG
        BSR     COMMA
        EA      #4,D3,LONG
        RTS
@1      BSR     DATARD6
        BRA     COMMAD3
CASE9A  CMP.W   #3,D4
        BNE.S   CASE9B
        OPCODE  6,'SUBA.W'
EAADDR  CMP.W   #7,D4
        BNE.S   @1
        EA      D5,D6,#TRUE
        BRA.S   @2
@1      BSR     EA56F
@2      BSR     COMMA
        ADDRREG D3
        RTS
CASE9B  CMP.W   #7,D4
        BNE.S   CASE9C
        OPCODE  6,'SUBA.L'
        BRA.S   EAADDR
        RTS
CASE9C  OPCODE  3,'SUB'
        BRA     DEANEAD
;
; unassigned
;
;
; L1010 - CORE Routines
;
; D7 is bits0to15
; D6 is bits0to2
; D5 is bits3to5
; D4 is bits6to8
; D3 is bits9to11
CASEA

        BTST    #11,D7                          ; see if tool trap
        BEQ.S   osTrap                          ; an os trap
        OPCODE  7,'ToolBox'
        BRA.S   goTrap

osTrap
        OPCODE  7,'OSTrap '
goTrap
        APPND   1,'$'                           ; add hex sign to operand
        MOVE.W  D7,-(A7)
        BSR     HEX4

        .IF     TNames

        BSR     TABDEST                         ; tab to next 15th, print '; '

        MOVEQ   #0,D0                           ; clear trap number
        MOVE.W  D7,D0                           ; get low word
        AND.W   #$1FF,D0                        ; make it a trap number

        BTST    #11,D7                          ; is it a tooltrap?
        BNE.S   @0                              ; yes, skip restriction

        BCLR    #8,D0                           ; restrict < 256
```

```
@0      MOVE.L  A3,A0                   ; A0 = ptr to operand
        MOVEQ   #0,D1                   ; clear out length
        MOVE.B  (A0)+,D1                ; get count
        ADD.L   D1,A0                   ; now A0 pts to first open char

        BSR     LookupTrap              ; look up the trap name

        ADD.B   #10,(A3)                ; bump char count by 10
        .ENDC

        RTS
;
; Compare, exclusive or
;
CASEB   BTST    #8,D7
        BEQ.S   CASEBA
        CMP.W   #1,D5
        BNE.S   CASEBA
        MOVE.W  D7,D0
        AND.W   #$C0,D0
        CMP.W   #$C0,D0
        BEQ.S   CASEBA
        OPCODE  4,'CMPM'
        APPNDSZ D4
        EA      #3,D6,#FALSE
        BSR     COMMA
        EA      #3,D3,#FALSE
        RTS
CASEBA  BTST    #8,D7
        BEQ.S   CASEBB
        MOVE.W  D7,D0
        AND.W   #$C0,D0
        CMP.W   #$C0,D0
        BEQ.S   CASEBB
        OPCODE  3,'EOR'
        APPNDSZ D4
        BRA     DATAEA
CASEBB  CMP.W   #3,D4
        BNE.S   CASEBC
        OPCODE  6,'CMPA.W'
        BRA     EAADDR
        RTS
CASEBC  CMP.W   #7,D4
        BNE.S   CASEBD
        OPCODE  6,'CMPA.L'
        BRA     EAADDR
        RTS
CASEBD  OPCODE  3,'CMP'
        APPNDSZ D4
        BSR     EALONG
        BRA     COMMAD3
;
; And, Multiply, Add decimal, Exchange
;
CASEC   CMP.W   #5,D4
        BNE.S   @1
        TST.W   D5
        BNE.S   @1
        OPCODE  3,'EXG'
        BSR     DATARD3
        BRA     COMMAD6
```

```
@1      CMP.W   #5,D4
        BNE.S   @2
        CMP.W   #1,D5
        BNE.S   @2
        OPCODE  3,'EXG'
        ADDRREG D3
        BSR     COMMA
        ADDRREG D6
        RTS
@2      CMP.W   #6,D4
        BNE.S   BCDADD
        CMP.W   #1,D5
        BNE.S   BCDADD
        OPCODE  3,'EXG'
        BSR     DATARD3
        BSR     COMMA
        ADDRREG D6
        RTS
BCDADD  MOVE.W  D7,D0
        AND.W   #$1F0,D0
        CMP.W   #$100,D0
        BNE.S   SMUL
        OPCODE  4,'ABCD'
        BRA     AORSBCD
SMUL    CMP.W   #7,D4
        BNE.S   UMUL
        OPCODE  4,'MULS'
        BRA     EA2D3
UMUL    CMP.W   #3,D4
        BNE.S   @1
        OPCODE  4,'MULU'
        BRA     EA2D3
@1      OPCODE  3,'AND'
        BRA     DEANEAD
;
; Add
;
CASED   BTST    #8,D7
        BEQ.S   CASEDA
        MOVE.W  D7,D0
        AND.W   #$30,D0
        BNE.S   CASEDA
        MOVE.W  D7,D0
        AND.W   #$C0,D0
        CMP.W   #$C0,D0
        BEQ.S   CASEDA
        OPCODE  4,'ADDX'
        BRA     ADDSUBX
CASEDA  CMP.W   #3,D4
        BNE.S   CASEDB
        OPCODE  6,'ADDA.W'
        BRA     EAADDR
        RTS
CASEDB  CMP.W   #7,D4
        BNE.S   CASEDC
        OPCODE  6,'ADDA.L'
        BRA     EAADDR
        RTS
CASEDC  OPCODE  3,'ADD'
        BRA     DEANEAD
;
; PUTSHFT -- put shift opcode
```

```
;
PUTSHFT XJPB    SHFTTBL,D0,A0,D1,PUTSHFT
SHIFT0  OPCODE  3,'ASR'
        RTS
SHIFT1  OPCODE  3,'ASL'
        RTS
SHIFT2  OPCODE  3,'LSR'
        RTS
SHIFT3  OPCODE  3,'LSL'
        RTS
SHIFT4  OPCODE  4,'ROXR'
        RTS
SHIFT5  OPCODE  4,'ROXL'
        RTS
SHIFT6  OPCODE  3,'ROR'
        RTS
SHIFT7  OPCODE  3,'ROL'
        RTS
;
; Shifts and rotates
;
CASEE   B6T07   D0
        CMP.W   #3,D0
        BNE.S   DRSHIFT
        MOVE.W  D7,D0
        LSR.W   #8,D0
        AND.W   #7,D0
        BSR     PUTSHFT
        APPNDSZ #1
        BRA     EA56F
DRSHIFT MOVE.W  D5,D0
        AND.W   #3,D0
        ADD.W   D0,D0
        BTST    #8,D7
        BEQ.S   @1
        ADDQ    #1,D0
@1      BSR     PUTSHFT
        APPNDSZ D4
        BTST    #5,D7
        BEQ.S   @2
        BSR     DATARD3
        BRA.S   COMMAD6
@2      USE8IFO D3
        SHORTIM D0
COMMAD6 BSR     COMMA
        BRA     DATARD6
;
; unassigned
;
CASEF   ILLEGAL
        RTS
;
;       MAIN TRANSFER TABLE
;
MAIN    .WORD   CASE0-B,CASE1-B,CASE2-B,CASE3-B
        .WORD   CASE4-B,CASE5-B,CASE6-B,CASE7-B
        .WORD   CASE8-B,CASE9-B,CASEA-B,CASEB-B
        .WORD   CASEC-B,CASED-B,CASEE-B,CASEF-B
;
;       BITTBL
;
BITTBL  .BYTE   OPBTST-T0,OPBCHG-T0,OPBCLR-T0,OPBSET-T0
```

```
;
;        COTBL
;
COTBL    .BYTE   COS0-CASEOC,COS1-CASEOC,COS2-CASEOC,COS3-CASEOC
         .BYTE   COS4-CASEOC,COS5-CASEOC,COS6-CASEOC,COS7-CASEOC
;
;        C4TBL
;
C4TBL    .WORD   C4S0-B,C4S1-B,C4S2-B,C4S3-B
         .WORD   C4S4-B,C4S5-B,C4S6-B,C4S7-B
;
;        C4S4TBL
;
C4S4TBL  .BYTE   C4S4S0-T1,C4S4S1-T1,C4S4S2-T1,C4S4S3-T1
;
;        C4S7TBL
;
C4S7TBL  .BYTE   C4S7S0-T2,C4S7S1-T2,C4S7S2-T2,C4S7S3-T2
;
;        C4S7S1T
;
C4S7S1T  .BYTE   C4S7S10-T3,C4S7S11-T3,C4S7S12-T3,C4S7S13-T3
         .BYTE   C4S7S14-T3,C4S7S15-T3,C4S7S16-T3,C4S7S17-T3
;
;        MISC TRANSFER TABLE
;
MISC     .BYTE   MISC0-C4S7S16,MISC1-C4S7S16,MISC2-C4S7S16,MISC3-C4S7S16
         .BYTE   MISC4-C4S7S16,MISC5-C4S7S16,MISC6-C4S7S16,MISC7-C4S7S16
;
;        C5DBCC TRANSFER TABLE
;
C5DBCC   .WORD   C5DB0-B,C5DB1-B,C5DB2-B,C5DB3-B
         .WORD   C5DB4-B,C5DB5-B,C5DB6-B,C5DB7-B
         .WORD   C5DB8-B,C5DB9-B,C5DB10-B,C5DB11-B
         .WORD   C5DB12-B,C5DB13-B,C5DB14-B,C5DB15-B
;
;        C5SCC TRANSFER TABLE
;
C5SCC    .WORD   C5SCC0-B,C5SCC1-B,C5SCC2-B,C5SCC3-B
         .WORD   C5SCC4-B,C5SCC5-B,C5SCC6-B,C5SCC7-B
         .WORD   C5SCC8-B,C5SCC9-B,C5SCC10-B,C5SCC11-B
         .WORD   C5SCC12-B,C5SCC13-B,C5SCC14-B,C5SCC15-B
;
;        C6BCC TRANSFER TABLE
;
C6BCC    .WORD   C6BCC0-B,C6BCC1-B,C6BCC2-B,C6BCC3-B
         .WORD   C6BCC4-B,C6BCC5-B,C6BCC6-B,C6BCC7-B
         .WORD   C6BCC8-B,C6BCC9-B,C6BCC10-B,C6BCC11-B
         .WORD   C6BCC12-B,C6BCC13-B,C6BCC14-B,C6BCC15-B
;
;        SHFTTBL TRANSFER TABLE
;
SHFTTBL  .BYTE   SHIFT0-PUTSHFT,SHIFT1-PUTSHFT,SHIFT2-PUTSHFT,SHIFT3-PUTSHFT
         .BYTE   SHIFT4-PUTSHFT,SHIFT5-PUTSHFT,SHIFT6-PUTSHFT,SHIFT7-PUTSHFT
;
;        EAMAIN TRANSFER TABLE
;
EAMAIN   .WORD   EACAS0-B,EACAS1-B,EACAS2-B,EACAS3-B
         .WORD   EACAS4-B,EACAS5-B,EACAS6-B,EACAS7-B
;
;        EAC7 TRANSFER TABLE
;
```

```
EAC7    .WORD   EAC7S0-B,EAC7S1-B,EAC7S2-B,EAC7S3-B
        .WORD   EAC7S4-B,EAC7S5-B,EAC7S6-B,EAC7S7-B
;
;       THEEND -- USED BY SHELL TO COPY DISASSEMBLER
;
        .WORD   0,0,0
THEEND  .WORD   0
;
```