PDP8 Simple Assembly Language (SAL)

This language permits a program to be input as a number of sections. A section is a sequence of instructions preceded by a section heading 'π <OCTAL WORD>' which gives the address of the first instruction in the section. A section may contain labels and instructions which reference the labels, but references to labels in other sections are not permitted. The instructions in a section must not cross a page boundary. A section is terminated either by the start of the next section or, in the case of the last section, by the entry directive. The entry directive takes the form 'E <OCTAL WORD>' where the octal word is the entry address.

Some instructions in SAL may be written in mnemonic form and all may be written in octal. Constants may be written in octal or signed decimal. Where the address field of a mnemonic function is relevant it should be expressed as either 3 octal digits or as a label. Following a label '*' may be used to indicate indirection. A line containing '.' will cause a halt. Spurious 'CR's' will be faulted.

Whilst the program is being loaded some diagnostics are printed. At each section heading the symbol # is printed, followed by the total number of faults in octal. If a fault is detected in an instruction or constant the symbol '*' followed by the line number (in octal) is printed. However, many syntax faults will not be detected. At the start of each section the fault count is incremented by one for each label referred to but not set in the previous section.

If the fault count is zero when the enter directive is read an E will be printed and the program entered. If non-zero, the fault count will be printed and the program will not be entered.

Operating Instructions

To load the binary tape containing SAL the following sequence must be loaded by hand and entered at line 1.

1	7106
2	7106
3	7106
4	6014
5	6011
6	5205
7	6012
10	7430
11	3613
12	5201
13	6377

This sequence will load and enter the SAL assembler. At the end of the binary tape there is a program which makes a copy of SAL in store stack 1. There are two entry points to the SAL assembler stored in stack 1. The normal entry point (6400) prints 8 line feeds and makes a fresh copy of SAL in stack 0 which is entered. The second entry point (6500) is to be used when a program fails. It prints lines 0 to 17 in octal then halts. Because of this lines 0 to 17 should be used for quantities whose value faults can be diagnosed. When the fault print halts it may be used to print further areas of store by:-

setting a starting address on the switch registor
operating 'CONT'
setting a count on the switch registor
operating 'CONT'

On completion the print routine will halt and it may be restarted by repeating the above procedure.

SYNTAX OF SIMPLE ASSEMBLY LANGUAGE (SAL)

<PROGRAM> ::= <SECTIONS><ENTER ST>

<SECTIONS> ::= <SECTION> | <SECTIONS> <SECTION>

<ENTER ST> ::= E<OCTAL WORD> <NEWLINE>

<SECTION> ::= <STARTING ADDRESS> <WORD SEQUENCE>

<STARTING ADDRESS> ::= *<OCTAL WORD> <NEWLINE>

<WORD SEQUENCE> ::= <LABELLED WORD> | <WORD SEQUENCE> <LABELLED WORD>

<LABELLED WORD> ::= <WORD> <NEWLINE> | <LABEL> <WORD> <NEWLINE>

<WORD> ::= <INST> | <CONST> | .

<INST> ::= <MNEUMONIC CODE> <ADDR PART>

<CONST> ::= <OCTAL WORD> | +<INT> | -<INT>

<MNEUMONIC CODE> ::= AND | CLA | CLL | CMA | CML | DCA | HLT |
ISZ | IAC | JMS | JMP | RAR | RAL | RTR |
RTL | SMA | SZA | SPA | SNA | SNL | SZL |
SKP | TAD | CIA |

<ADDR PART> ::= <LABEL > | <LABEL>* | <OD> <OD> <OD> | <EMPTY>

<LABEL> ::= (<INT>)

<OCTAL WORD> ::= <OD> <OD> <OD> <OD>

<INT> ::= <DD> | <INT> <DD>

<NEWLINE> ::= 'CR' 'LF' | <COMMENT> 'CR' 'LF'

<COMMENT> ::= :<TEXT>

m7600

```
(20)    0000        :read and edit
        7300        :cla.clL
         tad(10)
        dca(17)     :reset overwritten inst
        tad(1)
        dca 010     :set ptr
        isz 005     :inc line count

(3)     dca 007     :set shift (f.s. initially)
(4)     6014
(5)     6011
        jmp(5)
        6012        :acc=next ch
        and(6)
        tad 007     :add set 'bit'
        tad(7)
        dca 060
        tad 400     :acc=accth entry in table
        spa
        jmp(9)      :jump if fn.letter
        and (1)
        tad(11)     :form switch jump
        dca (12)
        tad 400
        and (1)
        tad (13)
(2)     cla
        szl
         jmp(9)     :jump if code > 4
(12)    0000        :computed
(11)    jmp(14)
(14)    jmp(10)     :cr
        jmp (4)     :sp etc
```

```
        jmp(3): fs
        jmp(15): ls
        jmp(16): :
(1)     0017
(6)     0037
(7)     7500        :addr of conv table
(13)    7773
(15)    tad(19)     :change to ls
        jmp (3)
(19)    0040
(9)     7300        :cll cla
        tad 400
(17)    dca 410     :store 'symbol'
        jmp (4)
(16)    tad (2)
        dca (17)    :insert cla
        jmp (4)
(10)    dca 410
        tad (1)
        dca 016
        jmp (20)*:exit
(40)    0000        :read octal word
        7300        : cll cla
        tad (41)
        dca 000     :set count

(44)    ral
        rtl         :acc <-3
        dca 001     :acc=> dump

        tad 416     :acc=next 'symbol'
        dca 002     :acc=> dump'
        tad 002
        and (1)     :acc + %0017
```

```
        tad (42)   :acc+(-10)
        sza
        jmp(31) *  :jump if not od
        tad 002
        rtr
        rtr
        cll
        and (43)   :acc=od
        tad 001    :acc+ow
        isz 000
        jmp (44)   :repeat 7 times
        jmp (40) *:exit

(41)    7774
(42)    -10
(43)    0007
(60)    0000       :read dec.int
        7300       :cll cla
(59)    dca 001    :acc=>dump
        tad 416    :acc=next'symbol'
        dca 002    :acc=>dump'
        tad 002
        and (1)
        tad (61)
        sza
        jmp (62)   :jmp if not 6,9
(63)    tad 002    :acc='symbol'
        rtr
        rtr        :acc->4
        and (1)    :acc & 0017
        dca 002    :acc=>dump'
        cll
        tad 001    :acc=dump
```

```
        rtl
        tad 001
        ral         :acc * 10
        tad 002     :acc+dump'
        jmp(59)

(62)    iac
        sna
        jmp (63) :jmp if octal digit
        7300     :cll cla
        tad 001
        jmp (60)*:exit
(61)    -11
(50)    jms (40):entry sequence
        dca 007
        jms (90)*
        tad 004
        sza
        jmp (93) :jump if faults
        tad (96)
        jms (70)*:print e
        jmp 407  :enter
(93)    jms (80)*:print fault count
        7402     :halt

(90)    7000
(96)    0305
(70)    7131
(80)    7136

(31)    7370
```

```
n/400     :function table
0027      :n-1 names
0222      :and
0560      :cla
0567      :cll
0600      :cma
0607      :cml
1020      :dca
1575      :hlt
2316      :isz
2001      :iac
2614      :jms
2612      :jmp
5413      :rar
5407      :ral
5733      :rtr
5727      :rtl
6200      :sma
6340      :sza
6240      :spa
6220      :sna
6227      :snl
6347      :szl
6152      :skp
6402      :tad
0500      :cia
```

```
0000        :functions
7200
7100
7040
7020
3000
7402
2000
7001
4000
5000
7010
7004
7012
7006
7500
7440
7510
7450
7420
7430
7510
1000
7041
```

```
π/500     :conversion table
0001      :Ts
1412      :0
1613      :8
1677      :1
1512      :4
1544      :5
1777      :?
1001      :sp
1452      :2
1717      :<
1737      :=
1270      :+
1237      :)
1552      :6
1555      :.
0520      :cr
1432      :1
1757      :>
1037      :|
1633      :9
1214      :(
1532      :5
0241      :lf
1177      :'
1257      :*
1472      :3
1331      :-
3763      :ls
1572      :7
0577      :/
1317      :,
3761      :er
```

```
0002    :fs
6412    :p
6203    :h
2617    :x
6102    :d
6515    :t
6307    :l
1001    :sp
2057    :b
6453    :r
6245    :j
6656    :z
2157    :f
2557    :v
6351    :n
1066    :m
6020    :a
2437    :q
6224    :i
2637    :y
2127    :e
2537    :u
6330    :m
3777    :_
6061    :c
6474    :s
6266    :k
0001    :ls
2177    :g
2577    :w
2377    :o
3761    :er
```

n7200

```
        dca 004  :fault count=0
        dca 005  :line count=0
        tad (21) :sal main routine
(34)    dca 017  :ni=0177
        tad (20)
        dca 000
        tad (100)
        dca 010
(101)   dca 410  :clear label list
        isz 000
        jmp (101)
(33)    jms (20)*:read line
(107)   tad 416  :acc=1st'symbol'
        spa
(25)    jmp (22) :jmp if fn letter
        and (1)
        tad (23) :compute switch jump
        dca (24)
(24)    0000     :switch
(23)    jmp (25)
        jmp(26)  :n
        jmp (50)*:e
        jmp (28) :+
        jmp (29) *:-
        jmp (30) :od
        jmp (31) :89
        jmp (32)*:(
        hlt
        jmp (33)
        jmp (31)
(21)    0177
(26)    jms (90)*:count labels not set
```

```
        tad (94)
        jms (70)*:print n
        cla
        tad 004
        jms (80)*:print faults
        jms (40)*:read octalword
        tad (119)
        jmp (34)
(119)   -1
(28)    jms (60)*:read dec int
        jmp (120)
(29)    jms (60)*:read dec int
        cia      :negate
        jmp (120)

(30)    7240
        tad 016
        dca 016  :reset line ptr
        jms (40)*:read octal word
        jmp (120)

(22)    and (1)  :sal main routine continued
        rtl
        rtl      :1st letter<-4
        dca 000
        tad 416
        and (1)  :acc=2nd letter
        tad 006  :+1st
        rtl
        rtl      :acc<-4
        dca 000  :
        tad 416
        and (1)
        tad 006  :acc=(1st let<-8) + (2nd <-4) + 3rd
```

```
        cia         :negate acc
        dca 000     :search table sequence
        tad (8)
        dca 010     :set table ptr
        tad (8)*
        cma
        dca 001     :set count

(36)    tad 410     :acc=table entry
        tad 000     :acc+(-req pattern)
        sna
        jmp (35)    :found match
        7300        :cla cll
        isz 001
        jmp (36)    :repeat
        jmp (31)    :illegal code
(35)    tad 010
        tad (8)*
        dca 010     :set ptr to 2nd half of table
        tad 410     :acc=table entry
(120)   dca 417     :store fn part of inst
        tad 017
        dca 003
        tad 416     :acc=next symbol
        and (1)
        tad (37)    :+(-12)
        sza
        jmp (45)*   :jmp if label
        tad 016     :acc=line ptr
        tad (38)    :acc-2
        dca 016
        tad 016
        dca 015
```

```
        tad (39) :acc='zero symbol'
        dca 415
        jms (40)*:read octal 'addr'
        tad 403  :add fn
        dca 403
        jmp (33) :read next inst

(37)    -10
(38)    -2
(39)    1412
(20)    7000
(40)    7065
(32)    7066
(45)    7022
(50)    7750
(60)    7717
(70)    7131
(80)    7136
(90)    7000
(100)   6577
(1)     00%17
(94)    0243
(95)    0252
        0000
(8)     7400

(31)    isz 004
        cla
        tad (95)
        jms (70)*:print *
        cla
        tad 005
        jms (80)*:print line count
        jmp (33)
```

```
π7000


(90)    0000      :count unset labels
        tad (20)
        dca 000   :count
        tad (100)
        dca 010   :ptr
(92)    tad 410
        sna
        jmp (91) :jmp label unused
        spa
        jmp (91) :jmp label set
        isz 004  :fault count +1
(91)    cla
        isz 000
        jmp (92)
        jmp (90)*:exit

(20)    7600
(100)   6577      :addr of label list
(40)    7665

(45)    tad (111)
        sza
        jmp (33)*
        jms (60)*:read label
        tad (100):
        dca 000
        tad 400   :acc=value of label
        and (98) :acc + 0377
        tad 403
        dca 403   :and value of label to fn

        tad 400
        spa
        jmp (99) :jump if label set
        cla
```

```
        tad 005  :form reference ptr
        and (102):+0177
        tad (103):+0200
(99)    dca 400  :store in label list
        tad 416
        tad (104)
        sza      :test for *
        jmp (33)*
        tad 403
        tad (97) :set indirect bit
        dca 403
        jmp (33)*
(60)    7717
(97)    0400
(98)    0377
(102)   0177
(103)   0200
(104)   6521
(106)   4200
(107)   7214
(111)   -2
(33)    7213

(32)    jms (60)* :read label
        tad (106)
        dca 000
(110)   tad 400 :acc=value of label
        sza
        jmp (105) :jmp if not undefined
        tad 017
        iac       :acc=ni
        and (102) :+0177
        tad (106) :+4200
```

```
        dca 400  :store value of label
        jmp (107)*
(31)     7370

(105)   spa
        jmp (31)*:fault label set twice
        and (102):+0177
        dca 001
        tad 017
        and (106):+7600
        tad 001
        dca 001  :store ptr to inst
        tad 401
        and (90) :+0377
        dca 400  :reset link in label list
        tad 017
        and (102):+0177
        tad (112):+0201
        dca 002
        tad 401
        and (109):+7400
        tad 002  :add value of label to fn
        dca 401
        jmp (110)
(108)   7000
(109)   7400

(70)    0000     :print symbol subroutine
(71)    6041
        jmp (71)
        6046
        jmp (70)*

(80)    0000     :print octal word subroutine
        dca 000  :store octal word
```

```
        tad (82) :initialise count
        dca 001
(86)    tad 000  :acc=octal word
        7106     :clear link and acc ↑2
        ral      :acc↑1
        dca 000  :store acc
        tad 000
        ral      :acc ↑1
        and (84) :acc + 0007
        tad (85) :acc+code for '0'
        jms (70) :print octal digit
        cla
        iz 001   :count +1, -> if ≠0
        jmp (86)
        tad (83)
        jms (70)
        tad (81)
        jms (70)
        cla
        jmp (80)*:exit
(83)    0215
(81)    7775
(84)    0007
(85)    0260
(82)    7774
(112)   0201
```