```
 GNU assembler version 2.34 (x86_64-alpine-linux-musl)
         using BFD version (GNU Binutils) 2.34.
 options passed : -aghlms=sectorlisp.lst -g -mtune=i386
 input file     : sectorlisp.S
 output file    : sectorlisp.o
 target         : x86_64-alpine-linux-musl
 time stamp     : 2021-10-26T23:25:01.000-0700
```

```
                    /*-*- mode:unix-assembly; indent-tabs-mode:t; tab-width:8; coding:utf-8     -*-
                    vi: set et ft=asm ts=8 tw=8 fenc=utf-8                                     :vi
                    ┌──────────────────────────────────────────────────────────────────────────────
                    │ Copyright 2020 Justine Alexandra Roberts Tunney
                    │ Copyright 2021 Alain Greppin
                    │
                    │ Permission to use, copy, modify, and/or distribute this software for
                    │ any purpose with or without fee is hereby granted, provided that the
                    │ above copyright notice and this permission notice appear in all copies.
                    │
                    │ THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
                    │ WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
                    │ WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
                    │ AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
                    │ DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
                    │ PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
                    │ TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
                    │ PERFORMANCE OF THIS SOFTWARE.
                    └──────────────────────────────────────────────────────────────────────────────*/

                    // LISP meta-circular evaluator in a MBR

                    .set NIL,               1
                    .set ATOM_T,            9
                    .set ATOM_QUOTE,        13
                    .set ATOM_COND,         25
                    .set ATOM_ATOM,         35
                    .set ATOM_CAR,          45
                    .set ATOM_CDR,          53
                    .set ATOM_CONS,         61
                    .set ATOM_EQ,           71

                    .set q.token,   0x4000
                    .set q.str,     0x4080
                    .set boot,      0x7c00

                    ////////////////////////////////////////////////////////////////////////////////
                    .section .text,"ax",@progbits
                    .globl  _start
                    .code16

0000 EB25           _start: jmp     .init                           # some bios scan for short jump
                    .type kSymbols,@object;
                    kSymbols:
0002 4E494C00               .ascii "NIL\0T\0QUOTE\0COND\0ATOM\0CAR\0CDR\0CONS\0EQ"
     54005155
     4F544500
     434F4E44
     0041544F

                    .type .init,@function
0027 EA000060       .init:  ljmp    $0x600>>4,$_begin               # end of bios data roundup page
     00
002c 0E             _begin: push    %cs                             # memory model cs=ds=es = 0x600
002d 0E                     push    %cs
002e 0E                     push    %cs
002f 1F                     pop     %ds
```

```
 53 0030 07                     pop     %es
 54 0031 17                     pop     %ss
 55 0032 B90076                 mov     $0x7c00-0x600,%cx
 56 0035 89CC                   mov     %cx,%sp
 57 0037 FC                     cld
 58 0038 31C0                   xor     %ax,%ax
 59 003a 8EE0                   mov     %ax,%fs                 # fs = &q.mem
 60 003c 31FF                   xor     %di,%di
 61 003e F3AA                   rep stosb                       # clears our bss memory
 62 0040 BF8040         main:   mov     $q.str,%di
 63 0043 BE0000                 mov     $kSymbols,%si
 64 0046 B92500                 mov     $37,%cx
 65 0049 F3A4                   rep movsb
 66 004b B20A           0:      mov     $'\n',%dl
 67 004d E81300                 call    GetToken
 68 0050 E83000                 call    GetObject
 69 0053 BA0100                 mov     $NIL,%dx
 70 0056 E84C01                 call    Eval
 71 0059 E85900                 call    PrintObject
 72 005c B00D                   mov     $'\r',%al
 73 005e E89300                 call    PutChar
 74 0061 EBE8                   jmp     0b
 75
 76                     GetToken:                               # GetToken():al, dl is q.look
 77 0063 BF0040                 mov     $q.token,%di
 78 0066 88D0           1:      mov     %dl,%al
 79 0068 3C20                   cmp     $' ',%al
 80 006a 7602                   jbe     2f
 81 006c AA                     stosb
 82 006d 91                     xchg    %ax,%cx
 83 006e E87F00         2:      call    GetChar                 # bh = 0 after PutChar
 84 0071 92                     xchg    %ax,%dx                 # dl = q.look
 85 0072 3C20                   cmp     $' ',%al
 86 0074 76F0                   jbe     1b
 87 0076 3C29                   cmp     $')',%al
 88 0078 7605                   jbe     3f
 89 007a 80FA29                 cmp     $')',%dl
 90 007d 77E7                   ja      1b
 91 007f 883D           3:      movb    %bh,(%di)
 92 0081 91                     xchg    %cx,%ax
 93 0082 C3                     ret
 94
 95                     GetObject:                              # called just after GetToken
 96 0083 3C28                   cmpb    $'(',%al
 97 0085 747C                   je      GetList
 98 0087 BE0040                 mov     $q.token,%si
 99                     .Intern:
100 008a 89F3                   mov     %si,%bx                 # save s
101 008c BF8040                 mov     $q.str,%di
102 008f 30C0                   xor     %al,%al
103 0091 B1FF           0:      mov     $-1,%cl
104 0093 57                     push    %di                     # save 1
105 0094 A6             1:      cmpsb
106 0095 7507                   jne     2f
107 0097 3A45FF                 cmp     -1(%di),%al
108 009a 75F8                   jne     1b
109 009c EB10                   jmp     4f
```

```
 110 009e 5E            2:      pop     %si                             # drop 1
 111 009f 89DE                  mov     %bx,%si                         # restore s
 112 00a1 F2AE                  repne scasb
 113 00a3 3A05                  cmp     (%di),%al
 114 00a5 75EA                  jne     0b
 115 00a7 57                    push    %di                             # StpCpy
 116 00a8 AC            3:      lodsb
 117 00a9 AA                    stosb
 118 00aa 84C0                  test    %al,%al
 119 00ac 75FA                  jnz     3b
 120 00ae 58            4:      pop     %ax                             # restore 1
 121 00af 0580BF                add     $-q.str,%ax                     # stc
 122 00b2 11C0                  adc     %ax,%ax                         # ax = 2 * ax + carry
 123 00b4 C3            .ret:   ret
 124
 125                    PrintObject:                                    # PrintObject(x:ax)
 126 00b5 A801                  test    $1,%al
 127 00b7 97                    xchg    %ax,%di
 128 00b8 7410                  jz      .PrintList
 129                    .PrintAtom:
 130 00ba D1EF                  shr     %di
 131 00bc 8DB58040              lea     q.str(%di),%si
 132                    .PrintString:                                   # nul-terminated in si
 133 00c0 AC                    lodsb
 134 00c1 84C0                  test    %al,%al
 135 00c3 74EF                  jz      .ret                            # -> ret
 136 00c5 E82C00                call    PutChar
 137 00c8 EBF6                  jmp     .PrintString
 138                    .PrintList:
 139 00ca B028                  mov     $'(',%al
 140 00cc FF7502        2:      push    2(%di)                          # save 1 Cdr(x)
 141 00cf 8B3D                  mov     (%di),%di                       # di = Car(x)
 142 00d1 E81600                call    .PutObject
 143 00d4 58                    pop     %ax                             # restore 1
 144 00d5 83F801                cmp     $NIL,%ax
 145 00d8 740C                  je      4f
 146 00da A801                  test    $1,%al
 147 00dc 97                    xchg    %ax,%di
 148 00dd B020                  mov     $' ',%al
 149 00df 74EB                  jz      2b
 150 00e1 B0F9                  mov     $249,%al                        # bullet (A·B)
 151 00e3 E80400                call    .PutObject
 152 00e6 B029          4:      mov     $')',%al
 153 00e8 EB0A                  jmp     PutChar
 154                    .PutObject:                                     # .PutObject(c:al,x:di)
 155 00ea E80700                call    PutChar                         # preserves di
 156 00ed 97                    xchg    %di,%ax
 157 00ee EBC5                  jmp     PrintObject
 158
 159                    GetChar:
 160 00f0 31C0                  xor     %ax,%ax                         # get keystroke
 161 00f2 CD16                  int     $0x16                           # keyboard service
 162                                                                    # ah is bios scancode
 163                                                                    # al is ascii character
 164                    PutChar:
 165                    #       push    %bx                             # don't clobber di,si,cx,dx
 166                    #       push    %bp                             # original ibm pc scroll up bug
```

```
 167 00f4 BB0700                mov     $7,%bx                  # normal mda/cga style page zero
 168 00f7 B40E                  mov     $0x0e,%ah               # teletype output al cp437
 169 00f9 CD10                  int     $0x10                   # vidya service
 170                    #       pop     %bp                     # preserves al
 171                    #       pop     %bx
 172 00fb 3C0D                  cmp     $'\r',%al               # don't clobber stuff
 173 00fd 75B5                  jne     .ret
 174 00ff B00A                  mov     $'\n',%al
 175 0101 EBF1                  jmp     PutChar                 # bx volatile, bp never used
 176
 177 0103 E85DFF        GetList:call    GetToken
 178 0106 3C29                  cmpb    $')',%al
 179 0108 747E                  je      .retF
 180 010a E876FF                call    GetObject
 181 010d 50                    push    %ax                     # save 1
 182 010e E8F2FF                call    GetList
 183 0111 96                    xchg    %ax,%si
 184 0112 5F                    pop     %di                     # restore 1
 185 0113 EB16                  jmp     Cons
 186
 187                    ////////////////////////////////////////////////////////////////////////////////
 188
 189 0115 83FF01        Evlis:  cmp     $NIL,%di                # Evlis(m:di,a:dx):ax
 190 0118 741C                  je      1f
 191 011a FF7502                push    2(%di)                  # save 1 Cdr(m)
 192 011d 8B05                  mov     (%di),%ax
 193 011f 52                    push    %dx                     # save a
 194 0120 E88200                call    Eval
 195 0123 5A                    pop     %dx                     # restore a
 196 0124 5F                    pop     %di                     # restore 1
 197 0125 50                    push    %ax                     # save 2
 198 0126 E8ECFF                call    Evlis
 199 0129 96                    xchg    %ax,%si
 200 012a 5F                    pop     %di                     # restore 2
 201                    #       jmp     Cons
 202 012b 97            Cons:   xchg    %di,%ax
 203 012c 8CE7                  mov     %fs,%di
 204 012e 57                    push    %di
 205 012f AB                    stosw
 206 0130 96                    xchg    %si,%ax
 207 0131 AB                    stosw
 208 0132 8EE7                  mov     %di,%fs
 209 0134 58                    pop     %ax
 210 0135 C3                    ret
 211 0136 97            1:      xchg    %di,%ax
 212 0137 C3                    ret
 213
 214 0138 83FF01        Pairlis:cmp     $NIL,%di                # Pairlis(x:di,y:si,a:dx):ax
 215 013b 7417                  je      1f
 216 013d FF7502                push    2(%di)                  # save 1 Cdr(x)
 217 0140 FF7402                push    2(%si)                  # save 2 Cdr(y)
 218 0143 8B3D                  mov     (%di),%di
 219 0145 8B34                  mov     (%si),%si
 220 0147 E8E1FF                call    Cons                    # preserves dx
 221 014a 5E                    pop     %si                     # restore 2
 222 014b 5F                    pop     %di                     # restore 1
 223 014c 50                    push    %ax                     # save 3
```

```
 224 014d E8E8FF                call    Pairlis
 225 0150 96                    xchg    %ax,%si
 226 0151 5F                    pop     %di                     # restore 3
 227 0152 EBD7                  jmp     Cons                    # can be inlined here
 228 0154 92            1:      xchg    %dx,%ax
 229 0155 C3                    ret
 230
 231 0156 A801          Apply:  test    $1,%al                  # Apply(fn:ax,x:si:a:dx):ax
 232 0158 750E                  jnz     .switch
 233 015a 97                    xchg    %ax,%di                 # di = fn
 234 015b 8B7D02        .lambda:mov     2(%di),%di              # di = Cdr(fn)
 235 015e 57                    push    %di                     # save 1
 236 015f 8B3D                  mov     (%di),%di               # di = Cadr(fn)
 237 0161 E8D4FF                call    Pairlis
 238 0164 92                    xchg    %ax,%dx
 239 0165 5F                    pop     %di                     # restore 1
 240 0166 EB76                  jmp     .EvCadr
 241 0168 83F847        .switch:cmp     $ATOM_EQ,%ax
 242 016b 772F                  ja      .dflt1
 243 016d 8B3C                  mov     (%si),%di               # di = Car(x)
 244 016f 3C2D          .ifCar: cmp     $ATOM_CAR,%al
 245 0171 7503                  jne     .ifCdr
 246 0173 8B05                  mov     (%di),%ax
 247 0175 C3                    ret
 248 0176 3C35          .ifCdr: cmp     $ATOM_CDR,%al
 249 0178 7504                  jne     .ifAtom
 250 017a 8B4502                mov     2(%di),%ax
 251 017d C3                    ret
 252 017e 3C23          .ifAtom:cmp     $ATOM_ATOM,%al
 253 0180 750A                  jne     .ifCons
 254 0182 F7C70100              test    $1,%di
 255 0186 7511                  jnz     .retT
 256 0188 B80100        .retF:  mov     $NIL,%ax                # ax = NIL
 257 018b C3                    ret
 258 018c 8B7402        .ifCons:mov     2(%si),%si              # si = Cdr(x)
 259 018f 8B34                  mov     (%si),%si               # si = Cadr(x)
 260 0191 3C3D                  cmp     $ATOM_CONS,%al
 261 0193 7496                  je      Cons
 262 0195 39FE          .isEq:  cmp     %di,%si
 263 0197 75EF                  jne     .retF
 264 0199 B009          .retT:  mov     $ATOM_T,%al             # ax = ATOM_T
 265 019b C3                    ret
 266 019c 56            .dflt1: push    %si                     # save x
 267 019d 52                    push    %dx                     # save a
 268 019e E80400                call    Eval
 269 01a1 5A                    pop     %dx                     # restore a
 270 01a2 5E                    pop     %si                     # restore x
 271 01a3 EBB1                  jmp     Apply
 272
 273 01a5 A801          Eval:   test    $1,%al                  # Eval(e:ax,a:dx):ax
 274 01a7 753A                  jnz     Assoc
 275 01a9 97                    xchg    %ax,%di                 # di = e
 276 01aa 8B05                  mov     (%di),%ax               # ax = Car(e)
 277 01ac 83F80D                cmp     $ATOM_QUOTE,%ax         # maybe CONS
 278 01af 7410                  je      Cadr
 279 01b1 8B7D02                mov     2(%di),%di              # di = Cdr(e)
 280 01b4 83F819                cmp     $ATOM_COND,%ax
```

```
 281 01b7 740E                  je      Evcon
 282 01b9 50            .Ldflt2:push    %ax                             # save 2
 283 01ba E858FF                call    Evlis                           # preserves dx
 284 01bd 96                    xchg    %ax,%si
 285 01be 58                    pop     %ax                             # restore 2
 286 01bf EB95                  jmp     Apply
 287
 288 01c1 8B7D02        Cadr:   mov     2(%di),%di                      # contents of decrement register
 289 01c4 8B05                  mov     (%di),%ax                       # contents of address register
 290 01c6 C3                    ret
 291
 292 01c7 57            Evcon:  push    %di                             # save c
 293 01c8 8B3D                  mov     (%di),%di                       # di = Car(c)
 294 01ca 8B05                  mov     (%di),%ax                       # ax = Caar(c)
 295 01cc 52                    push    %dx                             # save a
 296 01cd E8D5FF                call    Eval
 297 01d0 5A                    pop     %dx                             # restore a
 298 01d1 5F                    pop     %di                             # restore c
 299 01d2 83F801                cmp     $NIL,%ax
 300 01d5 7505                  jne     2f
 301 01d7 8B7D02                mov     2(%di),%di                      # di = Cdr(c)
 302 01da EBEB                  jmp     Evcon
 303 01dc 8B3D          2:      mov     (%di),%di                       # di = Car(c)
 304 01de E8E0FF        .EvCadr:call    Cadr                            # ax = Cadar(c)
 305 01e1 EBC2                  jmp     Eval
 306
 307 01e3 83FA01        Assoc:  cmp     $NIL,%dx                        # Assoc(x:ax,y:dx):ax
 308 01e6 89D6                  mov     %dx,%si
 309 01e8 749E                  je      .retF
 310 01ea 8B1C                  mov     (%si),%bx                       # bx = Car(y)
 311 01ec 8B0F                  mov     (%bx),%cx                       # cx = Caar(y)
 312 01ee 39C8                  cmp     %cx,%ax
 313 01f0 7504                  jne     1f
 314 01f2 8B4702                mov     2(%bx),%ax                      # ax = Cdar(y)
 315 01f5 C3                    ret
 316 01f6 8B5402        1:      mov     2(%si),%dx                      # dx = Cdr(y)
 317 01f9 EBE8                  jmp     Assoc
 318
 319                    .type .sig,@object;
 320                    .sig:
 321 01fb CECECE        .fill 510 - (. - _start), 1, 0xce
 322 01fe 55AA          .word 0xAA55
```

```
DEFINED SYMBOLS
        sectorlisp.S:23     *ABS*:0000000000000001 NIL
        sectorlisp.S:24     *ABS*:0000000000000009 ATOM_T
        sectorlisp.S:25     *ABS*:000000000000000d ATOM_QUOTE
        sectorlisp.S:26     *ABS*:0000000000000019 ATOM_COND
        sectorlisp.S:27     *ABS*:0000000000000023 ATOM_ATOM
        sectorlisp.S:28     *ABS*:000000000000002d ATOM_CAR
        sectorlisp.S:29     *ABS*:0000000000000035 ATOM_CDR
        sectorlisp.S:30     *ABS*:000000000000003d ATOM_CONS
        sectorlisp.S:31     *ABS*:0000000000000047 ATOM_EQ
        sectorlisp.S:33     *ABS*:0000000000004000 q.token
        sectorlisp.S:34     *ABS*:0000000000004080 q.str
        sectorlisp.S:35     *ABS*:0000000000007c00 boot
        sectorlisp.S:42     .text:0000000000000000 _start
        sectorlisp.S:48     .text:0000000000000027 .init
        sectorlisp.S:44     .text:0000000000000002 kSymbols
        sectorlisp.S:49     .text:000000000000002c _begin
        sectorlisp.S:62     .text:0000000000000040 main
        sectorlisp.S:76     .text:0000000000000063 GetToken
        sectorlisp.S:95     .text:0000000000000083 GetObject
        sectorlisp.S:273    .text:00000000000001a5 Eval
        sectorlisp.S:125    .text:00000000000000b5 PrintObject
        sectorlisp.S:164    .text:00000000000000f4 PutChar
        sectorlisp.S:159    .text:00000000000000f0 GetChar
        sectorlisp.S:177    .text:0000000000000103 GetList
        sectorlisp.S:99     .text:000000000000008a .Intern
        sectorlisp.S:123    .text:00000000000000b4 .ret
        sectorlisp.S:138    .text:00000000000000ca .PrintList
        sectorlisp.S:129    .text:00000000000000ba .PrintAtom
        sectorlisp.S:132    .text:00000000000000c0 .PrintString
        sectorlisp.S:154    .text:00000000000000ea .PutObject
        sectorlisp.S:256    .text:0000000000000188 .retF
        sectorlisp.S:202    .text:000000000000012b Cons
        sectorlisp.S:189    .text:0000000000000115 Evlis
        sectorlisp.S:214    .text:0000000000000138 Pairlis
        sectorlisp.S:231    .text:0000000000000156 Apply
        sectorlisp.S:241    .text:0000000000000168 .switch
        sectorlisp.S:234    .text:000000000000015b .lambda
        sectorlisp.S:304    .text:00000000000001de .EvCadr
        sectorlisp.S:266    .text:000000000000019c .dflt1
        sectorlisp.S:244    .text:000000000000016f .ifCar
        sectorlisp.S:248    .text:0000000000000176 .ifCdr
        sectorlisp.S:252    .text:000000000000017e .ifAtom
        sectorlisp.S:258    .text:000000000000018c .ifCons
        sectorlisp.S:264    .text:0000000000000199 .retT
        sectorlisp.S:262    .text:0000000000000195 .isEq
        sectorlisp.S:307    .text:00000000000001e3 Assoc
        sectorlisp.S:288    .text:00000000000001c1 Cadr
        sectorlisp.S:292    .text:00000000000001c7 Evcon
        sectorlisp.S:320    .text:00000000000001fb .sig

NO UNDEFINED SYMBOLS
```