```
-rw-------. 1 257 Nov 23  2010 interp.c
-rwx------. 1  32 Apr  1  2010 interp.env
-rw-------. 1  90 Nov 23  2010 interp.out
-rwx------. 1  50 Nov 23  2010 interp.run
-rwx------. 1  85 Nov 23  2010 interp.sh

================================================================
File: interp.c
----------------------------------------------------------------
     1  // $Id: interp.c,v 341.3 2010-11-23 18:40:08-08 - - $
     2
     3  #include <stdio.h>
     4  #include <stdlib.h>
     5
     6  int main (int argc, char **argv) {
     7     for (int argi = 0; argi < argc; ++argi) {
     8        printf ("argv[%d]=%s\n", argi, argv[argi]) ;
     9     }
    10     return EXIT_SUCCESS;
    11  }
================================================================


================================================================
File: interp.sh
----------------------------------------------------------------
     1  #!/bin/sh
     2  # $Id: interp.sh,v 1.1 2010-11-23 18:40:59-08 - - $
     3  interp.run foo bar baz
================================================================


================================================================
File: interp.env
----------------------------------------------------------------
     1  #!/usr/bin/env interp -foo -bar
================================================================


================================================================
File: interp.out
----------------------------------------------------------------
     1  argv[0]=interp
     2  argv[1]=-foo -bar
     3  argv[2]=./interp.run
     4  argv[3]=foo
     5  argv[4]=bar
     6  argv[5]=baz
================================================================


================================================================
File: interp.run
----------------------------------------------------------------
     1  #!interp -foo -bar
     2  some line 2
     3  last line in file.
================================================================

```

```
 1: /* $Id: goto-sm.c,v 341.2 2012-11-16 20:49:45-08 - - $ */
 2: /*
 3: * tos of stack is cached in a register.
 4: * *sp is actually second from top of stack.
 5: *
 6: * gcc specific coding:
 7: * In order to make the core interpreter loop as fast as possible,
 8: * any tricks are good tricks, regardless of their so-called software
 9: * engineering quality.  The program is simple and regular, so the
10: * tricks hopefully will not lead to excessive obfuscation.
11: *
12: * -- Using goto *SW[*ip++] instead of a switch statement causes gcc
13: *    to omit the bounds check it must normally do with a switch.
14: *
15: */
16:
17: #include <inttypes.h>
18:
19: typedef uint8_t   ubyte;
20: typedef int8_t    sbyte;
21: typedef uint32_t  uword;
22: typedef int32_t   sword;
23: typedef void     *ugoto;
24:
25: enum Opcode{
26:    LDC0, LDCP, LDCN, LDC2, LDC4,
27:    LDL , LDL2, STL , STL2, SKL , SKL2,
28:    ADD , SUB , MUL , DIV , REM ,
29:    AND , OR  , XOR , SLL , SRL , SRA ,
30:    NEG , POS , COM ,
31:    EQ  , NE  , LT  , LE  , GT  , GE  ,
32:    JMP ,
33:    JZ  , JNZ ,
34:    JEQ , JNE , JLT , JLE , JGT , JGE ,
35: };
36:
37: void interp(){
38:    static ugoto SWITCH[] = {
39:       &&_LDC0, &&_LDCP, &&_LDCN, &&_LDC2, &&_LDC4,
40:       &&_LDL , &&_LDL2, &&_STL , &&_STL2, &&_SKL , &&_SKL2,
41:       &&_ADD , &&_SUB , &&_MUL , &&_DIV , &&_REM ,
42:       &&_AND , &&_OR  , &&_XOR , &&_SLL , &&_SRL , &&_SRA ,
43:       &&_NEG , &&_POS , &&_COM ,
44:       &&_EQ  , &&_NE  , &&_LT  , &&_LE  , &&_GT  , &&_GE  ,
45:       &&_JMP , &&_JZ  , &&_JNZ ,
46:    };
47:    register ubyte *ip;
48:    register sword *sp;
49:    register sword *fp;
50:    register sword tos; /* cache for *sp */
51:    register ugoto *SW = SWITCH;
52:
53:    #define _IP0     register sword w = (sbyte)(ip[0]);
54:    #define _IP(N)   w = w<<8 | ip[N];
55:    #define UIP1     ( *ip++ )
56:    #define SIP2     ({ _IP0; _IP(1); ip+=2; w; })
57:    #define SIP4     ({ _IP0; _IP(1); _IP(2); _IP(3); ip+=4; w; })
58:    #define FETCH    { goto Fetch; }
59:
```

```
 60:
 61:    Fetch: goto *SW[UIP1];
 62:
 63:    _LDC0: *sp++ = tos; tos = 0;                    FETCH;
 64:    _LDCP: *sp++ = tos; tos = UIP1;                 FETCH;
 65:    _LDCN: *sp++ = tos; tos = UIP1 - 256;           FETCH;
 66:    _LDC2: *sp++ = tos; tos = SIP2;                 FETCH;
 67:    _LDC4: *sp++ = tos; tos = SIP4;                 FETCH;
 68:
 69:    _LDL : *sp++ = tos; tos = fp[UIP1];             FETCH;
 70:    _LDL2: *sp++ = tos; tos = fp[SIP2];             FETCH;
 71:    _STL : fp[UIP1] = tos; tos = *--sp;             FETCH;
 72:    _STL2: fp[SIP2] = tos; tos = *--sp;             FETCH;
 73:    _SKL : fp[UIP1] = tos;                          FETCH;
 74:    _SKL2: fp[SIP2] = tos;                          FETCH;
 75:
 76:    _ADD : tos = *--sp + tos;                       FETCH;
 77:    _SUB : tos = *--sp - tos;                       FETCH;
 78:    _MUL : tos = *--sp * tos;                       FETCH;
 79:    _DIV : tos = *--sp / tos;                       FETCH;
 80:    _REM : tos = *--sp % tos;                       FETCH;
 81:
 82:    _AND : tos = *--sp & tos;                       FETCH;
 83:    _OR  : tos = *--sp | tos;                       FETCH;
 84:    _XOR : tos = *--sp ^ tos;                       FETCH;
 85:    _SLL : tos = *--sp << tos;                      FETCH;
 86:    _SRL : tos = (uword)(*--sp) >> tos;             FETCH;
 87:    _SRA : tos = *--sp >> tos;                      FETCH;
 88:
 89:    _NEG : tos = - tos;                             FETCH;
 90:    _POS : tos = + tos;                             FETCH;
 91:    _COM : tos = ~ tos;                             FETCH;
 92:
 93:    _EQ  : tos = *--sp == tos;                      FETCH;
 94:    _NE  : tos = *--sp != tos;                      FETCH;
 95:    _LT  : tos = *--sp <  tos;                      FETCH;
 96:    _LE  : tos = *--sp <= tos;                      FETCH;
 97:    _GT  : tos = *--sp >  tos;                      FETCH;
 98:    _GE  : tos = *--sp >= tos;                      FETCH;
 99:
100:    _JMP :                ip += SIP2;               FETCH;
101:    _JZ  : if( tos == 0 ) ip += SIP2; tos = *--sp;  FETCH;
102:    _JNZ : if( tos != 0 ) ip += SIP2; tos = *--sp;  FETCH;
103:
104: }
```