Viši programski jezici i asembler (primjer C-- i asembler za Z80)

gets.c

```
/* gets */
gets(s)          /* CITA string */
char s[];
{
  int  i;
        s[0]='x';
        for (i=0 ; (s[i]=getchar())!='\n' ; i++ )
            ;
        s[i]='\0';
}
```

---

puts.c

```
/* puts */
puts(s)          /* Ispis STRINGA */
char s[];
{
int i;
        for (i=0;s[i];i++) putchar(s[i]);
        putchar('\n');
}
```

---

getchar.c

```
/* getchar */
getchar() /*unbuffered single character CONSOLE input */
{
        return(getc(CONIN));
}
```

---

putchar.c

```
/* putchar */
putchar(c) /* unbuffered single character CONSOLE output */
char c;
{
        putc(c,CONOUT);
}
```

---

getc.c

```
/* getc */
getc(fd) /* unbuffered single character input */
int fd;
{
        char c;
        read(fd, &c, 1);
        return(c);
}
```

---

putc.c

```
/* putc */
putc(c,fd) /* unbuffered single character output */
char c; int fd;
{
        write(fd, &c, 1);
}
```

---

read.c

```
/* read */
/*
** Read from fd       (M. Zagar, 1987-10-22)
**
** Entry: fd  = file descriptor
**        buf = address of the target buffer
**         n   = number of bytes to read
** Exit:  returns a count of the bytes actually read
** Use:   rd(fd) interface in "poc_maza.xx"
*/

read(fd, buf, n)
int fd, n;
char *buf;
{
   int cnt; cnt = 0;
   while(n--) {
        *buf++ = rd(fd);
        ++cnt;
   }
   return (cnt);
}
```

---

write.c

```
/* write */
/*
** Write  to fd       (M. Zagar, 1987-10-22)
**
** Entry: fd  = file descriptor
**        buf = address of the source buffer
**         n   = number of bytes to write
** Exit:  returns a count of the bytes actually written or
**        -1 if an error occured
** Use:   wr(*buff++,fd) interface in "poc_maza.xx"
*/
write(fd, buf, n)
int fd, n;
char *buf;
{
   int cnt; cnt = 0;
```

```
   while(n--) {
        wr(*buf++, fd);
        cnt++;
        }
   return (cnt);
}
--------------------
#asm
        .z80
        cseg
RDCHR       EQU   0D7H  ; MAZA-Z80 con. read
WRCHR       EQU   0DCH ; MAZA-Z80 con. write
CMD1 EQU    020H   ; warm start MONIT.
        LD      SP,06000H ; STACK beginning
        CALL  MAIN ; C progr. entry
        JP      CMD1 ; return to MONITOR
#endasm

#define CONIN       0       /* CONSOLE input */
#define CONOUT      1       /* CONSOLE output*/

/* rd */

rd(fd)
int fd;
{
        fd;
#asm
        LD E,L
        CALL RDCHR
        LD L,A
#endasm
}
-------------------
/* wr */
wr(ch, fd)
char ch;
int  fd;
{
        ch;
#asm
        LD D,L
#endasm
        fd;
#asm
        LD E,L
        CALL WRCHR
#endasm
}
;
```

```
;-----call.asm:C-- arithmetic and logic library (Z80)
;
_call:: EXT    _end
;
CCDCAL::
        JP      (HL)
;
CCDDGC::
        ADD   HL,DE
        JR      CCGCHAR
;
CCDSGC::
        INC     HL
        INC     HL
        ADD   HL,SP
;
;FETCH A SINGLE BYTE FROM THE ADDRESS IN HL AND SIGN INTO HL
CCGCHAR::
        LD      A,(HL)
;
;PUT THE ACCUM INTO HL AND SIGN EXTEND THROUGH H
CCARGC::
CCSXT::
        LD      L,A
        RLCA
        SBC   A,A
        LD      H,A
        RET
;
CCDDGI::
        ADD   HL,DE
        JR      CCGINT
;
CCDSGI::
        INC     HL
        INC     HL
        ADD   HL,SP
;
;FETCH A FULL 16-BIT INTEGER FROM THE ADDRESS IN HL INTO HL
CCGINT::
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        RET
;
CCDECC::
        INC     HL
        INC     HL
        ADD   HL,SP
```

```
        LD      D,H
        LD      E,L
        CALL    CCGCHAR
        DEC     HL
        LD      A,L
        LD      (DE),A
        RET
;
CCINCC::
        INC     HL
        INC     HL
        ADD     HL,SP
        LD      D,H
        LD      E,L
        CALL    CCGCHAR
        INC     HL
        LD      A,L
        LD      (DE),A
        RET
;
CDPDPC::
        ADD     HL,DE
CCPDPC::
        POP     BC              ;;RET ADDR
        POP     DE
        PUSH    BC
;
;STORE A SINGLE BYTE FROM HL AT THE ADDRESS IN DE
CCPCHAR::
PCHAR::     LD      A,L
        LD      (DE),A
        RET
;
CCDECI::
        INC     HL
        INC     HL
        ADD     HL,SP
        LD      D,H
        LD      E,L
        CALL    CCGINT
        DEC     HL
        JR      CCPINT
;
CCINCI::
        INC     HL
        INC     HL
        ADD     HL,SP
        LD      D,H
        LD      E,L
        CALL    CCGINT
```

```
        INC     HL
        JR      CCPINT
;
CDPDPI::
        ADD     HL,DE
CCPDPI::
        POP     BC              ;;RET ADDR
        POP     DE
        PUSH    BC
;
;STORE A 16-BIT INTEGER IN HL AT THE ADDRESS IN DE
CCPINT::
PINT:: LD       A,L
        LD      (DE),A
        INC     DE
        LD      A,H
        LD      (DE),A
        RET
;
;INCLUSIVE "OR" HL AND DE INTO HL
CCOR::
        LD      A,L
        OR      E
        LD      L,A
        LD      A,H
        OR      D
        LD      H,A
        RET
;
;EXCLUSIVE "OR" HL AND DE INTO HL
CCXOR::
        LD      A,L
        XOR     E
        LD      L,A
        LD      A,H
        XOR     D
        LD      H,A
        RET
;
;"AND" HL AND DE INTO HL
CCAND::
        LD      A,L
        AND     E
        LD      L,A
        LD      A,H
        AND     D
        LD      H,A
        RET
;
;IN ALL THE FOLLOWING COMPARE ROUTINES, HL IS SET TO 1 IF THE
```

```
;  CONDITION IS TRUE, OTHERWISE IT IS SET TO 0 (ZERO).
;
;TEST IF HL = DE
;
CCEQ::
        CALL  CCCMP
        RET   Z
        DEC   HL
        RET
;
;TEST IF DE != HL
CCNE::
        CALL  CCCMP
        RET   NZ
        DEC   HL
        RET
;
;TEST IF DE > HL (SIGNED)
CCGT::
        EX    DE,HL
        CALL  CCCMP
        RET   C
        DEC   HL
        RET
;
;TEST IF DE <= HL (SIGNED)
CCLE::
        CALL  CCCMP
        RET   Z
        RET   C
        DEC   HL
        RET
;
;TEST IF DE >= HL (SIGNED)
CCGE::
        CALL  CCCMP
        RET   NC
        DEC   HL
        RET
;
;TEST IF DE < HL (SIGNED)
CCLT::
      CALL CCCMP
      RET  C
      DEC HL
      RET
;
;COMMON ROUTINE TO PERFORM A SIGNED COMPARE OF DE AND HL
; THIS ROUTINE PERFORMS DE - HL AND SETS THE CONDITIONS:
; CARRY REFLECTS SIGN OF DIFFERENCE (SET MEANS DE < HL)
```

```
; ZERO/NON-ZERO SET ACCORDING TO EQUALITY.
CCCMP::
        LD      A,H             ;;INVERT SIGN OF HL
        XOR     80H
        LD      H,A
        LD      A,D             ;;INVERT SIGN OF DE
        XOR     80H
        CP      H               ;;COMPARE MSBS
        JR      NZ,CCCMP1       ;;DONE IF NEQ
        LD      A,E             ;;COMPARE LSBL
        CP      L
CCCMP1::    LD      HL,1            ;;PRESET TRUE COND
        RET
;
;TEST IF DE >= HL (UNSIGNED)
CCUGE::
        CALL    CCUCMP
        RET     NC
        DEC     HL
        RET
;
;TEST IF DE < HL (UNSIGNED)
CCULT::
        CALL    CCUCMP
        RET     C
        DEC     HL
        RET
;
;TEST IF DE > HL (UNSIGNED)
CCUGT::
        EX      DE,HL
        CALL    CCUCMP
        RET     C
        DEC     HL
        RET
;
;TEST IF DE <= HL (UNSIGNED)
CCULE::
        CALL    CCUCMP
        RET     Z
        RET     C
        DEC     HL
        RET
;
;COMMON ROUTINE TO PERFORM UNSIGNED COMPARE
; CARRY SET IF DE < HL
; ZERO/NONZERO SET ACCORDINGLY
CCUCMP::
        LD      A,D
        CP      H
```

```
        JR      NZ,UCMP1
        LD      A,E
        CP      L
UCMP1:  LD      HL,1
        RET
;
;SHIFT DE ARITHMETICALLY BY HL AND RETURN IN HL
CCASR::
        EX      DE,HL
        DEC     E
        RET     M
        LD      A,H
        RLA
        LD      A,H
        RRA
        LD      H,A
        LD      A,L
        RRA
        LD      L,A
        JR      CCASR+1
;
;SHIFT DE ARITHMETICALLY LEFT BY HL AND RETURN IN HL
CCASL::
        EX      DE,HL
        DEC     E
        RET     M
        ADD     HL,HL
        JR      CCASL+1
;
;SUBTRACT HL FROM DE AND RETURN IN HL
CCSUB::
        LD      A,E
        SUB     L
        LD      L,A
        LD      A,D
        SBC     A,H
        LD      H,A
        RET
;
;FORM THE TWO'S COMPLEMENT OF HL
CCNEG::
        CALL    CCCOM
        INC     HL
        RET
;
;FORM THE ONE'S COMPLEMENT OF HL
CCCOM::
        LD      A,H
        CPL
        LD      H,A
```

```
        LD      A,L
        CPL
        LD      L,A
        RET
;
;MULTIPLY DE BY HL AND RETURN IN HL (SIGNED MULTIPLY)
CCMULT::
MULT::  LD      B,H
        LD      C,L
        LD      HL,0
MULT1:  LD      A,C
        RRCA
        JR      NC,MULT2
        ADD     HL,DE
MULT2:  XOR     A
        LD      A,B
        RRA
        LD      B,A
        LD      A,C
        RRA
        LD      C,A
        OR      B
        RET     Z
        XOR     A
        LD      A,E
        RLA
        LD      E,A
        LD      A,D
        RLA
        LD      D,A
        OR      E
        RET     Z
        JR      MULT1
;
;DIVIDE DE BY HL AND RETURN QUOTIENT IN HL,
;REMAINDER IN DE (SIGNED DIVIDE)
CCDIV::
DIV::   LD      B,H
        LD      C,L
        LD      A,D
        XOR     B
        PUSH    AF
        LD      A,D
        OR      A
        CALL    M,CCDENEG
        LD      A,B
        OR      A
        CALL    M,CCBCNEG
        LD      A,16
        PUSH    AF
```

```
        EX      DE,HL
        LD      DE,0
CCDIV1: ADD     HL,HL
        CALL    CCRDEL
        JR      Z,CCDIV2
        CALL    CCCMPBCDE
        JP      M,CCDIV2
        LD      A,L
        OR      1
        LD      L,A
        LD      A,E
        SUB     C
        LD      E,A
        LD      A,D
        SBC     A,B
        LD      D,A
CCDIV2: POP     AF
        DEC     A
        JR      Z,CCDIV3
        PUSH    AF
        JR      CCDIV1
CCDIV3: POP     AF
        RET     P
        CALL    CCDENEG
        EX      DE,HL
        CALL    CCDENEG
        EX      DE,HL
        RET
;
;NEGATE THE INTEGER IN DE (INTERNAL ROUTINE)
CCDENEG::LD     A,D
        CPL
        LD      D,A
        LD      A,E
        CPL
        LD      E,A
        INC     DE
        RET
;
;NEGATE THE INTEGER IN BC (INTERNAL ROUTINE)
CCBCNEG::LD     A,B
        CPL
        LD      B,A
        LD      A,C
        CPL
        LD      C,A
        INC     BC
        RET
;
;ROTATE DE LEFT ONE BIT (INTERNAL ROUTINE)
```

```
CCRDEL::    LD      A,E
        RLA
        LD      E,A
        LD      A,D
        RLA
        LD      D,A
        OR      E
        RET
;
;COMPARE BC TO DE (INTERNAL ROUTINE)
CCCMPBCDE::LD   A,E
        SUB     C
        LD      A,D
        SBC     A,B
        RET
;
;LOGICAL NEGATION
CCLNEG::
        LD      A,H
        OR      L
        JR      NZ,$+6
        LD      L,1
        RET
        LD      HL,0
        RET
;
;EXECUTE "SWITCH" STATEMENT
;
; HL = SWITCH VALUE
;(SP)->SWITCH TABLE
;       DW ADDR1, VALUE1
;       DW ADDR2, VALUE2
;       ...
;       DW 0
;    [JMP default]
;       continuation
;
CCSWITCH::
        EX      DE,HL           ;;DE = SWITCH VALUE
        POP     HL              ;;HL ->SWITCH TABLE
SWLOOP:     LD      C,(HL)
        INC     HL
        LD      B,(HL)          ;;BC-> CASE ADDR, ELSE 0
        INC     HL
        LD      A,B
        OR      C
        JR      Z,SWEND             ;;DEFAULT OR CONTINUATION CODE
        LD      A,(HL)
        INC     HL
        CP      E
```

```
        LD      A,(HL)
        INC     HL
        JR      NZ,SWLOOP
        CP      D
        JR      NZ,SWLOOP
        LD      H,B             ;;CASE MATCHED
        LD      L,C
SWEND:  JP      (HL)
;
;
        END
```

---

primjer.c

```
int  glmemi;
char glmemc;
char ch[]="mnopr";
main(){
   int  lmemi, x, y, z, w[2];
   char lmemc, a, b, c;
   int  *plmem;

   x=1; y=2;
   glmemi=1;
   z=x+y;
   w[1] = z;
   plmem=5;
   *plmem=6;
   lmemi= &plmem;
   a='A'; b='B';
   c=a+b;
   a=ch[0];

   func(a,b,x);
   x++;
   y--;
   z+=y;
   !w[2];

   if (x) x=5;
   if (x) x=5;
   else   x=6;

   if (x != 1) x=5;

   if (x != 0) x=5;
}
func(d,e,n) char d,e; int n;{
   n=d*e;
   return(n);
}
```

```
% ccz-- -l3 primjer.c
/* primjerZ80.mac */
;int glmemi;
glmemi::
     DW 0
;char glmemc;
glmemc::
     DB 0
;char ch[]="mnopr";
ch::
     DB 109,110,111,112,114,0
;
;main(){
CC1:
main::
;    int lmemi, x, y, z, w[2]; // 16,14,12,10,8,6
;    char lmemc, a, b, c;      // 5,4,3,2
;    int *plmem;               // 0
;
;    x=1; y=2;
     LD HL,-18
     ADD HL,SP
     LD SP,HL
     LD HL,14
     ADD HL,SP
     EX DE,HL;;
     LD HL,1
     CALL CCPINT##
     LD HL,12
     ADD HL,SP
     EX DE,HL;;
     LD HL,2
     CALL CCPINT##
;    glmemi=1;
     LD HL,1
     LD (glmemi),HL
;    z=x+y;
     LD HL,10
     ADD HL,SP
     PUSH HL
     LD HL,16
     ADD HL,SP
     CALL CCGINT##
     PUSH HL
     LD HL,16
     ADD HL,SP
     CALL CCGINT##
     POP DE
     ADD HL,DE
     POP DE
     CALL CCPINT##
```

```
;    w[1] = z;
     LD HL,6
     ADD HL,SP
     LD DE,2
     ADD HL,DE
     PUSH HL
     LD HL,12
     ADD HL,SP
     CALL CCGINT##
     POP DE
     CALL CCPINT##
;    plmem=5;
     LD HL,0
     ADD HL,SP
     PUSH HL
     LD HL,5
     POP DE
     CALL CCPINT##
;    *plmem=6;
     POP HL
     PUSH HL
     PUSH HL
     LD HL,6
     POP DE
     CALL CCPINT##
;    lmemi= &plmem;
     LD HL,16
     ADD HL,SP
     EX DE,HL;;
     LD HL,0
     ADD HL,SP
     CALL CCPINT##
;    a='A'; b='B';
     LD HL,4
     ADD HL,SP
     EX DE,HL;;
     LD HL,65
     LD A,L
     LD (DE),A
     LD HL,3
     ADD HL,SP
     EX DE,HL;;
     LD HL,66
     LD A,L
     LD (DE),A
;    c=a+b;
     LD HL,2
     ADD HL,SP
     PUSH HL
     LD HL,6
     ADD HL,SP
```

```
        CALL CCGCHAR##
        PUSH HL
        LD HL,7
        ADD HL,SP
        CALL CCGCHAR##
        POP DE
        ADD HL,DE
        POP DE
        LD A,L
        LD (DE),A
;       a=ch[0];
        LD HL,4
        ADD HL,SP
        PUSH HL
        LD HL,ch
        CALL CCGCHAR##
        POP DE
        LD A,L
        LD (DE),A
;
;       func(a,b,x);
        LD HL,4
        ADD HL,SP
        CALL CCGCHAR##
        PUSH HL
        LD HL,5
        ADD HL,SP
        CALL CCGCHAR##
        PUSH HL
        LD HL,18
        ADD HL,SP
        CALL CCGINT##
        PUSH HL
        CALL func
        POP BC
        POP BC
        POP BC
;       x++;
        LD HL,14
        ADD HL,SP
        PUSH HL
        CALL CCGINT##
        INC HL
        POP DE
        CALL CCPINT##
        DEC HL
;       y--;
        LD HL,12
        ADD HL,SP
        PUSH HL
        CALL CCGINT##
```

```
        DEC HL
        POP DE
        CALL CCPINT##
        INC HL
;       z+=y;
        LD HL,10
        ADD HL,SP
        PUSH HL
        CALL CCGINT##
        PUSH HL
        LD HL,16
        ADD HL,SP
        CALL CCGINT##
        POP DE
        ADD HL,DE
        POP DE
        CALL CCPINT##
;       !w[2];
        LD HL,6
        ADD HL,SP
        LD DE,4
        ADD HL,DE
        CALL CCGINT##
        CALL CCLNEG##


;
;       if (x) x=5;
        LD HL,14
        ADD HL,SP
        CALL CCGINT##
        LD A,H
        OR L
        JP Z,CC3
        LD HL,14
        ADD HL,SP
        PUSH HL
        LD HL,5
        POP DE
        CALL CCPINT##
;       if (x) x=5;
CC3:
        LD HL,14
        ADD HL,SP
        CALL CCGINT##
        LD A,H
        OR L
        JP Z,CC4
        LD HL,14
        ADD HL,SP
        PUSH HL
```

```
      LD HL,5
      POP DE
      CALL CCPINT##
;    else   x=6;
      JP CC5
CC4:
      LD HL,14
      ADD HL,SP
      PUSH HL
      LD HL,6
      POP DE
      CALL CCPINT##
CC5:
;
;    if (x != 1) x=5;
      LD HL,14
      ADD HL,SP
      CALL CCGINT##
      PUSH HL
      LD HL,1
      POP DE
      CALL CCNE##
      LD A,H
      OR L
      JP Z,CC6
      LD HL,14
      ADD HL,SP
      PUSH HL
      LD HL,5
      POP DE
      CALL CCPINT##
;
;    if (x != 0) x=5;
CC6:
      LD HL,14
      ADD HL,SP
      CALL CCGINT##
      LD A,H
      OR L
      JP Z,CC7
      LD HL,14
      ADD HL,SP
      EX DE,HL;;
      LD HL,5
      CALL CCPINT##
;}
CC7:
      LD HL,18
      ADD HL,SP
      LD SP,HL
      RET
```

```
;func(d,e,n) char d,e; int n;{
func::
;    n=d*e;
      LD HL,2
      ADD HL,SP
      PUSH HL
      LD HL,8
      ADD HL,SP
      CALL CCGCHAR##
      PUSH HL
      LD HL,8
      ADD HL,SP
      CALL CCGCHAR##
      POP DE
      CALL CCMULT##
      POP DE
      CALL CCPINT##
;    return(n);
      POP BC
      POP HL
      PUSH HL
      PUSH BC
      RET
;}
                ; Code produced by ccz-- (V2.0) cross compiler
      END
------------------------------
```

primjerM68K.asm

```
;int glmemi;
glmemi::
      DC.W 0
;char glmemc;
glmemc::
      DC.B 0
;char ch[]="mnopr";
ch::
      DC.B 109,110,111,112,114,0
;
;main(){

CC1:
main::
;    int lmemi, x, y, z, w[2];
;    char lmemc, a, b, c;
;    int *plmem;
;
;    x=1; y=2;
      MOVE.W #-22,D3
      ADD.W A7,D3
      MOVE.W D3,A7
```

```
      MOVE.W #18,D3
      ADD.W A7,D3
      EXG D3,D2;;
      MOVE.W #1,D3
      JSR CCPINT##
      MOVE.W #16,D3
      ADD.W A7,D3
      EXG D3,D2;;
      MOVE.W #2,D3
      JSR CCPINT##
;     z=x+y;
      MOVE.W #14,D3
      ADD.W A7,D3
      MOVE.W D3,-(A7)
      MOVE.W #20,D3
      ADD.W A7,D3
      JSR CCGINT##
      MOVE.W D3,-(A7)
      MOVE.W #20,D3
      ADD.W A7,D3
      JSR CCGINT##
      MOVE.W (A7)+,D2
      ADD.W D2,D3
      MOVE.W (A7)+,D2
      JSR CCPINT##
      .
      .
                ; Code produced by ccm-- (V2.0) cross compiler
      END
---------------------------
```

%cc -S primjer.c

```
/*primjer SPARC.s */
      .section  ".text",#alloc,#execinstr
      .align    8
      .skip     16

      ! block 0

      .global   main
      .type     main,2
main:
      save %sp,-136,%sp

      ! block 1
.L17:
! File primjer.c:
!     1    int glmemi;
!     2    char glmemc;
!     3    char ch[]="mnopr";
!     4
```

```
!     5    main(){
!     6       int lmemi, x, y, z, w[2];
!     7       char lmemc, a, b, c;
!     8       int *plmem;
!     9
!10   x=1; y=2;
      mov  1,%l0
      st   %l0,[%fp-12]
      mov  2,%l0
      st   %l0,[%fp-16]

!11   z=x+y;
      ld   [%fp-12],%l0
      ld   [%fp-16],%l1
      add  %l0,%l1,%l0
      st   %l0,[%fp-20]

!    12       w[1] = z;

      ld   [%fp-20],%l0
      st   %l0,[%fp-24]

!    13       plmem=5;

      mov  5,%l0
      st   %l0,[%fp-36]

!    14       *plmem=6;

      mov  6,%l1
      ld   [%fp-36],%l0
      st   %l1,[%l0+0]

!    15       lmemi= &plmem;

      add  %fp,-36,%l0
      st   %l0,[%fp-8]

!    16       a='A'; b='B';

      mov  65,%l0
      stb  %l0,[%fp-30]
      mov  66,%l0
      stb  %l0,[%fp-31]

!    17       c=a+b;

      ldsb [%fp-30],%l0
      sll  %l0,24,%l0
      sra  %l0,24,%l2
      ldsb [%fp-31],%l0
```

```
      sll  %l0,24,%l0
      sra  %l0,24,%l1
      add  %l2,%l1,%l0
      stb  %l0,[%fp-32]

!   18       a=ch[0];

      sethi     %hi(ch),%l0
      or   %l0,%lo(ch),%l0
      ldsb [%l0+0],%l0
      stb  %l0,[%fp-30]

!   19
!   20       func(a,b,x);

      ldsb [%fp-30],%l0
      sll  %l0,24,%l0
      sra  %l0,24,%l1
      ldsb [%fp-31],%l0
      sll  %l0,24,%l0
      sra  %l0,24,%l0
      ld   [%fp-12],%l2
      mov  %l1,%o0
      mov  %l0,%o1
      mov  %l2,%o2
      call func
      nop

!   21       x++;

      ld   [%fp-12],%l0
      add  %l0,1,%l0
      st   %l0,[%fp-12]

!   22       y--;

      ld   [%fp-16],%l0
      sub  %l0,1,%l0
      st   %l0,[%fp-16]

!   23       z+=y;

      ld   [%fp-20],%l0
      ld   [%fp-16],%l1
      add  %l0,%l1,%l0
      st   %l0,[%fp-20]

!   24       !w[2];

      ld   [%fp-20],%l0
      cmp  %l0,%g0
```

```
     bne  .L19
     nop

     ! block 2
.L20:
     mov  1,%l0
     st   %l0,[%fp-40]
     ba   .L21
     nop

     ! block 3
.L19:
     mov  0,%l0
     st   %l0,[%fp-40]

     ! block 4
.L21:

!   26       if (x) x=5;

     ld   [%fp-12],%l0
     cmp  %l0,%g0
     be   .L22
     nop

     ! block 5
.L23:
     mov  5,%l0
     st   %l0,[%fp-12]

     ! block 6
.L22:

!   27       if (x) x=5;

     ld   [%fp-12],%l0
     cmp  %l0,%g0
     be   .L24
     nop

     ! block 7
.L25:
     mov  5,%l0
     st   %l0,[%fp-12]
     ba   .L26
     nop

     ! block 8
.L24:
!   28       else   x=6;
```

```
     mov  6,%l0
     st   %l0,[%fp-12]

     ! block 9
.L26:

!   30       if (x != 1) x=5;

     ld   [%fp-12],%l0
     cmp  %l0,1
     be   .L27
     nop

     ! block 10
.L28:
     mov  5,%l0
     st   %l0,[%fp-12]

     ! block 11
.L27:
!   32       if (x != 0) x=5;

     ld   [%fp-12],%l0
     cmp  %l0,0
     be   .L29
     nop

     ! block 12
.L30:
     mov  5,%l0
     st   %l0,[%fp-12]

     ! block 13
.L29:
.L16:
     jmp  %i7+8
     restore
     .size     main,(.-main)
     .align    8
     .align    8
     .skip     16

     ! block 0

     .global   func
     .type     func,2
func:
     save %sp,-104,%sp
     st   %i2,[%fp+76]
     st   %i1,[%fp+72]
     st   %i0,[%fp+68]
```

```
     ! block 1
.L33:

! File primjer.c:
!   34    }
!   35    func(d,e,n) char d,e; int n;{
!   36       n=d*e;
     ldsb [%fp+71],%l0
     sll  %l0,24,%l0
     sra  %l0,24,%l1
     ldsb [%fp+75],%l0
     sll  %l0,24,%l0
     sra  %l0,24,%l0
     mov  %l1,%o0
     mov  %l0,%o1
     call .mul
     nop
     mov  %o0,%l0
     st   %l0,[%fp+76]

!   37       return(n);
     ld   [%fp+76],%l0
     st   %l0,[%fp-4]
     ba   .L32
     nop

     ! block 2
.L32:
     ld   [%fp-4],%l0
     mov  %l0,%i0
     jmp  %i7+8
     restore
     .size     func,(.-func)
     .align    8

     .section  ".data",#alloc,#write
     .global   ch
ch:
     .ascii    "mnopr\000"
     .type     ch,#object
     .size     ch,6
     .common   glmemc,1,1
     .common   glmemi,4,4
     .file     "primjer.c"
     .xstabs   ".stab.index","Xa ; V=3.1
; R=WorkShop Compilers 4.2 30 Oct 1996 C 4.2",60,0,0,0
     .xstabs
     ".stab.index","/export/home/mario/nastava/OPIPI98/C
;/opt/SUNWspro/bin/../SC4.2/bin/cc         -S         primjer.c
-W0,-xp",52,0,0,0
     .xstabs   ".stab.index","main",42,0,0,0
```

```
      .ident    "acomp: WorkShop Compilers 4.2 30 Oct 1996 C4.2"
      .global   __fsr_init_value
__fsr_init_value = 0x0
```

VAX-785

```
      .
      .
;     x=1;
      movl $1,-8(fp)
;     y=2;
      movl $2,-12(fp)
;     z = x + y;
      addl3 -12(fp),-8(fp),r0
      movl r0,-16(fp)
      .
      .
```

CV - MOTOROLA 68020

```
      .
      .
;     x=1;
      movl #0x1,a6@(-0x8)
;     y=2;
      movl #0x2,a6@(-0xc)
;     z = x + y;
      movl a6@(-0x8),d0
      addl a6@(-0xc),d0
      movl d0, a6@(-0x10)
      .
      .
```
