```
===============================+===============================
factorial.c-
--------
// C-F10
// factorial function
int factorial(int n)
{
        int i, f;

        f = i = 1;
        while (i<=n) {
              f = f*i;
              i++;
        }

        return f;
}

void main()
{
        int n;
        n = input();
        n = factorial(n);
        output(n);
        outnl();
}


===============================+===============================
factorial.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  factorial.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
 26:  OUTNL  3,3,3      Output a newline
 27:     LD  3,-1(1)    Load return address
 28:     LD  1,0(1)     Adjust fp
 29:    LDA  7,0(3)     Return
* END of function outnl
* BEGIN function factorial
 30:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 31:    LDC  3,1(6)     Load constant
 32:     ST  3,-3(1)    Store variable i
 33:     ST  3,-4(1)    Store variable f
* WHILE
 34:     LD  3,-3(1)    Load variable i
 35:     ST  3,-5(1)    Save left side
 36:     LD  3,-2(1)    Load variable n
 37:     LD  4,-5(1)    Load left into ac1
 38:    SUB  4,4,3      Op <=
 39:    LDC  3,1(6)     True case
 40:    JLE  4,1(7)     Jump if true
 41:    LDC  3,0(6)     False case
 42:    JGT  3,1(7)     Jump to while part
* DO
* BEGIN compound statement
* EXPRESSION STMT
 44:     LD  3,-4(1)    Load variable f
 45:     ST  3,-5(1)    Save left side
 46:     LD  3,-3(1)    Load variable i
 47:     LD  4,-5(1)    Load left into ac1
 48:    MUL  3,4,3      Op *
 49:     ST  3,-4(1)    Store variable f
* EXPRESSION STMT
 50:     LD  3,-3(1)    load lhs variable i
 51:    LDA  3,1(3)     increment value of i
 52:     ST  3,-3(1)    Store variable i
* END compound statement
 53:    LDA  7,-20(7)   go to beginning of loop
 43:    LDA  7,10(7)    No more loop
* ENDWHILE
* RETURN
 54:     LD  3,-4(1)    Load variable f
 55:    LDA  2,0(3)     Copy result to rt register
 56:     LD  3,-1(1)    Load return address
 57:     LD  1,0(1)     Adjust fp
 58:    LDA  7,0(3)     Return
* END compound statement
* Add standard closing in case there is no return statement
```

```
 59:    LDC  2,0(6)     Set return value to 0
 60:     LD  3,-1(1)    Load return address
 61:     LD  1,0(1)     Adjust fp
 62:    LDA  7,0(3)     Return
* END of function factorial
* BEGIN function main
 63:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 64:     ST  1,-3(1)    Store old fp in ghost frame
 65:    LDA  1,-3(1)    Load address of new frame
 66:    LDA  3,1(7)     Return address in ac
 67:    LDA  7,-67(7)   CALL input
 68:    LDA  3,0(2)     Save the result in ac
 69:     ST  3,-2(1)    Store variable n
* EXPRESSION STMT
 70:     ST  1,-3(1)    Store old fp in ghost frame
 71:     LD  3,-2(1)    Load variable n
 72:     ST  3,-5(1)    Store parameter
 73:    LDA  1,-3(1)    Load address of new frame
 74:    LDA  3,1(7)     Return address in ac
 75:    LDA  7,-46(7)   CALL factorial
 76:    LDA  3,0(2)     Save the result in ac
 77:     ST  3,-2(1)    Store variable n
* EXPRESSION STMT
 78:     ST  1,-3(1)    Store old fp in ghost frame
 79:     LD  3,-2(1)    Load variable n
 80:     ST  3,-5(1)    Store parameter
 81:    LDA  1,-3(1)    Load address of new frame
 82:    LDA  3,1(7)     Return address in ac
 83:    LDA  7,-78(7)   CALL output
 84:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
 85:     ST  1,-3(1)    Store old fp in ghost frame
 86:    LDA  1,-3(1)    Load address of new frame
 87:    LDA  3,1(7)     Return address in ac
 88:    LDA  7,-64(7)   CALL outnl
 89:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
 90:    LDC  2,0(6)     Set return value to 0
 91:     LD  3,-1(1)    Load return address
 92:     LD  1,0(1)     Adjust fp
 93:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,93(7)    Jump to init
* BEGIN Init
 94:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
* END init of global array sizes
 95:    LDA  1,0(0)     set first frame at end of globals
 96:     ST  1,0(1)     store old fp (point to self)
 97:    LDA  3,1(7)     Return address in ac
 98:    LDA  7,-36(7)   Jump to main
 99:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
ainit.c-
```

```
--------
// C-F10
int g[10];

void main() {
    int m[11];
    m[0] = 111;
    {
        int c[12];
        c[0] = 222;
        output(*g);
        output(*m);
        output(*c);
        outnl();
    }
    {
        int d[13];
        d[0] = 333;
        output(*g);
        output(*m);
        output(*d);
        outnl();
    }
    {
        static int s[14];
        s[0] = 444;
        output(*g);
        output(*m);
        output(*s);
        outnl();
    }
    output(*g);
    output(*m);
    outnl();
}
==============================+==============================
ainit.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  ainit.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
 26:  OUTNL  3,3,3      Output a newline
 27:     LD  3,-1(1)    Load return address
 28:     LD  1,0(1)     Adjust fp
 29:    LDA  7,0(3)     Return
* END of function outnl
* BEGIN function main
 30:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
 31:    LDC  3,11(6)    load size of array m
 32:     ST  3,-2(1)    save size of array m
* EXPRESSION STMT
 33:    LDC  3,0(6)     Load constant
 34:     ST  3,-41(1)   Save index
 35:    LDC  3,111(6)   Load constant
 36:     LD  4,-41(1)   Restore index
 37:    LDA  5,-3(1)    Load address of base of array m
 38:    SUB  5,5,4      Compute offset of value
 39:     ST  3,0(5)     Store variable m
* BEGIN compound statement
 40:    LDC  3,12(6)    load size of array c
 41:     ST  3,-14(1)   save size of array c
* EXPRESSION STMT
 42:    LDC  3,0(6)     Load constant
 43:     ST  3,-41(1)   Save index
 44:    LDC  3,222(6)   Load constant
 45:     LD  4,-41(1)   Restore index
 46:    LDA  5,-15(1)   Load address of base of array c
 47:    SUB  5,5,4      Compute offset of value
 48:     ST  3,0(5)     Store variable c
* EXPRESSION STMT
 49:     ST  1,-41(1)   Store old fp in ghost frame
 50:    LDA  3,-1(0)    Load address of base of array g
 51:     LD  3,1(3)     Load array size
 52:     ST  3,-43(1)   Store parameter
 53:    LDA  1,-41(1)   Load address of new frame
 54:    LDA  3,1(7)     Return address in ac
 55:    LDA  7,-50(7)   CALL output
 56:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
 57:     ST  1,-41(1)   Store old fp in ghost frame
```

```
  58:    LDA  3,-3(1)    Load address of base of array m
  59:     LD  3,1(3)     Load array size
  60:     ST  3,-43(1)   Store parameter
  61:    LDA  1,-41(1)   Load address of new frame
  62:    LDA  3,1(7)     Return address in ac
  63:    LDA  7,-58(7)   CALL output
  64:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  65:     ST  1,-41(1)   Store old fp in ghost frame
  66:    LDA  3,-15(1)   Load address of base of array c
  67:     LD  3,1(3)     Load array size
  68:     ST  3,-43(1)   Store parameter
  69:    LDA  1,-41(1)   Load address of new frame
  70:    LDA  3,1(7)     Return address in ac
  71:    LDA  7,-66(7)   CALL output
  72:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  73:     ST  1,-41(1)   Store old fp in ghost frame
  74:    LDA  1,-41(1)   Load address of new frame
  75:    LDA  3,1(7)     Return address in ac
  76:    LDA  7,-52(7)   CALL outnl
  77:    LDA  3,0(2)     Save the result in ac
 * END compound statement
 * BEGIN compound statement
  78:    LDC  3,13(6)    load size of array d
  79:     ST  3,-27(1)   save size of array d
 * EXPRESSION STMT
  80:    LDC  3,0(6)     Load constant
  81:     ST  3,-41(1)   Save index
  82:    LDC  3,333(6)   Load constant
  83:     LD  4,-41(1)   Restore index
  84:    LDA  5,-28(1)   Load address of base of array d
  85:    SUB  5,5,4      Compute offset of value
  86:     ST  3,0(5)     Store variable d
 * EXPRESSION STMT
  87:     ST  1,-41(1)   Store old fp in ghost frame
  88:    LDA  3,-1(0)    Load address of base of array g
  89:     LD  3,1(3)     Load array size
  90:     ST  3,-43(1)   Store parameter
  91:    LDA  1,-41(1)   Load address of new frame
  92:    LDA  3,1(7)     Return address in ac
  93:    LDA  7,-88(7)   CALL output
  94:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  95:     ST  1,-41(1)   Store old fp in ghost frame
  96:    LDA  3,-3(1)    Load address of base of array m
  97:     LD  3,1(3)     Load array size
  98:     ST  3,-43(1)   Store parameter
  99:    LDA  1,-41(1)   Load address of new frame
 100:    LDA  3,1(7)     Return address in ac
 101:    LDA  7,-96(7)   CALL output
 102:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
 103:     ST  1,-41(1)   Store old fp in ghost frame
 104:    LDA  3,-28(1)   Load address of base of array d
 105:     LD  3,1(3)     Load array size
 106:     ST  3,-43(1)   Store parameter
 107:    LDA  1,-41(1)   Load address of new frame
```

```
108:    LDA  3,1(7)     Return address in ac
109:    LDA  7,-104(7)  CALL output
110:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
111:     ST  1,-41(1)   Store old fp in ghost frame
112:    LDA  1,-41(1)   Load address of new frame
113:    LDA  3,1(7)     Return address in ac
114:    LDA  7,-90(7)   CALL outnl
115:    LDA  3,0(2)     Save the result in ac
* END compound statement
* BEGIN compound statement
116:    LDC  3,14(6)    load size of array s
117:     ST  3,-11(0)   save size of array s
* EXPRESSION STMT
118:    LDC  3,0(6)     Load constant
119:     ST  3,-41(1)   Save index
120:    LDC  3,444(6)   Load constant
121:     LD  4,-41(1)   Restore index
122:    LDA  5,-12(0)   Load address of base of array s
123:    SUB  5,5,4      Compute offset of value
124:     ST  3,0(5)     Store variable s
* EXPRESSION STMT
125:     ST  1,-41(1)   Store old fp in ghost frame
126:    LDA  3,-1(0)    Load address of base of array g
127:     LD  3,1(3)     Load array size
128:     ST  3,-43(1)   Store parameter
129:    LDA  1,-41(1)   Load address of new frame
130:    LDA  3,1(7)     Return address in ac
131:    LDA  7,-126(7)  CALL output
132:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
133:     ST  1,-41(1)   Store old fp in ghost frame
134:    LDA  3,-3(1)    Load address of base of array m
135:     LD  3,1(3)     Load array size
136:     ST  3,-43(1)   Store parameter
137:    LDA  1,-41(1)   Load address of new frame
138:    LDA  3,1(7)     Return address in ac
139:    LDA  7,-134(7)  CALL output
140:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
141:     ST  1,-41(1)   Store old fp in ghost frame
142:    LDA  3,-12(0)   Load address of base of array s
143:     LD  3,1(3)     Load array size
144:     ST  3,-43(1)   Store parameter
145:    LDA  1,-41(1)   Load address of new frame
146:    LDA  3,1(7)     Return address in ac
147:    LDA  7,-142(7)  CALL output
148:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
149:     ST  1,-41(1)   Store old fp in ghost frame
150:    LDA  1,-41(1)   Load address of new frame
151:    LDA  3,1(7)     Return address in ac
152:    LDA  7,-128(7)  CALL outnl
153:    LDA  3,0(2)     Save the result in ac
* END compound statement
* EXPRESSION STMT
154:     ST  1,-41(1)   Store old fp in ghost frame
155:    LDA  3,-1(0)    Load address of base of array g
```

```
156:     LD  3,1(3)     Load array size
157:     ST  3,-43(1)   Store parameter
158:    LDA  1,-41(1)   Load address of new frame
159:    LDA  3,1(7)     Return address in ac
160:    LDA  7,-155(7)  CALL output
161:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
162:     ST  1,-41(1)   Store old fp in ghost frame
163:    LDA  3,-3(1)    Load address of base of array m
164:     LD  3,1(3)     Load array size
165:     ST  3,-43(1)   Store parameter
166:    LDA  1,-41(1)   Load address of new frame
167:    LDA  3,1(7)     Return address in ac
168:    LDA  7,-163(7)  CALL output
169:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
170:     ST  1,-41(1)   Store old fp in ghost frame
171:    LDA  1,-41(1)   Load address of new frame
172:    LDA  3,1(7)     Return address in ac
173:    LDA  7,-149(7)  CALL outnl
174:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
175:    LDC  2,0(6)     Set return value to 0
176:     LD  3,-1(1)    Load return address
177:     LD  1,0(1)     Adjust fp
178:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,178(7)   Jump to init
* BEGIN Init
179:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
180:    LDC  3,10(6)    load size of array g
181:     ST  3,0(0)     save size of array g
* END init of global array sizes
182:    LDA  1,-26(0)   set first frame at end of globals
183:     ST  1,0(1)     store old fp (point to self)
184:    LDA  3,1(7)     Return address in ac
185:    LDA  7,-156(7)  Jump to main
186:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
var.c-
--------
// C-F10
//
int g, ga[10];

void cat(int x, xa[])
{
    output(x);
    output(xa[3]);
    output(*xa);
    outnl();
}

void dog(int x, xa[])
{
    output(x);
    output(xa[3]);
    outnl();

    x = 668;
    xa[1+2] = 669;
    output(x);
    output(xa[3]);
    outnl();

    g = 670;
    ga[2+1] = 671;
    output(g);
    output(ga[3]);
    outnl();

    cat(x, xa);
    cat(g, ga);
}

void fox()
{
    static int s, sa[11];

    sa[3] = 1000;
    s = 777;
    sa[3] -= s + (s = 999);
    output(s);
    output(sa[3]);
    outnl();
    cat(s, sa);
}

void main()
{
    int y, ya[12];

    y = 666;
    ya[1+2] = 667;
    output(y);
    output(ya[3]);
    outnl();

    dog(y, ya);

    output(y);
    output(ya[3]);
    outnl();

    fox();
}
==============================+===============================
var.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  var.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address 
  2:     IN  2,2,2      Grab int input 
  3:     LD  3,-1(1)    Load return address 
  4:     LD  1,0(1)     Adjust fp 
  5:    LDA  7,0(3)     Return 
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address 
  7:     LD  3,-2(1)    Load parameter 
  8:    OUT  3,3,3      Output integer 
  9:    LDC  2,0(6)     Set return to 0 
 10:     LD  3,-1(1)    Load return address 
 11:     LD  1,0(1)     Adjust fp 
 12:    LDA  7,0(3)     Return 
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address 
 14:    INB  2,2,2      Grab bool input 
 15:     LD  3,-1(1)    Load return address 
 16:     LD  1,0(1)     Adjust fp 
 17:    LDA  7,0(3)     Return 
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address 
 19:     LD  3,-2(1)    Load parameter 
 20:   OUTB  3,3,3      Output bool 
 21:    LDC  2,0(6)     Set return to 0 
 22:     LD  3,-1(1)    Load return address 
 23:     LD  1,0(1)     Adjust fp 
 24:    LDA  7,0(3)     Return 
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address 
 26:  OUTNL  3,3,3      Output a newline 
 27:     LD  3,-1(1)    Load return address 
 28:     LD  1,0(1)     Adjust fp 
 29:    LDA  7,0(3)     Return 
* END of function outnl
* BEGIN function cat
 30:     ST  3,-1(1)    Store return address. 
* BEGIN compound statement
* EXPRESSION STMT
 31:     ST  1,-4(1)    Store old fp in ghost frame 
 32:     LD  3,-2(1)    Load variable x
 33:     ST  3,-6(1)    Store parameter 
 34:    LDA  1,-4(1)    Load address of new frame 
 35:    LDA  3,1(7)     Return address in ac 
 36:    LDA  7,-31(7)   CALL output
 37:    LDA  3,0(2)     Save the result in ac 
* EXPRESSION STMT
 38:     ST  1,-4(1)    Store old fp in ghost frame 
 39:    LDC  3,3(6)     Load constant 
 40:     LD  4,-3(1)    Load address of base of array xa
 41:    SUB  3,4,3      Compute offset of value 
 42:     LD  3,0(3)     Load the value 
 43:     ST  3,-6(1)    Store parameter 
 44:    LDA  1,-4(1)    Load address of new frame 
```

```
  45:    LDA  3,1(7)     Return address in ac
  46:    LDA  7,-41(7)   CALL output
  47:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  48:     ST  1,-4(1)    Store old fp in ghost frame
  49:     LD  3,-3(1)    Load address of base of array xa
  50:     LD  3,1(3)     Load array size
  51:     ST  3,-6(1)    Store parameter
  52:    LDA  1,-4(1)    Load address of new frame
  53:    LDA  3,1(7)     Return address in ac
  54:    LDA  7,-49(7)   CALL output
  55:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  56:     ST  1,-4(1)    Store old fp in ghost frame
  57:    LDA  1,-4(1)    Load address of new frame
  58:    LDA  3,1(7)     Return address in ac
  59:    LDA  7,-35(7)   CALL outnl
  60:    LDA  3,0(2)     Save the result in ac
 * END compound statement
 * Add standard closing in case there is no return statement
  61:    LDC  2,0(6)     Set return value to 0
  62:     LD  3,-1(1)    Load return address
  63:     LD  1,0(1)     Adjust fp
  64:    LDA  7,0(3)     Return
 * END of function cat
 * BEGIN function dog
  65:     ST  3,-1(1)    Store return address.
 * BEGIN compound statement
 * EXPRESSION STMT
  66:     ST  1,-4(1)    Store old fp in ghost frame
  67:     LD  3,-2(1)    Load variable x
  68:     ST  3,-6(1)    Store parameter
  69:    LDA  1,-4(1)    Load address of new frame
  70:    LDA  3,1(7)     Return address in ac
  71:    LDA  7,-66(7)   CALL output
  72:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  73:     ST  1,-4(1)    Store old fp in ghost frame
  74:    LDC  3,3(6)     Load constant
  75:     LD  4,-3(1)    Load address of base of array xa
  76:    SUB  3,4,3      Compute offset of value
  77:     LD  3,0(3)     Load the value
  78:     ST  3,-6(1)    Store parameter
  79:    LDA  1,-4(1)    Load address of new frame
  80:    LDA  3,1(7)     Return address in ac
  81:    LDA  7,-76(7)   CALL output
  82:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  83:     ST  1,-4(1)    Store old fp in ghost frame
  84:    LDA  1,-4(1)    Load address of new frame
  85:    LDA  3,1(7)     Return address in ac
  86:    LDA  7,-62(7)   CALL outnl
  87:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
  88:    LDC  3,668(6)   Load constant
  89:     ST  3,-2(1)    Store variable x
 * EXPRESSION STMT
  90:    LDC  3,1(6)     Load constant
  91:     ST  3,-4(1)    Save left side
  92:    LDC  3,2(6)     Load constant
  93:     LD  4,-4(1)    Load left into ac1
  94:    ADD  3,4,3      Op +
  95:     ST  3,-4(1)    Save index
  96:    LDC  3,669(6)   Load constant
  97:     LD  4,-4(1)    Restore index
  98:     LD  5,-3(1)    Load address of base of array xa
  99:    SUB  5,5,4      Compute offset of value
 100:     ST  3,0(5)     Store variable xa
 * EXPRESSION STMT
 101:     ST  1,-4(1)    Store old fp in ghost frame
 102:     LD  3,-2(1)    Load variable x
 103:     ST  3,-6(1)    Store parameter
 104:    LDA  1,-4(1)    Load address of new frame
 105:    LDA  3,1(7)     Return address in ac
 106:    LDA  7,-101(7)  CALL output
 107:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
 108:     ST  1,-4(1)    Store old fp in ghost frame
 109:    LDC  3,3(6)     Load constant
 110:     LD  4,-3(1)    Load address of base of array xa
 111:    SUB  3,4,3      Compute offset of value
 112:     LD  3,0(3)     Load the value
 113:     ST  3,-6(1)    Store parameter
 114:    LDA  1,-4(1)    Load address of new frame
 115:    LDA  3,1(7)     Return address in ac
 116:    LDA  7,-111(7)  CALL output
 117:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
 118:     ST  1,-4(1)    Store old fp in ghost frame
 119:    LDA  1,-4(1)    Load address of new frame
 120:    LDA  3,1(7)     Return address in ac
 121:    LDA  7,-97(7)   CALL outnl
 122:    LDA  3,0(2)     Save the result in ac
 * EXPRESSION STMT
 123:    LDC  3,670(6)   Load constant
 124:     ST  3,0(0)     Store variable g
 * EXPRESSION STMT
 125:    LDC  3,2(6)     Load constant
 126:     ST  3,-4(1)    Save left side
 127:    LDC  3,1(6)     Load constant
 128:     LD  4,-4(1)    Load left into ac1
 129:    ADD  3,4,3      Op +
 130:     ST  3,-4(1)    Save index
 131:    LDC  3,671(6)   Load constant
 132:     LD  4,-4(1)    Restore index
 133:    LDA  5,-2(0)    Load address of base of array ga
 134:    SUB  5,5,4      Compute offset of value
 135:     ST  3,0(5)     Store variable ga
 * EXPRESSION STMT
 136:     ST  1,-4(1)    Store old fp in ghost frame
 137:     LD  3,0(0)     Load variable g
 138:     ST  3,-6(1)    Store parameter
 139:    LDA  1,-4(1)    Load address of new frame
 140:    LDA  3,1(7)     Return address in ac
 141:    LDA  7,-136(7)  CALL output
 142:    LDA  3,0(2)     Save the result in ac
```

```
* EXPRESSION STMT
143:     ST  1,-4(1)    Store old fp in ghost frame
144:    LDC  3,3(6)     Load constant
145:    LDA  4,-2(0)    Load address of base of array ga
146:    SUB  3,4,3      Compute offset of value
147:     LD  3,0(3)     Load the value
148:     ST  3,-6(1)    Store parameter
149:    LDA  1,-4(1)    Load address of new frame
150:    LDA  3,1(7)     Return address in ac
151:    LDA  7,-146(7)  CALL output
152:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
153:     ST  1,-4(1)    Store old fp in ghost frame
154:    LDA  1,-4(1)    Load address of new frame
155:    LDA  3,1(7)     Return address in ac
156:    LDA  7,-132(7)  CALL outnl
157:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
158:     ST  1,-4(1)    Store old fp in ghost frame
159:     LD  3,-2(1)    Load variable x
160:     ST  3,-6(1)    Store parameter
161:     LD  3,-3(1)    Load address of base of array xa
162:     ST  3,-7(1)    Store parameter
163:    LDA  1,-4(1)    Load address of new frame
164:    LDA  3,1(7)     Return address in ac
165:    LDA  7,-136(7)  CALL cat
166:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
167:     ST  1,-4(1)    Store old fp in ghost frame
168:     LD  3,0(0)     Load variable g
169:     ST  3,-6(1)    Store parameter
170:    LDA  3,-2(0)    Load address of base of array ga
171:     ST  3,-7(1)    Store parameter
172:    LDA  1,-4(1)    Load address of new frame
173:    LDA  3,1(7)     Return address in ac
174:    LDA  7,-145(7)  CALL cat
175:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
176:    LDC  2,0(6)     Set return value to 0
177:     LD  3,-1(1)    Load return address
178:     LD  1,0(1)     Adjust fp
179:    LDA  7,0(3)     Return
* END of function dog
* BEGIN function fox
180:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
181:    LDC  3,11(6)    load size of array sa
182:     ST  3,-13(0)   save size of array sa
* EXPRESSION STMT
183:    LDC  3,3(6)     Load constant
184:     ST  3,-2(1)    Save index
185:    LDC  3,1000(6)  Load constant
186:     LD  4,-2(1)    Restore index
187:    LDA  5,-14(0)   Load address of base of array sa
188:    SUB  5,5,4      Compute offset of value
189:     ST  3,0(5)     Store variable sa
* EXPRESSION STMT
```

```
190:    LDC  3,777(6)   Load constant
191:     ST  3,-12(0)   Store variable s
* EXPRESSION STMT
192:    LDC  3,3(6)     Load constant
193:     ST  3,-2(1)    Save index
194:     LD  3,-12(0)   Load variable s
195:     ST  3,-3(1)    Save left side
196:    LDC  3,999(6)   Load constant
197:     ST  3,-12(0)   Store variable s
198:     LD  4,-3(1)    Load left into ac1
199:    ADD  3,4,3      Op +
200:     LD  4,-2(1)    Restore index
201:    LDA  5,-14(0)   Load address of base of array sa
202:    SUB  5,5,4      Compute offset of value
203:     LD  4,0(5)     load lhs variable sa
204:    SUB  3,4,3      op -=
205:     ST  3,0(5)     Store variable sa
* EXPRESSION STMT
206:     ST  1,-2(1)    Store old fp in ghost frame
207:     LD  3,-12(0)   Load variable s
208:     ST  3,-4(1)    Store parameter
209:    LDA  1,-2(1)    Load address of new frame
210:    LDA  3,1(7)     Return address in ac
211:    LDA  7,-206(7)  CALL output
212:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
213:     ST  1,-2(1)    Store old fp in ghost frame
214:    LDC  3,3(6)     Load constant
215:    LDA  4,-14(0)   Load address of base of array sa
216:    SUB  3,4,3      Compute offset of value
217:     LD  3,0(3)     Load the value
218:     ST  3,-4(1)    Store parameter
219:    LDA  1,-2(1)    Load address of new frame
220:    LDA  3,1(7)     Return address in ac
221:    LDA  7,-216(7)  CALL output
222:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
223:     ST  1,-2(1)    Store old fp in ghost frame
224:    LDA  1,-2(1)    Load address of new frame
225:    LDA  3,1(7)     Return address in ac
226:    LDA  7,-202(7)  CALL outnl
227:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
228:     ST  1,-2(1)    Store old fp in ghost frame
229:     LD  3,-12(0)   Load variable s
230:     ST  3,-4(1)    Store parameter
231:    LDA  3,-14(0)   Load address of base of array sa
232:     ST  3,-5(1)    Store parameter
233:    LDA  1,-2(1)    Load address of new frame
234:    LDA  3,1(7)     Return address in ac
235:    LDA  7,-206(7)  CALL cat
236:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
237:    LDC  2,0(6)     Set return value to 0
238:     LD  3,-1(1)    Load return address
239:     LD  1,0(1)     Adjust fp
240:    LDA  7,0(3)     Return
```

```
* END of function fox
* BEGIN function main
241:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
242:    LDC  3,12(6)    load size of array ya
243:     ST  3,-3(1)    save size of array ya
* EXPRESSION STMT
244:    LDC  3,666(6)   Load constant
245:     ST  3,-2(1)    Store variable y
* EXPRESSION STMT
246:    LDC  3,1(6)     Load constant
247:     ST  3,-16(1)   Save left side
248:    LDC  3,2(6)     Load constant
249:     LD  4,-16(1)   Load left into ac1
250:    ADD  3,4,3      Op +
251:     ST  3,-16(1)   Save index
252:    LDC  3,667(6)   Load constant
253:     LD  4,-16(1)   Restore index
254:    LDA  5,-4(1)    Load address of base of array ya
255:    SUB  5,5,4      Compute offset of value
256:     ST  3,0(5)     Store variable ya
* EXPRESSION STMT
257:     ST  1,-16(1)   Store old fp in ghost frame
258:     LD  3,-2(1)    Load variable y
259:     ST  3,-18(1)   Store parameter
260:    LDA  1,-16(1)   Load address of new frame
261:    LDA  3,1(7)     Return address in ac
262:    LDA  7,-257(7)  CALL output
263:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
264:     ST  1,-16(1)   Store old fp in ghost frame
265:    LDC  3,3(6)     Load constant
266:    LDA  4,-4(1)    Load address of base of array ya
267:    SUB  3,4,3      Compute offset of value
268:     LD  3,0(3)     Load the value
269:     ST  3,-18(1)   Store parameter
270:    LDA  1,-16(1)   Load address of new frame
271:    LDA  3,1(7)     Return address in ac
272:    LDA  7,-267(7)  CALL output
273:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
274:     ST  1,-16(1)   Store old fp in ghost frame
275:    LDA  1,-16(1)   Load address of new frame
276:    LDA  3,1(7)     Return address in ac
277:    LDA  7,-253(7)  CALL outnl
278:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
279:     ST  1,-16(1)   Store old fp in ghost frame
280:     LD  3,-2(1)    Load variable y
281:     ST  3,-18(1)   Store parameter
282:    LDA  3,-4(1)    Load address of base of array ya
283:     ST  3,-19(1)   Store parameter
284:    LDA  1,-16(1)   Load address of new frame
285:    LDA  3,1(7)     Return address in ac
286:    LDA  7,-222(7)  CALL dog
287:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
288:     ST  1,-16(1)   Store old fp in ghost frame
```

```
289:     LD  3,-2(1)    Load variable y
290:     ST  3,-18(1)   Store parameter
291:    LDA  1,-16(1)   Load address of new frame
292:    LDA  3,1(7)     Return address in ac
293:    LDA  7,-288(7)  CALL output
294:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
295:     ST  1,-16(1)   Store old fp in ghost frame
296:    LDC  3,3(6)     Load constant
297:    LDA  4,-4(1)    Load address of base of array ya
298:    SUB  3,4,3      Compute offset of value
299:     LD  3,0(3)     Load the value
300:     ST  3,-18(1)   Store parameter
301:    LDA  1,-16(1)   Load address of new frame
302:    LDA  3,1(7)     Return address in ac
303:    LDA  7,-298(7)  CALL output
304:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
305:     ST  1,-16(1)   Store old fp in ghost frame
306:    LDA  1,-16(1)   Load address of new frame
307:    LDA  3,1(7)     Return address in ac
308:    LDA  7,-284(7)  CALL outnl
309:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
310:     ST  1,-16(1)   Store old fp in ghost frame
311:    LDA  1,-16(1)   Load address of new frame
312:    LDA  3,1(7)     Return address in ac
313:    LDA  7,-134(7)  CALL fox
314:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
315:    LDC  2,0(6)     Set return value to 0
316:     LD  3,-1(1)    Load return address
317:     LD  1,0(1)     Adjust fp
318:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,318(7)   Jump to init
* BEGIN Init
319:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
320:    LDC  3,10(6)    load size of array ga
321:     ST  3,-1(0)    save size of array ga
* END init of global array sizes
322:    LDA  1,-25(0)   set first frame at end of globals
323:     ST  1,0(1)     store old fp (point to self)
324:    LDA  3,1(7)     Return address in ac
325:    LDA  7,-85(7)   Jump to main
326:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
exp.c-
--------
void main()
{
        true and 111 < 222 + 333*444;
        444*333 + 222 > 111 and true;
}
==============================+==============================
exp.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  exp.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
 26:  OUTNL  3,3,3      Output a newline
 27:     LD  3,-1(1)    Load return address
 28:     LD  1,0(1)     Adjust fp
 29:    LDA  7,0(3)     Return
* END of function outnl
* BEGIN function main
 30:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 31:    LDC  3,1(6)     Load constant
 32:     ST  3,-2(1)    Save left side
 33:    LDC  3,111(6)   Load constant
 34:     ST  3,-3(1)    Save left side
 35:    LDC  3,222(6)   Load constant
 36:     ST  3,-4(1)    Save left side
 37:    LDC  3,333(6)   Load constant
 38:     ST  3,-5(1)    Save left side
 39:    LDC  3,444(6)   Load constant
 40:     LD  4,-5(1)    Load left into ac1
 41:    MUL  3,4,3      Op *
 42:     LD  4,-4(1)    Load left into ac1
 43:    ADD  3,4,3      Op +
 44:     LD  4,-3(1)    Load left into ac1
 45:    SUB  4,4,3      Op <
 46:    LDC  3,1(6)     True case
 47:    JLT  4,1(7)     Jump if true
 48:    LDC  3,0(6)     False case
 49:     LD  4,-2(1)    Load left into ac1
 50:    JEQ  3,1(7)     Op AND
 51:    LDA  3,0(4)
* EXPRESSION STMT
 52:    LDC  3,444(6)   Load constant
 53:     ST  3,-2(1)    Save left side
 54:    LDC  3,333(6)   Load constant
 55:     LD  4,-2(1)    Load left into ac1
 56:    MUL  3,4,3      Op *
 57:     ST  3,-2(1)    Save left side
 58:    LDC  3,222(6)   Load constant
 59:     LD  4,-2(1)    Load left into ac1
 60:    ADD  3,4,3      Op +
 61:     ST  3,-2(1)    Save left side
 62:    LDC  3,111(6)   Load constant
 63:     LD  4,-2(1)    Load left into ac1
 64:    SUB  4,4,3      Op >
 65:    LDC  3,1(6)     True case
 66:    JGT  4,1(7)     Jump if true
 67:    LDC  3,0(6)     False case
 68:     ST  3,-2(1)    Save left side
 69:    LDC  3,1(6)     Load constant
 70:     LD  4,-2(1)    Load left into ac1
 71:    JEQ  3,1(7)     Op AND
 72:    LDA  3,0(4)
* END compound statement
* Add standard closing in case there is no return statement
 73:    LDC  2,0(6)     Set return value to 0
 74:     LD  3,-1(1)    Load return address
 75:     LD  1,0(1)     Adjust fp
 76:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,76(7)    Jump to init
* BEGIN Init
 77:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
* END init of global array sizes
 78:    LDA  1,0(0)     set first frame at end of globals
 79:     ST  1,0(1)     store old fp (point to self)
 80:    LDA  3,1(7)     Return address in ac
 81:    LDA  7,-52(7)   Jump to main
 82:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
logic.c-
--------
void main()
{
    bool a, b;
    a = true;
    b = false;


    a and b;
    a or b;
    not a;

    111>222;
    111<222;
    111==222;
    111>=222;
    111<=222;
    111!=222;
}

==============================+==============================
logic.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  logic.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
```

```
  26:  OUTNL  3,3,3      Output a newline
  27:     LD  3,-1(1)    Load return address
  28:     LD  1,0(1)     Adjust fp
  29:    LDA  7,0(3)     Return
 * END of function outnl
 * BEGIN function main
  30:     ST  3,-1(1)    Store return address.
 * BEGIN compound statement
 * EXPRESSION STMT
  31:    LDC  3,1(6)     Load constant
  32:     ST  3,-2(1)    Store variable a
 * EXPRESSION STMT
  33:    LDC  3,0(6)     Load constant
  34:     ST  3,-3(1)    Store variable b
 * EXPRESSION STMT
  35:     LD  3,-2(1)    Load variable a
  36:     ST  3,-4(1)    Save left side
  37:     LD  3,-3(1)    Load variable b
  38:     LD  4,-4(1)    Load left into ac1
  39:    JEQ  3,1(7)     Op AND
  40:    LDA  3,0(4)
 * EXPRESSION STMT
  41:     LD  3,-2(1)    Load variable a
  42:     ST  3,-4(1)    Save left side
  43:     LD  3,-3(1)    Load variable b
  44:     LD  4,-4(1)    Load left into ac1
  45:    JGT  3,1(7)     Op OR
  46:    LDA  3,0(4)
 * EXPRESSION STMT
  47:     LD  3,-2(1)    Load variable a
  48:    LDC  4,1(6)     Load 1
  49:    SUB  3,4,3      Op NOT
 * EXPRESSION STMT
  50:    LDC  3,111(6)   Load constant
  51:     ST  3,-4(1)    Save left side
  52:    LDC  3,222(6)   Load constant
  53:     LD  4,-4(1)    Load left into ac1
  54:    SUB  4,4,3      Op >
  55:    LDC  3,1(6)     True case
  56:    JGT  4,1(7)     Jump if true
  57:    LDC  3,0(6)     False case
 * EXPRESSION STMT
  58:    LDC  3,111(6)   Load constant
  59:     ST  3,-4(1)    Save left side
  60:    LDC  3,222(6)   Load constant
  61:     LD  4,-4(1)    Load left into ac1
  62:    SUB  4,4,3      Op <
  63:    LDC  3,1(6)     True case
  64:    JLT  4,1(7)     Jump if true
  65:    LDC  3,0(6)     False case
 * EXPRESSION STMT
  66:    LDC  3,111(6)   Load constant
  67:     ST  3,-4(1)    Save left side
  68:    LDC  3,222(6)   Load constant
  69:     LD  4,-4(1)    Load left into ac1
  70:    SUB  4,4,3      Op ==
  71:    LDC  3,1(6)     True case
  72:    JEQ  4,1(7)     Jump if true
 73:    LDC  3,0(6)     False case
* EXPRESSION STMT
 74:    LDC  3,111(6)   Load constant
 75:     ST  3,-4(1)    Save left side
 76:    LDC  3,222(6)   Load constant
 77:     LD  4,-4(1)    Load left into ac1
 78:    SUB  4,4,3      Op >=
 79:    LDC  3,1(6)     True case
 80:    JGE  4,1(7)     Jump if true
 81:    LDC  3,0(6)     False case
* EXPRESSION STMT
 82:    LDC  3,111(6)   Load constant
 83:     ST  3,-4(1)    Save left side
 84:    LDC  3,222(6)   Load constant
 85:     LD  4,-4(1)    Load left into ac1
 86:    SUB  4,4,3      Op <=
 87:    LDC  3,1(6)     True case
 88:    JLE  4,1(7)     Jump if true
 89:    LDC  3,0(6)     False case
* EXPRESSION STMT
 90:    LDC  3,111(6)   Load constant
 91:     ST  3,-4(1)    Save left side
 92:    LDC  3,222(6)   Load constant
 93:     LD  4,-4(1)    Load left into ac1
 94:    SUB  3,4,3      Op !=
 95:    JEQ  3,1(7)     Jump if true
 96:    LDC  3,1(6)     True case
* END compound statement
* Add standard closing in case there is no return statement
 97:    LDC  2,0(6)     Set return value to 0
 98:     LD  3,-1(1)    Load return address
 99:     LD  1,0(1)     Adjust fp
100:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,100(7)   Jump to init
* BEGIN Init
101:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
* END init of global array sizes
102:    LDA  1,0(0)     set first frame at end of globals
103:     ST  1,0(1)     store old fp (point to self)
104:    LDA  3,1(7)     Return address in ac
105:    LDA  7,-76(7)   Jump to main
106:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
factorialr.c-
--------
// C-F10
// recursive factorial function
int factorial(int n)
{
        if (n<2) return 1;
        else return n*factorial(n-1);
}

void main()
{
        int n;
        n = input();
        n = factorial(n);
        output(n);
        outnl();
}


==============================+==============================
factorialr.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  factorialr.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
 26:  OUTNL  3,3,3      Output a newline
 27:     LD  3,-1(1)    Load return address
 28:     LD  1,0(1)     Adjust fp
 29:    LDA  7,0(3)     Return
* END of function outnl
* BEGIN function factorial
 30:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* IF
```

```
 31:     LD  3,-2(1)    Load variable n
 32:     ST  3,-3(1)    Save left side
 33:    LDC  3,2(6)     Load constant
 34:     LD  4,-3(1)    Load left into ac1
 35:    SUB  4,4,3      Op <
 36:    LDC  3,1(6)     True case
 37:    JLT  4,1(7)     Jump if true
 38:    LDC  3,0(6)     False case
 39:    JGT  3,1(7)     Jump to then part
* THEN
* RETURN
 41:    LDC  3,1(6)     Load constant
 42:    LDA  2,0(3)     Copy result to rt register
 43:     LD  3,-1(1)    Load return address
 44:     LD  1,0(1)     Adjust fp
 45:    LDA  7,0(3)     Return
* ELSE
 40:    LDA  7,6(7)     Jump around the THEN
* RETURN
 47:     LD  3,-2(1)    Load variable n
 48:     ST  3,-3(1)    Save left side
 49:     ST  1,-4(1)    Store old fp in ghost frame
 50:     LD  3,-2(1)    Load variable n
 51:     ST  3,-6(1)    Save left side
 52:    LDC  3,1(6)     Load constant
 53:     LD  4,-6(1)    Load left into ac1
 54:    SUB  3,4,3      Op -
 55:     ST  3,-6(1)    Store parameter
 56:    LDA  1,-4(1)    Load address of new frame
 57:    LDA  3,1(7)     Return address in ac
 58:    LDA  7,-29(7)   CALL factorial
 59:    LDA  3,0(2)     Save the result in ac
 60:     LD  4,-3(1)    Load left into ac1
 61:    MUL  3,4,3      Op *
 62:    LDA  2,0(3)     Copy result to rt register
 63:     LD  3,-1(1)    Load return address
 64:     LD  1,0(1)     Adjust fp
 65:    LDA  7,0(3)     Return
 46:    LDA  7,19(7)    Jump around the ELSE
* ENDIF
* END compound statement
* Add standard closing in case there is no return statement
 66:    LDC  2,0(6)     Set return value to 0
 67:     LD  3,-1(1)    Load return address
 68:     LD  1,0(1)     Adjust fp
 69:    LDA  7,0(3)     Return
* END of function factorial
* BEGIN function main
 70:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 71:     ST  1,-3(1)    Store old fp in ghost frame
 72:    LDA  1,-3(1)    Load address of new frame
 73:    LDA  3,1(7)     Return address in ac
 74:    LDA  7,-74(7)   CALL input
 75:    LDA  3,0(2)     Save the result in ac
 76:     ST  3,-2(1)    Store variable n
* EXPRESSION STMT
```

```
 77:     ST  1,-3(1)    Store old fp in ghost frame
 78:     LD  3,-2(1)    Load variable n
 79:     ST  3,-5(1)    Store parameter
 80:    LDA  1,-3(1)    Load address of new frame
 81:    LDA  3,1(7)     Return address in ac
 82:    LDA  7,-53(7)   CALL factorial
 83:    LDA  3,0(2)     Save the result in ac
 84:     ST  3,-2(1)    Store variable n
* EXPRESSION STMT
 85:     ST  1,-3(1)    Store old fp in ghost frame
 86:     LD  3,-2(1)    Load variable n
 87:     ST  3,-5(1)    Store parameter
 88:    LDA  1,-3(1)    Load address of new frame
 89:    LDA  3,1(7)     Return address in ac
 90:    LDA  7,-85(7)   CALL output
 91:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
 92:     ST  1,-3(1)    Store old fp in ghost frame
 93:    LDA  1,-3(1)    Load address of new frame
 94:    LDA  3,1(7)     Return address in ac
 95:    LDA  7,-71(7)   CALL outnl
 96:    LDA  3,0(2)     Save the result in ac
* END compound statement
* Add standard closing in case there is no return statement
 97:    LDC  2,0(6)     Set return value to 0
 98:     LD  3,-1(1)    Load return address
 99:     LD  1,0(1)     Adjust fp
100:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,100(7)   Jump to init
* BEGIN Init
101:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
* END init of global array sizes
102:    LDA  1,0(0)     set first frame at end of globals
103:     ST  1,0(1)     store old fp (point to self)
104:    LDA  3,1(7)     Return address in ac
105:    LDA  7,-36(7)   Jump to main
106:   HALT  0,0,0      DONE!
* END Init
==============================+==============================
comb.c-
--------
// C-F10
int comb(int n; int r)
{
     int i; int j; int c;

     c=1;

     i=n;
     j=1;
     while (j<=r) {
          c = c*i/j;
          i--;
          j++;
     }
     return c;
}

void main()
{
     int max; int n;
     bool xx;
     int r;

     max = 20;

     n=1;
     while (n<=max) {
          r=0;
          while (r<=n) {
               output(comb(n, r));
               r++;
          }
          outnl();
          n++;
     }
}



==============================+==============================
comb.tm
--------
* C- compiler version C-F10
* Built: Nov 27, 2010
* Author: Robert B. Heckendorn
* File compiled:  comb.c-
* BEGIN function input
  1:     ST  3,-1(1)    Store return address
  2:     IN  2,2,2      Grab int input
  3:     LD  3,-1(1)    Load return address
  4:     LD  1,0(1)     Adjust fp
  5:    LDA  7,0(3)     Return
* END of function input
* BEGIN function output
  6:     ST  3,-1(1)    Store return address
  7:     LD  3,-2(1)    Load parameter
  8:    OUT  3,3,3      Output integer
  9:    LDC  2,0(6)     Set return to 0
 10:     LD  3,-1(1)    Load return address
 11:     LD  1,0(1)     Adjust fp
 12:    LDA  7,0(3)     Return
* END of function output
* BEGIN function inputb
 13:     ST  3,-1(1)    Store return address
 14:    INB  2,2,2      Grab bool input
 15:     LD  3,-1(1)    Load return address
 16:     LD  1,0(1)     Adjust fp
 17:    LDA  7,0(3)     Return
* END of function inputb
* BEGIN function outputb
 18:     ST  3,-1(1)    Store return address
 19:     LD  3,-2(1)    Load parameter
 20:   OUTB  3,3,3      Output bool
 21:    LDC  2,0(6)     Set return to 0
 22:     LD  3,-1(1)    Load return address
 23:     LD  1,0(1)     Adjust fp
 24:    LDA  7,0(3)     Return
* END of function outputb
* BEGIN function outnl
 25:     ST  3,-1(1)    Store return address
 26:  OUTNL  3,3,3      Output a newline
 27:     LD  3,-1(1)    Load return address
 28:     LD  1,0(1)     Adjust fp
 29:    LDA  7,0(3)     Return
* END of function outnl
* BEGIN function comb
 30:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 31:    LDC  3,1(6)     Load constant
 32:     ST  3,-6(1)    Store variable c
* EXPRESSION STMT
 33:     LD  3,-2(1)    Load variable n
 34:     ST  3,-4(1)    Store variable i
* EXPRESSION STMT
 35:    LDC  3,1(6)     Load constant
 36:     ST  3,-5(1)    Store variable j
* WHILE
 37:     LD  3,-5(1)    Load variable j
 38:     ST  3,-7(1)    Save left side
 39:     LD  3,-3(1)    Load variable r
 40:     LD  4,-7(1)    Load left into ac1
 41:    SUB  4,4,3      Op <=
 42:    LDC  3,1(6)     True case
 43:    JLE  4,1(7)     Jump if true
 44:    LDC  3,0(6)     False case
 45:    JGT  3,1(7)     Jump to while part
* DO
* BEGIN compound statement
* EXPRESSION STMT
 47:     LD  3,-6(1)    Load variable c
 48:     ST  3,-7(1)    Save left side
 49:     LD  3,-4(1)    Load variable i
 50:     LD  4,-7(1)    Load left into ac1
 51:    MUL  3,4,3      Op *
 52:     ST  3,-7(1)    Save left side
 53:     LD  3,-5(1)    Load variable j
 54:     LD  4,-7(1)    Load left into ac1
 55:    DIV  3,4,3      Op /
 56:     ST  3,-6(1)    Store variable c
* EXPRESSION STMT
 57:     LD  3,-4(1)    load lhs variable i
 58:    LDA  3,-1(3)    decrement value of i
 59:     ST  3,-4(1)    Store variable i
* EXPRESSION STMT
 60:     LD  3,-5(1)    load lhs variable j
 61:    LDA  3,1(3)     increment value of j
 62:     ST  3,-5(1)    Store variable j
* END compound statement
 63:    LDA  7,-27(7)   go to beginning of loop
 46:    LDA  7,17(7)    No more loop
```

```
* ENDWHILE
* RETURN
 64:     LD  3,-6(1)    Load variable c
 65:    LDA  2,0(3)     Copy result to rt register
 66:     LD  3,-1(1)    Load return address
 67:     LD  1,0(1)     Adjust fp
 68:    LDA  7,0(3)     Return
* END compound statement
* Add standard closing in case there is no return statement
 69:    LDC  2,0(6)     Set return value to 0
 70:     LD  3,-1(1)    Load return address
 71:     LD  1,0(1)     Adjust fp
 72:    LDA  7,0(3)     Return
* END of function comb
* BEGIN function main
 73:     ST  3,-1(1)    Store return address.
* BEGIN compound statement
* EXPRESSION STMT
 74:    LDC  3,20(6)    Load constant
 75:     ST  3,-2(1)    Store variable max
* EXPRESSION STMT
 76:    LDC  3,1(6)     Load constant
 77:     ST  3,-3(1)    Store variable n
* WHILE
 78:     LD  3,-3(1)    Load variable n
 79:     ST  3,-6(1)    Save left side
 80:     LD  3,-2(1)    Load variable max
 81:     LD  4,-6(1)    Load left into ac1
 82:    SUB  4,4,3      Op <=
 83:    LDC  3,1(6)     True case
 84:    JLE  4,1(7)     Jump if true
 85:    LDC  3,0(6)     False case
 86:    JGT  3,1(7)     Jump to while part
* DO
* BEGIN compound statement
* EXPRESSION STMT
 88:    LDC  3,0(6)     Load constant
 89:     ST  3,-5(1)    Store variable r
* WHILE
 90:     LD  3,-5(1)    Load variable r
 91:     ST  3,-6(1)    Save left side
 92:     LD  3,-3(1)    Load variable n
 93:     LD  4,-6(1)    Load left into ac1
 94:    SUB  4,4,3      Op <=
 95:    LDC  3,1(6)     True case
 96:    JLE  4,1(7)     Jump if true
 97:    LDC  3,0(6)     False case
 98:    JGT  3,1(7)     Jump to while part
* DO
* BEGIN compound statement
* EXPRESSION STMT
100:     ST  1,-6(1)    Store old fp in ghost frame
101:     ST  1,-8(1)    Store old fp in ghost frame
102:     LD  3,-3(1)    Load variable n
103:     ST  3,-10(1)   Store parameter
104:     LD  3,-5(1)    Load variable r
105:     ST  3,-11(1)   Store parameter
106:    LDA  1,-8(1)    Load address of new frame
```

```
107:    LDA  3,1(7)     Return address in ac
108:    LDA  7,-79(7)   CALL comb
109:    LDA  3,0(2)     Save the result in ac
110:     ST  3,-8(1)    Store parameter
111:    LDA  1,-6(1)    Load address of new frame
112:    LDA  3,1(7)     Return address in ac
113:    LDA  7,-108(7)  CALL output
114:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
115:     LD  3,-5(1)    load lhs variable r
116:    LDA  3,1(3)     increment value of r
117:     ST  3,-5(1)    Store variable r
* END compound statement
118:    LDA  7,-29(7)   go to beginning of loop
 99:    LDA  7,19(7)    No more loop
* ENDWHILE
* EXPRESSION STMT
119:     ST  1,-6(1)    Store old fp in ghost frame
120:    LDA  1,-6(1)    Load address of new frame
121:    LDA  3,1(7)     Return address in ac
122:    LDA  7,-98(7)   CALL outnl
123:    LDA  3,0(2)     Save the result in ac
* EXPRESSION STMT
124:     LD  3,-3(1)    load lhs variable n
125:    LDA  3,1(3)     increment value of n
126:     ST  3,-3(1)    Store variable n
* END compound statement
127:    LDA  7,-50(7)   go to beginning of loop
 87:    LDA  7,40(7)    No more loop
* ENDWHILE
* END compound statement
* Add standard closing in case there is no return statement
128:    LDC  2,0(6)     Set return value to 0
129:     LD  3,-1(1)    Load return address
130:     LD  1,0(1)     Adjust fp
131:    LDA  7,0(3)     Return
* END of function main
  0:    LDA  7,131(7)   Jump to init
* BEGIN Init
132:     LD  0,0(0)     Set the global pointer
* BEGIN init of global array sizes
* END init of global array sizes
133:    LDA  1,0(0)     set first frame at end of globals
134:     ST  1,0(1)     store old fp (point to self)
135:    LDA  3,1(7)     Return address in ac
136:    LDA  7,-64(7)   Jump to main
137:   HALT  0,0,0      DONE!
* END Init
```