C code
int c,d;
main()
{
    int a,b;
a = 1;
b = 2;
c = 3;
b = bill(a,&c);
a = b + 3;
return;
}

int bill(int x, int *y)
{
    int I,b,d;
    I = x + *y;
b = I + 3;
x = 64;
c = 58;
d = 72;
return I;
}

Assembly code
main: MOV BP,SP ; initialize BP for main()
; int a,b;
    ADD SP,4 ; reserve space for a,b
; a = 1;
    MOV 2[BP],1 ; all local variables
; b = 2;
    MOV 4[BP],2 ; are stored on the stack.
; c = 3;
    MOV [8000],3 ; global variables are.
; b = bill(a,&c);
    CALL BILL ; stored in the data seg.
    PUSH BX ; push all registers
    ; except for AX
    PUSH BP
    PUSH 2[BP] ; push 1st argument, x
    MOV 8000 ; push 2nd argument, &y
    CALL BILL ; make BP point to 1st arg.
    SUB BP,4 ; remove x, &y from stack
    POP BP ; restore all registers
    ; except for AX
    POP BX
    MOV 4[BP],AX ; arg. is passed through AX
; a = b + 3;
    MOV AX,4[BP]
    ADD AX,3
    MOV 2[BP],AX
; return;
    SUB SP,4 ; remove a, b from stack
    RET

BILL:
; int I,b,d;
    ADD SP,6 ; reserve space for I, b, d
; I = x + *y;
    MOV AX,[BP]
    MOV SI,2[BP] ; SI = 8000 = &c
    ADD AX,[SI] ; gets memory loc. (8000)
    MOV 6[BP],AX
; b = I + 3;
    MOV AX,6[BP]
    ADD AX,3
    MOV 8[BP],AX
; x = 64;
    MOV [BP],64
; c = 58;
    MOV [8000],58 ; gets global c
; d = 72;
    MOV 10[BP],72 ; gets local d.
; return I;
    MOV AX,6[BP] ; result returned through AX
    SUB SP,6 ; remove I, b, d from stack
    RET

symbol table while compiling main()
symbol table
  c [8000]
d [8002]
a 2[BP]
b 4[BP]

symbol table while compiling bill()
symbol table
  c [8000]
d [8002]
x [BP]
y 2[BP]
I 6[BP]
b 8[BP]
d 10[BP]
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>0003</td>
<td>c [8000] data segment</td>
</tr>
<tr>
<td>8002</td>
<td>d [8002]</td>
<td></td>
</tr>
<tr>
<td>FF00</td>
<td></td>
<td>code segment</td>
</tr>
<tr>
<td>FF37</td>
<td></td>
<td>call point</td>
</tr>
<tr>
<td>FFF0</td>
<td></td>
<td>stack segment (grows down)</td>
</tr>
</tbody>
</table>

**SP.BP->**
- A000: ret addr. Here before 1 then again after 11
- 0001: a 2[BP]
- 0002: b 4[BP] Here after 1 and before 2 then again after 2 registers

**BP->**
- 0001: x [BP]
- 8000: y 2[BP]
- FF37: ret. addr. Here before 3 and after 9
- 0003: 1 6[BP]
- 0007: b 8[BP]
- 0072: d 10[BP] Here after 3 but before 9

**SP->**
- If making a function call from bill the registers, arguments, return address and local variables will be pushed starting here.

**Comments**
- Local variables are declared on the stack while global variables, those with file scope, are declared in the data segment.

- The entries in the symbol table for local variables only exist while compiling that block. Once the compiler finishes with the current block it deletes all of the entries in the symbol table that correspond with the variables declared in the block. The global variables remain in the symbol table since they have file scope.

- The convention in my example is to have the BP register point to the beginning of the argument list. Variables that are passed to a function are called arguments.

- When a function is called the stack grows as follows:
  1. The calling function pushes all of the registers excluding AX.
  2. The calling function pushes all of the arguments.
  3. The CALL instruction is executed pushing the return address.
  4. The function called pushes all of its local variables or may just reserves space for them.

- When the function finishes the stack shrinks as follows:
  1. The function called removes all of its local variables from the stack.
  2. The RET instruction is executed removing the return address from the stack.
  3. The calling function removes the arguments from the stack.
  4. The calling function restores all of the registers, excluding AX.

- The return result, the data returned through the function name, is passed to the calling function through the AX register.