C to LC3 Exercises...What was the C code that resulted in the compiler generating the assembly code shown below...

Solutions

Symbol table generated by compiler:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
<th>Offset</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>int</td>
<td>0</td>
<td>global</td>
</tr>
<tr>
<td>B</td>
<td>int</td>
<td>2</td>
<td>global</td>
</tr>
<tr>
<td>X</td>
<td>int</td>
<td>0</td>
<td>main</td>
</tr>
<tr>
<td>Y</td>
<td>int</td>
<td>-1</td>
<td>main</td>
</tr>
<tr>
<td>Z</td>
<td>int</td>
<td>-2</td>
<td>main</td>
</tr>
</tbody>
</table>

Code 1:

```
AND R0, R0, #0       Initialize R0 to 5 and store this value into R5 + offset 0 which is
ADD R0, R0, #5       address of local variable X
STR R0, R5, #0       X= 5;
```

Code 2

```
AND R1, R1, #0       Initialize R1 to 3 and store into address R4 + offset 0..
ADD R1, R1, #3       R4 is global pointer, so this accesses global variable A
STR R1, R4, #0       A =3;
```

Code 3:

```
LDR R0, R5, #0       Load A, X into registers.
LDR R1, R4, #0       This adds values in R1 and R0 and stores into address of local var Y
ADD R2, R1, R0
STR R2, R5, #-1
Y =A+X;
```

The above code can be optimized by observing that R1,R0 contain A,X and they did not change..So rather than reloading X,A from memory, it can use the values stored in R0,R1.
Code 4:
LDR R1, R5, #0  Load value of local variable X into register R1, and then add 5, and
ADD R1, R1, #5  store this in memory address R5 + (-1) which is local var Y
STR R1, R5, # -1  Y = X +5;

Code 5:
LDR R0, R4, #2  Load value of global var B into R0 and store this into local var Z
STR R0, R5, # -2  Z = B;

Code 6:
ADD R0, R5, # -1  Set R0 equal to R5 + (-1) which is address of local var X
STR R0, R5, # -2  store this value (the address of X) to local variable Z
Z = &Y;