Two equivalent ways to think of a **counter machine**:

1. A stack with a bottom marker, say $Z_0$, and one other symbol, say $X$, that can be placed on the stack. Thus, the stack always looks like $XXX \cdots XZ_0$, or specifically, $X^nZ_0$.

2. A device that holds a non-negative integer with operations *increment-by-1*, *decrement-by-1*, and *test-for-zero*. 
1. Every language accepted by a counter machine is recursively enumerable.
   - Counter machines are special cases of stack machines, which are special cases of multitape TMs, which accept only recursively enumerable languages.

2. Every language accepted by a one-counter machine is a CFL.
   - A one-counter machine is a special case of a one-stack machine; i.e., a PDA.
Theorem 1

If a language $L$ is accepted by a TM $M$, then $L$ is accepted by a two-stack machine.

Proof.

Idea: Use one stack to hold what is to the left of the tape, and use the other to hold what is to the right of the tape.
Consider $L = \{a^n b^n c^n \mid n \geq 0\}$, TM $M_1$ where $L(M_1) = L$, and $w \in \{a, b, c\}^*$ where $|w| = n$.

So, we can derive an equivalent 2-stack machine, $M_2$ ...
If a stack has \( r - 1 \) symbols, think of the stack contents as a base-\( r \) number with the symbols as the digits 1 through \( r - 1 \).
So, for the 3-counter machine, $M_3$, use one counter for each stack plus one "scratch" counter.
Consider the string \( w = aaabbbcccc \) and the ID \( aaqabbbcccc \) for \( M_1 \) where \( q \in Q \) ...
2-stacks-to-3-counters conversion

which is represented by $M_3$ as

![Diagram showing $M_3$ and its connections to stacks $S_L$ and $S_R$.]

Note: To move $M_1$’s read-write head one cell to the right requires popping $X_i$ from $S_R$ and pushing $X_i$ into $S_L$ for $M_2$. 
2-stacks-to-3-counters conversion

Consider the move from $aaqabbbccc$ to $aaarbbbcce$ in $M_1$ where $q, r \in Q$ ...
2-stacks-to-3-counters conversion

Read top symbol of $S_R = \text{store the remainder, } C_2 \text{ modulo } r, \text{ into } C_3$. 

![Diagram showing the conversion process with states and transitions labeled]
2-stacks-to-3-counters conversion

Pop from $S_R = \text{divide } C_2 \text{ by } r$, discarding the remainder.
2-stacks-to-3-counters conversion

Push into $S_L =$ multiply $C_1$ by $r$, then add the value stored in $C_3$. 

\[ \text{Push into } S_L = \text{multiply } C_1 \text{ by } r, \text{ then add the value stored in } C_3. \]
Two-Counter Machines are Turing-equivalent

Theorem 2

*Every recursively enumerable language is accepted by a two-counter machine.*

Proof.

- Develop a constructive algorithm for a 3-counters-to-2-counters conversion. Do this by representing the 3 counters $i$, $j$, and $k$, by a single integer $m = 2^i 3^j 5^k$.

- Store the number $m = 2^i 3^j 5^k$ in one counter and use the other one as a "scratch" counter.

- Test if $i = 0$ by moving count from one counter to the "scratch," counting modulo 2 in the state. $i = 0$ iff the number $m = 2^i 3^j 5^k$ is not divisible by 2. Tests for $j = 0$ and $k = 0$ analogous.
Proof.

- Incrementing counters $i$, $j$, or $k$ is equivalent to multiplications of the count $m = 2^i3^j5^k$ by 2, 3, or 5; respectively.
- Decrementing counters $i$, $j$, or $k$ is equivalent to divisions of the count $m = 2^i3^j5^k$ by 2, 3, or 5; respectively.