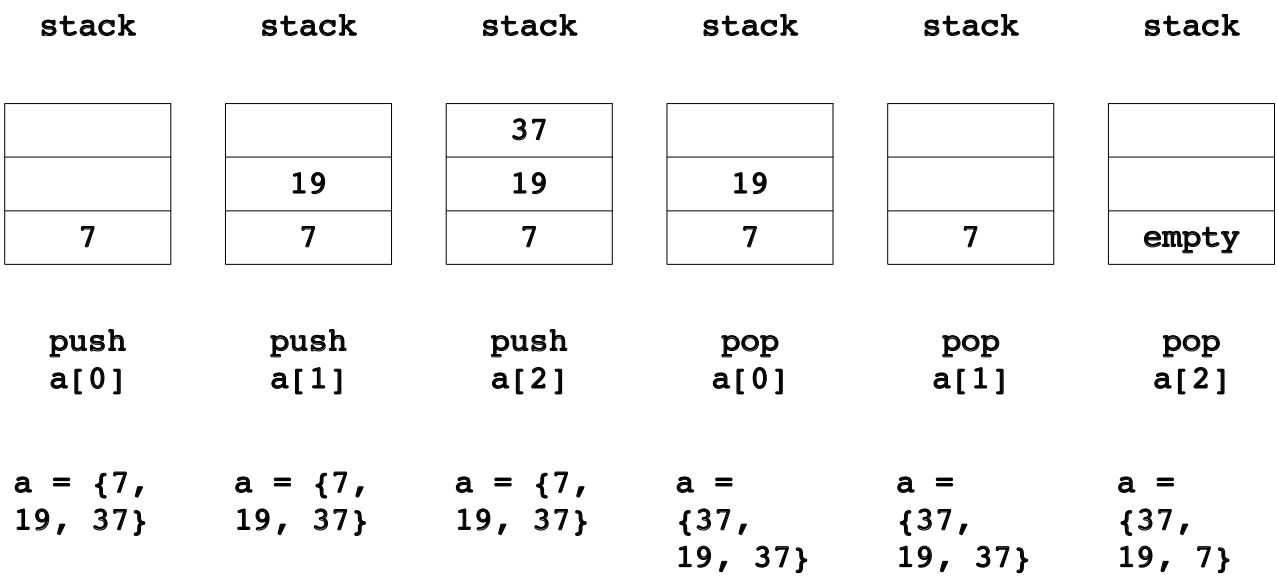
The following diagram shows the progress and the results.



The following sections provide more detail regarding the implementation and applicable instructions.

7.2 Stack Implementation

The current top of the stack is pointed to by the **$sp** register. The stack grows downward in memory and it is generally expected that all items pushed and/or popped should be of word size (32-bit).

There is no push or pop instruction. Instead, you must perform the push and pop operations manually.

While it is possible to push/pop items of various sizes (byte, halfword, etc.) it is not recommended. For such operations, it is recommended to use the entire word (4-bytes).

7.3 Push

For example, a push would subtract the **$sp** by 4 bytes and then copy the operand to that location (in that order). The instructions to push **$t9** would be implemented as follows:

```
subu   $sp, $sp, 4
sw     $t9, ($sp)
```

Which will place the contents of the **$t9** register at the top of the stack.

7.4 Pop

A pop would copy the top of the stack to the operand and then add 4 bytes (in that order). To pop the stack into **$t2**, the instructions would be as follows:

```
lw     $t2, ($sp)
addu   $sp, $sp, 4
```

Which will copy the contents of the top of the stack into the **$t2** register.

7.5 Multiple push's/pop's

The preferred method of performing multiple pushes or pops is to perform the **$sp** adjustment only once. For example, to push registers, **$s0**, **$s1**, and **$s2**:

```
subu   $sp, $sp, 12
sw     $s0, ($sp)
sw     $s1, 4($sp)
sw     $s2, 8($sp)
```

And, the commands to pop registers, **$s0**, **$s1**, and **$s2** as as follows:

```
lw     $s0, ($sp)
lw     $s1, 4($sp)
lw     $s2, 8($sp)
addu   $sp, $sp, 12
```

By performing the stack adjustment only once, it is more efficient for the architecture to execute.

7.6 Example Program, Stack Usage

The following example uses a stack to reverse the elements in an array. The program will push all elements of the array to the stack and then pop all elements back into the array. This will place the elements back into the array in reverse order based on the basic functionality of the stack.

```
#  Example to reverse values in an array
#  by using the stack.
```