- In this case our subroutine is trivial and uses few registers.
  But in general you may need several registers in a subroutine.
- If these registers are being used to hold data in the caller., then we cannot change them in the subroutine.
- Solution: we need to save the registers at the start of the sub-routine, and then restore them before returning. We can use the concept of a stack data structure.

```
.define   STACK   255    // bottom of memory.
MAIN:     mvi     r6, #STACK
          mvi     r1, #1   // used to subtract, add.
          mv      r5, r7
          mvi     r7, #my_sub
          :
my_sub:   sub.    r6, r1   // save regs on stack.
          st.     r2, [r6]
          sub.    r6, r1
          st.     r3, [r6]
          sub.    r6, r1
          st.     r4, [r6]   // done saving regs.
          :
          // Subroutine can modify r2, r3, r4
ld:       r4, [r6]
add:      r6, r1
ld        r3, r6
add       r6, r1
```

Push onto stack.

STACK.
r6 → 252 : r4
r6 → 253 : r3
r6 → 254 : r2
r6 → 255 : ?

pop off stack.

```
ld    r2, r6
add   r6, r1
add   r5, r1
add   r5, r1
mv    r7, r5
```