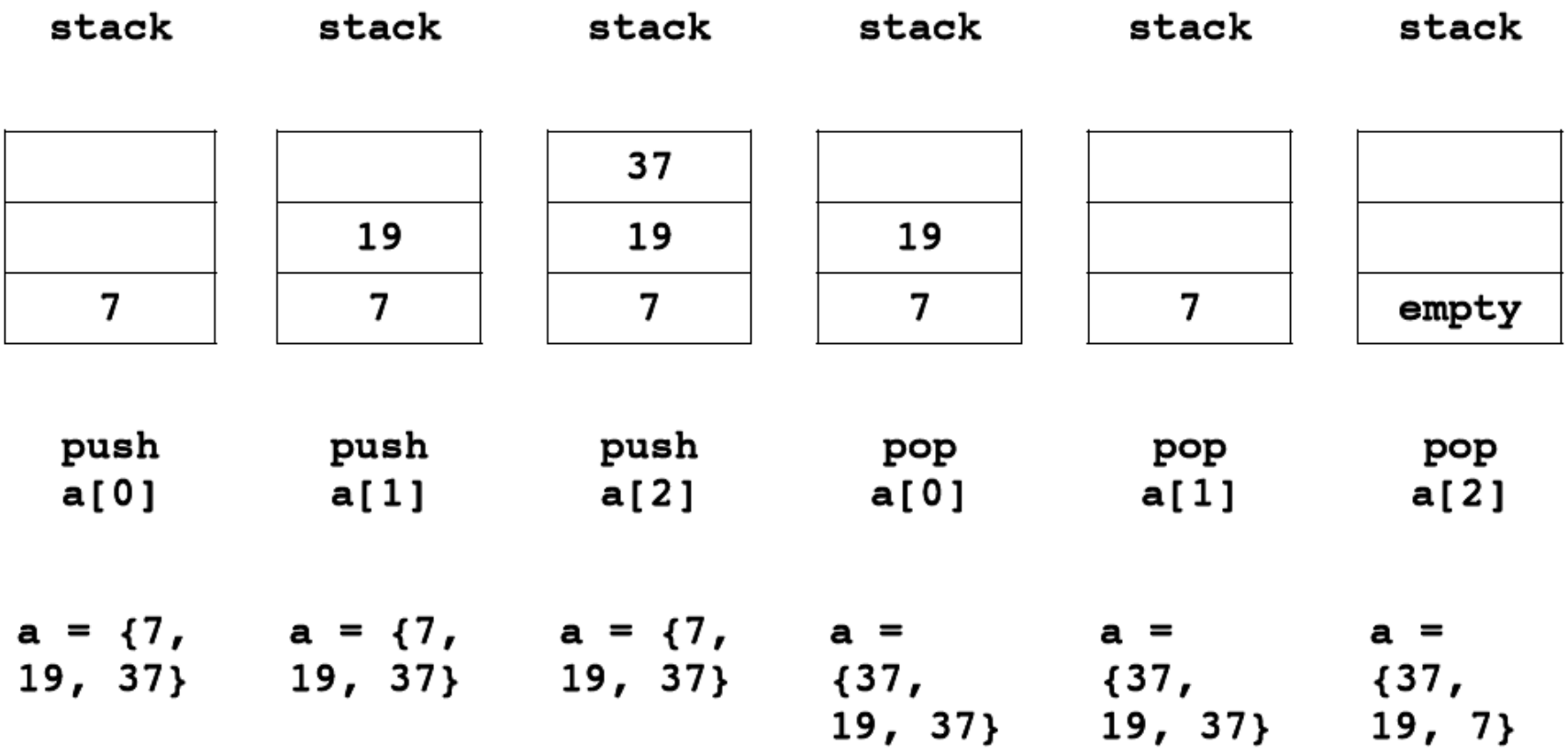
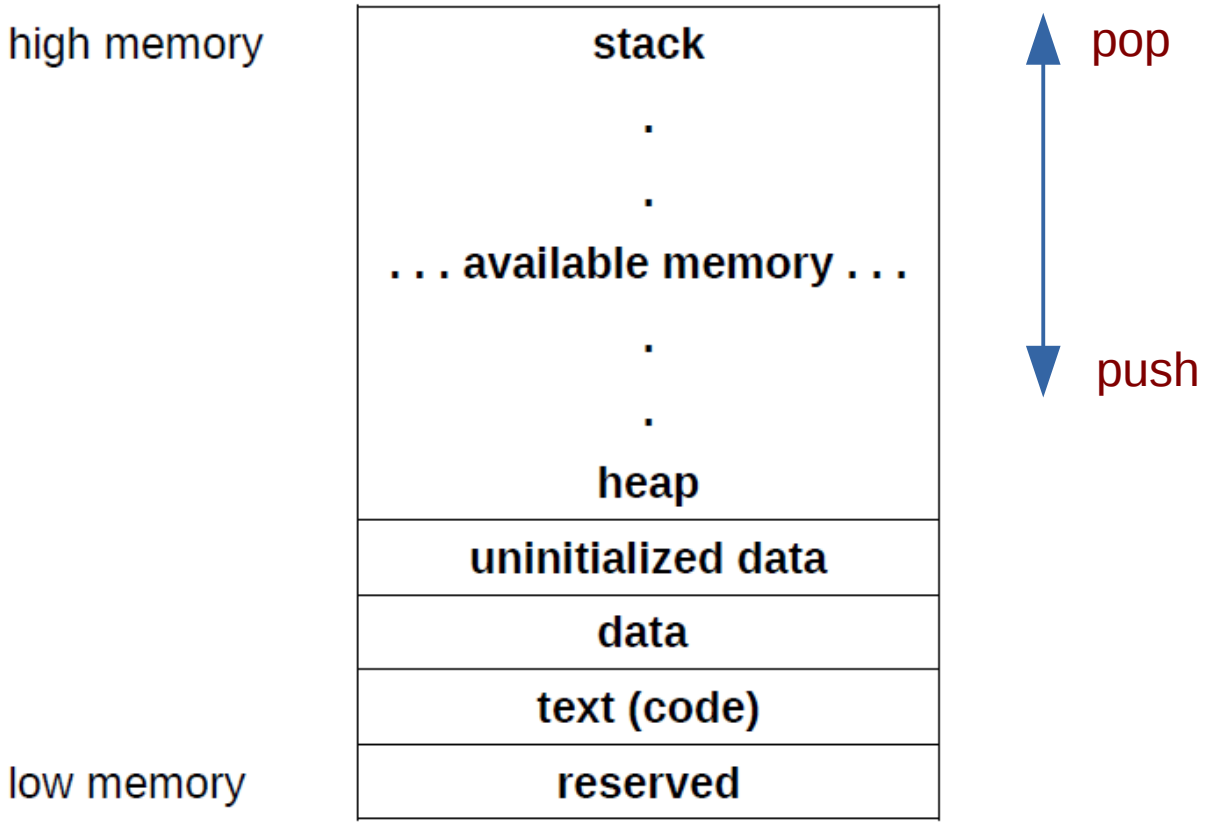
Stack

Stack is a data structure used to store data. In computer hardware, stack is a portion of memory reserved for storing and retrieving data via ***PUSH*** and ***POP*** instructions, respectively.



Stack Implementation

The **RSP** register is used to point to the current top of stack in memory. In X86 architecture, the stack is implemented growing downward in memory.



Stack Instructions

A push operation puts data at the top of stack (ToS), and a pop operation takes data off the top of stack. The format for these instructions are:

push <operand64>
pop <operand64>

Instruction	Explanation
push <op64>	Push the 64-bit operand on the stack. First, adjusts **rsp** accordingly (**rsp**-8) and then copy the operand to [**rsp**]. The operand may not be an immediate value. Operand is not changed.

Examples:	`push rax`
	`push qword [qVal] ; value`
	`push qVal ; address`

pop <op64>	Pop the 64-bit operand from the stack. Adjusts **rsp** accordingly (**rsp**+8). The operand may not be an immediate value. Operand is over-written.

Examples:	`pop rax`
	`pop qword [qVal]`
	`pop rsi`

Stack Operations

For a push operation:

1. The **RSP** register is decreased by 8 (1 quadword).
2. The operand is copied to the stack at **[RSP]**.

The operand is not altered. The order of these operations is important.

For a pop operation:

1. The current top of the stack, at **[RSP]**, is copied into the operand.
2. The **RSP** register is increased by 8 (1 quadword).

***** Important *****

Stack does not provide any mechanism for tracking elements, programmers must keep tracking of elements stored in stack.

Stack Example

watis@ThinkPad-E570 ~/Desktop/EN812700AssemblyLanguageProgramming/code

File Edit View Search Terminal Help

```
section .data
        ; Define constants
        EXIT_SUCCESS    equ 0   ; successful operation
        SYS_exit        equ 60  ; call code for terminate
        ; Define Data.
        numbers dq 121, 122, 123, 124, 125
        len dq 5
section .text
global _start
_start:
        ; Loop to put numbers on stack.
        mov rcx, qword [len]
        mov rbx, numbers
        mov r12, 0
        mov rax, 0
pushLoop:
        push qword [rbx+r12*8]
        inc r12
        loop pushLoop
        ; All the numbers are on stack (in reverse order).
        ; Loop to get them back off. Put them back into
        ; the original list...
        mov rcx, qword [len]
        mov rbx, numbers
        mov r12, 0
popLoop:
        pop rax
        mov qword [rbx+r12*8], rax
        inc r12
        loop popLoop

        mov rax, SYS_exit ; call code for exit
        mov rdi, EXIT_SUCCESS ; exit with success
        syscall
                                                1,1           All
```