Laboratory 9 Notes
X86 Stack

- Certain instructions implicitly modify the stack pointer (*push, pop, call, ret*)
- %rsp (stack pointer) always holds a pointer into the current stack frame

**push src**  
1. Make space on the stack by decrementing %rsp:  
   %rsp ← %rsp − 8

2. Move *src* to the stack:  
   (%rsp) ← src

---

**Initial state of the stack**

%rsp=0x7ffffff8

**Push** a word-size value in %rax on the stack  
(decrement %rsp and move Src to (%rsp)  
(assume %rax = 0x000000002030405)

**push %rax**

%rsp=0x7fffffff0

---

0x02030405
**pop dest**

1. Move contents of top of stack to the *dest*

   \[ \text{dest} \leftarrow \%\text{rsp} \]

2. Release space on the stack by incrementing \%rsp.

   \[ \%\text{rsp} \leftarrow \%\text{rsp} + 8 \]

---

**Initial State of Stack**

<table>
<thead>
<tr>
<th>%rsp=0x7fffffff0</th>
<th>0x02030405</th>
<th>0x02030405</th>
</tr>
</thead>
<tbody>
<tr>
<td>%rsp=0x7fffffff8</td>
<td>0x02030405</td>
<td>0x02030405</td>
</tr>
</tbody>
</table>

**Pop a word-size value from the stack.**

- Pop \%rbx
  - (%rbx gets 0x0000000002030405)

- \%rsp=0x7fffffff8
**call** function  1. Pushes the *return address* on stack (return address is the address of the instruction *following* the function call)
   \[
   \%\text{rsp} \leftarrow \%\text{rsp} - 8 \\
   (\%\text{rsp}) \leftarrow \%\text{rip} \text{ (already updated for next instruction)}
   \]

2. Puts the starting address of the *function* in \%rip:
   \[
   \%\text{rip} \leftarrow \text{starting address of function}
   \]

**ret**  1. Pops the return address from the top of the stack into \%rip (to resume execution of the *calling* function).
   \[
   \%\text{rip} \leftarrow (\%\text{rsp}) \\
   \%\text{rsp} \leftarrow \%\text{rsp} + 8
   \]
Conventions for drawing stack diagrams

To record the contents of the stack to understand how the stack is used, using the following notation:

- We use the model of memory where the stack has low addresses at the bottom and high at the top. Each row in the stack represents an 8-byte value. The initial \%rsp with a subscript of 0 is pointing to the top of the current stack frame.

- Trace the effect on the stack of executing each instruction in the program by moving the position of the \%rsp when it changes, (incrementing the subscript for each new value), and by recording new values on the stack as they are stored there.

- When the stack starts to empty, continue with the same notation, except use the right hand side of the stack diagram to indicate the changes.

- Also record changes to relevant registers.