Basic `MOV` Instruction

- All data transfer instructions require one source specifier `S` and one destination specifier `D`.
  - Restriction: Both `S` and `D` can't refer to memory locations.
  - Restriction: If `S` is immediate, it is at most 32 bits wide.
- Width of transfer is encoded by the ASM suffix of the opcode.
- `MOV`$\{B, W, L, Q\}$ `S, D`: $D \leftarrow S$
- `MOVABSQ I, R`: To handle a full 64-bit immediate source operand.

Width-Expanding `MOV` Instructions

- Used to transfer a narrower source operand to a wider destination, which must be a register.
- Widths of source and destination are encoded by the ASM suffix of the opcode.
- `MOVZ`$\{BW, BL, BQ, WL, WQ\}$ `S, D`: $D \leftarrow \text{ZeroExtend}(S)$. No `MOVZLQ`: why?
- `MOVS`$\{BW, BL, BQ, WL, WQ, LQ\}$ `S, D`: $D \leftarrow \text{SignExtend}(S)$
- `CLTQ`: `%rax` ← SignExtend(`%eax`)

Stack Manipulation Instructions

- `PUSHQ S`:

  R[%rsp] ← R[%rsp] – 8

  M[R[%rsp]] ← S

- `POPQ D`:

  D ← M[R[%rsp]]

  R[%rsp] ← R[%rsp] + 8

Unary and Binary Operations

Arithmetic		Logical		Shift	
`incX D`	$D \leftarrow D + 1$	`notX D`	$D \leftarrow \sim D$		
`decX D`	$D \leftarrow D - 1$				
`negX D`	$D \leftarrow -D$				

Arithmetic		Logical		Shift	
`addX S, D`	$D \leftarrow D + S$	`xorX S, D`	$D \leftarrow D \hat{} S$	`salX k, D`	$D \leftarrow D \ll k$
`subX S, D`	$D \leftarrow D - S$	`orX S, D`	$D \leftarrow D \mid S$	`shlX k, D`	$D \leftarrow D \ll k$
`imulX S, D`	$D \leftarrow D \times S$	`andX S, D`	$D \leftarrow D \& S$	`sarX k, D`	$D \leftarrow D \gg_A k$
				`shrX k, D`	$D \leftarrow D \gg_L k$

`X` $\in$ {`B`, `W`, `L`, `Q`}

Points To Note

- Since computational instructions can specify only two operands, one of them is also treated as the destination.
  - This leads to update-in-place semantics, where the old value at the destination is overwritten by the result value – and is therefore lost.
  - If it is necessary to preserve the old value, copy it explicitly (e.g., with a `MOV` instruction).
- The shift amount (in bits) can be specified either as an immediate or as the contents of register `%cl`.
  - If using `%cl`, only the necessary number of lower-order bits (as determined by the ASM suffix of the opcode) are used: 3 for `B`, 4 for `W`, 5 for `L`, 6 for `Q`.