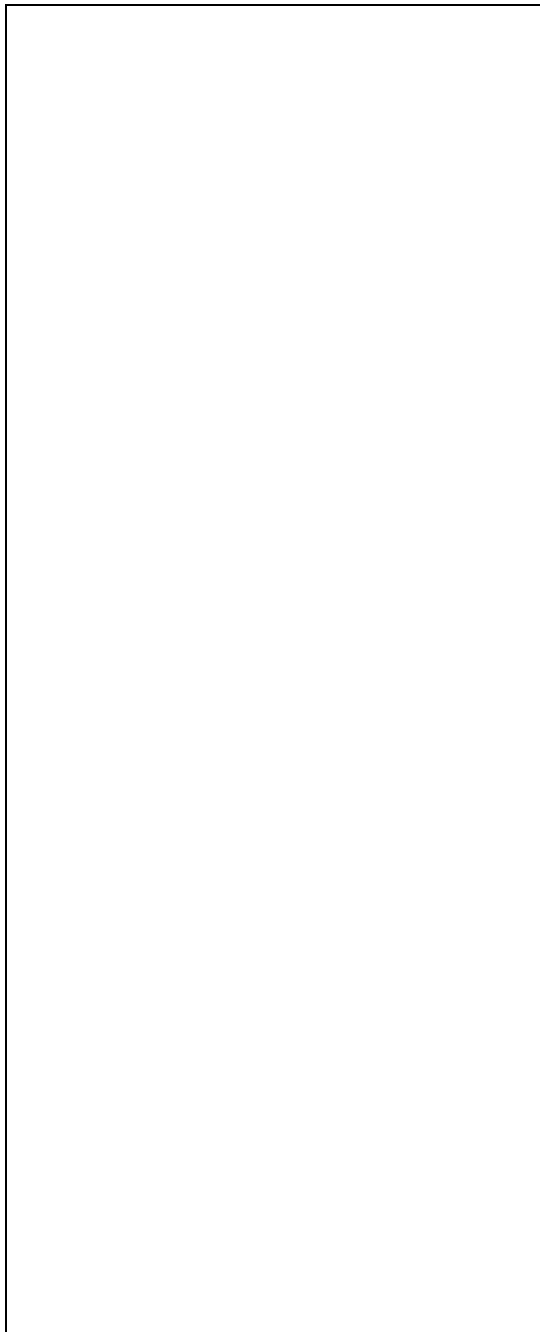


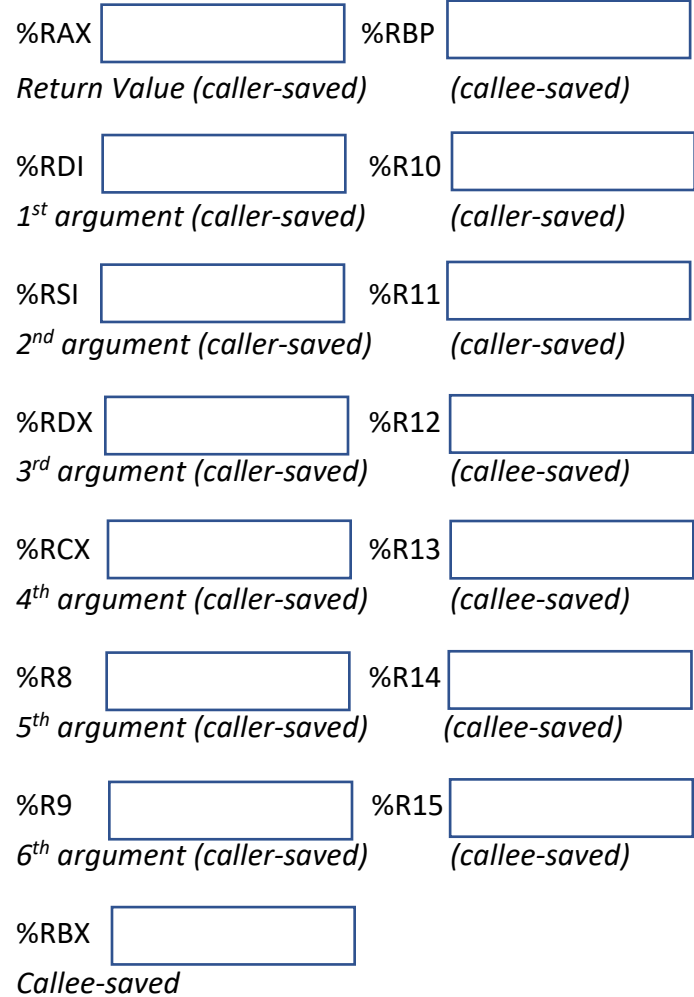
Higher Memory
Addresses

STACK DIAGRAM

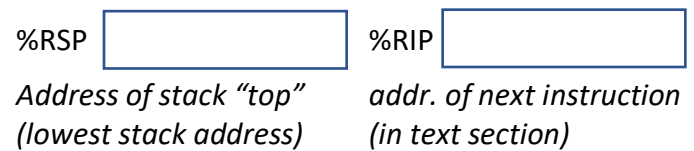


Lower Memory
Addresses

REGISTERS DIAGRAM



SPECIAL REGISTERS



COMMON INSTRUCTIONS

- mov a, b** – copy a into b
- movs a, b** – store sign-extended a into b
- movz a, b** – store zero-extended a into b
- lea a, b** – store address of memory addressing expression a in b

- push a** – push a onto stack
- pop a** – pop a value from the top of the stack into a

- call target** - push return address onto the stack and jump to target label/address
- ret** – pop return address from stack and jump there

- add a, b** – store sum a+b into b
- sub a, b** – store difference b-a into b
- imul a, b** – store signed product a*b into b
- and a, b** – store bitwise AND a&b into b
- or a, b** – store bitwise OR a|b into b
- shl/sal a, b** – store left shift b<<a into b
- shr a, b** - store logical right shift b<<a into b
- sar a, b** – store arithmetic right shift b<<a into b

- cmp a, b** – set condition codes based on difference b-a
- test a, b** – set condition codes based on bitwise AND a&b

- jg** – jump if greater than (zero)
- je** – jump if equal to (zero)
- jne** – jump if not equal to (zero)
- jle** – jump if less than or equal to (zero)
- jmp target** – jump to target

MEMORY ADDRESS SYNTAX

$D(R_b, R_i, S) \Rightarrow \text{Mem}[\text{Reg}[R_b] + S * \text{Reg}[R_i] + D]$
 S can only be 1, 2, 4, or 8

Remember that **lea** calculates an address but does not access the address.