Computer Systems Organisation

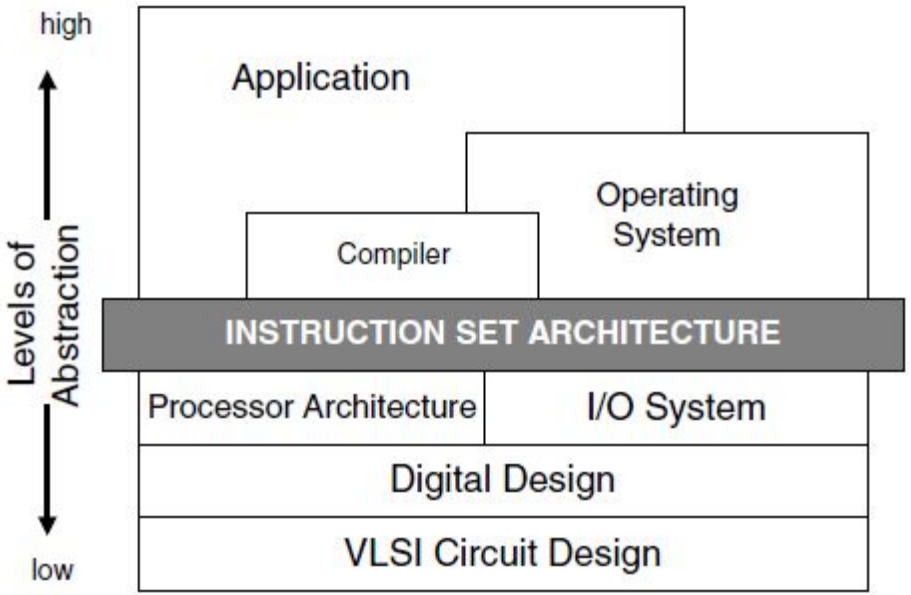
Tutorial 1

Prepared by: AadilMehdi Sanchawala, Rohan Chacko

Instruction Set Architecture

Instruction Set Architecture

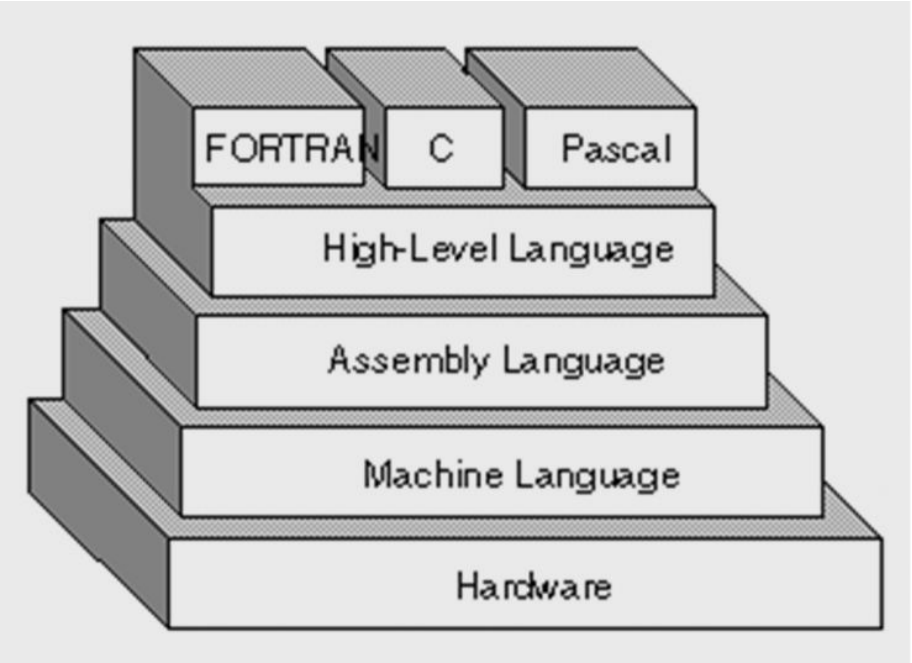
- ISA is an abstraction for the Software to interface with the Hardware.
- An instruction set architecture (ISA) is an abstract model of a computer.
- An ISA can have multiple realizations or implementations.
- It defines the supported data types, the registers, the hardware support for managing main memory fundamental features, and the input/output model of a family of implementations of the ISA.



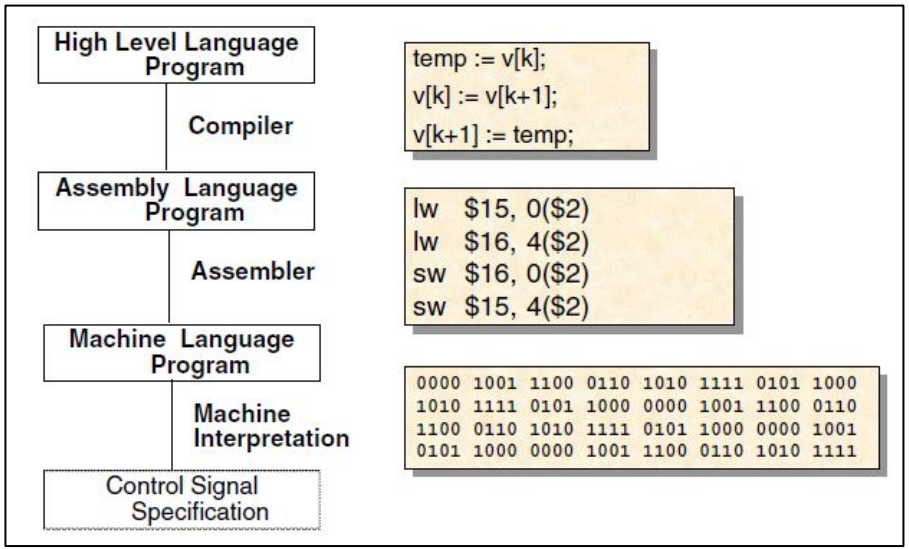
Levels of Programming Abstraction

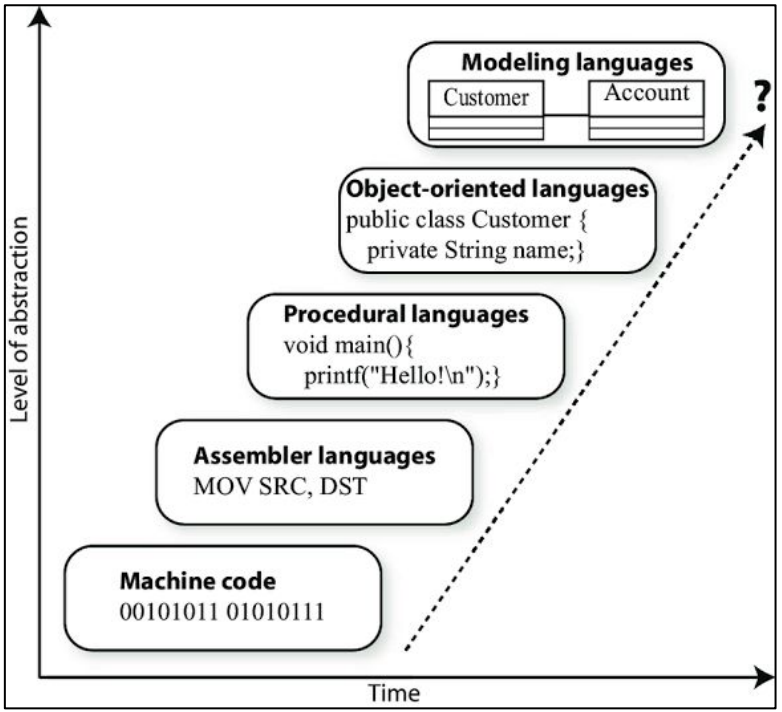
Levels of Programming Abstractions

- Programs can be written in various levels of abstraction,
  - High level like C++, python, C, Java
  - Assembly level Languages
  - Instruction Opcodes
  - Machine Languages
- As the level of abstraction increases, ease of programmability also increases. (Tradeoff: we lose the fine-grained control of the underlying hardware)



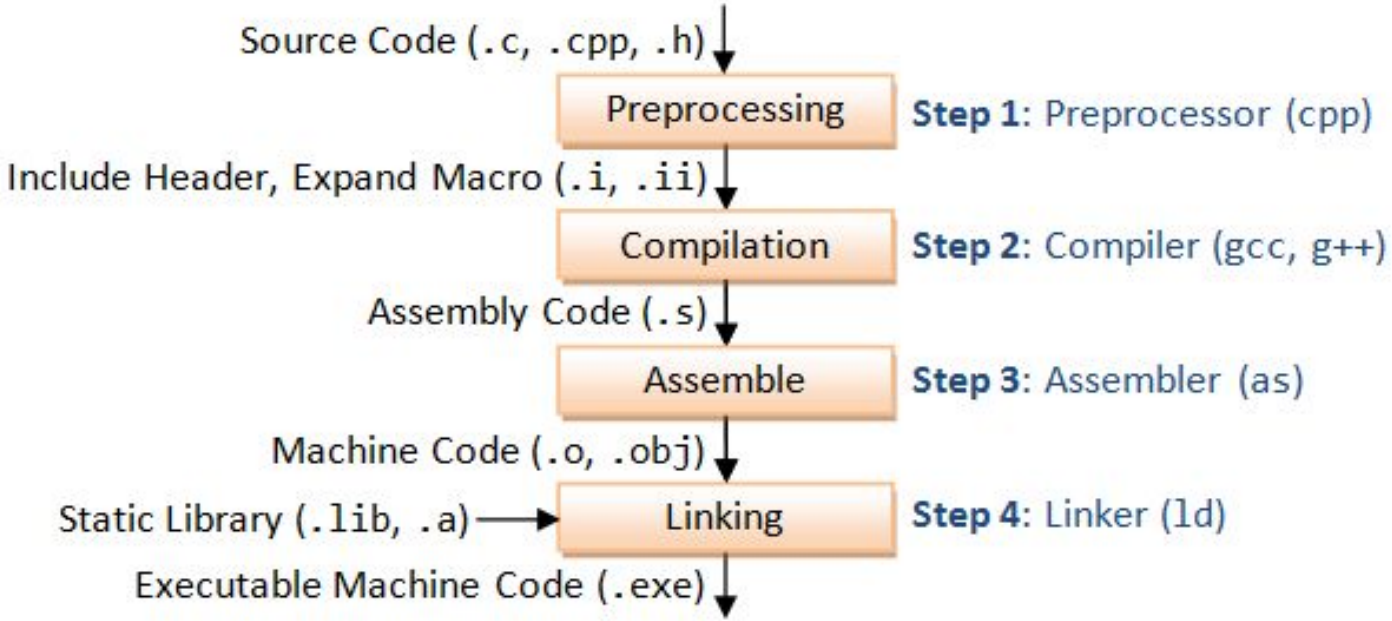
Levels of Programming Abstractions





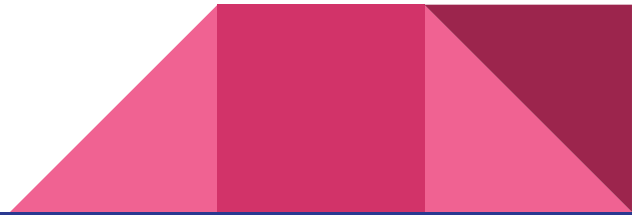
Steps of a program compilation

Steps of a program compilation



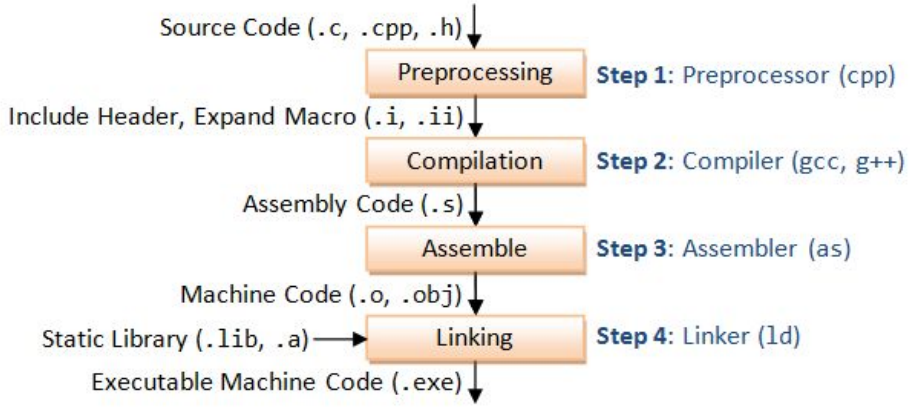
Steps of a program compilation

- Before the source code heads into compilation, the preprocessor takes the source code and strips away comments and interprets any preprocessor directives(such as macros and header files).
- Next, the C compiler will convert the preprocessed source code into assembly code(.s extension). (Viewable with gcc -S command.)
- The assembler then takes the assembly code and converts it into a binary representation of our program called an object file(with .obj in Windows and .o in UNIX systems). (Viewable with gcc -c command.)



Steps of a program compilation

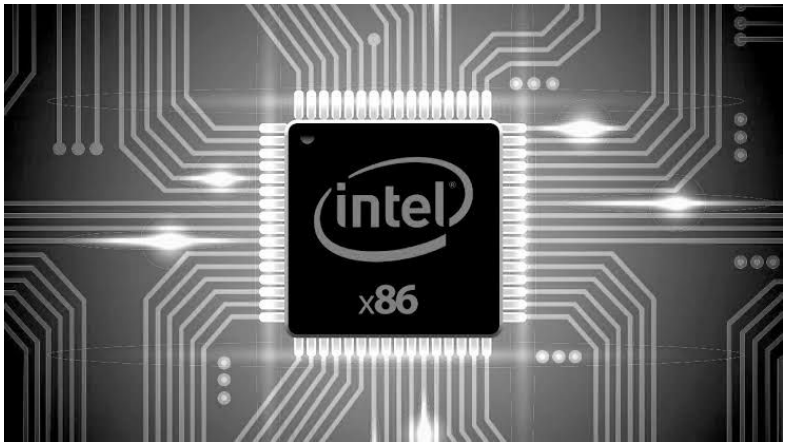
- In the Linking step, multiple object files will be linked together to create one executable. For example, we are using a standard C library function, printf() in our program to print our string. The linker will locate and include the associated code into the final output file.



Intel x86 Processors

Intel x86 Processors

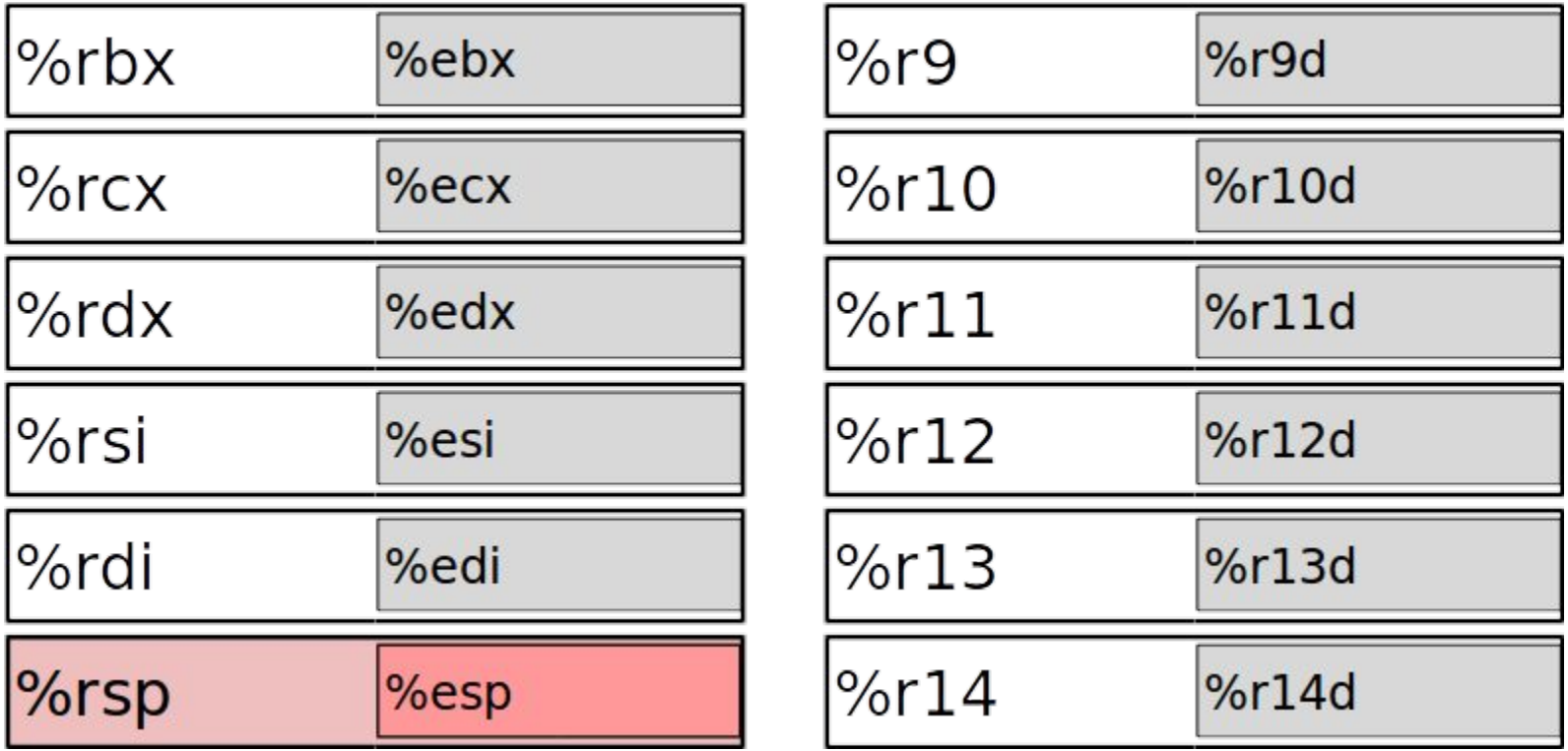
- Why Intel x86?
  - They dominate the market.
  - Present in almost every laptop, desktop and server.
  - Evolutionary design
- 32 bit ISA => **IA32**
- 64 bit ISA => **x86-64**



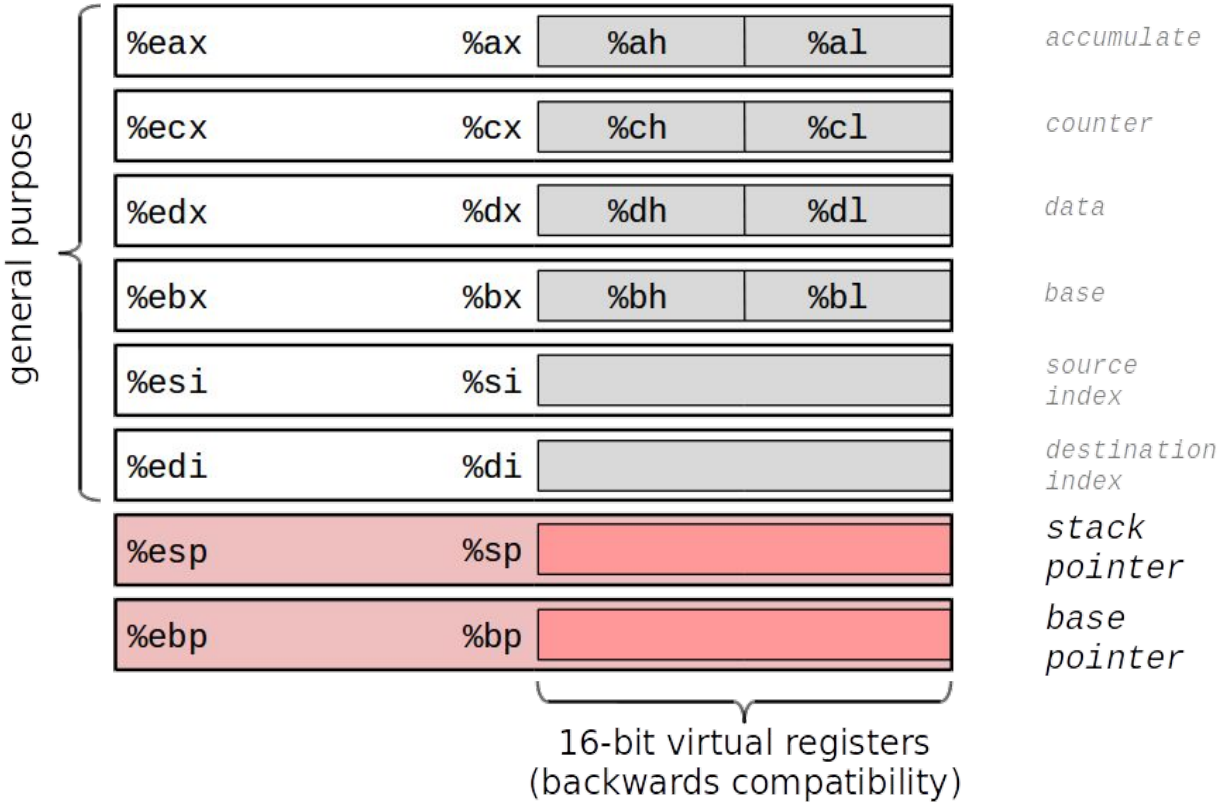
x86-64

x86-64 Registers

x86-64 Integer Registers



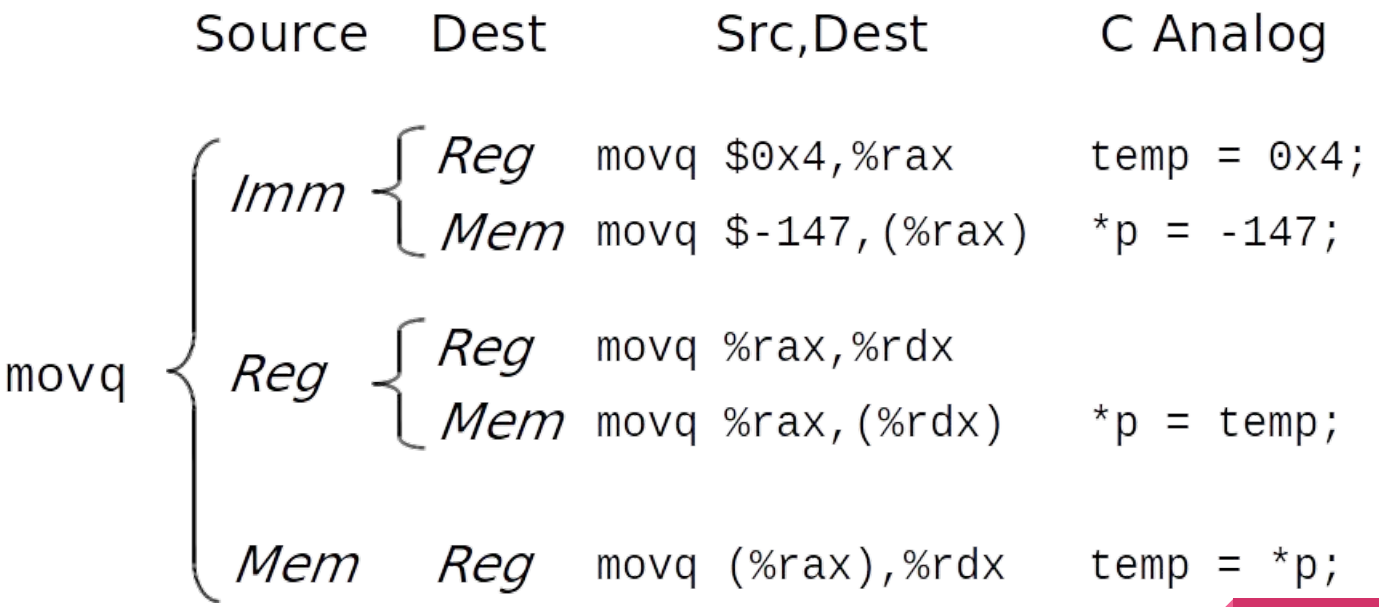
IA32 Integer Registers



x86-64 Data Transfer Instructions

movq

movq


Source
Dest
Src,Dest
C Analog
movq
Imm
Reg
movq $0x4,%rax
temp = 0x4;
Mem
movq $-147,(%rax)
*p = -147;
Reg
Reg
movq %rax,%rdx
Mem
movq %rax,(%rdx)
*p = temp;
Mem
Reg
movq (%rax),%rdx
temp = *p;

x86-64 Arithmetic and Logical Operators

x86-64 Arithmetic and Logical Operators - Two Operand Instructions

- addq Src, Dest Dest = Dest + Src
- subq Src, Dest Dest = Dest Src
- imulq Src, Dest Dest = Dest*Src
- salq Src, Dest Dest = Dest << Src (Also called shlq)
- sarq Src, Dest Dest = Dest >> Src (Arithmetic)
- shrq Src, Dest Dest = Dest >> Src (Logical)
- xorq Src, Dest Dest = Dest ^ Src
- andq Src, Dest Dest = Dest & Src
- orq Src, Dest Dest = Dest | Src

x86-64 Arithmetic and Logical Operators - Single Operand Instructions

- incq Dest Dest = Dest + 1
- decq Dest Dest = Dest 1
- negq Dest Dest = Dest
- notq Dest Dest = ~Dest