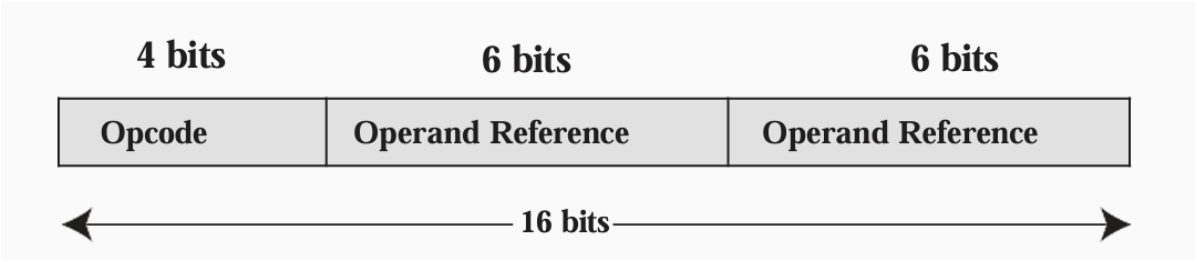
Lec-6-Instruction Sets: Characteristics and Functions（指令集：特性和功能）

ISA是什么

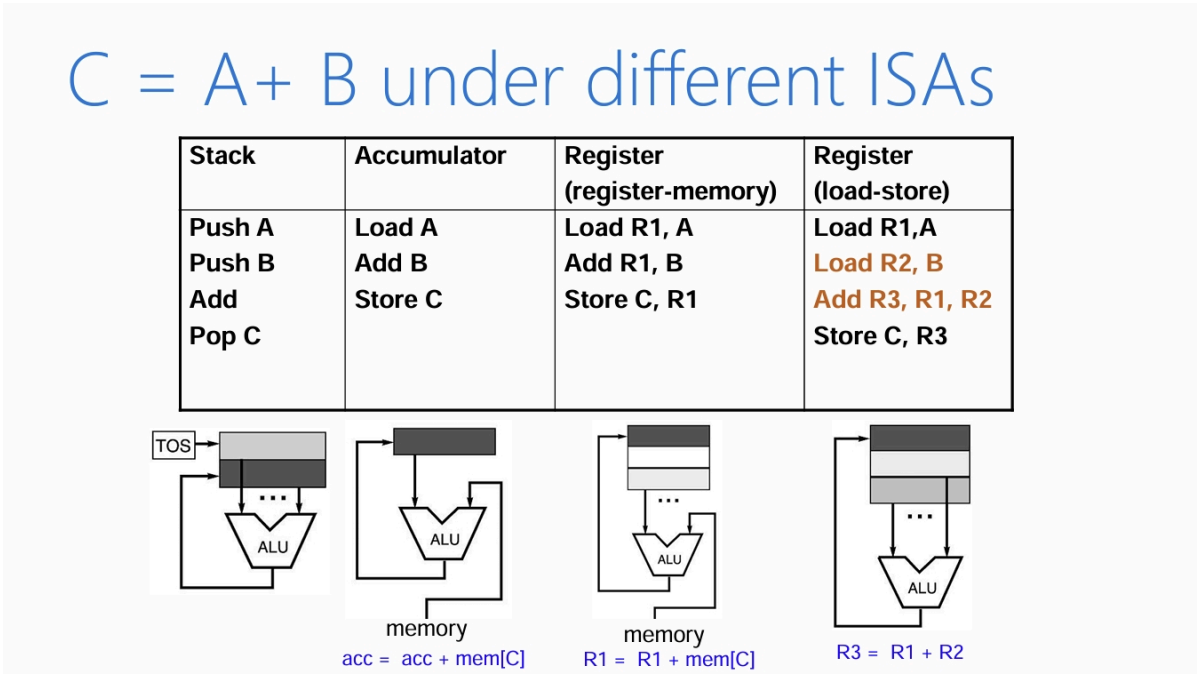
ISA是指令集架构（Instruction Set Architecture）的缩写，它定义了计算机硬件能够理解和执行的指令集。



1. **Operation code (Opcode)**:
   - 这是**操作码**，长度为 4 位。操作码告诉计算机要执行什么操作，例如加法、减法、乘法等。
2. **Source operand reference**:
   - 这是源**操作数**引用，长度为 6 位。它指定了操作数的来源，即操作的数据从哪里获取。
3. **Result operand reference**:
   - 这是结果操作数引用，长度为 6 位。它指定了操作结果应该存储在哪里。

Different ISAs (stack, accumulator, register-memory, register-register)

这个了解就好，个人感觉不会考



在计算机体系结构中，堆栈架构（Stack Architecture）是一种指令集架构，其中操作数通常存储在堆栈中，而不是在寄存器中。以下是关于堆栈架构的一些优缺点的详细解释：

优点

1. **紧凑的代码**：
   - **原因**：在堆栈架构中，指令通常不需要明确指定操作数的位置，因为操作数总是位于堆栈的顶部。例如，一个加法操作只需要一个指令（如 `ADD`），而不需要指定操作数的位置，因为操作数已经隐含地存储在堆栈的顶部.
   - **结果**：这使得指令本身可以更短，从而生成更紧凑的代码.
2. **简单的操作**：
   - **原因**：`PUSH` 和 `POP` 操作只需要一个显式操作数，即要压入或弹出的数据。这简化了指令的格式和执行过程.
   - **结果**：指令的编码和解码过程更简单，硬件实现也更为直接.
3. **低硬件要求**：
   - **原因**：由于指令格式简单且操作数管理主要依赖于堆栈，因此对硬件的要求相对较低。不需要复杂的寄存器管理逻辑.
   - **结果**：可以降低硬件的复杂性和成本.
4. **编译器编写简单**：
   - **原因**：堆栈架构的指令集相对简单，编译器不需要处理复杂的寄存器分配和优化问题.
   - **结果**：编写编译器相对容易，尤其是在早期计算机技术发展阶段，编译器的开发成本较低.

缺点

1. **堆栈成为瓶颈**：
   - **原因**：所有操作都依赖于堆栈的顶部，因此频繁的操作会导致堆栈的频繁访问和更新.
   - **结果**：在高并发或多任务环境中，堆栈的访问速度可能成为性能瓶颈，影响整体执行效率.
2. **流水线能力有限**：

    - **原因**：堆栈架构的指令执行依赖于堆栈的状态，指令之间存在较强的依赖关系，难以实现指令的并行执行.
    - **结果**：难以有效地实现流水线技术，限制了处理器的性能提升.
3. **优化编译器难以编写**：
    - **原因**：由于指令执行依赖于堆栈的状态，编译器需要考虑更多的上下文信息来进行优化，如指令重排、寄存器分配等.
    - **结果**：编写能够有效优化代码的编译器变得更加困难，因为需要处理复杂的堆栈操作和指令依赖关系.

总的来说，堆栈架构在某些简单应用和早期计算机中具有一定的优势，但在现代高性能计算环境中，其缺点逐渐显现，因此现代处理器通常采用更复杂的寄存器架构（如RISC和CISC）来提高性能和灵活性.

RISC（精简指令集计算） vs CISC（复杂指令集计算）

RISC

- Simple implementation
- Register-register, fixed-format 32-bit instructions, efficient pipelines(寄存器-寄存器，固定格式32位指令，高效的流水线)
- Compilers do the hard work(编译器承担繁重的工作)

CISC

- Simple compilers (assists hand-coding, many addressing modes, many specialised instructions)

  简单的编译器（协助手工编码，多种寻址模式，许多专用指令）：

  编译器相对简单，可以协助手工编码，支持多种寻址模式和许多专用指令.
- Code compactness(生成的代码更加紧凑)

Byte ordering（字符排序）

- **Big Endian（大端序）**：
    - 最低位字节具有最高的地址.
- **Little Endian（小端序）**：
    - 最低位字节具有最低的地址.