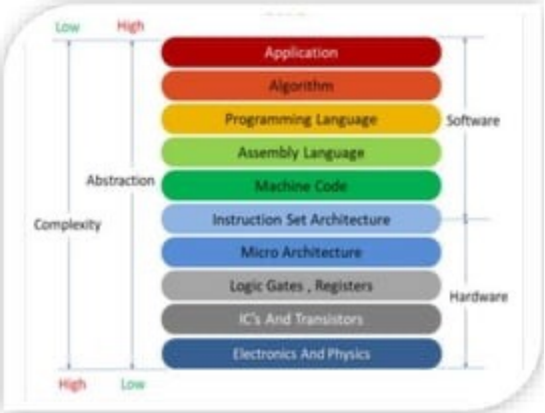
Instruction Set Architecture

INSTRUCTION SET ARCHITECTURE (ISA)

ISA (instruction set architecture)

- A well-defined hardware/software interface
- The **"contract"** between software and hardware
- **Functional definition** of storage locations & operations
- Storage locations: registers, memory
- Operations: add, multiply, branch, load, store, etc
- **Precise description** of how to invoke & access them

- How operations are implemented
- Which operations are fast and which are slow and when
- Which operations take more power and which take less
- Instructions
  - Bit-patterns hardware interprets as commands
  - Instruction → Insn (instruction is too long to write in slides)



A LANGUAGE ANALOGY FOR ISAS

Communication

- Person-to-person → software-to-hardware

Similar structure

- Narrative → program
- Sentence → insn
- Verb → operation (add, multiply, load, branch)
- Noun → data item (immediate, register value, memory value)
- Adjective → addressing mode

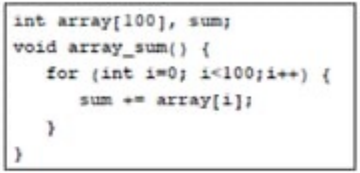
Many different languages, many different ISAs

- Similar basic structure, details differ (sometimes greatly)

Key differences between languages and ISAs

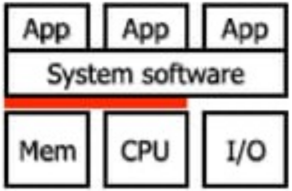
- Languages evolve organically, many ambiguities, inconsistencies
- ISAs are explicitly engineered and extended, unambiguous

PROGRAM COMPILATION


```
int array[100], sum;
void array_sum() {
   for (int i=0; i<100;i++) {
      sum += array[i];
   }
}
```

Program written in a "high-level" programming language
- C, C++, Java, C#
- Hierarchical, structured control: loops, functions, conditiona
- Hierarchical, structured data: scalars, arrays, pointers, structures

Compiler: translates program to **assembly**
- Parsing and straight-forward translation
- Compiler also optimizes
- Compiler itself another application … who compiled compile



INSTRUCTION EXECUTION MODEL

The computer is just finite state machine

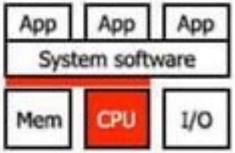
- **Registers** (few of them, but fast)
- **Memory** (lots of memory, but slower)
- **Program counter** (next insn to execute)
    - Called "instruction pointer" in x86

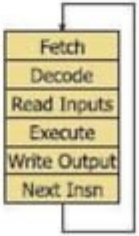
A computer executes **instructions**

- **Fetches** next instruction from memory
- **Decodes** it (figure out what it does)
- **Reads** its **inputs** (registers & memory)
- **Executes** it (adds, multiply, etc.)
- **Write** its **outputs** (registers & memory)
- **Next insn** (adjust the program counter)

Program is just "data in memory"

- Makes computers programmable ("universal")





Instruction → Insn

THE SEQUENTIAL MODEL

Basic structure of all modern ISAs

- Often called VonNeuman, but in ENIAC before

Program order: total order on dynamic insns

- Order and named storage define computation

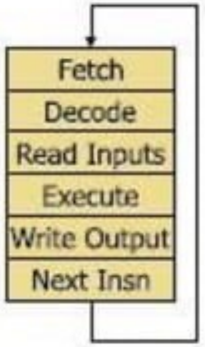
Convenient feature: **Program counter (PC)**

- Insn itself stored in memory at location pointed to by PC
- Next PC is next insn unless insn says otherwise

Processor logically executes loop at left

Atomic: insn finishes before next insn starts

- Implementations can break this constraint physically
- But must maintain illusion to preserve correctness



WHAT MAKES A GOOD ISA?

Central Processing Unit (CPU) , Microprocessor

Programmability

- Easy to express programs efficiently?

Performance/Implementability

Easy to design high-performance implementations?

More recently

Easy to design low-power implementations?

Easy to design low-cost implementations?

Compatibility

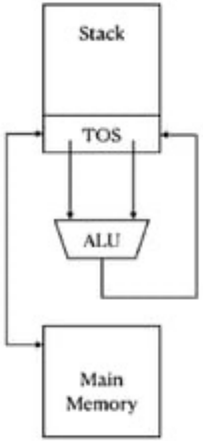
Easy to maintain as languages, programs, and technology evolve?

x86 (IA32) generations: 8086, 286, 386, 486, Pentium, PentiumII,
PentiumIII, Pentium4, Core2, Core i7, ...

TYPES OF INSTRUCTION SETS

1. Stack Architecture:

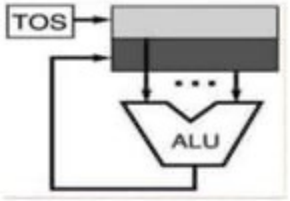
- The operands are put into the stack. Operations take place at the top two locations on the stack, destroying the operands, and leaving the result on the top of the stack.
- Two operations, **push and pop**, have one operand. Push a value at the top of the stack, or pop the top element of the stack to a destination.
- This architecture was used in computers in 1960s.



TYPES OF INSTRUCTION SETS

1. Stack Architecture:

Suppose we want to ask a stack architecture machine to conduct the operation C = A + B, where A, B, and C are memory addresses. The result of A + B are stored at location C.
- Then, to implement the operation in a stack architecture, we
need the following steps:



PUSH A # push value at location A to the top of the stack

PUSH B # push value at location B to the top of the stack

ADD # add the top two elements of the stack and save the result back to the top of the stack

POP C # store the value at the top of the stack (which is the result) to location C

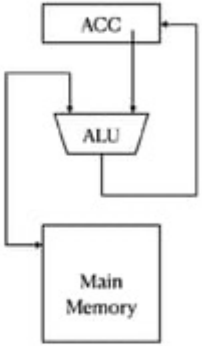
TYPES OF INSTRUCTION SETS

2. Accumulator Architecture

It places one operand in the accumulator and one in memory. The one in the accumulator is implicit, and the one in the memory is an explicit memory address. The result of ALU is written back to the accumulator.

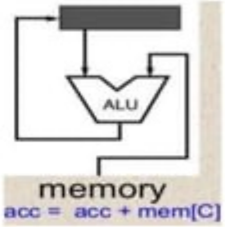
- The operand in the accumulator is loaded from memory using the command LOAD, and the result is stored in memory from accumulator using the command STORE.

- This architecture was used by early computers like IBM 7090. Today, it's used by some microprocessor chips, such as some digital signal processors.



TYPES OF INSTRUCTION SETS

2. Accumulator Architecture



Suppose we want to ask an accumulator architecture machine to conduct the same operation C = A + B as above.
- Then, to implement the operation in an accumulator architecture, we need the following steps:

LOAD A # load value at location A from main memory to ACC

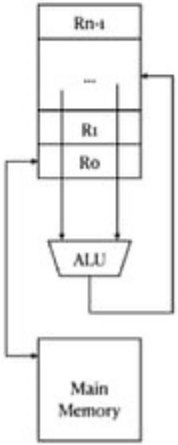
ADD B # fetch the value at location B, add it with the value at ACC (which is the value A), and save the result back to ACC

STORE C # store the value at ACC (which is the result of A + B) to location C

TYPES OF INSTRUCTION SETS

3. Register-Set Architecture: Modern CPUs have a number of **general-purpose registers (GPR's)** for internal storage: at least 8 and as many as 32.

- GPR machines are the most flexible of all, because they allow the user to access any of several registers.
- The load/store commands must typically specify a register and memory location, to indicate both source and destination of the data.
- Any register can be used as an accumulator, an index register, or for temporary storage, etc., so many variables and addresses can be manipulated without extra memory accesses.
- The major bad point of having many registers is the large amount of internal CPU state that must be saved/restore for calls, returns, and interrupts.
- There are 8 general-purpose registers in the 8086 microprocessor.

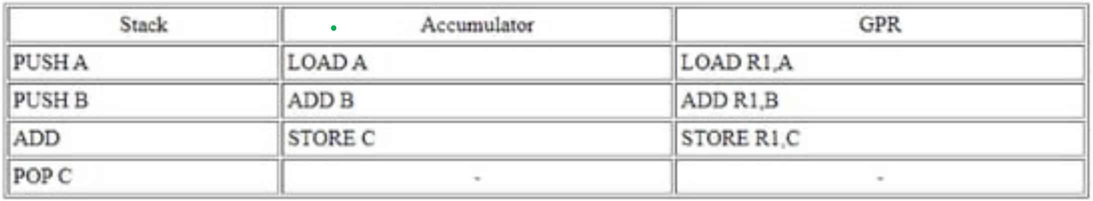


EXAMPLE

Lets look at the assembly code of

C = A + B;

in all 3 architectures:



Stack	Accumulator	GPR
PUSH A	LOAD A	LOAD R1,A
PUSH B	ADD B	ADD R1,B
ADD	STORE C	STORE R1,C
POP C	-	-

CISC & RISC ARCHITECTURE

Need Of CISC:

In the past, it was believed that hardware design was easier than compiler design

- Most programs were written in assembly language

❖ Hardware concerns of the past:
  - Limited and slower memory
  - Few registers

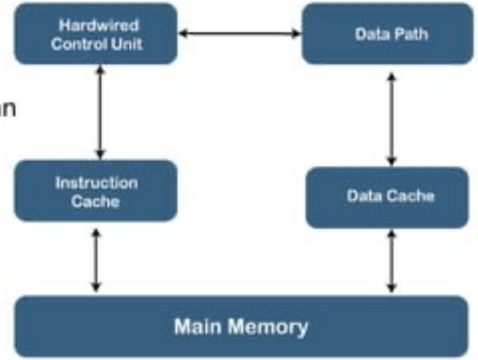
Register:

Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as Processor registers.

CISC & RISC ARCHITECTURE

CISC, which stands for Complex Instruction Set Computer.

- Each instruction executes multiple low level operations.
- Ex. A single instruction can load from memory, perform an arithmetic operation, and store the result in memory.
- Smaller program size.



RISC Architecture

CISC & RISC ARCHITECTURE

CISC Characteristics

- A large number of instructions.
- Some instructions for special tasks used infrequently.
- A large variety of addressing modes (5 to 20).
- Variable length instruction formats.

Disadvantages :

However, it soon became apparent that a complex instruction set has a number of disadvantages:

- These include a complex instruction decoding scheme, an increased size of the control unit, and increased logic delays.

CISC & RISC ARCHITECTURE

RISC: RISC Stands for Reduced Instruction Set Computer.

- It is a microprocessor that is designed to perform a smaller number of types of computer instruction so that it can operate at a higher speed.

RISC Characteristics :

- Relatively few instructions
  - 128 or less
- Relatively few addressing modes.
- Memory access is limited to LOAD and STORE instructions.
- All operations done within the registers of the CPU.
  - This architectural feature simplifies the instruction set and encourages the optimization of register manipulation.

  - An essential RISC philosophy is to keep the most frequently accessed operands in registers and minimize register-memory operations.

CISC & RISC ARCHITECTURE

CISC

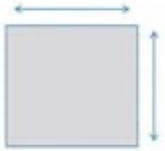
- Complex Instructions.
- ADD AX,[BX + SI + 600H]
- Many operations in single instruction.

RISC

- Simpler or reduced instructions.
- LOAD R1, addresss1
  LOAD R2, address2
  ADD R1, R2
  STORE address1, R1
- One instruction one operation.

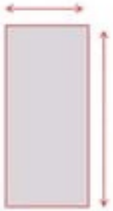
CISC & RISC ARCHITECTURE

CISC



Code size is smaller but complicated.

RISC



Code size is larger but simpler.

CISC & RISC ARCHITECTURE

CISC	RISC
▪ Fewer register.	▪ Large number of registers.
▪ These registers are designed for special purposes.	▪ Here registers are identical so any register can be used for any purpose.
▪ CISC designs provide a large number of addressing modes.	▪ RISC designs have single addressing modes.

CISC & RISC ARCHITECTURE

CISC	RISC
▪ Slower to execute.	▪ Faster execution.
▪ Difficult to decode.	▪ Easy to decode.
▪ Instruction size varies in different instructions.	▪ Same instruction size in every instructions.
▪ Complex circuit design.	▪ Circuit design is simpler.

CISC & RISC ARCHITECTURE

LOW END & MOBILE SYSTEM

- ARM ARCHITECTURE
  - Android based systems/ Apple iPhone/ Nintendo GBA etc.
- MIPS line
  - in PlayStations, Nintendo 64 etc.
- Atmel AVR
  - Xbox handheld controllers to BMW cars

HIGH END RISC & SUPERCOMPUTING

- MIPS
  - used in embedded system in routers, used by Digital Equipment Corporation etc.
- IBM'S Power Architecture
  - In many IBM's supercomputers, workstations etc.
- Alpha
  - In Single-board computers, Servers & Supercomputers from Digital Equivalent Cooperation etc.

CISC & RISC ARCHITECTURE

ARM BASED PRODUCTS.

Lego Mindstrome Robot
ARM7

Apple iPhone
ARM11

Paison Series game consoles
ARM7TDMI

ARM Architecture

- Developed by Advanced RISC Machines (ARM).
- ARM makes 32-bit & 64-bit RISC multi-core processors.
- Features of ARM architecture:
  - A **load/store** architecture
  - An **orthogonal** instruction set.
  - Fixed instruction width
  - Mostly single clock-cycle execution.
  - Enhanced power-saving design.
  - Hardware virtualization supports.

QUESTIONS:

- **Q1: What is Instruction Set Architecture?**
  - An Instruction Set Architecture (ISA) is **part of the abstract model of a computer that defines how the CPU is controlled by the software**. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.
- **Q2: What is Instruction Execution Model?**
  - The three cycle instruction. execution model. • **To execute a program, the microprocessor "reads" each instruction from memory, "interprets" it, then "executes" it.**
- **Q3: What is Stack Architecture?**
  - The stack is **a list of data words**. It uses the Last In First Out (LIFO) access method which is the most popular access method in most of the CPU. A register is used to store the address of the topmost element of the stack which is known as Stack pointer (SP).

QUESTIONS:

- **Q4: Difference between CISC and RISC.**
  - The primary difference between RISC and CISC architecture is that **RISC-based machines execute one instruction per clock cycle**. RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program. In a **CISC processor**, each instruction performs so many actions that it takes several clock cycles to complete. The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction.
- **Q5: What is Register?**
  - A processor register (CPU register) is **one of a small set of data holding places that are part of the computer processor**. A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters). Some instructions specify registers as part of the instruction.