PES UNIVERSITY ONLINE

RISC V Architecture

Mahesh Awati
Department of Electronics and Communication Engg.

RISC V ARCHITECTURE

UNIT 2 – Instructions: The Language of Computer

Mahesh Awati

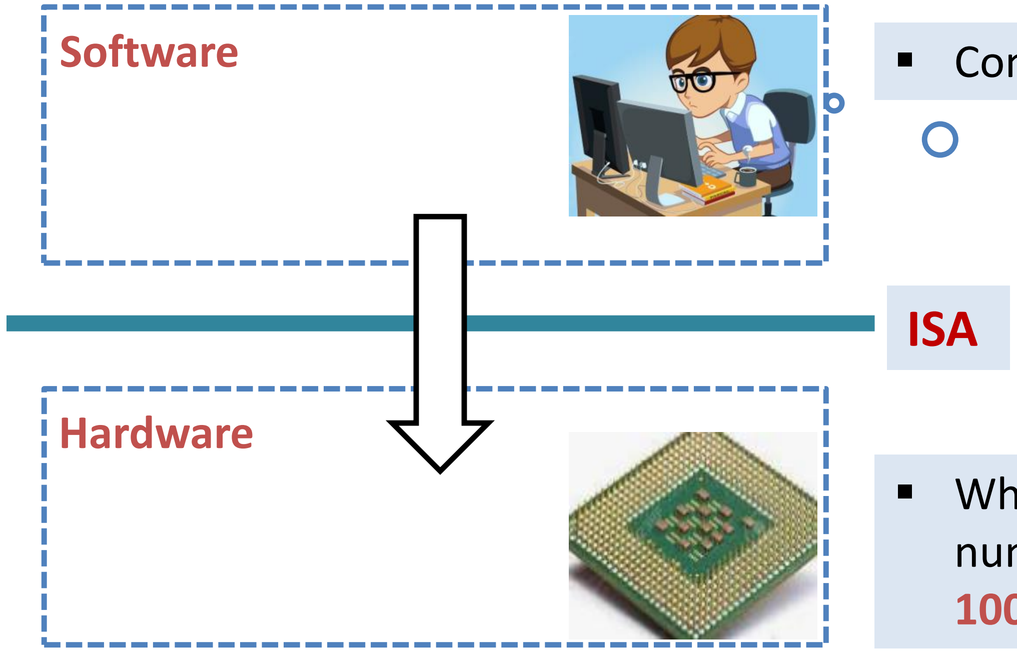
Department of Electronics and Communication Engineering

Instructions – Language of Computer

Introduction

Computer hardware understands computer language called **Instructions**, and it's vocabulary is called an **Instruction set**

Starting from a **notation that looks like a restricted programming** language (written by people), we refine it step-by-step until you see the **actual language** of **a real computer** (read by the computer)



- Computers are slaves to our commands

Can we Imagine of writing commands in binary ?

- What computers understand is numbers in base 2, or *binary numbers*.
  1001010100101110 – Function x

Instructions – Language of Computer

Introduction

Computer hardware understands computer language called **Instructions**, and it's vocabulary is called an **Instruction set**

- **Different computers have different instruction sets.** But with **many aspects in common / similar**
- This **similarity** of instruction sets occurs because **all computers are constructed from hardware technologies based on similar underlying principles** and because there are a few basic operations that all computers must provide.
- Early computers had very **simple instruction sets**
  - Simplified implementation
- Many modern computers also have simple instruction sets

Computer designers have a common goal:

✓ To find a **language that makes** it **easy to build the hardware and the compiler** while maximizing performance and minimizing cost and energy.

Instructions – Language of Computer

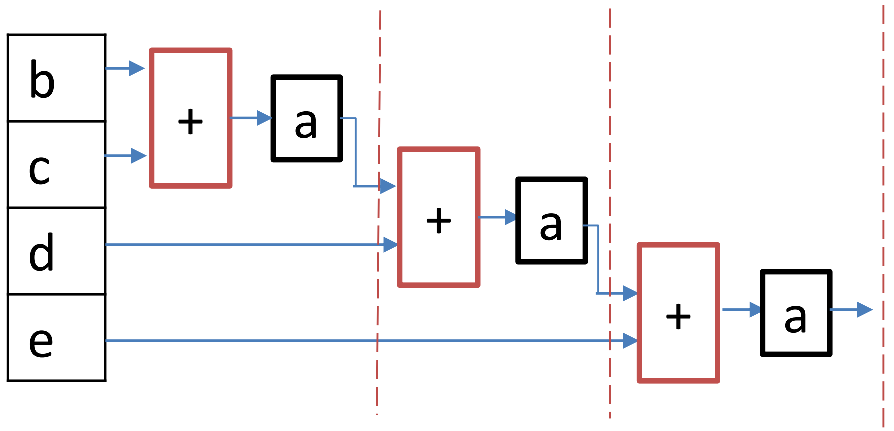
Operations of the Computer

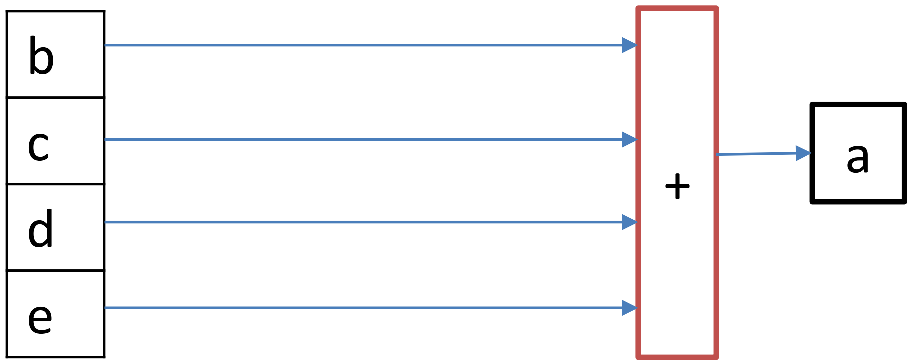
Hardware

- There must certainly be instructions for performing the fundamental arithmetic operations
- Every computer must be able to perform arithmetic. The RISC-V assembly language notation

  add a, b, c // Adds the two variables b and c and stores the sum in a
- Each RISC-V **arithmetic instruction performs only one operation and must always have exactly three variables.**
- Suppose we want to place the sum of four variables **b, c, d, and e** into **variable a**. How it can be done ?

Does RISC V perform
add a, b, c, d, e   ?????





RISC V can't perform operation on more than two variables

Operations of the Computer

Hardware

- The natural number of operands for **an operation like addition is three:** the two numbers being added together and a place to put the sum.

Why Not more than two variables ???

- **Conforms to the philosophy of keeping the hardware simple**: hardware for a **variable number of operands is more complicated** than hardware for a fixed number.

- This situation illustrates the **first of three underlying principles of hardware design:**

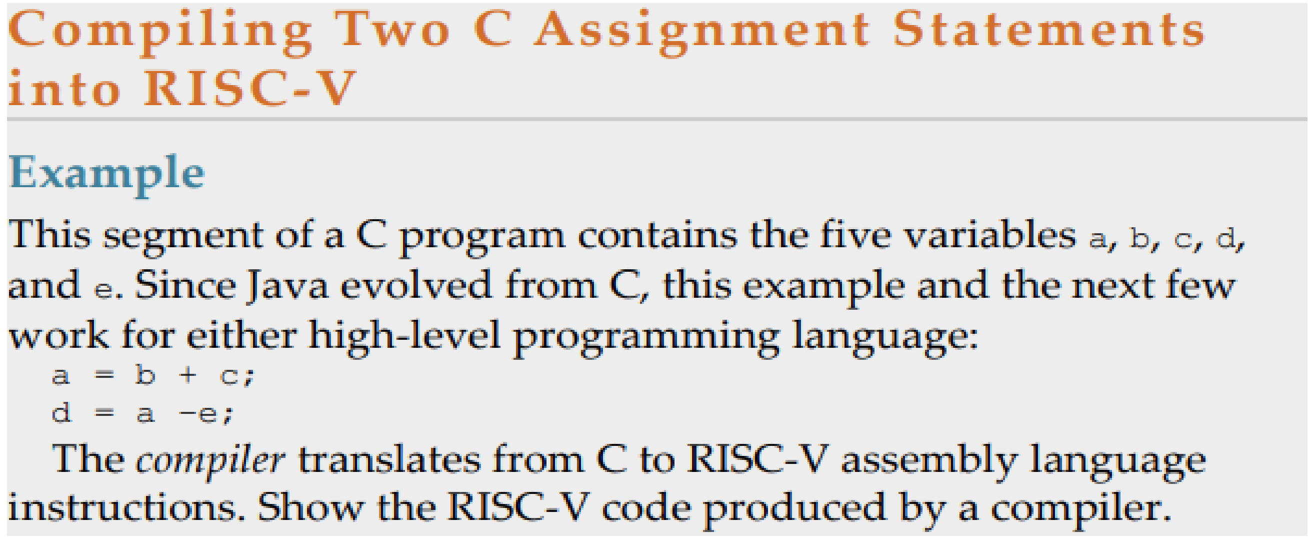
Design Principle 1: Simplicity favours Regularity.

Instructions – Language of Computer

Operations of the Computer

Hardware

Relationship of programs written in higher-level programming languages to programs in this more primitive notation.



Compiling Two C Assignment Statements into RISC-V

Example

This segment of a C program contains the five variables `a`, `b`, `c`, `d`, and `e`. Since Java evolved from C, this example and the next few work for either high-level programming language:

```
a = b + c;
d = a -e;
```

The *compiler* translates from C to RISC-V assembly language instructions. Show the RISC-V code produced by a compiler.

```
add a, b, c
sub d, a, e
```

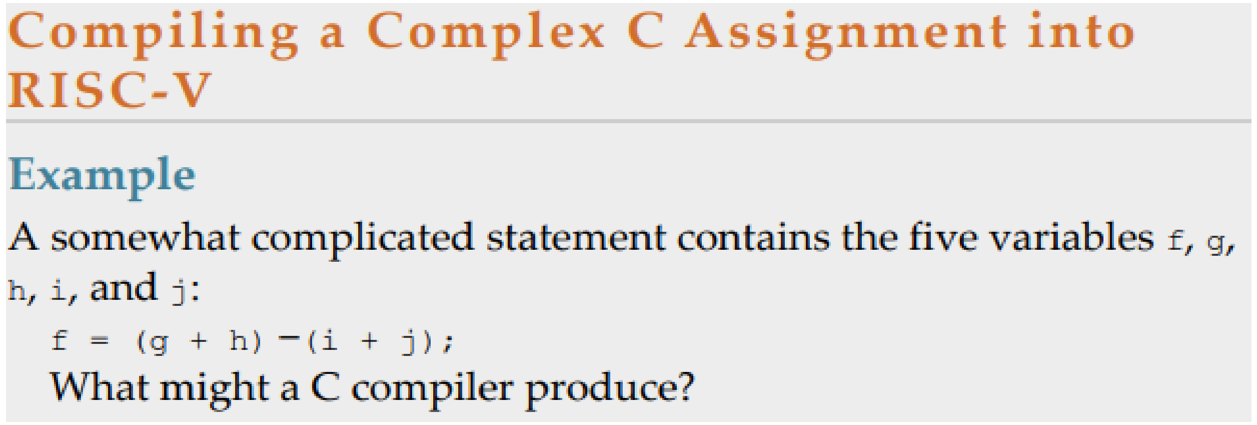
Reference : Computer Architecture with RISC V - The Hardware/Software Interface: RISC-V Edition by David A. Patterson and John L. Hennessy

Instructions – Language of Computer

Operations of the Computer

Hardware

Relationship of programs written in higher-level programming languages to programs in this more primitive notation.



Compiling a Complex C Assignment into RISC-V

Example

A somewhat complicated statement contains the five variables `f`, `g`, `h`, `i`, and `j`:

```
f = (g + h) − (i + j);
```

What might a C compiler produce?

```
f = (g + h) −(i + j);
```

```
add t0, g, h
add t1, i, j
sub f, t0, t1
```

Does Compiler generate Single Assembly Instruction ????

- The compiler must break this statement into several assembly instructions, since only one operation is performed per RISC-V instruction.
- The compiler creates a temporary variable, called t0,t1.. so on to store intermediate results

PES UNIVERSITY ONLINE

THANK YOU

Mahesh Awati
Department of Electronics and Communication

mahesha@pes.edu

+91 9741172822