

# **Computer Organization**

Instruction Set Architecture

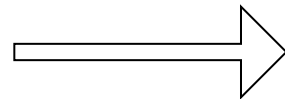
B.Tech. II (CSE)

# Instruction Set Architecture

## ■ C code

A=b+c;

D=e+f;

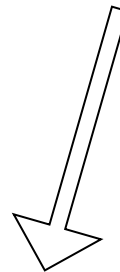


Compiler

## ■ Assembly Code

Add \$s3, \$s2, \$s1

Add \$s7, \$s5, \$s6



Encoding straight  
forward

## ■ Machine Code

...0...1..

...0...1..

# Instruction Set Architecture

## ■ C code

A=b+c;

D=e+f;

Machine  
Independent

## ■ Assembly Code

Add \$s3, \$s2, \$s1

Add \$s7, \$s5, \$s6

Compiler

Machine  
Dependent

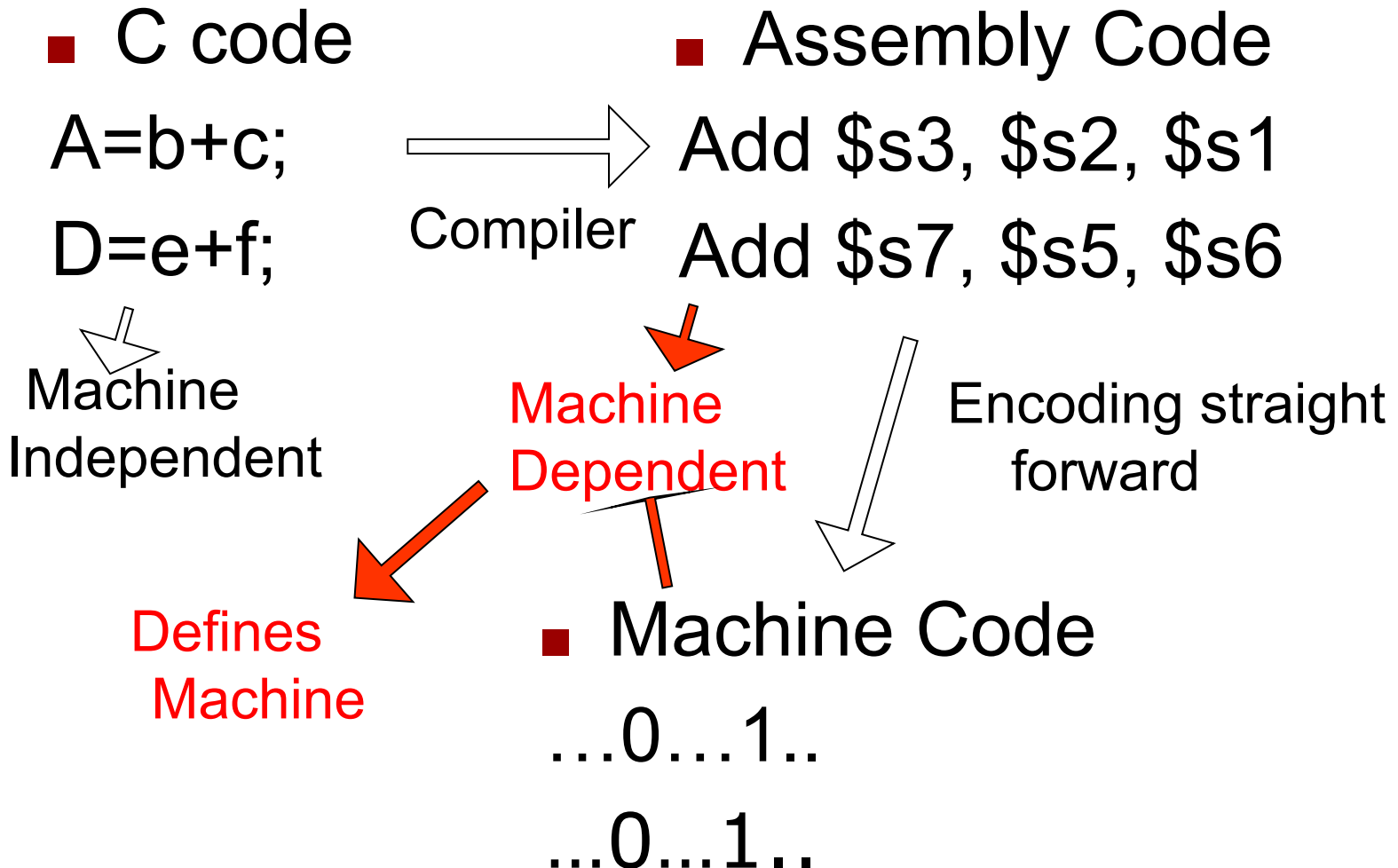
Encoding straight  
forward

Defines  
Machine

## ■ Machine Code

...0...1..

...0...1..



# Instruction Set Architecture

■ C code

A=b+c;

D=e+f;



MIPS

The diagram illustrates the translation of C code to different instruction set architectures. It features a central C code block with two lines: 'A=b+c;' and 'D=e+f;'. Three arrows originate from this block: one points to 'MIPS' on the left, one points to 'X86' at the bottom center, and one points to 'ARM' on the right. The arrows are double-lined with open arrowheads.

X86

ARM

# Instruction Set Architecture

■ C code

A=b+c;

D=e+f;

MIPS  
Instruction  
Set

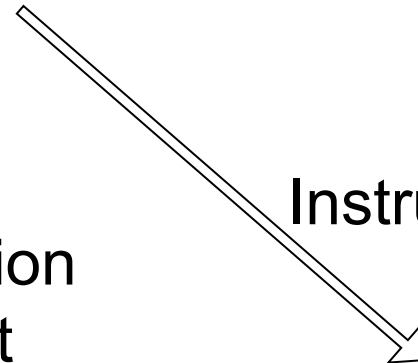
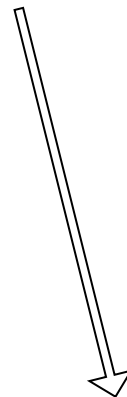
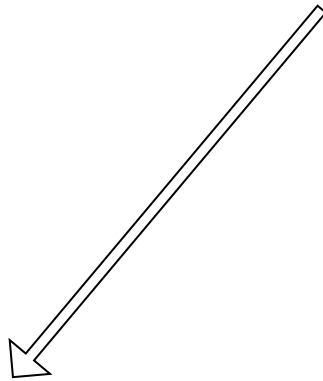
MIPS  
Assembly  
code

X86  
Instruction  
Set

X86  
Assembly  
Code

ARM  
Instruction Set

ARM  
Assembly  
Code



# Instruction Set Architecture

## ■ Assembly Code

Add \$s3, \$s2, \$s1

Add \$s7, \$s5, \$s6

Instruction set is the interface  
between hardware and  
software

## **Instruction Set Design**

- Central part of any system design
- Allows abstraction, independence

# Why?

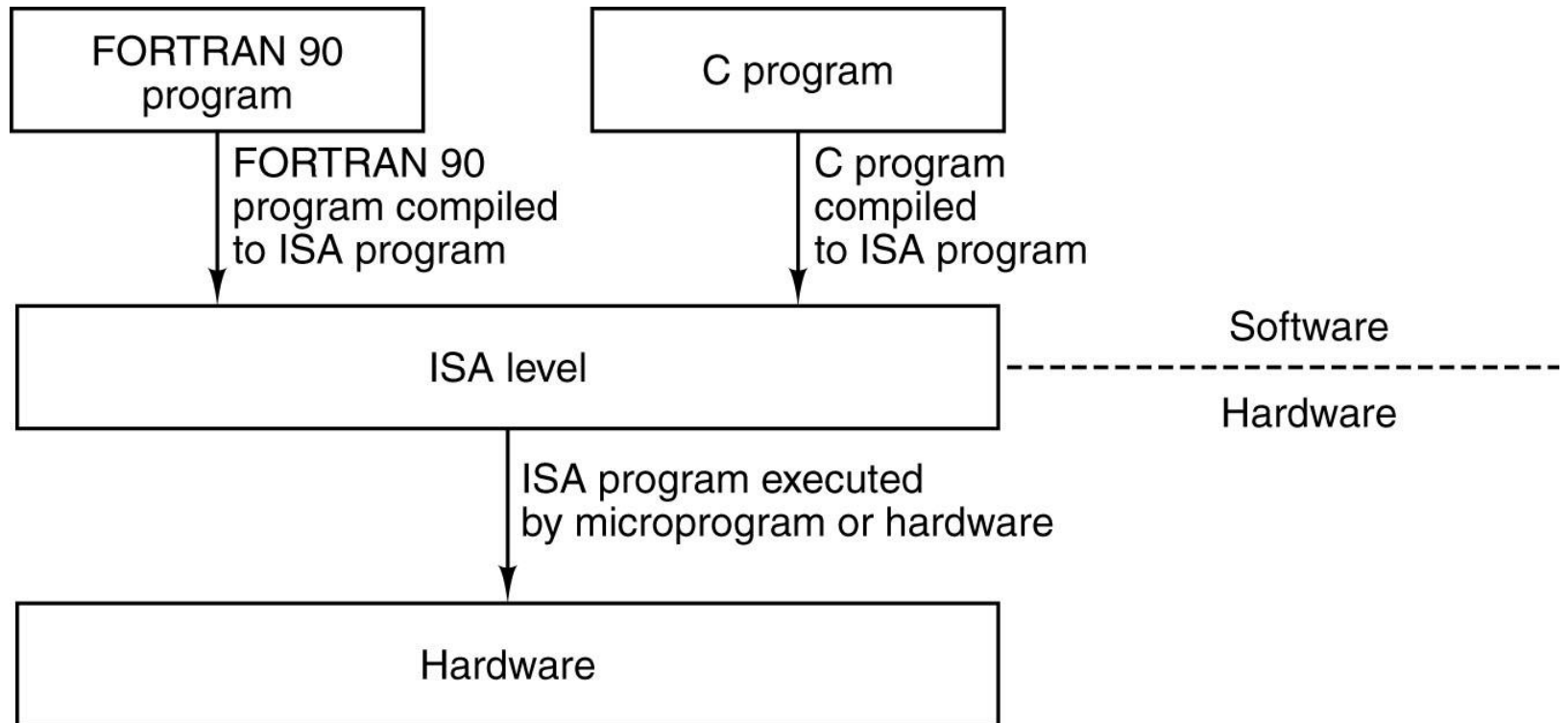
- Early days, new computer having its own new set of instructions
- Needed to allow backward compatibility

# Topics

- Instruction Set Architecture
- Key of ISA using MIPS
  - ✱ Design Principles
  - ✱ Instructions
  - ✱ Instruction formats
  - ✱ Addressing modes



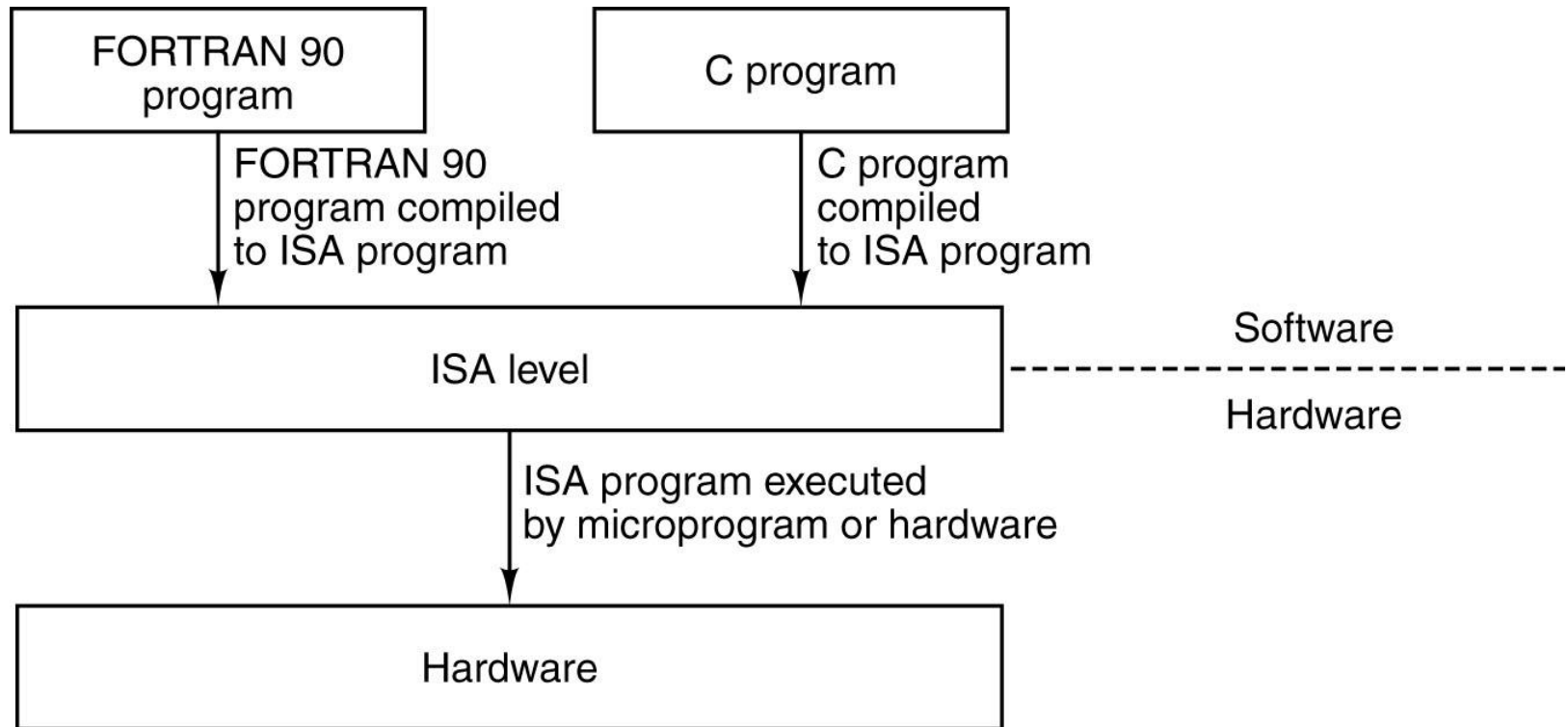
# ISA



# ISA or Instruction Set

- The level - between the high-level languages and the hardware
- When new hardware architecture comes along ...
  - ✱ Can add new features to exploit new hardware capabilities
  - ✱ Need to maintain backward compatibility

# ISA



ISA-level code is what a compiler outputs

# ISA

- ISA-level code is what a compiler outputs
- Compiler writer needs to know
  - ✱ Memory model
  - ✱ Types of registers are available
  - ✱ What instructions are available
    - Instruction formats
    - Opcodes
  - ✱ Exceptional conditions

# ISA

- An ISA includes a specification of the set of opcodes (machine language), the native commands implemented by a particular processor
- Related to programming includes
  - ✱ Native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O

# ISA

- Distinguished from the microarchitecture
  - ✱ MAL which is the set of processor design techniques used to implement the instruction set
- Computers with different microarchitectures can share a common instruction set
- For example:
  - ✱ The [Intel](#) The Intel [Pentium](#) The Intel Pentium and the [AMD](#) The Intel Pentium and the AMD [Athlon](#) The Intel Pentium and the AMD Athlon implement nearly identical versions of the [x86](#) instruction set, but have radically different internal designs

# ISA

## ■ Stored Program Concept

### ✱ Fetch & Execute Cycle

- Instructions are ***fetched*** and put into a special register
- Bits in the register ***control*** the *subsequent actions* (= *execution*)
- Fetch the next instruction and ***repeat***

## ■ Instructions

### ✱ Encoded in binary, called machine code

Opcode	Operand Reference	Operand Reference
--------	-------------------	-------------------

# ISA Instructions

- More primitive than higher level languages,
  - ✱ e.g., no sophisticated control flow such as *while* or *for* loops
- Different computers have different instruction sets
  - ✱ But with many aspects in common
- Computers have very simple instruction sets
  - ✱ Makes the Implementation Simple



# Instruction Set

- The complete collection of instructions that are understood by a CPU
  - ✱ Can be considered as a functional spec for a CPU
    - Implementing the CPU in large part is implementing the machine instruction set
- Machine Code is rarely used by humans
  - ✱ Binary numbers / bits
  - ✱ Usually represented by human readable assembly codes
  - ✱ In general, one assembler instruction equals one machine instruction

# Elements of an Instruction

- Operation code (Op code)
  - ✱ Do this
- Source Operand reference
  - ✱ To this
- Result Operand reference
  - ✱ Put the result here
- Next Instruction Reference
  - ✱ When you have done that, do this...
  - ✱ Next instruction reference often implicit (sequential execution)

# Operands

- Main memory (or virtual memory or cache)
  - ✱ Requires address
- CPU register
- I/O device
  - ✱ Several forms:
    - Specify I/O module and device
    - Specify address in I/O space
    - Memory-mapped I/O just another memory address

# Sample Instruction Format



# Key of ISA