Politechnika Wrocławska

# Computer Architecture and Organization

Lecture 2

Dr. Radosław Michalski
Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology

Version 1.0, spring 2017

## Source and licensing

The most current version of this lecture is here:
https://github.com/rmhere/lecture-comp-arch-org

## Overview of this lecture

## How a CPU works?

### Introduction

**Video**

In One Lesson - from Curiosity to Clarity
How a CPU Works

## Instruction Set Architecture

### Introduction

The computer ISA defines all of the programmer-visible components and operations of the computer. An interface between hardware and software.

- memory organization
  - address space - how many locations can be addressed
  - addressibility - how many bits per location
- register set
  - how many
  - what size
  - how used
- instruction set
  - opcodes (operation selection codes)
  - data types (byte or word)
  - addressing

## Instruction Set Architecture

### Instruction set

- number of different operations
- kinds of operations
- number of operands
- operands location
- operands type
- how to specify operands
- format of instruction
- how many formats

# Instruction Set Architecture

## Operations

- data handling and memory operations
  - set register to a fixed value
  - copy data from a memory location to a register, or vice versa
  - read and write data from hardware devices
- arithmetic and logic operations
  - add, subtract, multiply, or divide
  - increment, decrement
  - bitwise operations (AND, OR, NOT etc.)
  - comparisons
- control flow operations
  - branching
  - calling

## Instruction Set Architecture

### A typical instruction

$$C = A + B$$

$$\text{(dst operand)} = \text{(src operand)}\ \text{(operation)}\ \text{(src operand)}$$

## Instruction Set Architecture

### Operands

- where we store operands
  - registers, memory, stack, accumulator
- how many operands
  - 0, 1, 2, 3
- operand referencing
  - direct, immediate, indirect
- types and sizes of operands
  - byte, int, float, double, string, vector

# Instruction Set Architecture

## Operands location - stack

**Stack**

- the operands are implicitly on top of the stack
- pros: simple expression evaluation (reverse Polish notation), short instructions
- cons: a stack cannot be randomly accessed

**C = A + B**

- PUSH A
- PUSH B
- ADD
- POP C

## Instruction Set Architecture

### Operands location - accumulator

**Accumulator**

- one operand is implicitly the accumulator
- pros: short instructions
- cons: high memory traffic, since accumulator is temporary

**C = A + B**

- LOAD A
- ADD B
- STORE C

## Instruction Set Architecture

### Operands location - GPRs

**General purpose registers**

- all operands are explicitly mentioned, they are either registers or memory locations
- pros: easy code generation, long-term storage in registers
- cons: longer instructions

**C = A + B**

- LOAD R1, A
- ADD R1, B
- STORE R1, C

## Instruction Set Architecture

### GPRs - classification

How many operands can be memory-addressed?

- **register - register**: ADD R3, R1, R2
- **register - memory**: ADD R1, A
- **memory - memory**: ADD C, A, B

## Instruction Set Architecture

### Operand referencing

Register (direct):

- operands in registers
- add R4, R3

Immediate:

- operand provided directly
- add R4, 5

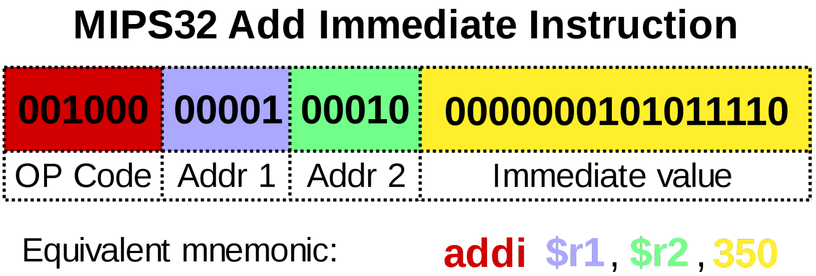
Base (displacement):

- address of operand is a sum of immediate and value in register
- add R4, 100(R1)

There are also other modes. Keep in mind that not all ISAs implement all of these.

## Instruction Set Architecture

### Instruction format



MIPS32 Add Immediate Instruction

| 001000 | 00001 | 00010 | 0000000101011110 |
|---|---|---|---|
| OP Code | Addr 1 | Addr 2 | Immediate value |

Equivalent mnemonic: addi $r1 , $r2 , 350

*German - Mips32 addi*, CC BY-SA 3.0

## Instruction Set Architecture

### Instructions - considerations

- fixed length
  - more instructions needed
  - faster, no decoding
  - Alpha, ARM, MIPS, PowerPC
- varying length
  - more flexibility
  - slower due to decoding
  - VAX, Intel 80x86
- hybrid model
  - IBM 360/70, MIPS16

## Instruction Set Architecture

### Sources

- M. Martin, *“Introduction to Computer Architecture”*, University of Pennsylvania, PA, USA (course materials)
- H. Jiang, *“Computer Architecture”*, University of Nebraska-Lincoln, NE, USA (course materials)
- E. MacDonald, *“Microprocessors”*, University of Texas at El Paso, TX, USA (course materials)

## Instruction Set Architecture

### Introduction

**Video**

In One Lesson - from Curiosity to Clarity

This is What's Inside a CPU

## Architectural considerations

### Introduction

- the CPU's front-end is its ISA
- the details on how a particular command is executed do not need to be public
- multiple implementations of a single ISA
- competitive edge gained on how the instructions are executed
  - speed
  - power consumption
  - reliability

## Architectural considerations

### Optimizing speed

Where the competition is?

- multiple implementations of a single ISA
- competitive edge gained on how the instructions are executed

Types of processors:

- general purpose processors
- specialized processors, coprocessors:
    - GPU - graphics processing unit
    - TCP offload engines
    - cryptographic accelerators

## Architectural considerations

### Optimizing power consumption

How to optimize power consumption?

- voltage reduction
- frequency reduction
- capacitance reduction
- clock gating
- reducing switchning activities

## Architectural considerations

### From watts to microwatts

How much power this CPU consumes?

- Intel® Core™ i7-6800K - 140W
- Intel® Core™ i7-4600M - 37W
- ARM® Cortex® A9 - 1.9W

How much power your CPU consumes?

- check the specifications of your PC, laptop, mobile phone, or tablet processor.

## Architectural considerations

### CPU complexity

Looking for trade-off:

- more complex/sophisticated instructions
- simpler instructions (e.g., fixed length)
- more clock cycles vs. less clock cycles

Yet, does complexity have some lower or upper bounds?

## Architectural considerations

### Introducing RISC architecture

Acronyms:

- CISC - complex instruction set computing
- RISC - reduced instruction set computing

Until RISC came, most typical approach was CISC:

- complex instructions
- more cpu clock ticks needed to perform an operation

## Architectural considerations

### Sources

- IBM Icons of Progress, *"RISC Architecture"*, IBM (article)
- J.C. Scott, But How Do It Know? - The Basic Principles of Computers for Everyone, 2009 (book and related videos)

## Final slide

### Questions? Comments?

If you have any ideas on how to improve these lectures,
please submit them as issues in this repository:

https://github.com/rmhere/lecture-comp-arch-org